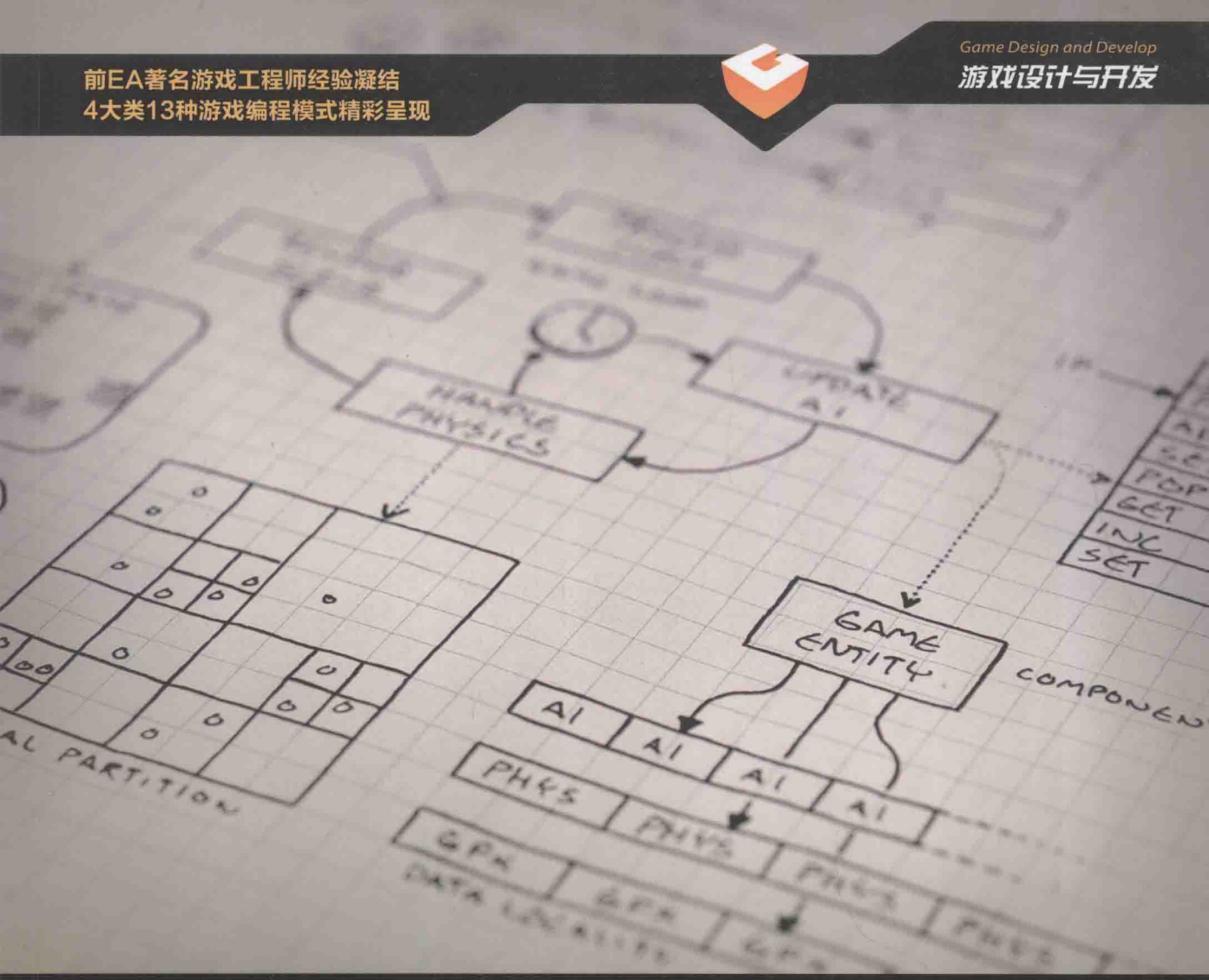
前EA著名游戏工程师经验凝结
4大类13种游戏编程模式精彩呈现

Game Design and Develop
游戏设计与开发



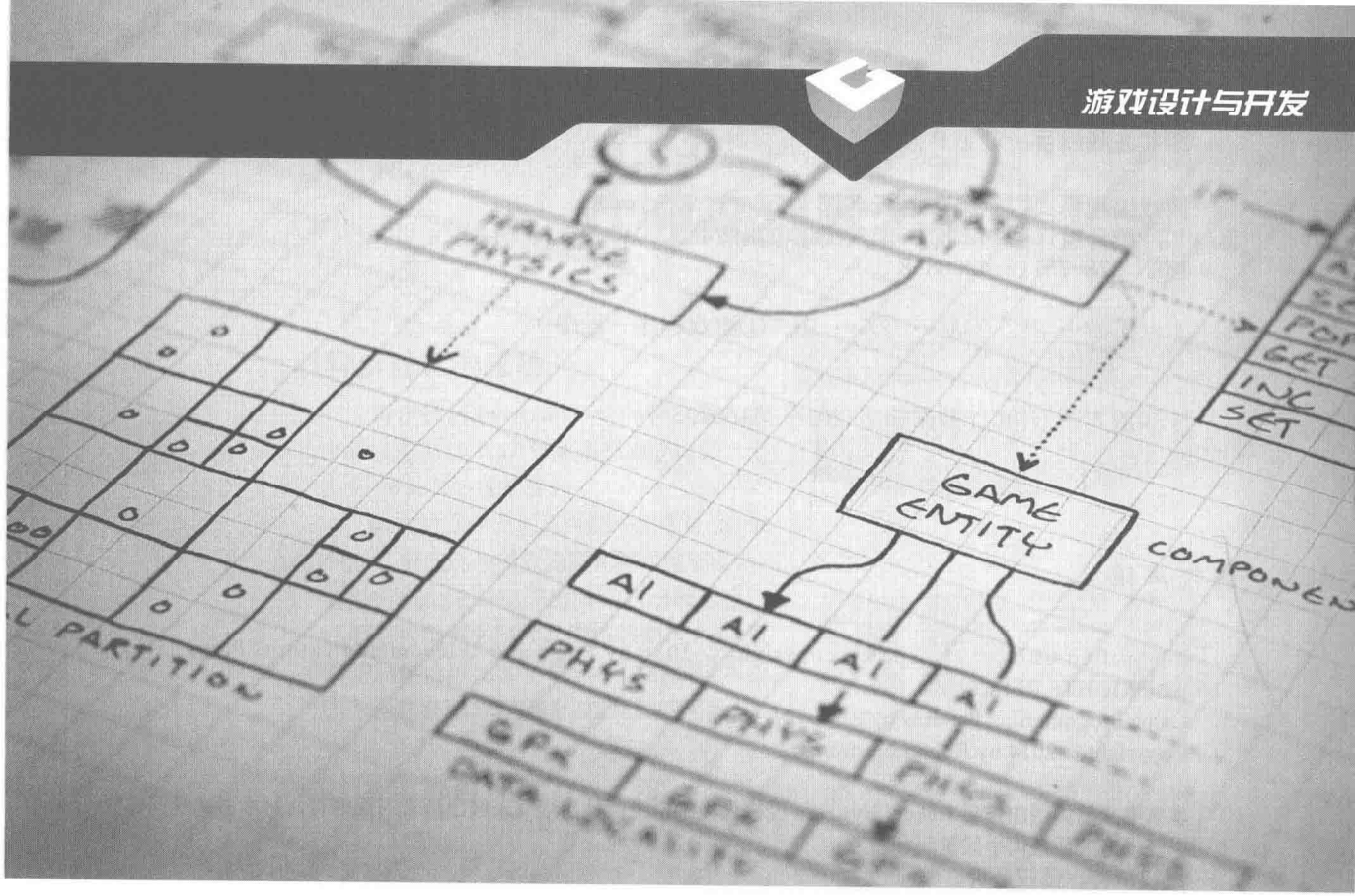
Game Programming Patterns

游戏编程模式

[美] Robert Nystrom 著
GPP翻译组 译

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

游戏设计与开发



游戏编程模式

[美] Robert Nystrom 著
GPP翻译组 译

人民邮电出版社
北京

图书在版编目（CIP）数据

游戏编程模式 / （美）尼斯卓姆（Robert Nystrom）著 ; GPP翻译组译. -- 北京 : 人民邮电出版社, 2016.9
ISBN 978-7-115-42688-8

