第 1 章

简　介

可曾想象，有朝一日你会自己写一个计算机游戏然后靠它赚钱？有了 Apple 的 iTunes App Store 及其配套移动设备，如 iPhone、iPod Touch 和 iPad，要实现这个梦想不再是一件难事。当然，这并不表示开发一个游戏有多简单，你仍然需要学习很多游戏开发和编程的知识。不过，既然你选择了阅读本书，我就有理由相信你已经下定决心踏上这条游戏开发之路了。恭喜你选择了一个可能是全世界最有趣的游戏开发引擎——iPhone 版 cocos2d。

使用 cocos2d 的开发者可能有很多不同的专业背景，来自不同的领域。有些人(比如我)可能是已经从事游戏开发好几年甚至几十年的专业人士，还有一些人可能是刚开始接触 iOS 平台游戏开发，甚至有些人是刚转行加入这一充满激情的游戏开发领域。不管你属于哪一类人，我保证，看完这本书一定能够有所收获。

是这样一种信念让我们走到一起：我们都热爱游戏，也热爱设计和编写游戏。本书十分推崇这种信念，并会向读者介绍一些能够帮助简化游戏开发过程的工具。最重要的是，这本书会教大家写一些很有借鉴意义的小游戏，从中你可以学会如何把一些理论知识运用到现实的游戏开发中。

有些书会整页整页地教读者怎样用一些特定的游戏编程 API 来写一个无聊的战机类太空游戏(Asteroids)，我读到这种书的时候总是觉得特别没劲。我觉得一本好书应该向大家介绍游戏编程的理念和开发工具，因为这些东西是永恒的，不会随着 API 或编程喜好的变化而变化。我读编程书籍和游戏开发书籍已经有 20 多年了，我认为最有价值的书是那些高于技术本身的，是能够让我明白为什么这个地方会这样设计、这样编程、这样做有什么好处的书。所以本书不仅会关注游戏代码的含义，更会关注它的工作原理以及在哪些处理上需要根据情况权衡利弊。

我希望你能学着写出一些有价值的、能在 App Store 上热卖并且受玩家欢迎的游戏。我会介绍这本书里的示例游戏背后深藏的思想和技术理念，当然，我也会告诉你在游戏编程中如何使用 cocos2d 和 Objective-C。本书源代码中有大量注释，它们可以帮助你正确地理解代码的含义。

学习别人的源代码并且根据注释去关注一些重要设计对我来说是学习新知识的最好方法(我想它对你来说也会是一个很棒的方法)。你可以对这本书的随书源代码加以修改，进而做出自己的游戏。我非常期待在不久的将来能够玩到你的游戏！完成你的游戏千万别忘了告诉我！你可以通过我的邮箱(steffen@learn-cocos2d.com)联系我，也可以登录我的网站(http://www.learn-cocos2d.com)给我留言。

1.1 选择 iPhone 版 cocos2d 的理由

游戏开发者在选择游戏引擎时首先会对他们要选择的产品做一些评估。综合以下因素，我认为 cocos2d 对许多开发者来说会是一个非常棒的选择。

1.1.1 免费

首先，cocos2d 是免费的。不需要花钱就可以用它来进行开发。其次，你可以随心所欲地开发 iPhone、iPod 和 iPad 应用，无论免费还是收费都可以。说真的，这是完全没有附加限制的。

cocos2d 几乎是由 Ricardo Quesada 一个人开发出来的。假如你想资助他对 cocos2d 进行一些后续开发，或者想购买他的一些收费源码项目，可以登录 http://www.cocos2d-iphone.org/store 访问 cocos2d Store。

1.1.2 开源

cocos2d 的第二个好处就是它是开源的，这就意味着可以自由地学习游戏引擎的源代码，或者在需要时对引擎作些改动。可以从 http://www.cocos2d-iphone.org/download 下载源代码。

