HZ BOOKS
华章IT

由AMD和清华大学专家联袂推出的异构计算扛鼎之作

本书结合了作者的最新科研成果，对光线追踪和稀疏矩阵算法的应用进行全面剖析

本书立足实战和应用，案例丰富，可操作性强

高性能计算技术丛书

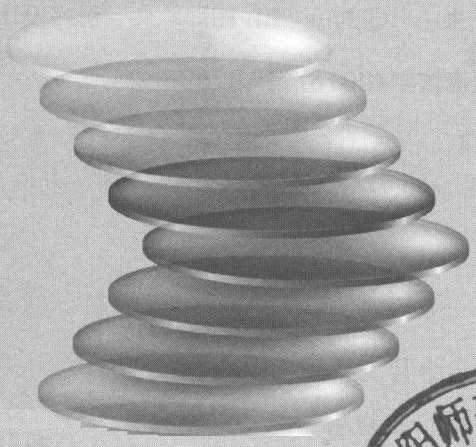
Introduction to OpenCL Programming for Heterogeneous Processor

异构处理器OpenCL编程导论

邓仰东 朱茂华 刘春峰◎编著

机械工业出版社
China Machine Press

高性能计算技术丛书



Introduction to OpenCL Programming for Heterogeneous Processor

异构处理器OpenCL编程导论

邓仰东 朱茂华 刘春峰◎编著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

异构处理器 OpenCL 编程导论 / 邓仰东，朱茂华，刘春峰编著 . —北京：机械工业出版社，2016.7
（高性能计算技术丛书）

ISBN 978-7-111-54330-5

I. 异… II. ①邓… ②朱… ③刘… III. 图形软件 – 程序设计 IV. TP391.41

中国版本图书馆 CIP 数据核字（2016）第 167996 号

异构处理器 OpenCL 编程导论

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：余 洁 责任校对：殷 虹
印 刷：北京诚信伟业印刷有限公司 版 次：2016 年 8 月第 1 版第 1 次印刷
开 本：186mm × 240mm 1/16 印 张：15.5
书 号：ISBN 978-7-111-54330-5 定 价：69.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzit@hzbook.com

Forword 推荐序

拥抱 OpenCL，拥抱异构计算

自 20 世纪 70 年代初以 4004 为最初代表的 CPU 诞生，至今已有 40 多年的时间。期间，处理器的体系结构经历了多次演进，但总的来说，处理器性能的提升主要还是获益于众所周知的摩尔定律。然而，由于不可逾越的物理限制（硅的晶格尺寸在 20℃时大约为 0.5nm，意味着对于现在主流工艺中比较先进的 14nm 工艺，一个晶体管的沟道长度内仅可容纳数十个硅原子），摩尔定律即将失效。再加上内存墙（memory wall）和功耗等各种限制，人们逐渐将未来的处理器性能提升寄希望于体系结构的革新。

对于传统的处理器体系结构，除缓存之外，通常是通过流水、多发射、多核、多线程等并行技术来提升处理器的性能，但目前这些技术的发展也都遇到了难以逾越的障碍。人们意识到，是时候对传统的并行处理技术进行重大变革了。

20 世纪末，异构计算技术获得了半导体公司的注意，以 GPU（Graphics Processing Unit）为代表的各种硬件加速器纷纷出现。很快，GPU 不再局限于 3D 图形处理，而是在各种科学计算领域获得了广泛应用。今天，随着互联网大数据，特别是各种图像、视频、语音等非结构化数据的快速增长，以 CPU+GPU 为典型代表的异构计算平台已在更多的生态和应用中提供着数十倍于 CPU 的性能。

并行 / 异构计算在本质上是关于通信和协作的，这就要求整个计算平台的各个异构模块之间不仅在物理上，而且在逻辑上也紧密地耦合在一起，从而最大限度地提高编程效率和处理性能。各半导体厂商也因此推出了不同的软件解决方案。这些解决方案各具特色，并且在很多应用中获得了成功，然而，在标准和实现的开放性、跨平台的普适性等方面还未能完全满足用户的期望。

2008 年，苹果公司提出了 OpenCL 规范，并在与 AMD、Intel 等公司的合作下逐步完善。

随后，这一规范被提交给 Khronos Group，并由其管理和维护。由于其良好的通用性和开放性，OpenCL 目前获得了各大公司的广泛支持，既包括 ARM、Altera、Qualcomm 这样的芯片公司，也包括 IBM、华为这样的系统公司，还包括 Adobe 等重要的软件公司。同时，越来越多的国内外高校也开始基于 OpenCL 开展教学和科研工作。