Ⅰ. ①游… Ⅱ. ①尼… ②G… Ⅲ. ①游戏程序－程序设计 Ⅳ. ①TP311.5

中国版本图书馆CIP数据核字(2016)第160703号

版权声明

◆ 著 [美] Robert Nystrom
译 GPP 翻译组
责任编辑 陈冀康
责任印制 焦志炜

◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路 11 号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
三河市海波印务有限公司印刷

◆ 开本：800×1000 1/16
印张：20.75
字数：437 千字 2016 年 9 月第 1 版
印数：1 – 3 000 册 2016 年 9 月河北第 1 次印刷

著作权合同登记号 图字：01-2015-1268 号

定价：69.00 元

读者服务热线：(010)81055410 印装质量热线：(010)81055316
反盗版热线：(010)81055315

内容提要

游戏开发一直是热门的领域，掌握良好的游戏编程模式将是开发人员的必备技能。本书细致地讲解了游戏开发需要用到的各种编程模式，并提供了丰富的示例。

全书共 6 篇 20 章。第 1 篇概述了架构、性能和游戏的关系，第 2 篇回顾了 GoF 经典的 6 种模式。第 3 篇到第 6 篇，按照序列型模式、行为型模式、解耦型模式和优化型模式的分类，详细讲解了游戏编程中常用的 13 种有效的模式。

本书提供了丰富的代码示例，通过理论和代码示例相结合的方式帮助读者更好地学习。无论是游戏领域的设计人员、开发人员，还是想要进入游戏开发领域的学生和普通程序员，都可以阅读本书。

作者简介

Robert Nystrom 是一位具备超过 20 年职业编程经验的开发者，而其中大概一半时间用于从事游戏开发。在艺电（Electronic Arts）的 8 年时间里，他曾参与劲爆美式足球（Madden）系列这样庞大的项目，也曾投身于亨利·海茨沃斯大冒险（Henry Hatsworth in the Puzzling Adventure）这样稍小规模的游戏开发之中。他所开发的游戏遍及 PC、GameCube、PS2、XBox、X360 以及 DS 平台。但最傲人之处在于，他为开发者们提供了开发工具和共享库。他热衷于寻求易用的、漂亮的代码来延伸和增强开发者们的创造力。

Robert 与他的妻子和两个女儿定居于西雅图，在那里你很有可能会见到他正在为朋友们下厨，或者在为他们上啤酒。

译者简介

GPP 翻译组是一群游戏开发技术爱好者为了翻译本书简体中文版而成立的一个兴趣小组。GPP 小组的成员如下：

赵卫兵（ChildhoodAndy）

游戏开发爱好者，曾从事游戏开发，《Chipmunk2D Physics》官方文档译者，泰然网成员之一。崇尚开源、分享精神，目前就职于 58 同城。

屈光辉（子龙山人）

Cocos2d-x 核心开发者，Cocos Creator 核心开发者，《Cocos2D 权威指南》第二作者，泰然网早期创始成员之一，Cocos2d 社区知名博主，emc-china，org 创始人。专注于移动游戏开发和游戏 UI 框架开发及优化。

郑炯彬

“90 后”，香港科技大学研究生在读，视觉算法程序员，户外爱好者，自认为最离不开三样东西：书、音乐、NULL。

陈侃

游戏开发者、游戏爱好者。同样热爱文字和美术，致力于富有创造力和艺术性的工作。

姜召阳

从事移动游戏开发行业，一枚文艺帝都程序员，平时喜欢参与开源项目、读书和切磋篮球。

特别感谢其他的译者：许新星、唐宏洋、张植臻和洪孝强。

前言

在五年级的时候，我和我的小伙伴们获准使用一个放置着几台非常破旧的 TRS-80s[1] 的闲置教室。为了激励我们，一位老师找到了一份印有一些简单 BASIC 程序的打印文档给我们。

