游戏研发系列

手把手教你架构3D游戏引擎

3D GAME ENGINE

姜雪伟 著

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

游戏研发系列

手把手教你架构 3D 游戏引擎

姜雪伟 著

電子工業出版社
Publishing House of Electronics Industry
北京 • BEIJING

内 容 简 介

本书主要介绍如何利用 3D 固定流水线编写游戏引擎，以及在已编写引擎的基础上开发游戏，全书共分 10 章，主要内容包括游戏引擎简介、数学知识、材质和光照、固定流水线、游戏引擎架构、3D 引擎底层封装、3D 引擎封装、游戏设计实现、地图编辑器、3D 可编程流水线。本书重点介绍 3D 固定流水线编程中涉及的矩阵和向量之间的换算，最后一章介绍了 GPU 编程，也就是常说的 3D 可编程流水线。

本书内容深入浅出，通俗易懂。本着实用性原则，本书并不是面面俱到，但是所讲的知识都可以在实际开发中去运用，可供已经从事 3D 游戏开发的人员或者大中专院校毕业的学生参考，也适合培训机构作为 3D 编程的培训教材。

图书在版编目（CIP）数据

手把手教你架构 3D 游戏引擎 / 姜雪伟著. —北京：电子工业出版社，2017.1
（游戏研发系列）
ISBN 978-7-121-30318-0

Ⅰ. ①手… Ⅱ. ①姜… Ⅲ. ①三维动画软件—游戏程序—程序设计 Ⅳ. ①TP391.414

中国版本图书馆 CIP 数据核字（2016）第 270379 号

策划编辑：张 迪
责任编辑：底 波
印 刷：三河市华成印务有限公司
装 订：三河市华成印务有限公司
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：787×1 092 1/16 印张：20.75 字数：526 千字
版 次：2017 年 1 月第 1 版
印 次：2017 年 1 月第 1 次印刷
印 数：3 000 册 定价：59.00 元

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）88254888，88258888。

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

本书咨询联系方式：（010）88254469，zhangdi@phei.com.cn。

前言

<<<<< PREFACE

当今，由于国家对文化产业的扶持力度加大，极大地促进了游戏这一行业的繁荣发展，越来越多的 IT 从业人员加入游戏开发或 VR/AR 虚拟现实和增强现实开发中。由于游戏行业是一种技术浓缩型的行业，作为一名从业者，要想在游戏行业尤其是 3D 游戏行业内生存、发展，没有过硬的技术是不行的。可是，多年的从业经历让我见到过许许多多的年轻人，他们在公司里只能写一些简单的逻辑，导致理论和技术长时间得不到提高，长期被边缘化，随时都有被辞退的风险。究其原因，是因为他们对于游戏基础理论知识缺乏运用到实战的经验，在公司的游戏开发中无法提供价值。在和他们接触的过程中，我发现一个很普遍的现象：这些能力欠佳的年轻人也有着强烈的欲望去学习专业知识，只是市场上的 3D 引擎书籍讲述得偏理论化，而且理论和技术点比较琐碎、脱节，与他们的游戏研发项目没有必然的联系，有时熬夜看完一本书，脑子里闪现的只是一些零散的知识点，在平时的工作实践中根本用不上，久而久之，也就失去了学习兴趣，得过且过了。作为他们的前辈，我也有相似的经历，耽误了很多时间，我一直想着怎么能让后来者少走弯路，理论联系实践。笔者经过多年的经验积累，终于奉上了这本呕心沥血之作。本书的目的就是要打破常规，将理论知识贯穿到游戏引擎的架构设计和游戏开发中，经典的案例就像万能公式一样，可以使读者边学习边运用到当前的游戏项目中去，实用性超强！初学者认真学完本书后，会更深入地理解引擎架构原理及数学知识在游戏中的运用，对自己能力的提升是非常好的助推器。

本书致力于用最通俗的语言讲述初学者最需要的知识，采用图文并茂的方式讲解各知识点在游戏实战中的运用，让读者更容易理解和掌握。本书涉及游戏开发中的知识点并没有做到面面俱到，而是讲述了游戏开发中最基础的知识，并将其封装到引擎开发中去，希望读者把 3D 游戏开发的基石打牢靠，为以后攀登 3D 引擎高峰打好系统理论基础。

本书的编写不同于以往的侧重理论知识讲解的书籍，对知识点的讲解侧重实用技术和基础知识相结合的方式，把真正游戏开发中常用的技术知识点抽离出来给大家讲解，采用的是一边讲解理论知识，一边与游戏案例结合的方式，真正做到理论与实践相结合与系统化。

