

疯狂 iOS讲义 (提高篇)

李刚 编著

疯狂源自梦想

技术成就辉煌

疯狂源自梦想

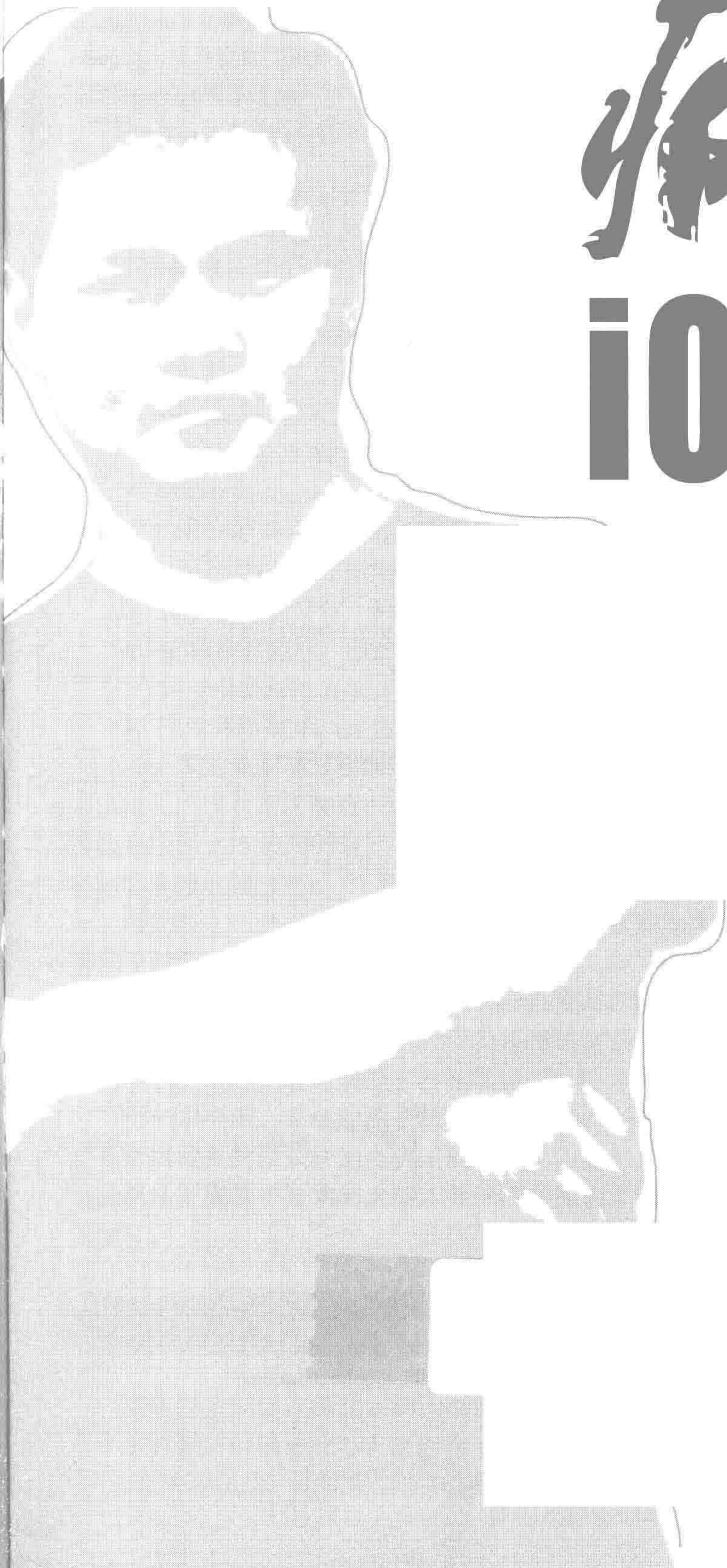
技术成就辉煌



中国工信出版集团



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn



疯狂 iOS讲义 (提高篇)

李 刚 编著

电子工业出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书基于《疯狂 iOS 讲义（下）》升级而来，已真正升级成 Swift、Objective-C 双语讲解。本书所有案例全部提供 Swift、Objective-C 两个语言的版本。本书基于最新的 iOS 9.3、Xcode 7.3 平台，全面介绍 iOS 应用开发的高级知识。本书包含多点触摸与手势处理、应用国际化、数据存储、SQLite 数据库与 Core Data、iOS 多媒体开发、加速计和陀螺仪、多线程、网络编程、XMPP 即时通信、定位、地图、推送机制、iCloud 服务、HealthKit 框架等内容，本书最后还通过疯狂软件商城 App 整合介绍了 iOS App 的界面开发、自定义 UI 控件、手势处理、网络通信、本地数据存储、在线支付等功能的用法。读者在阅读本书之前，建议先掌握《疯狂 iOS 讲义（基础篇）》中的知识。

本书并不局限于介绍 iOS 编程的各种理论知识，而是从“项目驱动”的角度来讲授理论，全书一共包括近百个实例，这些示范性的实例既可帮助读者更好地理解各知识点在实际开发中的应用，也可供读者在实际开发时作为参考、拿来就用。如果读者在阅读本书时遇到技术问题，可以登录疯狂 Java 联盟（<http://www.crazyit.org>）发帖，笔者将会及时予以解答。

本书为所有打算深入掌握 iOS 编程的读者而编写，适合各种层次的 iOS 学习者和开发者阅读，也适合作为大学教育、培训机构的 iOS 教材。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

