

疯狂 iOS讲义(下)

iPhone/iPad
高级应用与手游开发

李刚 肖文吉 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





疯程

iOS讲义(下)

iPhone/iPad

高级应用与手游开发

李刚 肖文吉 编著



电子工业出版社

Publishing House of Electronics Industry

北京•BEIJING

内 容 简 介

本书以 iOS 7.0、Xcode 5.0 为平台,全面介绍了 iOS 应用开发的高级知识。本书最后两章还通过 3 个实际的手游介绍了 cocos2d 框架的用法,也包括了手游开发中用到的瓦片地图、粒子效果和物理引擎等实用技术。本书是《疯狂 iOS 讲义》下册,书中不包含 Objective-C 编程的基本语法、iOS 应用开发的基础知识等内容。本书内容包含多点触摸与手势检测、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、定位、地图、推送机制、iCloud 服务等内容,本书最后还通过疯狂打飞机、萌仙、愤怒的小鸟等实例介绍了 cocos2d 框架的功能和用法。读者在阅读本书之前,建议先掌握《疯狂 iOS 讲义》上册中的知识。

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”的角度来讲解理论。全书一共包括近百个实例,这些示范性的实例既可帮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了技术问题,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。但如果只是想简单涉猎 iOS 应用开发,则本书过于庞大,不适合阅读。

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. 下, iPhone/iPad 高级应用与手游开发/李刚, 肖文吉编著. —北京: 电子工业出版社, 2014.3
ISBN 978-7-121-22437-9

I. ①疯… II. ①李… ②肖… 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(2014)第 019929 号

策划编辑: 张月萍

责任编辑: 葛 娜

印 刷: 北京中新伟业印刷有限公司

装 订: 三河市皇庄路通装订厂

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: 39.25 字数: 1134千字 彩插: 2

印 次: 2014年3月第1次印刷

印 数: 4000册 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。

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

服务热线: (010) 88258888。



前言

《疯狂 iOS 讲义》下册来了——作为上册的延续，本书的内容更加深入，更加“有趣”，如果你已经挺过了上册中 Objective-C 语法的介绍、iOS 界面开发中繁多的 UI 控件，恭喜你顺利进入下册学习了。本书内容包括《萌仙》模块、《打飞机》、《贪食蛇》、《天气预报》、《愤怒的小鸟》等大量精彩的 iOS 应用和游戏，本书不仅会细致地介绍开发这些应用和游戏所需的基础知识，而且会真正带领读者从零开始，逐步完成这些应用和游戏。如果读者认真学完本书，并掌握书中内容，将可以完全胜任 iOS 应用和游戏开发。

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》上册内容的读者来说，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应用、游戏感兴趣，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，则可能有以下三种情况：

- 技术本身就简单，只要简单的介绍即可。
- 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可以说清的一门技术，而《疯狂 iOS 讲义》并不打算只归纳该技术的要点，而是力求全面、细致地介绍 iOS 开发的相关内容，《疯狂 iOS 讲义》包括了两大块知识：Objective-C 语言和 iOS 平台的应用开发，因此不得不将《疯狂 iOS 讲义》分为上、下册（最初打算用一本书介绍完，但后来发现这个想法做不到）。

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，则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。

有一点小小的遗憾：出于篇幅限制，《疯狂 iOS 讲义》下册中没有包括 SpriteKit（iOS 7 新增的游戏引擎）相关内容，除此之外，本书基本全面介绍了 iOS 7 新增的 API，这是考虑到目前 iOS 游戏开发者普遍依然采用 cocos2d，因此本书暂时没有涉及 SpriteKit 相关内容。

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的图书数量少——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，每个国内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，如果你坚持认为：国外图书具有更好的“思想”，那么你胜了——因为“思想”这东西，太难以把握了，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 Objective-C 编程，想在 iPhone、iPad 上开发应用，则可以选择本书；如果读者想通过编程进而悟道，领悟人生思想，那么应该放弃选择本书。

