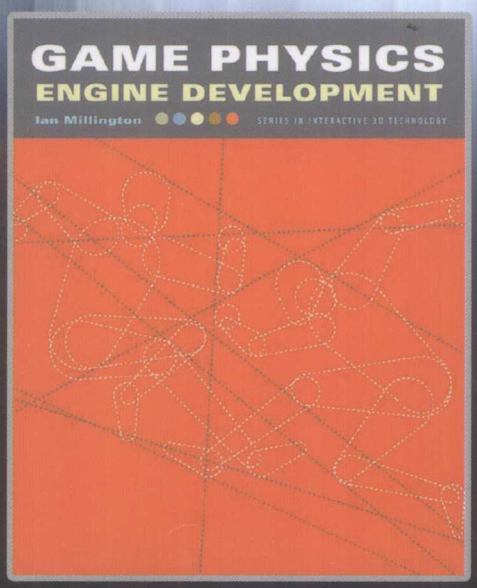
CRC Press
Taylor & Francis Group

游戏物理引擎开发

(美)Ian Millington 著
徐明亮 李强 宋伟 译

Game Physics Engine Development



清华大学出版社

游戏物理引擎开发

（美）Ian Millington 著

徐明亮 李 强 宋 伟 译

清华大学出版社

北 京

内 容 简 介

本书详细阐述了与游戏物理引擎相关的高效解决方案及相应的数据结构和算法，主要包括粒子数学、运动定律、粒子物理引擎、合力、弹力、硬约束条件、质体物理引擎、旋转操作的数学知识、刚体运算定律、刚体物理引擎、碰撞检测、生成碰撞、碰撞处理方案、静态接触和摩擦力、稳定性和优化问题、整合方案以及其他物理引擎等内容。此外，本书还提供了相应的算法、代码以及伪代码，以帮助读者进一步理解相关方案的实现过程。

本书适合作为高等院校计算机及相关专业的教材和教学参考书，也可作为相关开发人员的自学教材和参考手册。

图书在版编目（CIP）数据

游戏物理引擎开发/（美）米灵顿（Millington, I.）著；徐明亮，李强，宋伟译. —北京：清华大学出版社，2013
书名原文：Game Physics Engine Development

ISBN 978-7-302-34456-8

I. ①游… II.①米… ②徐… ③李… ④宋… III. ①三维动画软件-游戏程序-程序设计
IV. ①TP391.41

中国版本图书馆 CIP 数据核字（2013）第 275637 号

责任编辑：赵洛育
封面设计：刘　超
版式设计：文森时代
责任校对：王　云
责任印制：李红英
出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京富博印刷有限公司
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　张**：21.75　　**字　数**：503 千字
版　　次：2013 年 11 月第 1 版　　**印　次**：2013 年 11 月第1 次印刷
印　　数：1～4000
定　　价：79.00 元

产品编号：035590-01

译 者 序

在当今游戏产业中，游戏软件的开发都是基于游戏引擎技术，游戏引擎在游戏中起着“发动机”的作用。游戏引擎把开发中可能遇到的大多数技术难点以非常规的编程技巧高效地实现，并封装成易用的通用运行框架以及相关的辅助工具集。经过多年的发展与完善，游戏引擎已经发展和进化成为由许多个相互联系但又相对独立的子系统耦合而成的复杂系统，主要包括图形渲染系统、资源管理系统、音频与视频系统、物理引擎系统、网络通信系统、脚本系统、人工智能系统等。

这其中，物理引擎是游戏引擎技术的重要组成部分，它侧重实现游戏世界中的物理效果，如基本的运动建模、光影效果、碰撞检测，以及复杂的群组控制、流体模拟、粒子特效等，从而使得游戏虚拟世界中的表现效果具有真实感。随着游戏行业的迅猛发展，游戏玩家对游戏中表现效果的真实性与合理性提出了越来越高的要求。因此，在游戏体验中，如何缩小游戏世界与真实世界的对比差距，游戏物理引擎将起到决定性的作用。

