CENGAGE Learning

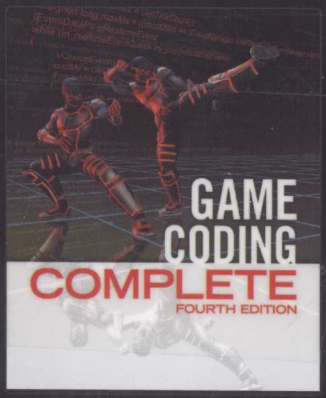
游戏编程经典入门读物最新版

Game Design and Develop

游戏设计与开发

游戏编程权威指南

（第4版）



GAME CODING COMPLETE

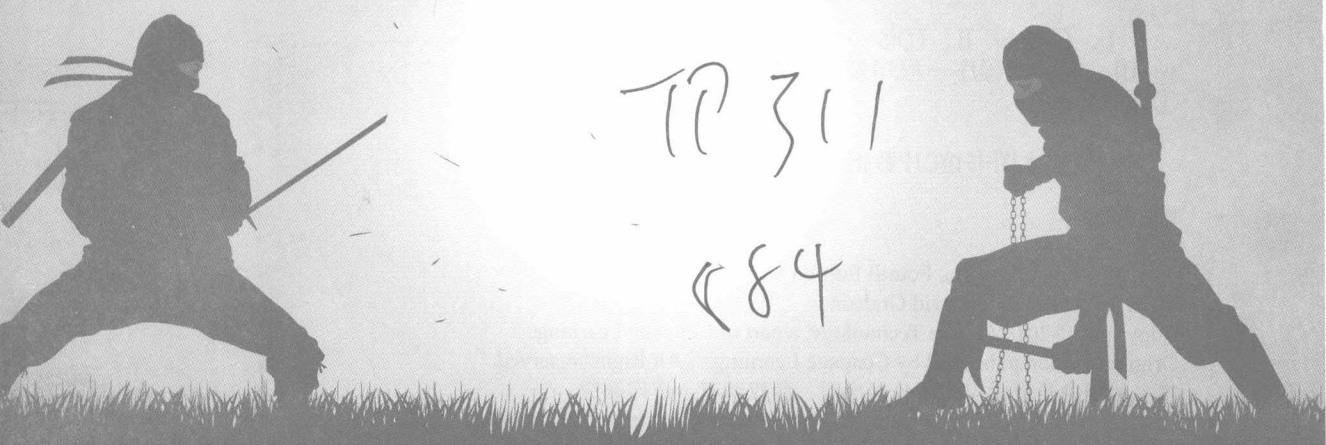
[美] Mike McShaffry David "Rez" Graham 著

师蓉 李静 李青翠 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



游戏编程权威指南

（第4版）

[美] Mike McShaffry　David "Rez" Graham　著
师蓉　李静　李青翠　译

人民邮电出版社
北京

图书在版编目（CIP）数据

游戏编程权威指南 : 第4版 / (美) 沙福瑞 (McShaffry,M.), (美) 格雷海姆 (Graham,D.R.) 著 ; 师蓉, 李静, 李青翠译. -- 北京 : 人民邮电出版社, 2016.3
ISBN 978-7-115-41034-4

Ⅰ. ①游… Ⅱ. ①沙… ②格… ③师… ④李… ⑤李… Ⅲ. ①游戏程序－程序设计－指南 Ⅳ. ①TP311.5-62

中国版本图书馆CIP数据核字(2016)第018295号

◆ 著 ［美］Mike McShaffry David "Rez" Graham
译 师 蓉 李 静 李青翠
责任编辑 陈冀康
责任印制 张佳莹 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
北京天宇星印刷厂印刷
◆ 开本：800×1000 1/16
印张：43.5
字数：1 068 千字 2016 年 3 月第 1 版
印数：1-3 000 册 2016 年 3 月北京第 1 次印刷
著作权合同登记号 图字：01-2012-9283 号

定价：99.00 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

本书是一本优秀的游戏开发权威指南，是开发、部署、运行商业游戏的必备读物。

全书分为 4 个部分共 24 章。第一部分是游戏编程基础，主要介绍了游戏编程的定义、游戏架构等基础知识。第二部分是让游戏跑起来，主要介绍了初始化和关闭代码、主循环、游戏主题和用户界面等。第三部分是核心游戏技术，主要介绍了一些更为复杂的代码示例，如 3D 编程、游戏音频、物理和 AI 编程等。第四部分是高级知识和综合应用，主要介绍了网络编程、多道程序设计和用 C#创建工具等，并利用前面所讲的知识开发出一款简单的游戏。

本书适合游戏开发人员、游戏架构设计人员和游戏引擎用户参考阅读，也适合想要进入游戏开发领域的读者阅读。

序 1

请允许我先承认一些事情。首先，我从来没有给任何书写过序。我写过书，但从来没有写过序。说实话，我读书时通常都会略过这些内容，所以可能也不会有人阅读我写的这些内容。这让我可以略感安心地承认第二点：我不是程序员。从来不是，以后也不会是，虽然我曾经努力过。尽管我对编程一窍不通，但我仍然活得好好的。我不是在祈求你的同情，我只是想告诉你，就算你不思考“该死的，如果我能从 BSP 树中知道 Z 缓冲就好了”，你也不会虚度时光。如果你已经是一名程序员，那么你相对我而言进入电子游戏行业的优势更大。（如果你不是一名程序员，千万不要学习我的做法，而要听取我的建议——马上开始学习编程，越快越好！Mike 在本书中给出了一些建议，请留意。）