本书不仅详细介绍了最新版的 OpenCL 2.0 的编程规范，还提供了丰富的代码实例与练习。此外，邓仰东教授和几位博士结合自己在清华大学的教学与科研经验，开发了与本书配套的课程教案与学生实验指导教辅（可从 www.hzbook.com 上下载）。因此，本书既利于开发者快速开展 OpenCL 实战，也可作为高校高年级本科生或研究生相关课程的教材或参考书。笔者在工作中与本书的几位作者有不少接触。几位专家不仅具有深厚的技术背景与丰富的教学经验，而且还是热心的技术布道者。作者在百忙之中将自己的经验所得进行总结，与大家分享讨论，希望能对国内 OpenCL 异构计算的发展有所帮助，令人不胜钦佩。

希望通过本书的出版，能够帮助更多的同道中人，大家一起来探索异构计算的世界，让更多的行业和用户获益于 OpenCL 的出色特性！

AMD（中国） 时昕博士

Preface 前　言

这是一本关于图形处理器计算的书，也是一本为图形处理器程序员编写的书，但不是一本关于图形程序的书。图形处理器本来是专门处理图形显示的处理器，所以 GPU 总是与图形程序还有图形编程语言 OpenGL 联系在一起，但是本书提到的“计算”和“程序”就是每个程序设计初学者所说的“计算”和“程序”，与图形完全没有任何关系。

本书使用的语言是 OpenCL，是使用 GPU 进行通用计算的编程语言。这里所说的“通用”，其实就是“图形之外”的意思。那么，为什么要使用图形处理器做通用计算呢？第 1 章会深入讨论这个问题，这里我们只需要记住简单答案：①图形处理器可以同时支持大量并行任务，其峰值处理能力超过当前所有其他处理器；②如果应用程序具备足够的并行性，则图形处理器比当前所有其他处理器的计算速度都要快。

这里用到了“并行”的概念。计算机科学中的“并行处理”定义是：由若干处理单元协作完成某一计算任务，并且这些动作的完成过程一般在时间上有所重叠。并行处理其实是自然界最常见的现象。我们所生活的世界本质上就是并行的，大家只要想象高速公路上的车流就可以获得直观的感觉。如果从公路角度看，公路就是处理车流的装置。首先，公路分成多个车道，相当于提供了多个并行处理单元，这是空间的并行；其次，一条车道上也可以同时行驶多辆汽车，只是这些车辆在空间上是顺序分布的，这是时间上的并行，类似于 CPU 内部的流水线。公路上的汽车在统一行为协议（即交通驾驶规则）下以协作和竞争并存的方式分享资源。我们不仅拥有并行的外部世界，人脑也是并行工作的。人类思维和意识的基本单元是神经元，它们能够通过电化学反应对周围的刺激产生响应。人脑中的 100 亿 ~ 150 亿个神经元细胞可联结为异常复杂的网络，思维活动被映射到这个网络并进行高度并行处理。

虽然并行现象无处不在，人类制造的计算机却从顺序处理机制起步。在计算机技术的早期发展阶段，硬件成本昂贵，没有足够资源去采用并行结构。随着集成电路技术的发展，硬

件资源越来越多，从 20 世纪八九十年代开始发展起来的流水线、超标量计算机和超长指令字等技术为 CPU 引入了并行执行硬件，但是在编程上仍然沿用顺序语言，由硬件或编译器寻找指令间的并行性。这种隐含并行模型其实是绝好的硬件“封装”方法，程序无需任何修改就可以在更新一代的处理器上获得更好的性能。2000 年以后，随着指令间并行性的逐渐饱和，计算机体系结构发生了重大转变，普遍的做法是在处理器芯片上部署多个指令执行内核，从而利用仍在不断增长的集成电路资源。换言之，未来不会有更快的处理器内核，但是能够使用的内核数量会越来越多。这一趋势的意义极为深远：首先，我们并不了解兼顾性能、功耗和可编程性的最佳并行处理器体系结构，因此未来的并行处理器必然会出现一个群雄并起的阶段；其次，目前的处理器能够提供任务级、数据级、线程级等多种并行方式，然而，自动并行化技术远未成熟，只能由程序员指定具体的并行方式，以最大限度地发挥并行处理器的能力，所以众多崭新的并行编程语言必然应运而生；最后，主流计算平台要求从算法设计到代码执行的全面并行化，这还是人类文明史上的第一次，我们将不可逆转地走向并行计算之路。