本书有什么特点



本书以《疯狂 iOS 讲义》上册为基础，本书中不包含 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检测、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、定位、地图、推送机制、iCloud 服务等内容，本书还通过疯狂打飞机、萌仙、愤怒的小鸟等实例介绍了 cocos2d 框架的功能和用法。

本书大部分章节由李刚执笔，本书最后两章由疯狂软件教育中心的肖文吉老师执笔，最后由李刚统稿。

本书保持了“疯狂体系”的一贯风格：操作步骤详细，编程思路清晰，语言平实。认真看完本书，把书中所有示例都练习一遍，本书带给你的只是 9 个字：“看得懂、学得会、做得出”。本书不会让你认识一堆新名词，只会让你学会实际的 iOS 应用开发。只要读者认真阅读本书，并掌握书中知识，那么就完全可以胜任企业中 iOS 应用开发、iOS 手机游戏开发的要求。

读者在阅读本书时如果遇到知识上的问题，则可以登录疯狂 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交流、学习。

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 图书中内容最全面、体系最庞大的。本书上、下册全方位地涵盖了 Objective-C 语法、iOS 开发基础知识，iOS 开发高级知识、cocos2d 游戏开发框架等内容。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。全书有近百个实例，几乎每个知识点都可以找到对应的参考实例。书中还提供了天气预报、疯狂打飞机、萌仙、愤怒的小鸟等真实的应用和手游，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，认真掌握本书上、下册的知识，即可完全胜任实际企业中 iOS 开发的要求。

本书写给谁看



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



2014-1-8

博文视点诚邀精锐作者加盟

十载耕耘奠定专业地位

以书为证彰显卓越品质

《C++Primer (中文版) (第5版)》、《淘宝技术这十年》、《代码大全》、《Windows内核情景分析》、《加密与解密》、《编程之美》、《VC++深入详解》、《SEO实战密码》、《PPT演义》……

“圣经”级图书光耀夺目,被无数读者朋友奉为案头手册传世经典。

潘爱民、毛德操、张亚勤、张宏江、咎辉Zac、李刚、曹江华……

“明星”级作者济济一堂,他们的名字熠熠生辉,与IT业的蓬勃发展紧密相连。

十年的开拓、探索和励精图治,成就博古通今、文圆质方、视角独特、点石成金之计算机图书的风向标杆:博文视点。

“凤翱翔于千仞兮,非梧不栖”,博文视点欢迎更多才华横溢、锐意创新的作者朋友加盟,与大师并列于IT专业出版之巔。

英雄帖

江湖风云起,代有才人出。

IT界群雄并起,逐鹿中原。

博文视点诚邀天下技术英豪加入,

指点江山,激扬文字

传播信息技术,分享IT心得

• 专业的作者服务 •

博文视点自成立以来一直专注于IT专业技术图书的出版,拥有丰富的与技术图书作者合作的经验,并参照IT技术图书的特点,打造了一支高效运转、富有服务意识的编辑出版团队。我们始终坚持:

善待作者——我们会把出版流程整理得清晰简明,为作者提供优厚的稿酬服务,解除作者的顾虑,安心写作,展现出最好的作品。

尊重作者——我们尊重每一位作者的技术实力和生活习惯,并会参照作者实际的工作、生活节奏,量身制定写作计划,确保合作顺利进行。

提升作者——我们打造精品图书,更要打造知名作者。博文视点致力于通过图书提升作者的个人品牌和技术影响力,为作者的事业开拓带来更多的机会。



联系我们

博文视点官网: <http://www.broadview.com.cn>

CSDN官方博客: <http://blog.csdn.net/broadview2006/>

投稿电话: 010-51260888 88254368

投稿邮箱: jsj@phei.com.cn



@博文视点Broadview



微信公众账号 博文视点Broadview



博文视点精品图书展台

专业典藏



移动开发



大数据 · 云计算 · 物联网



数据库



Web 开发



程序设计



软件工程



办公精品



网络营销



第 1 章 多点触摸与手势检测	1	3.1.2 获取 tmp 目录	41
1.1 响应者链	2	3.1.3 文件保存策略思考	41
1.2 响应触碰方法	3	3.2 应用程序参数与用户默认设置	42
1.3 使用手势处理器 (UIGestureRecognizer)	3	3.2.1 使用 Settings Bundle	42
1.3.1 使用 UITapGestureRecognizer 处理点击手势	4	3.2.2 使用 NSUserDefaults 读取、 保存应用程序参数	50
1.3.2 使用 UIPinchGestureRecognizer 处理捏合手势	6	3.3 属性列表	54
实例：通过捏合手势缩放图片	7	实例：备忘录	55
1.3.3 使用 UIRotationGestureRecognizer 处理旋转手势	9	3.4 对象归档和原生 I/O API	58
实例：通过旋转手势旋转图片	9	3.5 使用 SQLite 3 数据库	58
1.3.4 使用 UISwipeGestureRecognizer 处理轻扫手势	10	3.5.1 iOS 的 SQLite 编程	58
实例：贪食蛇	12	3.5.2 创建数据库和表	63
1.3.5 使用 UIPanGestureRecognizer 处理拖动手势	18	3.5.3 使用 SQL 语句执行查询	63
1.3.6 使用 UILongPressGestureRecognizer 处理长按手势	19	实例：英文生词本	64
实例：长按添加按钮	20	3.5.4 使用 sqlite3 工具	67
1.4 创建和使用自定义手势处理器	21	3.5.5 使用 SQLite Manager 管理数 据库	69
1.4.1 开发自定义手势处理器	21	3.6 使用 Core Data 框架	71
1.4.2 使用自定义手势处理器	23	3.6.1 Core Data 简介	71
1.5 本章小结	24	3.6.2 初始化 Core Data 项目	73
第 2 章 国际化	25	3.6.3 设计实体模型	76
2.1 iOS 应用国际化的基础	26	3.6.4 使用 Core Data 实现数据的增、 删、改、查	78
2.1.1 iOS 应用的国际化思路	26	3.6.5 管理实体的关联关系	84
2.1.2 Objective-C 国际化支持的语言 和国家	28	实例：图书管理系统	84
2.2 国际化界面设计文件	29	3.7 本章小结	95
2.3 国际化应用程序的显示名称	32	第 4 章 多媒体应用开发	96
2.4 国际化图片	34	4.1 音频和视频的播放	97
2.5 国际化文本	35	4.1.1 使用 System Sound Services 播 放音效	97
2.6 本章小结	38	4.1.2 使用 AVAudioPlayer 播放音乐	98
第 3 章 iOS 的数据存储与 IO	39	4.1.3 使用 MPMediaPickerController 选择系统音乐	102
3.1 应用程序沙盒	40	实例：简单音乐播放器	103
3.1.1 获取 Documents 目录	41	4.1.4 使用 MPMoviePlayerController 播放视频	106
		4.2 使用 AVAudioRecorder 录制音频	108
		4.3 拍照和录制视频	112
		4.3.1 使用 UIImagePickerController 拍照和录制视频	112