疯狂 iOS 讲义. 提高篇 / 李刚编著. —北京：电子工业出版社，2016.8
ISBN 978-7-121-29700-7

I. ①疯… II. ①李… III. ①移动终端—应用程序—程序设计 IV. ①TN929.53

中国版本图书馆 CIP 数据核字（2016）第 190887 号

策划编辑：张月萍

责任编辑：葛 娜

印 刷：三河市华成印务有限公司

装 订：三河市华成印务有限公司

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：787×1092 1/16 印张：36.75

字数：1036 千字

彩插：2

版 次：2016 年 8 月第 1 版

印 次：2016 年 8 月第 1 次印刷

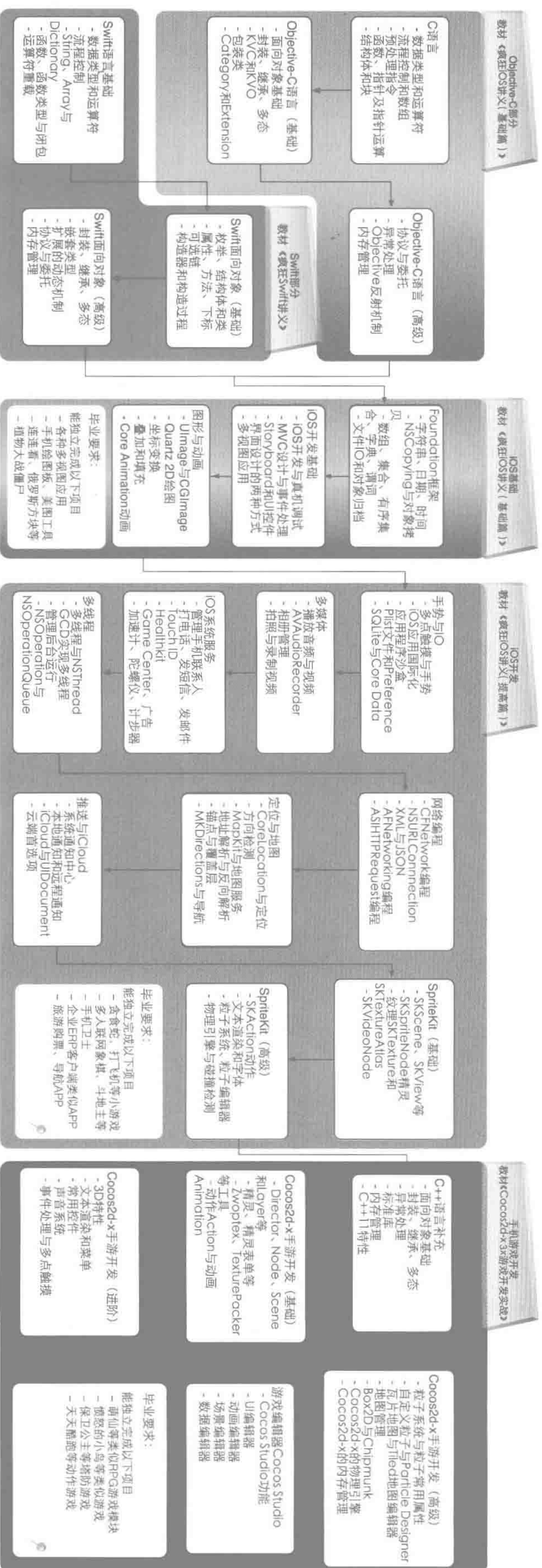
定 价：99.00 元（含光盘 1 张）

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）88254888，88258888

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

本书咨询联系方式：010-51260888-819 faq@phei.com.cn。

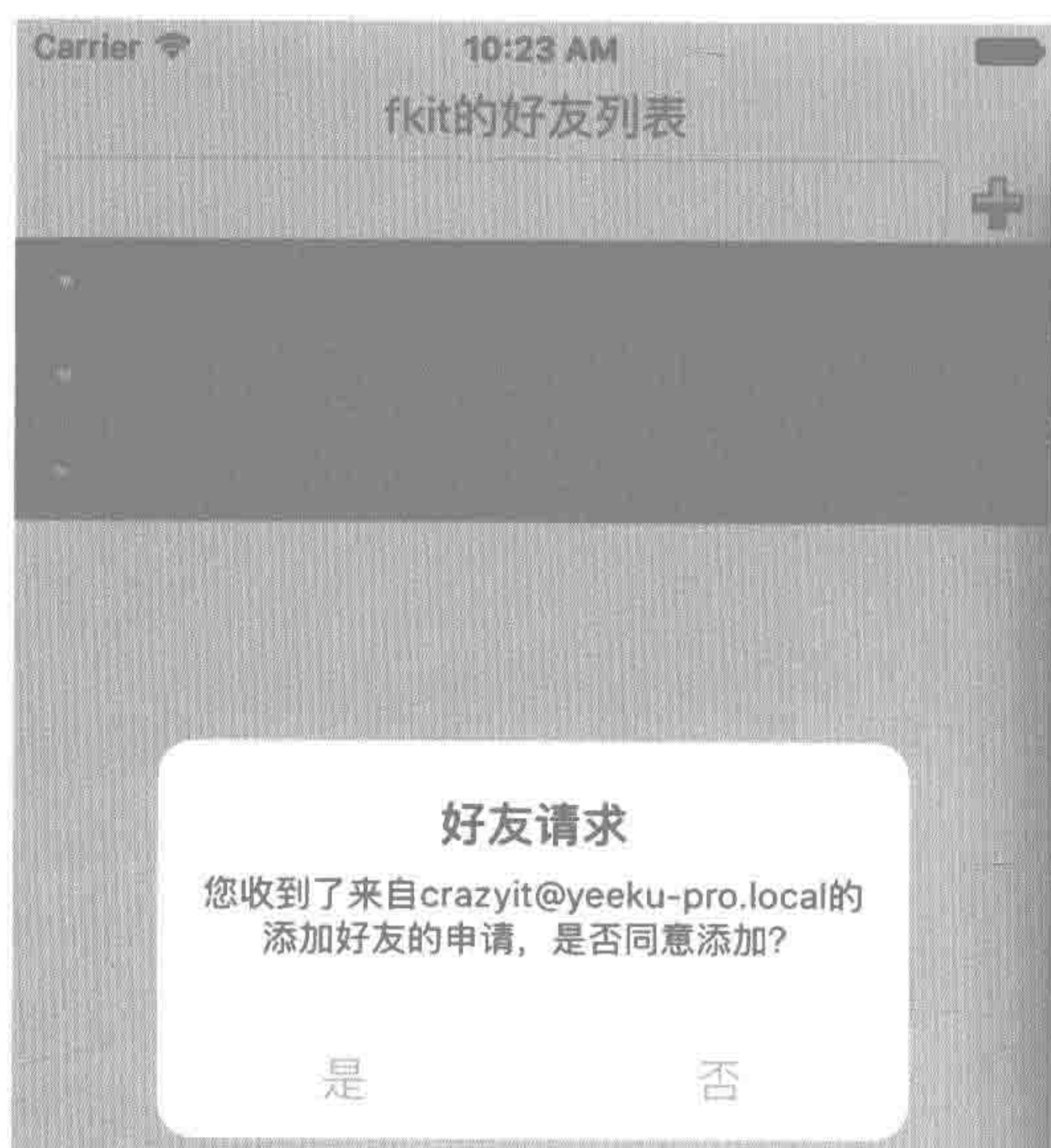
疯狂iOS学习路线图



精彩手游，尽在掌控

疯狂iOS讲义
(基础篇) (提高篇)