当时，计算机上的音频磁带驱动器是坏掉的，所以每次我们想要运行一些代码的时候，都不得不仔细地从头开始键入代码。这使得我们更喜欢那些只有几行代码的程序：

```
10 PRINT "BOBBY IS RADICAL!!!"
20 GOTO 10
```

如果计算机打印足够多的次数，或许它会神奇的变成现实哦[2]。

即便如此，整个过程还是充满了艰辛。我们不懂得如何编程，所以一个小的语法错误便让我们感到很费解。程序出毛病是家常便饭，而此时我们只能重头再来。

在这叠文档的最后部分，是一个真正的“怪物”：一个代码量占据几页篇幅的程序。我们思量良久，这才鼓起勇气去尝试它，不过它极为诱人——标题写着“巨魔洞穴”。我们不知道它是做什么的，不过听起来像是个游戏，还有什么能比亲手写一款计算机游戏更酷呢？

我们从没让这个程序真正运行起来过。一年后，我们搬出了那个教室（后来当我了解了一点 BASIC 时，才知道那只是一个供桌面游戏使用的角色生成器，而并非一款完整的游戏）。命中注定，从那之后，我立志要成为一个游戏程序员。

在我十几岁时，我的家人搞了一台装有 QuickBASIC 的 Macintosh，之后又装了 THINKC。我几乎整个暑假都在那上面倒腾游戏。自学是缓慢而痛苦的。我能轻松地让一些代码运行起来（也许是一张屏幕地图或者一

---

[1] 见[维基百科 TRS-80s](http://en.wikipedia.org/wiki/TRS-80)。译者注：TRS-80s 于 1977 年诞生，是第一批问世的微型计算机之一。

[2] 这里指的是计算机反复打印第 10 行代码的语句“BOBBY IS RADICAL!!!”，作者开玩笑地说会变成现实。

个小型猜谜游戏），但随着程序增大，编码变得越来越难。

起初，我的挑战在于让程序运行起来。后来，我开始琢磨如何编写超出我大脑思考范围的更大些的程序。我开始试图寻找一些关于如何组织程序的书籍，而不只是读一些关于“如何用C++编程”之类的书籍。

我的许多夏天都是在路易斯安那州南部的沼泽中捕蛇和乌龟来度过的。如果户外不是那么酷热的话，这将很可能是一本爬虫学的书，而不是讲游戏编程的书。

几年很快过去，一位朋友给了我一本书：《设计模式：可复用面向对象软件的基础》（Design Patterns: Elements of Reusable Object-Oriented Software）。终于来了！这就是我从青少年开始便一直寻找的那本书！我一口气将它一字不漏地读完了。虽然我仍纠结于自己的程序，但是看到别人也如此挣扎并提出了解决方案，也如释重负。原本赤手空拳的我终于有工具可使了。

在2001年，我得到了自己梦寐以求的工作：EA（Electronic Arts）的软件工程师。我迫不及待地想看一下真正的游戏，以及工程师们是如何组织它们的。像Madden Football这样的大型游戏到底是个什么样的架构？不同系统之间是怎么交互的？他们是怎么让一套代码库在不同平台上运行的？

这是我和这位朋友第一次见面，在5分钟自我介绍之后，我坐在他的沙发上，在接下来的几个小时里，我聚精会神地阅读而完全忽视了他。我感觉从那以后自己的社交能力还是至少有那么一丁点儿提升的。

分解阅读源码是一种震撼人心且令人惊奇的体验。图形、人工智能、动画和视觉效果方面，都有十分出众的代码。我们公司有人懂得如何榨取CPU的每一个周期并加以善用。一些我甚至不知道能否实现的东西，这些家伙一个早上就能搞定。