在当前众多的新兴并行处理器和编程模型中，GPU 通用计算具有独到的特色和优势。经历 30 年的发展，GPU 已经拥有成熟的体系结构、完备的工具链和庞大的用户规模。GPU 的最初设计目的是高速图形显示，该任务具有先天的高度并行性，因此 GPU 体系结构始终是为支持海量并行处理而设计的，而人类对视觉效果的无尽追求使得 GPU 发展成为高度优化的硬件平台。随着众多研究人员意识到 GPU 在通用计算方面的潜力，NVIDIA 首先推出了图形处理器通用编程技术，从而在高性能计算领域占据了一定的市场份额。随着众多 GPU 制造商的纷纷跟进，大家开始意识到兼容性问题。GPU 市场仍然处于“战国”时代，桌面 GPU 市场由 Intel、NVIDIA 和 AMD 三家公司分享，后两者主导高端 GPU。而移动 GPU 的竞争者更多，Imagination、ARM、NVIDIA、Qualcomm、Samsung 等公司也都有自己的产品。为了借鉴 OpenGL 的成功经验，使得基于不同硬件体系结构的 GPU 能够在源代码级实现兼容，各大 GPU 及其下游产品制造商合作制定了 OpenCL 图形处理器通用计算语言标准，并由 Khronos 集团发布和维护。作为 OpenGL 的姊妹语言，OpenCL 也是跨平台兼容的标准化语言。实际上，任何处理器只要提供相应编译器、驱动和动态运行时（runtime）环境，就可以执行 OpenCL 程序，著名的可编程芯片制造商 Altera 甚至可以将 OpenCL 编译到硬件上执行。当然，OpenCL 的语法和语义是针对图形处理器硬件设计的，只有具有相应结构的硬件才能高效执行 OpenCL 代码。

OpenCL 和 CUDA 也经常被称为异构编程语言，这是因为相应代码总是使用 CPU 和 GPU 这两种在硬件体系结构和代码执行方式上有很大区别的计算资源。一方面，目前的

GPU 还不能运行操作系统，所以 GPU 通用计算程序总是在 CPU 上启动；另一方面，在 GPU 通用计算程序中也允许使用 C/C++ 语言编写在 CPU 上执行的代码。从目前的技术看，CPU 和 GPU 分别适应不同的计算结构，双方在应用角度上的互补关系远大于竞争关系，这种定位在可预见的未来并不会发生改变，因此异构计算代表了未来的方向。大部分异构平台一般采用 CPU 处理器芯片 +GPU 处理器芯片的形式，而对于在软件层面整合两种计算资源，唯有 AMD 公司迈出了关键一步，通过定义异构系统体系结构（Heterogeneous System Architecture，HSA）实现了 CPU 和 GPU 的紧密耦合，使两种处理器集成在同一芯片上并共享片外存储器。

介绍到这里，本书的舞台就已经搭建好了。本书共有 14 章，第 1 ~ 3 章从图形处理器的历史开始，依次讨论 GPU 硬件体系结构和 HSA，以及具有代表性的 HSA 处理器。第 4 ~ 7 章介绍 OpenCL 编程，包括基本语法、并行组织方式、事件和任务队列，以及 OpenCL 2.0 引入的高级特征。第 8 ~ 13 章用一组实例由浅入深地讲述并行程序设计和编程技巧。第 14 章介绍了一个比较复杂的应用实例——光线追踪图形渲染，总结了 OpenCL 程序设计技术。

