TURING

5

2013/04/18

码农

codeMaker()

App 创富传奇

愤怒的小鸟：第52次运气

苹果应用开发与营销

App Store提交申请全攻略

移动应用UI设计反模式一瞥

当Steve遇上Steve

目录

编者的话

专题：App 创富传奇

1 愤怒的小鸟：第 52 次运气

10 Dave Wooldridge 谈苹果应用开发与营销

16 当苹果的设备越来越多，如何正确检测不同设备的功能?

26 iOS 应用如何初始化基本数据模型?

34 App Store 提交申请全攻略

47 移动应用 UI 设计反模式一瞥

人物

55 阿怪：二进制？十二进制?

践行

66 Objective-C 和 C++ 的区别有哪些？为什么苹果会选择前者?

69 Objective-C 在其他平台也有应用吗?

八++

76 当 Steve 遇上 Steve：曾经打造苹果的少年们

出版的未来

88 图书平台化：《东京艺术空间》电子出版实践

鲜阅

101 Trip Hawkins：离开苹果之后，我创立了 EA

书榜

111 大家都在读什么？

妙评

114 好书妙评之《具体数学》

编者的话

App 创富传奇

编者 / 李盼

这期的专题关于 App(le)，关于创富，也关于传奇。码农们面朝键盘背朝天的编码，偶尔也该摘掉头顶的草帽，乘乘凉，听听坊间的传奇。

“听说同样是编码，有些人赚了百万的米刀。”“说实话，那些东西挺简单，一看就知道是怎么实现的。”是这样吗？也许是。收获瞬间成功的传奇人物也是普通人，只不过多一点运气罢了。可是如果需要“52 次运气”的话，可能需要的压根就不是运气。

各路淘金者们蜂拥到 App Store 淘金。“不就是挖吗？”这种方式码农很擅长，反正都是编码。无论挖到挖不到，都祝你们享受阳光，编码快乐。本期《码农》也精心搜罗了一些“开挖”方面的 Tip，希望有些是你闻所未闻的。

可是在这热闹的景象下，有人也提出了异议，他是一位跨界码农，界的另一面是码农们只能在屏幕上看到的“演艺圈”。他说走红毯不是重点，但是要定期参加码农聚会，他觉得创业“很没意思”，他还认为编码和写歌是一回事儿。是啊，那些曾经开心的编码少年们，你们可以改变世界，甚至创造出苹果这个大庄家，但是扪心自问，两个 Steve（Woz & Jobs），你更愿意做哪个呢？

有人说《码农》越做越像饭后甜点。So，enjoy your desserts!

专题：App 创富传奇

愤怒的小鸟：第 52 次运气

作者 /Chris Stevens

设计师，记者。设计过众多优秀的应用程序，其中最著名的当属 iPad 版《爱丽丝漫游仙境》。他曾是《每日电讯报》的技术专栏作家，后为《泰晤士报》撰稿。他还是一名插图画家、编剧和电影剪辑师。曾供职于 BBC、CNET、EMAP、 华纳兄弟和《连线》（Wired）杂志，并以记者的身份获得过《卫报》传媒奖。

译者 / 曾文斌

罗维奥公司成立于 2004 年，最初叫做瑞鲁德（Relude），创立者是芬兰的一对堂兄弟——尼克拉斯·赫德（Niklas Hed）和米凯尔·赫德。一开始，米凯尔的父亲、企业家卡吉·赫德（Kaj Hed）向公司资助了 100 万的资金。如果没有这笔钱，他们可能还没成功就花光了所有的现金。在外界看来，《愤怒的小鸟》是一夜成名，但实际上，罗维奥公司经历了财务困境、伤心失望、52 个游戏的失败和机遇面前的巨大阻力。如今，《愤怒的小鸟》坐在 App 商店之巅，成为了 iPhone 上标志性的成功典范。然而，在此之前，这个家族公司陷入过困境，甚至差点倒闭，最后才勉强逃脱了破产的命运。

兄弟同心，其利断金

和《涂鸦跳跃》的兄弟俩一样，米凯尔和尼克拉斯从小就对制作电子游戏有着浓厚的兴趣。小时候，他们就一起讨论游戏。尼克拉斯对物理游戏特别感兴趣，他会编写代码向朋友炫耀。对编程的着迷，让他后来在赫尔辛基大学拿到了计算机科学学位。他的堂兄米凯尔则在芬兰军队服役了一年，然后上了商学院，他先是在欧洲学习，后来去了美国。

毕业后，尼克拉斯试图说服米凯尔和他一起创建一个电子游戏公

来源：Chillingo 授权使用，©2011 年 Rovio Mobile

米凯尔·赫德，风靡全球的游戏《愤怒的小鸟》的创作者之一

司，专门销售手机游戏。那时，iPhone 还没有问世，每一份游戏合同的签订都必须经过与手机运营商小心翼翼、苦口婆心的谈判。这是很麻烦的事，米凯尔一开始很不愿意参与。

但是，尼克拉斯很会劝说。他赢得了为一款早期智能手机制作游戏的机会，受这一胜利的鼓舞，他最终成功说服了米凯尔：基于这次胜利，他们可以在移动游戏市场大展身手。

尼克拉斯还得到了彼得·韦斯特贝卡的支持，后者是惠普公司颇有影响力的通讯行业专家。

贵人韦斯特贝卡

在 2003 年的智能手机游戏竞赛中，韦斯特贝卡认识了尼克拉斯，尼克拉斯赢得了那次比赛。比赛由惠普公司赞助，韦斯特贝卡则是评委之一。他对尼克拉斯印象深刻，并且告诉他一定要“创立自己的移动软件公司”。当时，韦斯特贝卡还不知道，七年后，在罗维奥大获成功之时，他会辞去自己的工作，加入这家公司，负责市场营销工作。

米凯尔租了办公场所，瑞鲁德公司（罗维奥的原名）开始承包其他游戏开发者的项目。然而，米凯尔与他的父亲卡吉（这家新生公司的实际投资人）发生了矛盾，导致公司的运营情况迅速变得糟糕起来。到 2005 年中期，公司的情况不断恶化。米凯尔离开了公司，开始从事独立漫画书的出版。

没有了米凯尔对公司发展方向的掌控，罗维奥陷入了更深的危机。尽管公司的客户名单上继续添加着令人印象深刻的名字，如电子艺界和南宫梦，但公司内部普遍感觉，创作一个热门游戏的目标似乎无法实现，而这本应该是公司的工作重心。随着开发成本的增加，罗维奥公司越来越偏离正轨，开始遭受巨额的亏损。

烧钱，烧钱

2009 年初，罗维奥公司的财务状况已经火烧眉毛了，这正应了公司的名字，“罗维奥”在芬兰语里的意思就是“篝火”。在最为惨淡的日子里，罗维奥公司被迫裁员，将 50 名员工减少到了 12 名。就在那时，大家对现实开始有了清楚的认识：罗维奥公司原本是要成为一家热门游戏公司，但问题是，他们连一个热门的游戏都没有。

就在公司要倒闭的时候，尼克拉斯突然得到了启发，他说服卡吉重新请回了米凯尔，让米凯尔掌握公司的大方向，并把公司的重心从承接工程上转移了出来。是什么事情启发了尼克拉斯，让他作出这一重要决策呢？答案就是 iPhone 的问世和 App 商店的推出。

尼克拉斯意识到，iPhone 解决了罗维奥公司的很多难题。公司不必再依赖大游戏公司的营销团队，也不必再与运营商苦苦谈判。App 商店意味着有了全世界范围的分销渠道，意味着可以直接接触几百万名客户。苹果公司可以成为他们的发行部门，而他们只需集中精力，找出那颗珍贵而难得的宝石：一个轰动的应用程序。

为其他游戏公司做的开发工作使罗维奥暂时腾不出时间。2009 年春天，罗维奥的首席游戏设计师杰科 · 伊萨洛（Jaakko Iisalo）打开 Photoshop，开始设计游戏的图形。伊萨洛经常向团队展示他的创意，这正是他许多创意中的一个：他画了一些彩色的鸟，每只鸟的颜色跟色块相同。轻拍色块，就会导致同样颜色的鸟被弹出去撞碎。屏幕效果看上去有点像后来的《愤怒的小鸟》，伊萨洛觉得这会很有戏。

伊萨洛向尼克拉斯和米凯尔展示了屏幕效果，他们立刻被小鸟愤怒的表情和不会飞的荒诞吸引了。团队决定将猪作为小鸟的敌人添加进去，并把猪设计成绿色，暗示猪流感传染病。后来，他们

来源：罗维奥授权使用，©2011 年 Rovio Mobile

《愤怒的小鸟》的角色形象由杰科·伊萨洛早期的素描发展而来

来源：Chillingo 授权使用，©2011 年 Rovio Mobile

很多玩家发现，《愤怒的小鸟》特别容易让人上瘾

又创造了一个背景故事，用猪偷鸟蛋来解释为什么小鸟要发动残酷的攻击。

设计师很快发现，每次对代码进行微小的修改后，程序员在测试时都会陷进游戏里玩上几局，而不是立刻去做手头的工作。

“对于我们之前的游戏，大多数朋友和家人通常都是匆匆看一眼，然后给一些泛泛的好评，”米凯尔后来说道，“但是对于《愤怒的小鸟》，大家的反应几乎都是一样的：他们抓起 iPhone，找到一个安静的角落，便开始玩个不停，直到他们手里的手机被抢走。”这真的是特别令人上瘾的玩意儿。

游戏概念由来已久

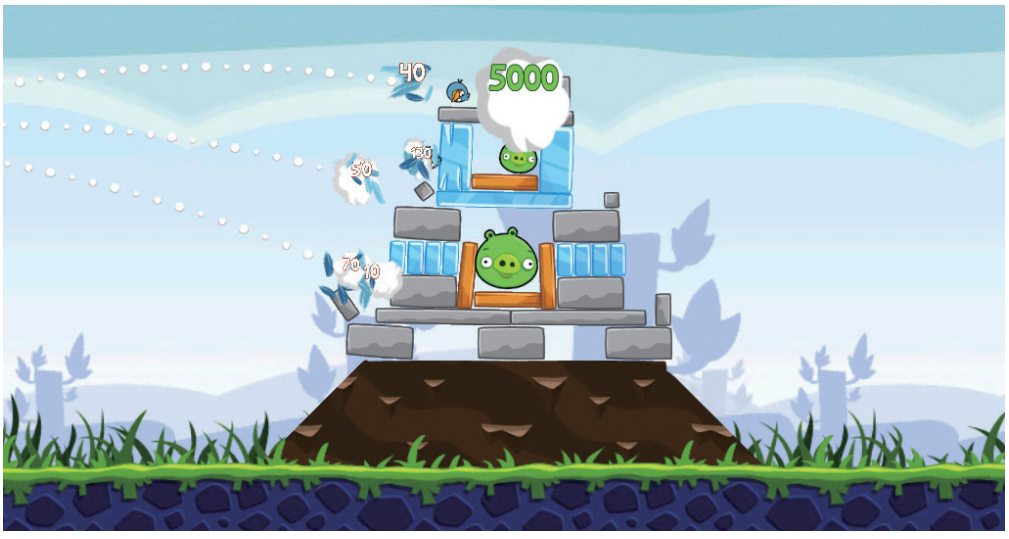
《愤怒的小鸟》并非完全原创。可以说，几乎没有哪个伟大的发明是完全原创的。但是，在这个游戏中，罗维奥公司将各种元素组合得既新奇又出色。《愤怒的小鸟》脱胎于一些把物体用力投向可摧毁的建筑的物理游戏，如《城堡粉碎战》（Crush the Castle）和它的前身《城堡云》（Castle Cloud）。其他不少投掷动物的游戏也都包含了《愤怒的小鸟》的物理机制，但它们缺乏对游戏操控、角色个性和图形的巧妙结合。《飞天刺猬》（Hedgehog Launch）和《丢乌龟》（Toss the Turtle）就是早期的两个投掷动物游戏。然而，尽管这些早期的游戏对《愤怒的小鸟》有很大的影响，有些罗维奥的批评者甚至认为这些游戏都包含了《愤怒的小鸟》的一些元素，但是，只有《愤怒的小鸟》把这些不同的游戏元素统一完美地组合在了一起。罗维奥公司的技巧在于，它能识别好的游戏元素并对其加以改造。后来，米凯尔说《愤怒的小鸟》“真正把所有元素糅合在了一起”。

在发布游戏之前，罗维奥对游戏令人上瘾的特质十分兴奋，公司的员工也都认为自己拥有了一个足以引起轰动的产品。尼克拉斯和米凯尔确信，这个游戏将使罗维奥公司打一场翻身仗，结束公司多年的混乱和动荡。他们决定投资 10 万欧元开发这个游戏，并于 2009 年年底在 App 商店推出。那年的 12 月，罗维奥公司全体员工都屏住了呼吸，等待游戏发布后的反响。

然而，《愤怒的小鸟》竟然栽了。

从反响不佳到统治世界

美国和英国这两个最大的 App 商店市场只是瞟了一眼这个新游戏，然后耸了耸肩，完全没把《愤怒的小鸟》当回事。对这种市场反应感到震惊之余，罗维奥公司注意到了一个有趣的现象：在小一些的市场，更容易赢得竞争。于是，公司决定把重点先放在像芬兰这样的小市场。这样，只需每天几百份的销量，《愤怒的小鸟》就可以登上芬兰 App 商店榜首。相比之下，在美国至少需要一天几千份的销量。罗维奥重点进攻的几个欧洲国家有了不错的市场反应，《愤怒的小鸟》一共卖出了 3 万多份。罗维奥公司认为，在这些市场的成功有助于说服苹果公司选荐公司的游戏。但是，罗维奥不打算自己去找苹果公司，而是决定和当时的独立游戏出版



来源：Chillingo 授权使用，©2011 年 Rovio Mobile

罗维奥为游戏添加了关卡并制作了一个精简版

公司 Chillingo 合作。

Chillingo 是一家英国游戏开发公司，产品质量高，有着良好的声誉，而且与苹果公司的关系也不错。这对罗维奥而言，有着不可估量的利用价值，因为《愤怒的小鸟》要想成功，还需要最后一环：App 商店的选荐。

在罗维奥和 Chillingo 的努力下，《愤怒的小鸟》得到了苹果公司的注意。他们达成一致，让这款游戏在 2010 年 2 月的第二个星期成为英国 App 商店上的本周最佳游戏。尼克拉斯和米凯尔把这当做将《愤怒的小鸟》推上榜首的唯一机会。于是，他们开始了一系列的行动，给游戏增加了 40 多个新关卡，做了一个精简（免费）版，还设计了一个宣传片放在 YouTube 上。他们设计这次具有战略意义的活动，只为在得到苹果公司的推荐后，给玩家留下深刻的印象。他们做到了。

苹果公司 App 商店的应用程序下载量已经超过 100 亿。如果能够把新颖的创意、强大的功能、独特的设计和出色的市场营销巧妙地结合在一起，你的应用程序也有机会一鸣惊人。《App 创富传奇》分享了 App 商店里激动人心的故事，告诉你如何将应用的创意变成滚滚财源。通过大量真实的传奇故事和面对面的访谈，作者带领你深入神奇的 App 开发世界，让你与那些 App 富翁和名人们近距离接触。你将一窥这些梦想家成功背后的故事，获悉他们第一手的秘密，了解这些“卧室程序员”是怎样一夜之间成为百万富翁的。

在获得苹果公司选荐的第一个星期，《愤怒的小鸟》一下就从默默无闻升到了榜首。尼克拉斯和米凯尔惊呆了。然而，这仅仅是个开始。同年 4 月，《愤怒的小鸟》在美国市场也夺得了第一名。《愤怒的小鸟》现象已经开始了，而且势不可挡。这些诞生在芬兰罗维奥公司电脑屏幕上的彩色小鸟，正以它们那不会飞的身躯，在世界各地（从东京到开罗）的每一台 iPhone 屏幕上弹来弹去。简单而灵巧的设计为这款游戏赢得了几百万用户。

米凯尔告诉 GamePro：“好的游戏必须易学难精，这是古老的智慧。‘易学’特别重要，当你一看到屏幕，就应该知道要做什么。”

专题：App 创富传奇

Dave Wooldridge 谈
苹果应用开发与营销

Dave Wooldridge

“他不是在开发 Mac 和 iOS 应用，就是在写作”

Dave Wooldridge，Electric Butterfly 创始人，多年从事用户界面设计和 iOS 应用开发工作。著有《苹果应用开发与营销（第 2 版）》、“The Developer Sketchbook" 系列，并合著有 Beginning

图灵社区：可否简要介绍一下你投身应用开发工作的过程？

Dave：1995 年，我创办 Electric Butterfly 公司时，最早从事的是网页开发工作。在 1997 年，我开始为 Mac 开发应用软件，其中既有自有应用，也有为客户开发的项目。移动技术一直都很吸引我，因此当近几年来手机和平板电脑应用的市场出现井喷时，开发移动应用也就成为公司业务的自然延伸。在过去几年中，iOS 应用开发已经很快成为公司的核心业务。

图灵社区：能否从你的众多成功应用当中挑选一例，向中国读者介绍一下其成功的关键？

Dave：Qello 是我们开发的一个应用，已经在 App Store 上线，可供全球的用户下载。它适用于 iPhone、iPad 和 iPod touch，可以让用户在移动 iOS 设备上舒适地观看音乐会和音乐纪录片。

苹果公司青睐那些具有漂亮用户界面和抓人用户体验的应用，这

iPad Development for iPhone Developers (Apress)。

记者 / 楼伟珊

两点也是我们在设计和开发过程中优先考虑的重点。2012 年 5 月，App Store 将 Qello 选入“新品推荐”，大大增加了应用的曝光率，使它在九十多个国家的下载量排名中名列前茅。

成功应用的另一个关键因素是专注于把一件事做好，在这里就是，在有趣、易用的移动环境中把高清品质的音乐会呈现给用户。许多开发者试图把太多功能塞进一个应用中，误以为这样可以提升应用的价值。但事实上，如果你去看看 App Store 里的那些成功应用，它们大多数是把一个功能做到极致。这时，少即是多。

还有一个重要因素是使应用社会化。我们在 Qello 中深度整合了 Facebook 和 Twitter，并将在后续升级中增加更多社会化功能。通过允许用户在朋友中构建与 Qello 相关的圈子，分享各自喜欢的音乐会，不仅可以方便粉丝们相互交流，发现感兴趣的新音乐，还可以提升 Qello 的知名度，增加它在 App Store 上的下载量。

图灵社区：作为 Electric Butterfly 的创始人，可否介绍一下公司在应用业务方面的发展重点，以及在积累用户和口碑方面的最佳实践？

Dave：我一直算不上是个游戏玩家，因此我也对开发游戏没什么兴趣。应用开发和游戏开发对开发者有不同的技能要求。我喜欢的是开发生产力应用，那些帮助人们进行 GTD 的应用。因此，应用开发一直以来是公司的关注重点。开发一个应用意味着对客户作出了长期承诺，需要持续提供更新和支持，所以这个项目最好是你真正感兴趣，并愿意数年如一日为之工作的。

至于积累用户和口碑，坚实的营销方案的确很重要，但无论多少营销都无法使一个糟糕的产品变得成功。因此，作为开发者，你

的首要职责是针对特定需求和用户创造出尽可能最佳的应用，一个精致、吸引人的高品质产品。其次，要通过电子邮件、Twitter、Facebook 等与用户建立起直接联系，并提供专业的客户支持。这可以为你赢得忠诚的客户，他们会把你的应用推荐给其他人。

图灵社区：App Store 为个人开发者提供了很好的机遇。在你看来，要成为优秀的应用开发人员，需要具备哪些素质呢?

Dave：要想在 App Store 取得成功，要求具备多方面的技能，比如在界面设计、程序开发、市场营销、公共关系、竞争研究以及商业运营等方面的专长。在理想状况下，你可以网罗一个专家团队来提供项目所需的各种技能，你也可以通过学习，做得自力更生不求人。但不论如何，作为独立开发者，你不能只想着编程。不妨把你自己想象成一个企业家，要为自己的事业负全部责任。

我在学校学的是平面设计和商业，毕业获得的学位是市场营销方面的。因此，我是个百分百自学成才的程序员和界面设计师。在过去二十多年中，我阅读了数以百计的开发类图书，自学了数十种程序语言和工具，并经常在互联网上检阅教程和技术文档。我从未有一刻停止过学习，通过持续不断的自我教育，去努力适应这个变化万千的软件开发世界。我想，这也是每位开发者提升自身价值的关键所在。

不要停止自我教育，也不要只想着改进编程技能。要留意所有与你所选开发平台相关的事情，比如新的 iOS 商业模式、第三方服务或市场机会等。要关注你的竞争对手表现如何，分析和借鉴其他应用开发者的经验和教训。还可以通过社会化媒体和技术会议

与同行分享你的开发经验。

图灵社区：作为应用开发者，技术、设计和推广等方面的能力应该都很重要。在产品的不同阶段，你是如何排列这些能力的优先级并投入时间的?

Dave：作为独立开发者，至关重要的是有效的时间管理和项目管理。我从 1995 年就开始自己出来做事，对于如何最大可能地利用每一个工作日也算有点心得。虽然这要求巨大的投入和专注，但一旦你找到适合自己的常规，你就会变得习以为常。

为客户开发的项目总是有严格的时间限制，在截止日期前所有要求的元素都必须就位。因此，不妨把自有项目也视作为客户开发的项目，给它安排一个截止日期。通过为每个项目制定进度表，并为不同的开发阶段设置优先级和里程碑，我得以让自己的日常任务保持可控并符合进度。

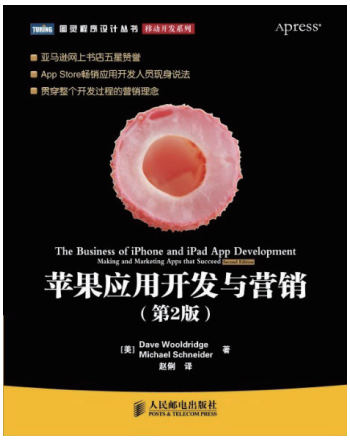
图灵社区：我们看到，你的《苹果应用开发与营销》一书涉及应用开发的设计、技术、营销、法律等很多方面。请问你是如何掌握这么多不同领域的知识，并能总结成书的?

Dave：正如我前面提到的，我从未有一刻停止过学习所有与 iOS 应用开发、设计和运营相关的东西。当然对于这本书，我并没有完全依赖我个人的知识和经验。事实上，我花了数月时间研究市场，采访其他开发者。我一直很喜欢写作，因此当 Apress 联系到我，让我写这本书时，我很珍惜这个能把个人思考和经验与研究和案例分析结合起来的机会。

图灵社区：这个问题是来自我们的读者，他们希望你能回答一个挑战性的问题：假如产品上线后，迅速获得大量差评，你会如何处理？

Dave：首先，你要竭尽全力避免这种情况出现，比如可以给用户提供一个直接反馈的渠道。要做到这一点，简单在应用中增加一个反馈框，用以接收所有客户支持请求即可。如果你接收到大量相同的问题，你可能需要在应用的 FAQ/Help 中添加内容，解答这些常见的问题。另外，你可以在 App Store 页面的“内容提要”末尾请求用户不要把 bug 报告和功能请求写到“用户评价”中，同时说明通过电子邮件和应用内的反馈框，用户可以直接联系到开发者并得到及时的回应。

当然，要想避免所有的负面评价是不可能的，即便那些最成功的应用也难免得到一些差评。因此，我们的目标应该是减少那些可以避免的负面评价，比如报告 bug 或抱怨使用问题等。为此，你需要告知用户，你正在处理这些问题，并会在后续的升级中修正它们。简短发布一个计划就可以达到这个目的，途径可以是在 App Store 页面的“内容提要”当中，也可以是在你的网站或所有社会化媒体账户上，比如 Twitter 和 Facebook。如果你的应用支持推送通知或消息之类的，你也可以把这些信息在用户下次打开应用时直接告知他们。倘若用户知道你已经开始着手处理他们遇到的问题，他们公开抱怨的可能性也会降低一些。最后，有一点是不言而喻的，那就是你永远要以专业、及时的方式回应所有的客户支持请求。对待每一位消费者要像对待宝贵的投资人一样，让他们自感位尊言重。



如何在这个拥挤不堪的市场中让自己的产品引起消费者注意，已经成了开发者不得不面对的问题。《苹果应用开发与营销》的目的就是要给 iOS 开发人员指出一条康庄大道，让 iOS 应用在消费者中间建立知名度并保持长销畅销。作者将整个流程分解为“规划”、“开发”和“发布”三大步，逐阶段讲解怎样把营销理念纳入其中每个环节。

图灵社区：读者学习完你的书之后，还可以从哪些地方持续学习应用的开发和营销？可否建议一些资源？

Dave：这本书的附录提供了大量应用开发和营销的在线资源。对于学习 iOS 开发，我总是会推荐非常有用的 StackOverflow 网站，以及苹果的 iOS 开发中心上面所有精彩的技术文档和示例代码。许多顶尖的 iOS 开发者会不时发表一些博客文章和开发教程，因此，在 Twitter、Google+ 和 Facebook 找到并关注他们，留意其中的有用链接和他们分享的经验之谈。

最后，我要感谢这次采访，并且要特别感谢那些买了我的书的人。我希望这本书会对你们有所帮助，也衷心祝愿你们在今后的应用开发历程中一切顺利。

查看图灵访谈英文原文

当苹果的设备越来越多

如何正确检测不同设备的功能?