本书手把手地教读者如何从无到有架构游戏引擎，以及如何去利用架构好的游戏引擎开发设计游戏，不借助于任何第三方库文件，自己动手封装底层库，更能让读者知道 DirectX 或 OPENGL、OPENGLES 提供的库接口的实现原理，真正做到深入引擎最底层。游戏中使用的模型是开源的 Ogre 模型插件实现的，这样整本书的引擎架构就是一整套完整的引擎设计，以后也可以做跨平台移植。在本书的最后，给大家介绍了 Shader 编程案例的实现，作为 3D 引擎开发者，GPU 编程是必不可少的。目前一些知名的游戏公司，如网易、腾讯、蜗牛等都有公司自己研发的 3D 引擎，这也体现了公司自己的研发能力。随着 VR/AR 的崛起，更期待支持 VR/AR 多种设备的 3D 引擎。毕竟国内真正从事 3D 引擎开发的人相对来说比较少，所以在 3D 引擎开发方面，市场潜力还是非常大的。

目录

CONTENTS

游戏引擎简介

十多年前，笔者刚踏入游戏公司做研发时，公司当时并没有 3D 游戏引擎，公司研发游戏都是在上款已研发好的产品基础上进行修改的，换句话说就是“换皮”。做的时间久了会遇到相同的代码逻辑在不同的游戏项目里面重复出现的现象，行业里面俗称 “重复的造轮子”，开发者在开发产品时通常的做法就是复制粘贴，导致项目出现 bug 的概率非常高，大大降低了研发效率。为了改变这种现状，公司安排专人尝试着把游戏里面常用的代码抽离出来，重新做一个新工程编译成静态库 LIB 或动态库 DLL，这样 3D 引擎的雏形开始形成了。随着工程的代码量不断增加完善，久而久之，3D 游戏引擎就形成了。开发 3D 游戏引擎目的是简化游戏制作的复杂度，缩短游戏开发周期，降低产品制作成本，因为封装好的引擎可开发多种类型的产品。

学习引擎首先要明白什么是引擎？引擎是用来做什么的？在现实生活中，很多开发者对引擎的概念比较模糊，市面上也有很多关于 3D 游戏引擎的书籍，但介绍的都不是很清晰，或者介绍的面太广不利于初级开发者学习。在这里打个形象的比喻介绍 3D 游戏引擎，以汽车的发动机为例，驱动汽车在公路上行驶的是发动机引擎，汽车发动机引擎是驱动汽车行驶的动力源。不论是油箱还是驱动轴等，都要受发动机的控制，换句话说，发动机就相当于一个控制模块，控制所有其他设备运行。发动机的性能也决定了汽车的性能，围绕发动机引擎可以制造出多款不同型号的汽车。知识来源于生活，3D 游戏引擎的原理与汽车发动机引擎的原理是类似的，游戏的逻辑模块也是在游戏引擎的基础上实现的。游戏开发者只要调用引擎提供的接口编写逻辑，引擎的渲染效率直接影响游戏运行效率，引擎的渲染品质直接决定了游戏的渲染品质。当然用同一款引擎可以做出许许多多款游戏，最直观的就是 Unity 引擎，使用 Unity 引擎研发的产品种类非常多，接下来介绍引擎的工作原理。

1.1 引擎的原理

随着 IT 产业的蓬勃发展，越来越多的开源 3D 引擎、商业 3D 引擎都涌现出来了，其中最具代表的是广泛运用于移动端开发的 Unity3D 引擎和虚幻 4 引擎，以及开源图形学 OGRE 引擎。Unity 引擎在市面上比较流行，相信大多数 IT 开发者都使用过，先以 Unity3D 引擎为例介绍引擎的原理，Unity3D 引擎提供了一个功能非常强大的编辑器供用

户开发使用。支撑编辑器运行的是许多已经封装好的 DLL 文件，读者可通过其安装目录查看许多 DLL 动态库文件，这些封装的 DLL 动态库就是 Unity3D 引擎底层封装提供的。游戏开发者在使用其开发游戏写具体逻辑时，也需要利用 C#脚本调用动态库 DLL 封装的函数接口，从而能够执行引擎底层的代码，实现想要的效果。接口的使用可通过查看 Unity 帮助文档获取，它的底层代码的实现对用户来说是不可见的，但这不妨碍使用者开发项目，因为使用者只要知道函数的功能就可以了。