承认了这两点后，我想我已经没有任何威信了。你们真是一群幸运的家伙，因为这本书的作者非常可靠。Mike McShaffry（游戏行业众人皆知的“Mike 先生”）才是主角。Mike 是一个天才的生存者。他是很有发言权的人，因为上苍作证，他那丰富的实践经历足以让他有资格说点什么。

Mike 丰富的游戏开发经验足以覆盖整个领域。当团队只有十几个人的时候他就已经在那里了，然后他经历了 20、30、50 人的阶段。他经历过创业，也为行业内最大的发行商工作过，他开发过“传统”和“非传统”的游戏——从“创世纪”到“21 点”，单机、多玩家、在线、离线等你能够想到的所有东西。对于 PC 游戏来说，他能以各种权威身份发言——程序员、设计师、项目经理、开发主管、工作室负责人……

而我有幸看着他随着每个新角色和每个新项目学习成长。从他获得第一份游戏编程工作时我就在那里了。那是在 1990 年，他申请在 Origin 公司的编程工作，而我就是其中的面试人之一（好像是很久以前了，不是吗，Mike?）。

在“Martian Dreams”游戏中，他是从一名程序员做起，但当项目结束时，他已经是推动游戏向终点前进的引擎了。没了 Mike 就不会有这款游戏。他的努力、奉献、对游戏的热爱、绝妙的设计、天生的领导才能、协调左右大脑的能力（我并不是说他愿意长时间工作），促使我们更加努力地工作，保证游戏总有一些独特之处（至少我们这些为之努力过的人这么认为——虽然这款游戏卖得并不是特别好！）。

我承认，我甚至不记得我是否在“Martian Dreams”游戏中给过 Mike“主程序员”的头衔，但他肯定当之无愧。这个家伙就是一台机器，他工作的时间要比我大多数同事都长（我是说游戏行业的同事）。在同样的时间内，他能更高效地完成更多工作。这对我们这些平凡人来说很不公平。当 Mike 来了状态时，他是无人能敌的。而他几乎总是在状态中——“Martian Dreams”项目后，Mike 在“创世纪Ⅶ、Ⅷ、Ⅸ”和很多其他游戏中也是这么做的。真是太可怕了！

回想起来，当时的我们更年轻、更幼稚。那些长时间的工作对 Mike 和我们其他人来说都是一种暗示，暗示着我们没有一点软件开发或者游戏设计的概念。因此，我们不得不长时间地工作，我们如此努力只是为了不让努力付之东流。

阅读这本书时，我不仅惊奇于 Mike 这些年学到的知识。我总是会想，如果我们当时能得到这本书中的信息，我们就能做很多事情，我们的游戏就能更优秀。但当时没有任何人对游戏、编程实践和软件开发有足够的了解。我们是在摸索中总结出了这些。

今天，有很多书教给你如何编写代码。还有一些书会进一步告诉你，让游戏和文字处理程序或者本地医疗提供者（我们习惯称他们“大夫”）计费系统不同的是什么。但即使是今天，也没有几本书能等效地提供核心编程建议和核心开发过程、进度安排、调试、团队建设等信息。

开发过程？团队建设？谁在乎这些？你只是想编写代码，对吗？如果你和我认识的大多数程序员一样，那么这就是你心里的想法。但是，你错了。过去程序员可以关上门编写代码，他们不必关心他们的工作如何融合到游戏开发的大环境中。这种做法在 10 年前可能行得通（只是可能），但现在已经行不通了。当团队随着时间一天天发展壮大，随着时间线的延长和预算的增加，过程和团队问题已经成为每个人都需要关心的问题。

Mike 明白这些，他在第 1 章就很清楚地进行了说明。他说：“作为优秀的开发者，你应该对这个行业的真实需求有深刻的认识”。简单地说，这就是这本书的特别之处。多数人认为只要有热情和才智就足以让他们进入游戏行业，足以确保他们从第一份工作起就一帆风顺。他们会说：“我一直在玩游戏，而且我是最棒的程序员。还有什么要问的？雇用我吧！”

我现在就告诉你，还有很多事情需要了解，这可能就是本书中最重要的一课。游戏非常复杂，它们的创建需要艺术和科学的独特结合（有些人将它称为“魔法”，这不是完全没有道理的）。只有亲身经历后，你才能理解游戏开发对于技能的苛求。至少我曾经有过错误的认识，但这正是 Mike 大展身手的时候。他已经亲身经历过这些，他可以将你从麻烦、痛苦、创伤、关系破裂和公司失败中拯救出来。在游戏中发生这些情况再平常不过了。不管你怎么想，它并不只是荣耀、名誉、财富和强烈的个人满足。

