

# GPU Gems 2

PEARSON  
Addison  
Wesley

Programming Techniques for High-Performance Graphics and General-Purpose Computation

# GPU精粹 2

——高性能图形芯片和通用计算编程技巧



(美) Matt Pharr 编著  
龚敏敏 翻译



NVIDIA



清华大学出版社

# GPU精粹 2

——高性能图形芯片和通用计算编程技巧

(美) Matt Pharr 编著  
龚敏敏 翻译

清华大学出版社

北 京

Authorized translation from the English language edition, entitled GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation, 0-321-29431-9 by Matt Pharr, published by Pearson Education, Inc, publishing as Addison-Wesley, Copyright © 2006.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

CHINESE SIMPLIFIED language edition published by PEARSON EDUCATION ASIA LTD., and TSINGHUA UNIVERSITY PRESS Copyright © 2007.

北京市版权局著作权合同登记号 图字: 01-2005-6215

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话: 010-62782989 13501256678 13801310933

#### 图书在版编目(CIP)数据

GPU 精粹 2——高性能图形芯片和通用计算编程技巧/(美) 法尔(Pharr, M.) 编著; 龚敏敏 翻译.

—北京: 清华大学出版社, 2007.5

书名原文: GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation

ISBN 978-7-302-13943-0

I. G… II. ①法… ②龚… III. 计算机图形学 IV. TP391.41

中国版本图书馆 CIP 数据核字(2006)第 120371 号

责任编辑: 王 军 徐燕萍

装帧设计: 孔祥丰

责任校对: 成凤进

责任印制: 王秀菊

出版发行: 清华大学出版社

<http://www.tup.com.cn>

[c-service@tup.tsinghua.edu.cn](mailto:c-service@tup.tsinghua.edu.cn)

社 总 机: 010-62770175

投稿咨询: 010-62772015

地 址: 北京清华大学学研大厦 A 座

邮 编: 100084

邮购热线: 010-62786544

客户服务: 010-62776969

印 刷 者: 北京鑫丰华彩印有限公司

装 订 者: 北京鑫海金澳胶印有限公司

经 销: 全国新华书店

开 本: 185×260 印 张: 36.5 字 数: 888 千字

附光盘 1 张

版 次: 2007 年 5 月第 1 版

印 次: 2007 年 5 月第 1 次印刷

定 价: 128.00 元

---

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题,请与清华大学出版社出版部联系调换。联系电话: 010-62770177 转 3103 产品编号: 019896-01

# 序

---

在专门用于 PC 图形绘制的硬件出现之前，工业上的第一个 3D 立体游戏用了基于 CPU 的软件渲染。那时，受 John Carmack 在 Doom 和 Quake 上的先锋程序启发，我写了第一个 Unreal Engine。尽管 CPU 很慢，分辨率很低，但 20 世纪 90 年代中期成了图形和游戏时间上的分水岭。几乎每个月都出现新的视觉效果，其中以 Quake 的光照贴图和阴影与 Unreal 的彩色光和体雾这样的里程碑最为显著。随着固定功能 3D 加速器的出现，那个时代渐渐逝去了。由于缺乏驱动革新和变异的可编程性，3D 游戏变得模糊了。

今天，3D 图形进入了新的复兴时代，由完全可编程的 GPU(Graphic Processing Units, 图形处理单元)驱动，拥有了超过十年前上千倍的图形处理能力。现代的高级编程语言联合难以置信的并行计算能力，使今天的 GPU 引燃了革新和创造的强大爆炸，可以轻易地完成实时的模糊阴影、精确的照明模型和真实的材质影响。但是可编程性最重要的好处是，只要可以找到一个算法来表达您的想法，就能用 GPU 完成一切工作。本书示范了很多这样由想法变成的算法。

下面看一下当今的图形程序员所拥有的资源。首先，您可以访问一个执行可编程着色算法的 GPU，每秒可以进行几百亿次浮点运算，它是您的工具。如果您可以把问题移入像素和顶点的范畴内，就可以利用 GPU 的强大功能。其次，您有一个 CPU，是系统的通用计算引擎。CPU 发送指令给图形处理单元、管理资源，并与外部世界交互。最后，您有美工的内容——纹理贴图、网格和其他多媒体数据，GPU 可以在渲染过程中组合、过滤和程序化地修改它们。

本书中的精华在于把这些资源以新的方式用于渲染真实感场景、处理图像和产生特效。这么做，许多原来的图形规则可能被打破了。GPU 快而灵活，使用户可能多次渲染一个给定的对象，把场景分解成它的组件——灯光、阴影、反射和后期处理效果等。也可以用 GPU 完成非图形任务，如碰撞检测、物理和数值计算；在纹理贴图中可以编码着色程序用的任意数据，如向量、位置或查找表。虽然具有真实感的效果已经在 GPU 上实现，但它不是唯一选择：还有非真实渲染技术，如卡通着色，夸张的运动模糊和光扩散(bloom)，以及在好莱坞产品中时常见到的其他效果。