但是这些优秀代码所依托的架构往往是事后想出来的。他们太专注于功能以至于忽视了组织架构。模块之间的耦合现象很普遍，新功能往代码库里见缝插针，而不顾其是否契合。这些所见令我幻想破灭，看起来许多程序员，就算他们心血来潮地翻开过《设计模式》一书，恐怕能看完单例就很不错了。

当然，也不是真的那么糟糕。我曾设想游戏程序员们坐在放满白板的象牙塔中，连续几周冷静地讨论代码架构的细节。实际情况是，我眼前这份代码是别人在紧张的期限里赶工出来的。他们尽了自己最大的努力，同时，我逐渐认识到，他们竭尽全力的结果通常是编写出了十分优秀的代码。我写游戏代码的时间越长，就越能发现隐藏在这些代码之下的可贵之处。

遗憾的是，“隐藏”一词往往说明了问题。宝藏埋在代码深处，而许多人正在它们之上路过（优秀的代码被许多人视而不见）。我看到过同事努力想改造出一个好的解决方案，那时，他们所需要的示例代码就隐藏在他们脚下的代码库之中。

这个问题正是本书力图解决的。我挖掘并打磨出自己在游戏代码中所发现的最好的设计模式，在此一一呈现给大家，以便我们将时间节省下来创造新事物，而不是重新造轮子。

市面上已有的书籍

目前市面已经有数十多本游戏编程的书籍。为什么还要再写一本？

我见过的大多数游戏编程书籍无非两类。

- **关于特定领域的书籍**。这些针对性较强的书籍带领你深入地探索游戏开发的一些特定方面。它们会教你 3D 图形、实时渲染、物理仿真、人工智能或音频处理。这些是众多游戏程序员在自己的职业生涯中所专注的领域。
- **关于整个游戏引擎的书籍**。相反，这些图书试图涵盖整个游戏引擎的各个部分。它们的目标是构建一整套适合某个特殊游戏类型的引擎系统，这类通常是 3D 第一人称射击游戏。

我喜欢这两类书，但我觉得它们仍留下了一些空白。讲特定领域的书很少会谈及你的代码块如何与游戏的其他部分交互。你可能擅长物理和渲染，但是你知道如何优雅地将它们拼合起来吗？

第二类书籍涵盖了这类问题，但我往往发现这类书通常都太过庞大、太过空泛。特别是随着移动和休闲游戏的兴起，我们正处在众多类型的游戏共同发展的时代。我们不再只是照搬 Quake[1] 了。当你的游戏不适合这个模型时，这类阐述单一引擎的书籍就不再合适了。

这种分类讲解风格的另外一个例子，就是广受大家喜爱的《游戏编程精粹》系列。

相反，这里我想要做的，更倾向于分门别类。本书的每个章节都是一个独立的思路，你可以将它应用到你的代码里。你也可以针对自己制作的游戏来决定以最恰当的方式将它们进行混搭。

本书和设计模式有什么联系

任何名字中带有“模式”的编程书籍都和经典图书《设计模式：可复用面向对象软件的基础》有所联系。这本书由 Erich Gamma、Richard Helm、Ralph Johnson 和 John Vlissides 编著（这 4 人也称为“Gang of Four”，即本书所提到的“GoF”四人组）。

[1] 《雷神之锤》，第一个真 3D 实时演算的 FPS 游戏。

本书的英文原名是 Game Programming Design Patterns，并不是说 GoF 的书不适用于游戏。恰恰相反，在本书第 2 篇中介绍了众多来自 GoF 著作的设计模式，同时强调了在它们游戏开发中的运用。

从另一面说，我觉得这本书也适用于非游戏软件。我也可以把这本书命名为《More Design Patterns》，但我认为游戏开发有更多迷人的例子。难道你真的想要阅读的另外一本关于员工记录和银行账户例子的设计模式图书吗？

