# <<OpenGL超级宝典>>

## 图书基本信息

书名：<<OpenGL超级宝典>>

13位ISBN编号：9787115274564

10位ISBN编号：7115274568

出版时间：2012-5

出版时间：人民邮电出版社

作者：Richard S. Wright,Nicholas Haemel,Graham Sellers,Benjamin Lipchak

页数：698

字数：1259000

译者：付飞,李艳辉

## 内容概要

《OpenGL超级宝典(第5版)》是OpenGL及3D图形编程最好的入门指南，涵盖了使用最新版本的OpenGL进行编程所需要的主要知识。
全书分三部分，共16章，另有3个附录。
第一部分包括第1章到第7章，介绍如何构建一个使用OpenGL的程序、如何设置3D渲染环境，以及如何创建基本对象和光线并对他们进行着色。
然后深入研究如何使用OpenGL，并向读者介绍GLSL，以及如何创建自己的着色器。
第二部分包括第8章到第12章，将进行更深入的研究，而懂得如何应用这些高级特性将使读者超越业余3D玩家的水平。
这一部分不仅能够使我们掌握更多的可视化效果，同时也考虑了性能表现。
第三部分包括第13章到第16章，着重介绍OpenGL如何支持和连接Windows、Mac OS X、Linux和掌上设备。
附录部分给出了更多阅读建议、术语表和API参考介绍。

《OpenGL超级宝典(第5版)》适合希望精通OpenGL以便对图形编程和3D图形知识进行扩展的程序员阅读，也可以帮助经验丰富的OpenGL程序员学习如何移植自己的应用程序。
本书既可以作为学习OpenGL的教材，也可以作为随时查阅的参考手册。

## 作者简介

Richard S. Wright, Jr.是Software Bisque公司的资深软件工程师，在这家公司使用OpenGL开发多媒体宇航和天文软件。
他曾经是OpenGL ARB在实时3D领域的代表人物，编写了大量基于OpenGL的游戏、科学与医学应用程序、数据库可视化工具和教育软件。

Nicholas Haemel在ATI和AMD的8年中引导了3D图形硬件和软件结构设计与开发，并对OpenGL的3.0、3.1、3.2和3.3标准作出了贡献。

Graham Sellers是AMD的OpenGL小组的一位管理者，领导着一个OpenGL软件开发团队，致力于开发AMD的OpenGL驱动程序。
他是ARB中的AMD代表，并对核心OpenGL 3.2、3.3和4.0规范作出了贡献。

Benjamin Lipchak是苹果公司的软件工程主管，领导着一个致力于图形开发技术和标准测试程序的团队，并负责iPhone和iPod touch的OpenGL ES一致性测试。
他曾经在AMD管理一个OpenGL ES驱动程序小组，并领导着Khronos的OpenGL生态系统小组，在那里他创建了OpenGL SDK和OpenGL Pipeline等刊物。

## 书籍目录

第一部分　基本概念

第1章　3D图形和OpenGL简介　2
1.1　计算机图形的简单历史回顾　2
1.1.1　进入电子时代　3
1.1.2　走向3D　3
1.2　3D图形技术和术语　6
1.2.1　变换(Transformation)和投影(Projection)　6
1.2.2　光栅化(Rasterization)　6
1.2.3　着色　7
1.2.4　纹理贴图　8
1.2.5　混合　9
1.2.6　将点连接起来　9
1.3　3D图形的常见用途　9
1.3.1　实时3D　10
1.3.2　非实时3D　12
1.3.3　着色器　12
1.4　3D编程的基本原则　13
1.4.1　并非工具包　13
1.4.2　坐标系统　13
1.4.3　投影：从3D到2D　17
1.5　总结　19
第2章　入门指南　20
2.1　什么是OpenGL?　20
2.1.1　标准的演化　21
2.1.2　OpenGL的未来　24
2.2　使用OpenGL　27
2.2.1　支持阵容　28
2.2.2　OpenGL API特性　29
2.2.3　OpenGL错误　31
2.2.4　确认版本　31
2.2.5　使用glHint获取线索　32
2.2.6　OpenGL状态机　32
2.3　建立Windows项目　33
2.3.1　包含路径　34
2.3.2　创建项目　35
2.3.3 添加文件　36
2.4　建立Mac OS X项目　38
2.4.1　自定义创建设置　38
2.4.2　创建新项目　39
2.4.3　框架、头文件和库　41
2.5　第一个三角形　43
2.5.1　要包含什么　45
2.5.2　启动GLUT　45
2.5.3　坐标系基础　47