苹果 精彩案例



苹果 XMPP即时通信之好友请求



苹果 XMPP即时通信之聊天界面



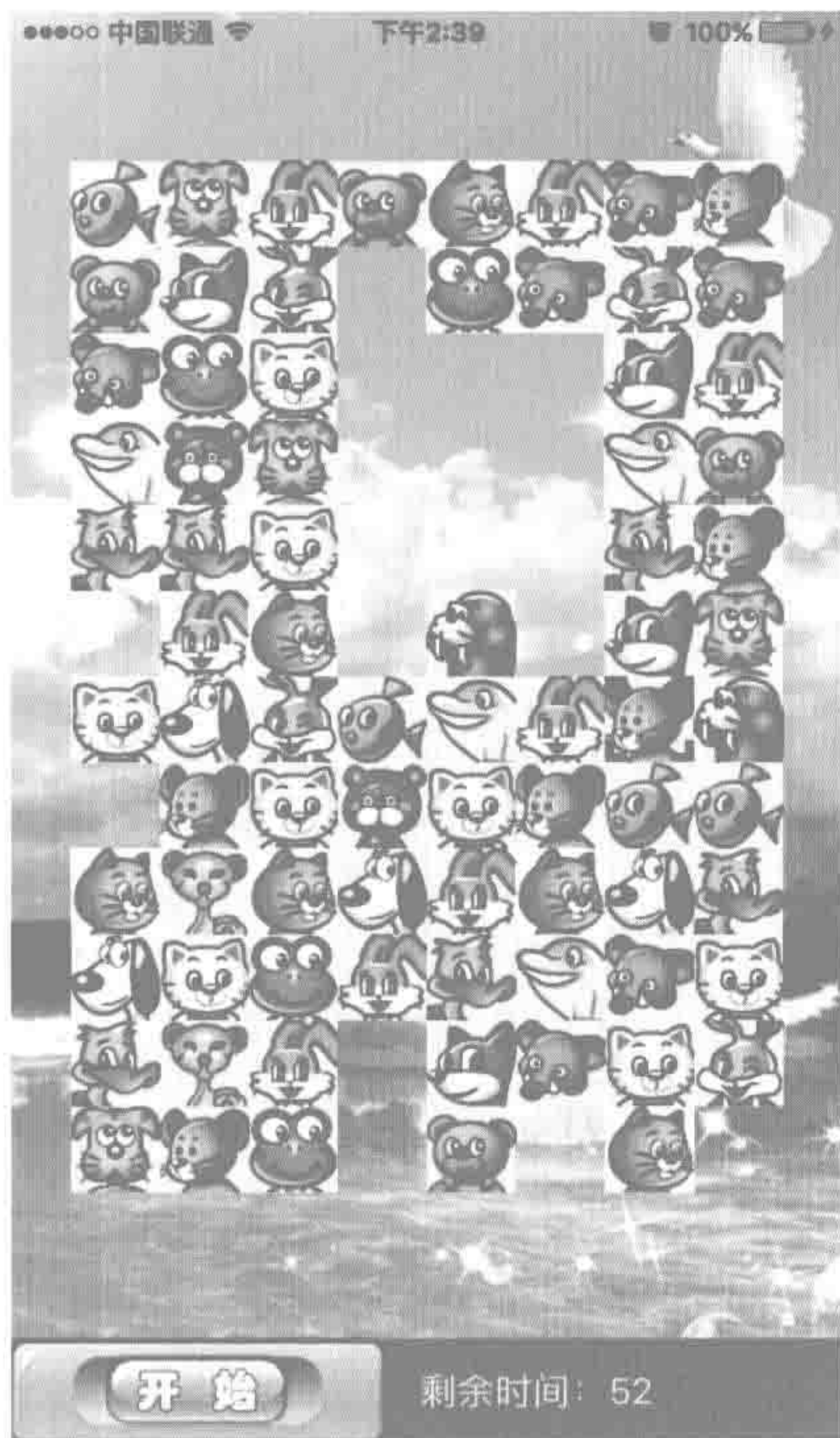
苹果 打飞机

精彩手游，尽在掌控

疯狂iOS讲义
(基础篇) (提高篇)



精彩案例



疯狂连连看1



疯狂连连看2

确认订单

收件人: 黄勇

收件地址: 广州市天河区车陂大岗路津宏大厦3011

配送方式: 顺丰快递 (24小时发货)

支付方式: 微信支付

商品列表

	经典Java EE企业应用 实战——基于 WebLogic/JBoss的J... 电子工业出...	¥ 59.20 X1
	疯狂Ajax讲义—— Prototype/jQuery+DWR +Spring+Hibernate整... 电子工业出...	¥ 100.00 X1
	疯狂Java讲义 (附光盘)	¥ 100.00
合计: ¥ 259.20		