设计简洁、高效的代码从而最大限度地发挥并行硬件的能力，是每一位严肃对待工作的程序员发自心底的愿望。谨以此书献给严肃的程序员们。

目　录 *Contents*

推荐序

前　言

第 1 章　GPU 计算的发展历程 …… 1

1.1　计算机图形学的发展 …… 2

1.2　图形流水线 …… 6

1.3　GPU 的发展过程 …… 8

1.4　GPU 通用计算的发展历程 …… 15

参考文献 …… 18

第 2 章　现代图形处理器的体系结构 … 20

2.1　计算机体系结构基础 …… 21

2.2　GPU 的设计思想 …… 23

2.3　NVIDIA 图形处理器 …… 25

2.3.1　总体体系结构 …… 26

2.3.2　流多处理器 …… 29

2.3.3　流多处理器的扩展 …… 31

2.3.4　存储器 …… 34

2.4　AMD Graphics Core Next 图形处理器体系结构 …… 37

2.4.1　GCN 计算单元 …… 38

2.4.2　GCN 缓存 …… 40

2.4.3　GCN GPU …… 41

2.5　Imagination PowerVR 移动图形处理器体系结构 …… 42

参考文献 …… 44

第 3 章　异构系统体系结构和融合处理器 …… 45

3.1　HSA 基本概念 …… 47

3.2　异构系统体系结构存储器模型 …… 50

3.2.1　虚拟存储器的地址空间 …… 51

3.2.2　缓存一致性 …… 52

3.2.3　内存一致性 …… 53

3.3　异构任务队列式调度机制 …… 54

3.4　任务抢占和内容切换 …… 57

3.5　HSA 中间语言 …… 57

3.6　AMD 的 HSA 硬件 …… 60

习题 …… 62

参考文献 …… 63

第 4 章　OpenCL 基本概念 …… 64

4.1　OpenCL 程序的工作流程 …… 64

4.2　OpenCL 平台与设备 …… 65

4.3　上下文、命令队列、kernel 函数 …… 70

4.4　存储对象 …… 72

4.5　Hello World 例程 …… 73

习题 …… 82

参考文献 …… 82

第 5 章　OpenCL 并行编程基础 …… 83

5.1　并行线程组织 …… 83
5.2　OpenCL 存储器模型 …… 86
5.3　数据类型 …… 88
5.4　运算符 …… 92
5.5　函数 …… 93
5.6　矢量相加例程 …… 96
5.7　矩阵相乘的 OpenCL 例程 …… 98
5.7.1　矩阵相乘 OpenCL 代码 …… 98
5.7.2　矩阵相乘分块计算的 OpenCL 代码 …… 100
5.7.3　提高计算密度的分块矩阵相乘 …… 102
5.7.4　使用矢量计算的分块矩阵相乘 …… 105
习题 …… 107
参考文献 …… 107

第 6 章　OpenCL 事件和队列 …… 108

6.1　命令、命令队列和事件 …… 109
6.2　事件的定义和基本用法 …… 112
6.3　事件对象与用户事件 …… 115
6.4　双设备队列管理例程 …… 115
6.5　使用事件进行性能剖析例程 …… 116
习题 …… 118
参考文献 …… 118

第 7 章　OpenCL 2.0 高级特征 …… 119

7.1　共享虚拟存储器 …… 119
7.1.1　粗粒度缓冲区共享虚拟存储器 …… 120
7.1.2　细粒度缓冲区共享虚拟存储器 …… 122
7.1.3　细粒度系统级共享虚拟存储器 …… 126
7.2　管道 …… 127
7.3　嵌套并行 …… 129
7.4　工作组函数 …… 130
7.5　通用地址空间 …… 131
习题 …… 132
参考文献 …… 133

第 8 章　并行程序设计方法 …… 134

8.1　并行程序设计的复杂性 …… 134
8.2　程序性能剖析 …… 135
8.2.1　程序执行时间 …… 135
8.2.2　程序性能剖析的概念 …… 136
8.2.3　使用 gprof 剖析程序性能 …… 137
8.2.4　解读性能剖析结果 …… 139
8.3　寻找并行性 …… 140
8.3.1　可行性分析 …… 140
8.3.2　数据依赖性 …… 141
8.4　并行化实例 …… 142
8.4.1　顺序逻辑仿真算法 …… 144
8.4.2　同步并行逻辑仿真 …… 145
8.4.3　保守型并行逻辑仿真 …… 145
8.5　并行化设计方法学 …… 146
8.6　降低并行程序的开发难度 …… 148
习题 …… 149
参考文献和深入阅读 …… 150

第 9 章　*N* 体问题 …… 152

9.1　串行算法 …… 153
9.2　GPU 端 OpenCL 程序 …… 155
9.3　CPU 端 OpenCL 程序 …… 157
9.4　双 GPU 的 OpenCL 程序 …… 160
习题 …… 164
参考文献 …… 164

第 10 章　归约问题 …… 165

10.1　直观并行归约算法 …… 166
10.2　减少闲置线程 …… 168
10.3　改进局部内存访问 …… 169
10.4　避免内存访问冲突 …… 171
10.5　减少同步操作 …… 172
10.6　优化顺序和并行工作比例 …… 174
习题 …… 177
参考文献 …… 177

第 11 章　快速傅里叶变换 …… 178

11.1　傅里叶变换原理 …… 179
11.2　快速傅里叶变换算法 …… 180
11.3　简化基 –2 FFT 算法内核 …… 183
11.4　通用基 –2 FFT 算法内核 …… 184
11.4.1　Twiddle 因子 …… 184
11.4.2　数据访问模式 …… 185
11.4.3　64 点 FFT …… 187
习题 …… 190
参考文献 …… 190

第 12 章　稀疏矩阵——向量积 …… 191

12.1　稀疏矩阵数据格式 …… 192
12.2　对角稀疏矩阵 …… 195
12.3　COO 格式稀疏矩阵 …… 201
12.4　CSR 格式稀疏矩阵 …… 203
12.5　ELL 格式稀疏矩阵 …… 206
习题 …… 207
参考文献 …… 207

第 13 章　直方图 …… 208

13.1　直方图的计算 …… 209
13.2　简单的并行直方图计算 …… 210
13.3　数据值范围较小情形下的直方图算法 …… 212
13.4　数据值范围较大情形下的加权直方图算法 …… 214
习题 …… 216
参考文献 …… 216

第 14 章　光线追踪算法的 OpenCL 实现 …… 217

14.1　光线追踪算法综述 …… 219
14.2　光线追踪流水线解析 …… 224
14.3　OpenCL 光线追踪程序 …… 228
14.3.1　代码框架 …… 228
14.3.2　主机端代码及详解 …… 229
14.3.3　设备端代码及详解 …… 232
参考文献 …… 237