作者 /Rob Napier

Rob Napier 喜欢徒步旅行，还是一位自豪的父亲。2005 年开始从事 Mac 开发，iPhone SDK 第一版发布时就开始开发 iPhone 应用了。他是 The Daily、PandoraBoy 和 Cisco Mobile 的作者，Stack Overflow 的

过去，开发者最常犯的错误就是在只有两个设备时（iPod Touch 和 iPhone），通过检测设备型号名称以及检查它是否是 iPhone 来判断设备的功能。这种方法奏效了一年左右，很快，当带有新硬件传感器的新设备出现时，这种方法就变得很容易出错。比如，iPod touch 的第一个版本并没有麦克风，但到了 iPhone OS 2.2 版时，用户可以通过外接麦克风 / 耳机来添加一个。如果你的代码根据型号名称来判断设备的功能，那它仍然能工作，但并不正确，也不是应该采取的处理方式。

检测设备及判断功能

看看如下的代码片段，它用来判断 iPhone 的功能。

检测麦克风的错误方法

```
if(![[UIDevice currentDevice].model
isEqualToString:@"iPhone"])  {
          UIAlertView *alertView = [[UIAlertView alloc]
initWithTitle:@"Error"
message:@"Microphone not present"
```

主要贡献者，维护着博客 Cocoaphony。

译者 / 武海峰

```
delegate：self
       cancelButtonTitle：@"Dismiss"
otherButtonTitles：  nil];
        [alertView show];
    }
```

这段代码的问题在于，这位开发者大胆地假设只有 iPhone 有麦克风。这段代码最开始表现良好。但在 iOS 2.2 版中，苹果向 iPod touch 增加了外接麦克风的功能，这段代码就会阻止用户使用该应用。另一个问题是这段代码在检查苹果后来推出的设备（如 iPad）时，会出现一个错误。

你应该用其他方法来检测硬件或传感器的可用性，而不是假设设备具有某些功能。这些方法分散在各种框架中，不知道这是幸运还是不幸。

现在来了解正确查看设备功能的各种方法，并将它们放在为 UIDevice 类编写的类别扩展中。

检测硬件和传感器

首先要知道，你需要查看所需的硬件或传感器是否存在，而不是假设设备有哪些功能。举个例子，你不能假设只有 iPhone 才有麦克风，而应该使用 API 来查看麦克风是否存在。下面这段代码的第一个优势在于，它能自动兼容将来推出的新设备和外接麦克风。

第二个优势呢？这段代码只有一行。

检查麦克风可用性的正确方法

使用 AudioSessionPropertyListeners 与观察 NSNotification 事件很像。当你向一个类中增加一个属性监听器时，需要负责在适当的时候移除它。在前面这个例子中，由于在 viewDidLoad 中增加了属性监听器，所以需要在 didReceiveMemoryWarning 方法中移除它。

```
- (BOOL) microphoneAvailable  {
AVAudioSession *session = [AVAudioSession sharedInstance];
return session.inputIsAvailable;
}
```

对于麦克风，你还需要检测输入设备变化的提醒。也就是当用户插入麦克风时，除了在 viewDidAppear 中做相应的修改外，还要激活 UI 上的 Record 按钮。听起来挺酷，不是吗？下面就是具体的实现方法。

检查是否插入麦克风

```
void audioInputPropertyListener(void* inClientData,
AudioSessionPropertyID inID, UInt32 inDataSize, const
void *inData)  {
    UInt32 isAvailable = *(UInt32*)inData;
    BOOL micAvailable = (isAvailable > 0);
    // 加入更新 UI 的代码
}
- (void)viewDidLoad  {
    [super viewDidLoad];
AudioSessionAddPropertyListener(
kAudioSessionProperty_AudioInputAvailable,
audioInputPropertyListener, nil);
}
```

这里，你要做的就是为 kAudioSessionProperty_AudioInputAvai-lable 增加一个属性监听器，然后在回调中检查它的值。

只要增加很少几行代码，就能够写出正确的设备检测代码了。下一步，你需要扩展这段代码，从而支持其他的硬件和传感器。

1. 检测摄像头类型

iPhone 最初只有一个摄像头，后来在 iPhone 4 中增加了一个前置摄像头。iPod touch 直到第四代才有摄像头。iPhone 4 有前置摄像头，iPad 1（比 iPhone 4 大）却没有摄像头，而后来的 iPad 2 同时有了前置和后置摄像头。所有这些都意味着你不应该在假设设备功能的前提下编写代码。实际上使用 API 更方便。

UIImagePickerController 类含有检测源类型可用性的类方法。

检测是否存在摄像头

```
- (BOOL) cameraAvailable  {
    return  [UIImagePickerController
isSourceTypeAvailable:
UIImagePickerControllerSourceTypeCamera];
}
```

检测是否存在前置摄像头

```
- (BOOL) frontCameraAvailable
{
#ifdef __IPHONE_4_0
    return  [UIImagePickerController
isCameraDeviceAvailable:
UIImagePickerControllerCameraDeviceFront];
#else
  return NO;
#endif
}
```

检测前置摄像头，需要运行在 iOS 4 或更高版本中。枚举类型 UIIm

-agePickerControllerCameraDeviceFront 只在 iOS 4 及更高版本中可用，因为所有带有前置摄像头的设备（iPhone 4 和 iPad 2）使用的都是 iOS 4 及更高版本。所以你用到了宏，如果设备使用的是 iOS 3 或更低版本就返回 NO。

类似地，可以检查设备上的摄像头是否具备视频录制功能。iPhone 3GS 及更新设备的摄像头支持录制视频。你可以使用以下代码来检查。

检测摄像头是否支持视频录制

```
- (BOOL) videoCameraAvailable  {
  UIImagePickerController *picker =
[[UIImagePickerController alloc] init];
// 首次调用前面的方法，检查是否存在摄像头
if(![self cameraAvailable])  return NO;
NSArray *sourceTypes = [UIImagePickerController availab
leMediaTypesForSourceType：
UIImagePickerControllerSourceTypeCamera];

   if (![sourceTypes containsObject：(NSString *)
kUTTypeMovie]){
    return NO;
  }
  return YES;
}
```

这段代码会枚举给定摄像头的可用媒体类型，然后判断它是否包含 kUTTypeMovie。

2. 检测照片库是否为空

如果你在使用摄像头，几乎总会用到照片库。在调用UIImagePicker显示用户相册前，需要确保它里面有照片。可以用检查摄像头是否存在的方法来查看相册是否为空，只要将UIImagePickerControllerSourceTypePhotoLibrary或UIImagePickerControllerSourceTypeSavedPhotosAlbum作为源类型传过去就行了。

3. 检测摄像头闪光灯是否存在

到目前为止，带有摄像头闪光灯的唯一设备是iPhone 4〔1〕。在未来几年里，将会有越来越多的设备带有摄像头闪光灯。使用UIImagePickerController的类方法来检查摄像头闪光灯是否存在很容易。

〔1〕翻译本书时，iPhone 4S和iPhone 5均已发布，它们也带有摄像头闪光灯。

——译者注

检测摄像头闪光灯是否存在

```
- (BOOL) cameraFlashAvailable  {
#ifdef __IPHONE_4_0
  return [UIImagePickerController isFlashAvailableForCa
meraDevice：
UIImagePickerControllerCameraDeviceRear];
#else
  return NO;
#endif
}
```

如果陀螺仪是你的应用中一个重要功能，而你的目标设备上没有陀螺仪，那么必须用其他输入方法来设计应用。或者也可以在应用的info.plist中的UIRequiredDeviceCapablities键中指定它们，防止没有陀螺仪的设备安装该应用。本章稍后会进一步介绍这个键。

4. 检测陀螺仪是否存在

陀螺仪是iPhone 4上新增的一个有意思的传感器。iPhone 4之后发布的设备，包括new iPad和iPhone 5，都有陀螺仪。陀螺仪用于测量设备物理位置的相对变化。相比之下，加速计只能测量力的大小，而不能测量扭动。有了陀螺仪，游戏开发者甚至可能实现六轴控制，类似于索尼PlayStation 3控制器或任天堂的Wii控制器提供的功能。你可以使用CoreMotion.framework提供的API来检测陀螺仪是否存在。

检测陀螺仪是否存在

```
- (BOOL) gyroscopeAvailable  {
#ifdef __IPHONE_4_0
   CMMotionManager *motionManager = [[CMMotionManager
alloc] init];
  BOOL gyroAvailable = motionManager.gyroAvailable;
  return gyroAvailable;
#else
  return NO;
#endif
}
```

5. 检测指南针或磁力计

指南针可用性可以使用CoreLocation.framework中的CLLocationManager类来检查。调用CLLocationManager中的headingAvailable方法，如果返回值为真，你就可以在应用中使用指南针。指南针在与位置有关的应用和用到了增强现实技术的应用中用处比较大。

6. 检测视网膜屏幕

作为 iOS 开发人员，你已经知道只要为应用中用到的每个资源增加一个 @2x 的图片文件，就可以满足视网膜屏幕（Retina Display）的需要。但如果要从远程服务器下载图片，采用视网膜屏幕的设备需要下载的图片分辨率为普通屏幕图片分辨率的两倍。

照片浏览器应用就是一个很好的例子，它类似于 Flickr 查看器或 Instagram。当用户在 iPhone 4、new iPad 或 iPhone 5 上启动该应用时，下载的图片分辨率应该为非视网膜屏幕设备上图片分辨率的两倍。一些开发者选择忽略它，直接为所有设备下载高分辨率图片，但这有点浪费带宽，甚至通过 EDGE 下载时可能要慢得多。相反，你应该在判断出设备使用的是视网膜屏幕之后再下载高分辨率图片。这项检查很容易。

检查设备使用的是否是视网膜屏幕

```
- (BOOL) retinaDisplayCapable  {
int scale = 1.0;
UIScreen *screen = [UIScreen mainScreen];
if([screen respondsToSelector: @selector(scale)])
  scale = screen.scale;
if(scale == 2.0f) return YES;
else return NO;
}
```

在这段代码中，你会找到设备的 mainScreen，然后检查该设备是否可以显示适用于视网膜屏幕的高分辨率图片。这样，如果苹果推出了外接视网膜屏幕（可能是更新的苹果影院显示器），支持目前这一代 iPad 直接以视网膜模式输出，那么你的应用无需任何修改依然能工作。

7. 检测振动提醒功能

在写本书时，只有各个版本的 iPhone 具备振动提醒功能。很遗憾，没有公开的 API 用于检测设备是否支持振动功能。不过，AudioToolbox.framework 有两个方法用来选择性地振动不同版本的 iPhone：

```
AudioServicesPlayAlertSound(kSystemSoundID_Vibrate);
AudioServicesPlaySystemSound(kSystemSoundID_Vibrate);
```

第一个方法会振动 iPhone，而在 iPod touch/iPad 上则会发出“哔哔”的响声。第二个方法只会振动 iPhone。在不支持振动的设备上，它什么都不做。如果你正在开发的一款游戏通过振动设备来提示危险，或是开发一款迷宫游戏，当玩家撞到墙时发出振动，那么应该用第二个方法。第一个方法用来提醒用户，包括振动和发出哔哔声，而第二个方法只能用来发出振动。

8. 检测远程控制功能

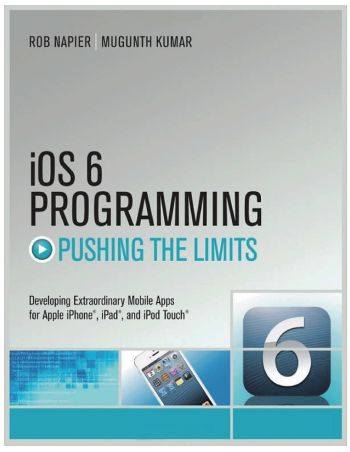
iOS 应用可以处理按下外接耳机上的按钮触发的远程控制事件。处理这类事件，使用如下方法接收通知：

```
[[UIApplication sharedApplication] beginReceivingRemote
ControlEvents];
```

在 firstResponder 中实现如下方法：

```
remoteControlReceivedWithEvent:
```

调用如下方法，确保在不需要这些事件时关闭该功能：



《iOS 6 编程实战》从安全、多任务处理、在多平台上运行、模块和函数式编程、高级文本布局等多个方面讲述了如何为 iPad,iPhone 和 iPod touch 打造杀手级应用，如何将性能最大化，并且从中赚取最大的价值。本文选自《iOS 6 编程实战》。

```
[[UIApplication sharedApplication] endReceivingRemoteCo
ntrolEvents];
```

9. 检测拨打电话功能

可以检查设备是否支持拨打电话，方法是查看它是否能打开 tel 类型的 URL。UIApplication 类的 canOpenURL：方法可以很方便地检查设备上是否有能够处理特定类型 URL 的应用。在 iPhone 上，tel：这类 URL 由 iPhone 上的电话应用来处理。该方法还可以用来检查能够处理给定 URL 的具体应用是否已安装到设备上。

检测拨打电话功能

```
- (BOOL) canMakePhoneCalls  {
  return [[UIApplication sharedApplication]
canOpenURL: [NSURL URLWithString: @"tel: //"]];
}
```

可用性小提示 在 iPod touch 上，开发者应该完全隐藏面向电话的功能。举个例子，如果你开发一个显示因特网上电话号码列表的黄页应用，应该只在能够拨打电话的设备上显示拨打电话的那个按钮。不要只是简单地禁用它（因为用户做什么都无法启用它）或是显示一个错误提醒。有先例说明，在 iPod touch 上显示一个“Not an iPhone”错误提醒会导致该应用无法通过苹果应用审核部门的审查。

iOS 应用

如何初始化基本数据模型?

作者 /Matt Drance
前苹果公司布道师。在苹果公司工作了 8 年，然后创立了一家 iOS 开发与咨询公司 Bookhouse Software。他还为 Pragmatic Studio 培训 iOS 开发者，在 Apple Outsider（appleoutsider.com）上分享对行业的见解。

问题

我们启动的每个项目，最终都免不了从以前的项目或苹果公司的示例代码模板中复制与粘贴事务性的核心数据（Core Data）代码。对于每个新项目，我们需要可靠而切实可行的着手点。

解决方案

核心数据在减少数据库操作所需的代码方面，效果非常出色。然而，在每个使用核心数据的项目中，仍然还有很多冗余的工作。我们需要设置持久存储、模型和托管对象上下文（managed object context），确保支持模型版本间的迁移。如果放置的地方不对，这些代码可能会在项目中产生讨厌的混乱和依赖关系。

本攻略介绍一种基本的数据模型。几乎每个基于核心数据的 app，都需要标准的初始化工作，这将由这种数据模型来完成。我们可以方便地修改或子类化这个模型类，使它满足我们 app 的需要。

除了编码、写作、教学以及履行超级奶爸的义务之外，Matt 还喜欢在北加州滑雪和赛车。

译者 / 刘威

基本数据模型的首要任务，是把每个基于核心数据的应用程序都需要执行的底层初始化操作抽象出来。它初始化托管对象模型、加载持久存储协调器（persistent storage coordinator），并生成默认的托管对象上下文。这个类默认使用 SQLite 存储。

只读的 managedObjectModel 属性是懒初始化的，它指向一个模型文件，默认的文件名为应用程序包标识符的最后一个部分，扩展名为 .momd。比如，如果包标识符为 com.pragprog.BasicDataModel，模型的文件名就是 BasicDataModel.momd。我们可以通过修改 -modelName 和 -pathToModel 来设置模型文件的名字和路径。

```
BasicDataModel/Shared/PRPBasicDataModel.m
- (NSManagedObjectModel *)managedObjectModel {
    if (managedObjectModel == nil) {
        NSURL *storeURL = [NSURL fileURLWithPath:[self
pathToModel]];
        managedObjectModel = [[NSManagedObjectModel
alloc]
                              initWithContentsOfURL:
storeURL];
    }
    return managedObjectModel;
}

BasicDataModel/Shared/PRPBasicDataModel.m
- (NSString *)modelName {
    return [[[NSBundle mainBundle] bundleIdentifier]
pathExtension];
}

- (NSString *)pathToModel {
    NSString *filename = [self modelName];
    NSString *localModelPath = [[NSBundle mainBundle]
```

```
pathForResource：filename

ofType：@"momd"];
     NSAssert1(localModelPath, @"Could not find '%@.
momd'", filename);
    return localModelPath;
}
```

使用有版本控制的模型

如果使用核心数据，从一开始就使用有版本的模型是有好处的。要是开始时用的是显式的 .mom 文件，后来改为有版本的 .momd 文件夹，可能会导致项目中出现陈旧或相互冲突的数据。开发过程中，这会带来难以定位的错误，在使用 mergedModelFromBundles: 来生成模型的时候更是如此。如果是在产品从 1.0 升级到 2.0 时做的改动，这些让人头疼的问题可能跑到用户那里去。所以要从一开始就使用 .momd 文件。

本攻略使用更明确的 -[NSManagedObjectModel initWithContentsOfURL:]，因为在包中可能会加入最初没有想到的数据模型。随 Xcode 4 一起发布的模板就是以这种方式创建的，但较早的项目有可能不是。请检查自己的项目，确保已经做好了版本迁移的准备。

只读的 persistentStoreCoordinator 属性是懒初始化的，预设为在模型版本间执行自动轻量迁移。这样，如果添加的模型新版本只是做了小的改动，并支持轻量迁移，我们就不需要做额外的工作。

请记住，迁移并不总是轻而易举的。对核心数据模型的基本改动，比如添加属性或修改属性名称，通常可以自动迁移。但是，很多情况下，模型的改动太复杂，因而无法自动迁移。在对已经发布

的数据模型进行改动之前，请先阅读苹果公司 iOS Dev Center 里的“核心数据模型版本控制与数据迁移编程指南”（Core Data Model Versioning and Data Migration Programming Guide）。另外，一定要先在 Xcode 里添加模型版本，然后再作修改。千万不要修改没作管理的模型版本！

```
BasicDataModel/Shared/PRPBasicDataModel.m
NSURL *storeURL = [NSURL fileURLWithPath: [self
pathToLocalStore]];
NSPersistentStoreCoordinator *psc;
psc = [[NSPersistentStoreCoordinator alloc]
          initWithManagedObjectModel: self.
managedObjectModel];
NSDictionary *options = [NSDictionary
dictionaryWithObjectsAndKeys:
                              [NSNumber numberWithBool:
YES],
                          NSMigratePersistentStoresAutom
aticallyOption,
                              [NSNumber numberWithBool:
YES],
                          NSInferMappingModelAutomatical
lyOption,
                          nil];
NSError *error = nil;
if (![psc addPersistentStoreWithType: NSSQLiteStoreType
                       configuration: nil
                                 URL: storeURL
                             options: options
                               error: &error]) {
      NSDictionary *userInfo = [NSDictionary
dictionaryWithObject: error
                                                forKey:
NSUnderlyingErrorKey];
    NSException *exc = nil;
     NSString *reason = @"Could not create persistent
```

```
store.";
    exc = [NSException onWithName：NSInternalInconsiste
ncyException
                              reason：reason
                            userInfo：userInfo];
    @throw exc;
}
persistentStoreCoordinator = psc;
```

这段代码还有一个功能，它支持在数据库不存在时，对跟 app 一起发布的数据库进行“预安装”。我们在创建模型的持久存储之前进行这一处理，所以在程序第一次启动时可以向用户提供一些占位数据。这部分不是必需的，也不必禁掉这段代码。如果不需要占位数据，就不必提供预安装的数据库文件。

```
BasicDataModel/Shared/PRPBasicDataModel.m
NSString *pathToLocalStore = [self pathToLocalStore];
NSString  *pathToDefaultStore  =  [self
pathToDefaultStore];
NSError *error = nil;
NSFileManager  *fileManager  =  [NSFileManager
defaultManager];
BOOL noLocalDBExists = ![fileManager fileExistsAtPath：
pathToLocalStore];
BOOL defaultDBExists = [fileManager fileExistsAtPath：
pathToDefaultStore];
if (noLocalDBExists && defaultDBExists) {
      if  (![[NSFileManager  defaultManager]
copyItemAtPath：pathToDefaultStore

toPath：pathToLocalStore

error：&error]) {
        NSLog(@"Error copying default DB to %@ (%@)",
              pathToLocalStore, error);
    }
}
```

NSError 的使用

按照苹果公司的NSPersistentStore的参考文档和通常的Cocoa惯例，除非API返回了表示出错的值（通常是nil或NO），否则不应直接查看传给API的NSError参数。如果没有遵守这个指南，就可能导致难以捉摸而严重的程序bug。

数据模型背后的各个文件（模型文件、工作数据库和预安装的“默认”数据库）的位置，被抽象到存取器方法中，我们可以对它们进行修改或覆盖，以定制这些路径。

- storeFileName返回SQLite数据库的名字，它与模型（.momd）文件的命名方式类似：如果模型叫BasicDataModel.momd，那么存储文件就叫
- BasicDataModel.sqlite。
- pathToLocalStore返回应用程序沙盒（app sandbox）中活动数据库的路径。默认为~/Documents/<storeFileName>。
- pathToDefaultStore返回应用程序包中默认数据库的路径，用于预安装。

```
BasicDataModel/Shared/PRPBasicDataModel.m
- (NSString *)storeFileName {
        return  [[self  modelName]
stringByAppendingPathExtension:@"sqlite"];
}

-(NSString *)pathToLocalStore {
   NSString *storeName = [self storeFileName];
   NSString *docPath = [self documentsDirectory];
    return [docPath stringByAppendingPathComponent:
```

```
storeName];
}

-(NSString *)pathToDefaultStore {
   NSString *storeName = [self storeFileName];
   return [[NSBundle mainBundle] pathForResource：store
Name ofType：nil];
}
```

NSManagedObjectContext 类是大多数核心数据操作的主要接口。我们基本的数据模型以懒加载的方式创建单一“主”上下文，用于其全部查询。因为这是个“基本”模型，所以没有下功夫去做多上下文或多线程的支持。如果需要使用多个上下文，可以简单地修改这个类，生成新的上下文，或者使用主上下文的持久存储协调器来动态创建。

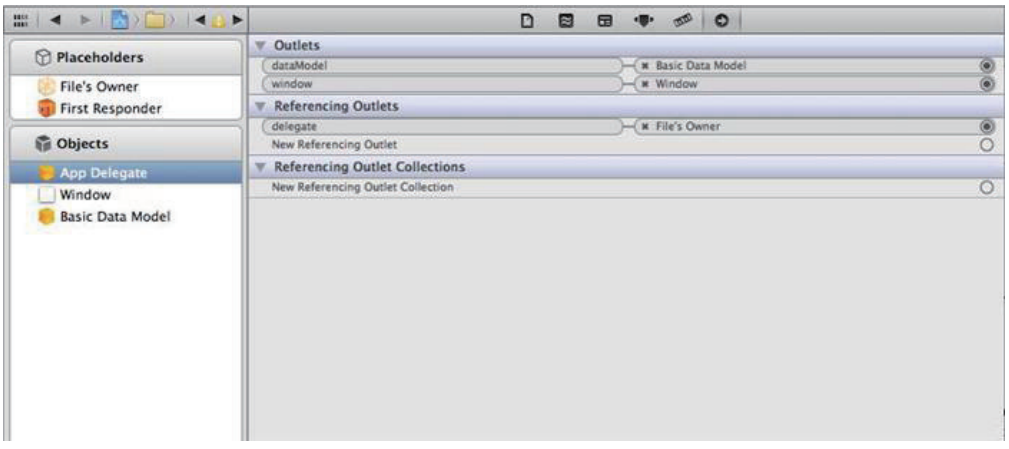
```
BasicDataModel/Shared/PRPBasicDataModel.m
- (NSManagedObjectContext *)mainContext {
    if (mainContext == nil) {
          mainContext = [[NSManagedObjectContext alloc]
init];
           NSPersistentStoreCoordinator *psc = self.
persistentStoreCoordinator;
           [mainContext setPersistentStoreCoordinator：
psc];
    }

    return mainContext;
}
```

本攻略附带的 BasicDataModel 项目使用 Interface Builder 创建模型，并把它连接到应用程序委托，如图 38 所示。根据需要，可以把模型或者来自模型的 NSManagedObjectContext，传给其他需要访问模型数据的对象。请记住，使用一个上下文的持久存储协调器，可以创建新的托管对象上下文。所以，需要的话，我们可

你可能看到过其他 app 里很酷的特性和技巧，却苦于没有时间仔细弄清楚个中原理。《iOS 应用开发攻略》针对 iOS 开发所常见的问题和模式提供了一系列简洁、实用的解决方案。书中收录了最新的 iOS 软件开发的最佳做法，涵盖了应用开发及构建优雅解决方案的必备知识，能让你的 iPhone 和 iPad 应用程序开发实力更上一个台阶。本文选自《iOS 应用开发攻略》。

以不对模型做任何引用，而只是引用 app 委托。

我们可以在代码中或使用 Interface Builder 来实例化 `BasicDataModel`。Interface Builder 可以很方便地识别出模型，并把它连接到 app 委托的 `dataModel` 属性。