疯狂软件之确认订单

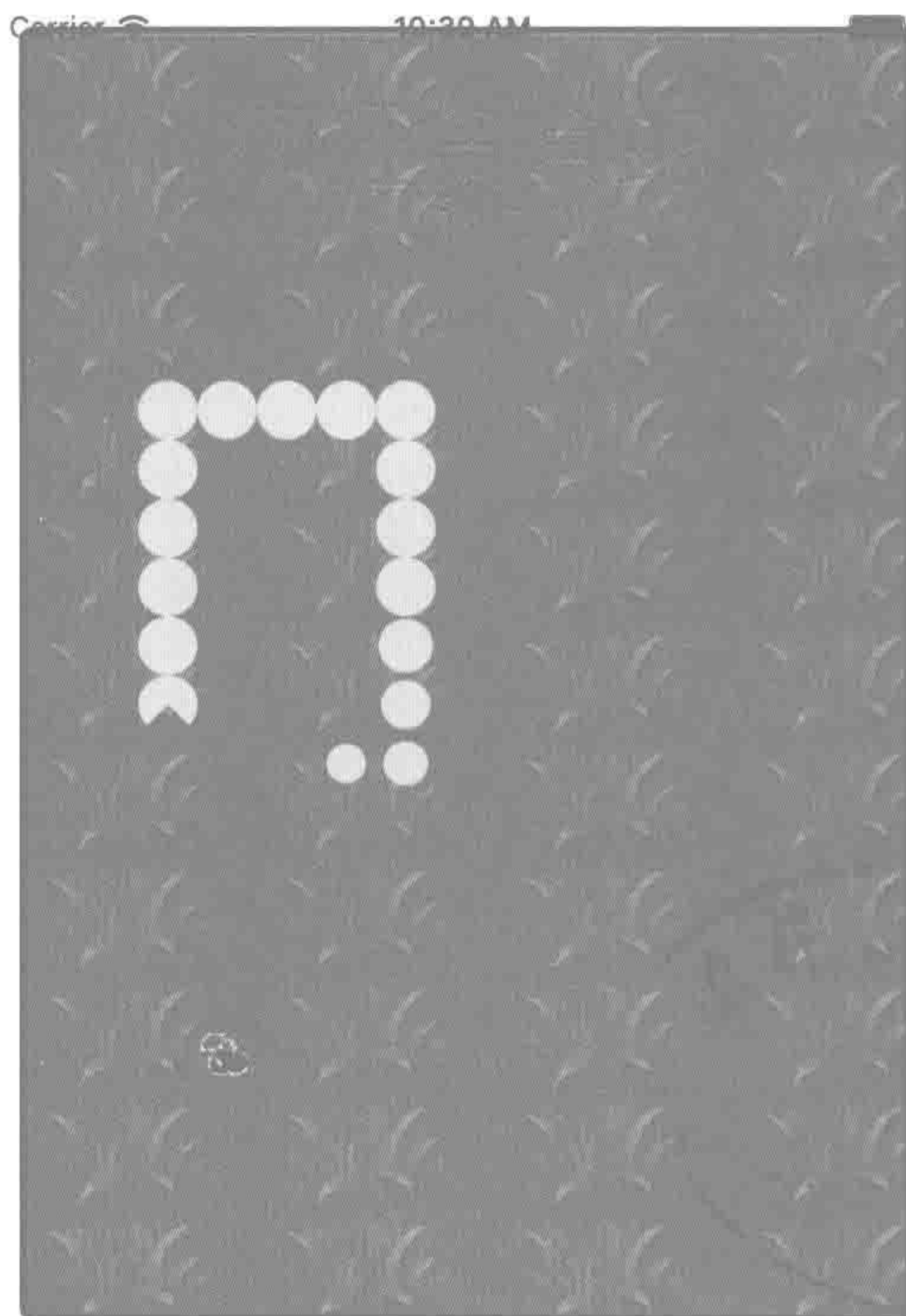


疯狂软件之物品分类

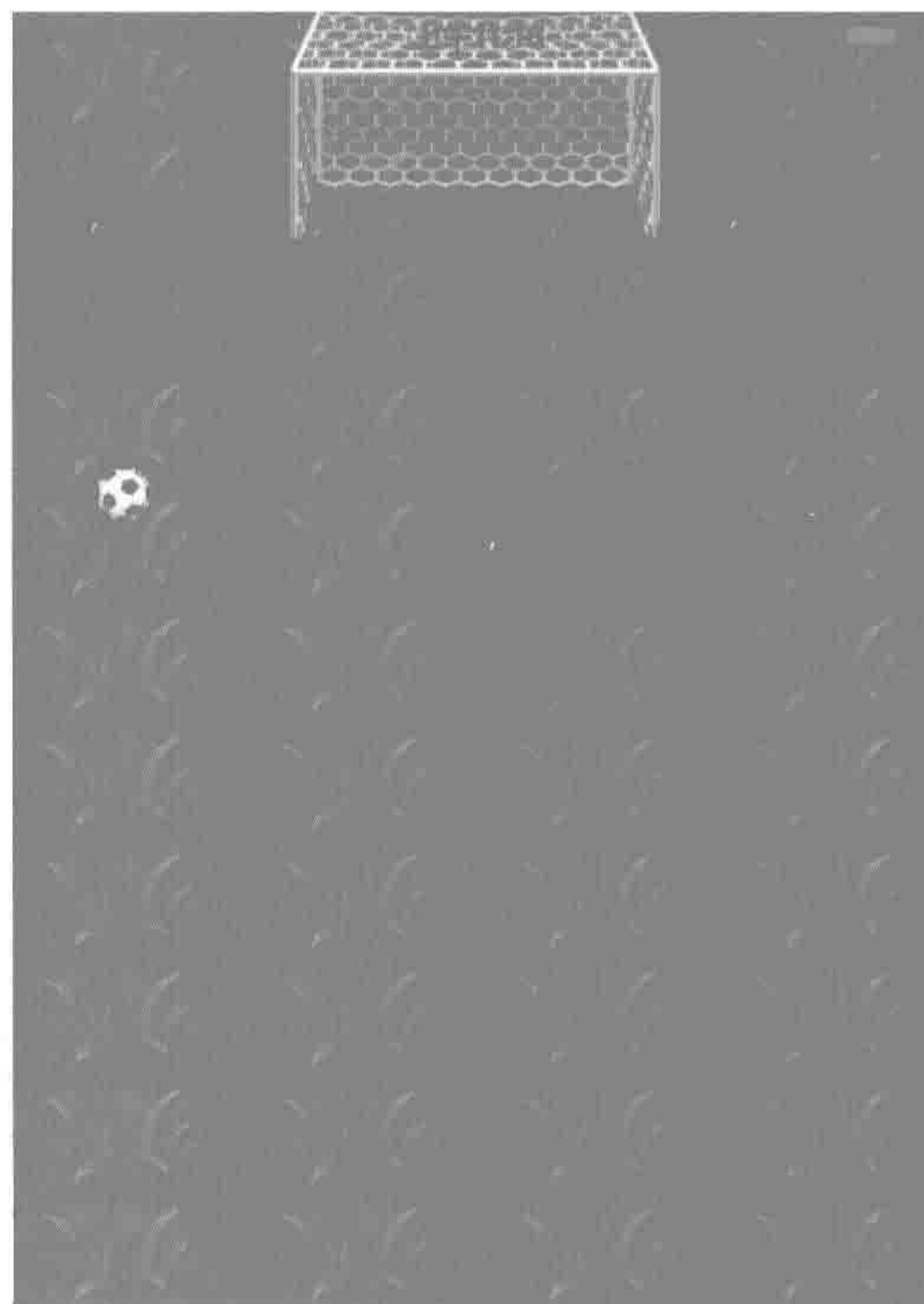
精彩手游，尽在掌控

疯狂iOSi#X
(基础篇) (提高篇)

苹果 精彩案例



苹果 疯狂贪食蛇



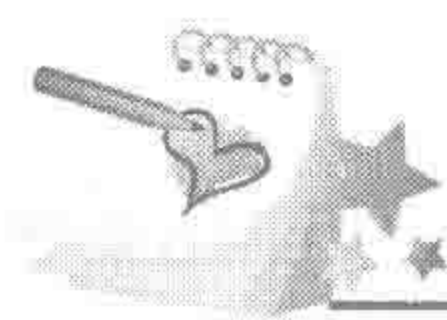
苹果 怪物足球



苹果 天气预报首页



苹果 天气预报详情



前言

本书基于《疯狂 iOS 讲义（下）》升级而来，已真正升级成 Swift、Objective-C 双语讲解。本书所有案例全部提供 Swift、Objective-C 两个语言的版本。读者通过本书可以同时掌握 Swift、Objective-C 两种语言的开发能力。

书中详细介绍了《贪食蛇》《打飞机》《天气预报》《XMPP 即时通信客户端》《疯狂软件商城 App》等大量精彩的 iOS App 的开发，本书不仅细致地介绍了开发这些应用所需的基础知识，而且真正带领读者从零开始，逐步完成这些应用。如果读者认真学完本书，并掌握了书中内容，将可以完全胜任 iOS 应用开发。