3D 引擎本身也是一种 3D 软件，主要负责处理游戏虚拟世界的渲染，3D 引擎架构设计其实非常复杂，用到的知识点非常多，比如设计模式、多线程编程、算法、GPU 编程等，但是不管多么复杂，其最基本架构思想还是模块化开发，就比如打篮球一样，不论采用哪种战术跑位，最基本的还是挡拆战术。继续模块思想的讨论，以模块化思想设计的 3D 引擎便于扩展，可以有效地减少模块之间的耦合性。开发 3D 游戏引擎时，切记面面俱到，换句话说，3D 游戏引擎的主要功能是渲染，在这方面做得比较好的是开源的 Ogre 图形学引擎及商业引擎虚幻 4，它们核心功能只是负责 3D 渲染，做 3D 引擎该做的事情。对于 Unity3D 引擎，它在 3D 引擎渲染的基础上还增加了网络库等功能，对于引擎来说显得过于臃肿，当然这里不是说 Unity3D 引擎不好，只是其功能太多，对开发者来说并不一定是好事情。对引擎自身来说也不一定是好事情，因为这样引擎的功能失去了重点，所以一提到 Unity3D 引擎，大家的第一印象是其跨平台功能与引擎核心功能渲染不搭边，而对于 UE4 引擎，大家的第一印象是渲染，UE4 在渲染这方面做得非常专一，结果就是 UE4 引擎渲染功能比 Unity3D 引擎更强大。当然 Unity3D 引擎的优点也是非常多的，比如引擎的架构设计使用了组件的理念，使用脚本写逻辑，可以跨多个平台等，而且对于初学者上手非常快，这也是它能快速普及的一个主要原因。

如果要开发一款引擎具体如何做？引擎由哪些主要模块组成？游戏引擎涵盖的模块非常多，它是一个处理游戏所有逻辑的系统。引擎渲染功能是否强大，决定了游戏渲染品质的高低，以及游戏渲染运行的流畅度。下面简单介绍一下通用的 3D 游戏引擎架构，如图 1-1 所示。

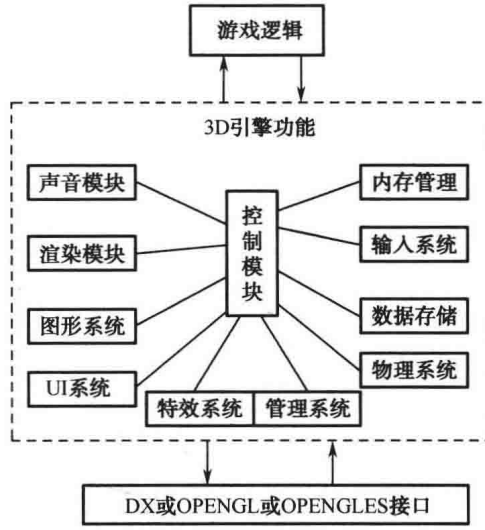

图 1-1　游戏引擎架构

游戏逻辑是最顶端的，也就是所说的游戏产品，一般的游戏开发者都是开发这层的逻

辑。3D 引擎包括了很多功能模块，图 1-1 所列举的只是比较核心的引擎模块，引擎最终会编译成动态库 DLL 或者静态库 LIB，方便游戏逻辑开发者调用其函数接口。当然引擎模块内容对用户是不可见的，就像一个黑匣子一样，它里面封装了游戏开发用到的所有模块，比如图 1-1 所示的声音模块、渲染模块、内存管理、输入系统、物理系统等。游戏引擎的渲染模块会选择使用微软的 DirectX 库和开源跨平台的 OPENGL 和 OPENGLES 图形库，图 1-1 所示的是传统的引擎架构图。市面上很多公司研发的跨平台 3D 引擎，都是基于 OPENGLES 图形库开发的，比如网龙、网易、腾讯、蜗牛、完美等知名 IT 公司都有自己研发的引擎，引擎的研发也代表着一个公司的研发实力。对于程序员来说，掌握了 3D 引擎开发技术，就等于掌握了游戏的核心技能，在国内研发 3D 引擎的人毕竟是少数，所以 3D 游戏引擎研发市场缺口很大。所以作为程序员来说，应该努力去掌握 3D 引擎研发技能，这样在 IT 激烈的竞争中才能立于不败之地。

本书引擎的封装没有使用任何图形库，这么做的目的是为了让读者能够更好地理解和学习引擎底层。本书实现的引擎最基本的运算包括矩阵之间的运算、向量之间的运算及矩阵和向量之间的运算，它们都是手动编写代码封装的，非常有利于初学者学习底层技术，最后会手把手教读者如何使用封装好的 3D 引擎开发游戏逻辑。本书的引擎架构如图 1-2 所示。