在我写 Unreal 最初的软件渲染器 7 年之后，我的公司开始开发一种新的游戏引擎——Unreal Engine 3。它是针对今天的现代 GPU 能力设计的。它拥有难以置信的功能！以前需要建立 300 个多边形的带有静态光和纹理贴图的场景，现在则可以把带有真实感材质特效的动态逐像素光照和阴影组合进有上百万个多边形的场景。我们已经体会到了程序员和美工可以获得的能力和灵活性的爆炸性增长。但是，当很多东西已经在图形的发展中改变时，一些事实仍然保持不变：图形需要工程学、艺术性和发明的独特组合，任何一个都是无可

---

取代的；随着硬件性能指数般地增加，创新正以难以置信的速度发展，并且图形编程充满极大乐趣！

在本书中，您将学到很多新知识，并提高洞察力，还会了解很多简单整洁的想法，它们都可以轻易地在今天的图形硬件上应用。但是这里提供的技术只是您冒险旅途的一个出发点——真正的乐趣和机遇在于，发现新的方法来定制和组合这些精华，并且发明出新东西。

Tim Sweeney

Epic Games 的创办人和技术指导



截图来自 Unreal Engine 3 的技术演示，<http://www.unrealtechnology.com>

# 前 言

---

“GPU 精粹”的第 1 卷构思于 2003 年春季，即在第一代完全可编程的 GPU 出现不久之后。结果，书在一年之内就发行了，而且很快变成了一本畅销书，书中提供了一些最佳想法，让您可以尽可能地利用最新的可编程图形硬件。

GPU 编程是一个变化快速的领域，而且写续篇的时机已经成熟。自从出现可编程图形处理器以来，它们以难以置信的速度变得更快更灵活。早期的可编程 GPU 只在顶点级支持可编程，而今天复杂的逐像素程序已经很常见了。一年前，实时 GPU 程序的长度经常是几十条指令，而今年的 GPU 可以处理有上百条指令的复杂程序，并仍可以以交互的速率渲染。可编程的图形甚至已经超越了 PC 并快速地扩展到控制台、手持式游戏设备和移动电话。

直到最近，关注性能的开发可能仍在考虑用汇编语言编写 GPU 程序。然而，现在高级 GPU 编程语言已经普及，开发者为 GPU 编写汇编程序而烦恼已经极其罕见了，这要感谢编译器的进步和 GPU 能力的快速提高。作为对比的是，游戏开发者从使用 CPU 汇编语言发展到用高级语言写游戏花费了多年时间。

这种快速的变化自然使“精粹”风格的书适合于汇集这种尖端技术，并把它扩散到开发者论坛。通过公认专家编写的特色章节，本书提供了该领域中众多最令人兴奋的新思路。

图形硬件和编程环境的革新已经在如何使用可编程性方面激发了进一步的革新。虽然可编程着色一直是离线软件渲染的主要产品，但在 GPU 上，可编程性的出现导致了用于可编程着色的各种新技术的发明，现在的发展远远地超过了程序化模式生成和纹理合成，GPU 上着色器尖端技术的使用正在快速进入全新的领域，这引领了动画、光照和粒子系统等方面新技术的发展。

GPU 的灵活性和速度已经培养了对“在 GPU 上做超越计算机图形的计算”的浓厚兴趣：在 GPU 上的通用计算(或 GPGPU)。“GPU 精粹”系列中的本书用了大量篇幅讲解这个新主题，包括 GPGPU 编程技术的简介、若干代表性的应用和关键算法的深入讨论。随着 GPU 性能的增长继续强于 CPU 的增长，对越来越多的程序员来说，这些主题将会越来越重要，因为 GPU 将对很多计算密集型应用提供更好的结果。

基于这种背景，我们为参与《GPU 精粹 2》发出一个公开号召。而得到的回应是压倒性的：在开放提交的短时间内就收到了 150 多章的提议，包括关于 GPU 编程的各个主题。本书只能包含其中的 1/3；许多优秀的提议无法被包括进来纯粹是因为篇幅所限。编辑们很艰难地把那些内容削减成本书的 48 章。我们感谢提交了提议的每个人。

本书中的内容都通过了一系列严格的复审过程，本书的编辑、各章的作者以及在某些情况下来自 NVIDIA 的其他书评人都仔细地读过这些内容并给出了改进或修改建议。由于书评人为本书提供了高质量的反馈，因此最终内容的质量有了显著地改进。我们感谢所有的书评人，他们为本书的编撰付出了大量时间和努力。