虽然本书的知识点足够深入、实用，但对于已经掌握《疯狂 iOS 讲义（基础篇）》的读者来说，本书的知识依然很容易看懂、理解，并很容易动手做出来。只要读者真正对书中的应用感兴趣，并愿意扎扎实实地动手实践，完全可以很好地学会、掌握书中这些知识点。

有些读者可能会被市面上某些薄薄的图书所吸引：他们以为阅读一本薄薄的图书更容易掌握这门技术。但实际上，无论你学还是不学，技术就在那里；无论作者写还是不写，技术就在那里。如果一本技术图书太过单薄，可能有以下三种情况：

- 技术本身就简单，只要简单介绍即可。
- 技术并不简单，但图书只介绍了该技术的皮毛，读者学起来很轻松，但实际开发时就感觉“书到用时方恨少”。
- 技术并不简单，该图书高度概括、提纲挈领地归纳该技术的要点。

iOS 开发绝不是三言两语就可以说清的一门技术，而《疯狂 iOS 讲义》并不打算只是提纲挈领地归纳该技术的要点，而是力求全面、细致地介绍 iOS 开发的相关内容，包括 Objective-C 语法介绍和 iOS 应用开发，尤其是 iOS 应用开发部分，本身包含的知识点确实比较多，读者只有全面掌握这些内容，才能真正在企业 iOS 开发岗位上游刃有余。

可能有人会觉得提纲挈领的书看上去压力没那么大，这其实是一个假象，如果仅凭某些提纲挈领的要点归纳就可以掌握一门开发技术，那只能说明其悟性很好（或有很强的技术功底），如果真是这样，那么看一本全面、细致的图书应该更易理解；反过来，如果觉得《疯狂 iOS 讲义》内容太多，不容易掌握，反而希望通过某些提纲挈领的要点归纳即可掌握这门开发技术，那更不现实——详细的书都不容易掌握，仅凭一些提纲挈领的要点归纳反而能学会？

写一本全面、系统的图书比写一本薄薄的书辛苦多了！如果只是写一本薄薄的书，可以随便介绍点常见的知识，加点例子就行了。但如果打算全面、系统地讲清楚一门技术，就需要高屋建瓴地把握整个技术的大纲，并理清各知识点之间的关联，然后还需要逐个介绍每个知识点的细节。

本书创作感言



多年过去以后，疯狂系列图书用事实证明：国内图书的质量，其实也可以很好，关键在于作者是否愿意付出全部心血去创作一本图书。《疯狂 Android 讲义》《疯狂 Java 讲义》《轻量级 Java EE 企业应用实战》这些图书在亚马逊、京东、当当等网店的销量排行榜的事实证明了国内原创

图书更适合中国开发者。

但总有些人说：还是国外的书更好——这里无意贬低国外引进的优秀图书。无论国外还是国内都可以产生优秀的图书，根据区域划分图书质量本身就是简单、粗暴的。国外作者固然创作了大量优秀的技术图书，但国内同样有一些作者在努力坚持。

“还是国外的书好”这句话之所以有人说，无非有两个原因：

- 从整体质量来看，国内原创的、优秀的图书数量确实比国外的优秀图书数量少——因此从统计学的角度来看，“还是国外的书好”这句话有一定的道理。
- 从心理学分析，说这句话的人往往是嫉妒：这些人觉得自己很行，但要么不能创作图书，要么创作的图书市场反映很差。这些人为了找回心理的平衡，不断地告诉自己：国内图书都很差，这样就可以把失败的原因归咎于所处的环境，同时又可打压周围的优秀者，从而获得心理安慰。

无论上面哪种原因，其实都应该成为国内程序员奋斗的动力。正因为国内原创的、优秀的图书数量较少，所以我们更应该努力，因为我们也是“国内程序员”的成员，提高国内软件行业的水平，提高国内原创、优秀图书的数量，国内每个软件从业人员责无旁贷。

心怀嫉妒的人，身边总是无处不在：在这些人口中，张三做的某某事很容易，李四做的某某事很容易，但这些人自己啥也不做。对于技术图书的作者而言，既不需要政府背景，也没有资金准入门槛，唯一的要求是掌握这门技术，并愿意努力介绍它——积极的人，应该通过努力把事情做好来证明自己，而不是通过语言否定别人来证明自己。

当然，如果有人坚持认为：国外图书具有更好的“思想”，那么他胜了——因为“思想”这东西，太难说清了，而本书只是一本介绍编程的工具书，并不是一本“形而上”的哲学书。如果读者想学习 Swift 或 Objective-C 编程，想在 iPhone、iPad 上开发应用，可以选择本书；如果读者想通过编程进而悟道、领悟人生哲学，应该放弃选择本书。

本书有什么特点



本书以《疯狂 iOS 讲义（基础篇）》为基础，因此本书不包含 Objective-C 语法、iOS 界面编程、图形图像处理等基础知识。

本书包含多点触摸与手势处理、应用国际化、数据存储、SQLite 数据库与 Core Data、iOS 多媒体开发、加速计和陀螺仪、多线程、网络编程、XMPP 即时通信、定位、地图、推送机制、iCloud 服务等内容，本书还穿插介绍了各种实用的 iOS App。

认真看完本书，把书中所有实例都练习一遍，本书带给你的只是 9 个字：“看得懂、学得会、做得出”。本书不能让你认识一堆新名词，只会让你学会实际的 iOS 应用开发。

本书还具有如下几个特点：

1. 知识全面，覆盖面广

不得不说，本书是市面上所有 iOS 图书中内容较为全面、体系庞大的。本书基础篇、提高篇全方位地涵盖了 Objective-C 语法、iOS 开发基础知识、iOS 开发高级知识等内容。这些知识将带领读者从零开始，逐步掌握 iOS 开发的基础，直至步入 iOS 开发殿堂。

2. 内容实际，实用性强

本书并不局限于枯燥的理论介绍，而是采用了“项目驱动”的方式来讲授知识点，全书近百个实例，几乎每个知识点都可找到对应的参考实例。还提供了《贪食蛇》《打飞机》《天气预

报》《XMPP 即时通信客户端》《疯狂软件商城 App》等真实的应用，具有极高的参考价值。

3. 讲解详细，上手容易

本书保持了“疯狂系列”的一贯风格：操作步骤详细、编程思路清晰、语言平实。只要读者有学习的决心和毅力，认真掌握本书基础篇、提高篇的知识，即可完全胜任实际企业中 iOS 开发者的要求。