使用 Interface Builder 初始化模型

在应用程序中使用核心数据，意味着我们需要做一些多余的工作。把这些代码分离于一处以便于复用，可以减少每个项目的工作量，并避免粗心引起的 bug。

专题：App 创富传奇

App Store 提交申请全攻略

作者 /Todd Moore

当 Todd 还是一名学生时，他就在 CIA 实习了，由此开始了他的职业生涯。Todd Moore 后来创立了 TMSOFT，致力于创建精彩绝伦的智能手机应用和游戏。很少有开发者能够有 2 个应用同时上 iTunes 付费下载应用的 Top 20，而他就是其中的一个。他最受欢

我将仔细讲述向 App Store 提交申请的整个过程。自从 iTunes Connect 网站上线以来，已经有了大量的改进。提交申请的过程也得到了简化，现在已经出现了许多自动检测，可立即验证你提交的申请。除了申请待审核的时间，整个过程耗时最长的是创建所有屏幕截图和应用程序简介。只要这些完成了，实际的提交过程是相当简单的，不会花费很长时间。

屏幕截图

我需要你截取一些引人注目的应用程序屏幕图像。你可以使用 Mac 上的 iPhone 仿真器来截屏，但我发现，最简单的方法还是使用设备自身的截屏功能。想要在设备上进行截屏，你可以同时按下屏幕锁定按钮和 Home 键。你将会看到 iPhone 的屏幕闪了一下，然后屏幕快照就被存储在 Camera Roll 中了。我建议截取大量不同的屏幕图像，以确保展示你游戏独一无二的部分。第一张屏幕截图最关键，它需要能反映出游戏的良好操作。我做的第一个屏幕截图（如图 1 所示），是带比分显示的游戏截图。然后再添加其他截图，例如展示了桌上冰球游戏所有不同选项的标题界面。这些看起来都很不错，将会有助于用户作出购买决定。

迎的游戏《21 点算牌器》（Card Counter）得到了 Engadget、《洛杉矶时报》和 CNET TV 的推荐，而最受欢迎的应用《催眠白噪音》（White Noise）得到了 iTunes、美国《健康杂志》、《华盛顿邮报》、《个人电脑杂志》和《Jimmy Fallon 深夜脱口秀》的推荐。

译者 / 杨晓琪

如果你使用仿真器来截屏，那么你可以使用键盘快捷键 Command+Shift+4，按下空格，然后点击仿真器窗口。这会取到整个仿真器的内容，包括 iPhone 画面，然后以图像格式保存在桌面上。这幅图像用在你的网站上可能还不错，但是要用在 App Store 上你就需要将它裁剪为真实设备的屏幕大小。只要使用你常用的照片编辑器，然后将仿真器的屏幕内容裁剪为下面所提到的正确尺寸。

每个应用可以有 5 张截图，所以尽可能多提供一些。如果你无法提供足够有特色的截图，那也没关系，就用这些吧。虽然要求至少提供 1 幅截图，但请记住，大部分人决定是否购买你的游戏很大程度上取决于你展示的截图，所以提供最好的图像是非常重要的。截图可以是 320 像素 ×480 像素的标准分辨率，或者是 Retina 显示的 640 像素 ×960 像素的高分辨率，并且横竖屏均可。如果你的游戏支持 Retina 显示，那么最好使用相应的格式。我总是使用我的第 4 代 iPod 或 iPhone 4 来截屏，这样就能产生高分辨率的图像。如果你使用的是旧版的设备，可以指定为标准分辨率。

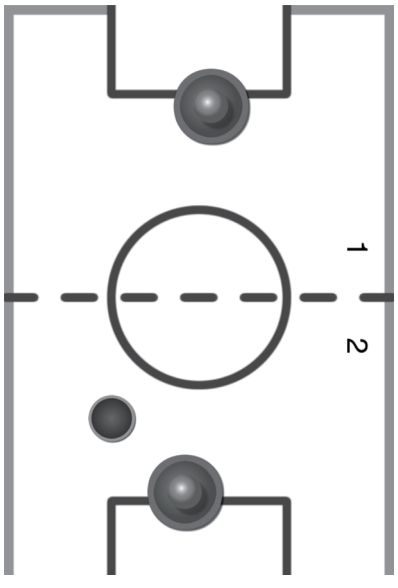


图 1　桌上冰球游戏截图

在提交之前我对照片进行了重命名，这样我就能清楚地知道我要上传到 iTunes Connect 的图片顺序。例如，AirHockey1–5.png 可以提示我先提交 AirHockey1.png，接着是 AirHockey2.png，以此类推。试着让所有东西都井然有序，因为你很有可能会再需要这些照片的。

创建应用程序简介和关键字

首先，你需要决定游戏叫什么名字？名字在 App Store 上必须是独一无二的，以区别所有其他应用。最好的方法是，先在 iTunes App Store 上搜索下你想取的游戏名字，看其是否已经存在。你还要确保这个名字没有被某一家公司事先注册，这就要在 Google、United States Patent 和 Trademark Office（www.uspto.gov） 网站上搜索下此名字了。你的游戏应该是独一无二的，这样用户才能轻松找到它。当你在 iTunes App Store 上搜索应用时，此应用程序名称会被检索到。

现在，让我们来创建应用程序简介。当用户在 iTunes 上关注你的应用时，首先读到的就是这段文字。在 iTunes 中，这一字段不会被检索到，所以这仅仅是将你的游戏是什么和它的特色是什么传达给潜在用户的一种方式。桌面版的 iTunes 只显示应用程序简介的头 3 行，其他部分则省略。在设备上，通常我只阅读第一段，然后就滑到屏幕截图去了。所以，开始的几句话是相当重要的。试着在第一句话中就引起读者的注意。以下是我为桌上冰球游戏所写的简介的第一段内容。

> 现在，你可以将桌上冰球游戏放进口袋中了！真实的游戏体验，直观的多触点控制，而且还是免费的。你可以和朋友面对面进行游戏或是挑战多种难度等级的电脑。聆听冰球撞击球拍的声音，观看冰球滑动时真实的物理行为，体验进球时那激动人心的一刻。

确保应用程序简介是真实可信的，并且只提及游戏当前所支持的功能。千万不要在应用程序简介中提及有关价格的信息，苹果不赞成这么做。这是因为价格信息会被转化为 iTunes 用户本地的货

币，所以，如果你说“只要 0.99 美元”，这会造成英国用户在购买按钮上看到不一致的信息。如果说“免费”或“限时免费”这是可以的，这将有助于吸引更多的用户在收费之前来玩你的游戏。

接下来你需要填写关键字，这部分非常重要。当用户在 iTunes 上搜索应用时，就会用到关键字和应用程序名。你需要挑选出精炼的关键字，以帮助用户找到你的应用。App Store 刚开放的时候，使用的是应用程序简介作为搜索匹配，但是很快就被人误用了。用户会输入一长串的关键字进行搜索，结果只能是没有搜索到任何一款应用。苹果接下来做了一次调整，不再检索应用程序简介，而是要求开发者提供关键字，而且关键字的长度被限制在 100 个字符以内。不再允许添加一长串的关键字，这样你就不得不理智地进行挑选。每个关键字都以逗号隔开，不需要包含空格。逗号同样也被计算在最大字符长度的限制之内。你可能会发现，使用字符计数器会很有用，这样你就能确保使用尽可能多的字符了。网上有很多在线的字符计数器，你可以将关键字粘贴进去，然后获得字符总数。在 iTunes Connect 允许的范围内，提交尽可能多的关键字，这是相当重要的。

> Google AdWords 是一个很出色的网站，它可以帮助你找到不错的关键字。你不仅可以通过此站点来访问 Google 的搜索结果，而且它包含了一个关键字优化工具，可以帮助你找到用户搜索某些术语时所使用的关键字。我发现它是查找流行关键字非常有用的工具，而且还是免费的。当用户想查找“Air Hockey”时，他们会搜索什么内容？ Google 关键字优化工具会告诉你答案。

向 iTunes Connect 提交元数据

现在，你已经有了屏幕截图、应用程序简介和关键字，接下来你将向苹果提交这些信息。iTunes 需要让用户找到你的应用，而 iTunes Connect 网站就是让你提供所有这些信息的地方。这里还管理着你的合同、税收和账单信息；如果你的应用是收费的，那么这些都是必要信息。另外，只要应用在 App Store 上可供下载，iTunes Connect 就会为你提供应用的销售信息。

登录到 iTunes Connect，然后点击 Manage Your Applications。在这一界面中，你可以更新已有的应用程序或是添加新的。点击 Add New App 按钮，开始提交应用信息的过程。转到一个新的界面，这里系统会询问你 App Name、SKU Number 和 Bundle ID。App Name 要求是唯一的，且少于 255 个字节。当你提交时，会被告知名字是否是唯一的。我总是在 iTunes 上先搜索下，看是否有其他相似的名字。但这么做并不总是有效的，因为开发者在真正将应用提交到商店之前可以将一个名字保留 90 天。

> 你可以在真正提交应用之前就注册一个应用程序名，因为在真正发布之前，这可以提供充分的时间去为它做宣传。应用程序名一经注册，那么其他人都无法在之后的 90 天内再次注册它。确保在这一时间段内提交你的应用，因为如果没来得及提交，那么你注册的名字将过期，而且你将再也无法注册该名字了。

SKU Number 是你必须分配给应用程序的唯一编号。虽然它要求数字，但你也可以使用字母，我通常使用的是全部大写的应用程序名，并在必要的地方加上下划线。例如，如果我提交的应用叫 Air Hockey，那么我会使用 AIR_HOCKEY 作为 SKU Number。这

是唯一你将来无法更改的元数据，因为它只会在内部和销售报告上用到，而永远不会出现在用户面前。

Bundle ID 是一个下拉框，你可以从中选择任何在 iOS Provisioning Portal 中创建的 AppID。就我而言，我注册了一个通用的 App ID——com.toddmoore.，我提交的多个应用都是用的这个 App ID。它不支持 Apple Push Notification 服务或 Game Center，因此，如果你需要这些特性，就必须为应用程序创建特殊的 App ID。只要你创建了 App ID，它就会出现在下拉框中。由于我选择的是一个通用的 App ID，所以还添加了额外的 Bundle ID 前缀以使它成为唯一的标识。这需要与创建 Xcode 项目时所用的 Bundle ID 相同。在本例中，我使用 AirHockey 作为后缀，这样完整的 Bundle ID 就设置为 com.toddmoore.AirHockey，与项目中所使用的匹配。

现在，这些信息已经输入完毕，点击 Continue 按钮，查看是否有任何问题。发现的任何错误都会显示出来，如“你输入的 App Name 已被使用”。当时我想使用 Air Hockey 名字时就碰到了这个错误，因为别人已经取过这个名字了。我尝试了一些不同名字，如 Air Hockey Free、Air Hockey Pro 以及 Air Hockey Extreme，但是这些名字都已被占用了。于是我决定在最后添上本书的书名组成应用程序名 Air Hockey：Tap, Move, Shake——总之，比起之前几个，它是一个更好的标题，因为如果有人在 iTunes 上搜索本书，那也会显示这款应用。

在下一个界面中，你将要设置 Availability Date 和 Price Tier。我通常将有效日期设置在一个月之后。只要应用程序审核通过，我就会将日期修改到我想要发布应用的那一天。如果将日期设置为

过去的某一天，则会使你的应用排在 iTunes 发布日期列表很靠后的位置。从而导致你的应用难以被发现，发布应用的第一天将会很糟糕，因为人们很难从列表中找到它。如今，苹果终于修复了这个问题，所以使用更早的日期不再会对你不利，只是一旦审核通过，应用程序就会发布。

你可以为应用定一个价格，或是选择免费。就我而言，我准备使这款游戏免费，以让任何人都能玩到，或许用户还会学到一些有关怎样构建游戏的内容。如果你的游戏想要收费，那么就需要选择一个 Price Tier。在美国市场当前的价格段是在 0.99 美元到 999.99 美元之间。点击 Pricing Matrix 链接，查看每个价格段对应的价格和收益情况，收益通常是销售额的 70%。我倾向于将 iPhone 游戏的价格定得稍低一些，如 0.99 美元或 1.99 美元。App Store 上的 Top Paid 排名是基于售价而非收益，因此，如果你希望你的应用能进入排行榜的前列，通常便宜的价格能争取到更高的排名。还有一个 Top Grossing 排名，这个是基于整体收益情况的，显示了谁在 App Store 赚了最多的钱。这是一个有趣的排名，因为它显示了开发者的赚钱能力（包括免费应用中的内置付费）。经常去访问下 App Store，看看哪款应用最赚钱。在你有一堆想法的时候，这能帮助你决定做什么类型的游戏或应用。点击 Continue 按钮提交这些信息，然后转到下一个界面。

接下来我们需要指定 Version Number。如果你已经在其他地方发布过这款游戏，例如在 Mac App Store 上，而且都是基于相同的源代码或功能设定，那么你通常会使用相同的版本。如果这是你的第一个版本，一般指定为 1.0。在一段时间之后，你总是会更新应用程序，这就需要增加版本号，让新的版本号高于之前的。例如，你提交了 1.0 版本，当你再次提交更新时，可以改为 1.1 或 1.01。

更新应用程序时，可以遵循能反应更新主次的格式。我使用的格式是 MAJOR.MINOR.REVISION，应用只有在添加了主要功能时才会增加 MAJOR 位，增加 MINOR 位则表示添加了次要功能，而 REVISION 位的增加则只是修正了当前版本的一些 bug。这就是我实现的版本方案，并且对我很有帮助。但对于每一个位数的增加，你也可以有不同的实现。如果你改变了多个位数，那么最高位就需要增加。例如，你可能提交了版本从 3.0.5 到 3.1.2 的更新，第二位从 0 增加到了 1。你也可以将版本从 3.1.2 更新到 4.0，表示这是版本从 3 到 4 的主要更新。

举一个简单的例子。首先，你提交了 Air Hockey 的 1.0 版本，此版本只支持双人游戏；在接下来的版本中添加了电脑玩家，这是一个足够大的特性，可以成为 2.0 版本；然后又在 2.0 的基础上为电脑增加了不同的难度等级，那就可以命名为 2.1 版本；在发布 2.1 版本之后，你发现程序有一个小 bug 需要修复，那就再提交 2.1.1 版本来解决问题，这是合理的，因为此版本只是修复了 bug 而并未添加任何额外的功能。

现在，填写如下剩余的元数据。

- **简介**：将之前已经完成的应用程序简介粘贴进来。这不是一个强大的文本编辑器，所以不同的字体或类型（如粗体和斜体）无法被格式化。只要确保你粘贴进来的文本在每一段之后保留一个硬回车就行了。
- **主要分类**：主要分类选择 Games，然后挑选两个子分类。我为这款游戏选择的子分类是“Sports”和“Family”，不过你也可以选择任何你希望的子分类。
- **次要分类**：这项分类只用在当用户在指定的分类中搜索 iTunes

时。我选择 Entertainment 作为次要分类。应用不会列在此分类中，只有用户在搜索（而非浏览）此分类时才会显示。其实主要分类比次要分类要重要得多。

- **关键字**：将前一节中已经准备好的关键字粘贴进来。其中的字数被限制在 100 个字节以内，如果超出了范围，你可能会收到一个错误提示。如果真是这样，就需要删除一些相对不是那么重要的关键字直到符合要求为止。
- **版权信息**：我将我的 Paddles 和 Air Hockey 的版权信息设置为 2011 Todd Moore。如果你已经注册了一家公司，那么可以使用公司的名字。
- **联系电子邮件地址**：最好建一个非个人的电子邮件地址。我使用 support@toddmoore.com 作为联系地址。这一电子邮件地址将会出现在许多网站上，所以会收到大量的垃圾邮件。
- **技术支持网址**：这可以是你的个人网站。我使用 http：//toddmoore.com，大家可以在上面与我联系。
- **应用程序网址**：这一项是可选的，但还是建议填写。许多网站都会抓取 iTunes 上的应用列表，然后将所有的链接发布在它们自己的网站上。这样有助于在搜索引擎中提升你网站的排名。我使用 http：//tod-dmoore.com/book 作为我的应用程序网址。
- **审核备注**：你可以向苹果审核你应用程序的人员发送任何消息。如果你的应用要求账户登录，那么你就需要提供一个试用账号供审核人员登录。你还可以发送一些小技巧或很酷的东西让审核人员检查。如果你在游戏中放置了一枚复活节彩蛋，那就需要让审核人员知道如何来获取它。
- **等级评定**：我选择了 None，这一项提供了一个等级为 4 的年龄排名，这决定了父母设置过年龄限制的设备是否可以下载你的应用。苹果保留了调整这些值的权利。
- **EULA**：如果你需要创建自己的 End User License Agreement，

可以添加这部分的信息。如果你没有提供，那么会将标准的 EULA 会添加到应用中。我没有提交自定义的 EULA，而是使用苹果提供的标准 EULA。如果你包含的一些库需要独立的许可，那么可以在这里添加。

- **上传**：在应用程序图标上点击 Choose File 按钮，然后选择第 3 章创建的 512 像素 ×512 像素的大图标。现在，添加前一节 iPhone 和 iPod touch 屏幕截图中已创建的所有的截图。由于提交的并不是通用的应用，所以无须上传 iPad 截图，反之则必须上传。

点击 Save 按钮，现在应用程序元数据就已保存在草稿表单中了。只要在提交应用程序之前，你都可以取回这些信息并进行编辑，如图 2 所示。应用程序现在处于一个新的状态——Prepare for Upload，这是你接下来要做的事情。如果你打算之后再提交应用，那么要做的就已经完成了。当你准备提交时，再次打开应用程序信息页面，点击 View Details 打开当前版本，然后按下 Ready to Upload Binary 按钮，应用程序会进入 Waiting for Upload 状态，此时将进行应用归档并提交审核。

归档和提交

你可以将应用程序归档提交到 iTunes Connect 或和其他人分享。在 Release 模式下构建应用程序，能产生更高效的代码。到目前为止，运行的应用程序都是构建于 Debug 模式的。应用程序的最终版本将不包含调试符号，而且经过优化，代码将变得更高效。当编译器优化代码时，它会创建一个完全不同的二进制文件，而且可能导致不同的应用程序行为。出于这一原因，在将应用提交到 App Store 之前，一定要在设备上测试 Release 构建的应用程序，这点非常重要。

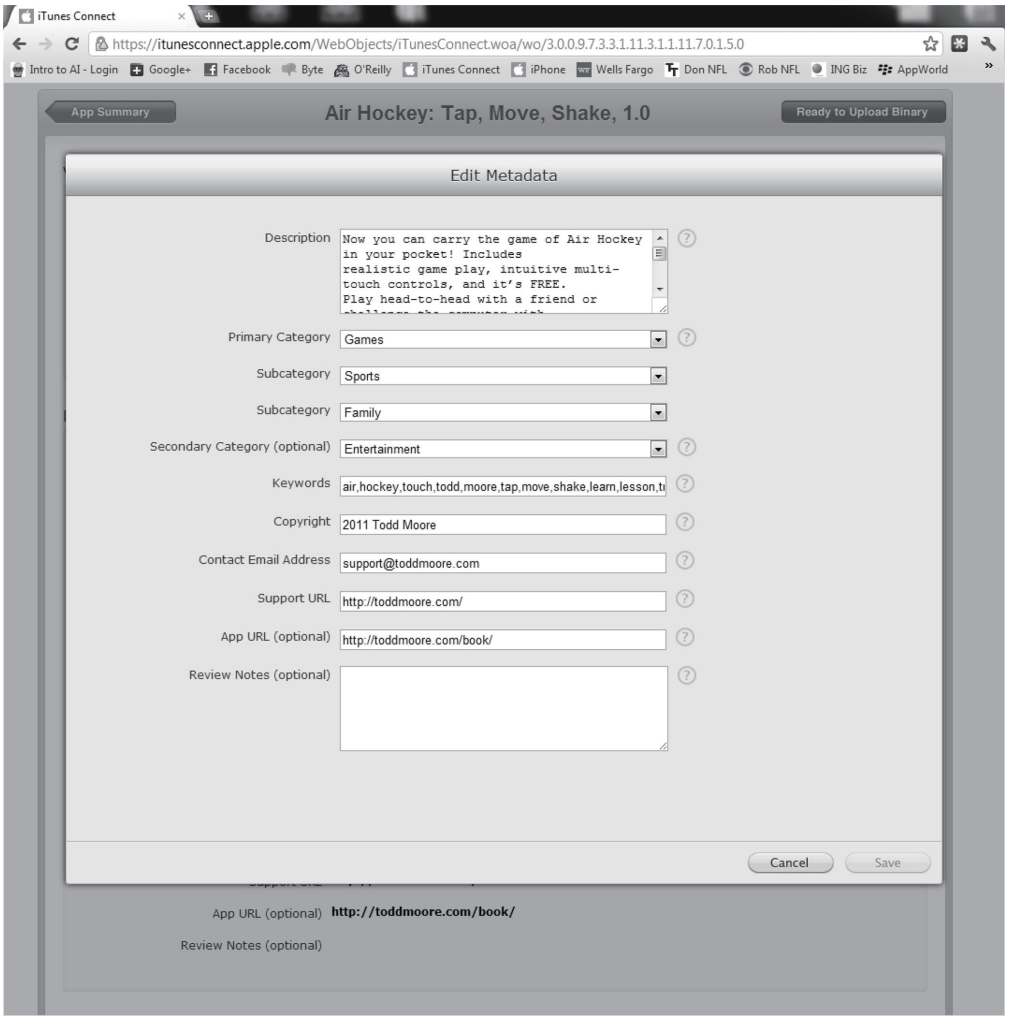


图 2　编辑应用元数据

为了在 Release 模式下构建应用程序，你需要编辑你当前的 Scheme。选择 Product → Edit Scheme，改变运行方式，将构建配置从 Debug 更改为 Release。如果你是根据此配置来构建应用程序的，那么运行的应用程序也会是同样的配置（向 App Store 提交应用时使用的配置）。全面测试应用程序，以保证运行时的所有功能都与 Debug 模式相同。

完成测试之后，选择 Product → Archive 来构建并压缩应用程序，用于提交到 App Store。此时会打开 Organizer，在这里你可以访问所有为每个应用程序创建的归档。你可以从 Xcode 中选择 Window → Organizer 来访问 Organizer。归档构建完成，并在 Organizer 中选中它，此时你就可以选择向 iTunes Connect 网站验证或提交应用了，如图 3 所示。

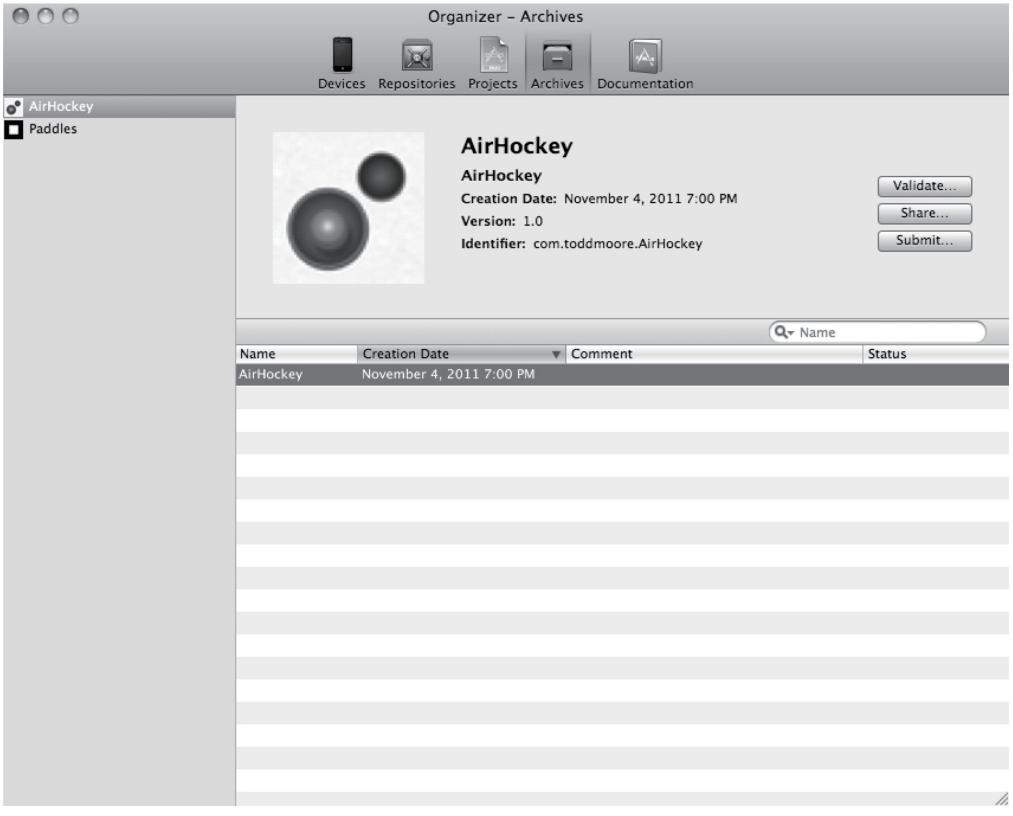


图 3　Xcode Organizer

你应该首先验证应用程序，因为 iTunes Connect 会针对应用程序二进制文件运行一系列的自动测试。系统会提示你输入 iTunes Connect 账号，之后是 Signing Identity，这和你用于对象签名的 App Store 账号是相同的。这并不意味着你的应用程序这样就能