也就是说，尽管这里介绍的模式在其他软件中也是有用的，但我觉得它们特别适合应对游戏工程中普遍会遇到的挑战，例如：

- 时间和顺序往往是一个游戏的架构的核心部分。事情必须依照正确的顺序和正确的时间发生。
- 开发周期被高度压缩。众多程序员必须在不牵涉他人代码、不污染代码库的前提下对一套庞大而错杂的行为体系进行快速的构建与迭代。
- 所有这些行为被定义后，游戏便开始互动。怪物撕咬英雄，药水混合在一起，炸弹炸到敌人和朋友……诸如此类。这些交互必须很好地进行下去，可不能把代码库给搅成一团毛线球。
- 最后，性能在游戏中至关重要。游戏开发者永远在榨取平台性能这件事上赛跑。多削掉一个 CPU 周期，你的游戏就有可能从掉帧和差评迈入 A 级游戏和百万销量的天堂。

设计模式一书本身也源自前人的灵感。创造一种模式语言来描述问题的开放性解决方案，该想法来自《A Pattern Language》，由 Christopher Alexander（和 Sarah Ishikawa、Murray Silverstein 一起）完成。

这是一本关于框架结构的书（就像真正的建筑结构中建筑与墙体和材料之间的关系），作者希望他人能够将其运用作其他领域问题的解决方案。设计模式（Design Patterns）正是 GoF 在软件领域的一个尝试。

如何阅读本书

本书大致分为三大部分。第一篇是介绍和框架。这包括前言和第 1 章。

第二篇，再探设计模式，回顾了 GoF 中的一些设计模式。在这个部分的每一章中，我都会试图给出自己对该模式的认识，以及对模式与游戏开发之间关联的看法。

最后部分是这本书的重头戏。这部分呈现了我认为十分有用的 13 种设计模式。它们分为 4 篇：序列型模式、行为型模式、解耦型模式和优化型模式。

这些模式使用一致的文本组织结构来讲述，以便你将该书作为参考并能快速找到你所需要的内容。

- **目的**部分简单介绍了该模式以及其力图解决的问题。以此作为开篇，以便你能够快速翻阅本书并根据自己眼下的问题对号入座。
- **动机**部分描述了一个可引用该模式的示例问题。不同于具体的算法，模式只有运用到具体问题中时方能见其真章。教模式而不举具体例子，

就像教烤面包而不提面团一样。这个部分提供“面团”，之后的部分将会教你如何“烘培”。

- **模式**部分会提炼出前面示例中的模式本质。如果你想了解该模式枯燥的书面描述，就是这部分了。如果你已经熟悉了该模式，这部分也是一个很好的复习，确保你没有忘记该模式的要素。
- 到目前为止，该模式只是就一个单一的例子来解释的。但你怎么知道该模式是否适用于其他问题呢？**使用情境**对模式何时使用以及何时不该使用提供了一些指导。**使用须知**部分会指出使用该模式时带来的后果和风险。
- 如果你也像我一样，需要借助具体的实例才能真正的理解，那么**示例**部分正满足你的需要。它一步一步地展示这个模式的完整实现，以便你可以看到模式究竟是如何工作的。
- 模式和单一的算法不同，因为模式是开放式的。每次使用模式的时候，你实现的方式有可能会有不同。接下来**设计决策**部分，会探讨这个问题，并告诉你在应用模式时可供考虑的不同选项。
- 每章以一个短小的参考部分作为结束，它会告诉你该模式和其他模式的关联并指出使用该模式的一些真实的开源代码。

关于示例代码

这本书中的示例代码用 C++编写，但是这并不意味着这些模式仅能在 C++下发挥作用或者说 C++比其他语言要好。几乎所有的语言都适用，虽然有些模式确实倾向于有对象和类的语言。

我选择 C++有几个原因。首先，它是现行商业游戏中最流行的语言，是该行业的通用语言。另外，作为 C++基石的 C 语言的语法也是 Java、C#、JavaScript 和许多其他语言的基础。即使你不懂 C++，也没有关系，这里的示例代码基本上是你无需花太多力气就足以能够理解的。