## 读者对象

我们希望读者熟悉计算机图形学和 GPU 编程的基础，包括图形 API，如 Direct3D 和 OpenGL，以及 GPU 语言，如 HLSL，GLSL 和 Cg。对 GPGPU 编程感兴趣的读者可能发现对并行编程的概念有一些基本了解是有帮助的。

游戏、可视化应用程序和其他交互式应用程序的开发者，以及计算机图形学的研究者，将会发现本书是一个无价的日常资源。尤其是下一代控制台的开发者将会发现很多及时和可用的内容。

## 范例

本书附有 CD-ROM，内容包含书中所描述技术的代码示例、视频和其他示范。这张 CD-ROM 是对书中想法的有价值的补充。在很多情况下，由作者提供的可实践范例将会提供更多的启迪。您可以在本书的网站 [http:// developer.nvidia.com/GPUGems2/](http://developer.nvidia.com/GPUGems2/) 找到样章、更新的 CD-ROM 内容和补充材料等。

## 致谢

本书融入了许多人的巨量工作。首先，撰稿人在很短的时间内写了很多很好的章节。他们的努力已经成为有价值的、及时的且能启发思维的合集。

部分编辑 Kevin Bjorke, Cem Cebenoyan, Simon Green, Mark Harris, Craig Kolb 和 Matthias Wloka——在本书上花费了很多时间，他们与作者一起修改、润色章节，直到它们变得出色为止，和作者商议最好的 GPU 编程实践，并且友善地提醒他们截止日期。没有这些编辑们的专注和奉献，我们现在仍然处于提交的提议状态。Chris Seitz 也细心地关注有关本书编撰过程中的许多法律、后勤和商业问题。

NVIDIA 公司的其他人也是本书的贡献者。我们在此感谢 Spender Yuen，他耐心地在书的图片和封面上做了令人惊奇的工作。当插图的数量超过 150 幅后，Helen Ho 也来帮忙了。我们感谢 Caroline Lie 和她的团队，他们持续不断地支持我们的项目。同样地，在我们进行项目开发时，Teresa Saffaie 和 Catherine Kilkenny 总是乐意帮着我们审稿。Jim Black 协调开发者和撰稿者之间的通信，包括 Tim Sweeney，感谢他为本书撰写了这么精彩的序言。

Addison-Wesley 的 Peter Gordon, Julie Nahil 和 Kim Boedigheimer 监管了这个项目，并帮助督促生产流水线，我们才可以尽可能及时地发行本书。Christopher Keane 的审稿技巧和 Jules Keane 的协助不可估量地改善了内容，最后完成的时候，Curt Johnson 为本书的上市提供了帮助。

NVIDIA 管理团队一些成员的支持对本项目的成功是功不可没的。Mark Daly 和 Dan Vivoli 发现了为“GPU 精粹”系列增加第 2 卷的价值并始终支持本书。Nick Triantos 给了 Matt 时间为这个项目工作并在若干 GPGPU 章节上给予了反馈。Jonah Alben 和 Tony Tamasi 对关于 GeForce 6 系列架构的章节提供了深刻见解和有价值的反馈。我们真诚地感谢 Jen-Hsun Huang，他建立了这个项目，构建出创新的、具有挑战性的和深谋远虑的环境，使 NVIDIA 公司成为一个如此令人愉快的工作地点。