让《iOS 游戏开发：从创意到实现》这本完全 DIY 编程指南来帮助你！通过本书的教程，你将亲手开发一款真正的 iOS 游戏，并全面了解 Xcode 和 Objective-C 的相关内容，同时还能学到如何实现游戏逻辑、精美的图像效果、游戏物理仿真效果、音效以及电脑 AI。在完全不了解苹果开发工具的情况下，作者 Todd Moore 仅靠自学就在一个星期内创建了一款 iPhone 游戏。如今，他正全职开发智能手机应用和游戏。有了本书的佐助，每个人都可以实现自己的游戏开发构思，让作品顺利打入 App Store。本文选自《iOS 游戏开发：从创意到实现》。

通过审核了，但至少可以让你知道与应用程序相关的东西都很重要，如尺寸正确的图标。如果验证失败了，你会收到一条错误信息和关于如何解决问题的描述。如果你需要更多的信息，可以在 Google 上搜索返回的错误，通常你会发现其他开发者也有着同样的问题并从中找到解决问题的方法。

如果你的应用程序已经通过了验证，那就可以提交给 iTunes Connect 了。步骤与验证应用程序时的相同，但这次是选择 Submit 而不是 Validate 来开始此过程。输入所有相同的信息，包括 iTunes Connect 账号和 App Store 证书。提交上传完毕，你会看到二进制文件被退回或是正在等待审核。

如果二进制文件被退回，则你必须重新提交并修复报告中的所有错误。最常见的原因是没有使用正确的用于 App Store 发布的证书。如果是这样，你就需要返回到项目中，然后确保项目目标构建选项中存在正确的指定为 Release 模式的 App Store 签名证书。

专题：App 创富传奇

移动应用 UI 设计反模式一瞥

作者 /Theresa Neil

Theresa Neil 是德克萨斯州奥斯汀市的一名用户体验咨询师。她组建了一个用户体验设计小组，其中的设计师和开发人员都具有十分丰富的从业经验。她的设计小组与客户紧密协同，致力于创建能令人产生愉悦、提升工作效率和自信心的产品。她最新的项目是为餐馆设计一款桌面应用程序，让客户能在自己的餐桌前点餐。其他

常见反模式：标新立异（Novel Notion）、隐喻错位（Metaphor Mismatch）、愚蠢的对话框（Idiot Boxe）、图表垃圾（Chart Junk）、按钮海（Oceans of Button）

什么是反模式？维基百科的定义为：

> 反模式也称为“陷阱”（Pitfall），通常指对于某些问题所采取的没有实质性革新的、糟糕的解决办法。人们已将其作为界面设计的一种模式进行了研究，以便在后续的设计中避免犯同样的错误，也希望在对无法工作的系统进行调查时也能识别出此类模式。这一术语源自计算机科学，很明显是受到“四人组”的 Design Patterns 一书的影响，该书介绍了一些高质量编程的方法。
>
> ——维基百科，反模式

更多项目，请浏览她的网站 www.theresaneil.com，或访问她的 Twitter 账户 @theresaneil。

译者 / 王军锋

与软件设计中的反模式类似，以下反模式也是界面设计中要避免的常见设计陷阱，本篇文章由于篇幅限制，将只介绍“标新立异”这一种反模式，阅读全文，请移步图灵社区。

标新立异

标新立异式设计的目的在于求新、与众不同、突出创意或创新。但大部分时候，这种做法是错误的，用户难以理解，产品很难使用。BUI*Gallery 的创始人 RichardGunther 关于这一反模式的描述为：“你通常能区分出移动应用开发团队是否具有传统网络应用开发的背景。这些团队常常会尝试把旧有的交互模型引入到新的应用平台，并设计一些非标准 UI 元素来凸显他们的‘创意’。”

标新立异式的反模式在移应用中随处可见，从主要导航到个别的控件、交互方式或手势设计。Weight Watchers 应用就包含有许多标新立异式的反模式，最明显的莫过于主导航的设计。其开发团队创造出了一个自定义菜单容器，其中包含有跳板式的主导航（见图 1）。

图 1　Weight Watchers 应用

这种反模式非常糟糕！产品臃肿得难以使用，界面元素的组织十分随意，那些能把访问者变成用户的重要产品功能隐匿不现（在跳板式导航的第二页）。虽然我能明白 Weight Watchers 的用意，但如果他们采用了标准的导航模式、精心组织菜单选项，应用可能会更容易上手（见图 2 和图 3）。

图 2　选项卡式的菜单选项，线框图（左图）；可滚动的选项卡选项，线框图（右图）

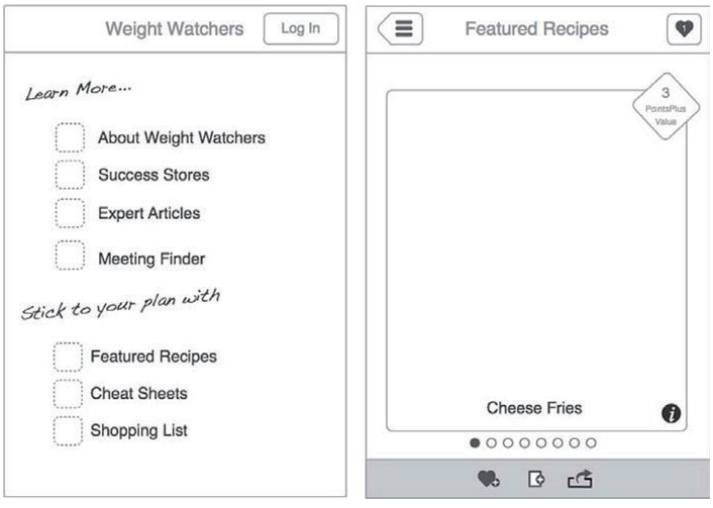


图 3　列表式的菜单选项，线框图（左图）；内部屏幕的概念（右图）

iPhone 上的 Fly Delta 应用的菜单就很不同寻常，但现在它已被另外一种“标新立异”的反模式取代——用户（执行滑动操作后）会发现有页脚内容，其中包括有“主页面”按钮和一个指向“Contact Us”的链接（见图 4）。

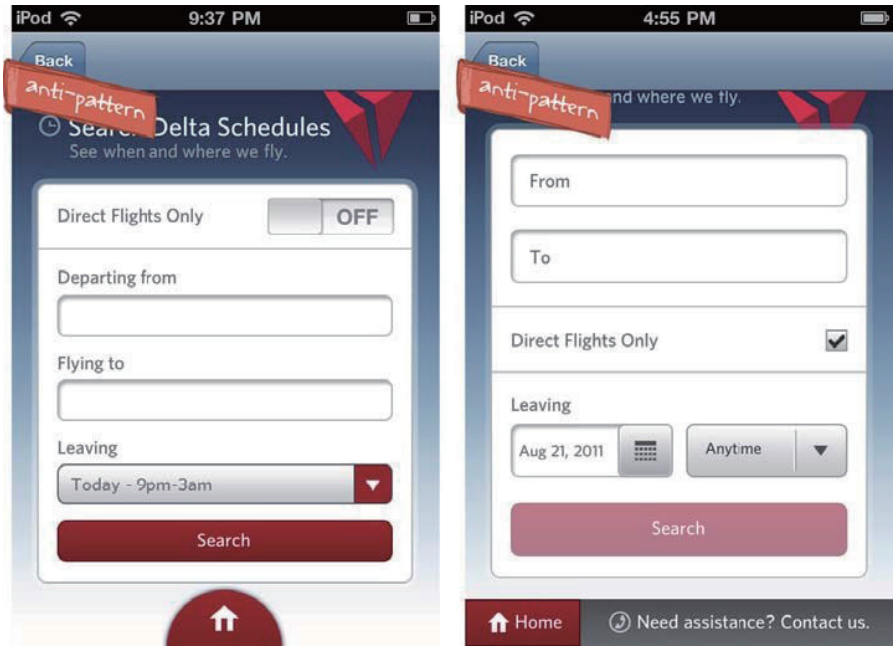


图 4　Fly Delta 先前（左图）和新的（右图）菜单设计

如果产品支持单击一次就能访问主界面的功能，标准的 iOS 选项卡也能起到很好的效果。Contact Us 本应该显示在页面的右上角，而不是“躲”在页脚处。这种菜单设计能称得上是“创新”吗？完全不能，它只会让应用更难用。

有时，富有天赋和经验的设计团队能充分利用技巧把“标新立异”式的反模式变得更可用一些。Gowalla 的设计师将标准的选项卡菜单反其道行之，设计出了漂亮且可用的菜单。Forecast 试图模仿 Gowalla 的设计，但结果却是画虎不成反类犬（见图 5）。

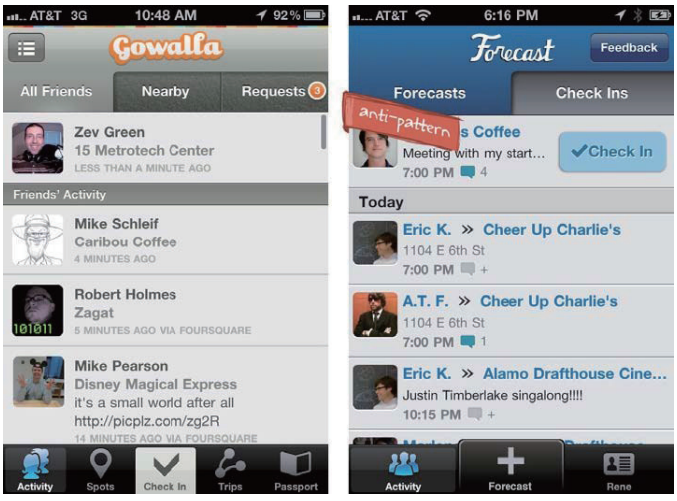


图 5　Gowalla 和 Forecast 应用

Forecast 中只有两个风格化的选项卡，用户必须仔细观察才能确定自己当前看到的是 Forecasts 还是 Check Ins 的信息。视觉设计并没有起到应有的效果，Check Ins 选项卡的凹进效果看上去好像用户选择了它，但其实不然（见图 6）。

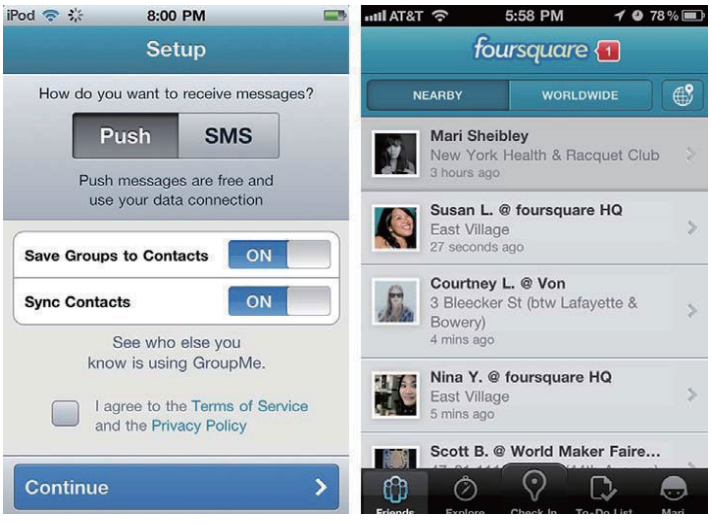


图 6　凹进风格通常表示用户已选择的选项

一个简单的触发栏就能揭开用户的困惑。实际上，在仔细了解过这款应用的使用流程之后，我认为程序应该根据时间的先后关系，以单独的“活动”选项卡显示用户到底选择的是 Forecasts 还是 Check Ins，这才是最佳的设计方案。这样的设计需要为 Forecast 建立一个用户可识别的 Forecast 图标，用以暗示好友是预报，还是已到了某个地点。

Forecast 也设计了一个自定义选项卡栏，但却有点简陋。这类自定义工具栏非常适合那些只包括一个主要的调用操作的应用，如把内容发布到 Pinterest 或通过 Instagram 分享。实际上，Forecast 只包括两个主要的操作——Forecast 和 Check Ins（见图 7 和图 8）。

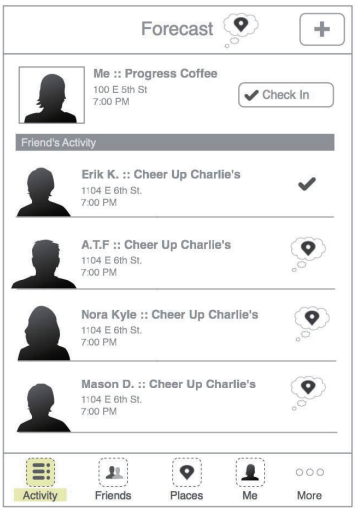


图 7　采用 iOS 系统标准标签的线框图

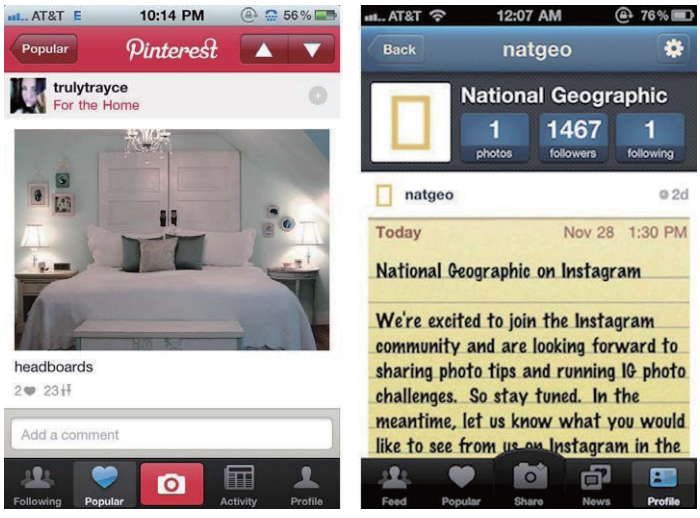


图 8　Pinterest 和 Instagram 应用

风格统一的“Add Forecast”按钮应该起到很好的作用，吸引新用户对其进行操作，完成软件的第一次使用。另一种方案是设计“Forecast”按钮的图标，并将其放在自定义工具栏内（见图 9）。

《移动应用 UI 设计模式》这本简要的手册提供了 70 多种移动应用设计模式作为参考，包括了从当前 iOS、Android、BlackBerry、WebOS、Windows Mobile 以及 Symbian 平台中提取的超过 400 张的应用截图。用户体验设计大师 Theresa Neil 将向你介绍包括反模式在内的 10 种不同类型的设计模式。无论你正在设计一款简单的 iPhone 应用，还是开发适用于当前市场上所有流行移动操作系统的应用，这些设计模式都能助你一臂之力。本文选自《移动应用 UI 设计模式》。

图 9　能吸引用户的 Add Forecast 按钮（左图），带有自定义工具栏的备选方案（右图）

如果你想要对应用进行创新，请把关注点放在主要的功能和所提供的服务上，对于界面设计，最好还是遵循以往的设计惯例。如果你设计了一个自定义控制项，一定要对其进行严格的测试和修正，最终达到可用性的要求。

读《码农》吐吐槽 还能赚银子！

《码农》电子刊如今慢慢悠悠已经出版了 5 期，从一开始的好评如潮，到现在的“怎么这么抽象啊”“赶脚内容没有以前好了？”“一下就翻完了，没什么内容啊”……小编表示压力很大，现在的码农读者太难伺候了，明明是本免费杂志，￥%#%&*%@# ￥*@ ￥#@！

但是，一看到好评，盼盼姐还是会热泪盈眶，热血沸腾，各种正能量啊“这么好质量的杂志还免费，天上掉馅饼呀”“希望一直出！每一版都读了，很值得推荐！！”……

所以《码农》还是会一直做下去，但是，是好是坏，您得给个话儿啊！

活动规则：在吐槽贴中留言，选出任何一期中你最喜欢的文章和最不喜欢的文章，即可获得图灵社区银子 2 两！凡吐槽吐得掷地有声者，加赠银子 3 两（共 5 两）！（怎样使用银子兑换图书）

活动时间：本活动长期有效。

阿怪：二进制？十二进制？

陈志翰，笔名阿怪，台湾知名资深音乐製作人。为张惠妹、陶晶莹、萧淑慎、曾宝仪、范逸臣、萧敬腾、古巨基、苏永康等多位知名歌手创作出眾多膾炙人口的歌曲。除了深厚的资歷之外，在业界更是音乐製作的权威代表之一，为了配合雪碧广告而专为阿妹打造的《给我感觉》也因在亚洲地区连续播放使

陈志翰（@Sir 阿怪），来自台湾，他是唱片制作人，为张惠妹、范逸臣写过主打歌；他还是码农，每周都要去参加“嘿嘿星期四”的 coding 聚会。关于他，我们有无数的问号，但是在访谈结束后，他的跨界生活似乎又变得顺理成章，不紧不慢。“写歌和写程序对于我来说就是一件事。”“写程序写得好好的干嘛创业呢？”本期跨界码农陈志翰，将向你展示一种不一样的生活态度。他的不同来自于何处，可能看完之后，你就会有自己的答案。

会写歌的数学系学生

> 我从来都没有讨厌过数学老师，就算老师讨厌我也没关系，你能把我怎么样呢，你能咬我啊？

图灵社区：你大学是学什么的？

因为我喜欢数学，大学读的是数学系，我们系就没有几个女生，

阿妹成为名副其实的亚洲天后，而《I Believe》更是让范逸臣成为唱片店问询率最高的未发片新人。

仅有的几位也很难算得上女人。台大数学系读一年，被退学。当时没有想到他们（校方）那么强势，不去上学就会被退学（笑）。后来换了一家大学，文化大学，又被退学。最后我终于在东吴大学数学系毕业了。这种事习惯就好。我们只有前十名可以继续读硕士，读博士。那其他人只能当中小学老师、补习老师，而我又不喜欢当老师。

但是后来我在英国读硕士时，也会接家教的工作。那边的小孩如果数学不好也不会讨厌数学，因为他们的老师都很喜欢数学。他们可能会说："你不会没有关系，可能你还没有开窍，但是拜托你不要讨厌数学哦。"这是一种感情问题，很多人是因为讨厌数学老师，才导致后来数学不好。像我，就是比较幸运的了。我从来都没有讨厌过数学老师，就算老师讨厌我也没关系，你能把我怎么样呢，你能咬我啊？（笑）

图灵社区：你的演艺圈生涯是怎么开始的？

我从高中的时候就开始自己玩些音乐器材，我国中（初中）开始学吉他。其实主要是在鬼混。我大三开始就做杨德成的 BOM boy（扛麦克风的）。数学系读不好，自己心里也知道，但是还能写点东西，而且还爱慕虚荣，想认识明星，我就琢磨怎么能混进演艺圈。当时刚好有人在做电影《麻将》的原声带，我就去问杨导（杨德昌）："我在写歌，能不能听一下？"他听完了之后，觉得还不错，于是我就为张震写了一首《无能为力》，那是我第一首拿出来发表的歌曲。那个时候数字音乐还不发达，当时最实在的事是硬盘剪辑、硬盘录音。我做音乐的初期，依赖人手动的成分还是很大，跳进跳出都要人手动调整。要搞线性剪接像鬼一样的困难。我妈是一位舞蹈学院教授，我从小就拿 16 厘米或者超 8 的胶片在剪辑。我

总是在问：“我真的可以剪吗？”

这就是真正的剪辑，因为用到了剪刀嘛。

图灵社区：你现在的工作需要和数学或者编程相关的知识吗？

我现在的工作只需要纸跟笔就可以完成了。

但是现在哪有行业会完全不涉及计算机呢？比如小燕姐的女儿在美国学的就是调色专业。做这种工作的机器（操作系统）都是Linux，硬体就要焊上去。如果需要自定义就一定需要 Linux。如果需要做别的事可能用别的操作系统会好一点，但是如果这台机器只是用来调色的话，那 Linux 就是能够让硬体和软体最好地粘合在一起的系统。如果有更进一步的要求，你只能自己写一个 OS 了。所以简单的办法就是，只需要把 driver 搞定就可以了。

图灵社区：很喜欢你为阿妹写的《三天三夜》。

中国好声音还没有给我《三天三夜》的版税，这件事是和经纪公司在谈，他们应该是需要付很多钱的。每次张惠妹开演唱会，就会有朋友给我打电话，喊：“阿怪你听一下！”然后就只听见全场吼叫（“三天三夜……”）而且他们还一定会让我把这首歌听完，才说：“阿怪，你有没有听到啊？”

无论如何，还是要谢谢大家捧场。

图灵社区：“你看起来还挺像码农的。”

阿怪：“我的裤子是 Diesel，我觉得不会有码农会穿 Diesel 吧。这可是意大利版的哦。（真的看不出来。）”

图灵社区：“可是你穿上身之后你的气质就把它掩盖掉了，而且你的衬衫也很有码农范儿。”

阿怪：“这是 Mark Jacob！（经检验，真的是）这一身打下来要 5、6 千块呢。”

图灵社区：“那我们把你的造型照下来，让大家猜一猜吧。”

阿怪：“……好悲伤哦。Diesel 听到也会很悲伤的。没办法啦，这就是我的气质，压不住啦。”

图灵社区：“嗯，有一种说法就是人要穿衣服，不能让衣服穿人。那你现在就做到了嘛！”

码农基因

> “那你现在怎么不混演艺圈啦？”
> “演艺圈的人好无聊啊。”

图灵社区：你是怎么开始学编程的?

我小学三年级的时候就开始在 Apple 2 上写程序，其实不是真正的 Apple2 啦，只是一台兼容机。“买房子，送电脑”，因为当时房子冗余，所以经常搞一些杂七杂八的活动，不知道他们（房地产商）在哪里搭上了一家山寨 Apple2 的厂商。这台 Apple2 长得跟真的一模一样，128K，这个数字当时说起来好骄傲哦。最开始的爱好就是打游戏，然后就是改游戏，改无敌不死之类的。

图灵社区：现在在写什么程序吗？还在不断学习吗?

我现在在做的主要是自己在改一些框架。如果要写什么东西一定不是用学的，而是先把需要做的事想清楚。比如如果要写框架，就先搞清楚 http，但也不用搞得太清楚。只要把人家丢过来什么，然后应该吐出去什么弄明白就行了。然后就可以开始写了，其实说穿了，你只需要知道 socket 编程就可以了，这个谁不会啊？但是这个过程中就会发生一些杂七杂八的事。关键是没有人会逼着我交稿，所以我就有足够的精力去把这些都搞明白。这不是在学。这就像是听音乐，一开始你可能听布鲁斯，后来你就会弹布鲁斯，会唱布鲁斯。这时候你可能忽然觉得爵士也不错啊。这能算是在学吗？或者只是在找新的玩意儿。因为我已经不以编程为生了，所以我一边写，一边看还有什么新玩具。

我有什么问题就去 IRC 解决（一种即时聊天的聊天室，按照不同话题分类）。去 StackOverflow 上问问题就要很认真，而我这个人又超不认真，这不太符合我的个性（笑）。在 IRC 上只要缠上什么人，然后给他发信息就可以了。只要他不幸看到了，一般都会忍不住回你的。

图灵社区：那你在未来有没有什么特别想做的项目?

我想写一个 midi 转简谱软件，因为现在没有什么简谱排版软件。这个用 LaTex 就不行了，虽然它可以应付很多场合，而且很多人都很崇拜 Knuth，听到 Knuth 的名字好像就要举起右手敬礼似的。但是很多人都只用简谱，而且用简谱的话不用管 do 在哪里，大多数人都没有绝对音感。简谱可以用来应付大多数人。

图灵社区：写程序和写音乐对你来说有什么相通之处?

学习的部分其实是整件事里面最无聊的，但是一旦你熬过去了，最终就可以说点什么了。搞这些东西最快乐的时候其实是，你弄出一些东西，而这些东西是你以前想都没想过的，这才是最爽的！但是做音乐的时候总需要开会，我很不喜欢这个部分。

写程序是一个你重新了解已经知道的东西的过程。我维护过很多语言的代码，也都是在什么都不会的情况下。写程序就是你诚实的交代，你心目中的事是一件什么样的事的过程。

对我来说，（写程序和写音乐）其实都是一件事，就是对一件我已经知道的事，找到一个新的说法。要找到一种方式来表达，（对于写程序来说就是）说得更加干净利落。这个时候要是找到这种方法，就是最快乐的事。也就是“找到一个新的搞法。”

图灵社区：说实话，你这样的跨界跨得还是挺大的，有没有想过自己为什么这么特别?