第 1 章 Chapter 1

GPU 计算的发展历程

人类似乎对计算有着天然的追求，几乎所有的自然语言[⊖]都至少有“1”和“2”的数字概念。文明史其实也是一部计算的历史，人类总是在追求更强的计算能力。公元前 3000 年古巴比伦就发明了算盘，1822 年查尔斯·巴贝奇设计出第一台机械计算机。1936 年图灵提出了通用计算机模型，1943 ~ 1945 年第一台电子计算机在宾夕法尼亚大学诞生，标志着计算机时代的来临。20 世纪 60 年代集成电路技术的发展为计算硬件注入了空前的活力。1971 年第一款通用微处理器芯片 4004 的出现标志着人类进入个人计算机时代。20 世纪八九十年代是计算机体系结构的黄金时期，随着流水线和超标量等指令间技术的不断深入，CPU 性能不断提升，而编程模型不需要任何改变就可以获得更高性能。2000 年以后，一方面指令间并行技术发展逐渐遭遇瓶颈，单核 CPU 的性能趋于饱和。另一方面，半导体工业仍然能够提供更高的集成度。针对这种形式，直观的解决方法是在单个集成电路上部署多个处理器，所以多核（几个到几十个内核）处理器甚至众核（几百个到上千个内核）处理器应运而生。与单核处理器主要利用指令间并行性不同，多核和众核处理器依赖线程级与数据级并行性来提高处理能力，换言之，需要程序员设计并行算法并且在代码中指定并行的方式。主流计算工具开始实现全面并行化，这是人类计算史上的第一次，因此具有深远的意义。

在过去短短几年中，并行处理器无处不在，即使是手机的应用处理器，也会装备多个内核。然而，并行算法设计、程序开发和编译技术的发展却滞后很多。这一方面是由于人类思维方式组织在顺序模型中（虽然执行基础是高度并行的），直接进行并行思维非常困难；另一方面是由于现有技术储备不足，虽然并行计算已经过多年发展，但主要用在高性能计算领域，其问题、方法甚至程序员群体都与当前有着巨大的区别。正如《量化计算机体系结构》

⊖ 迄今为止，只有南美洲亚马逊地区的极少数人使用的 Pirahã 语言被某些学者认为完全没有数字概念。

的两位作者——计算机体系结构大师和诺依曼奖获得者 John L. Hennessy 和 David Patterson 指出的那样，正如 50 年前的编程危机，易于编程的高性能并行处理器体系结构和并行编程模型是计算机科学与技术史上面临的最大挑战之一[1]。

严峻的挑战也会带来崭新的机会。50 年前的编程危机催生出高级编程语言，而现在也能看到大量新型计算机体系结构和编程模型在不断涌现。在这本书里，我们关注图形处理器（GPU）。当前的 GPU 一般以单片集成电路的形式出现，同时也是单片处理能力最高的处理器。顾名思义，图形处理器是为图形应用这一类特殊计算而生的，那么，GPU 是怎样成为通用计算处理器的呢？本章首先扼要回顾计算机图形学的发展，从中整理出 GPU 的概念；接下来整理 GPU 的发展脉络；最后介绍 GPU 通用计算的发展历程。

1.1 计算机图形学的发展

图形处理器是计算机图形学（Computer Graphics，CG）高度发展的结果。计算机图形学研究用计算机生成适合人眼观看的二维或三维图像表征。由于通过眼睛采集图像是人类感知信息最重要的方式，所以图形显示是计算机信息处理过程中人机交互的核心手段。自 1960 年“计算机图形学”术语的出现，该学科已经历经 50 年的发展。以计算机游戏和电影制作为代表的娱乐业应用、以工程 CAD 为代表的可视化应用，以及各类图形界面应用取得了长足进步，已经并且继续深刻地改变着人类生活、工作和娱乐的方式。由于人眼对图形细节有着极高的分辨能力，随着计算机技术的高速发展和人类对用户体验的不断追求，计算机图形学至今仍然是计算机科学研究的热点。

表 1-1 列出了计算机图形学发展历程中的主要里程碑。令人惊奇的是，早在 1944 年，MIT 开展的“ Whirlwind”计算机项目中就已经出现了现代计算机图形学的关键概念。该计算机用于冷战时期的防空作战指挥，对快速、有效的图形交互具有强大需求，开始使用阴极射线显示器、光笔等现代图形交互手段。在此基础上，MIT 又设计了 TX-0 和 TX-2 计算机，当时还是博士研究生的计算机图形学大师 Ivan Sutherland 为 TX-2 设计了 Sketchpad 图形交互工具。如图 1-1 所示，Sketchpad 允许使用光笔在显示器上绘制和操作图形。Sketchpad 包含现代计算机图形学的几乎所有核心概念。因此，Ivan Sutherland 被认为是计算机图形学的祖父级人物。