本书的译者长期从事游戏引擎技术方面的研究与应用工作，在工作中深感开发和用好一个物理引擎极为不易。同时，国内目前尚无一本全面、深入、系统介绍游戏物理引擎的专业教程。因此，本书的引进和翻译出版非常及时。中国有着庞大的游戏用户群，但在游戏开发方面，我们与世界一流游戏开发水平仍存在较为明显的差距。目前，我国游戏产业的发展日益得到巨大关注，已经上升到信息产业的战略高度，并表现出强劲的发展势头。希望本译著对进一步推动我国游戏产业相关领域的研究与应用起到推动作用。

本书共有 18 章和 4 个附录，其中第 1~8 章和第 9~14 章分别由郑州大学信息工程学院软件工程系徐明亮副教授和宋伟副教授翻译，其余章节由李强翻译。杭州师范大学特聘教授许威威博士对全书译稿进行了审定和指导。除此之外，王晓晓、梁洪娇、李亚楠、姚楷锋、孙年果、丁妍、李中卉、王典、胡子文、朱利平、刘凌、史云龙、李保金、皮雄飞、孙健、王巍、程聪、李海俊、赵雷、潘冰玉、王梅、朱琳琳、吴帆、米玥、李莉、张欢欢也参加了本书的翻译工作，在此一并表示感谢！

本书在翻译过程中，历经艰难，数易其稿，期间得到了许多同事、朋友的关心与支持，有许多人都为本书的翻译付出了大量的心血和劳动，虽不能一一列出，但在此请允许译者对他们表示诚挚的感谢！

本书的翻译工作得到了国家科技支撑计划课题（2013BAH23F01）、国家自然科学基金（61202207 和 61272392）、中国博士后科学基金（2012M520067 和 2013T60706）、教育部博士点基金（20124101120005）、浙江省科技厅公益性技术应用研究计划（2012C21028）、河南省教育厅科学技术研究重点项目（13A520414 和 13A520453）的支持，在此表示感谢！

由于技术和英文语言理解方面的水平所限，书中难免会存在一些错误和不当之处，敬请读者批评指正，我们将不胜感激。

前　言

本人于 20 世纪 80 年代踏上游戏编程之路，当时正是 8 位机风行的年代，低预算与较短的开发周期鼓励研发人员不断尝试和创新，其中不乏某些游戏精品。至今，我依然对两款游戏印象深刻，它们均采用了真实的物理行为作为游戏体验的核心内容。

第一款游戏由 Jeremy Smith 开发，并针对英国 BBC Micro 家用计算机平台而发布，即 Thrust。该游戏根据街机游戏 Gravitar 改编，其中藤叶状船只游弋于地下溶洞之间，且通过 2D 物理模拟对其加以支配。游戏任务是窃取大型燃料棒，并通过线缆连接至船体。随后，相对简单的船体模型演变为两个大型对象之间的交互行为。游戏本身颇具挑战性且令人欲罢不能，因而不失为一款经典游戏。

第二款游戏则是 Exile，由 Peter Irvin 和 Jeremy Smith 联袂开发。该游戏具有一定的创造性且令人印象深刻，游戏所蕴含的技术亮点已超出了物理范畴，例如游戏对象的过程式构建方式。

游戏 Exile 中的物理内容体现于游戏中的各个对象上。其中，发射物沿弹道轨迹行进；玩家可投掷手榴弹，且爆炸冲击波可掀翻周围物体；玩家可负重前行且身体呈弯曲状态；另外，玩家还可漂流于水面上。总之，游戏 Exile 首次采用了相对完整的物理引擎。

尽管游戏 Exile 发布于 1988 年，但其物理编码内容并不落伍。本人于 1999 年涉足游戏物理引擎程序设计，并在一款赛车游戏中负责车辆模型的物理引擎开发，期待中一个月的开发周期最终变得遥遥无期。

开始阶段，物理问题并未引起足够重视，因而产生了各种各样的问题。例如，硬质悬挂弹簧将车辆以螺旋方式弹射至无限远处；摩擦力导致轮胎高速旋转时产生振动现象；硬质表面呈现为软质橡胶材质。对此，我曾尝试了多种处理方案，包括冲量方案、Jacobian 方案、规约坐标方案以及模拟物理方案，其学习曲线不同于之前任何游戏编码经历。