1.1.3 Objective-C

另外，cocos2d 是用 Objective-C 编写而成的，Objective-C 是苹果公司用于开发 iPhone 应用程序的本地编程语言(native programming language)。由于 iPhone SDK 也是用 Objective-C 编写而成的，因此对于使用 cocos2d 的开发者来说，要理解苹果公司的官方文档和使用 iPhone SDK 提供的 API 并不困难。

其他很多有用的 API，如 Facebook Connect 和 OpenFeint，也是用 Objective-C 编写而成的，所以要集成它们也非常容易。

注意：

相对于 Objective-C 而言，你可能更喜欢别的编程语言，但我还是建议你学习 Objective-C。我原本有很深的 C++和 C#背景，而且 Objective-C 语法乍一看还挺古怪，所以一开始我并不情愿去学这个据说是又陈旧又过时的编程语言。果不其然，有段时间我的日子过得相当挣扎，我必须摒弃已经养成的编程习惯和思维模式才能弄清楚怎样用

Objective-C 来写程序。

但是，千万不要因为困难就放弃学习 Objective-C。你确实需要花点时间去习惯它，但是这样的付出马上就会得到回报(只要教程和文档足够充分)。所以，再多努力都是值得的!

1.1.4 2D 游戏引擎

显然，cocos2d 中的“2d”已经表明了它是一个专注于开发 2D 游戏的引擎，这在当今众多的 iOS 游戏引擎中算是少见的。虽然它也可以用于加载并显示 3D 对象，但是你必须自己写出 3D 渲染代码，否则就只能采用别的方法来加载和显示 3D 模型了。我想说的是，iOS 设备是一个非常理想的 2D 游戏平台。它们通常比较容易开发，也比较容易理解。很多情况下，它们对硬件的要求比较低，因此你可以创建更色彩鲜明、更细致的图形。

1.1.5 物理引擎

目前有两种集成在 cocos2d 里的物理引擎可供选择：Chipmunk 和 Box2D。这两种物理引擎仅仅在编写它们的语言上有一些细微的差别：Chipmunk 是用 C 编写而成的，而 Box2D 是用 C++编写而成的，但它们的功能几乎完全一样。当然，如果仔细比较，还是会发现一些差别。不过，想要根据这些差别来作出选择，就必须对物理引擎的工作机制有非常深入的理解。通常，应该选择一个你觉得比较容易理解的且提供的文档比较好的物理引擎，所以大多数开发者都比较倾向于使用 Box2D。而且，因为 Box2D 也采用了面向对象的思想，所以与 Objective-C 一起使用会比较方便。

1.1.6 技术难度较低

游戏开发者最喜欢 cocos2d 的地方就在于它把底层的 OpenGL ES 代码封装得特别好。大多数图形都是用简单的精灵类(CCSprite)来显示的，而精灵对象又是根据图像文件创建的。也就是说，一个精灵对象就是一个具有缩放、翻转和着色能力的纹理，只要简单地对精灵对象相应的 Objective-C 属性稍作修改就可以完成这些效果。你并不需要关心 OpenGL ES 代码的具体实现，这就是 cocos2d 的美妙之处。

同时，你可以灵活地对任意游戏对象在任意时刻添加自己的 OpenGL ES 代码。而且，Cocoa Touch 中的用户界面元素在 cocos2d 中也是适用的。

cocos2d 引擎不仅封装了 OpenGL ES 的实现细节(这样就不用费尽心思去理解那些错综复杂的步骤了)，还对一些比较通用的操作进行了高度的抽象，其中包括一些实现起来需要大量 iPhone SDK 背景知识的操作。但是，cocos2d 并不会阻止你去接触底层的实现。

1.1.7 依然需要编程

总的来说，cocos2d 确实简化了 iOS 游戏的开发过程，但出色的编程技巧依然是需要掌握的。其他 iOS 游戏引擎，如 Unity、iTorque 和 Shiva，都着重于向用户提供工具箱和工作流来降低对编程能力的要求。使用这些引擎固然方便，但也失去了一些技术上的自由，