4.3.2 使用 AVFoundation 拍照和录 制视频	116	7.1.1 线程和进程	190
实例：完全自定义的相机	118	7.1.2 多线程的优势	191
4.4 使用 AVFoundation 生成视频缩 略图	135	7.2 使用 NSThread 实现多线程	192
4.5 本章小结	137	7.2.1 创建和启动线程	192
第 5 章 管理手机	138	7.2.2 线程的状态	194
5.1 使用 AddressBook 管理联系人	139	7.2.3 终止子线程	194
5.1.1 删除联系人	142	7.2.4 线程睡眠	195
5.1.2 添加联系人	144	实例：使用线程下载网络图片	196
5.1.3 修改联系人	147	7.2.5 改变线程优先级	197
5.2 使用 AddressBookUI 管理联系人	151	7.3 线程同步与线程通信	199
5.2.1 使用 ABNewPersonViewController 添加联系人	153	7.3.1 线程安全问题	199
5.2.2 使用 ABUnknownPersonViewController 显示未知联系人	154	7.3.2 使用@synchronized 实现同步	201
5.2.3 使用 ABPeoplePickerNavigationController 选择联系人	156	7.3.3 释放对同步监视器的锁定	203
5.2.4 使用 ABPersonViewController 显示指定联系人	157	7.3.4 同步锁 (NSLock)	203
5.3 使用 UIApplication 打电话、发短信	159	7.3.5 使用 NSCondition 控制线程 通信	205
5.4 使用 MFMessageComposeViewController 发短信	160	实例：生产者-消费者	205
5.5 使用 MFMailComposeViewController 发送邮件	162	7.4 使用 GCD 实现多线程	208
5.6 本章小结	165	7.4.1 创建队列	209
第 6 章 加速计与陀螺仪	166	7.4.2 异步提交任务	210
6.1 iOS 支持的加速计和陀螺仪	167	实例：使用 GCD 下载图片	212
6.1.1 iOS 加速计和陀螺仪的理论 基础	167	7.4.3 同步提交任务	213
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	168	7.4.4 多次执行的任务	214
6.1.3 主动请求获取加速度数据、陀 螺仪数据、磁场数据	172	7.4.5 只执行一次的任务	215
实例：怪物足球	173	7.5 后台运行	216
6.2 感知设备移动	178	7.5.1 进入后台时释放内存	216
实例：水平仪	181	实例：疯狂飞机大战	216
6.3 iOS 7 新增的计步器	184	7.5.2 进入后台时保存状态	223
6.4 iOS 7 新增的 CMMotionActivityManager	186	7.5.3 请求更多的后台时间	223
6.5 本章小结	188	7.6 使用 NSOperation 与 NSOperationQueue 实现多线程	225
第 7 章 多线程	189	7.6.1 使用 NSInvocationOperation 和 NSBlockOperation	226
7.1 线程概述	190	实例：使用 NSBlockOperation 下载 图片	226
		7.6.2 定义 NSOperation 子类	227
		7.7 本章小结	229
		第 8 章 iOS 网络应用编程	230
		8.1 检测网络状态	231
		8.1.1 检查网络状态	231
		8.1.2 监听网络状态改变	234
		8.2 使用 CFNetwork 实现 TCP 协议的通信	235
		8.2.1 IP 地址与端口号	236
		8.2.2 TCP 协议基础	236