我小的时候就隐隐约约地感觉我能够做点什么，我真的不知道自己在自信什么，我的功课也不好。都是第二天考试，完全不知道

谢工（右一）和李盼（左一）在采访阿怪（左二）

要考什么，然后课本整本拿过来开始看。但是有一点我可以说的就是，我的阅读量真的很大。

我觉得自己主要是运气好，一直坚持做自己喜欢的事。在我还不知道自己能搞出什么名堂的时候，这叫做任性。“居然被我猜中了，我居然可以混出一点名堂！”你不觉得回想过去，会捏一把冷汗吗？如果我没有遇到杨导（杨德昌），如果我没有遇到当年提携我的张雨生老师，我其实就是一个自以为会写歌的数学补习班老师。我觉得近 20 年来都很少有比我更幸运的人。如果再给我一次机会，我真的不知道自己有没有那个种，再干一次。我当时入行的时候

其实不过是爱慕虚荣而已。

对我来说什么都是娱乐。其实写什么都一样，能不能赚钱就看上帝了。

“你这种表达方式和苦逼码农很不一样啊。他们一般就是学习、奋斗，练武功秘笈，最后就是：‘我终于炼成了！’”

“可是大侠最后都有泡妞的环节哦。”

“对于码农来说，这些都被剪掉了。”

“所以最后只剩基友了？”

不同

> “台湾信息管制？他敢管！他敢管我就敢把总统府黑掉，你以为我不会啊。”

图灵社区：台湾的老一辈会不会觉得搞 IT 的就是修电脑的？

其实这种事都赖自己啦，你可能是从国中开始就自告奋勇帮很多亲戚朋友修电脑了，活该啊！

图灵社区：内地的程序员和台湾的程序员在你看来，最大的区别是什么？

我从 2010 年开始在北京租房，因为租了房子，所以就要来住，否则多浪费。

我觉得大陆程序员可能还是要穷一点，生活压力也大一些。我感觉他们好像不太出来玩。在台湾，我自己会固定参加一个叫 hacking Thursday 的活动。我们管它叫的“嘿嘿，星期四”。在一个类似的活动中，你没事的话就可以来分享一些东西，有问题也可以来问一下。我有问题也会跑去问，比如说我刚装上 Debian 的时候就有很多问题要问，然后就会急着找人帮我解决实际问题。在那里也会了解到别人都在玩什么有意思的东西，比如说我也是在活动中听说了 Ubuntu，然后连夜搞来玩，跳槽过去，然后再跳回来。等自己知识累积得够多了，也可以去帮助别人。具体到“嘿嘿星期四”这个活动可能要更专业一些，地点都在咖啡店，一个月会有一次主题活动。但是本质是大家聚在一起乱哈哈。大家规定，不许写公司的程式。我们虽然经常跑题，但是这也是乐趣所在。经常会有一些艺术家来找我们，08、09 年的时候，有一些艺术学院开始搞 Aduino，其实就是装置艺术，需要响应的装置。他们说：“听说需要一种叫做 C++ 的语言。”我就说：“哦，这个我会一点，我查一下哦。”

我在北京发现的这种类型的聚会都是洋人，他们都在工体对面一家叫做 tram 的地方。这样的聚会就是聚在一起，然后放出风声，然后杂七杂八的人就都会来了。来了之后就会发生一些还蛮无聊的事（笑）。重点就是聚在一起玩啊。

记者 / 谢工、李盼

图灵社区：台湾的创业气氛是怎样的呢?

大陆的码农都很喜欢创业，不知道为什么。我们那创业的都是忽悠的人，写程序写得好好的干嘛创业呢？除非是我想要一个什么东西，实在是没有人做，而我又实在很想要它出现，我才会考虑创业。创业这种东西我觉得就是你看一样东西不顺眼到受不了了，就想：妈的，我来吧！剩下的那种创业就无聊死了。我感觉大陆程序员都还挺着急的，可能是看到身边有人一夜暴富。但是我不喜欢这样做事情。在台湾可能工作 5、6 年的程序员月薪会有两万人民币左右，更高的也会有。但是话说回来，钱哪会有够的时候呢?

图灵社区：大陆和台湾的开源项目在你看来有什么不同?

大陆其实有很多不错的开源项目，但是大家交流都用中文。要知道，台湾人、韩国人、日本人大家都很认命地摸摸鼻子，然后就写英文了，虽然大家的英文也都很破，但是这样至少意味着，法国人也可以帮忙给我们送 patch 啊。要是洋人实在看不懂，人家就不来帮忙了，多可惜。比如说我那时候帮忙人家写 LXDE（这是一个纯台湾的项目，轻量级的桌面系统），我也不能说自己的英文写出来多漂亮，但是我写的那种破英文，也会有法国人来帮忙。我觉得大家是不知道这个世界上，除了自己，也有其他好人。首先要做到的是信任，让自己可以被别人相信，然后要相信别人。

图灵社区：对现在的生活满意吗?

我觉得自己的命已经很好了，我可以看到怎么用算盘计算，也亲眼看到数字时代从无到有，就在我身边发生。我参與过进过坎城（戛纳）电影节的电影，那没有什么，只不过是拍一部电影而已。我参與过全亚洲卖 500 万张的唱片，那也没有什么，只不过是好好做唱片。重点是，它不是我的重点。这些都没有什么大不了的。就像拉斯维加斯的总统套房，我也睡过，没什么大不了的。我觉得最重要的是要做自己喜欢的事，而且要做得实在。

Objective-C 和 C++ 区别有哪些，为什么苹果会选择前者?

作者 / 池建强

十多年软件从业经验，先后在洪恩软件和用友集团瑞友科技工作，目前任职瑞友科技 IT 应用研究院副院长。喜爱苹果的产品和技术，在微信公众平台维护“Mac 技巧”，每天推送人文与技术。

图灵社区 ID：池建强

有人在学习 Objective-C 的时候，经常会有这样的感觉，这东西不就是加了面向对象的 C 吗，而且类的语法还那么古怪。这东西跟 C++ 有什么本质区别？为什么苹果会选择 Objective-C 而不是 C++？

这个问题要从 Objective-C 的语言特性谈起。Objective-C 是 C 的扩展，设计思路上借鉴了 Smalltalk 的面向对象和消息机制的思想。从我个人使用过的面向对象语言来看 Obj-C 是对消息传递支持的最彻底也最显式的。Objective-C 的类中定义的方法都是消息传递，而且类和消息之间是运行时绑定的，运行时编译器把消息发送转换成对 objc_msgSend 方法的调用。其它的 C++、Java、Python、Ruby 都体现得不明显，更倾向于对象的封装和抽象。

Objective-C 和 C++ 基本上是两门语言，没有太大的关系。Objective-C 本身是静态语言，编译后就是机器码，执行效率很高，但引入了很多类似 Python、Ruby 的动态特性，像动态类型推断，id，selector，block 等特性，所以又非常灵活。用惯了 Java 或 C++ 会觉得 Objective-C 的语法很怪，但如果你用学习新事物抛弃

旧思路的方式去学习这门语言的话，你会很快爱上它的消息式的编程风格，加上 XCode，无论是写 Mac 应用还是 iOS 应用，都会非常得心应手。

学习 Objective-C 不需要有 C 的背景，倒是学 Objective-C 的时候顺手可以把 C 也学了或温习下，由于 Objective-C 是 C 的超集，所以在 Objective-C 环境执行 C 程序毫无问题。

关于苹果为什么采用 Objective-C 的问题，说明一下，其实不是苹果采用了 Objective-C，而是乔布斯创建的 Next 公司的操作系统 NeXTstep 采用了 Objective-C 作为原生语言。

Objective-C 在计算机编程语言中有着不短的历史，80 年代初

Brad Cox 和 Tom Love 发明了 Objective-C，1988 年乔布斯的 Next 公司获得了这门编程语言的授权，并开发出了 Objective-C 的语言库和 NeXTstep 的开发环境。NeXTstep 是以 Mach 和 BSD 为基础，Objective-C 是其语言和运行库。后来的事大家都比较清楚了，苹果买了 NeXTstep，乔布斯回归苹果，NeXTstep 也成了 Max OS X 的基础。以后发展越来越好，Objective-C 反而成了苹果的当家语言，现在基本上是苹果在维护这门语言的发展。

随着苹果的 APP 帝国不断壮大，这门语言也得到了长足的发展，从 1.0 到 2.0，从面向对象的 C 语言扩展，到内存引用计数管理，属性管理，引入块的概念，实现自动引用计数，优化编译器，简化语法等等。Objective-C 在 2011 年和 2012 年分别获得了 TIOBE 评选的年度语言，目前排名第三。

苹果 CEO 库克在 2012 年的 WWDC 大会上宣布，苹果已经为全球开发者支付了超过 50 亿美金的分成收入，时至今日，估计已经远远超过 50 亿了，对于开发者来说，这是一门能够独立创富的编程语言。

还有一点不能不提，第一台万维网的 Server 就是一个叫蒂姆 · 伯纳斯 · 李的大牛在 NextStep 上写的，包括浏览器。所以，咱们得感谢 Objective-C，要不然还不知道互联网会发展成啥样呢 ..... ■

Objective-C 在其他平台也有应用吗?

作者 / Jiva DeVoe

拥有 25 年的软件开发经验，是专门开发 iPhone 和 Mac OS X 应用的 Random Ideas 软件公司的创始人，已有多个 iPhone 应用成为苹果广告中的推荐应用。此外，他还是 *Cocoa Touch for iPhone OS 3 Developer Reference* 的作者。他的

虽然到目前为止最好的 Objective-C 编码平台来自苹果公司，但它们绝不仅适用于苹果公司的平台。Objective-C 在 Linux、BSD 甚至 Windows 等其他平台都有相当久远的历史。根据具体需求，你会发现一些能很好地支持这些替代平台的开源社区。本文将简要介绍一些其他的平台，并告诉你在哪里可以找到更多关于它们的信息。

在其他平台上使用 Objective-C 时面临的最大的挑战在于对能使 Objective-C 变得强大的框架的支持。移植 Objective-C 语言是一件琐碎的事。由于 GNU 编译器集合（gcc）开始支持 Objective-C，Objective-C 在几乎所有 gcc 支持的平台都可用。但是，移植核心框架是一项更加艰巨的任务。

可以确定的是，Foundation 框架已经有最广泛的跨平台支持。而 Cocoa 和一些高层框架通常在其他平台上就不可用了。也有例外情况，不过老实说，如果你要在 OS X 之外的平台运行应用，在考虑使用 GUI 框架时就要特别小心。

博客地址为 www.random-ideas.net。

译者 / 林本杰

也就是说，能最好地支持通常和 Objective-C 一起使用的所有框架的主要有两个项目。它们是 GNUstep 和 Cocotron。这两个开源项目在移植性技术方面使用完全不同的方法，不过结果是一样的，即支持在 Linux、Windows、BSD 和其他平台编写和编译使用 Cocoa 和 Foundation 的 Objective-C 代码。

使用 GNUstep

这些项目中最古老的项目是 GUNstep 项目，它的历史可以一直追溯到最初的 NeXTstep 时代。事实上，这个项目最初的开发是为了提供一个闭源项目 NeXTstep 平台的开源替代品。为此，它很好地重建了 NeXTstep 的桌面环境，包括图标、文件浏览器、邮件客户端等。参见图 1。

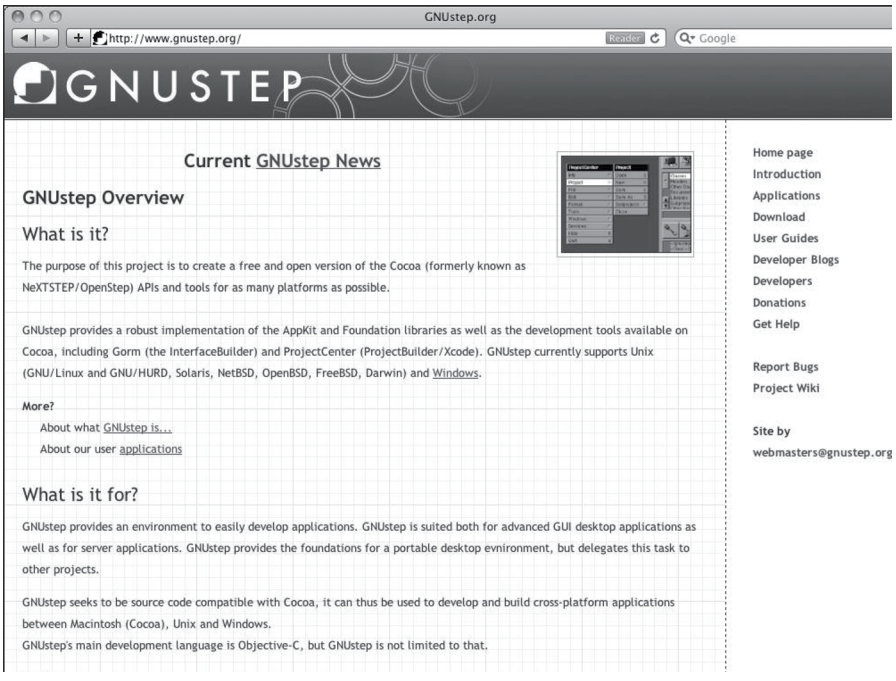


图 1　运行在 Unix 上的 GNUstep 环境

因为其久远的历史，GNUstep 项目对 Foundation 和 Cocoa 有一些最好的支持。不过，因为它们的真正的目标是复制 NeXTstep 环境，而不是模拟运行应用的 Mac OS X 或平台的原生组件集，而它们实际上包含了整个 NeXTstep 环境的组件集。这意味着如果你选择通过该项目将应用移植到 Windows，你的应用在 Windows 上运行时，看起来像一个 NeXTstep 应用。这包括所使用的菜单类型的方方面面。此外，还有一些和运行 GNUstep 应用所需的文件系统相关的问题。最后要确认的一点，为了在 Windows 上运行一个 GNUstep 程序，你必须安装完整的 GNUstep 文件系统以及众多支持库。

这一切可以让一个普通用户困惑。出于完整性考虑，在这里需要提到的是，过去几年间为 GNUstep 添加皮肤支持做了一些努力。这样你至少可以创建一个 Windows 应用，并使用 Windows 的组件集。换言之，GNUstep 致力于图形应用的可移植性，并不断改善，但是目前还说不上完美。总之，对于 Foundation 框架，GNUstep 的项目有一些到处可用的最好、最全面的支持。我想说如果你的应用是一个命令行应用，比如服务器，并且打算移植到 Windows 或 Linux，那么 GNUstep 项目将很可能可以帮助你实现这些。可以说，利用 Mac OS X 上已经写好的现成代码并移植到 Windows 的一个较好的方式就是利用 Objective-C 和 GNUstep 移植后台、底层非界面的代码。你可写一个原生的图形应用，通过进程间通信进行通信或者将它作为库来链接。这样做可以两全其美。一方面统一了业务逻辑代码，同时还能向用户提供熟悉的原生用户界面。

关于 GNUstep 项目的更多信息，可以访问它们的站点 http：//www.gnustep.org。

使用 Cocotron

另一个最近的跨平台 Objective-C 方面的尝试就是 Cocotron 项目的创建。在将应用移植到 Mac OS X 之外的平台方面，Cocotron 项目采用不同的方式。Cocotron 为 Xcode 提供了一个的交叉编译器环境，这样你可以在 Mac OS X 平台上 Xcode 里交叉编译你的应用。通过该交叉编译器，你可以编译 Windows、Linux 或其他 UNIX 桌面版本。以这种方式交叉编译的应用，观感和行为都和原生的应用一致。

Cocotron 的工作原理就是利用 Xcode 在编译代码时支持多个工具链和 SDK 的能力。为 iPhone OS 或者 Mac OS X 编译代码时，只需要单击一个按钮的技术使得 Cocotron 可以发挥它的威力。

如果你觉得将 Mac OS X 和 Xcode 作为开发环境最惬意，那 Cocotron 是一个完美的解决方案。它支持在 Mac OS X 上进行所有开发，然后简单地改变 SDK 并为目标平台重新编译应用。

这似乎还不够，Cocotron 还有一些对 Cocoa 和 Foundation 框架的最好的第三方支持。它的实现已足以开发同时部署在 Mac OS X 和 Windows 平台上的 Cocotron 商业级应用。

Cocotron 相对落后的最大领域在网络、线程处理和一些高层次的框架支持。不过正在积极努力中，而且如果你有足够的预算，你甚至可以和 Cocotron 的维护者订立契约来改善你的应用所需要的具体的库。

图 2 展示了一个使用 Cocotron 编写的运行在 Windows 和 Mac OS X 上的应用。

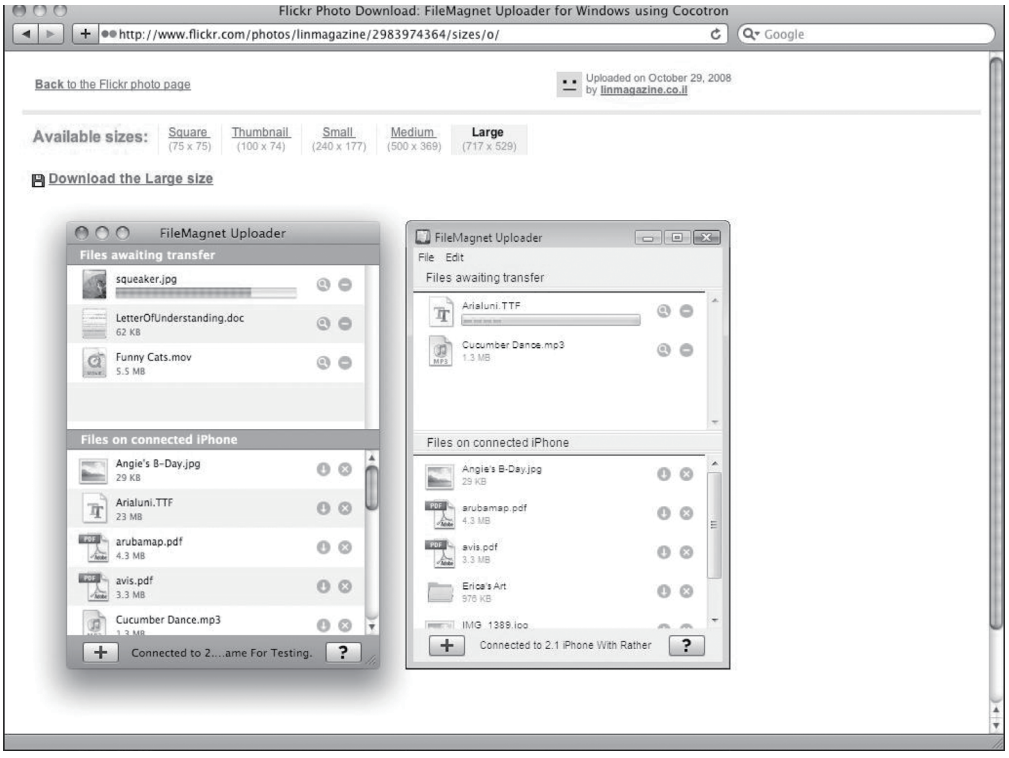


图 2　使用 Cocotron 编写的应用

如果你不方便将 Mac OS X 作为主要的开发环境，Cocotron 可能不适合你。不过，如果你的目标是通过尽可能小的改动将 Mac OS X 上的应用移植到 Windows 上，那 Cocotron 应该是一个好选择。最起码我会说，下载项目并尝试用它编译你的应用。用不到一天的时间，你就会知道你的应用是否是那些可以简单通过这个项目来移植的应用。

关于 Cocotron 项目的更多信息，可以访问它的站点 www.cocotron.org。

使用其他开源库

Cocotron 和 GNUstep 不是仅有的苹果 Objective-C 框架的开源替代方案。还有一些其他的实现，不过它们大多数局限于一定领域或不再维护了。我的建议是，考虑在 Mac OS X 或 iPhone 之外的平台上使用 Objective-C 时，评估一下我这里提到的两个项目，看它们是否满足你的需求，如果是，那你很幸运！

展望未来

我想写本书的一个原因是我觉得 Objective-C 是一个未得到充分重视的语言。它的“姐妹”C++、Java 和其他语言都受到更多的关注。我认为其中的一个原因是 Objective-C 通常只在 Mac OS X 上进行开发时用到。

随着 iPhone 和 iPad 的出现，Objective-C 得到了大量新的关注，而且学习语言的新用户出现了前所未有的增长。这有利有弊。

好的一面是因为这可以确保这个平台还会存活许多年，但不好的一面是它显露了这个优秀的语言在标准化和文档化方面的不足。

不像 C 和 C++，Objective-C 没有一个 ISO 标准组织来推动它的规范。有些人可能会说这其实是一件好事。事实上，这也许是 Objective-C 优雅而且非常适合它的任务的原因之一。然而，问题是由于缺少标准化，Objective-C 在它的核心平台之外很少使用。这意味着以后这个平台上新的开发者都被限制在 Mac OS X 或者 iPhone OS 上进行开发。

《好学的 Objective-C》作者是顶尖的 Mac 开发人员和专业作家。通过本书的详尽指引，即使是编程新手也可以迅速学会 Objective-C。本书全方位地介绍了 Objective-C，从基础知识到资深程序员所使用的高级技术等众多主题，其中包括内存管理、多个框架的结合使用、线程安全的技巧，以及 Xcode 的详细用法等。本文选自《好学的 Objective-C》。

没有人知道未来可能有哪些新的平台和硬件。学习一门语言，就是对这门语言和它运行所依赖的平台上进行智力投资。为了确保在 Objective-C 上的投资可以在未来获取收益，我们需要在核心平台之外推动 Objective-C 语言的发展。这需要得到 UNIX 和 Windows 等其他平台的更多的支持和采用。我不敢说支持 Objective-C 的 ISO 标准化，但是我肯定乐意看到第三方项目在跨平台支持方面的改进。

可以肯定的是本文向你介绍的一些项目就是以此为目标，这是非常好的，我们应该帮助并推动这些项目发展，尽管它们可能并不是在我们自己喜欢的平台上。让 Objective-C 存在下去并有一个支持跨平台的编程环境，有助于你在这里的投资在未来更有价值。

八 ++

当 Steve 遇上 Steve
——曾经打造苹果的少年们

作者 /Dan Lyons

Dan Lyons 是当今最知名的技术博客 ReadWrite 的主编。之前他在 Newsweek 做过科技编辑，并为《福布斯》编写科技报道长达 10 年之久。Dan 一直以来都致力于将他在商业、媒体、硅谷等方面的洞见都以轻松诙谐的方式展现在他世界知名的博客上。

译者 / 武卫东

沃兹（Steve Wozniak）回忆了他当年和 16 岁的少年乔布斯（Steve Jobs）的友情，当年他们怎样一起听盗版的鲍勃 - 迪伦歌曲磁带，以及苹果公司起家后的事情。他回忆起他们两人好几次闹意见分歧——但强调说他们从来没有当面争执或者争斗——而且为何他相信史蒂夫是死在幸福之中的。沃兹谈到了早年他和史蒂夫在伯克利的大学宿舍楼里挨个敲门兜售所谓“蓝盒子”，那是一种违规的电子设备，可以用来免费打长途电话。他还提到他们曾恶作剧地往梵蒂冈打电话，以及有一回两人被人抢劫，从劫匪的枪口下逃生。

Dan：史蒂夫早年的时候是个什么样子?

Woz：史蒂夫很有条理，很有业务头脑。我在的时候他从来不去尝试做计算机设计的事情。他认为不值得去做。从我们认识开始，他就找到了别的方向。我就是个自命不凡的设计师，过于超前，所以没有人愿意跟着我做事。

Dan：和史蒂夫是怎么想到要做第一个苹果产品的，就是 Apple I 电脑?

Woz：哦，乔布斯还不知道的时候，就已经有很多人看到了这个产品。我那时在 Homebrew 计算机俱乐部，而乔布斯远在俄勒冈搞公社活动，在一个果园里工作。我们当时没有什么联系。不过我灵机一动，想要促成这个革新。我们俱乐部的人都认为个人计算机会影响每个人的生活，我们认为每一个人都应该有一台小计算机，这个小东西上面有不少开关，还有些奇怪的数字，而人们都应该学会编程，从而操作计算机。我们没有想到它最后会是现在这样的形状。

我从来没有想到要运作一家企业。我那时在惠普有份理想的工作。每周我都会去参加俱乐部的聚会，我把我设想的 Apple I 电脑草图拿给大家看，根本没想到什么版权，啥也没想，就说：“嘿，伙计们，我这里有个便宜的方法可以造台计算机。”我会在一台电视机上做演示。

然后乔布斯从俄勒冈回来，来我们俱乐部转转，他发现了我的设计里蕴藏的商机。只有我这个方案做出来的电脑才是用户可以承受的。我们首先想到的是只做印刷电路板，我们花 20 美元的成本，做好后卖 40 美元之类的。我已经把电脑草图散发给大家了，不过史蒂夫觉得我们可以做一家公司。

那其实是我们的第 5 个产品了。我们一直是一人一半地合伙，我们是最好的朋友。我们先做了蓝盒子。下一个产品是我在一个保龄球场看到的 Pong 游戏机，于是我回来自己用 28 个芯片也做了一个 Pong。我那时在惠普是设计计算器的。史蒂夫看到了我的 Pong，就跑去 Atari 公司，给他们看，结果他们就雇用了他。他们是否以为他参与了设计，我就不得而知了，我也一点儿都不在意。他们给了他一份工作，让他上夜班。他们说他跟人相处不太好，他很是特立独行，容易与人发生摩擦，所以就让他一个人上夜班。

我们下一个项目是史蒂夫说 Atari 的老板 Nolan Bushnell 想要一个一人玩的敲砖的游戏。他说如果我们能用很少的芯片做出来，就能得一大笔钱。于是我们就把这个做了出来，从 Atari 拿到了钱。

Dan：有传言说史蒂夫在那笔交易里面骗了你一些钱。

Woz：传言说的没错。但这事对我不重要。我有工作，史蒂夫需要钱，去继续参加农村的公社活动什么的。因此我们做出了 Breakout 游戏机，那本来是个一人要干半年的活，结果我们四个昼夜就干出来了。我们那个设计很精巧。

再下一个我们一起做的项目是，我们见过有人用一个巨大的电传机连着一只的调制解调器，拨号上 ARPAnet。这个电传机贵得跟小汽车一样。用这种方式可以连到 12 所大学，以访客身份登录，

然后在远程的计算机上做事情。这在我看来真是不可思议。我知道可以拨号到当地的分时系统的公司，但是远程访问大学里的计算机真是令人难以置信。于是我回家自己设计了一个。我设计了一个视频终端，可以通过调制解调器访问斯坦福大学，而后进入ARPAnet，最后列出一串各大学计算机的清单。

远程的计算机可以在我的电视机上用字母对话。我做 Pong 游戏时显示的是拍子和小球，这时我则用了一个字符生成器。设计出的终端成本很低。我们把它卖给了一家叫做 Call Computer（呼叫计算机）的公司。这样他们就拥有了很便宜的终端，而史蒂夫和我则平分了这笔钱。

Dan：给我讲讲做蓝盒子的事情。

Woz：那是在反主流文化的日子里。我是反战的，史蒂夫则热衷于嬉皮士方面的东西。我不是为了挣钱才做这个东西，只是想做个东西玩玩，并不是为了节省电话费。我老实得很，不会去用蓝盒子打长途电话。除非我玩恶作剧的时候，我们把世界上的电话信号传送过来，让声音出现在身边的电话机里。我们确实搞过这种恶作剧。我会给巴黎的酒店打电话预订房间。我们在伯克利的学生宿舍里挨个敲门，推销蓝盒子。价钱是 150 元。

Dan：你们真的曾装成是亨利 - 基辛格给梵蒂冈打电话?