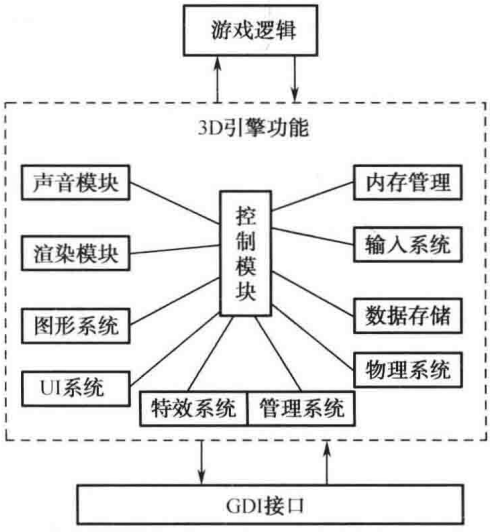

图 1-2　本书引擎架构

与传统引擎最大区别是：二者在于渲染绘制接口不同，前者是基于 OPENGL、OPENGLES 或 DirectX 提供的绘制接口。而本书采用的是 Windows 的 GDI 接口，GDI（图形设备接口）相对来说是针对 Windows 操作系统的，对于没有 OPENGL、OPENGLES 或 Direct3D 使用经验的开发者同样适用。不论是采用哪种接口，其最重要的是要学习其编程思路，编程思路是最重要的，语言只是一个工具而已。

当然对于个人开发者来说，开发一款功能很强大的引擎是不现实的，其涉及的模块太多，一个人无法在短期之内完成。本书讲述的引擎是一款相对来说非常弱小的 3D 引擎，旨在告诉读者如何去架构一款自己的 3D 游戏引擎，以及如何去用代码封装底层算法等。通过本书的学习，能够真正做到对 3D 引擎开发融会贯通，这样在工作开发中对使用市面上其他 3D 引擎开发游戏会有一种柳暗花明又一村的感觉，从而帮开发者解开 3D 游戏引

擎的神秘面纱。说了这么多，可能有读者会问，开发一款引擎需要具备什么样的条件？需要学习哪些技术才可以快速提高编程技能，这些问题网上有很多网友问过，在这里跟大家介绍一下。

1.2 开发 3D 引擎具备的条件

很多游戏开发者认为自己能写逻辑就等于掌握了游戏开发技术，引擎对自己来说无足轻重，这种想法是有问题的。首先你写逻辑是在引擎的基础上写的，其次如果你对引擎一无所知，非常不利于你去深入理解逻辑开发。相反，如果你对引擎有深入的研究，你调用引擎提供的接口时可以很清楚地知道其内部是如何实现的，这样有助于写一些引擎的辅助功能算法，提升自己的编程能力。笔者以前在网龙工作时，项目组就有一位逻辑程序员在做功能时，需要在原有引擎接口的基础上增加一些算法编程以满足游戏玩法，当时他并没有求助引擎组的同事，而是自己写出来了，因为他自己平时就喜欢钻研 3D 游戏引擎，了解引擎接口内部是如何实现的，这本身就证明了他的编程能力。最后将其吸纳到拥有公司最核心技术的引擎项目组，薪酬和职位同时得到了提升。因为在公司里面会写算法，或者会 GPU 编程的人相对来说非常少，如果你掌握了 3D 引擎开发技术，不仅对于逻辑开发更加有利，而且对于你技能提升帮助非常大。

学习 3D 引擎需要经历一个由浅入深的过程，首先要了解最基本的一些 3D 知识，比如固定流水线、可编程流水线、3D 模型结构、骨骼动画等，并且能利用 Dirct3D 或 OPENGL 或 OPENGLES 这些图形库中任何一个，做个简单的 Demo。通过 Demo 了解程序运行的原理后，再尝试封装一些简单算法，一些常用函数，利用这个简单封装做一款小的游戏 Demo。最后再研究一下 GPU 编程，比如 CG、HLSL 语言的语法，在 GPU 编程的基础上再学习一些大型游戏开发中常用的算法，比如 PSSM 实时阴影算法，Bloom 算法等这些与 Shader 编程紧密相关的后处理渲染算法。

目前各大游戏公司，比如网易、腾讯、EA、任天堂等国内外知名的 IT 公司，还有许多新兴的 VR/AR 公司非常紧缺 3D 引擎资深程序或者图形学引擎开发人员，开出了非常诱人的薪水和待遇，从中可以看出，目前游戏市场对于这类人才需求还是非常紧缺的。这正是一个学习 3D 引擎开发的好机会，俗话说“机不可失，时不我待”。

笔者曾经在国内知名 IT 游戏公司参与过 3D 游戏引擎项目组的研发，利用业余时间也开发过 3D 游戏引擎，从底层算法到架构设计都是一个人完成的，当然开发周期也是比较长的。回到正题，3D 引擎这么重要，那作为新手应该如何着手学习？换句话说，关于开发 3D 引擎要具备哪些条件？网络上有许多这方面的解答，笔者经过多年对 3D 游戏引擎的开发研究和实践经验，在此主要总结了以下四点供大家学习参考。