这本书的目的不是教你学习 C++。示例会尽可能保持简单，但它可能并不符合优良的 C++编码风格或用法。阅读代码时要理解代码所传达的思想，而不是代码本身的表达。

特别一提的是，示例代码没有采用“现代”C++（C++11）或更高版本风格。它没使用标准库并很少使用模板。这是“糟糕”的 C++代码，但我仍希望保留这一特色，这样会对那些从 C、Objective-C、Java 和其他语言转来的读者更加的友好。

为了避免浪费篇幅，你已经看过的或者和模式不相关的代码，有时会

在例子中省略，通常用省略号来表示省去的代码。

例如有一个函数，它完成某项工作并返回一个值。同时讲解的模式只关心返回值，不关心其具体的工作内容。在这种情况下，示例代码看起来会像这样：

```
bool update()
{
  // Do work...
  return isDone();
}
```

何去何从

设计模式是软件开发中一个不断变化和扩展的部分。这本书延续了GoF的文献所开启的过程，并分享他们眼中的那些软件设计模式，而这一进程也不会因本书的完成而就此终止。

你是这个过程的核心之一。只要你开发了你自己的模式或提炼（或者反驳！）这本书中提到的模式，你就是在为软件社区贡献力量。如果你对书中的内容有任何建议、修正或者其他反馈，请与我联系。

致谢

我估计只有写过书的人才知道写书的过程中会遇到多少麻烦，但是还有另外一些人也知道写书的负担究竟有多重——那就是那些不幸和作者关系亲密的人。我是在妻子 Megan 煞费苦心地为我节省的空余时间里写完这本书的。洗盘子和为孩子洗澡或许不能叫做“写作”，但是没有她的这些付出，这本书也不会出版。

当我还是 EA（Electronic Arts）的一名程序员时，便开始写这本书了。我认为公司的同事们并不完全了解这本书的技术细节，但是我对 Michael Malone、Olivier Nallet 和 Richard Wifall 的支持表示感谢，感谢他们为前几章提供了详细、深刻的反馈。

写到大约一半的时候，我决定不做一名传统的出版者。我知道这意味着会失去编辑的指导，但是我收到了许多读者发送的电子邮件，他们告诉我希望这本书怎么写。我没有校对者，但是我收到了超过 250 份的 bug 报告，来帮助我改进写作。我也曾缺乏按计划写作的动力，但当我完成每一章并收到来自读者的鼓励的时候，我又有了充足的精神动力。

他们称这为“自出版”，但是“众包出版”更加贴切。写作是一份孤独的工作，但是我从未孤单过。即使整个写作过程持续了两年时间，但我总能不断得到鼓励。如果没有一堆人不断提醒我他们在期待着更多的章节，我绝不会想要继续写作并完成这本书。

对每一位发邮件的或者评论过的，点赞的或者收藏的，发微博的或者转发了的，任何帮助过我的，或者将本书告诉朋友的，或者给我提交一份 bug 报告的朋友们：我内心充满了对你们的感激。完成这本书是我人生中最大的目标之一，是你们帮我实现了它。

感谢你们！

我并不是没有文字编辑。Lauren Briese 在我需要的时候帮助了我，并出色地完成了工作。

特别感谢 Colm Sloan，他仔细地把每个章节阅读了两遍，并给了我大量出色的反馈。这都出自他内心的善意。我欠他一份人情。

多数游戏程序员所面临的最大挑战就是完成他们的游戏。许多游戏止步于其高度复杂的代码库面前，而最终没能问世。游戏编程设计模式正是为解决此问题而生。带着多年上市 3A 级大作的经验，本书收集了许多已经实证的设计模式来帮助解构、重构以及优化你的游戏，书中将各大模式以菜单的形式分立以便开发者们各取所需。

你将学会如何编写一个健壮的游戏循环，如何应用组件来组织实体，并利用 CPU 缓存来提升游戏性能。本书将带你深入了解脚本引擎如何对行为进行编码，以及四叉树和其他空间划分等优化引擎的手段，并为你展示其他经典的设计模式是如何应用于游戏之中的。

目录

第 1 篇　概述

第 2 篇　再探设计模式

第 3 篇 序列型模式