最后，我们感谢 NVIDIA 公司的所有的同事，感谢他们一天天推进计算机图形学的发展，他们的努力使这一项目成为可能。

Matt Pharr, NVIDIA 公司

Randima (Randy) Fernando, NVIDIA 公司



# 目 录

## 第 I 部分 几何复杂性

### 第 1 章 实现照片级真实感的虚拟

#### 植物 ..... 5

#### 1.1 场景管理 ..... 6

##### 1.1.1 种植栅格 ..... 6

##### 1.1.2 种植策略 ..... 6

##### 1.1.3 实时优化 ..... 7

#### 1.2 草层 ..... 7

##### 1.2.1 通过溶解模拟 Alpha 透明 ..... 9

##### 1.2.2 变化 ..... 10

##### 1.2.3 光照 ..... 11

##### 1.2.4 风 ..... 12

#### 1.3 地面杂物层 ..... 12

#### 1.4 树和灌木层 ..... 13

#### 1.5 阴影 ..... 14

#### 1.6 后处理 ..... 15

##### 1.6.1 天空圆顶辉散 ..... 16

##### 1.6.2 全场景辉光 ..... 16

#### 1.7 本章小结 ..... 17

#### 参考文献 ..... 18

### 第 2 章 使用基于 GPU 几何体剪切图

#### 的地形渲染 ..... 19

#### 2.1 几何体剪切图简介 ..... 19

#### 2.2 GPU 实现概览 ..... 21

##### 2.2.1 数据结构 ..... 22

##### 2.2.2 剪切图大小 ..... 22

#### 2.3 渲染 ..... 23

##### 2.3.1 活动层 ..... 23

##### 2.3.2 顶点和索引缓冲区 ..... 23

##### 2.3.3 视锥剪切 ..... 24

##### 2.3.4 DrawPrimitive 调用 ..... 25

##### 2.3.5 顶点着色器 ..... 25

##### 2.3.6 像素着色器 ..... 27

#### 2.4 更新 ..... 28

##### 2.4.1 升采样 ..... 28

##### 2.4.2 残差 ..... 29

##### 2.4.3 法线图 ..... 30

#### 2.5 结果和讨论 ..... 30

#### 2.6 本章小结和改进 ..... 31

##### 2.6.1 顶点纹理 ..... 31

##### 2.6.2 去掉法线图 ..... 31

##### 2.6.3 不需要存储空间的地形

##### 合成 ..... 31

#### 参考文献 ..... 31

### 第 3 章 几何体实例化的内幕 ..... 33

#### 3.1 为什么要对几何体实例化? ..... 34

#### 3.2 定义 ..... 34

##### 3.2.1 几何体包 ..... 34

##### 3.2.2 实例属性 ..... 35

##### 3.2.3 几何体实例 ..... 35

##### 3.2.4 渲染和纹理场景 ..... 35

##### 3.2.5 几何体批次 ..... 36

#### 3.3 实现 ..... 37

##### 3.3.1 静态批次 ..... 38

##### 3.3.2 动态批次 ..... 39

##### 3.3.3 顶点常量实例化 ..... 40

##### 3.3.4 几何体实例 API 批次 ..... 43

#### 3.4 本章小结 ..... 46

#### 参考文献 ..... 48

第 4 章 分段缓冲	49
4.1 问题空间	49
4.2 解决方案	50
4.3 方法	50
4.3.1 分段缓冲的第一步	50
4.3.2 分段缓冲的第二步	50
4.3.3 分段缓冲的第三步	51
4.4 改进分段缓冲技术	51
4.5 本章小结	51
参考文献	51
第 5 章 用多流来优化资源管理	53
5.1 概览	53
5.2 实现	55
5.2.1 DirectX 9.0 中的多流	55
5.2.2 资源管理	57
5.2.3 处理顶点	59
5.3 本章小结	63
参考文献	63
第 6 章 让硬件遮挡查询发挥作用	65
6.1 引言	65
6.2 受益于遮挡查询的场景	66
6.3 遮挡裁减	66
6.4 层的停等方法	67
6.4.1 为什么使用层	67
6.4.2 层结构	67
6.4.3 层的算法	68
6.4.4 问题 1: 停滞	68
6.4.5 问题 2: 查询的额外开销	68
6.5 一致性层裁减	69
6.5.1 想法 1: 猜测	69
6.5.2 想法 2: 提升, 提升	70
6.5.3 算法	70
6.5.4 实现细节	71
6.5.5 停滞比较少的原因	73
6.5.6 查询较少的原因	73
6.5.7 如何遍历层	73
6.6 优化	74
6.6.1 用真正的几何体查询	74

6.6.2 只有 Z 的渲染遍	74
6.6.3 近似的可见性	74
6.6.4 保守的可见性测试	74
6.7 本章小结	75
参考文献	76

第 7 章 带有位移映射的细分表面自适应镶嵌	77
7.1 细分表面	77
7.1.1 一些定义	78
7.1.2 Catmull-Clark 细分	78
7.1.3 用细分来镶嵌	79
7.1.4 面片化表面	80
7.1.5 GPU 镶嵌算法	80
7.1.6 致密镶嵌	84
7.2 位移映射	84
7.2.1 改变平滑度测试	85
7.2.2 用法线映射着色	85
7.3 本章小结	86
参考文献	86

第 8 章 使用距离函数的逐像素位移映射	87
8.1 简介	87
8.2 准备工作	89
8.3 距离映射算法	89
8.4 计算距离图	92
8.5 着色器	92
8.5.1 顶点着色器	92
8.5.2 片段着色器	92
8.5.3 关于过滤的注意事项	94
8.6 结果	94
8.7 本章小结	95
参考文献	96