尽管开发时间超出了最后的期限（开发内容为向第三人称射击行为提供物理驱动），但公司人员依然对各个中间件物理系统进行了严格检测，本人也从中了解到各类方案的优缺点。在随后的开发岁月中，本人所编写的代码体现了一定的复用性，并应用于多个不同场合。随着经验的不断积累，本人也研发了多款物理引擎，并为诸多应用制定了相关的物理解决方案。关于如何获取最为简单的方案以及最佳物理效果，我自认为具有正确的判断力。

游戏开发已步入了物理模拟商品化这一阶段，即物理模拟常出现于各类游戏中，且开发公司均会研制内部物理库，或对某一主要的中间件处理方案提供应用许可。尽管物理行为日趋普遍，但其研发阶段仍可视为一个黑盒过程，即开发人员负责实现物理行为，其他团队则直接使用结果。

关于游戏物理学，相关信息以及参考资料均假设读者具有一定的数学和物理知识。然而，某些著作仅罗列了相应的物理内容，且不包含对应的应用框架结构。更为糟糕的是，

一些资料甚至包含了某些错误理论，进而使读者陷入困境。物理引擎通常较为复杂，且存在较大的优化和改良空间，部分内容还有待于进一步探讨研究。例如，在读者尝试实现 Lemke 枢轴算法之前，应于先期理解其基本概念并包含相应的代码测试框架。

本书源自个人初始阶段所经历的种种磨难，同时也希望本书能够填补这一方面的空白，且兼顾逻辑性和清晰易懂之特征，进而引领读者从零开始打造一款属于自己的物理引擎。本书仅是万里长征中的第一步，但却是坚实的一步，因为我们已然拥有了正确的方向。

本书附带光盘

本书附带光盘包含了源代码库，进而实现本书所讨论的各项技术以及示例程序。该代码库易于阅读，并辅以注释和应用示例。

本书附带光盘中的全部内容已上传至 www.tup.com.cn，请读者自行下载。

关于作者

Ian Millington，IPR Ventures 咨询公司合伙人，该公司致力于次世代技术的研发工作，涉及娱乐、建模技术以及模拟仿真技术。在此之前，他曾创办了 Mindlathe Ltd（计算机游戏领域内一家最大的专业 AI 中间件公司），并涉足多种游戏类型以及开发技术。Ian Millington 具有深厚的 AI 专业背景，包括复杂理论和自然计算博士项目研究。除此之外，他还发表了多篇专业学术论文和文章，内容涉及古生物学和超文本技术。同时，他还是 *Artificial Intelligence for Games* 一书的作者（该书由 Morgan Kaufmann 出版社于 2006 年出版）。

致谢

本书旨在打造一款健壮的游戏物理引擎，该过程历尽艰辛并得到了编码人员以及数学家们的巨大帮助，本书的出版源自他们所发表的论文和文章、SIGGRAPH 大会演讲以及所发布的源代码。当然，我的感谢名单远不止于此，这里要着重感谢 Chris Hecker，Andrew Watkin 和 David Barraf，其思想奠定了本书的编写风格。

感谢本书技术评审小组成员所付出的艰辛劳动，他们是 Philip J. Schneider，Jonathan Purdy 博士以及 Eitan Grinspun，他们的建议极大地提升了本书的编写质量、可读性以及有效性。另外，这里还要特别感谢 Dave Eberly，他对细节的关注令我受益匪浅。

本书编写时恰逢 R&D 咨询公司创业期，因而本书献给我的妻子 Mel，感谢她两年来陪我度过无数个不眠之夜。

目　录

第 2 部分 质量集合体物理

第 3 部分 刚体物理系统

第 4 部分　碰撞检测系统

第 5 部分 接触型物理系统

第6部分 扩展引擎

第1章 概 述

物理学一直是计算机游戏的热点，若缺乏物理引擎的支持，动作类游戏便会黯然失色。同时，这一趋势也逐渐蔓延至其他风格的游戏中，包括战略游戏以及猜谜类游戏。针对这一发展趋势，多家中间件公司均提供了高性能的物理模拟器。另外，多款大制作游戏均体现了商业级物理引擎之特征。