Mike 在这本书中介绍了很多方面的内容。我很喜欢“Mike's Tales from the Pixel Mines”中的内容。

当然，对于已经是程序员的人来说，书中详细讲述了游戏编程的特别之处（相信我，它真的非常特别）。本书还介绍了很多其他编程书籍没有给予足够重视的内容。

不管你是不是程序员，这些都能让你成为最有效率的开发人员。这可不是你在其他地方能学到的。你必须经历整个过程（和其中的痛苦！）。或者你必须找到一位导师，花费几年的时间学到他全部的知识。再或者，你可以停止阅读这篇序，开始阅读这本书吧！

你还在等什么？

——Warren Spector，Junction Point 工作室的创始人

序2

在他的职业生涯中，Rez 完成了很多很酷的工作，遇到了很多优秀的人。因此，当他邀请我为本书写序时我感到非常骄傲。

我想他选择我是因为我是他 Origin 故事的见证者。Origin 故事非常有趣（不管蜘蛛侠打败了多少坏蛋，粉丝们仍然希望听到他被放射性蜘蛛咬伤的故事），它们也非常具有教育意义。如果你正在阅读这篇序言，你可能也会思考如何制作自己的 Origin 故事，让自己在电子游戏编程行业中获得成功。你从 Rez 的故事中可以学到很多。

我第一次见到 Rez 是在 2005 年，当时他到我开办的一家小公司“Super-Ego Games”面试。我们当时相信人们需要可以在游戏机上玩的互动情景喜剧，而且我们正准备提供这类游戏。（我们如何做到的，以及后来发生了什么本身就是一个有趣的故事，但这是另外一个话题了。）在面试 Rez 的时候，我们已经说服了一家发行商，并且我们当时正在组建一个团队。

不管你是否相信，Rez 当时面试的根本不是工程师的职位，而是对游戏设计感兴趣的 QA 负责人。当 Rez 走进我们的办公室时，我们注意到的第 1 件事并不是他的身高（尽管他非常高），也不是他的军靴、老式的黑军装或褪色的黑色迷彩服，而是他 25.4 厘米的蓝色莫霍克发型。

接下来我们注意到的是他对我们这四五名面试官轻松自如的态度。你可能觉得这没有什么大不了的，但如果把你放在当时的场景中你会怎么样呢？因为我们所有人都更有经验，都拥有大学本科以上的学位，并且都没有留着莫霍克发型。而且，这是例行公事才进行的面试。

我们注意到的第 3 件事情是他对游戏的热情和拥有的知识。他不只是玩了很多游戏，而且他非常有思想，能够清楚地表达他喜欢什么和他喜欢这些的原因。

我们注意到的第 4 件事情是“Farmer Bill's Almond Farm”。这是 Rez 学习游戏编程时自己制作的演示游戏。虽然它只有简单的图形和简陋的界面，但它充满了创意和乐趣。我们通过它看到了一个非凡的天才。（如果我们足够聪明的话，在 Farmville 面世前的几年，就已经在 Facebook 上发行这款游戏了。）

正如你所猜测的那样，我们雇用了 Rez。在我继续下一阶段的介绍之前，我想告诉想要进入这个行业的你们一些事情。

首先，QA 是进入游戏的一种好方法。它的门槛通常要比其他职位低，而且你学到的技能也适用于游戏开发的很多学科——编程、设计、项目管理和制作。

其次，可以说明游戏为什么有趣的能力非常重要。很多公司在大多数职位的面试中都很看重这一点。它不仅可以确定你是否有激情、是否有在游戏中实现它的决心，更重要的是，能看出你是否能“完成”这个产品。游戏工作室招聘各种人才来创建一款包含这么多声音和图片的产品，就是为了吸引和取悦用户。任何人做出的决策都可能会影响产品的最终体验。人们需要在有限的监管下独立自主地完成工作。不犯错误最保险的方式就是确保所有的员工都有相同的远见。毕竟我们都是知识型工人。我们要做的就是做出明智的决定。

完成产品并不意味着喜欢它。Rez 和我给男女老少都开发过游戏。虽然我们无法马上了解他们的需求，但我们可以转变角色并了解他们想要什么。了解并清楚地说明游戏为什么吸引玩家的能力是“完成”

产品的很大一部分。

再次，莫霍克发型在游戏行业中非常酷。我们是创新型行业，自我表现是非常受欢迎的。

最后，展示一些你创建的东西会起到很好的作用。它比语言本身更有说服力，可以很好地说明你的热情、能力和愿景。而且，制作一个演示游戏可以提高你的能力和自信心。（顺便说一句，本书可以帮助你创建能给你真正自信的演示程序。）

Rez 刚开始呆在 QA 部门，但我们承诺他：只要他能证明自己，他就可以成为一名设计师。没过多久他就证明了自己。Rez 勤奋地检查了很多输入。当他提交了关于 UtilEcon AI 系统的 bug 时，我很快意识到他应该直接调整这些参数（而不用告诉我们）。就是这样，他现在是一名设计师。他很快就开始实现可以让我们更轻松地管理系统数据的工具。不久之后，他就接管了核心系统的代码。

然后，Rez 必须做出决定。他既可以走设计师路线，又可以走工程师路线。Rez 遵循自己的内心，成为了一名工程师。回过头来想一想，我觉得 Rez 已经知道其利弊了。只要有好想法，游戏公司的所有人都可以提出设计意见，但只有程序员可以告诉机器要做什么。