Woz：是的。我们在一个宿舍里给人演示怎么用蓝盒子。我打到意大利，然后要罗马，然后要梵蒂冈。我告诉他们我是亨利 · 基辛格，正在莫斯科的一个峰会。那个时刻在意大利是早晨 5 点半。对方让我 1 个小时以后再去电话。我后来又打去，跟一个主教通话，他说他刚跟莫斯科的基辛格通过电话。

我们还有一回遇到了持枪抢劫。是在桑尼维尔的一家批萨店。有两个家伙看样子像是对我们的产品感兴趣，我们把他们带到一个付费电话旁，替他们给芝加哥打个电话。他们很喜欢，想要我们的蓝盒子，但是没有钱。我们就要走回车上，结果他们掏出枪来，抵着史蒂夫的脸。我们就把蓝盒子给了他们，但他们不知道怎么用，就给了我们一个电话号码，让我们打过去教他们用。我冒出了一个主意，想告诉他们一种方法让警察逮住他们，或者让他们打成收费电话。最后我们也没有这么做。要能这样，哇，可真是有意思。不过你可不想去跟拿着枪指着你的脸的人周旋下去。

Dan：你们俩可真是我们想不到的一对儿啊。

Woz：我们非常相像。我们会在伯克利猛逛商店，想找迪伦的盗

版磁带。史蒂夫对计算机很感兴趣，他真的很想用新出现的微处理器造出一台电脑来。他想象有一天这些电脑可以替换掉那些大家伙，每个人都可以拥有自己相对便宜的电脑。史蒂夫以前在剩余物品商店干过，那里买进东西很便宜，卖出去很贵。有回他告诉我能用 6 分钱买进，我简直都惊呆了，因为我知道他可是按 60 块卖出的啊。他觉得这太正常不过了，我则多多少少不这么看。怎么能这么干呢？我可不能这么巧取豪夺。但后来我们开始做苹果公司，我也开始接受了这样的建议，就是说为了发展应该追求很好的利润。

史蒂夫迫不及待地要这么做。麦克 - 马库拉是我们的导师，他告诉史蒂夫他应该在苹果公司做什么角色，也告诉我应该在公司做什么角色。他是教我们运营公司的导师。他非常低调，远离媒体，所以不太出名。但他看到了史蒂夫的天份。热情，激情，让人能够成功的那种思维方式。他从史蒂夫那里看到了。

马库拉以前在英特尔做过工程设计和市场营销，他很看重市场营销。他认为苹果应该是一个营销驱动的公司。是 Don Valentine 介绍他给我们的。Don 来到我们的车库，我给他展示 Apple II 的功能，他说："市场有多大？"我说："有一百万台。"他问我为什么，我说："有一百万无线电爱好者，而计算机的市场要比无线电大。"我们不太能找准点。乔布斯和我都没有公司业务经验，我们也都没有学过商学的课。我们没有开储蓄账户，我们也没有银行账户，我在公寓里一直付现金，我只能付现金，开的支票都有问题，总被退回来。

Dan：史蒂夫比你更深地介入公司业务吧？

Woz：他足够理解技术，知道我是最好的设计人员。他知道这一点。他见过许多其他公司，知道我们需要一个理解技术的生意人。他

就是一个技术型的生意人。他不是工程师，不做任何硬件或软件。

Randy Wigginton 跟我说，早期你是给人留下最深印象的人，不是乔布斯。

哦，我算是个超级棒的工程师。惠普公司先后五次拒绝考虑我的搞个人电脑的提议。后来看到了 Apple II 时，他们却说这是他们见过的最好的产品。大家很推崇我的工程设计能力。可是我从来不想钱的事，我要是运作一家公司，一定是很糟的。我想做个好人，想跟每个人都成为朋友。不错，是我先想到搞个人计算机，但是我不想让大家认为是我改变了世界。我只想被当作一个工程师，用非常有效的方式把许多芯片连接在一起，而且写出了匪夷所思的代码。我希望大家认为我是一个伟大的工程师。我很感谢有史蒂夫·乔布斯在那里，因为你需要一个有市场意识的人，他自己坚信计算机改变了人类。我呢，我不是为一家公司设计 Apple II，我只是为我自己，为了炫耀。我看过苹果公司所有最近的产品，比如 iPhone，iPad，甚至 Pixar，可以说史蒂夫做的每样东西都得是完美无缺的。因为那就是他。他创造的每一个产品都是史蒂夫·乔布斯。推出的产品绝对不允许有缺憾。这也是他如此严格控制质量的原因。

Dan：你一开始不想加入苹果公司，是真的吗？

Woz：我最初是说不加入的。我思想上很单纯，一直说不想让钱把我腐蚀了。我不愿意从马库拉那里拿一大笔钱，却放弃自己想在最伟大的惠普公司当一辈子工程师的梦想。我跟苹果说不。我说我可以在业余时间设计计算机。马库拉说，你必须离开惠普公司。我说：“不，我不必离开惠普，我可以晚上干活。”然后我所有的朋友都开始给我打电话，说我应该去苹果。我的朋友 Allen Baum

说：“你愿意待在惠普做个经理，然后变得有钱，还是去苹果做工程师，然后变得富有？”这正是我需要听的忠言。我意识到我不需要做什么运营的事情，我不想管一个公司。我太没有政治头脑了，可能早就该被甩掉。史蒂夫的做事方式很得罪人。他总是凶巴巴的，凡事总想冒尖领头。我很安静。我没有那种多动症，可是我很多搞技术的朋友却有。我很平静，没有什么大的起伏。

Dan：你们二人有一阵子闹翻了吗？我听说后来你很不高兴，因为史蒂夫开始重视 Mac 而不是 Apple II。

Woz：哦，我也做过 Mac 的工作，我全身心地相信这个技术。我认为那是苹果的未来。但是我认为我们不应该切断 Apple II 这条产品线，对外从来不再提起。Apple II 还是我们最大的现金牛。那些搞 Apple II 的人感觉很不好，而不是我。不过，我还是站出来替他们说话了。史蒂夫没有跟我私下聊过这些问题，他就是不喜欢听我说的这些话。我们之间最接近争吵的一次是我 1985 年离开苹果后，开了一家公司做通用遥控器。我去找 Frog 设计公司做设计。史蒂夫有一天也去了那里，看到他们为我设计的东西，他把它甩在墙上，说他们不可以为我做任何事情，说：“你们给沃兹做的任何东西都属于我。”我是我自己的，而且我对苹果公司还是很友善，但是史蒂夫就在那里大发雷霆。后来 Frog 的人告诉了我。这是唯一一次我们之间有了争斗，但严格说也不是发生在我俩之间。没有人见过我们直接发生争论。

Dan：1985 年你离开去创办新公司的时候，你们还有密切的合作吗？

Woz：那时我们已经隔得很远了。在苹果的早些年我们很亲近。不过后来史蒂夫成为高高在上的生意人，而我还是个工程师。我

们在公司的不同位置，联系沟通就不多了。那时我们已经很不相同了。

Dan：但就算你离开后创办了新公司，你依然还保留着是苹果公司的员工。你现在还是苹果员工，对吧?

Woz：我每两周从苹果领 200 块钱。一点小意思。

Dan：你们最早是什么时候从车库搬出来的?

Woz：我们在车库里做出 Apple I，好像是从它开始销售起，我们因为没有给自己开工资，所以银行的账户里积累有了 1 万块钱，这够我们在库珀蒂诺租个办公室的了。我记得我们搬进去的时候，还没有得到迈克 · 马库拉的初期投资。我们买了些桌子，没有隔断墙。我们雇了个总裁，迈克 · 斯科特。我很喜欢斯科特。他可能会很严厉，但也有轻松的一面。他把苹果带到了 IPO。你可能未曾听说过他，但是他真的十分重要。他做了一个手册，里面满是我的设计等信息。我想让这个东西随这计算机发出去。乔布斯认为我想发布这样的信息让人们能够使用计算机，但其实，我是想让这样的信息告诉人们，计算机是什么，又是怎么建造的。我已经记不起有多少次在各种场合见到别的 CEO 们，他们就是看了那个手册,才进入计算机行当的。如果我们今天想把那个手册出版出来，我认为苹果不会让那么多细节的东西公开出来。

Dan：你还记得当年首次见到史蒂夫的情景吗?

Woz：我休学了一年去挣学费。做程序员的工作，然后我告诉公司说我知道怎么设计计算机。头头说：“你要是能设计，我就给你提供零件。”于是我就设计了一个简单的计算机，他们把芯片给了

我。我和另一个朋友 Bill Fernandez 一起做，在他的车库里。Bill 说：“你应该见见史蒂夫 · 乔布斯，他在我们中学，懂这些数字的东西。也爱搞点恶作剧什么的。”于是史蒂夫就来了。我们先交流了各自搞恶作剧的经验，然后又开始谈到音乐。我们谈到迪伦，念他的歌词，做点分析。我们都认为迪伦比披头士重要，因为他的歌词有内涵。迪伦很严肃，不仅仅是娱乐。后来我们一起去听迪伦的演唱会，还会一起逛音乐商店，找迪伦的盗版磁带。我们找到了些小册子上有迪伦的访谈，就开车去圣克鲁兹找那个写访谈的人。他给我们看了一些我们没见过的迪伦的照片，还一起听了一些我们没听过的迪伦的音乐。

史蒂夫的生活方式是那种年轻嬉皮士式的，一无所有地过，也没有存款。我一直有工作，有收入。但是我很羡慕嬉皮士生活。那时正是越南战争期间，我对当权者毫不信任。这跟史蒂夫的嬉皮

士哲学很相符。我们崇敬那些抗议的学生。我们有很多共同点。人们都以为我们很不相像，但其实不是的。对史蒂夫而言，他在逐渐成熟，而我却没有什么长进。他承担了企业的责任，而我一直只想做个年轻人，享受生活的乐趣。等到人死的时候，应该死在幸福之中。对有些人来说，幸福就是取得事业的成功，与人争权夺利，在电话里冲别人发脾气，这些让他们感到幸福。史蒂夫是一个很严肃、有能力、训练有素的人，我则还是个心态年轻、没有约束的人。我很享受这个状态。

Dan：史蒂夫被挤出苹果公司后那几年发生了什么，使得他回来后成为了这么成功的一位 CEO？

Woz：我认为史蒂夫学会了自律。不仅是个人的自律，而且也有公司的自律，如何让公司实现目标。在他离开苹果的时候，他还是只相信自己的判断，而忽视别人的判断。离开苹果的时候，他心平气和地跟我说过些话，告诉我他要另外成立一家公司，因为他觉得自己生命的意义就在于创造出伟大的计算机。

几年以后，Pixar 推出了《玩具总动员》，史蒂夫告诉我说，所有其他公司也在做动画电影，但关键在于故事好不好。问题是一般人着实不知道哪个故事好，而他却知道。这就是远见。他召集了一帮天才，让 Pixar 出品的所有东西都能取得很大的成功。我不知道为什么史蒂夫死后大家都在念叨“苹果，苹果，苹果”，因为 Pixar 也很出色啊。何况那是个完全不同的领域。我总是更喜欢把史蒂夫和迪士尼去做比较，而不是和爱迪生比。

Dan：你认为史蒂夫是死在幸福中吗?

Woz：当然啦！假如他坐起来说：“我年轻的时候的梦想就是如

此。”那么，他已经百倍地实现了他的梦想。我认为他离开苹果公司的时候是不愉快的，但显然他死时是很幸福的。他现在已经成熟，领悟到了很多东西，明白了很多事理。能够亲眼看到这一天是很幸运的。任何人死的时候，都希望把该做的事情做完。

查看英文原文

查看本文完整版

图书平台化

——《东京艺术空间》电子出版实践

作者 /Craig Mod

Craig Mod 是一个独立作家、出版人和设计师，居住在加州湾区（也经常在东京）。他 是 MacDowell Colony 的写作会员，获得 2012 年的 TechFellow，2011 年时候在 Flipboard 做产品设计师。他的作品发表在新科学人，纽约时报，Codex 等出版物上。

让时光倒流到不久之前：那时出版业的境况还不错，Kindle 还没有火起来，EPUB 是东伦敦一间酒吧的名字，“mobi”指的是一种鲸鱼；那时我们会抱着厚厚的书，炫耀给周围人看；在博客上连载吸血鬼故事的女孩儿们还没被出版商发现，英国的家庭主妇们还没有机会看到暮光之城的同人小说；而我们也还没用上 Flipboard 或是 Zite。就是那个时候，2010 年，iPad 出现了，于是 Ashley Rawlings 和我在 Kickstarer 上开启了一个募捐活动，希望通过一本名为《东京艺术空间》的书来唤醒人们对生活的知觉;感谢上帝，我们真的改变了一些事情。

当我们在 Kickstarter 上募捐的那会儿，2 万 5 千美金绝不是个小数目。而现在如果你募集不到 10 万美金就几乎可以算作失败，不到 100 万美金都不好意思给别人说。在“众筹”的概念飞速发展的同时，电子出版也很快成为了图书售卖的主要渠道。“众筹”领域的变化显而易见，然而出版业变革中的成果和不足却几乎鲜为人知。

平台

在过去的两年中，一个简单却极具说服力的事实逐渐显现：图书的未来依赖于网络化的平台，而不是孤立的平台。相比于人机界面、导航、排版等表面的进步，平台的进步在更大的程度上决定了阅读的未来。平台造就了生态系统——包括生产、消费和分配——也对生态系统内电子图书和出版的所有重大变革产生了影响。图书和出版的后期消费阶段不仅仅意味着屏幕上的文字。

作为我们 Kickstarter 募捐活动的一部分，我们承诺推出电子版的《东京艺术空间》。最终我们如期交付了作品。我们希望（而且也是这样做的）充分展现日本艺术，而不是往你的电脑里塞几个文件就了事而已。

现在就让我们着眼于平台，回顾下过去的两年里所发生的事情，同时了解一下我们如何发展为今天这样。

电子版的《东京艺术空间》

我们发布了多种格式的《东京艺术空间》电子版，覆盖了多个平台。主要可以归为如下两个不同的生态体系：

- 开放系统（互联网）
- 封闭系统（iBook，Kindle 以及其他阅读器）

互联网

这本书需要一个大家都能访问到的页面，也就是给其所有内容提供一个公开的网上地址，我们选的是 http：//read.artspacetokyo.

com。整本书都在这个网站上，包括所有的采访、随笔和艺术空间的相关信息。你可以通过网址访问到所有内容。

为什么要这样做呢？因为我坚信电子图书会从这样的公开页面上获益。这一代的读者习惯于分享文字，如果你阻碍他们分享和获取电子信息，那你的书其实就像根本不存在一样，至少阅读它的人会更少。

我还认为在网上公开书的所有内容，能让这本书（电子版和纸质版）卖的更多。指向这个网站的链接会急速增长。read.artspacetokyo.com 是互联网上有关东京艺术的文献最多的网站。搜索流量也会相应的增加，再通过给每个页面加入销售链接，付费用户的转化也会相应增加。我们会及时通报相关数据。

虽然被称为“书”，但一样能做的像网站。在 read.artspacetokyo.com 上，我们特意没有遵循纸质版严格的线性排版，希望以更有机的方式无限扩展书的内容。

下面是网站在 27 寸显示器上的样子：

同样的网站，在 iPhone 上效果是这样：

其他细节：

- read.artspacetokyo.com 为移动设备做了专门的优化——这一点对于像《东京艺术空间》这样的手册来说尤为重要。
- 如果你在 iPhone 或 iPad 版上点击“添加到桌面”按钮，那么桌面上就会出现一个高清图标，点击图标后还有高清的启动画面。
- 通过一个简单的 JavaScript 库，能够让用户始终停留在“应用”里边而不跳出，即使有页面间跳转的链接也是如此。
- 我们在纸质版、pdf 格式、epub 格式和 mobi 格式的书中统一采用 Fedra Sans 字体，这多亏了 Typotheque 的 @font-face 给与的技术支持。

这就引出了一个重要的问题：

如果这些都是免费的，我们还怎么赚钱?

答案是平台和高级版产品。

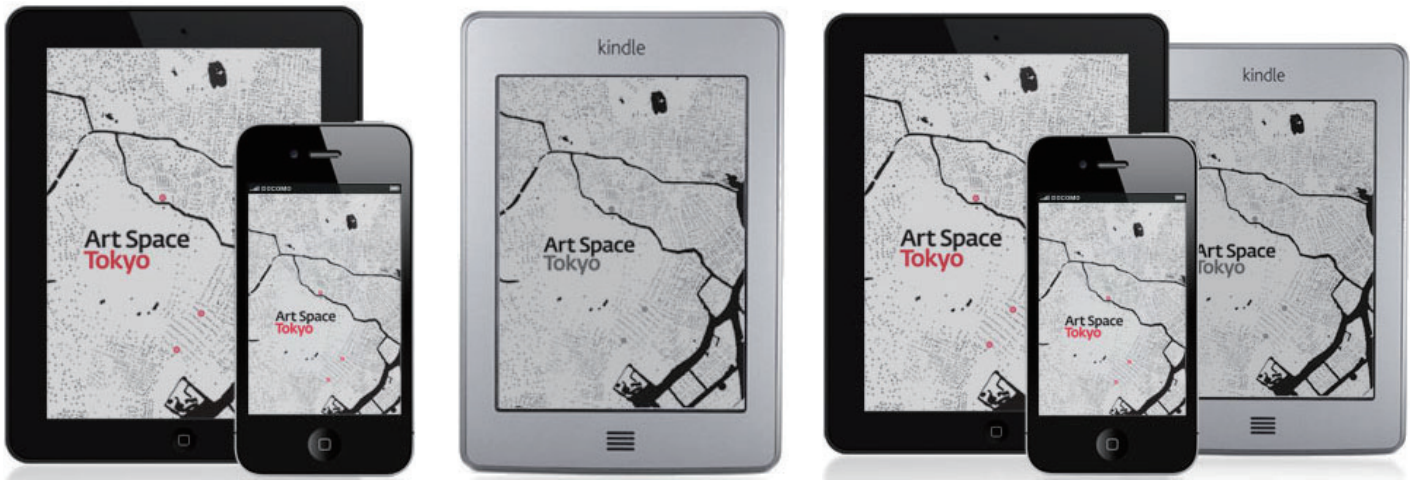
平台和高级版产品

我们的基础高级产品是纸质版的书；同时，我们认为购买依附于某个平台的（或者是无 DRM 限制的电子版，可以再任意平台上使用）电子版也有很大的价值。

举例来说，从亚马逊上直接购买一本 Kindle 电子书，就意味着同时获得了 Kindle 平台的所有好处（笔记保存和分享，社区高亮等）。书与平台的这种联系有实实在在的价值，而且会随时间流逝愈加有价值，值得为之付钱。

Kindle，iBooks 和 PDF

《东京艺术空间》可以在 Kindle、iBooks 和 Nook 上购买到官方版，



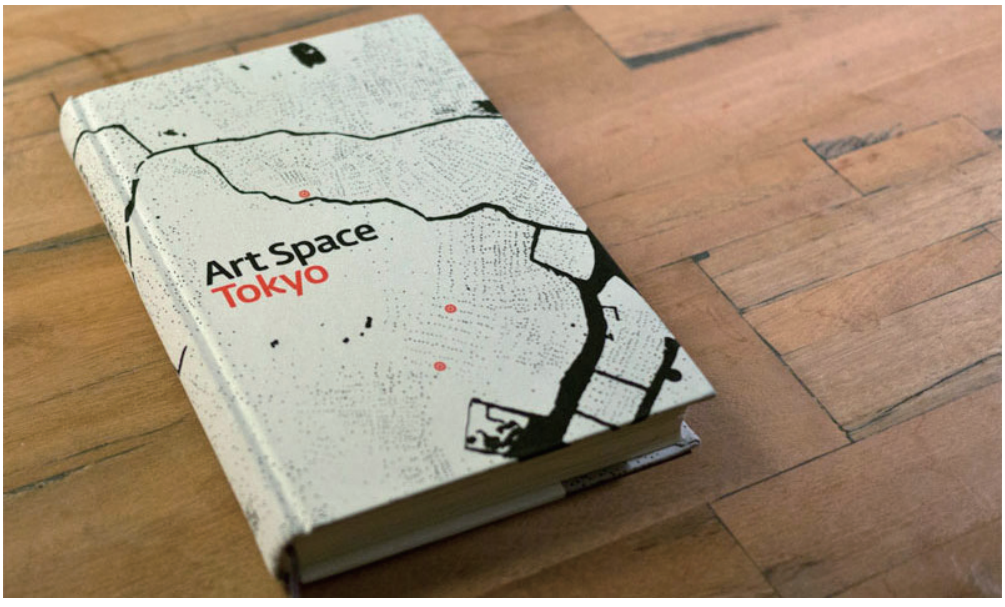
各个平台上都是 8.99 美元。如果你想在其他平台上阅读，我们也提供了一个多合一的数字包，里边包括了 mobi 格式、epub 格式和 pdf 格式的电子版，都没有 DRM 限制。这个数字包的价格是 14.99 美元。

纸质版 + 电子版

如果你购买纸质版，请告知我们，我们会免费赠送你一份电子版合集。当然，如果你之前购买了纸质版，同样也可以索取免费的电子版。给我们发邮件就行：support@prepostbooks.com

历程

当我们决定要做“iPad 版”时，态度非常明确，那就是绝不做成应用的格式。

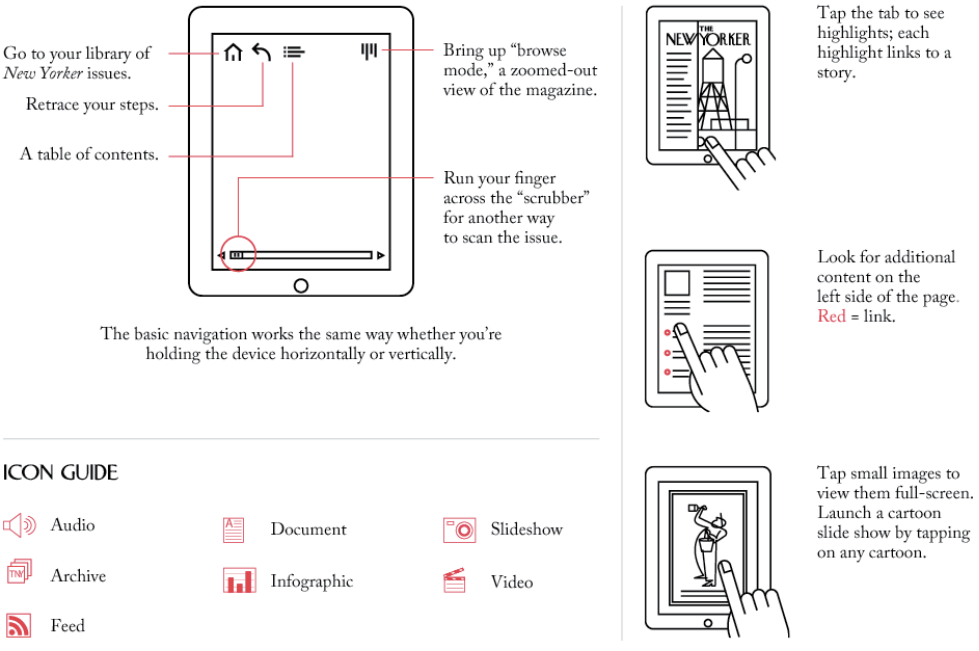


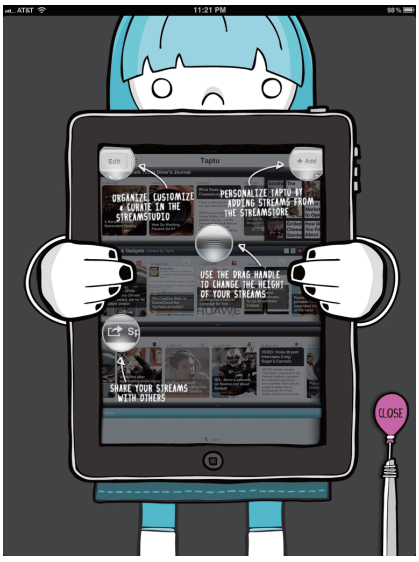
回顾一下：2010 年 5 月份 iPad 刚刚发布，还没有在 iPad 上出版书籍的成功案例，那时我们就对将图书做成应用格式的做法有很多疑虑，尤其对于本来有纸质版的书来说，更是显得过重过复杂。所以我们决定先集中精力做好 Kickstarter 项目的纸质版部分——修订、印刷和出版精装版本。

全力工作几个月之后，我们发布了精装版，看起来相当不错，我们的支持者也很满意。之后我们用了一个月的时间总结了项目期间收获的所有经验，并为社区反馈了一份完整的总结：Kickstartup。Kickstarter 甚至还给我们颁发了“最佳展示奖”。

开发应用？

2010 年秋天来临了，我们在调研为《东京艺术空间》开发应用的过程中，越发地觉得不靠谱。再“简单”的应用做起来也不容易，可以





说是很难。而无论是简单的还是复杂的，当时几乎没有一款产品做对。

这个时候，几家主流媒体发布了其各自的杂志应用，连线，流行科学，纽约人 是最早的一批。这些应用相当让人失望，文本都是图片格式，而且应用都很大，各个章节的下载过程十分别扭，总之对于阅读者来说相当不友好。这显然不像是数字阅读的正确方向，更别说我们的 东京艺术空间。

更糟糕的是导航，让人十分困惑——每个应用的导航都不同。阅读纸质版杂志时手动翻页的操作就很简洁易懂，可在这些应用追求“创新”和数字化的过程中，这种简洁和优雅都丢失掉了。

事实上，我们看着这些奇怪的杂志，越发地意识到 PDF 也许是个更好的主意，至少 PDF 是“真正的文本”，可以搜索，也能压缩地更小，导航也更容易使用：不断翻就是了。

简单的 PDF 文件

于是我们开始用 InDesign 折腾 东京艺术空间 的 PDF 版，包括横版和竖版。应该说是成功了，就是字儿有点儿小，略显笨拙，但你猜怎么着？我们的这个版本比市面上看到的大多数独立应用的用户体验都要好。

用 PDF 有什么好处呢?