8.2.3 使用 CFSocket 实现 TCP 服务 器端	237	第 9 章 使用 CoreLocation 定位	312
8.2.4 使用 CFSocket 实现 TCP 客 户端	241	9.1 iOS 的定位支持	313
实例：网络聊天程序	244	9.1.1 GPS 卫星定位	313
8.2.5 使用 CocoaAsyncSocket 实现 TCP 客户端	247	9.1.2 基站定位	314
8.3 使用 NSURLConnection	250	9.1.3 WiFi 定位	314
8.3.1 使用 NSURLConnection 从网 络获取数据	250	9.2 获取定位信息	314
8.3.2 使用 NSMutableURLRequest 向服务器发送数据	253	9.2.1 获取位置信息	315
8.4 XML 解析	255	9.2.2 使用 iOS 模拟器模拟位置	318
8.4.1 DOM 与 SAX	255	9.2.3 监控行车速度和行车距离	318
8.4.2 使用 NSXMLParser 解析 XML 文档	257	9.3 方向监测	320
8.4.3 使用 libxml2 解析 XML 文档	262	实例：指南针	322
8.4.4 使用 GDataXML 解析 XML 文档	266	9.4 区域监测	323
8.4.5 使用 GDataXML 生成、修改 XML 文档	268	9.5 本章小结	325
8.5 JSON 解析	271	第 10 章 使用 MapKit 开发地图服务	326
8.5.1 JSON 的基本知识	271	10.1 使用 MapKit 框架	327
8.5.2 使用 NSJSONSerialization 处理 JSON 数据	274	10.1.1 使用 MKMapView 控件	328
8.5.3 使用 SBJson 解析 JSON 数据	274	10.1.2 指定地图显示中心和显示 区域	329
8.5.4 使用 JSONKit 解析 JSON 数据	275	10.1.3 使用 iOS 7 新增的 MKMapCamera	332
8.6 使用 AFNetworking 实现网络通信	280	10.2 根据地址定位	333
8.6.1 提交 GET 请求与提交 POST 请求	280	10.2.1 地址解析与反向地址解析	333
实例：访问被保护资源	281	10.2.2 根据地址定位	337
8.6.2 处理 JSON 或 Plist 响应	284	10.3 在地图上添加锚点	340
8.6.3 处理 XML 响应	288	10.3.1 添加简单的锚点	340
8.6.4 上传文件	290	10.3.2 添加自定义锚点	343
8.7 使用 ASIHTTPRequest 框架实现网络 编程	293	10.4 在地图上添加覆盖层	345
8.7.1 发送同步或异步的 GET 请求	293	10.4.1 添加几何覆盖层	346
8.7.2 使用代码块	296	10.4.2 使用 iOS 7 新增的 MKTileOverlay 覆盖层	348
8.7.3 使用 NSOperationQueue 管理 请求	297	10.5 使用 iOS 7 新增的 MKDirections 获 取导航路线	350
8.7.4 发送请求参数和文件上传	299	实例：行车导航仪	351
实例：天气预报	300	10.6 本章小结	354
8.8 本章小结	311	第 11 章 推送机制	355
		11.1 使用 NSNotificationCenter 通信	356
		11.1.1 使用 NSNotificationCenter 监听系统组件的通知	358
		11.1.2 使用 NSNotificationCenter 监听自定义通知	359
		11.2 iOS 本地通知	361
		11.3 iOS 远程推送通知	364
		11.3.1 开发 Push 客户端应用	366