Rez 开始专心致志地编写代码，很快他就成为了我们公司的明星。在接下来的几个月，Rez 参与开发了我们的很多关键系统，如图形、动画、故事事件和用户界面。他对学习的渴望是无止境的，和他合作非常愉快，他会竭尽全力完成他的项目。

离开 Super-Ego Games 工作室后，Rez 在“Planet Moon”待了一年。他曾经参与开发了一款名为“Brain Quest”的 DS 游戏，这是一款小孩玩的游戏。之后，他开始开发“描绘生命：下一章”（Wii 平台）的 AI、动画和保存游戏系统。他的下一站是 Slipgate 工作室，他做的是一款 MMO 游戏的客户端编程和一些 UI 工作。接下来他又跳槽到 Play First 公司，在这个公司他参与开发了 iPad 端的“疯狂烹饪”（Diner Dash）和 iPhone 4 上的“婚礼进行曲”。Rez 现在待在 EA，他原来做的是“模拟人生：中世纪”和“模拟人生中世纪：海盗与贵族”扩展包的 AI 工作。现在，他是一款新“模拟人生”游戏的 AI 程序员。

和 Rez 共事过的所有人对他都有相同的印象。他比任何人都更有干劲，即使在项目开发最紧张的时刻他也会保持乐观和幽默的态度。他的热情颇具感染力。和他在一起，你会觉得游戏编程是世界上最有趣的事情。如果你让 Rez 解释他最近的项目，一定要保证你昨晚睡了一个好觉并喝了很多咖啡，因为他会告诉你很多内容和想法。

幸运的是，你通过阅读这本书就可以吸收他的思想。本书可以给你提供游戏开发核心领域的很多有趣且具有挑战力的工作，这些内容是这个想要与大家分享该领域的人花了很多时间精心挑选的。

享受这一切吧！

——Bo Lasater，Kixeye 执行制片人

致谢

Mike 的致谢

妈妈和奶奶 Hawker

谢谢你们从来不说我把太多时间花在玩游戏上，你们一直很相信我，你们的信任确实得到了回报。

爸爸和 Lynn

谢谢你们告诉我不应该害怕艰苦的工作。

Phil Hawker

谢谢你给我幽默感——我认为我在这里让它物有所用了。

Warren Spector 和 Richard Garriott

谢谢你们相信一个羞涩的大学生可以帮助制作我喜欢玩的游戏。

第 4 版 Beta 测试人员

James Leitch 和 Sascha Friedmann

封面设计师

封面是 Tre Ziemann 设计的。他现在是美国得克萨斯州奥斯汀 King's Isle 公司的一名 3D 美工。

第 4 版编辑

感谢策划编辑 Heather Hurley 让本书的第 4 版得以出版。

感谢我的编辑 Marta Justak 让我更像一个作家。

Rez 的致谢

我的爸爸 Robin Graham

谢谢你送我第一本编程书、第一台自己的计算机，谢谢你给我介绍科幻小说。

我的妈妈 Susan Angelos

谢谢你让我走自己的人生道路，谢谢你从来不觉得我玩电子游戏会毁掉我的人生。

Bo Lasater 和 Steve Matthews

谢谢你们雇用了一个虽然充满激情、但没有任何学历和经验的孩子。我的成功很大一部分原因是因为你们提供给我这个机会。

Steph Laberis

谢谢你支持另一个会耗费我大量时间的项目。

漫画

插入漫画是由 Steph Laberis 设计的。她现在住在加利福尼亚州伯克利，是一名插画家和角色设计师。

Robin McShaffry

谢谢你让 Mike 出来玩。

作者简介

Mike McShaffry 又名“Mike 先生”，他刚会敲键盘的时候就开始了游戏编程——实际上，他为了在一台老式 Commodore Pet 上用 BASIC 语言编写游戏而翘了 7 年级的数学课。为了一心一意地学习编程知识，他签订了休斯顿大学的一份延时毕业协议。让他和数学系主任吃惊的是，他实际上在 5 年半后才正式毕业。毕业后不久，他就进入了电脑游戏行业的新兵训练营：Origin Systems。他与 Warren Spector 和 Richard Garriott（又名“不列颠之王”）一起开发了《Martian Dreams》《创世纪 VII：暗黑大门》《创世纪 VIII：异教徒》《创世纪 IX：升腾》和《网络创世纪》。

在那里工作了 7 年后，Mike 辞职了，并于 1997 年创办了第一家公司：Tornado Alley。Tornado Alley 是一家从车库里出来的初级创业公司，它的目标是为孩子们制作一款“成年人禁止入内”的多玩家世界——这让 Mike 和 Tornado Alley 的所有员工都成为了一次国会听证会的中心。

创业的错误让 Mike 抛售了他剩余的 EA 股票，并促使他在 Glass Eye Entertainment 找一份稳定的工作，为他的朋友 Monty Kerr 工作。他在那里开发了 Microsoft Casino。短短 10 个月后，Monty 就要求 Mike 和他新组建的团队开办他们自己的公司：Compulsive Development，它专门和微软合作开发棋牌类游戏项目。