首先，必须掌握主流开发语言 C、C++、Java 或 C#至少一门编程语言，编程离不开数据结构，大学课程里面学的数据结构对于游戏开发非常重要，数据结构在游戏开发中主要用于数据存储及内存管理，开发 3D 引擎常用数据结构有数组、链表、哈希表等，以及常用的一些查找算法：快速排序、二叉树查找、二分查找等，对于常用的数据结构要重点掌握。为引起读者重视，在此再重点强调一下，数据结构对于游戏开发非常重要。

其次，线性代数对于开发同样非常重要，掌握线性代数的目的是在游戏开发中灵活地

运用向量、矩阵、四元数，以及欧拉角这些基本的数学运算解决问题。3D 固定流水线中的坐标变换和可编程流水线的顶点和像素转换都是使用线性代数运算完成的，运算主要涉及游戏开发中的物体移动、旋转、缩放，以及点乘和差乘等。

再次，现在的 3D 游戏引擎渲染都是基于 DirectX 或者 OPENGL、OPENGLES 这些图形库完成的，这就需要开发者能够熟练地使用图形库的接口开发程序，在当前移动端跨平台非常火热的情况下，建议大家学习 OPENGL 图形库。3D 引擎的核心功能就是对游戏的场景渲染和物体的材质渲染，对于材质中有 Alpha 通道的要做特殊处理，Alpha 通道就是说材质有透明的部分。在移动端为了减少透明材质的消耗，会通过 GPU 编程在 Shader 中进行处理，比如把有 Alpha 通道的图片切分成无 Alpha 通道和有 Alpha 通道的图片，通过 Shader 编程将其再合在一起。场景渲染使用的是后处理渲染效果，比如 Bloom、Blur、Ssao、Pssm 等。这些效果实现与 GPU 编程息息相关，要求大家至少掌握 GLSL、HLSL 或 CG 这些基于显卡 GPU 编程语言的一种。

最后是图形学算法，游戏要实现一些逼真的效果离不开算法的支持，比如柔体的模拟、刚体碰撞效果及曲线插值算法等。学习图形学算法，建议大家看一下《算法导论》这本非常经典的书籍，以上说的这些是引擎中比较深层次的知识点。另外，在框架设计方面，需要掌握设计模式里常用的如工厂模式、单件模式、状态模式、MVC 模式等。设计模式的使用不能靠死记硬背，需要将其灵活运用到项目开发中，这样才能逐步深入领会其设计精髓。初学者在使用设计模式的过程中会出现一个误区：千万不要为了使用设计模式而使用设计模式，过度使用会得到适得其反的效果，凡事要把握一个度。以前公司的同事为了使用设计模式，不论设计什么模块都要用设计模式，最后导致在模块扩充时出现了很多问题，举这个例子的目的是告诉读者好东西也不能乱用。笔者从没有在做架构设计时去特意考虑用哪种设计模式，都是顺其自然地使用，真正的用剑高手，可以做到手中无剑，心中有剑的境界。

以上四点是笔者对 3D 游戏引擎开发的心得体会，建议大家循序渐进地学习，优先掌握第一、第二点，其次第三点，最后是图形学算法，由易到难进行学习。希望读者通过本书的学习，能够对引擎开发有更深入的理解，写出更好的 3D 游戏引擎或更好的渲染算法。

游戏开发最核心的技术是算法，在游戏开发中占非常大的比重，也可以说是引擎核心中的核心，以笔者亲身开发项目经历给大家分享一下，希望起到抛砖引玉的效果。近期与高校科研机构合作研发了一款海水渲染 3D 引擎，可以逼真地模拟真实海浪，以及海浪在不同的天气实现不同的效果，模拟船在海中随海浪上下浮动和直升机在海浪上空悬停风力与海浪的作用效果等。在实现的过程中使用了很多算法，比如 Perlin 噪声算法、弗洛伊德算法，以及物理算法等。实现效果如图 1-3 所示，实现的是模拟海浪网格生成及海浪浪花的渲染算法。图 1-4 和图 1-5 实现的是物理引擎里面的海浪波纹的实时生成算法，图 1-6 和图 1-7 实现的是海浪表面的高光渲染算法和反射折射算法，二者都是基于 GPU 编程实现的。另外，笔者参与了多款网络游戏产品的制作，图 1-8 是网络游戏中场景及角色的渲染，所有截图都是成熟产品的实现效果。