而且还得付费。至于 cocos2d，要使用它确实需要花些功夫，但是相比其他引擎，cocos2d 更能凸显出游戏编程的本质。使用 cocos2d 的开发者关注的是游戏编程的核心问题，同时由于 cocos2d 良好的封装性，他们又不必真正地去处理最底层的实现。

1.1.8 超棒的 cocos2d 社区

在 cocos2d 社区，总有人能很快地回答你的问题，并且开发者可以自由地在这里分享知识和信息。

在这里，几乎每天都有新的教程和源码示例发布，而且大多数都是免费的。你会发现网上实在有太多太多的资源可供学习、带来灵感。

一旦游戏制作完成并在 App Store 上架，就可以把它推荐到 cocos2d 网站上。这么做至少可以帮助你获得其他开发者的关注，幸运的话还会得到一些非常有价值的反馈信息。

注意：

如果想要密切关注 cocos2d 社区的最新动态，建议你成为 Twitter 上 cocos2d 的粉丝(http://twitter.com/cocos2d)。

如果想成为我的粉丝，可以访问 http://twitter.com/gaminghorror。

在 Twitter 的搜索框内输入“cocos2d”，然后单击 Save this search 链接。这样你平时只需一次单击就可以经常地查看 cocos2d 的新帖子了。通常来说，这么做能让你看到一些非常有用但却容易忽略的 cocos2d 的相关信息。而且，你一定会在这里结识不少使用 cocos2d 的开发者。

1.2 注意事项

我想到两个对 cocos2d 开发者来说非常重要的问题，所以先在这里提一下。

1.2.1 Section 3.3.1

虽然“Section 3.3.1”这个名称听起来有点像是电影《星际旅行》里的某个秘密政府组织，但它实际上是苹果公司的开发者许可协议中的一个章节。随着 iPhone SDK 4 的发布，它已经成为某新条款的代名词。该条款或多或少地规定了开发者只可以使用 Objective-C、C、C++或 JavaScript 来进行开发。由于对 iOS 开发所使用的编程语言加以限制，该条款一经宣布便在 iOS 开发者中间引发了大量的讨论和担忧。

由于 cocos2d 完全是用 Objective-C 编写而成的，而且其使用的外部函数库，如 Chipmunk 和 Box2D 这两个物理引擎分别是用 C 和 C++编写而成的，因此只要开发者是直接使用 iPhone SDK 提供的 API，并且没有使用任何私有 API，cocos2d 的使用者就不需要为这个新条款感到担心。苹果公司官方不会因为 Section 3.3.1 而否定用 cocos2d 开发出的游戏和应用。

至于苹果公司官方为什么要设立条款来限制编程语言并且杜绝“中间层”，大家普遍认为这主要是为了防止用 Adobe Flash 开发出来的应用和游戏在 iOS 市场占据太大份额。

1.2.2 平台移植

也许你已经注意到了，很多开发平台上都有相应版本的 cocos2d 引擎，其中包括 Windows 和 Android。这些 cocos2d 引擎名称相同，而且基本的开发原理也是一样的，但它们是由不同的作者用不同的编程语言开发出来的，与 iPhone 版的 cocos2d 其实没有关系。例如，Android 版的 cocos2d 就是用 Java 编写的(Java 是 Android 设备的本地编程语言)。

如果你对平台移植感兴趣，想把游戏移植到其他平台上，就必须了解这一点：不同版本的 cocos2d 游戏引擎有着非常显著的差异。比如说，要将一个用 iPhone 版 cocos2d 开发出的游戏移植到 Android 平台就不是一件易事。首先，存在一些编程语言上的障碍，也就是说所有的 Objective-C 代码必须用 Java 重写一遍。重写完以后，还需要把所有调用过的 iPhone 版 cocos2d 的 API 改成相应的 Android 版本，并且保证你的代码中不含有任何目标平台不支持的特性。另外，每个平台都可能有自己不同的 bug、不同的技术限制和挑战。