然而，商业级软件包通常价格不菲，对于广大开发者而言，“定制型”物理解决方案通常兼具经济性、操控性以及灵活性。尽管如此，物理学依然充满了神秘感，且与数学密不可分，因而令人望而生畏。

2000年，我曾着手设计通用物理引擎，当时的情况可描述为：资料匮乏且无代码可用，同时，该领域内充斥着诸多自相矛盾的信息。天道酬勤，经过不懈的努力，一款商业级物理引擎终于诞生。当然，这一经历本身也是一个自我提高和学习的过程。在随后的5年中，它与其他商业级物理系统频繁地出现于多款游戏中。本书直接体现了作者多年的努力成果和开发经验。

关于游戏物理，相关书籍、网站和文章不胜枚举，然而却鲜有资料介绍物理引擎的解决方案，即适用于游戏中的、较为全面的模拟技术。本书旨在引领读者构建物理引擎，并以循序渐进的方式对其进行分析，进而体现构建过程中的设计决策。在本书的指导下，读者可对示例物理引擎实施进一步扩展，构建自己的物理系统，进而展示不同的设计理念。

1.1 游戏物理

物理学包含了丰富内容，并可进一步划分为数百个子学科，各学科对物理学中的不同问题予以关注，例如光学或星球中的核反应。

其中，某些内容适用于游戏，例如光学原理。据此，读者可尝试模拟光线的传播和反射行为，以使图像具有更为真实的外观——这正是光线跟踪的用武之地，在多个场合中均可看到它的身影。然而，尽管上述内容隶属于物理学范畴，但并非是本书所讨论的游戏物理学，本书将对此加以深度考察。

某些物理学知识与本书并不存在直接关联，例如，如果游戏的卖点并非是展示核反应这一复杂的技术，核物理模拟往往较少出现于游戏中。

实际上，游戏物理仅涉及物理学中的某些经典理论，即重力和其他作用力模式下的用于控制物体运动方式的物理定律。在学术领域内，部分定律已被最新理论所替代，例如相

对论和量子论。在游戏中，此类定律常用于展示实体对象的真实运动行为，包括质量、惯性、反弹行为以及浮力。

游戏物理学基本上与首批游戏同时诞生，并用于模拟粒子的运动方式，包括火花、爆竹、弹道模拟、焰火以及爆炸效果；近30年来，物理模拟还用于飞行模拟器中；近期，物理模拟广泛出现于汽车工业中，进而提升轮胎、悬挂系统以及发动机模型的精度。

随着处理性能的不断提升，视觉效果体现了更为丰富的内容。例如，箱体可随意移动、堆放；墙体被摧毁时所展现的灰飞烟灭效果——此类视效均属于刚体物理学范畴，读者可稍加扩展，即可获得“软物体”的行为方式，例如布料、旗帜以及绳索。近期出现的布娃娃系统即模拟了人类骨骼系统的行为方式，包括绊倒、跌倒以及挣扎时的各种动作。

本书将对物理学内容进行系统、全面的介绍，并不断完善物理引擎中的相关组件，进而支持粒子效果、飞行模拟、车辆物理行为、箱体对象的运动、对象被摧毁时的视效、布料以及布娃娃系统等内容。

1.2 物理引擎

尽管游戏物理学可视为一类新生事物，但近30年来，其实现方式发生了显著的变化。最初，各类视效仅对自身内容进行编程设计，对应物理学内容仅局限于某一游戏作品。例如，弓箭沿某一轨迹运行，开发人员针对该轨迹方程予以编程实现。除此之外，该方程别无他用。

针对简单的模拟操作（代码量较少且物理应用范围有限），该过程工作良好。稍后将会看到，基本的粒子系统将包含数百行代码，随着复杂度的不断增加，直接获取令人可信的物理效果将变得越发困难。例如，在首款Half-Life游戏中，当玩家移动箱体时，错误的物理实现代码使得箱体对象以一种奇怪的方式运动。物理效果的实现难度以及多款游戏中的同一效果迫使开发人员寻找通用的解决方案，进而实现代码的复用。