## 第 II 部分 着色、光照和阴影

第 9 章 S.T.A.L.K.E.R. 中的延期着色	101
9.1 引言	101
9.2 几种观点	102

9.3 优化	103	11.5 本章小结	132
9.3.1 优化的对象	103	参考文献	132
9.3.2 光照优化	104	第 12 章 基于贴面的纹理映射	133
9.3.3 G 缓冲区建立的优化	106	12.1 方法简介	134
9.3.4 阴影优化	108	12.2 纹理贴面的构造	135
9.4 改善质量	109	12.3 纹理贴面打包	135
9.4.1 “虚拟位置”的威力	109	12.4 纹理贴面映射	137
9.4.2 环境遮挡	110	12.5 mipmap 问题	138
9.4.3 材质和表面光照的交互	111	12.6 本章小结	140
9.5 反走样	111	参考文献	140
9.5.1 高效的调和映射	113	第 13 章 在 GPU 上实现 mental images 的 Phenomena 渲染器	141
9.5.2 处理透明	114	13.1 引言	141
9.6 尝试过但没有包含入最终 代码的内容	114	13.2 着色器和 Phenomena	142
9.6.1 高程图	114	13.3 用 Cg 实现 Phenomena	143
9.6.2 实时的全局照明	115	13.3.1 Cg 顶点程序和可变 参数	144
9.7 本章小结	115	13.3.2 片段程序着色器的 main() 入口点	145
参考文献	116	13.3.3 通用着色器接口	145
第 10 章 动态辐照度环境映射实时 计算	117	13.3.4 一个简单的着色器 例子	146
10.1 辐照度(irradiance)环境 映射	117	13.3.5 全局的状态变量	148
10.2 球面调和卷积	119	13.3.6 光着色器	149
10.3 映射到 GPU 上	120	13.3.7 纹理着色器	151
10.3.1 空域到频域	121	13.3.8 凹凸映射	152
10.3.2 卷积和恢复	122	13.3.9 环境着色器和体着 色器	153
10.4 以后的工作	123	13.3.10 返回结构体的着色器	154
10.5 本章小结	123	13.3.11 渲染毛发	154
参考文献	123	13.3.12 组合所有东西	155
第 11 章 近似的双向纹理函数	125	13.4 本章小结	155
11.1 引言	125	参考文献	156
11.2 采集	126	第 14 章 动态环境遮挡和间接光照	157
11.2.1 建立和采集	126	14.1 表面元素	158
11.2.2 汇集着色图	127	14.2 环境遮挡	158
11.3 渲染	128		
11.3.1 细节算法	128		
11.3.2 实时渲染	129		
11.4 结果	130		

14.2.1	多遍阴影算法	160	16.4.1	去除一个维度	184
14.2.2	改善性能	160	16.4.2	去除其他维度	184
14.3	间接光照和面光源	162	16.5	实现散射的着色器	185
14.4	本章小结	164	16.5.1	顶点着色器	185
	参考文献	164	16.5.2	片段着色器	187
<b>第 15 章</b>	<b>蓝图渲染和草图绘制</b>	<b>165</b>	16.6	增加高动态范围渲染	188
15.1	基本原理	166	16.7	本章小结	188
15.1.1	中间渲染结果	166		参考文献	189
15.1.2	边增强	166	<b>第 17 章</b>	<b>利用像素着色器分支的</b>	
15.1.3	深度子图形渲染	167		<b>高效模糊边缘阴影</b>	<b>191</b>
15.2	蓝图渲染	167	17.1	现有的阴影生成技术	191
15.2.1	深度剥离	167	17.2	用单张阴影图产生模糊	
15.2.2	析取可见边和不可		阴影		192
见边		169	17.2.1	模糊尖锐边缘阴影	192
15.2.3	合成蓝图	170	17.2.2	提高效率	195
15.2.4	深度屏蔽	171	17.2.3	实现细节	196
15.2.5	使用蓝图渲染显示		17.3	本章小结	199
建筑		171		参考文献	200
15.3	草图渲染	171	<b>第 18 章</b>	<b>将顶点纹理位移用于水的</b>	
15.3.1	边和颜色面片	172		<b>真实感渲染</b>	<b>201</b>
15.3.2	应用不确定性	172	18.1	水的模型	202
15.3.3	调整深度	173	18.2	实现	202
15.3.4	草图渲染的变体	173	18.2.1	水的表面模型	202
15.3.5	控制不确定性	174	18.2.2	实现细节	203
15.3.6	减少雨景效果	175	18.2.3	对高度图采样	203
15.4	本章小结	176	18.2.4	质量的提高与优化	204
	参考文献	176	18.2.5	渲染局部的扰动	208
<b>第 16 章</b>	<b>精确的大气散射</b>	<b>179</b>	18.3	本章小结	209
16.1	引言	179		参考文献	209
16.2	解散射方程	180	<b>第 19 章</b>	<b>通用的折射模拟</b>	<b>211</b>
16.2.1	Rayleigh 散射与 Mie		19.1	基本方法	212
散射		180	19.2	折射掩码	213
16.2.2	相位函数	181	19.3	示例	215
16.2.3	外向散射方程	181	19.3.1	水的模拟	215
16.2.4	内向散射方程	182	19.3.2	玻璃的模拟	217
16.2.5	表面散射方程	182	19.4	本章小结	219
16.3	实时渲染	182		参考文献	219
16.4	挤入着色器中	184			