总的来说，要将用 cocos2d 引擎编写的 iOS 游戏移植到其他含有 cocos2d 游戏引擎的平台上，所需要的工作量并不亚于把这个游戏在目标平台上用其他游戏引擎重写一遍。也就是说，并不存在一个平台转换器，可以让你轻轻一按就能万事大吉。不同平台上的 cocos2d 引擎唯一的相同之处仅在于名称和基本工作原理相同，仅此而已。

1.3 本书读者对象

我猜测你之所以挑选本书是因为它的书名吸引了你。也许你是想为 iPhone 或 iPad 开发 2D 游戏，并且选择了 iPhone 版 cocos2d 作为游戏引擎。或许你并不在乎选用什么游戏引擎，而仅仅是想为 iOS 设备开发 2D 游戏。又或许你使用 cocos2d 已有一段时间，现在想深入理解它。不管出于什么原因选择了本书，我保证你一定会有所收获！

1.4 阅读前提

几乎每本编程书籍都要求读者具备一些特定的知识，其中有些是必须知道的，还有些是有助于理解但不是必需的。本书也不例外。

1.4.1 编程经验

本书唯一的强制性要求就是你必须拥有一定程度的编程经验。你必须对一些编程概念，诸如循环、函数和类等有所理解。如果你以前编写过一些程序(要是使用过面向对象编程语言就更好了)，那么阅读本书应该没有什么问题。

还准备往下读吗？好样的！

1.4.2 Objective-C

读到这里就说明你一定有一些编程经验，不过你可能并没有用 Objective-C 写过程序

吧？我承认，知道它的人并不太多。

其实不懂 Objective-C 也能读这本书，但是如果对这个语言的基础知识有所理解的话，看起本书来就会轻松一些。假如对其他面向对象的编程语言，比如 C++、C#或 Java 有所理解，也许在读本书的过程中就能学会 Objective-C。不过老实说，尽管我在学 Objective-C 之前有 15 年左右的编程经验，用过 C++、C#以及各种各样的脚本语言，但是，要通过这本书弄清楚 Objective-C 对我来说还是很难。总会有一些细小却很麻烦的问题，让人一下子想不明白，于是在学习 cocos2d 的时候就很难集中注意力了。所以，最好手边常备一本 Objective-C 的参考书，不懂的时候可以随时翻阅。

我在学习 Objective-C 的时候，读的是由 Mark Dalrymple 和 Scott Knaster 写的、由 Apress 出版社出版的 *Learn Objective-C on the Mac* 一书。这本书真的是极其有用，如果你想学习 Objective-C 和 Xcode，那么我向你倾情推荐这本书。

另外，苹果公司的官方文档 *Introduction to the Objective-C Programming Language* 也是一份非常有价值的在线参考手册。你可以到以下网址进行阅读：http://developer.apple.com/mac/library/DOCUMENTATION/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html。

你可能觉得 Objective-C 有点吓人：代码中有很多方括号且内存管理方式复杂，另外 iPhone 上竟然没有垃圾回收机制！不必担心！

Objective-C 就像一件新潮的衣裳，仅仅是外表有些另类。它的一些基本编程概念，如循环、类、继承和函数调用，还是和其他编程语言一样的。不过，Objective-C 中的术语和其他语言是有差别的，比如说，Objective-C 开发者所说的“消息发送”实质上就是指“方法调用”。至于内存管理，cocos2d 已经为你尽量简化了这个过程，而且接下来我也将介绍一些比较基础、容易的规则，你以后可以遵循这些规则来进行开发。