- 没有开发成本。
- 文件相对较小。
- 在所有 iOS 设备上的 iBooks 中都能阅读、检索。
- 文本格式而不是图片。

如果你把 PDF 扔进 Dropbox 中，你就获得了一个完全跨平台、跨设备、始终同步的电子书。当然，这么做是有那么点儿不优雅，但仍然比做成应用要好。

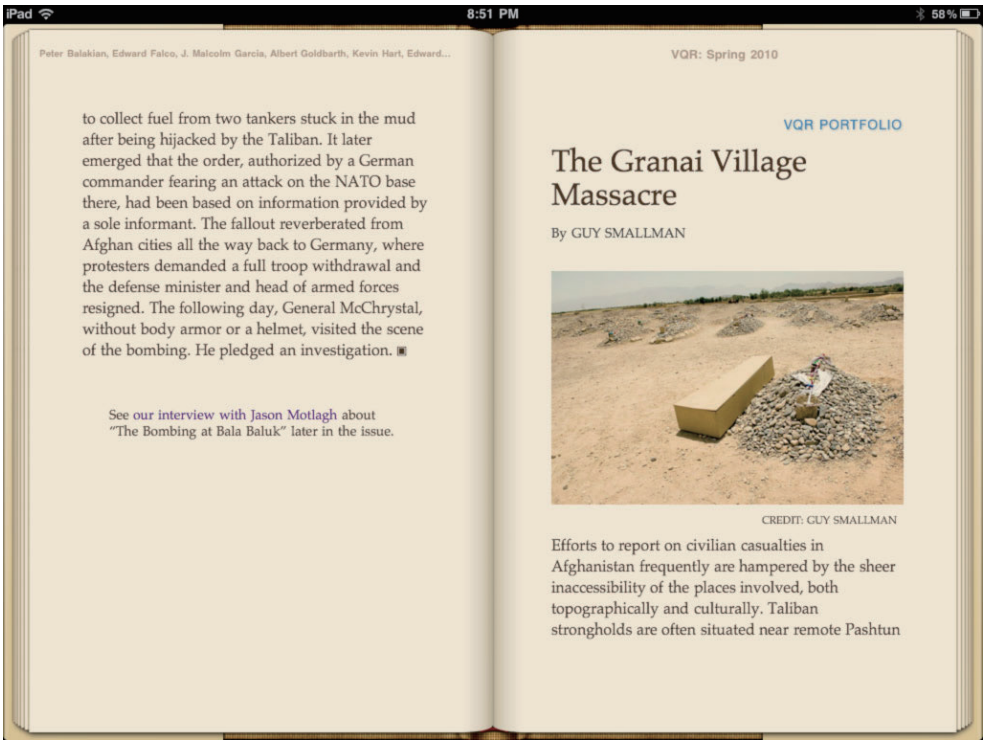
当然，作为在 iPad 上阅读的一个很常见的方式，PDF 不值得在这里再讨论，更别说作为 东京艺术空间 的“iPad 版”发布了。我想更深入地说一说。

EPUB

EPUB 第一次惊艳到我，是 2010 年我看到 waldorf Jaquith 的作品的时候。

我就不多说 iBooks 什么了，它实在是太过拟物化了——仿真的纸感和翻页、不成熟的排版、缺失连字号。我压根儿不想碰它，直到 Waldo 的 Virginia Quarterly Review 改变我的观点。他用了很大精力，制作了到目前为止我看到的最让人振奋的 iBooks 作品。

这本书的分页和标题都恰到好处，层次清晰，配图得当。当然，VQR 很简单，与应用程序的视觉复杂性比起来更是不值一提。但在他的排版中，却能看到一些特别的东西。更进一步说，这本书在公开标准之外，没有添加任何过头或累赘的东西：这两点在其



他任何杂志应用中都不曾有过。

同时，EPUB 3 出现了。Bill McCoy 在 2010 年的 Books in Browers 大会上，介绍了即将发布的 EPUB 3 标准。电子图书和网页设计慢慢开始融合。HTML/CSS/JS 即将被加入 EPUB 3 的核心部分。对于数字内容的消费者来说，这无疑个好消息；尤其是像我们这样会做网页的人。

电子墨水

不久之后，在 2010 年 11 月份我购买了第一台 Kindle，而且相当喜欢。即使现在，Kindle 的电子墨水也是我最喜欢的技术之一，也是我用过的最优雅的科技产品之一。当阅读的时候，你注意不

到它的存在；在全球各地都能连接网络；电池几乎可以永久续航；总之相当梦幻。

我的注意力主要集中到这两种技术上：iBooks · EPUB 3 组合，以及优雅的 Kindle 与其预期的 Kindle 生态链组合。

图书平台

2011 一整年，iBooks 逐步改善了其对复杂样式、自定义字体和连字符的处理效果，而且整体上的设计开始比 Kindle 更加友好。另一方面，Kindle 也逐步证明自己是一个真正的阅读 平台，有跨设备的整合、网页的效果、社交化的阅读，以及创新的借阅和分享功能。

通过遵循 EPUB 标准，并在这些已有的、尚好的平台上进行开发，你可以免费获得很多好功能。

- 他们都是真正的、完整的阅读平台。
- 其本身都主要是用来售卖、消费书籍类内容的。

- 他们都基于真正的文本内容：而不是图片格式的文本。
- 这些平台上的所有书籍都有标准化的交互体验。读者不用反复学习如何阅读。上 - 下 - 左 - 右？不用多虑：尽管滑动手指就行了。
- iBooks 和 Kindle 都在努力营造一种“图书馆”或阅读空间的心里模型，而不是“添加另一个应用程序”的感觉。
- Nobumasa Takahashi 的作品在 iPad，iPhone 甚至 Kindls 上看起来效果相当都赞。
- @font-face 的支持：我们获得了 Typotheque 出色的 Fedra 字体的使用许可，在纸质版和电子版之间保持了一致的 东京艺术空间 的品牌感觉。

iBooks 独有的：

- 页面布局更复杂一些，因此我们可以在章节开头、作者简介、采访等关键部分添加很多特殊效果。结合 Typotheque 的字体，我们的电子书显得相当漂亮。

Kindle 独有的：

- 从亚马逊直接购买所带来的认可度、信任和便捷。 读者高亮的部分和笔记都会被添加到《东京艺术空间》的 Kindle 语料数据库中。
- 随上一条而来的一些社区资源，比如“高亮最多的段落”。
- 如果用户的高亮和笔记设置为公开，那么他的标注和阅读状态会广播给 kindle.amazon.com 的粉丝。
- Kindle 电子书可以在多个设备上使用——包括 iOS 和 Android。只需购买一次，便可随处阅读。
- Kindle 用户在 Facebook 和 Twitter 上分享的高亮内容都有单独的展示界面，这就创造了一个 分享 - 查看 - 购买 的良性循环，

译者 / 张重骐

北邮人。学过设计，技术在学；前产品设计师，现任开发工程师。热爱美好，努力成为创造美好的人。图灵社区 ID： zhongqi

而这在目前的 iBooks 系统中是没有的。

在你获得这些好处的同时，你要放弃些什么呢？嗯，比如说就需要放弃很多的交互特性，你的出版物必须遵从“翻页”模型（而不能是“滚动”模型）。对于书的视觉语言，你的控制力也很小。（对除 Fire 以外的其他 Kindle 设备来说，几乎就没有任何控制力）

对一些书来说，这些缺失的特性至关重要。如果你的书讲故事的方式异于传统，那无论如何都要做一个应用。而《东京艺术空间》是一本相当“正常”的书，在写作之前就期望其有一种传统印刷图书的气质，而书的具体结构是在这个前提之下才逐步确定的。书中指向外部资源的链接能够增强“交互”的感觉。因此对于我们这本书来说，iBooks 和 Kindle 平台所带来的优势要超过其固有的不足，更别说自己设计、开发和维护一个独立应用程序的不足了。

我相信对其他很多书来说也是这样的，因此我很乐观地认为，我们制作《东京艺术空间》的经验可以作为其他同行的模板。而对于已经这样做了的同行，我希望这能让你更确信自己的选择。

查看英文原文

查看文章完整版

鲜　阅

Trip Hawkins：离开苹果之后，我创立了 EA

1982 年，特里普·霍金斯（Trip Hawkins）辞去苹果公司营销总监的职务，创立了美国艺电公司（Electronic Arts，下简称 EA）。经营至今，EA 已发展成为世界上最大的电子游戏发行商之一。EA 推出过很多奠定了电子游戏行业基础的游戏，比如 Pinball

作者 /Morgan Ramsay

Morgan Ramsay 是一位有着十余年经验的企业家和商业领袖，他擅长把小公司培育成伟大的公司。Ramsay 现在是娱乐传媒联合会的创建人和主席，该联合会是视频游戏产业的企业家，C 级总裁，以及高级经理的唯一联合会。他现在的工作就是致力于推进这个产业的进步。著有访谈集 Gamers at Work.

译者 / 张玳

Construction Set2 和 M.U.L.E.3 ，也推出过很多销量第一的游戏，比如《模拟人生》（The Sims）、《摇滚乐队》（Rock Band）和《麦登美式橄榄球》（Madden NFL）。此外，EA 的很多早期雇员也是该公司不断升值的财富，他们现在是公认的电子游戏业的领军人物。

1991 年，霍金斯制定了自主制作游戏平台的战略。这个战略催生了一家叫做 3DO 的独立公司，它成立之初是开发硬件平台，后发展为第三方游戏开发商。尽管开始受到欢迎，但 3DO 最终没能在竞争激烈的游戏机市场上稳步立足，于 2003 年宣布破产。霍金斯并没有为失败所困，就在同一年，他创立了 Digital Chocolate 公司，开发用于移动设备的游戏。今天，Digital Chocolate 成为了手机游戏界的领头羊，旗下的游戏涉及社交、电脑、游戏机等多个类别，包括 Millionaire City 和 MMA Pro Fighter 等热门作品。

兰姆塞：跟我讲讲你创立 EA 之前的事情吧。

霍金斯：当我还是个孩子时候，就发现自己有创造的天分，而且很爱玩游戏。当时我最喜欢的是那些使用棋子、纸牌和色子之类的模拟游戏，比如 Strat-O-Matic Football、Strat-O-Matic Baseball、《地下城与龙》和 Avalon Hill 的历史战棋游戏等，因为这些游戏模拟的主题我都很喜欢。我甚至还用纸牌、图版和色子设计了一款自己的足球模拟游戏。然而我很多朋友都觉得这游戏太复杂了——玩家必须像电脑那么聪明才能赢。这个游戏也算是我一个小小的商业败笔吧，但是我很喜欢这种创业的感觉。我发誓一定要再来一次，但下次要准备充分一点。

那个时候正是电视的黄金时期，所以朋友们都跟我说，和玩游戏相比，他们更愿意看电视，因为看电视不费脑子而且画面也很真实。

美国著名智库公司，为军方、政府及其他公司提供研究和情报分析等服务。

1971 年的时候，我接触了电脑组件和分时共享的概念，于是想若把游戏放到电脑里，就能在屏幕上显示和电视节目一样漂亮的画面了。从那时候开始，我就一直都在有意识地为 1982 年创立 EA 准备着。

1973 年的时候，我用了两年的时间，说服了哈佛大学让我从事自己感兴趣的领域“理论及实用博弈论”的研究，在这过程中，我用计算机来模拟各种社会问题，比如组织决策和预防核战。1975 年夏天，我在加州圣莫妮卡的一家智库公司 System Development Corporation 找到了一份工作，它是从兰德公司拆分出来的，主要做电脑软件。一天，一位同事吃完午饭回办公室之后跟我说，他刚刚去了一个商店，那里在出租 KSR-33 电传打字机终端，一小时只要 10 美元。我高兴地说：“太好了！家用电脑的时代快来了。我以后要做电脑游戏，让人们可以在家里用这种电脑玩游戏。”

我这位同事当时去的商店是迪克·海瑟开的一家叫做 The Computer Store 的商店，后来我们才知道这家店是世界上第一家电脑专卖店。这位同事还告诉我，有一家叫做英特尔的公司刚刚推出了世界上第一块 CPU 芯片。之后，我花了一个小时来整理思路，展望了一下这些技术将如何发展为家用。最后，我认为应该等这些技术再发展一下，硬件再普及几年，到 1982 年，我应该能建立自己的游戏软件公司了。后来我就按我想的做了。

兰姆塞：在开公司之前你做了些什么？

霍金斯：我完成了学业，1978 到 1982 年间又在苹果公司工作了一阵，实现了我另一个目标。这个目标就是在为其他公司工作期间学习如何建立自己的公司，同时，我还希望能够建立起家用电脑的供应渠道，因为这些买主将是我未来的顾客。在我加入苹果

的时候，公司只有 50 个员工。

从 1971 到 1982 的九年时间里，我逐渐完善了我对 EA 的设想。就这样，我存下了不少好点子。我是一个有创新精神的人，但是我也相信“所谓创新就是把旧有的东西用新的方式重新组合一下”。我在苹果公司的经历真是帮了大忙，让我把从消费电子产品身上学到的东西应用到了我们这个新兴的电脑行业中来。

兰姆塞：能举个例子说说吗?

霍金斯：比如说，直销模式的战略优势，也就是说，去掉中间分销商这个环节。这后来成为了苹果公司成功的关键因素。1980 年的时候，我发现，软件开发人员和工程师们就像是艺术家，甚至歌剧中的女主角，所以，我就把好莱坞媒体作为重点参考对象，尤其是唱片和图书。我研究了这些行业，然后发现，控制零售并建立销售渠道的战略价值对他们来说更加重要。

除此之外，我还决定把唱片录制的工作流程引入到游戏开发中来，这样，开发人员就有了更强大的工具，我们也能更轻松地在多个计算机平台上发布软件。我还首次把“制作人”、“督导”和“旗下品牌”等唱片界的概念引入了软件开发之中。我甚至在 EA 最早的一些游戏中模仿了唱片特定的包装形式，80 年代，这种包装形式还曾被 22 家竞争对手模仿。这些战略核心是 EA 创立的最关键准则。

兰姆塞：你是什么时候创立 EA 的？资金的问题又是如何解决的?

霍金斯:我自己出资,也没有合伙人,1982 年 5 月 28 日自己去注册。刚开始的时候，我一面在家里办公，一面寻找着美术师、销售渠

道合作伙伴、顾问和其他人才。但我是在那年秋天才开始招人的，我用了唐·瓦伦丁的风投公司的一间办公室，这是我和他在当年早些时候达成的协议。

公司的所有经费都由我出，包括电脑、租金和头十二个雇员的工资。直到 1982 年 12 月，瓦伦丁和其他一些风险投资商才为 EA 投了资。1983 年 5 月，我们推出了第一批游戏。就这样，EA 开始正常运转飞速发展了。

兰姆塞：在这个行当里，很多企业家都是团队创业，你呢？ EA 还有没有其他的联合创始人？

霍金斯：没有，就我一个创始人。公司创业要么是一个人先建立公司，然后招进最初的那些雇员，要么就是一群联合创始人协力建立公司，大家承担同样的风险。一个公司不会既有联合创始人又有创始人。

每个公司都需要雇员，但是早期雇员并不能算是联合创始人。我创立了三家公司，但 3DO 没有早期雇员号称自己是联合创始人。在我离开 EA 之后，有些人希望能把自己定位为“联合创始人”。

但事实是，我一个人花了十年的时间来独自完善公司的基础理念。公司第一年的运作经费完全由我承担。注册公司的时候，开设第一个办公室的时候，都只有我一个人。我亲自招聘了最早的那批雇员，给他们发薪水，有时候我还以个人名义贷款给他们，这样他们就可以买股票了。

兰姆塞：能举个例子告诉我你是怎么解决出问题的项目的吗？

霍金斯：年轻的开发者们会犯很多错，而且还常常不能按时完成任务，但是他们都非常有激情，动力也很足，只要把方向给他们引导好，他们就能发挥出最大的潜力。当时，我设计了一款叫做《一对一》的篮球游戏，然后找了一位叫埃里克·汉蒙德的开发人员来做。他当时做得很费劲，所以，在游戏完成前的几个月，我们把他从原居住地调到了我们公司的所在地，还在我座位的旁给他弄了个隔断，让他在那里工作。

兰姆塞：哪些游戏算是你的第一桶金?

霍金斯：EA 有六个首发游戏，分别交给不同的人员和公司来开发。在这些开发人员和公司里头，最顶尖的是比尔·巴吉、Free Fall Associates 公司和 Ozark Softscape 公司。EA 历史上最重要的一个游戏 Pinball Construct Set 就是比尔做的。Free Fall Associates 做了 Archon，直到今天我仍然很喜欢这个游戏。Free Fall Associates 是由琼·弗里曼和安妮·韦斯特佛夫妇以及另一位优秀的游戏设计师保罗·里奇一起创建的。Ozark Softscape 是应我的要求成立的团队，专攻简单的商业模拟类游戏，我给他们提供了设计这些游戏的经济学基础理论，比如进行生产活动时的学习曲线理论等。

我还为 M.U.L.E. 撰写了游戏手册，M.U.L.E. 是我至今仍然热爱的游戏之一。在天才设计师丹·邦顿的带领下，Ozark Softscape 为 M.U.L.E. 加入了很多既有趣又有创意的设计，把它打造成了一款富有魅力的经营模拟游戏。Pinball Construction Set、Archon 和 M.U.L.E. 这三个游戏都赢得了很多的奖项，而且，它们都是我们 1983 年的 5 月推出的首批游戏作品。

兰姆塞：为什么 EA 会上市?

霍金斯：在经历过 1980 年苹果公司上市之后，我创立了 EA。当时我就抱着一个想法，那就是把这家公司做好，好到可以上市。我的计划是找风险投资，然后每个员工都能占股份，而且我明白公司必须要有融资的渠道才能做大。我在整个 80 年代都和投行保持着很好的关系，而且早在 1986 年的时候，公司就已经有了上市的条件。但是我觉得这个时候公司还不够成熟，正好 1987 年就发生了股灾。

在 80 年代的时候，游戏机的处理能力不足，而且都很老式，所以 EA 只为更强大的电脑做游戏。电脑有海量的储存，而且可读可写，还有键盘、打印机、调制解调器等各种配件，再看看雅达利的游戏机，简直就是笑话。不过就在此时，任天堂给了游戏机第二次生命。无需软盘而又廉价的硬件和即插即玩的游戏卡带很明显可以把游戏推向更广大的市场。美国游戏行业认为任天堂会像雅达利那样昙花一现，所以大家都不理它，只希望它赶紧关门大吉。我不喜欢任天堂那些极端压榨式的授权条款，但是我却情不自禁地想进入卡带游戏市场。

1988 年的时候，我正在思考进入电子游戏的最佳策略，正好 Tengen 公司和任天堂打了一场官司，给了我灵感。我决定，我们应该关注 16 位游戏机市场，然后对世嘉公司的 16 位游戏机 Sega Genesis 进行逆向工程，以进入电子游戏领域。我觉得 EA 必须准备足够的钱才能执行这个计划，这其中包括万一被世嘉告了以后可能面临的巨额赔偿。在我做出了这个战略决定之后，上市筹钱的必要性就很明显了。很讽刺的是，虽然我们上市时只筹到了 8 百万美元，但是这笔钱 EA 却一分一厘都没有用过，也没必要用。1990 年的时候，我们和世嘉签订了一个非常不错的合约，钞票滚滚而来，完全没有出现任何法律问题。

兰姆塞：我们先说说 1991 年，你为什么会离开 EA？

霍金斯：当时我并不觉得我要离开 EA，但是结果我还是离开了。1988 年到 1990 年间，我决定为 EA 做一次最大的转型，进军电子游戏行业，策略是对一个新的、未经市场考验的游戏机平台进行逆向工程。1988 年，Tengen 对任天堂的游戏机进行了逆向工程，还告了任天堂。我关注了整个事态，直到最后 Tengen 遭遇惨败。

这个时候，世嘉推出了 16 位的 MegaDrive，后来又在日本改名叫做 Genesis。我很喜欢这台机器，然后调查了一种对其进行逆向工程的方法。只要这样做，EA 就能自由地在这个游戏机平台上发布游戏，而这个游戏机可能在 1989 年的时候就能登陆美国，1990 年登陆欧洲。那时，EA 已经有了 16 位的游戏品牌和技术，要把这些转移到 Genesis 上很容易。

这个计划就这样继续进行着，1989 年逆向工程完成，1990 年产品准备完毕。然后我又找到了世嘉，和他们定下了一个巨额战略合作协定，这让我们成为了好伙伴。事实证明，这次的合作不但为 EA 省下了 3500 万美元的授权费用，还把 EA 的市值从 6000 万美元坐火箭一般地推到了 20 亿美元，而这一切仅花了两三年的时间。虽然一切都尘埃落定，但是我仍然很担心 EA 以后的平台选择。

我担心以后任天堂和世嘉会联合起来，让 EA 无法享受到现在这样的自由发布权。当时，索尼还没进入游戏行业，电脑基本上跟游戏绝缘了。我认为，EA 现在应该主动出击，在游戏平台的开发上花点心思——可能赶不上微软或者世嘉的投入程度，但是至少要能推动市场的拓展，保证发布的自由。我希望消费者们和开发人员能用上 3D 图像、光学的储存介质和网络。

为了推动这个战略，我们在内部开展了一个特殊实验计划，不过这个计划最后从 EA 独立了出来。我当时迫不及待地想把这个计划独立出来，做成像“杜比环绕声”那种授权模式，我们为这个计划取名 3DO。1990 到 1994 年间，我是 EA 和 3DO 两家公司最大的股东，也同时担任两家公司的董事长。不过，由于在日程安排和工作重心上的冲突，两家公司逐渐疏远了。

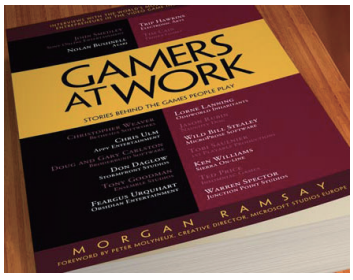
兰姆塞：放弃自己亲手创造出来的东西，你有什么样的感觉?