到 2002 年 8 月，Mike 作为工作室的咖啡师和领导，与其余的 20 名成员为微软开发了 3 款棋牌类游戏。Compulsive 后来被 Glass Eye Entertainment 收购了，继续负责 Glass Eye 增长的在线休闲游戏业务。

Mike 很渴望 3A 级的游戏机编程工作，2003 年他终于得到他梦寐以求的工作：Ion Storm 的《神偷：致命阴影》团队邀请 Mike 创建他们的第 3 人称摄像机技术，并在最后时刻调整角色的运动。这个为期两周的合同最终变成了与程序员亲密合作的 1 年。

虽然成为“其中的一分子”是一件很棒的事情，但它不可能永远持续下去。Mike 被招募来为总部在马里兰州的 BreakAway Games 在奥斯汀开创一家工作室。BreakAway Austin 的重点是 3A 级游戏机开发和美国军事和国防部的高端模拟。Mike 和 BreakAway Austin 的 3 个团队真的去过杜鲁门号航空母舰，它是美军的一种 CVN 级核动力航空母舰。他们坐的飞机落在了航空母舰上，与士兵共度了 4 天后回了家。后来他们创建了 24 蓝（24 Blue），一种用于模拟航空母舰甲板、飞机等各种环境的训练模拟器。

在离开了 BreakAway Austin 后，Mike 成立了一家咨询公司：MrMike。他认为他在游戏行业 18 年的经验足以确立良好的公司形象了。在两年的时间里，他帮助很多小游戏公司选择自己的游戏技术，巩固他们的生产实践，并向微软、EA、THQ 等公司推销游戏创意。他的一个客户——Red Fly 工作室——给他提供了一个他无法拒绝的 offer，因此他又做回了全职工作。

Mike 接受了执行制片人的工作，并帮助发行《蘑菇人：孢子大战》。他现在仍然在 Red Fly 工作室工作，他现在的职位是产品总监，偶尔兼任咖啡师。他仍然喜欢制作咖啡，并尝试给程序员、美工、设计师、音频工程师和制作人提供好建议。

他仍在坚持编写代码——最近在学习使用 Unity 游戏引擎和 C#，并尝试着改进 GameCode4 引擎。

如果 Mike 没有坐在电脑前敲键盘，那么他肯定是在骑着他的山地自行车到处溜达，或者与他的朋友在德克萨斯州奥斯汀享受美好的时光。

David “Rez” Graham 是一名自学成才的程序员，自从他可以拿起电子游戏控制器后他就成了一名

狂热的游戏玩家。他一直对游戏着迷，1996 年他的父亲送给他人生中第一本编程书籍。Rez 贪婪地阅读这本书，很快他就开始尝试编写自己的游戏了。他在 6 个月的时间内编写了 5500 行代码，完成了他的第一款游戏《Farmer Bill's Almond Farm》。这是一款在 Dos 6.2 上运行的、非常简单的冒险类游戏。Rez 并没有停下来，他继续编写游戏。

1998 年，他成功进入电子游戏行业，成为了《模拟城市 3000》的一名游戏测试人员；然后，他进入了 Microprose 公司的技术支持团队。20 世纪 90 年代该工作室关闭之后，Rez 暂时离开了游戏行业，开始管理柯达的一个 IT 团队。

到了适合回到电子游戏行业的 2005 年，Rez 在 Super-Ego Games 工作室找到了一份工作，他刚开始是作为他们源代码控制系统的一名 QA 工程师，不久后他就开始从事与设计和工程相关的工作。在之后的一个月时间里，Rez 都在为《明争暗斗》编写 AI 代码。他在这个工作室待了 2 年，在此期间他发行了一款儿童游戏:《芭比高校生冒险日记》,他开发了几个迷你游戏并扩展了 AI 系统。其余的时间，Rez 都在开发《明争暗斗》(PS3 平台) 的各种系统。2008 年年初，Rez 离开了 Super-Ego Games。他在 Planent Moon 待了一年，在这一年的时间里他参与开发了一款很小的儿童游戏《Brain Quest》(Gameboy DS 平台)。之后，Rez 负责《描绘生命：下一章》的 AI、动画和游戏保存。

2009 年，Rez 跳槽到了“Slipgate”公司，在那里他负责的是一款 MMO 游戏的客户端。离开 Slipgate 后，Rez 在一家名为“PlayFirst”的公司开发 iPhone 和 iPad 平台上的休闲类游戏。他参与开发了《疯狂烹饪》(iPad 平台)，并成为《婚礼进行曲》(iPhone4 平台) 的主工程师。

现在 Rez 在 EA 工作，他是一款即将发行的《模拟人生》游戏的主 AI 程序员。他在 2010 年年中跳槽到 EA，他最近发行的一个项目是《模拟人生中世纪：贵族与海盗》扩展包。Rez 已经在“游戏开发者大会”上进行了多次发言，并经常给高中生和大学生讲述如何进入游戏行业。