表 1-1 计算机图形学发展史的重要里程碑[2]

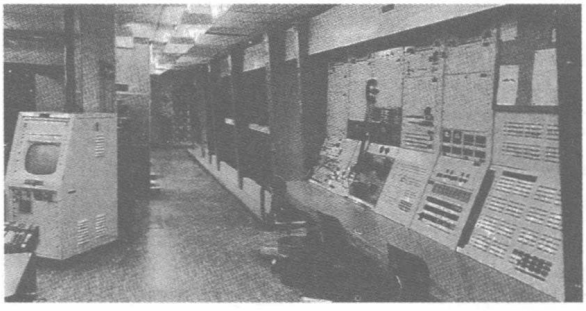
时间	计算机图形学发展的重要成就
1944 年	MIT 开展的“Whirlwind”项目首次系统化研究实时计算机图形显示
1960 年	“计算机图形学”术语出现
1962 年	第一个多玩家计算机游戏 Spacewar
1963 年	Ivan Sutherland 发明 Sketchpad
1965 年	首次计算机艺术展在斯图加特和纽约举办

（续）

时间	计算机图形学发展的重要成就
1967 年	• Coons Patch 复杂曲面建模算法提出 • NASA 设计了第一个全彩色实时飞行模拟器
1968 年	第一家计算机图形技术公司 Evans & Sutherland 成立
1968 年	首届计算机图形学会议（SIGGRAPH）召开
1970 年	随机访问存储器和光栅化显示器诞生
1973 年	多维可视化，Phong 渲染算法提出
1974 年	Z-Buffer 算法，纹理映射算法提出
1980 年	• 首届欧洲图形学（Eurographics）会议召开 • 光线追踪算法提出 • 迪斯尼制作第一部大量使用计算机合成图像（超过 20 分钟）的电影《TRON》
1982 年	• Silicon Graphics 公司设计硬件图形引擎（Geometry Engine） • 工业光魔（Industrial Light and Magic）公司制作第一部计算机动画电影《Startrek》 • 3D 计算机辅助设计软件公司 Autodesk 成立
1984 年	• 第一家专攻计算机动画电影的公司 Pixar 成立 • Apple 公司发布第一台拥有图形界面的个人计算机
1985 年	• 2D 图形标准 GKS（graphical kernel system）发布 • ATI 公司（后来被 AMD 公司收购）成立
1987 年	• 美国自然科学基金开始专门资助图形学研究 • Marching Cubes 可视化算法提出
1988 年	• 2D 图形标准 PHIGS 发布 • 3D 图形标准 ISO GKS-3D 发布
1989 年	• 第一部真实 3D 角色电影《Abyss》 • 3D 渲染软件 Renderman 公开发布
1991 年	数据可视化工具 OpenDX 推出
1992 年	Silicon Graphics 公司推出 OpenGL 标准、图形流水线标准化
1993 年	NVIDIA 公司成立
1994 年	Sony 公司的 Playstation 游戏机和任天堂的 N64 游戏机问世
1995 年	• Microsoft 公司定义 Direct3D 图形接口 • 3D 第一人称设计游戏 Quake 发布
1996 年	3DFX Voodoo 图形加速器获得巨大成功
1997 年	Titanic 获得奥斯卡特效奖，其中大量使用了人群仿真和流体力学仿真
2000 年	Sony 公司的 Playstation 2 问世
2001 年	Imagination 公司设计 PowerVR MBX GPU 内核，并在移动 GPU 市场取得成功
2002 年	• Microsoft Xbox 游戏机出现 • 奥斯卡首次颁发最佳动画形象奖并由 Shrek 获得
2009 年	第一部大量使用面部表情捕捉动画技术的电影《Avatar》

1960 ~ 1980 年是计算机图形学的“少年期”，图形渲染的基本理论在这 20 年得到充分发展，如曲面建模、着色渲染、Z-buffer 可见度判断、光线追踪和纹理映射等方法都是在这段时期形成的。同时，计算机图形学的主要应用也涌现出来：最早的带有图形特征的计算机