11.3.2	开发 Push 服务端程序	377	13.7.1	CCDirector 导演类的属性	430
11.4	本章小结	381	13.7.2	CCDirector 导演类的方法	430
第 12 章	iCloud 服务与应用	382	13.8	CCTexture 纹理类	431
12.1	iCloud 入门	383	13.8.1	纹理的相关概念	431
12.1.1	为应用开启 iCloud 服务	384	13.8.2	纹理相关的类	431
12.1.2	使用 NSMetadataQuery 查询 文档	389	13.9	CCSprite 精灵类	432
12.1.3	添加文档	393	13.9.1	CCSprite 精灵类的属性	432
12.1.4	编辑文档	395	13.9.2	CCSprite 精灵类的方法	432
12.1.5	删除文档	397	13.9.3	CCSpriteFrame 精灵帧	435
12.2	使用 iCloud 保存云端首选项	398	13.9.4	CCSpriteFrameCache 精灵 帧缓存	436
12.3	本章小结	401	13.9.5	CCSpriteBatchNode 精灵表单	436
第 13 章	使用 cocos2d 开发 2D 游戏	402	13.9.6	制作精灵表单	438
13.1	cocos2d 简介	403	13.9.7	精灵表单示例	440
13.1.1	cocos2d 的特点	403	13.10	cocos2d 中的动作	442
13.1.2	cocos2d 的主要功能	403	13.10.1	cocos2d 中的动作类	443
13.2	cocos2d 的下载与安装	404	13.10.2	基本动作	444
13.2.1	下载 cocos2d	404	13.10.3	组合动作	453
13.2.2	安装 cocos2d	405	13.10.4	Ease 动作	454
13.2.3	使用 cocos2d-iPhone 的帮助 文档	405	13.10.5	延迟动作	456
13.2.4	cocos2d 中内置的项	406	13.10.6	方法回调动作	456
13.3	创建 cocos2d 项目	407	13.10.7	代码块调用动作	460
13.3.1	HelloWorld cocos2d 项目	407	13.11	cocos2d 中的动画	462
13.3.2	cocos2d 项目如何支持 ARC	408	13.11.1	cocos2d 中动画相关的类	462
13.3.3	cocos2d 项目结构和代码 分析	413	13.11.2	简单动画效果	463
13.4	CCNode 节点类	419	13.11.3	使用精灵表单实现动画效果	464
13.4.1	CCNode 类的属性	419	13.12	cocos2d 中的文本	466
13.4.2	CCNode 类的方法	420	13.12.1	CCLabelTTF 类	466
13.5	CCScene 场景类	423	13.12.2	CCLabelBMFont 类	467
13.5.1	CCTransitionScene 场景 切换	423	13.13	cocos2d 中的声音	470
13.5.2	cocos2d 支持的场景过渡 效果	424	13.13.1	SimpleAudioEngine	470
13.6	CCLayer 类	424	13.13.2	游戏中的声音设置选项 功能实现	474
13.6.1	CCLayer 类的属性	425	13.14	疯狂打飞机游戏	477
13.6.2	CCLayer 类的常用方法	425	13.14.1	开始前的准备工作	477
13.6.3	CCLayer 类的作用	425	13.14.2	添加游戏菜单项功能	478
13.6.4	CCLayerColor 类的属性和 方法	425	13.14.3	预加载游戏资源	480
13.6.5	CCLayerGradient 渐变色层类	426	13.14.4	玩家飞机飞行效果	485
13.6.6	CCMenu 菜单类	426	13.14.5	玩家飞机 Touch 实现	487
13.7	CCDirector 导演类	429	13.14.6	背景滚动效果	488
			13.14.7	添加敌机	491
			13.14.8	玩家飞机添加子弹并射击	494
			13.14.9	添加背景音乐	498
			13.14.10	添加游戏积分统计	498
			13.14.11	添加游戏大 Boss	500
			13.15	本章小结	504

第 14 章 粒子效果、瓦片地图和物理引擎..... 505

14.1	粒子系统相关的类	506
14.1.1	CCParticleSystem 类	506
14.1.2	CCParticleSystemQuad 类	511
14.1.3	CCParticleBatchNode 类	512
14.2	cocos2d 中内置的粒子系统	512
14.2.1	使用 cocos2d 内置的粒子系统	513
14.2.2	手动创建粒子系统类	514
14.3	使用 Particle Designer 生成粒子效果 ...	517
14.3.1	Particle Designer 粒子工具的使用	518
14.3.2	使用 plist 文件创建粒子系统	520
14.4	瓦片地图	520
14.4.1	下载和安装 Tiled 地图编辑器	521
14.4.2	使用 Tiled 绘制地图	522
14.5	Tiled 地图相关的类	527
14.5.1	CCTMXTiledMap 类	527
14.5.2	CCTMXLayer 类	529
14.5.3	CCTMXObjectGroup 类	530
14.6	在项目中使用 Tiled 地图	530
14.7	真实手游：萌仙	535
14.7.1	设计地图场景	535
14.7.2	载入地图	535
14.7.3	地图的移动	537
14.7.4	物体遮挡效果	539