不管怎样，只要读者在阅读本书时遇到知识上的问题，都可以登录疯狂 Java 联盟 (<http://www.crazyit.org>) 与广大学习者交流，笔者也会通过该平台与大家一起交流、学习。

本书写给谁看



如果你已经具备一定的 iOS 应用开发基础，或已经学完了《疯狂 iOS 讲义（基础篇）》一书，那么将非常适合阅读此书；如果你对 Objective-C、Swift 的掌握还不熟练，比如对 Xcode 安装、Objective-C、Swift 基本语法都不熟练，建议遵从学习规律，循序渐进，暂时不要购买、阅读此书。

2016-5-20

目 录 CONTENTS

第 1 章 多点触摸与手势检测	1	3.2.2 使用 NSUserDefaults 读取、保存应用程序参数.....	57
1.1 响应者链.....	2	3.3 属性列表.....	62
1.2 响应触碰方法	3	实例：备忘录	62
1.3 使用手势处理器 (UIGestureRecognizer)	3	3.4 对象归档和原生 I/O API	66
1.3.1 使用 UITapGestureRecognizer 处理点击手势	4	3.5 使用 SQLite 3 数据库.....	66
1.3.2 使用 UIPinchGestureRecognizer 处理捏合手势	6	3.5.1 iOS 的 SQLite 编程.....	67
实例：通过捏合手势缩放图片.....	7	3.5.2 创建数据库和表.....	71
1.3.3 使用 UIRotationGestureRecognizer 处理旋转手势.....	9	3.5.3 使用 SQL 语句执行查询	71
实例：通过旋转手势旋转图片.....	9	实例：英文生词本	72
1.3.4 使用 UISwipeGestureRecognizer 处理轻扫手势.....	11	3.5.4 使用 sqlite3 工具.....	76
实例：贪食蛇	13	3.5.5 使用 SQLite Manager 管理数据库.....	77
1.3.5 使用 UIPanGestureRecognizer 处理拖动手势.....	19	3.6 使用 Core Data 框架.....	79
1.3.6 使用 UILongPressGestureRecognizer 处理长按手势.....	21	3.6.1 Core Data 简介.....	80
实例：长按添加按钮	22	3.6.2 初始化 Core Data 项目.....	82
1.4 创建和使用自定义手势处理器.....	24	3.6.3 设计实体模型.....	86
1.4.1 开发自定义手势处理器	24	3.6.4 使用 Core Data 实现数据的增、删、改、查.....	88
1.4.2 使用自定义手势处理器	26	3.6.5 管理实体的关联关系.....	95
1.5 本章小结.....	27	实例：图书管理系统.....	95
第 2 章 国际化.....	28	3.7 本章小结.....	102
2.1 iOS 应用国际化的基础	29	第 4 章 多媒体应用开发.....	103
2.1.1 iOS 应用的国际化思路.....	29	4.1 音频和视频的播放	104
2.1.2 国际化支持的语言和国家	31	4.1.1 使用 System Sound Services 播放音效.....	104
2.2 国际化界面设计文件	32	4.1.2 使用 AVAudioPlayer 播放音乐.....	106
2.3 国际化应用程序的显示名称	36	4.1.3 使用 MPMediaPickerController 选择系统音乐.....	110
2.4 国际化图片	39	实例：简单音乐播放器.....	111
2.5 国际化文本	41	4.1.4 使用 AVPlayer 播放视频.....	114
2.6 本章小结.....	43	4.2 使用 AVAudioRecorder 录制音频.....	121
第 3 章 iOS 的数据存储.....	44	4.3 拍照和录制视频.....	125
3.1 应用程序沙盒	45	4.3.1 使用 UIImagePickerController 拍照和录制视频.....	125
3.1.1 获取 Documents 目录	47	4.3.2 使用 AVFoundation 拍照和录制视频	130
3.1.2 获取 tmp 目录.....	47	实例：完全自定义的相机	132
3.1.3 文件保存策略思考	47	4.4 使用 AVFoundation 生成视频缩略图	143
3.2 应用程序参数与用户默认设置.....	48	4.5 本章小结.....	145
3.2.1 使用 Settings Bundle.....	48	第 5 章 管理手机.....	146
		5.1 使用 iOS 9 新增的 Contacts 管理联系人.....	147

5.1.1 查询联系人	151	7.3 线程同步与线程通信	218
5.1.2 删除联系人	154	7.3.1 线程安全问题	218
5.1.3 添加联系人	155	7.3.2 使用同步代码块实现同步	220
5.1.4 修改联系人	157	7.3.3 释放对同步监视器的锁定	223
5.2 使用 iOS 9 的 ContactsUI 管理联系人	163	7.3.4 同步锁 (NSLock)	223
5.2.1 使用 CNContactViewController 添加联系人	164	7.3.5 使用 NSCondition 控制线程通信	224
5.2.2 使用 CNContactViewController 显示未知联系人	165	实例：生产者-消费者	224
5.2.3 使用 CNContactPickerViewController 选择联系人	167	7.4 使用 GCD 实现多线程	228
5.3 使用 UIApplication 打电话、发短信	169	7.4.1 创建队列	229
5.4 使用 MFMessageComposeViewController 发送短信	171	7.4.2 异步提交任务	230
5.5 使用 MFMailComposeViewController 发 送邮件	174	实例：使用 GCD 下载图片	233
5.6 本章小结	178	7.4.3 同步提交任务	234
第 6 章 加速计与陀螺仪	179	7.4.4 多次执行的任务	235
6.1 iOS 支持的加速计和陀螺仪	180	7.4.5 只执行一次的任务	237
6.1.1 iOS 加速计和陀螺仪的理论基础	180	7.5 后台运行	238
6.1.2 基于代码块方式获取加速度数 据、陀螺仪数据、磁场数据	181	7.5.1 进入后台时释放内存	238
6.1.3 主动请求获取加速度数据、陀 螺仪数据、磁场数据	184	实例：疯狂飞机大战	238
实例：怪物足球	187	7.5.2 进入后台时保存状态	245
6.2 感知设备移动	192	7.5.3 请求更多的后台时间	246
实例：水平仪	195	7.6 使用 NSOperation 与 NSOperationQueue 实现多线程	248
6.3 健康相关传感器	199	7.6.1 使用 NSInvocationOperation 和 NSBlockOperation	249
6.3.1 使用 CMMotionActivityManager	199	实例：使用 NSBlockOperation 下载图片 ...	249
6.3.2 使用 CMPedometer 获取步行数据	201	7.6.2 定义 NSOperation 子类	251
6.4 使用 CMAltimeter 获取高度改变信息	203	7.7 本章小结	253
6.5 iOS 9 新增的 CMSensorRecorder	205	第 8 章 iOS 网络编程	254
6.6 本章小结	207	8.1 检测网络状态	255
第 7 章 多线程	208	8.1.1 检查网络状态	255
7.1 线程概述	209	8.1.2 监听网络状态改变	259
7.1.1 线程和进程	209	8.2 使用 CFNetwork 实现 TCP 协议的通信	260
7.1.2 多线程的优势	210	8.2.1 IP 地址与端口号	260
7.2 使用 NSThread 实现多线程	211	8.2.2 TCP 协议基础	261
7.2.1 创建和启动线程	211	8.2.3 使用 CFSocket 实现 TCP 服务器端 ...	262
7.2.2 线程的状态	213	8.2.4 使用 CFSocket 实现 TCP 客户端	266
7.2.3 终止子线程	213	实例：网络聊天程序	268
7.2.4 线程睡眠	215	8.2.5 使用 CocoaAsyncSocket 实现 TCP 客户端	271
实例：使用线程下载网络图片	215	8.3 使用 NSURLSession	275
7.2.5 改变线程优先级	217	8.3.1 使用 NSURLSession 从网络获 取数据	275
		8.3.2 使用 NSMutableURLRequest 向服 务器发送数据	280
		8.4 XML 解析	282
		8.4.1 DOM 与 SAX	282