## 第III部分 高质量渲染

第 20 章	快速三阶纹理过滤	225
20.1	高阶过滤	225
20.2	快速递归三次卷积	226
20.3	mipmapping	230
20.4	导数重建	232
20.5	本章小结	235
	参考文献	236
第 21 章	高质量反走样的光栅化	237
21.1	概述	237
21.2	降采样	239
21.2.1	与现有软硬件的对比	239
21.2.2	用 GPU 进行降采样	240
21.3	延伸	240
21.4	过滤器的细节	241
21.5	两遍分离式的过滤器	242
21.6	分块和累加	243
21.7	代码	243
21.7.1	渲染循环	244
21.7.2	降采样类	245
21.7.3	实现细节	246
21.8	本章小结	246
	参考文献	247
第 22 章	快速的预过滤线条	249
22.1	为什么尖锐的直线看起来很糟糕	249
22.2	限制信号的带宽	250
22.3	预处理	252
22.4	运行时	253
22.4.1	线段的建立(CPU)	253
22.4.2	表查找(GPU)	254
22.5	实现的问题	256
22.5.1	绘制宽线	256
22.5.2	组合多条线段	256
22.6	示例	256
22.7	本章小结	258
	参考文献	258

第 23 章	Nalu Demo 的头发动画和渲染	261
23.1	头发的几何体	262
23.1.1	布局 and 增长	262
23.1.2	控制头发	263
23.1.3	数据流	263
23.1.4	镶嵌	263
23.1.5	插值	264
23.2	动力学和碰撞	265
23.2.1	约束条件	265
23.2.2	碰撞	266
23.2.3	鳍	266
23.3	头发的着色	267
23.3.1	用于头发的实时反射模型	268
23.3.2	头发中实时的体化阴影	271
23.4	本章小结和未来的工作	274
	参考文献	274
第 24 章	使用查找表加速颜色变换	275
24.1	查找表的基础知识	275
24.1.1	一维查找表	275
24.1.2	三维查找表	276
24.1.3	插值	278
24.2	实现	278
24.2.1	把查找表映射到 GPU 的策略	278
24.2.2	Cg 着色器	278
24.2.3	系统集成	280
24.2.4	把三维查找表扩展到用于高动态范围图像	281
24.3	本章小结	282
	参考文献	282
第 25 章	Apple Motion 中的 GPU 图像处理	285
25.1	设计	285
25.1.1	喜爱的和厌恶的	285
25.1.2	选择语言	287
25.1.3	CPU 向后支持	287

25.2	实现	288
25.2.1	GPU 资源的限制	288
25.2.2	被零除	289
25.2.3	丢失的顶点分量	289
25.2.4	双线过滤	290
25.2.5	高精度存储	294
25.3	调试	294
25.4	本章小结	295
	参考文献	296
<b>第 26 章</b>	<b>实现改进的 Perlin 噪声</b>	<b>297</b>
26.1	随机但平滑	297
26.2	存储与计算	297
26.3	实现细节	298
26.4	本章小结	302
	参考文献	302
<b>第 27 章</b>	<b>高级的高质量过滤</b>	<b>303</b>
27.1	在 GPU 上实现过滤	303
27.1.1	访问图像样本	303
27.1.2	卷积过滤	304
27.2	数字图像的重采样	307
27.2.1	背景知识	307
27.2.2	反走样问题	307
27.2.3	图像重建	310
27.3	冲击过滤: 锐化图像的方法	312
27.4	过滤器的实现技巧	314
27.5	高级应用	314
27.5.1	时间变形	314
27.5.2	运动模糊的消除	314
27.5.3	自适应的纹理过滤	315
27.6	本章小结	315
	参考文献	315
<b>第 28 章</b>	<b>Mipmap 级的测量</b>	<b>317</b>
28.1	哪个 mipmap 层是可见的?	318
28.2	GPU 抢险队	318
28.2.1	像素点计数	318
28.2.2	引擎中的实际考虑	321