2.5.4　完成设置　50
2.5.5　言归正传　52
2.6　加点儿活力！
　53
2.6.1　特殊按键　53
2.6.2　刷新显示　54
2.6.3　简单的动画片　54
2.7　总结　55
第3章　基础渲染　56
3.1　基础图形管线　57
3.1.1　客户机-服务器　57
3.1.2　着色器　58
3.2　创建坐标系　60
3.2.1　正投影　60
3.2.2　透视投影　61
3.3　使用存储着色器　61
3.3.1　属性　62
3.3.2　Uniform值　62
3.4　将点连接起来　64
3.4.1　点和线　64
3.4.2　绘制3D三角形　68
3.4.3　单独的三角形　68
3.4.4　一个简单批次容器　72
3.4.5　不希望出现的几何图形　73
3.4.6　多边形偏移　78
3.4.7　裁剪　80
3.5　混合　81
3.5.1　组合颜色　81
3.5.2　改变混合方程式　84
3.5.3　抗锯齿　85
3.5.4　多重采样　87
3.6　小结　89
第4章　基础变换：初识向量/矩阵　90
4.1　本章是令人生畏的数学课吗　90
4.2　3D图形数学速成课　91
4.2.1　向量　91
4.2.2　矩阵　94
4.3　理解变换　95
4.3.1　视觉坐标　95
4.3.2　视图变换　96
4.3.3　模型变换　96
4.3.4　模型视图的二元性　98
4.3.5　投影变换　98
4.3.6　视口变换　99
4.4　模型视图矩阵　99
4.4.1　矩阵构造　100
4.4.2　运用模型视图矩阵　103

4.5　更多对象　105
4.5.1　使用三角形批次类　105
4.5.2　创建一个球体　106
4.5.3　创建一个花托　106
4.5.4　创建一个圆柱或圆锥　107
4.5.5　创建一个圆盘　108
4.6　投影矩阵　108
4.6.1　正投影　109
4.6.2　透视投影　110
4.6.3　模型视图投影矩阵　111
4.7　变换管线　113
4.7.1　使用矩阵堆栈　114
4.7.2　管理管线　115
4.7.3　加点调料　118
4.8　使用照相机和角色进行移动　119
4.8.1　角色帧　120
4.8.2　欧拉角：“卢克！
请使用帧”　121
4.8.3　照相机管理　121
4.8.4　添加更多角色　123
4.8.5　关于光线　125
4.9　小结　126
第5章　基础纹理　127
5.1　原始图像数据　128
5.1.1　像素包装　129
5.1.2　像素图　130
5.1.3　包装的像素格式　132
5.1.4　保存像素　133
5.1.5　读取像素　134
5.2　载入纹理　137
5.2.1　使用颜色缓冲区　138
5.2.2　更新纹理　138
5.2.3　纹理对象　139
5.3　纹理应用　140
5.3.1　纹理坐标　140
5.3.2　纹理参数　142
5.3.3　综合运用　144
5.4　Mip贴图　148
5.4.1　Mip贴图过滤　149
5.4.2　生成Mip层　150
5.4.3　活动的Mip贴图　150
5.5　各向异性过滤　158
5.6　纹理压缩　160
5.6.1　压缩纹理　160
5.6.2　加载压缩纹理　161
5.6.3　最后一个示例　162
5.7　小结　163