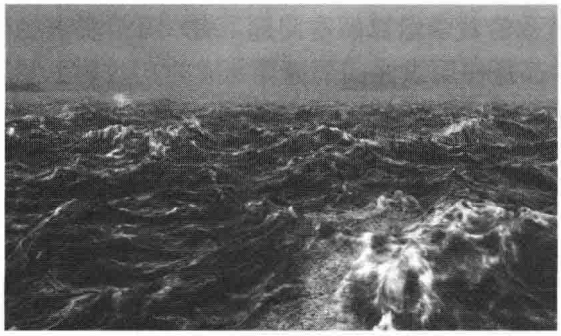

图 1-3　海浪生成图

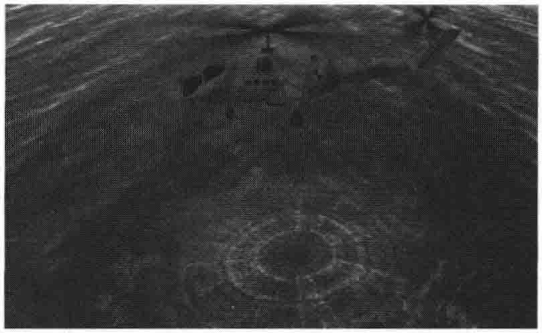

图 1-4　漩涡的生成

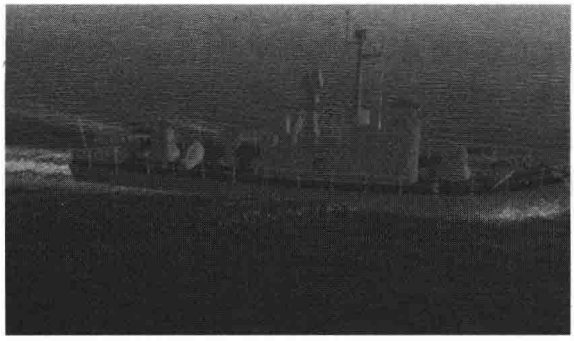

图 1-5　浪花的生成

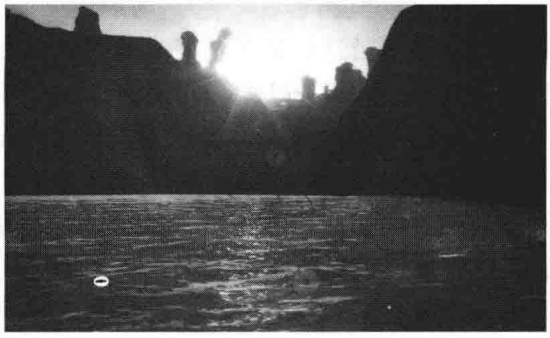

图 1-6　高光效果

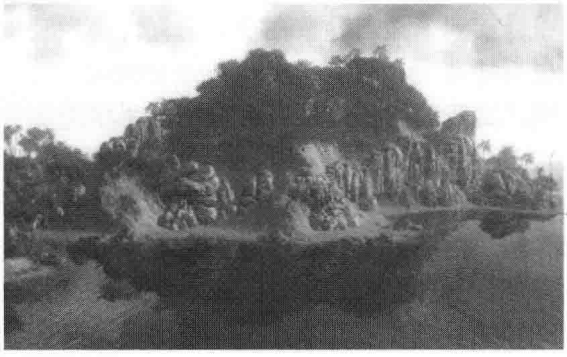

图 1-7　反射效果图

图 1-8　游戏渲染图

总结

技术的掌握不是一朝一夕能够完成的，关键在于坚持，首先要做的是打好基础，在真正掌握了基础知识后，再提升是非常快的。做游戏引擎也是知识积累的过程，虽然现在的 3D 引擎非常多，自己开发游戏引擎已经不现实，但要想真正地深入学习游戏开发核心技术，至少应该知道引擎的底层是如何实现的，换句话说就是其工作原理要搞清楚，这对于开发者来说非常重要。不要再把自己局限于代码操作工简单地写写逻辑，这对自己的技术提升没有任何帮助的。要做到对游戏开发知其然知其所以然，就必须学习 3D 引擎开发技术，特别是对刚踏入游戏行业的从业者，每天坚持不断地学习，如果能持续不断地学习 10 000 小时，你肯定会成为 3D 引擎高手。这就跟锻炼身体一样，其实锻炼身体非常能够

磨炼一个人的心性，锻炼一个人的意志力。自己尝试开发一款小的游戏引擎，即使不能成功，但过程也是非常重要的，你从中可以学习到很多知识，经验的积累对自己的发展帮助非常大，这也是为什么游戏公司招聘，首要的条件是应聘者要有实际开发项目工作经验。程序开发与数学知识是密不可分的，第 2 章介绍 3D 引擎的数学知识。