14.7.5	设置障碍物	542
14.7.6	寻路算法	544
14.7.7	随机出现怪物	549
14.7.8	玩家和怪物进行战斗	552
14.8	Box2D 物理引擎	559
14.8.1	物理引擎概述	559
14.8.2	Box2D 的核心概念	559
14.8.3	Box2D 常用的属性和方法	562
14.8.4	Box2D 模板项目 HelloBox2D	563
14.8.5	HelloBox2D 模板项目代码分析	565
14.8.6	cocos2d 和 Box2D	573
14.8.7	Box2D 开发步骤	575
14.9	开发 Box2D 项目	575
14.10	愤怒的小鸟	581
14.10.1	开始前的准备工作	581
14.10.2	制作游戏启动画面	581
14.10.3	进度条制作	582
14.10.4	游戏中的粒子效果	585
14.10.5	游戏选关操作	590
14.10.6	设计关卡数据	594
14.10.7	设计精灵类	596
14.10.8	游戏主界面	601
14.10.9	弹弓发射小鸟	604
14.10.10	整合 Box2D 物理引擎	609
14.10.11	游戏过关设计	613
14.11	本章小结	616

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,所有的对象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:

```
- (void) handleTapEvent:(UIEvent*) event
{
    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。

```
- (void) touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
{
    NSUInteger fingersNum = [touches count];
    NSUInteger tapCount = [[touches anyObject] tapCount]
    ...
}
```

重写这些触碰事件处理方法时, 第 1 个 `NSSet` 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。
- 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。

① 根据程序要处理的手势创建对应的手势处理器对象。创建手势处理器时需要指定 `target` 和 `action` 参数——当该控件上发生触碰手势后, 该 `target` 对象的 `action` 方法将会被激发。

② 如果该 UI 控件不允许用户交互, 则将该 UI 控件的 `userInteractionEnabled` 属性设为 YES; 如果希望该控件可支持多点触碰, 还需要将 `multipleTouchEnabled` 设为 YES。

③ 调用 UI 控件的 `addGestureRecognizer:` 方法添加该手势处理器。

UIGestureRecognizer 作为所有手势处理器的基类, 它提供如下常用的方法和属性。

- `- locationInView:(UIView *)view`: 返回该手势在 `view` 控件中的触碰位置。
- `- locationOfTouch:(NSUInteger)touchIndex inView:(UIView *)view`: 返回该手势中第 `touchIndex` 个触碰点在 `view` 控件中的触碰位置。
- `- numberOfTouches`: 返回该手势包含触碰点的数量 (也就是用户用了几个手指进行触碰)。
- `view`: 返回激发该手势的 UI 控件。
- `enabled`: 用于设置和返回该手势处理器是否可用。
- `state`: 获取该手势所处的状态, 比如手势刚开始时处于 `UIGestureRecognizerStateBegan` 状态, 手势结束时处于 `UIGestureRecognizerStateEnded` 状态。

➤➤ 1.3.1 使用 UITapGestureRecognizer 处理点击手势

UITapGestureRecognizer 还提供了如下两个属性。

- `numberOfTapsRequired`: 指定该手势处理器只处理几次触碰事件。
- `numberOfTouchesRequired`: 指定该手势处理器只处理几个手指的触碰事件。

接下来通过示例示范使用 UITapGestureRecognizer 处理点击手势。创建一个 Single View Application, 该应用包含一个应用程序委托类、一个视图控制器类以及配套的 Main.storyboard 界面设计文件。在 Interface Builder 中打开该界面设计文件, 将一个 UILabel 控件和一个 UIView 控件拖入界面设计文件中, 并将这些控件摆放整齐。应用界面设计效果如图 1.1 所示。

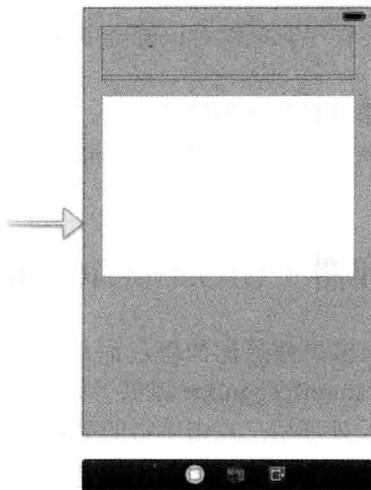


图 1.1 应用界面设计效果