第6章　跳出“盒子”：非存储着色器　164
6.1　初识OpenGL着色语言　164
6.1.1　变量和数据类型　165
6.1.2　存储限定符　168
6.1.3　真正的着色器　169
6.1.4　编译、绑定和连接　172
6.1.5　使用着色器　177
6.1.6　Provoking Vertex　178
6.2　着色器统一值　179
6.2.1　寻找统一值　179
6.2.2　设置标量和向量统一值　180
6.2.3　设置统一数组　180
6.2.4　设置统一矩阵　181
6.2.5　平面着色器　182
6.3　内建函数　184
6.3.1　三角函数　184
6.3.2　指数函数　184
6.3.3　几何函数　185
6.3.4　矩阵函数　185
6.3.5　向量相关函数　186
6.3.6　常用函数　187
6.4　模拟光线　189
6.4.1　简单漫射光　189
6.4.2　点光源漫反射着色器　191
6.4.3　ADS光照模型　194
6.4.4　Phong着色　197
6.5　访问纹理　199
6.5.1　只有纹理单元　200
6.5.2　照亮纹理单元　201
6.5.3　丢弃片段　203
6.5.4　卡通着色(Cell Shading)——将纹理单元作为光线　205
6.6　小结　207
第7章　纹理高级知识　208
7.1　矩形纹理　208
7.1.1　加载矩形纹理　209
7.1.2　使用矩形纹理　209
7.2　立方体贴图　212
7.2.1　加载立方体贴图　212
7.2.2　创建天空盒　213
7.2.3　创建反射　215
7.3　多重纹理　216
7.3.1　多重纹理坐标　217
7.3.2　多重纹理示例　217
7.4　点精灵(点块纹理)　219
7.4.1　使用点　220
7.4.2　点大小　220
7.4.3　综合运用　221

7.4.4　点参数　223
7.4.5　异形点　224
7.4.6　点的旋转　225
7.5　纹理数组　226
7.5.1　加载2D纹理数组　226
7.5.2　纹理数组索引　228
7.5.3　访问纹理数组　228
7.6　纹理代理　229
7.7　小结　230
第二部分　深入探索

第8章　缓冲区对象：存储尽在掌握　232
8.1　缓冲区　233
8.1.1　创建自己的缓冲区　233
8.1.2　填充缓冲区　234
8.1.3　像素缓冲区对象　235
8.1.4　缓冲区对象　241
8.2　帧缓冲区对象，摆脱窗口的限制　242
8.2.1　如何使用FBO　243
8.2.2　渲染缓冲区对象　243
8.2.3　绘制缓冲区　245
8.2.4　帧缓冲区的完整性　247
8.2.5　在帧缓冲区中复制数据　250
8.2.6　FBO综合运用　251
8.3　渲染到纹理　254
8.4　小结　259
第9章　高级缓冲区：超越基础水平　260
9.1　获得数据　260
9.1.1　映射缓冲区　261
9.1.2　复制缓冲区　262
9.2　控制像素着色器表现，映射片段输出　262
9.3　新一代硬件的新格式　264
9.3.1　浮点——最终的真正精确　264
9.3.2　多重采样　276
9.3.3　整数　279
9.3.4　sRGB　280
9.3.5　纹理压缩　281
9.4　小结　283
第10章　片段操作：管线的终点　284
10.1　裁剪——将几何图形剪切到希望的大小　285
10.2　多重采样　285
10.2.1　样本覆盖　285
10.2.2　样本遮罩　286
10.2.3　综合运用　287
10.3　模板操作　290
10.4　深度测试　292
10.4.1　深度截取　292

10.5 进行混合 293
10.5.1 混合方程式 293
10.5.2 混合函数 294
10.5.3 综合运用 295
10.6 抖动 296
10.7 逻辑操作 297
10.8 遮罩输出 298
10.8.1 颜色 298
10.8.2 深度 298
10.8.3 模板 298
10.8.4 用途 299
10.9 小结 299
第11章 高级着色器应用 300
11.1 高级顶点着色器 300
11.1.1 在顶点着色器中进行物理模拟 301
11.2 几何着色器 306
11.2.1 直通几何着色器 306
11.2.2 在应用程序中使用几何着色器 308
11.2.3 在几何着色器中丢弃几何图形 311
11.2.4 在几何着色器中修改几何图形 313
11.2.5 在几何着色器中生成几何图形 314
11.2.6 在几何着色器中改变图元类型 317
11.2.7 由几何着色器引入的新图元类型 319
11.3 高级片段着色器 321
11.3.1 片段着色器中的后期处理——颜色校正 322
11.3.2 片段着色器中的后期处理——卷积 323
11.3.3 在片段着色器中生成图像数据 326
11.3.4 在片段着色器中丢弃工作 328
11.3.5 逐片段控制深度 329
11.4 更高级的着色器函数 330
11.4.1 插值和存储限定符 330
11.4.2 高级内建函数 333
11.5 统一缓冲区对象 334
11.5.1 建立统一块 335
11.6 小结 342
第12章 高级几何图形管理 343
12.1 查询功能——收集OpenGL管线相关信息 343
12.1.1 准备查询 344
12.1.2 发出查询 345
12.1.3 取回查询结果 345
12.1.4 使用查询结果 346
12.1.5 让OpenGL决定 349
12.1.6 测量执行命令所需时间 350
12.2 在GPU内存中存储数据 352
12.2.1 使用缓冲区存储顶点数据 353
12.2.2 在缓冲区中保存顶点索引 356
12.3 使用顶点数组对象来组织缓冲区 358