在业余时间，Rez 喜欢玩桌面角色扮演游戏、听歌，尝试不同的项目和 AI 实验。

前言

欢迎阅读本书第 4 版

本书的第 1 版出版于 2003 年的夏天，当时的我正在经历人生中重要的变革。第 1 版给了我这样的机会，让我可以以旁观者的身份告诉程序员游戏编程世界发生了什么。写这本书是一个挑战，但我得到了很多回报。我听到世界各地的程序员说他们喜欢这本书，我发现这些故事、见解和编程技巧对他们很有帮助。第 2 版几乎是一次完全的改写，总共有 1100 页（第 1 版只有 700 页），而且它比第 1 版更受欢迎。2009 年，我在 James Clarendon、Jeff Lake、Quoc Tran 和 David "Rez" Graham 的帮助下出版了第 3 版，其中添加了 AI、多道程序设计、Lua 和 C#等内容。

3 年后，我给我的出版商 Cengage Learning 打电话，问他们要不要出版本书的第 4 版。他们说要，但我必须想办法挤出写这本书的时间。

我自然而然地想到了 Rez，他是我的一个朋友，也是第 3 版 AI 章节的合著者，还在本书网站上粘贴了关于线程的内容。我打电话给他，但我没有得到我想要的答案。他不仅想要帮忙，他想和我一起写这本书，因此他成为了我的合作伙伴。

现在你手里拿的这本书就是我们的成果。

代码在哪里？我必须输入吗

本书的第 1 版出版后不久，我就创建了一个网站来给读者提供资源和有用的信息。这个网站也成为了下载本书源代码示例和各种有趣内容的地方。从第 1 版开始这个网站就一直在"成长"，现在它已经成为了一个资源中心。因此如果你想要得到额外的帮助、源代码，或者你想要和其他游戏程序员分享你的想法，可以访问下面的网站：

www.mcshaffry.com/GameCode/

www.courseptr.com/downloads

这本书没有配套的 CD，因为我们在书出版后的很长一段时间内还会修改和调整源代码。有些好建议和修改甚至是来自读者。从 GameCode（或出版商的）网站下载代码，你肯定能得到最新的源代码和信息。

本书的组织结构

本书分为 4 个部分。

- **游戏编程基础（第 1～4 章）：**展示一些你在游戏编程工具箱中想要的东西，例如一个优秀的随机数生成器。它还介绍了游戏的主要组成部分和它们如何交互的。阅读完这部分内容后，你就可以了解游戏开发人员使用的实际架构了。

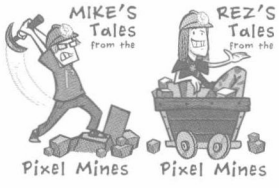
- **让游戏跑起来（第 5～9 章）：**现在是时候介绍如何将游戏的主要构件块组合起来，包括初始化和关闭代码、主循环、游戏主体、用户界面和输入设备代码。你会发现第一个让人深思的游戏代码实例。很多编程书籍通常都会把这些内容搪塞过去并立刻跳到很酷的 3D 代码。但实际上，不管你想要开发哪种类型的游戏，这部分内容都是你真正需要的。
- **核心游戏技术（第 10～18 章）：**这个部分包括很多更复杂的代码示例，例如，3D 编程、使用 Lua 编写脚本、游戏音频、物理和 AI 编程。
- **高级主题和将所有这些内容组合起来（第 19～24 章）：**在这一部分，你会发现网络、线程编程、用 C#创建工具和将本书的所有代码结合起来制作一款游戏等相关内容。我们还介绍了很棒的调试技巧，并用一整章内容来介绍当你发布一款商业游戏时的内心感受。

在本书中，你会看到很多用漫画图像标志的插入片段：

"Gotcha"介绍一些你应该注意的事项，很可能是因为 Rez 或者我已经犯了这个错误。但我们希望你可以避免它。

"Best Practice"（最佳实践）是多年来来之不易的经验教训。遵循这些"最佳实践"，你会很乐意这么做的。

Rez 和我都有很多关于游戏开发的故事。我们很喜欢在对方的章节中插入一些自己的故事，因此你需要通过这些卡通图像来加以区分。



你需要什么

如果你是一名程序员并且已经有了一些游戏编程经验，那么你就可以继续阅读本书。花一点时间浏览一下，你就会发现这本书是写给程序员的。当然，非程序员也可以从本书中学到一些知识，但本书中包含很多代码。

这些代码是用 C++、Lua 和 C#编写的。如何你不懂这些语言，那么示例代码可能会让你费些力气。但你肯定能从注释和解释中得到足够的理解，让你觉得没有花冤枉钱。

本书中的所有代码都是在 Visual Studio 2010 中运行的，至少当我们将它们复制到 Microsoft Word 中时，它们可以在 Visual Studio 2010 中运行。Rez 和我就是用这种方法写书的。我为我没有尽力保证这些代码在其他编译器中运行而道歉，例如 CodeWarrior 或 GNU C++。我希望能够得到你的原谅。我们认为

更应该将时间花在涵盖更多技术领域，而不是多编译器兼容的代码上。

Lua 的编写使用的是 Decoda IDE。由于 Lua 不是一种编译语言，因此你不必使用特殊的编辑器，记事本就可以了。但 Lua 脚本中包含一个 DEPROJ 文件，因此如果你使用的是 Decoda，那么项目已经为你准备好了。