游戏 Spacewar 在 1962 年出现[⊖]，1967 年 NASA 设计了第一个全彩色实时飞行模拟器，1980 年迪斯尼制作的电影《TRON》包含了超过 20 分钟的计算机合成动画。同期的硬件重大发明是随机访问存储器（Random Access Memory，RAM）和光栅化显示器，这些硬件技术的进步使得大众消费级别的高性能图形显示成为可能。在这段时间内，计算机图形学学术圈也逐渐形成：1968 年首届计算机图形学会议（SIGGRAPH）召开，至今其仍是该领域最为重要的学术会议和相关技术展示平台，与 1980 年出现的欧洲图形学会议以及此后出现的 SIGGRAPH Asia 并称为计算机图形学的三大会议。



a）TX-2 计算机

b）Ivan Sutherland 和 Sketchpad

图 1-1 Ivan Sutherland 与 Sketchpad 图形交互工具

20 世纪八九十年代是计算机图形学的“青年时代”。首先，计算机图形工业逐渐形成。著名的图形特效制作公司工业光魔、动画电影制片厂 Pixar 成立，图形 CAD 软件公司 AutoCAD、图形硬件制造商如 SGI、显卡制造商纷纷出现。Pixar 设计了著名的专业渲染软件 Renderman，几乎成为电影级渲染的标准；美国电影工业开始大量使用计算机图形学技术，其标志性电影《Titanic》大量使用人群仿真和流体力学仿真方法制作电影特效，并且于 1997 年获得奥斯卡特效奖。继 1984 年 Apple 公司的个人计算机出现后，具有图形化人机界面的个人计算机也得到了普及，Sony 公司和任天堂也分别推出了著名的游戏机平台 Playstation 和 N64。图形应用对硬件性能要求极高，早期的图形处理硬件在这样的大背景下应运而生，SGI 首先设计了著名的 Geometry Engine。此后一系列专注于图形硬件的公司如 ATI、S3、3DFX、NVIDIA 就像雨后春笋般出现了，纷纷推出专用图形处理芯片。由于此时图形硬件尚不具备编程能力，因此一般被称为图形加速器，以显卡的形式体现功能。1991 年 S3 公司首先设计出第一块 2D 图形加速芯片，1995 年 NVIDIA 公司设计首个 3D 图形加速芯片 NV1，1996 年 3DFX Voodoo 图形加速器在市场上取得巨大成功。从这时开始，高端图形显示硬件形成了以显卡作为功能载体的基本模式，而显卡功能以图形加速芯片为核心，辅之以图形存储器。此外，IBM 公司设计了专用数据可视化工具 OpenDX，标志着一个新的图形学应用出现。随着工业化程度的深入，工业界和学术界提出了图形处理的标准化需求。早期的标准包括 2D 图形标准 GKS、PHIGS 和 3D 图形标准 ISO GKS-3D，而随后 SGI 公司提出的 OpenGL 则

⊖ 有趣的是，早期的 Spacewar 是一个多用户游戏。

成为事实上的工业标准，Microsoft 公司也推出了 Direct3D，形成了图形标准的另一大阵营。OpenGL 和 Direct3D 的出现标志着计算机图形操作的标准化，使得图形应用跨平台兼容成为可能，极大加速了图形软硬件技术的发展，成为今天所有图形软件和硬件的基础。

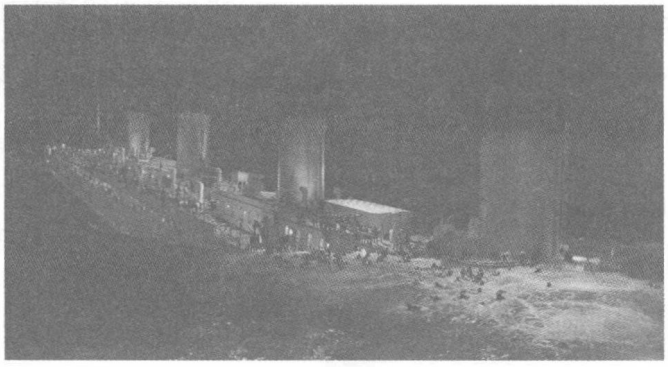


图 1-2 《Titanic》电影中的海水特效

2000 年后，计算机图形学进入加速发展阶段。在硬件方面，自 2001 年 NVIDIA 公司推出第一个可编程渲染图形处理器 GeForce 3 以来，图形处理器的概念正式出现。此后，计算机图形硬件市场形成了 NVIDIA 公司、ATI 公司（后来被 AMD 公司收购）和 Intel 公司三雄并立的局面。其中 Intel 公司专注于集成显卡，在中低端市场具有显著优势，而 NVIDIA 公司和 ATI 公司则在高端独立显卡市场展开白热化竞争，双方的竞争一度使得 GPU 芯片集成度的上升速度超过了摩尔定律。与此同时，移动平台同样需要 GPU。类似于早期计算机的 GPU 市场，移动 GPU 的竞争更加激烈，目前主要的移动 GPU 供应商包括 Imagination、QualComm、NVIDIA、ARM 等多家公司，其中 Imagination 和 QualComm 的市场份额较大，但并不具备绝对优势。另一方面，随着硬件技术的进步，使得全局光照和物理仿真等技术能够应用在各种图形中。好莱坞大片越来越依靠计算机制作的特效，动画人物的衣物、头发、皮肤、表情等普遍通过物理仿真技术生成，而光线追踪等物理真实渲染技术也被大量使用。2002 年开始，奥斯卡新设的一个奖项——最佳动画形象奖由 Shrek 获得。2009 年上映的《Avatar》是面部表情捕捉、光线追踪、物理仿真等技术的集大成体现。图 1-3 是《Avatar》中使用光线追踪技术制作的一个镜头，其中远处的植被和动物具有极其真实的光照效果。