1.5 本书内容

在本书中，我会分享一些我的游戏开发经验，从中大家可以看到交互游戏都是如何做出来的。我认为，学习编程绝不等同于去记住很多 API 方法。但是我在过去 20 年里读到了太多游戏开发书籍，它们都只是在遵循着“参考手册”的模式，而那根本就是 API 文档该做的事。从我 20 年前开始编程的时候，我就觉得我以后不可能只是看着一大堆编译器的说明书和参考手册去学习如何编程。那时候，这些书和手册都还是纸质版的，显然还没有在线版本(当时的互联网发展还不够成熟)。所以我桌上堆了差不多 15 英寸高的资料，着实令人望而生畏。

直到今天，有很多方法和 API 我还是记不住，而且以前背过的一些也常常忘记，所以我会一再去查阅它们。经过 20 年的编程生涯，我体会出了真正值得学习的东西——思想。好的编程思想和最佳实践经验是经得起时间考验的，而且对使用任何语言的编程都是有帮助的。而学习编程思想的最好方法就是去理解设计、架构和编码过程中采用的各种方法的基本原理。这就是我要着重关注的地方。

1.5.1 iOS 游戏开发新手将学会什么

别担心，我会让你尽可能容易地学会 cocos2d 里最重要的概念，强调一些你应该牢记的类、方法和概念，因为它们对于使用 cocos2d 进行编程实在是太基础、太常用了。

你也可以学到一些能够支持 cocos2d 或者能被它支持的工具。没有这些工具，你只能算是半个 cocos2d 程序员。你会用到像 Zwoptex 和 Particle Designer 这样的工具来制作越来越复杂、越来越具有挑战性的游戏。由于篇幅所限，本书的示例游戏还不够完整，可能还有些粗糙，而且我们也无法把每一行代码都解释清楚。但我会尽量在源代码中加入注释以便大家理解。

可以在这些游戏框架的基础上进行改进，进而完成你的游戏，我很期待看到你的成果！我认为给你一些可以继续开发的游戏框架会比用一整本书教你如何制作最典型的战机类太空游戏有用得多。

我主要是根据 App Store 上的关注度来选择示例游戏的。另外，因为有些开发者经常问一些在游戏开发中碰到的问题，所以我把这类具有代表性的游戏也加到了示例中。举例来说，画线类游戏(line-drawing game)是 cocos2d 游戏开发者们非常热衷的一种游戏，但是在这种游戏的开发过程中会遇到很多看似极其复杂的问题。

我看过很多其他程序员写的 cocos2d 代码，也参与过他们对于代码设计、结构和风格的讨论。在本书中，我采用了“继承+组合”的模式来作为示例代码的框架，在后面的章节中我会解释这样做的好处。另一个经常被问到的代码设计问题是：不同对象间应该如何沟通。不同的代码设计和代码结构都有各自的优缺点，我将介绍这些概念以帮助大家在今后写出一些更稳定、性能更高的代码。

1.5.2 iPhone 应用程序开发者将学会什么

作为 iPhone 应用程序开发者，你一定接触过 iPhone SDK 了？非常好！你最感兴趣的问题一定是如何在没有 Interface Builder 的情况下制作游戏吧？其实，除了 Interface Builder 以外，还有很多其他工具可供使用。它们可能不像苹果公司官方提供的工具那么炫，但也是很有用的。

另外，编程思路也需要进行些改变。在游戏编程中，你通常不会接收或发送事件，而是让更多的对象决定如何响应事件。考虑到性能问题，也为了减少用户输入的等待时间，游戏引擎的各个系统往往是密切相关的。在这里，大量的工作会在循环和更新方法(会在每帧被调用一次，或以用户指定的频率被调用)中完成。相比较而言，一个基于用户交互的应用会花大部分时间来等待用户输入，而游戏则会在后台不断地对数据和像素进行处理，即便玩家没有进行任何操作。所以，在游戏中，每时每刻都有很多事情在发生。而且，出于对性能的考虑，游戏代码也变得更加优化、更加高效。