本书中的代码有非常强烈的 Windows 倾向。我是一名 Windows 程序员，而在此之前我是一名 DOS 程序员。我曾经在 UNIX 上编写过《网络创世纪》的服务器端代码，但我远不是专家。本书中的大部分代码都假定你使用的是 Windows，和我选择一个编译器的原因相同，我没有修改代码来让它们支持其他操作系统的交叉编译。对于我来说，涵盖更多技术问题要比在 Linux 中检查代码好得多。

至于图形 API，我假设你使用的是 DirectX 11（或后续的版本）。代码支持 Direct3D 9 和 Direct3D 11，但本书只涵盖了 Direct3D 11。当然，我并不是反对 OpenGL，我只是不太擅长其细节。总之，如果你非常了解 C++、C#、Windows 和 DirectX，那就足够了。你的技术不必达到登峰造极的程度，但你至少应该能熟悉这些编程领域。

如果你是一个新手而且可能只了解一点点 C++，那也不要灰心丧气。我给你做了一个规划。在本书中，我会提到很多帮助我学习如何编程的书籍。它们也能帮助你，你可以同时阅读这些书籍。稍微集中一下注意力，你就能学到精湛的编程技术。我是通过代码来学习 C++、DirectX 和 Windows 编程的，本书中包含很多这类代码。

第三方库

本书使用 STL 作为通用数据结构。如果你不了解任何 STL 的相关内容，只要阅读完本书的示例即可，我敢保证你一定可以看懂这些示例。我不会尝试教你一些 STL 知识，这超出了本书的范围。你可以阅读一下 Nicolai M. Josuttis 写的《C++标准程序库》。当你觉得差不多的时候，再去阅读一下 Scott Meyer 写的关于 STL 的书籍，因为它们都非常棒。

STL 是一套经过良好测试的代码，有着广为理解的 API，在所有的平台上几乎都能使用。如果你还没有见过它，就马上放下书先研究一下吧。你永远不必为通用数据结构（例如，链表、动态数组、树）编写代码。使用<list>、<vector>和<map>可以省去你很多麻烦。

不管发生什么，只要可以使用 STL，就不要自己编写链表或树。STL 中所有的实现都经过了良好的测试。每个 bug 或实现中奇怪的地方都已经在互联网上被公开讨论过。相反，你自己的代码则没有。

源代码和编码规范

我很鄙视那些含有无法编译代码的技术书籍。我曾经诅咒过那些创造了充满错误和崩溃代码书籍的作者和编辑。现在，我却要加入他们的队伍。

Microsoft Word 不能很好地处理 C++源代码。由于本书是黑白打印的，因此代码的高亮显示必须被关闭。我现在理解了为什么那么多编程书籍都充满错误。我向每一位我曾经诽谤过的作者和编

辑道歉。直到我自己开始写书时，我才知道这有多困难，现在 Rez 也这么觉得。客套话少说！Rez 和我会一遍遍地检查本书中的代码，如果发现任何问题我们都会尽力修正。

现在我的良心好受一点了，你应该知道如何阅读本书中的代码了。

代码来自哪里

每一行源代码都源于真实的游戏。当然，这些代码并不是一字不差的。否则，我的大门会被一波波来自微软、EA、Mattel、Eidos 的律师挤破。你应该了解，Rez 在进入项目开发前先要签署一份协议。相反，为了保护雇佣 Rez 和我的那些公司的知识产权，我们对这些代码做了足够的修改。说实话，原始代码更难阅读。它通常包含我用任何方法都不可能包含进来的优化和外部引用。由于它们几乎是 30 年间不同编程经验的结合，你应该可以想象得到其风格和结构是多么杂乱。如果你想开发自己的游戏，那么本书的源代码应该可以给你提供一个良好的开端。你会发现本书中的某些框架很适合你自己的代码。我甚至希望书中的代码能够解决一些令你头痛的难题，让你把注意力放在游戏上。

本书的代码是在 Windows 平台上、使用 DirectX 9 和 11 应用框架、在 Visual Studio 2010 中编写和测试的。游戏主机编程是另一回事，我会在合适的位置指出这些差异。如果你想要在一台安装了 Windows 系统的机器上使用这些代码，并了解在 Xbox360、PS3 或 Wii 上编写这些代码有什么不同，你就选对书了。

本书的源代码基于 GNU 较宽松公共许可证。你可以在 http://www.gnu.org/ licenses/lgpl.html 上阅读该协议，它基本上意味着只要你相信 Rez 和我，你可以随意使用这些代码。如果你够疯狂的话，甚至可以在商业游戏中使用这些代码。但不要说 Rez 和我没有提醒你。

编码规范和风格

源代码规范非常重要。我并不是一个规范独裁者。我会给其他代码风格留出空间，必要时我很乐意采用其他合理的规范。我把它看成是我出国旅游时学的一些当地语言。当地人会很喜欢，你可能也能学到点东西。

Origin Systems 没有公司范围的编码规范。我在那里工作时，我至少是 3 种规范组织的成员。每次我们试图讨论 C++大括号风格时，会议都会变成一场大吵大闹的比赛。Origin 中有很多程序员不能接受其他人的风格。悲哀的是，甚至有人写了一个解析源文件的小工具来修改括号形式。真是疯了！