复用技术体现了某种通用特征，例如，弹道模拟器仅处理弓箭的运动轨迹，并于随后对其进行硬编码。若同一段代码需要处理子弹的运动行为，则软件系统须对特定的弹道轨迹进行抽象，进而生成具有某种共性的物理方案——这体现了物理引擎的含义，即基于物理学的公共代码段，而非针对各游戏方案进行程序设计。

这里的问题是，若存在特定代码对弓箭对象进行模拟，则该操作可胜任模拟过程中的全部任务；若采用通用引擎模拟任意弹道轨迹，针对当前弓箭对象，则需向引擎提供该对象的各类特征，因而需要设定弓箭、子弹以及箱体的属性。

这一重要差别须引起读者的关注。通常情况下，物理引擎可视为一类大型计算器，即执行数学运算并对物理行为进行模拟，且并不了解模拟的具象内容。对此，除了引擎自身之外，读者还需提供表达游戏关卡的特定游戏数据。

尽管各类游戏数据将贯穿于本书中，但本书重点并非讨论数据的获取方式。在商业级

引擎中，关卡编辑器使得设计人员可方便地设置箱体、旗帜、布娃娃对象以及飞行器对象，包括重量、在空气中的行进方式以及弹射行为等内容。

本书所讨论的物理引擎须通过渐增式数据对其进行驱动，此处假设相关数据业已存在。随着内容的不断深入，本书将对数据类型及其合理值进行详细分析。针对游戏中特定对象的各项属性，本书暂不对其开发工具进行讨论。

1.2.1 物理引擎的优点

物理引擎的优点主要体现在以下两方面：首先，物理引擎可缩短开发周期。若计划在多款游戏中加入物理效果，则可将其一次性地植入物理引擎中，并于随后简单地将其导入至各新增项目中。另外，本书所讨论的轻量级、多功能物理引擎其程序实现亦相对简单，数千行代码即可实现游戏所需的物理效果。

其次则是质量问题。随着时间的推移，游戏中将包含更多的物理效果，读者须在必要时对其予以实现，包括斗篷或旗帜的布料模拟器、漂流盒体的水流模拟器以及独立的粒子引擎。尽管各子引擎工作良好，但读者须花费大量的时间对其进行整合。当身披斗篷的角色位于水面上方时，其布料行为又当如何？若该角色潜入水面下方，且布料依然保持“迎风摆动”这一效果，则该场景与自然常识相悖。

物理引擎以一种可信方式提供了视觉效果的交互行为。在游戏 Half-Life 1 中，可移动的箱体对象构成了谜题的主要内容；而在 Half-Life 2 中，箱体物理行为则被物理引擎所替代，其视觉效果也更为丰富，例如漂流于水面上的箱体碎片、可任意堆放的物体等。

与采用 3 个独立代码段并对其实施有效整合相比，开发物理引擎并对水流、风以及布料实现正确处理则相对“简单”。

1.2.2 物理引擎的缺点

当然，物理引擎并非完美无缺，游戏中依然存在某些场合不适合使用物理引擎。

计算速度可视为一类较为常见的原因，通用物理引擎定位于处理器密集型组件，出于通用性考量，引擎往往不对对象类型作任何假设。当与简单的游戏场景协同工作时，通用性往往意味着处理能力的浪费。当然，这在游戏机或 PC 设备上并不是问题。然而，针对移动电话或 PDA 等小型手持设备，该问题则表现得较为明显。读者可在 PC 设备上编写一款基于纹理引擎的撞球类游戏，相比较而言，若采用特定的物理机制，同一款游戏可能在移动电话设备上运行得更为快捷。

另一个问题则来自数据需求。在最近开发的一款游戏中，其物理内容仅涉及风中飘扬的旗帜对象。对此，可采用商业级物理引擎，并由开发人员计算各旗帜的属性，包括质量和弹性等数据。随后，此类数据须传递至物理引擎中，进而对旗帜对象进行模拟。

针对当前需求，通常不存在适宜的关卡工具可提供有效的模拟数据。对此，我们特意