28.2.3	扩展	322
28.3	实验结果	324
28.4	本章小结	325
	参考文献	326

## 第IV部分 GPU 的通用计算: 初级读本

<b>第 29 章</b>	<b>流式体系结构和技术趋势</b>	<b>331</b>
29.1	技术趋势	331
29.1.1	核心技术趋势	331
29.1.2	后果	332
29.2	高性能计算的关键	334
29.2.1	高效计算的方法	334
29.2.2	高效通信的方法	335
29.2.3	与 CPU 对比	335
29.3	流式计算	336
29.3.1	流式编程模型	336
29.3.2	构建一个流式处理器	337
29.4	未来和挑战	338
29.4.1	技术趋势	338
29.4.2	功耗管理	338
29.4.3	支持更高的可编程性和功能性	339
29.4.4	来自 CPU 的 GPU 功能性 (或反之亦然)	339
	参考文献	339

<b>第 30 章</b>	<b>GeForce 6 系列 GPU 的体系结构</b>	<b>341</b>
30.1	GPU 如何适合于整体计算系统	342
30.2	整体系统体系结构	342
30.2.1	图形操作的功能结构图	343
30.2.2	非图形操作的功能结构图	346
30.3	GPU 特性	347
30.3.1	固定函数特性	348
30.3.2	着色器 Model 3.0 编程模型	349
30.3.3	支持的数据存储格式	353

30.4	性能	354	32.3	实现散列	373
30.5	达到最佳性能	354	32.3.1	转换成聚集	373
30.5.1	积极地使用 z 裁减	355	32.3.2	地址排序	374
30.5.2	加载数据时利用纹理 数学	355	32.3.3	渲染点	375
30.5.3	使用片段程序的分支	355	32.4	本章小结	375
30.5.4	尽可能使用 fp16 作中 间值	355		参考文献	376
30.6	本章小结	356			
<b>第 31 章</b>	<b>把计算概念映射到 GPU</b>	<b>357</b>	<b>第 33 章</b>	<b>在 GPU 上实现高效的并行 数据结构</b>	<b>377</b>
31.1	数据并行的的重要性	357	33.1	流式编程	377
31.1.1	哪种类型的计算可以很 好地映射到 GPU	357	33.2	GPU 存储器模型	379
31.1.2	示例: 在栅格上模拟	358	33.2.1	存储器体系结构	379
31.1.3	流通信: 聚集与散布	359	33.2.2	GPU 流类型	380
31.2	GPU 计算资源清单	359	33.2.3	GPU 核的存储器访问	381
31.3	CPU-GPU 类比	362	33.3	基于 GPU 的数据结构	382
31.3.1	流: GPU 纹理 = CPU 数组	362	33.3.1	多维数组	382
31.3.2	核: GPU 片段程序 = CPU “内循环”	362	33.3.2	结构体	387
31.3.3	渲染到纹理 = 反馈	362	33.3.3	稀疏数据结构	387
31.3.4	几何体光栅化 = 计算的 调用	363	33.4	性能考虑	391
31.3.5	纹理坐标 = 计算的域	363	33.4.1	依赖的纹理读取	391
31.3.6	顶点坐标 = 计算的 范围	363	33.4.2	计算频度和程序特化	391
31.3.7	缩减	363	33.4.3	Pbuffer Survival Guide	392
31.4	从类比到实现	364	33.5	本章小结	393
31.5	一个简单的例子	366		参考文献	393
31.6	本章小结	368	<b>第 34 章</b>	<b>GPU 流程控制习惯用法</b>	<b>395</b>
	参考文献	368	34.1	流程控制的挑战	395
<b>第 32 章</b>	<b>尝试 GPU 计算</b>	<b>369</b>	34.2	基本的流程控制策略	396
32.1	选择快速算法	369	34.2.1	判定	396
32.1.1	局部性	369	34.2.2	把分支向着流水线上端 移动	396
32.1.2	允许计算的准则	370	34.2.3	z 裁减	397
32.1.3	考虑下载和读回	371	34.2.4	分支指令	399
32.2	了解浮点	371	34.2.5	选择一种分支机制	399
			34.3	使用遮挡查询的数据依赖 循环	400
			34.4	本章小结	400
			<b>第 35 章</b>	<b>GPU 程序优化</b>	<b>401</b>
			35.1	数据并行计算	401

35.1.1	指令级并行性	401
35.1.2	数据级并行性	403
35.2	计算频率	404
35.2.1	循环内不变量的预 计算	405
35.2.2	用查找表进行预计算	406
35.2.3	避免内循环分支	407
35.2.4	swizzle 操作	407
35.3	评价和负载平衡	408
35.4	本章小结	409
	参考文献	410
<b>第 36 章 用于 GPGPU 应用程序的 流式缩减操作</b> 411		
36.1	通过紧缩来过滤	411
36.1.1	累加和扫描	412
36.1.2	通过搜索/聚集来散布	413
36.1.3	过滤性能	415
36.2	动机: 碰撞检测	416
36.3	用于细分表面的过滤	419
36.4	本章小结	421
	参考文献	421