12.4 高效地绘制大量几何图形 359
12.4.1 组合绘制函数 360
12.4.2 使用图元重启对几何图形进行组合 361
12.4.3 实例渲染 362
12.4.4 自动获得数据 367
12.5 存储变换的顶点——变换反馈 371
12.5.1 变换反馈 371
12.5.2 关闭光栅化 376
12.5.3 使用图元查询对顶点进行计数 376
12.5.4 使用图元查询的结果 378
12.5.5 变换反馈的应用实例 379
12.6 裁剪并确定绘制内容 386
12.6.1 裁剪距离——自定义裁剪空间 387
12.7 在OpenGL开始绘制时进行同步 389
12.8 小结 392
第三部分 特定平台应用

第13章 Windows上的OpenGL 394
13.1 Windows中的OpenGL实现 395
13.1.1 微软的OpenGL 395
13.1.2 现代图形驱动程序 395
13.1.3 扩展OpenGL 396
13.1.4 WGL扩展 398
13.2 基本窗口渲染 399
13.2.1 GDI设备环境 399
13.2.2 像素格式 400
13.2.3 OpenGL渲染环境 406
13.3 综合运用 409
13.3.1 创建窗口 410
13.4 全屏渲染 414
13.5 双重缓冲 415
13.5.1 消除视觉撕裂 415
13.6 小结 416
第14章 OS X上的OpenGL 417
14.1 OpenGL在Mac上的4种接口 417
14.2 在OpenGL中使用Cocoa 418
14.2.1 创建一个Cocoa程序 418
14.2.2 综合运用 423
14.2.3 双缓冲还是单缓冲 425
14.2.4 球体世界 425
14.3 全屏渲染 429
14.3.1 在Cocoa中进行全屏显示 430
14.4 CGL 435
14.4.1 同步帧速率 435
14.4.2 提高填充性能 436
14.4.3 多线程OpenGL 437
14.5 小结 437

第15章　Linux上的OpenGL　438
15.1　基础知识　438
15.1.1　简史　439
15.1.2　什么是X Window　439
15.2　入门讲解　439
15.2.1　检查OpenGL　440
15.2.2　设置Mesa　440
15.2.3　设置Mesa硬件驱动程序　441
15.2.4　设置GLUT 和 GLEW　441
15.2.5　创建OpenGL应用程序　442
15.3　GLX——X Window的接口　443
15.3.1　显示和X Window　444
15.3.2　配置管理和显示效果　444
15.3.3　窗口和渲染表面　447
15.3.4　OpenGL和GLX扩展　448
15.3.5　环境管理　448
15.3.6　同步　451
15.3.7　GLX查询　452
15.3.8　综合运用　453
15.4　小结　455
第16章　OpenGL ES：移动设备上的OpenGL　456
16.1　精简的OpenGL　456
16.1.1　ES指什么　457
16.1.2　历史概述　457
16.2　版本选择　458
16.2.1　ES 2.0　459
16.3　ES环境　463
16.3.1　应用程序设计的注意事项　463
16.3.2　有限环境的处理　464
16.3.3　定点数学　464
16.4　EGL: 新的窗口环境　465
16.4.1　EGL显示　466
16.4.2　创建窗口　467
16.4.3　环境管理　470
16.4.4　呈现缓冲区和渲染同步　471
16.4.5　更多关于EGL的内容　472
16.5　处理嵌入式环境　473
16.5.1　流行的操作系统　473
16.5.2　供应商特定扩展　473
16.5.3　个人玩家　473
16.6　苹果掌上平台　474
16.6.1　设置iPhone项目　474
16.6.2　移植到iPhone　477
16.7　小结　483
附录A　更多阅读建议　484
附录B　词汇表　486
附录C　(核心)OpenGL 3.3参考　489