8.4.2	使用 NSXMLParser 解析 XML 文档...	284	10.3.1	添加简单的锚点	376
8.4.3	使用 libxml2 解析 XML 文档	289	10.3.2	添加自定义锚点	380
8.4.4	使用 KissXML 解析 XML 文档	293	10.4	在地图上添加覆盖层	383
8.4.5	使用 KissXML 生成、修改 XML 文档	296	10.4.1	添加几何覆盖层	384
8.5	JSON 解析	299	10.4.2	添加 MKTileOverlay 覆盖层	386
8.5.1	JSON 的基本知识	299	10.5	使用 MKDirections 获取导航路线	389
8.5.2	使用 NSJSONSerialization 处理 JSON 数据	302	实例：行车导航仪	389	
8.5.3	使用 SBJson 解析 JSON 数据	302	10.6	本章小结	393
8.6	使用 AFNetworking 实现网络通信	307	第 11 章	消息推送与 XMPP 即时通信	394
8.6.1	提交 GET 请求与提交 POST 请求	307	11.1	使用 NSNotificationCenter 通信	395
实例：访问被保护资源	308	11.1.1	使用 NSNotificationCenter 监听 系统组件的通知	396	
8.6.2	处理 JSON 或 Plist 响应	312	11.1.2	使用 NSNotificationCenter 监听 自定义通知	398
8.6.3	处理 XML 响应	315	11.2	iOS 本地通知	400
8.6.4	上传文件	317	11.3	iOS 远程推送通知	405
8.7	使用 ASIHTTPRequest 框架实现网络编程	320	11.3.1	开发远程推送客户端应用	407
8.7.1	发送同步或异步的 GET 请求	321	11.3.2	开发推送通知的服务端程序	415
8.7.2	使用代码块	325	11.4	基于 XMPP 的即时通信	422
8.7.3	使用 NSOperationQueue 管理请求	326	11.4.1	XMPP 简介	422
8.7.4	发送请求参数和文件上传	328	11.4.2	下载和安装 ejabberd	424
实例：天气预报	330	11.4.3	下载和安装 XMPPFramework ...	427	
8.8	本章小结	340	实例：即时通信 App	428	
第 9 章	使用 CoreLocation 定位	341	11.4.4	注册登录	429
9.1	iOS 的定位支持	342	11.4.5	查询好友列表	434
9.1.1	GPS 卫星定位	342	11.4.6	好友上线、下线	441
9.1.2	基站定位	343	11.4.7	添加、删除好友	442
9.1.3	WiFi 定位	343	11.4.8	查询聊天室列表	443
9.2	获取定位信息	343	11.4.9	创建、加入聊天室	447
9.2.1	iOS 9 增强的后台定位	344	11.4.10	接收、发送聊天消息	449
9.2.2	使用 iOS 模拟器模拟位置	349	11.5	本章小结	459
9.2.3	监控行车速度和行车距离	349	第 12 章	iCloud 服务	460
9.3	方向监测	352	12.1	iCloud 入门	461
实例：指南针	353	12.1.1	为应用开启 iCloud 服务	462	
9.4	区域监测	355	12.1.2	使用 NSMetadataQuery 查询 文档	464
9.5	本章小结	358	12.1.3	添加文档	469
第 10 章	使用 MapKit 开发地图服务	359	12.1.4	编辑文档	472
10.1	使用 MapKit 框架	360	12.1.5	删除文档	474
10.1.1	使用 MKMapView 控件	361	12.2	使用 iCloud 保存云端首选项	475
10.1.2	指定地图显示中心和显示区域	362	12.3	使用 CloudKit 保存数据	479
10.1.3	在地图上使用 MKMapCamera	366	12.3.1	设计云端数据库	479
10.2	根据地址定位	367	12.3.2	查询云端记录	481
10.2.1	地址解析与反向地址解析	367	实例：云端图书管理	482	
10.2.2	根据地址定位	373	12.3.3	删除云端记录	485
10.3	在地图上添加锚点	376			

12.3.4	添加云端记录	486	14.1.1	系统功能简介	525
12.3.5	修改云端记录	488	14.1.2	系统架构设计	525
12.4	本章小结	490	14.2	发送请求的工具类	526
第 13 章	HealthKit 框架	491	14.3	应用界面设计	528
13.1	HealthKit 简介	492	14.4	显示热卖商品	531
13.1.1	HealthKit 常识	492	14.4.1	热卖商品的服务器端接口	532
13.1.2	HealthKit 的基础 API	493	14.4.2	加载显示热卖商品	532
13.1.3	HealthKit 的数据简介	494	14.5	显示商品详情	540
13.2	读取特征数据	494	14.5.1	显示商品	540
13.2.1	请求获取授权	495	14.5.2	加入购物车	544
13.2.2	读取特征数据	499	14.6	处理订单	547
13.2.3	查询样本数据	503	14.6.1	处理购物车	547
13.2.4	添加样本数据	509	14.6.2	确认订单	553
13.3	操作锻炼数据	510	14.6.3	微信支付	558
13.3.1	查询锻炼数据	511	14.7	分类浏览	570
13.3.2	添加锻炼数据	515	14.7.1	商品类别的服务器端接口	570
13.4	本章小结	523	14.7.2	加载、显示商品分类	570
第 14 章	疯狂软件商城 App	524	14.7.3	加载、显示指定分类的商品	573
14.1	系统功能简介和架构设计	525	14.8	商家介绍	576
			14.9	本章小结	578