## 第 V 部分 面向图像的计算

<b>第 37 章 GPU 上的八叉树纹理</b> 425		
37.1	一个 GPU 加速的层次结构: N3 树	426
37.1.1	定义	426
37.1.2	实现	427
37.2	应用 1: 在网格表面上色	431
37.2.1	建立八叉树	431
37.2.2	上色	432
37.2.3	渲染	432
37.2.4	把八叉树纹理转换成标准 2D 纹理	434
37.3	应用 2: 表面模拟	436
37.4	本章小结	437
	参考文献	438

<b>第 38 章 使用光栅化的高质量全局照 明渲染</b> 439		
38.1	通过光栅化的全局照明	440
38.2	最终聚集简介	441
38.2.1	两遍的方法	441
38.2.2	最终聚集	441
38.2.3	两遍方法的问题	442
38.3	通过光栅化的最终聚集	443
38.3.1	最终聚集光线的聚类	443
38.3.2	光线投射作为多次平行 投影	445
38.4	实现细节	446
38.4.1	初始化	446
38.4.2	深度剥离	446
38.4.3	采样	447
38.4.4	性能	447
38.5	GPU 上的全局照明渲染器	448
38.5.1	第一遍	448
38.5.2	生成可见点数据	448
38.5.3	第二遍	448
38.5.4	其他解决方案	449
38.6	本章小结	451
	参考文献	451

<b>第 39 章 使用逐步求精辐射度方法的 全局照明</b> 453		
39.1	辐射度的基础	454
	逐步求精	454
39.2	GPU 实现	455
39.2.1	使用半球投影的可 见性	456
39.2.2	构成因子的计算	458
39.2.3	选择下一个发射者	459
39.3	渐进细分	459
39.3.1	纹理四叉树	459
39.3.2	四叉数细分	460
39.4	性能	460
39.5	本章小结	460
	参考文献	461



第 40 章 GPU 上的计算机视觉	463	43.3 GPU 实现	499
40.1 引言	463	43.3.1 动态更新	499
40.2 实现框架	463	43.3.2 数据纹理的索引	499
40.3 应用示例	464	43.3.3 三角形划分	500
40.3.1 把一系列片段程序用于 计算机视觉	464	43.3.4 向量化	500
40.3.2 求和操作	467	43.4 试验结果	501
40.3.3 创建全景照片的方 程组	469	43.5 本章小结和工作展望	502
40.3.4 特征向量的计算	471	参考文献	502
40.4 并行计算机视觉处理	473	第 44 章 用于解线性方程组的 GPU 框架	505
40.5 本章小结	474	44.1 概述	505
参考文献	474	44.2 表示	506
第 41 章 延迟过滤: 困难数据格式的 渲染	477	44.2.1 “单浮点”的表示	506
41.1 引言	477	44.2.2 向量	506
41.2 为什么要延迟	478	44.2.3 矩阵	507
41.3 延迟过滤算法	479	44.3 运算	509
41.4 为什么它可以工作	481	44.3.1 向量运算	509
41.5 本章小结: 何时延迟	481	44.3.2 向量缩减	509
参考文献	482	44.3.3 矩阵与向量的积	510
第 42 章 保守光栅化	485	44.3.4 把所有的组合起来	511
42.1 问题定义	486	44.3.5 共轭梯度求解器	511
42.2 两种保守算法	487	44.4 一个偏微分方程的例子	512
42.2.1 剪切空间	487	44.5 本章小结	515
42.2.2 第一种算法	488	参考文献	516
42.2.3 第二种算法	489	第 45 章 GPU 上的期权定价	517
42.3 鲁棒性问题	492	45.1 期权概述	517
42.4 保守深度	492	45.2 Black-Scholes 模型	518
42.5 结果和本章小结	493	45.3 Lattice 模型	521
参考文献	494	45.3.1 二项模型	521
第 VI 部分 模拟与数值算法		45.3.2 欧式期权定价	522
第 43 章 蛋白质结构预测的 GPU 计算	497	45.4 本章小结	525
43.1 介绍	497	参考文献	526
43.2 Floyd-Warshall 算法以及 绑定距离的平滑	498	第 46 章 改进的 GPU 排序	527
		46.1 排序算法	527
		46.2 一种简单的方法	528
		46.3 快速排序	529
		46.3.1 实现奇偶合并排序	529