## 章节摘录

版权页： 插图： 事实上，即使我们不懂得那些高深的3D图形数学知识，也不会造成太大的妨碍，就像我们不需要懂得任何关于汽车结构和内燃机方面的知识也能每天开汽车一样。
但是，我们最好对汽车足够了解，以便能够意识到需要时常更换机油，定期向油箱加油，以及在轮胎花纹磨光时要更换轮胎。
这些知识使我们成为一位可靠（还有安全！
）的车主。
同样，如果想要成为一位可靠和有能力的OpenGL程序员，也要遵循同样的标准。
至少需要理解那些基础知识，才知道能做什么，以及哪些工具适合我们要做的工作。
如果是初学者，我们将会发现，经过一段时间的实践，就会渐渐理解矩阵和向量，并且培养出一种更为直观（和强大）的能力，能够在实践中充分利用本章所介绍的这些概念。
因此，即使我们还没有能力在脑海中默算出两个矩阵的乘法，也要明白矩阵是什么，以及这些矩阵对于OpenGL处理来说意味着什么。
但是，在清理线性代数的老课本（每个人都有，对吧？
）之前，不要紧张，GLTools库中有一个组建叫做Math3d，其中包含了大量好用的与OpenGL一致的3D数学例程和数据类型。
虽然我们不必亲自进行所有的矩阵和向量操作，我们仍然知道它们是什么，以及如何应用它们。
看，这样我们就二者兼得了。
4.23D图形数学速成课 关于3D图形数学的书籍数不胜数，我们发现了一些非常好的，附录A中列出了这些书籍。
我们并不会做出一副要讲完所有需要了解的重要问题的架势，甚至并不试图讲完所有应该了解的问题。
在本章，我们只涉及真正需要了解的。
如果已经是数学高手，那么就应该跳过这一部分，立即开始学习模型视图矩阵部分。
这并不只是因为您已经知道了我们将要讲解的内容，还因为大多数数学高手会因为没有提供足够的空间来讨论它们喜爱的齐次坐标空间特性而感到不快。
想象一下一个我们必须从一大群鳄鱼逃脱出来的电视真人秀吧。
我们到底要知道多少3D数学才能生存？
这就是接下来两部分的内容－3D数学生存技能。
鳄鱼才不在乎我们是不是真正懂得齐次坐标 空间呢。
4.2.1向量 在第1章和第2章，我们已经介绍了顶点和3D笛卡尔坐标。
基本上，一个顶点就是XYZ坐标空间上的一个位置，而在空间中给定的一个位置恰恰是由一个且只由一个单独的XYZ三元组定义的。
然而，一组XYZ值还能表示一个向量（实际上，从纯粹的数学思维上讲一个顶点其实同时也是一个向量……这里我们只讨论主要问题）。
在使用向量来操纵3D几何图形时，向量可能就变成了最重要的基本概念了（没有之一）。
这3个值（×、Y和Z）组合起来表示两个重要的值——一个方向和一个数量。
如图4.1所示为空间中（任意选择）的一个点，以及空间中从坐标系原点到这个点坐标的一条带箭头的线段。
在拼接三角形时，这个点可以视为一个顶点，而这个带箭头的线段则可以视为一个向量。
一个向量首先是空间中从原点指向这个点的方向。
在中，我们总是使用向量来表示带方向的量。
例如，X轴就是向量（1，0，0）。
在×方向为+1，而在Y方向和Z方向则为0。

## 编辑推荐

1、作者为OpenGL资深专家 2、内容全面，最佳学习用书 3、针对OpenGL 3.3全面升级?4、增加OpenGL for iPhone、iPad5、一刀未剪足本6、亚马逊五星图书

## 版权说明