数学知识

笔者以前在游戏论坛担任过技术版主，以及在 51CTO 教育网、CSDN 教育网、泰课在线等教育网站担任过高级讲师，跟学员或网友经常讨论技术问题，很多人问向量和矩阵在游戏开发中如何解决实际问题，为此笔者单独做了一些关于 3D 数学知识的课程讲座：《3D 游戏开发基础知识》、《3D 数学在 Unity 中的运用》，以及《3D 游戏引擎架构与设计》等视频课程的讲解，旨在帮助刚接触游戏行业及对游戏开发刚入门的初级人员，其实这些知识在大学里面已经学过了，现在参加工作了，要将所学知识运用到实践开发中。不论是 3D 引擎开发，还是游戏逻辑开发，都离不开数学理论知识的支撑，单纯的数学理论知识是非常枯燥的，本章开始将数学知识与游戏开发的实际案例结合起来讲解。将游戏中常用的向量、矩阵、四元数及齐次坐标等一一介绍给大家，希望对大家有所帮助，接下来先从向量开始。

2.1 向量

向量是所有 3D 引擎算法的基础，因此对游戏程序员来说掌握向量的运算非常重要，向量的基本运算包括向量的加法、减法、点乘、叉乘、向量单位化等。向量是由多个分量组成的，根据分量个数的多少，分为 2D 向量、3D 向量、4D 向量等。向量表示的是一条有向线段，是从一个点到另一个点的有向线段，定义的向量总是相对于原点的，而且向量是有方向和长度的。在 2D 游戏和 3D 游戏中定义的点分别是用两个量和三个量表示的，顶点与向量有何区别？单从外形上看二者是一样的，顶点和向量的区别会在后面的齐次坐标里面给大家详细讲解。接下来介绍向量的表达式：2D 向量是指由两个参数组成，表达式为 u=<x,y>，3D 向量是指由三个参数组成，表达式为 u=<x, y, z>，4D 向量表示为 u = <x, y, z, w>，也称为齐次坐标，5D 向量就是五个参数……以 2D 向量为例，在坐标系中如图 2-1 所示。

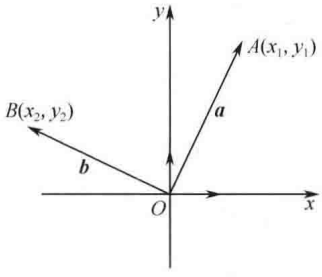

图 2-1　向量的表示

在坐标轴中有两个向量 $\boldsymbol{a}$、$\boldsymbol{b}$，其中向量 $\boldsymbol{a}$ = < x_1, y_1>，向量 $\boldsymbol{b}$ = <x_2, y_2>，二者是有方向的。

2.1.1　向量加法及运用

向量加法指的是两个向量的分量对应相加，简单地说两个向量相加的和就是以两个向量的边作为平行四边形长边的对角线表示，向量加法的表达式：$\boldsymbol{u}+\boldsymbol{v}=<u_x,u_y>+<v_x,v_y>=<u_x+v_x,\ u_y+v_y>$，在坐标系的表示如图 2-2 所示。对角线长的表示两个向量相加，知道了向量相加的公式，接下来看看它在游戏编程中经常用在哪里？或者说在游戏里面是如何运用的？如图 2-3 所示，图中有两个对象：一个是玩家，另一个是怪物，假设玩家具有自动寻找目标攻击技能，玩家是如何做到自动去攻击怪物呢？已知玩家在 3D 场景的坐标 V_0（x_0, y_0, z_0），怪物在 3D 场景中的坐标是 V_1（x_1, y_1, z_1）。为了让玩家能自动寻找到怪物，第一步要确定玩家追击的方向，这个方向如图 2-3 中所绘的从玩家指向怪物的箭头，用公式表示为 dir = (V_0 - V_1).Normalized，计算的结果是玩家追击怪物方向的单位化，方向是没有大小的，故进行单位化操作。取得玩家移动的方向后，接下来要计算玩家移动到靠近怪物的位置去攻击，这要用到向量的加法运算，玩家移动到怪物的位置是根据时间移动的，所以要在游戏运行的每一帧里面去调用，算式表达式：$V_1 = V_0$ + dir * time；这样玩家就会朝着敌人追击过去，当然在玩家移动的过程中可以加入玩家动作。还可以在移动算式前加个判定二者距离远近的表达式，计算玩家移动到距离怪物多远处停止，避免二者重叠在一起。这样利用两个向量的加法运算实现了玩家移动。