1.5.3 cocos2d 开发者将学会什么

你已经比较熟悉 cocos2d 了吧？你可能会怀疑是否能从本书中学习到新的知识。我想说：会的！你可以直接跳过前面的章节，不过你一定会对本书提供的游戏源代码感兴趣。

通过这些源代码，你可以看到我是如何对这些游戏进行架构，以及这样做的原因。也许你能从不同的游戏实现中得到灵感。而且随书源码中还有很多小提示，它们对你将来的编程生涯一定会有所帮助。最重要的是，这本书不是哪个名不见经传的 IT 怪人写的——也许他写完一本书就销声匿迹，更不会留下 email 地址或个人网站供你反馈意见。事实是，这个名不见经传的作者以后是可以随时联系上的。我的个人博客(http://www.learn-cocos2d.com)在 cocos2d 社区中算是比较活跃的，我会经常在博客上对本书进行更新和补充。

1.6 章节介绍

下面对全书各章作简要介绍：

第 2 章：入门 本章将教你搭建 cocos2d 开发环境、安装项目模板并制作首个 Hello World 项目。你将学习到 cocos2d 的基础知识，如场景和节点。

第 3 章：基础知识 本章将向你介绍 cocos2d 里最常用的类(比如 Sprite、Transitions 和 Action)以及它们的用法。

第 4 章：你的第一个游戏 本章将教你使用加速计来写出你的第一个游戏：敌人从天而降，玩家通过倾斜设备来躲过敌人的撞击。

第 5 章：游戏组件 本章将教你制作一个稍大一些的游戏。这个游戏要求有更好的代码结构。你将学习到场景和各节点间的层次关系以及游戏中对象间交换信息的不同方法。

第 6 章：深入了解精灵 通过本章的学习，你会知道什么是“纹理图册”(Texture Atlas)，我们为什么使用它，以及如何使用 Zwoptex 工具创建纹理图册。

第 7 章：滚屏射击游戏(上) 把纹理图册准备好以后，你将学习如何实现一款通过触屏来控制的滚屏射击游戏。

第 8 章：滚屏射击游戏(下) 没有敌人的话，我们的射击游戏就没有靶子了，对吧？所以，在本章我会教你如何实现移动、射击、生成敌方部落等效果来提高游戏的可玩性。

第 9 章：粒子效果 本章将教你使用 Particle Designer 工具来为横向滚屏游戏增添粒子效果。

第 10 章：瓦片地图 本章将教你用从第 9 章中学习到的知识来制作另一款非常流行的 iOS 游戏。

第 11 章：斜角瓦片地图 由于 cocos2d 支持 TMX 文件格式，因此本章将教你如何用 Tiled 编辑器来制作基于瓦片地图的游戏。

第 12 章：物理引擎 通过本章的学习，你将学会如何实现用手指决定物体移动方向的效果。

第 13 章：弹球游戏 这是学习 Chipmunk 和 Box2D 引擎的入门章节，也是你实现一些疯狂梦想的基础。

第 14 章：Game Center 在本章中，你会使用一些物理效果制作一个太空游戏，它集反重力、弹球与射击的元素于一身。虽然游戏本身比较不切实际，但是开发起来会非常有趣，而且我们会用到真正的物理效果。

第 15 章：番外篇 本章是关于周边技术及市场的介绍，而不是总结前文。本书到此

结束。但是别担心，你的游戏开发之旅不会结束。希望本章能够为你未来的开发之路提供一些启发。

1.7 问题和反馈

我一直希望大家既能容易地学会 cocos2d 和 iOS 游戏开发，又能对一些高级的游戏编程理念有所思考。不知我的目的是否已经达成。

如果本书有任何地方令你感到困惑，可以随时通过我的 email(steffen@learn-cocos2d.com)向我提问。我也会继续在我的网站(http://www.learn-cocos2d.com)上对一些可能在书中没有解释清楚的问题作补充和说明。欢迎大家随时提出反馈意见！