第 1 章

多点触摸与手势检测

本章要点

- 响应者链
- 响应触碰方法
- 使用 UITapGestureRecognizer 处理点击手势
- 使用 UIPinchGestureRecognizer 处理捏合手势
- 使用 UIRotationGestureRecognizer 处理旋转手势
- 使用 UISwipeGestureRecognizer 处理轻扫手势
- 使用 UIPanGestureRecognizer 处理拖动手势
- 使用 UILongPressGestureRecognizer 处理长按手势
- 自定义手势处理器的开发思路
- 创建和使用自定义手势处理器

多点触碰是 iPhone、iPad 设备的突出优点,有了这种多点触碰的支持,用户可以通过手指在屏幕上触碰、划过等动作来操作 iOS 应用。用户手指在屏幕上触碰、划过等行为,被统称为手势。iOS 应用通过检测用户手势,并针对用户手势做出响应,即可让 iOS 应用与用户手势交互。对于 iOS 应用来说,用户手势是一种非常重要的“输入方式”,尤其是在各种 iOS 游戏中,通过手势检测可让应用的操作方式更加多样化。

本章将会详细介绍 iOS 的手势处理器。本章将从响应者链、多点触碰等基础理论开始介绍,并通过丰富的示例介绍 iOS 内置的各种手势处理器。不仅如此,本章还会教读者如何开发自定义手势处理器。

1.1 响应者链

只要继承了 UIResponder 的对象就可作为事件的响应者,实际上 UIControl 继承了 UIView,UIView 又继承了 UIResponder,由此可见,所有的 UI 控件都可作为事件的响应者。

当用户与某个控件交互时,该控件将作为“第一响应者(First Responder)”,第一响应者将作为响应者链的开始,该事件首先被发送给第一响应者(也就是用户触摸屏幕的控件)。事件将沿着响应者链一直向下传播,直到被某个响应者处理。

通常来说,第一响应者都是 UIView 控件或 UIView 子类控件,当用户触摸该控件后,事件最先由该控件本身处理;如果该控件自身不处理事件,事件就会被传播到它对应的视图控制器(如果存在);如果该视图控制器不处理该事件,事件就会传播到包含该控件的父 UIView 控件(如果存在);如果该父 UIView 控件不处理该事件,事件就会传播到父 UIView 控件对应的视图控制器(如果存在)……直到顶层视图对应的视图控制器。

如果事件从第一响应者开始传播,一直传播到应用界面的顶层视图的视图控制器,该事件依然没有得到处理,接下来该事件就会传播到应用程序窗口(UIWindow 也继承了 UIView)对象;如果应用程序窗口也不处理该事件,该事件就会传播到 UIApplication(它也继承了 UIResponder);如果该 UIApplication 还不处理该事件,该事件就会传播到应用程序委托对象(使用 Xcode 的项目模板生成的所有 iOS 项目中的应用程序委托类都继承了 UIResponder)。

事件响应者链的典型传播路线如下:

First Responder → First Responder 的视图控制器(如果有) → 父容器(如果有) → 父容器的视图控制器(如果有) → UIWindow → UIApplication → 应用程序委托对象

如果某个事件经过上面完整的传播过程,依然没有被处理,那么该事件就会被丢弃。一般来说,上面响应者链的任何响应者处理该事件,该事件就会停止传播。

如果某个响应者“截获”了某个事件,那么该响应者要根据条件决定是否处理该事件。当响应者无法处理该事件时,则需要在处理方法中手动传递该事件。例如如下代码:

```
func handleTapEvent(event: UIEvent) {
    if(condition) {
        // 处理事件
    } else {
        self.nextResponder.handleTapEvent(event)
    }
}
```

上面粗体字代码手动调用了下一个响应者的 handleTapEvent:方法,这样即可手动将事件传播给下一个响应者。

1.2 响应触碰方法

如果希望自定义控件可以响应用户的触碰事件,则可以通过 `UIResponder` 的如下 4 个方法来实现。

- `- touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event`: 当用户手指开始接触控件或窗口事件时激发该方法。
- `- touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event`: 当用户手指在控件上移动时激发该方法。
- `- touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event`: 当用户手指结束触碰控件时激发该方法。
- `- touchesCancelled:(NSSet *)touches withEvent:(UIEvent *)event`: 当系统事件(比如内存低事件)中止了触碰事件时激发该方法。

如果希望用户在控件上按下手指激发相应的事件,则可通过重写如下方法来实现。

```
func touchesBegan(touches: Set<UITouch>, withEvent event: UIEvent?) {  
    let fingersNum = touches.count  
    let tapCount = touches.popFirst()?.tapCount  
    ...  
}
```

重写这些触碰事件处理方法时,第 1 个 `Set` 类型的参数代表了用户同时触碰控件的多个手指,如果该用户用 3 个手指触碰了该控件,那么该 `touches` 参数中将会包含 3 个 `UITouch` 对象。

`UITouch` 对象代表一个触碰事件,该对象提供了一个 `tapCount` 属性,该属性用于返回用户触碰屏幕的次数,比如用户只是轻轻地触碰屏幕一次,那么该属性将会返回 1。如果该属性返回 3,则代表用户快速触碰了 3 次屏幕。如果 `touches` 集合包含 3 个元素,而且 `tapCount` 属性返回 3,就代表用户使用 3 个手指触碰了 3 次控件。除此之外,`UITouch` 还提供了 `locationInView` 方法来获取该触碰事件在 `UIView` 控件中的触碰位置。

至于获取 `UITouch` 事件之后程序应该针对该触碰事件进行何种处理,则完全取决于应用的业务需要。

除此之外,iOS 为手指触碰事件提供了手势处理器,通过手势处理器可使用一致的编程模式来处理各种触碰事件,而且编程更加简单,因此一般推荐用户使用手势处理器来处理用户触碰事件。

1.3 使用手势处理器 (UIGestureRecognizer)

通过手势处理器处理用户触碰事件更加简单,而且无论处理哪种触碰手势,都可面向 `UIGestureRecognizer` 编程,`UIGestureRecognizer` 提供了如下子类。

- `UITapGestureRecognizer`: 处理用户点击手势的手势处理器。
- `UIPinchGestureRecognizer`: 处理用户捏合手势的手势处理器。
- `UIRotationGestureRecognizer`: 处理用户旋转手势的手势处理器。
- `UISwipeGestureRecognizer`: 处理用户滑动手势的手势处理器。
- `UIPanGestureRecognizer`: 处理用户拖动手势的手势处理器。
- `UIScreenEdgePanGestureRecognizer`: 处理用户在屏幕边缘的拖动手势的手势处理器。
- `UILongPressGestureRecognizer`: 处理用户长按手势的手势处理器。

使用手势处理器处理用户触碰手势的编程步骤如下。