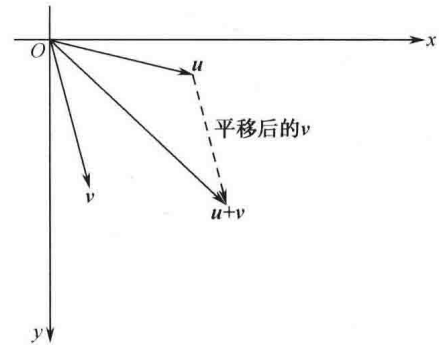

图 2-2　向量加法示意图

图 2-3　图玩家追击怪物

2.1.2 向量减法及运用

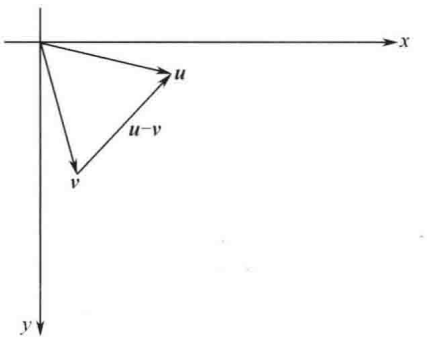

图 2-4 向量减法示意图

向量减法与加法正好相反，对应的是两个向量之间的各个分量相减，向量减法公式表示为 $\boldsymbol{u}-\boldsymbol{v}=<u_x,u_y>-<v_x,v_y>=<u_x-v_x,\ u_y-v_y>$。坐标系如图 2-4 所示，对角线短的表示两个向量相减，向量减法是与向量单位化和平方根紧密相关的。向量减法主要应用在两个方面：其一，减法可用于方向的计算；其二，减法也用于判断两个物体之间的距离。向量减法在游戏中的应用是非常广泛的，游戏中物体向另一个物体靠近时，首先要做的是将这个物体朝向另一个物体移动，比如在射击游戏里面，发射导弹追踪目标，由于目标是在不停的移动中，导弹必须不断修正其朝向目标的方向。它的本质就是向量的减法计算，如图 2-5 所示，物体在下降的过程中，导弹一直追踪着目标最后将其击毁。利用两个向量相减计算距离在游戏中也经常使用，比如怪物追踪玩家，在一定的距离之内它会锲而不舍，超过某个距离值，它就会放弃追踪返回出生地。距离的计算也是通过向量减法再平方根求得，公式会在后面给出，在 AI 算法中也经常使用追踪策略，这样的案例很多，在此就不一一列举了。

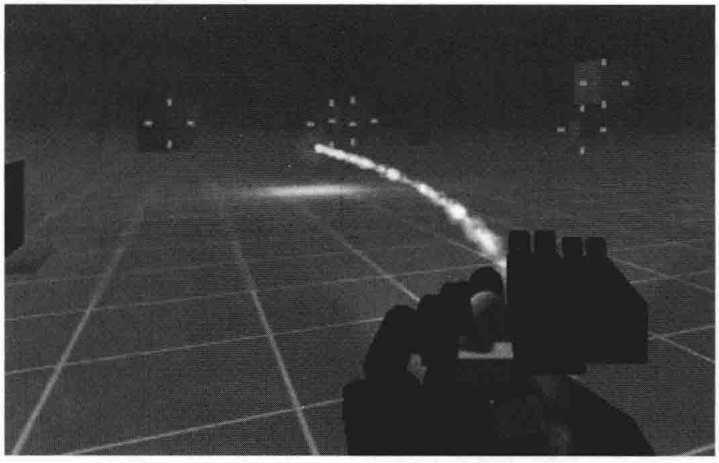

图 2-5 导弹追击锁定目标

2.1.3 向量点积及应用

向量点积主要是用于角度的计算，向量的点积计算公式：$\boldsymbol{u}\cdot\boldsymbol{v}=|\boldsymbol{u}|*|\boldsymbol{v}|*\cos(\text{theta})$，向量 $\boldsymbol{u}$ 的长度乘上 $\boldsymbol{v}$ 的长度再乘上 $\boldsymbol{u}$ 与 $\boldsymbol{v}$ 之间夹角的余弦。它的几何意义是 $\boldsymbol{u}$ 的长度与 $\boldsymbol{v}$ 在 $\boldsymbol{u}$ 上的投影长度乘积，或者是 $\boldsymbol{b}$ 的长度与 $\boldsymbol{a}$ 在 $\boldsymbol{b}$ 上的投影长度乘积，它是一个标量，有正负之分，互相垂直的向量的内积为 0。坐标系如图 2-6 所示。

点乘在游戏中是如何运用的？假设在游戏中玩家看到了某个怪物在自己位置的某个方向，如果它要正视怪物需要旋转一定角度才能正面看到，那这个角度的计算要用到向量的