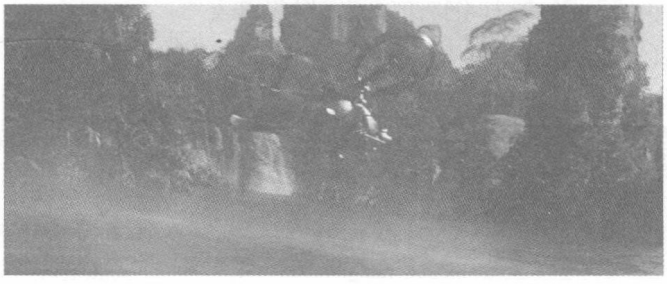


图 1-3 《Avatar》中用光线追踪技术制作的镜头

1.2 图形流水线

OpenGL 和 Direct3D 都定义了图形流水线（graphics pipeline）概念，包括一系列计算机图形显示过程中必须执行或可选的图形操作。目前，所有 GPU 的操作都是围绕图形流水线设计的。因此，接下来将分析图形流水线的主要操作，从中了解图形应用对 GPU 硬件体系结构提出了怎样的要求。当前的图形流水线非常复杂，下面以图 1-4 中只包含基本操作的简化图形流水线来说明。

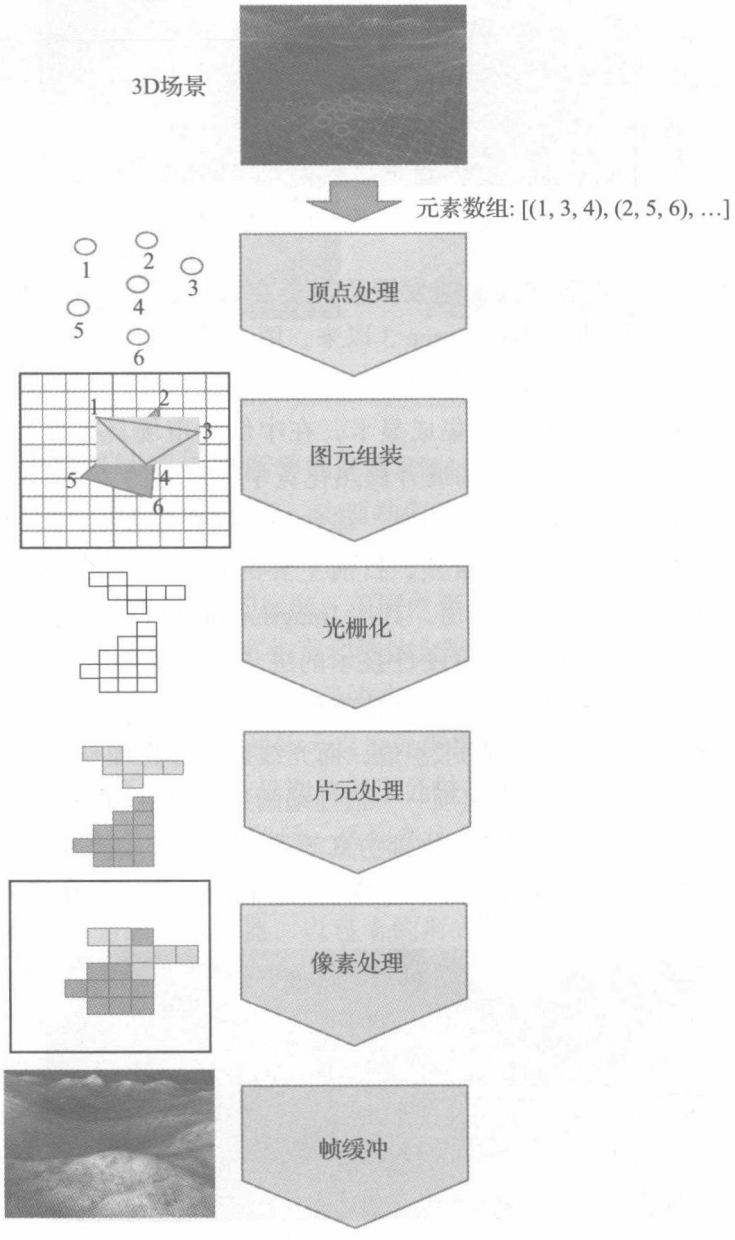


图 1-4 简化的图形流水线