霍金斯:我就像是两个孩子的父亲，其中一个是十几岁的叛逆少年，而另一个则是需要做开腔心脏手术的婴儿。我觉得照顾好婴儿是我的职责所在，而且我相信少年总会有长大成人并走上正轨的一天。3DO 运营了 12 年，曾经两度达到了一亿美元的营收大关。

3DO 无法存活下来的原因很多。我犯了很多错误，而且有很多理念也过于超前了。我和 EA 渐行渐远，这让我觉得很难过，因为 EA 也是我的孩子。我一直都很崇拜沃尔特·迪斯尼，并把他作为自己的榜样,他直到死都没有离开自己的公司。很遗憾3DO失败了，我愿意承担全部的责任。

生活会磨砺和打造我们的品性，通常我们会朝好的方向发展，因为随着年龄的增长，我们明白了什么是真，什么是善。有一件事我是明白的，那就是我迟早会推行 3DO 这一类的计划，因为当时我拼了命地想要突破自己的极限。我觉得我们这种年轻创业家们都差不多，总是在不断地追寻真我，总是想看看自己能走多远。只有栽了几次跟头之后，你才会真正明白自己的极限。

兰姆塞：招募移动和社交类的游戏开发人员时遇到过困难吗？我听说有些开发人员很不喜欢加入那些以商科毕业生为主导的团队，



在《Gamers at Work 中文版（暂定名）》里，作者对18位创业家进行了深度采访。他们中有的现在还呆在最初创业的那家公司，有的是第n次创业中，而有的则已经在享受退休生活了。本书选取了数位成功的软件游戏公司的创业者的创业成长经历，着重介绍他们的创业心得体会，具体包括企业家该有怎样的策略和执行力，如何发展长期的商业伙伴关系，如何规避致命的风险等。这些故事的讲述者们也述说了很多有趣的闲闻轶事，希望读者们能从这些故事中得到启示和快乐。

他们似乎比较喜欢由资深的游戏程序员来带领的团队。

霍金斯：我们差不多有350名雇员，5个重要办公室，分布在三个大洲上，因此员工背景都各不相同。我觉得对于员工来说，最重要的事情是受过良好的教育，天资聪颖，而且愿意学习和适应各种环境。过去，那些扎马尾辫的设计师和程序员们设计制作游戏，然后由那些西装革履的人把这些东西变成产品再摆到商店里，钱都给后面这群人拿了。运营这样的公司成本很高。现在，扎马尾辫的和穿西装的有时候是同一个人。不管怎么样，艺术和科学两种视角必须融合为一体。做创意的人们必须走出象牙塔，亲自用培养器皿来做做实验，而科学家则必须明白一定要有创意才能做出有趣的产品。

兰姆塞：你从很多角度来观察过这个行业，发行、开发甚至硬件你都做过。照你看来，对于创业公司的成长和长期可持续性来说最大的威胁是什么？

霍金斯：数字媒体已经把传统的价值链给击垮了，所以，过去所谓的“现金为王”和“渠道为王”的思想已经行不通了，它们已经不再是控制货架和运输的重要手段。现在，像苹果公司、Facebook和谷歌这样的数码平台公司，已经有能力控制整个产业链。

我们都要担心这些公司一家独大之后会干坏事。要想给这些巨头施加良性的影响，小的游戏开发商们就必须想办法来联合起来。只有聚集在一起，才能有讨价还价和参与政策制订的权利。再往前走，游戏产业将变成技术和知识产权的竞争。创业公司必须想办法加大在这些领域的投入。办法总是会有的。

书　榜

征服 C 指针

作者：前橋和弥

译者：吴雅明

书号：978-7-115-30121-5

图灵社区推荐：12

本书被称为日本最有营养的 C 参考书。作者是日本著名的“毒舌程序员”，其言辞犀利，观点鲜明，往往能让读者迅速领悟要领。书中结合了作者多年的编程经验和感悟，从 C 语言指针的概念讲起，通过实验一步一步地为我们解释了指针和数组、内存、数据结构的关系，展现了指针的常见用法，揭示了各种使用技巧。另外，还通过独特的方式教会我们怎样解读 C 语言那些让人“纠结”的声明语法，如何绕过 C 指针的陷阱。

前桥和弥（Maebasi Kazuya）1969 年出生，著有《彻底掌握 C 语言》、《Java 之谜和陷阱》、《自己设计编程语言》等。其一针见血的“毒舌”文风和对编程语言深刻的见地受到广大读者的欢迎。作者主页：http：//kmaebashi.com/。

Android 软件安全与逆向分析

作者：丰生强

书号：978-7-115-30815-3

图灵社区推荐：5

为了使读者对文中所讲述的内容有深刻的认识，并且在阅读时避免感到乏味，书中的内容不会涉及太多的基础理论知识，而更多的是采用动手实践的方式进行讲解，所以在阅读本书前假定读者已经掌握了 Android 程序开发所必备的基础知识，如果读者还不具备这些基础知识的话，请先打好基础后再阅读本书。

Objective-C 编程之道：iOS 设计模式解析

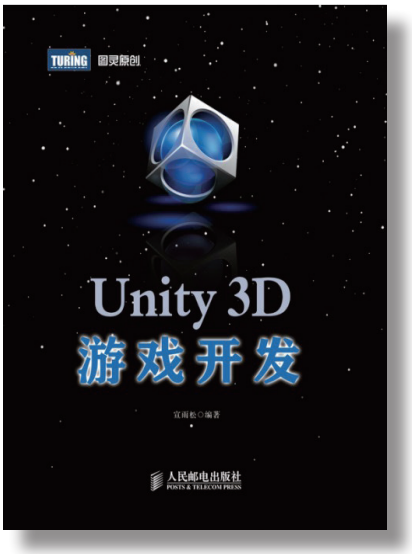
作者：Carlo Chung

译者：刘威

书号：978-7-115-26586-9

图灵社区推荐：4

iOS 应用程序的基础 Cocoa Touch 框架内容丰富、结构优美，通过将各种设计模式应用到其基础结构中，为第三方开发者提供了很好的可扩展性和灵活性。本书受到 GoF 的经典著作《设计模式》的启发，旨在引导大家掌握如何在 iOS 平台上以 Objective-C 语言实现 Cocoa Touch 开发所要用到的传统设计模式。



Unity 3D 游戏开发

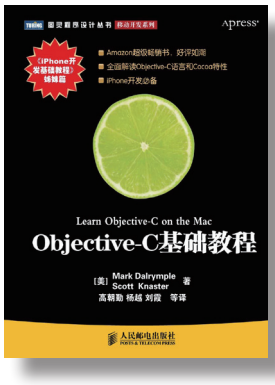
作者：宣雨松

书号：978-7-115-28381-8

图灵社区推荐：7

本书详尽介绍了 Unity 的安装、使用及深入开发等，并通过相应的实例来巩固知识点，是快速入门及提高 Unity 技术的必备书。愿本书能给我们大家带来越来越多由 Unity 开发的优秀游戏！

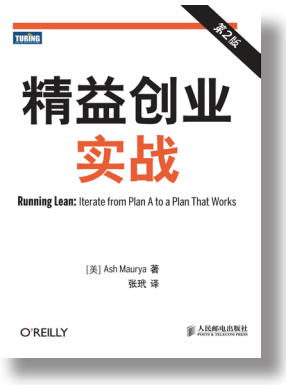
5



Objective-C 基础教程

作者：Mark Dalrymple, Scott Knaster
译者：高朝勤，杨越，刘霞
书号：978-7-115-20877-4
图灵社区推荐：3

6



精益创业实战

作者：Ash Maurya
译者：张玳
书号：978-7-115-30540-4
图灵社区推荐：8

7

大数据：互联网大规模数据挖掘与分布式处理

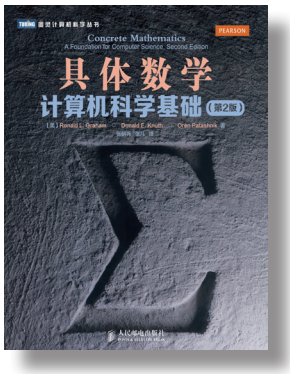
作者：Anand Rajaraman, Jeffrey D. Ullman
译者：王斌
书号：978-7-115-29131-8
图灵社区推荐：19

8

App 创富传奇

作者：Chris Stevens
译者：曾文斌
书号：978-7-115-31011-8
图灵社区推荐：11

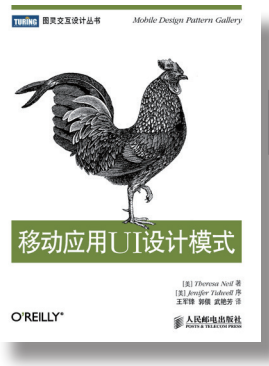
9



具体数学

作者：Ronald L.Graham, Donald E.Knuth, OrenPatashnik
译者：张明尧，张凡
书号：978-7-115-30810-8
图灵社区推荐：21

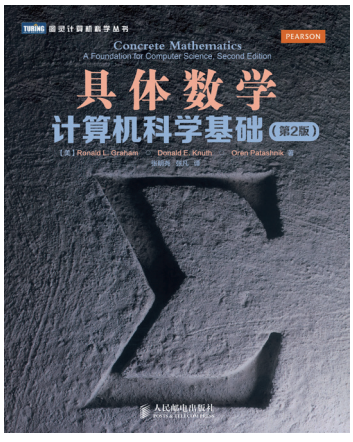
10



移动应用 UI 设计模式

作者：Theresa Neil
译者：郭偎，武艳芳，王军锋
书号：978-7-115-29648-1
图灵社区推荐：6

好书妙评之《具体数学》



《具体数学》由当今顶级数学家和计算机科学家合著的经典著作，自 1990 年出版以来经久不衰，并被世界多所知名大学采纳为教材，是当代计算机科学方面的一部重要著作。书中不仅讲述了数学问题和技巧，而且教导解决问题的方法，解说深入浅出，妙趣横生。大师们诙谐、细腻的笔触，描绘着数

请谨慎些 Please Be Discrete

作者：Mary P. Campbell "math geek"
213 位读者评价了这条书评，207 位认为有价值

什么是“具体”的数学，而不是其他类型的数学？作者解释说，书名 (Concrete Mathematics) 融合了数学的两个分支：连续 (CONtinuous) 数学和离散 (disCRETE) 数学。虽然许多人看似喜欢将这两个分支区别开来，但它们彼此互为基础。本书的论题（如和式、生成函数、数论）实际上是标准的离散数学课题；然而，本书处理它们时显示了内在的连续性（即：微积分）。没有微积分，生成函数就无法实现，它们的巨大威力也就无法用于级数的研究。

对于那些注重细节的人，尽管有些自作聪明的旁注，这是一本严肃的数学书。与大多数数学书（尤其是研究院的数学书）不同，本书没有省略任何东西。记号被详细解释（非常重要），常见的陷阱被指出（而不像通常那样，让学生在考试中遇到它们被碰得血淋淋），重要之处和不那么重要的地方都被指出。

还有，我不能不评论这些旁注，有些旁注是对读者的严重警告。

学工作中的欢乐和忧伤，那些或平淡、或深刻、或严肃、或幽默的涂鸦，更让我们在轻松愉悦的心境下体会数学的美妙。

例如，本书前言中有一条旁注“我建议混学分的学生不要上这门课。”有些旁注建议读者略读，有些旁注建议严肃对待。所有来自帮助创作本书的“友好的助教”的旁注，对学生可能面临的困难有更根本更详细的了解。不过，也有大量的双关语和冷笑话，时不时地逗逗读者：“空集不含点”、“但不是非贝塞尔 (Imbesselian) 函数”和“约翰 .316”令我会心一笑，但你必须自己发现它们。

对于那些已通过艰难的数学专业研究的人而言，阅读本书是件轻松愉快的事；至于其他的人，务请铭记，这是一本严肃的书，必须准备好做些实实在在工作。非常聪明的高中学生领会本书的难度不大。 我要提前指出，本书中有些习题是有深度的研究题。如果一道习题已经费了你一个星期的时间，就应该看看后面的答案，检查该题是否确实有解；他们给你习题时不一定会告诉你这一点。

因此，我极力推荐这本书给那些想要解决一些需要多费心思的难题的数学爱好者。这里的公式在“在实际生活中”确实派得上用场，尤其是对那些使用数学较多的人而言。

我希望每本书都这样写！

作者：Anthony Widjaja To
57 位读者评价了这条书评，57 位认为有价值

也许，这本书是我读过的写得最精美的书籍之一。书中呈现的证明皆简练。当你阅读书中的证明时，你能感觉到句子间联系的逻辑性、流畅性。部分原因是作者采用了简练有效的记号。[注：作者之一高德纳 (Donald Knuth)，是良好数学记号的最大支持者之一。参见他的著作《数学写作》(Mathematical Writing)。]

其他的书评已经对本书作出了总结。所以，我只能说，每一位计算机科学家和组合学家至少应该读读第 1、2、5、7、9 章。强烈推荐第 5 章。相信我：一旦你掌握了这些章节，你将能够做你同事所不能做的事情。即使只是熟悉了本书的记号，将有助于你发现证明，如果你无法以其他方式获得。在每个证明中，伟大的想法当然总是很重要的 - 但如果没有良好的记号，你很可能压根儿就没有想法。

据我所知，本书几乎没有什么不好的。我只会说，掌握书中的知识需要花费很多时间和精力。就我本人来说，当我对离散数学感兴趣时，我开始阅读第 1、2 章。花了约半年的业余时间读通这两章。学习“生成函数”课程时，我再次找到这本书。我发现，第 5、7 章是必不可少的。我也被迫重读第 2 章，因为，当涉及到处理和式及二项式系数时，我的讲师（和大多数人一样）只是挥了挥他的手。当然，我付出的所有努力终得报偿，在期末考试中我的朋友不能解答的问题，我能解答。

总之，我强烈推荐本书给每一位计算机科学家和组合学家。最后的评论是，如果你在用心学习具体数学，可能会发现生成函数几乎无处不在。想弄懂这些讨厌鬼，我强烈推荐塞奇威克 (Sedgewick) 和弗拉若莱 (Flajolet) 的《算法分析导论》(Introduction to Analysis of Algorithms) 和《分析组合数学》(Analytic Combinatorics)（尚未出版，但弗拉若莱的网站提供次定稿），以及维尔夫 (Wilf) 的《生成函数》(Generatingfunctionology)。

伟大的书 ... 有些评论者根本不懂她

作者：Wayne Folta
45 位读者评价了这条书评，45 位认为有价值

我已有本书第 1 版，到这里来看看第 2 版。有一些负面评论，基本上都是那些书评人有重大误解而做出的。所以我要添上我的书评。

首先，她是一本什么类型的书？她不是一本有许多重复的入门级数学书。她是一本由出色的教师以精炼的方式完成的有深度的数学书，要求学习的人掌握微积分、概率论等知识。你真的没必要把她当作一本代数入门课程（或其他东东）似的略读浅尝。（我不是精英。我没能进入斯坦福大学，也不认为自己是数学天才，这不是“我们与无知大众”的问题，因为我自己也纠结于书中的知识。）

其次，本书讲什么？一些评论者对书名中的“具体 (Concrete)”一词的说法有研究，说到底，她就是一本你必须掌握的理论计算机科学的离散数学书。（例如，本书讲述的离散运算与我们在学校学到的连续微积分相对应。）举个例子，在任何算法分析课程中你都将面对递归方程以及大量的离散数学。

第三，本书是如何组织的？初看起来，她显得很散乱。当你阅读时，作者似乎有“看，这些花”或“看，岩石下”之类的未标示的提示。但实际上，本书确实是按阶递进地编写的，例如由第 1 章的递归式（河内塔、若瑟夫问题）到第 7 章的生成函数。

也许能采用更为简洁的方式，但我认为在编排上并不会有太多的差异。可能他们会将某些章节全部扔到附录，但读者将不得不不停地在章节和附录间转换，因为这些知识联系得如此紧密。

第四，哪些其他书籍涉及这些知识？我没资格去谈论全部的书籍，但我必须说我当前上课用的三本算法分析书只是明确地给出了这些知识的基础且只给出了两种可能性：1) 摆弄公式，可能使用图

形表示，直至你能发现一个模式并作出猜测，然后用归纳法证明；2) 如果你的算法是一个特定的类别，将一些数字代入三步公式，如果其中的一步可用那么就得到答案。《具体数学》给你许多强大的工具去解决这类问题。

第五，本书的风格是什么？作者采用的是非正式的写作风格——跳出正式的数学和证明——带有由写作本书时的“试读者”贡献的旁注。

一些评论者对旁注持批评态度，我只有摇摇头庆幸我不需要和他们合作。是的，有些旁注只是些双关语和小幽默，但是可令读者从沉重的数学问题中获得放松。并且，许多旁注提供了重要提示及其他学习本书的学生的观点。我希望所有的技术书籍都有这样的旁注，但只有高德纳 (Knuth) 找到了不怕辛劳的出版商。

这就是我的观点。这是一本有份量的讨论高难度技术话题的优秀书籍。阅读她犹如用另一种语言将数学由代数至微分方程重学了一遍，也许那些持批评态度的评论者只是没有读懂她而已。如果你不是一个需要严格分析算法的程序员，不是一个只是为学习的乐趣而读书的人，请忽略她。

我最喜爱的数学书，毫无疑问

作者：一位勤奋好学的学生
30 位读者评价了这条书评，30 位认为有价值

这是迄今为止我最喜爱的数学书。我在普特南 (Putnam) 数学竞赛预备课程中被推荐了这本书，开始的时候我没有买（似乎太过分了）。我在学校图书馆里找到了她并用她做了些作业。研读了几晚，

我跑去买了一本（半价，多么划算！）当然，这本书并不适合每个人，但如果对离散数学感兴趣，一定要选这本书。

我还没有读完整本书（只是扎实读完六分之一，在买了这本书之后的课程学习中，我参考了约四分之三的内容）。我最喜爱本书以下几点：

- 厚实度和可读性合理平衡。有些数学书很厚重，以至于每一页需花一个小时或更多的时间去读懂。这本没那么沉，但确实比你第一年上的微积分课程难。如果你习惯如风掠过似的学习一章（包含 2~6 个概念还有一些练习题），你得适应慢下来，并做些认真的复读。我估计，依据读者对这些知识的了解程度，他们可能花费二十至六十小时来掌握每一章的知识。这包含了做章末习题的时间。这看起来需要大量的时间，相信我，当你读完本书时将学会这行里的所有门道。
- 直观的方法。我发现很多教科书采用线性方式花费许多的时间来阐明知识。在这种意义上来说，当知识被掌握之后，文章的组织结构是有意义的；但当你在学的时候，这种结构看上去却是随意并无序的。初读此书时，她就是非常直观的，你的意识想到哪，她的观点就流进哪；这个意义上说，她的思想跟你的思路行云流水般契合。大多数章节以一个（令人耳目一新的、困难的）问题开始，伴随着学习解决问题所需的技巧，问题的解答方法也就被你掌握了。当你读这本书时，你能学会非常有效的解决问题的技巧。
- 合适的深度。作者不是简单地教你如何处理特定类型的和式，而是深入剖析了这类问题后的原理，也包括“这行的门道”。在本书中我学到一些和式的处理技巧，我还需去别处瞧瞧，它

们非常好用！例如，一类“离散运算”的定义，让你非常轻松地得出（从 1 到 n 的）整数的 k 次幂的和式的封闭形式（使用了斯特林数，我敢肯定在网络上可以找到它。）

这些是我不喜欢的：

- 在数论方面别投入太深。涉及基础就好了，但不要陷入 RSA 算法。
- 别像我所预期的那样把连续数学知识混入。

总而言之，本书将离散数学的基本课题讲述得非常好。读完她之后你将完全掌握和式、离散概率及数论。对于你在本科课程不常遇到的离散数学知识，她给你惊人深度的展示，包括：

- 底函数 / 顶函数及其应用。
- 二项式系数（n 选取 k）。
- 生成函数。
- 特殊的数（斯特林数、斐波那契数、调和数以及其他的数）。
- 大 O 记号（这里终于涉及到计算机科学）。

必读此书者：

- 数学专业大学生。
- 喜欢但久未接触数学的人。
- 研习连续数学（微积分、高中数学、分析），寻求多元化思维的人。
- 钻研素数、斐波纳契数，或者研究奇妙的数学模式的人。

勿需打扰者：

- 糊涂。
- 数学天才。
- 电脑奇才，但不喜欢数学（尽管本书副标题是“计算机科学基础”，她是一本纯粹的数学书）。
- 只关心答案而不关心过程的工科学生（看看那些给一星评价的家伙）。
- 注意：本书并没有采用你习惯的“命题 - 证明 - 例题”的模式。你已经被警告过了。

这本书写得非常好，我保证，如果你属于“必读此书者”，你就不可能摆脱她，她会终身栖息于你的书架。

此外，关于书名。似乎有些人困惑为什么她被称为《具体数学》（看看那些给一星评价的家伙）。不是因为她呈现的是每日的、公式化的数学而称为具体数学。我从本书前言得知，之所以是“具体 (CONCRETE) 数学”，是因为“她融合了连续 (CONtinuous) 数学和离散 (disCRETE) 数学。”这是迄今为止我见过最傻的书名公式，我想作者们并不这么想。

最后，由于在本书订购页面没有看到目录，我按顺序将本书内容列在这里：

- 递归
- 和式
- 整值函数（底、顶、模）
- 数论
- 二项式系数（n 选取 k）
- 特殊的数（斯特林数、欧拉数、伯努利数、斐波那契数以及其他的数）

译者 / 江志强

毕业于厦门大学数学专业，计算机应用软件工程师，高级程序员。对数论有浓厚的兴趣，仔细阅读了《哈代数论 (第 6 版)》。高中时自学 Pascal，上大学后学习了《C 程序设计语言》(K&R)，现在编程主要用 C#。图灵社区 ID：空军。

- 生成函数
- 离散概率
- 渐近式（大 O 记号）

在你查阅习题答案之前，有 500 页的好东西。哦，对了，书里没有浅薄的注着“奇数题号的习题答案”的背页。每个问题都是完全解答，有些解答超过一页。如果这不是最好的学习方法，我不知道什么是。阅读愉快。■

更多亚马逊书评

欢迎加入
图灵社区

最前沿的IT类电子书发售平台

电子出版的时代已经来临。在许多出版界同行还在犹豫彷徨的时候，图灵社区已经采取实际行动拥抱这个出版业巨变。作为国内第一家发售电子图书的IT类出版商，图灵社区目前为读者提供两种DRM-free的阅读体验：在线阅读和PDF。

相比纸质书，电子书具有许多明显的优势。它不仅发布快，更新容易，而且尽可能采用了彩色图片（即使有的书纸质版是黑白印刷的）。读者还可以方便地进行搜索、剪贴、复制和打印。

图灵社区进一步把传统出版流程与电子书出版业务紧密结合，目前已实现作译者网上交稿、编辑网上审稿、按章发布的电子出版模式。这种新的出版模式，我们称之为“敏捷出版”，它可以让读者以较快的速度了解到国外最新技术图书的内容，弥补以往翻译版技术书“出版即过时”的缺憾。同时，敏捷出版使得作、译、编、读的交流更为方便，可以提前消灭书稿中的错误，最大程度地保证图书出版的质量。

现在购买电子书，读者将获赠书款20%的社区银子，可用于兑换纸质样书。

最方便的开放出版平台

图灵社区向读者开放在线写作功能，协助你实现自出版和开源出版的梦想。利用“合集”功能，你就能联合二三好友共同创作一部技术参考书，以免费或收费的形式提供给读者。（收费形式须经过图灵社区立项评审。）这极大地降低了出版的门槛。只要你有写作的意愿，图灵社区就能帮助你实现这个梦想。成熟的书稿，有机会入选出版计划，同时出版纸质书。

图灵社区引进出版的外文图书，都将在立项后马上在社区公布。如果你有意翻译哪本图书，欢迎你来社区申请。只要你通过试译的考验，即可签约成为图灵的译者。当然，要想成功地完成一本书的翻译工作，是需要有坚强的毅力的。

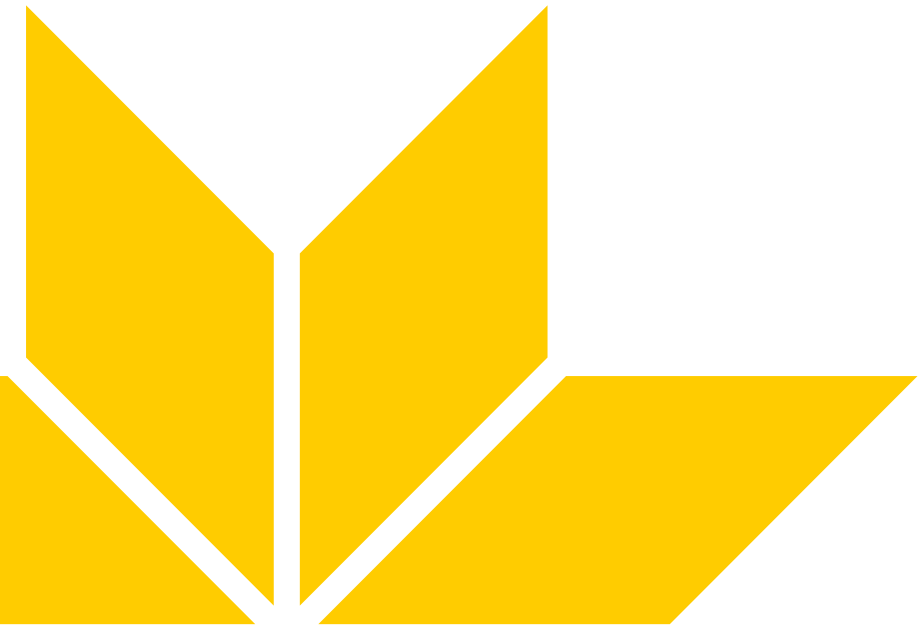
最直接的读者交流平台

在图灵社区，你可以十分方便地写作文章、提交勘误、发表评论，以各种方式与作译者、编辑人员和其他读者进行交流互动。提交勘误还能够获赠社区银子。

你可以积极参与社区经常开展的访谈、乐译、评选等多种活动，赢取积分和银子，积累个人声望。

图灵社区 出品

出版人：武卫东
编辑：李盼
顾问：杨帆、谢工、李松峰
设计：大胖



地址：北京市朝阳区北苑路13号院领地OFFICE C座603室
电话：010-51095181
微博：weibo.com/ituring
Email: ebook@turingbook.com