你的编码规范和风格只是为了给其他程序员或者是未来的自己传递有用的信息。

Rez 和我在本书中使用的编码风格和我们工作时使用的一样。唯一的区别是它们让代码变得更容易阅读。例如，本书中的源代码常常会消除明显的错误检测和处理。如果我们像在实际项目中那样使用每一行源代码，这本书的页数就会增加一倍。这是一次困难的取舍，但我觉得最好还是多给出一些示例，而舍弃那些很明显的东西。

使用前缀

Visual Studio 这类现代 IDE 会用一个工具提示来显示标识符类型，因此程序员不再需要使用会弄乱前缀的多余信息。相反，前缀主要用于显示范围。前缀的说明如下所示。

- g：用于全局变量——g_Counter。
- m：用于成员变量——m_Counter。
- p：用于指针变量——m_pActor。
- V：用于虚函数——VDraw()。
- I：用于接口类——class IDrawable。

我曾经见过某些人疯狂地使用前缀，把 3 个或者更多字母放在一个标识符前面。用匈牙利命名法编程一定很困难。这种风格的问题在于，有相同前缀的标识符看起来很相似。这就是为什么前缀越小越好，且一定要用一个下划线分隔标识符的原因——它可以传递有用的信息，而且不会破坏变量名的唯一性。在你自己的代码中，你可以将更多的前缀加到这个列表中。只要不把问题极端化即可。

作用域前缀是前缀的一个妙用。任何修改了全局作用域变量的程序员都应该被扇一个耳光以示警告。类成员变量和局部变量的作用域不同。当在同一个函数（如，构造函数）中使用它们时，“m”前缀是区分局部变量和成员变量的简单方法。

虚函数非常有用，但如果使用不当就会非常危险。虚函数的前缀可以提醒程序员他们应该调用父类的重载虚函数，而调用这个函数的开销非常大。

我认为给只定义了纯虚函数且没有任何数据成员的接口类使用前缀非常有用，这样程序员就可以安全地对它们进行多重继承。我尽量避免非接口类的多重继承，我建议你也这么做。否则，代码就会变得非常混乱并难以维护。

字母大写

我使用字母大写来区分不同类型的标识符，让它们更容易阅读。

- 参数和变量：总是以小写字母开头，每个混合词都使用大写字母开头——g_BufferLength、m_BufferLength、returnValue。
- 类、函数、类型定义和方法：总是以大写字母开头，每个混合词都使用大写字母开头——SoundResource、MemoryFile。
- 宏和常量：全部大写，用下划线来分隔混合词——SAFE_DELETE、MAX_PATH。

前两个大写风格可以帮助程序员区分类的定义和这些类的实例：

```
SoundResource soundResource;
MemoryFile memoryFile;
```

宏通常是痛苦和折磨的根源，应该以大写格式明显地声明它们。如果你想要找到一个宏的定义，只要简单地搜索#define 宏名即可。这让它们与函数和方法区分开。

正确使用 const 代码

我会尽最大努力在代码中正确使用 const，本书中也不例外。但我敢肯定你们当中一些绝对要求 const 正确的程序员会在我忽略的地方放入几千个 const 关键字。保证 const 的正确性相当痛苦，但它非常重要。给成员变量、函数返回值、指针和引用加入 const 可以给其他程序员传递有用的信息。

字符串和本地化

如果你开发的游戏只面向说英语的人，你就限制了销路。欧洲和亚洲、尤其是中国大陆都非常渴望高质量的游戏。虽然大多数玩家可以忍受，但他们更希望看到高质量的母语翻译版本。好的本地化技术需要用一本书介绍，还需要一门外语硕士学位。

我在本书中倾向于使用 std::string 和 std::wstring。它是一个非常有用的字符串类，虽然并非所有人都这么认为，但它是让我觉得最舒服的一个。

在真实游戏代码中使用字符串的最后一个注意事项是：调试字符串或对象名时，使用文字也没有关系。你可以将它们声明为：

```
if (impossibleError == true)
{
    OutputDebugString(_T("Someone enabled the impossible error flag!"));
}
```

注释

好的代码能自然地说明问题，我希望本书中的代码也是如此。好的变量名和逻辑应该可以免除文字说明的必要。在本书中，我在我认为需要的地方加入一些注释，但是你通常都可以在紧接着代码的下一段看到详细的解释。

在真正的游戏中，你应该在代码中加入足够的解释（可能是在文件开头），这样其他程序员才能弄清楚到底发生了什么。你现在输入代码中看起来觉得很明显的东西，随着时间的流逝会变成令人迷惑的混乱。对于我来说，在我编写一段代码的 3 个月后，我就搞不懂它们是要干什么了——如果我自己都搞不懂，怎么能指望别人理解它呢？

在开始一个项目时，我总是会在代码中加入一些良好的注释。而在项目结束时，我总是会对我自己和其他程序员感到失望——我们的确没有足够的时间。这已经司空见惯了。面临压力的项目中，注释会消失，因为程序员将全部时间都用在疯狂地编写代码上。最好的方法是以一种轻量级的注释策略开始，并尽可能地长时间坚持它。如果注释在项目的某个阶段出现缩减，在项目结束后，要尝试着回到代码中给它们加入注释。我在微软的一个好友告诉我说，发行产品时是添加注释的好机会。我强烈赞同这一点。