



李颖 薛海斌 朱伯立 朱仲立 编著

OpenGL

函数与范例解析手册

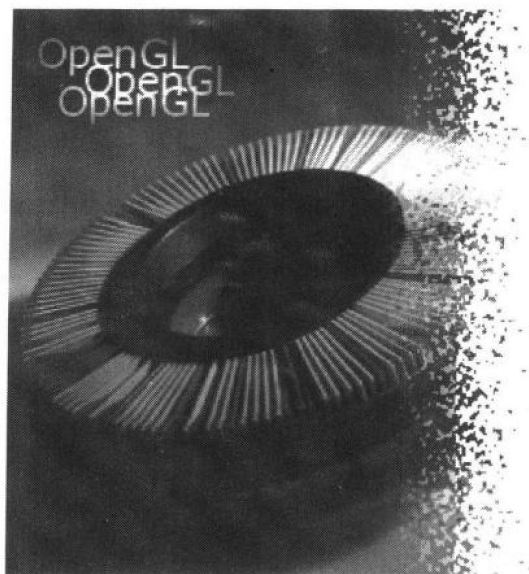
..... 工程师工具软件应用系列



国防工业出版社

TP391.41
192

国防工业出版社
www.ndip.com.cn



李颖 薛海斌 朱伯立 朱仲立 编著 ○

OpenGL ○

函数与范例解析手册

○○○○○○ 工程师工具软件应用系列



Z066755

内 容 简 介

本书主要介绍的是 OpenGL 的最新版本——OpenGL 1.2 版的命令函数。OpenGL 主要有三个函数库:GL 库、GLU 库和 GLUT 库,这三个库的所有函数在本书都有详细全面的介绍。同时,为了说明某些常用函数的使用方法,书中还给出了完整的程序示例,以便读者能够快速掌握这些函数的使用方法。

本书可以作为 OpenGL 程序设计者的辅助参考书,主要适用于 OpenGL 程序设计人员。

图书在版编目(CIP)数据

OpenGL 函数与范例解析手册/李颖等编著. —北京:
国防工业出版社,2002.1
(工程师工具软件应用系列)
ISBN 7-118-02588-7

I . O... II . 李 ... III . 图形软件,OpenGL - 程序设计 IV . TP391.41

中国版本图书馆 CIP 数据核字(2001)第 041286 号

国防工业出版社出版发行

(北京市海淀区紫竹院南路 23 号)

(邮政编码 100044)

北京奥隆印刷厂印刷

新华书店经售

*

开本 787×1092 1/16 印张 28 3/4 662 千字

2002 年 1 月第 1 版 2002 年 1 月北京第 1 次印刷

印数:1—3000 册 定价:39.00 元

(本书如有印装错误,我社负责调换)

前 言

作为三维图形硬件的软件接口,OpenGL 的主要目的是为了向帧缓存中绘制二维和三维几何物体,这些物体以一系列顶点或像素来描绘,顶点定义了几何物体,而像素则定义的是图像,OpenGL 对这些数据进行处理,以便把这些数据都转换为像素,从而构成帧缓存中最终所希望的图像,因此,利用 OpenGL 函数,程序员可以创建高质量的静态和动态三维彩色图像。

本书主要介绍的是 OpenGL 1.2 版的命令函数,在这一版中的 OpenGL 主要由以下三个库组成:GL 库、GLU (OpenGL Utility Library) 库和 GLUT 库 (OpenGL Utility Library Toolbox)。GL 库中包含的是 OpenGL 最基本的命令函数,利用这些函数可以执行如数据输入、指定光照和颜色、生成纹理坐标、转换窗口坐标、逻辑缓存操作,以及获取 OpenGL 状态变量等任务。GL 库中的函数均以 gl 字母开头。GLU 库是 OpenGL 的实用库,它是对 OpenGL 核心函数的补充,利用 GLU 库中的函数可以执行如管理纹理中使用的图像、转换坐标、绘制二维几何物体,以及绘制 NURBS 曲线和曲面等任务。GLU 库中的函数均以 glu 字母开头。GLUT 库是 OpenGL 的实用工具箱,它使得开发 OpenGL 程序变得更加容易,GLUT 库的函数主要执行如处理多窗口绘制、处理回调驱动事件、生成层叠式弹出菜单、绘制位图字体和笔画字体,以及各种窗口管理等任务。GLUT 库的函数均以 glut 字母开头。

本书中的函数是按照 GL 库、GLU 库和 GLUT 库的顺序编排的,对于每个库中的函数均按照函数功能进行了分类,并放在独立的章节中,同时给出了部分函数的程序示例代码,书中的示例程序均调试通过,可以作为读者学习和掌握 OpenGL 的很好的范例。本书的编写和程序调试工作由李颖、薛海斌和朱伯立完成。

本书对于需要使用 OpenGL 编制交互式三维真实场景的软件人员是一本非常有用的参考书。

作 者

2001 年 4 月

2001/4

目 录

第一篇 GL 库函数

第一章 使用颜色	1
1.1 glShadeModel —— 选择平面明暗模式或光滑明暗模式	1
1.2 glColor —— 设置当前颜色	2
1.3 glColorPointer —— 定义颜色数组	5
1.4 glIndex —— 设置当前颜色索引	8
1.5 glIndexPointer —— 定义颜色索引数组	11
1.6 glColorTableEXT——为目标调色板纹理指定调色板的格式和大小	12
1.7 glColorSubTableEXT——指定需要替代的目标纹理调色板的一部分	16
第二章 绘制几何图原及物体	19
2.1 glVertex —— 指定顶点	19
2.2 glVertexPointer —— 定义顶点数据数组	21
2.3 glArrayElement —— 指定用来绘制顶点的数组元素	22
2.4 glBegin, glEnd —— 限定一个或多个图原顶点的绘制	23
2.5 glEdgeFlag, glEdgeFlagv —— 指定边界标记	27
2.6 glPointSize —— 指定光栅化点的直径	28
2.7 glLineWidth —— 指定光栅化直线的宽度	31
2.8 glLineStipple —— 指定点划线	33
2.9 glPolygonMode —— 选择多边形光栅化模式	36
2.10 glFrontFace —— 定义正面多边形和反面多边形	37
2.11 glPolygonStipple —— 设置多边形点划图	38
2.12 glDrawElements —— 从数组数据绘制图原	41
2.13 glRect —— 绘制矩形	42
第三章 坐标转换	45
3.1 glTranslate —— 用平移矩阵乘以当前矩阵	45
3.2 glRotate —— 用旋转矩阵乘以当前矩阵	48
3.3 glScale —— 用缩放矩阵乘以当前矩阵	51
3.4 glViewport —— 设置视口	54
3.5 glFrustum —— 用透视矩阵乘以当前矩阵	55
3.6 glOrtho —— 用正视矩阵乘以当前矩阵	57
3.7 glClipPlane —— 指定切割几何物体的平面	60

第四章 堆栈操作	63
4.1 glLoadMatrix —— 用任意矩阵替换当前矩阵	63
4.2 glMultMatrix —— 用任意矩阵乘以当前矩阵	64
4.3 glMatrixMode —— 指定哪一个矩阵是当前矩阵	65
4.4 glPushMatrix, glPopMatrix —— 压入和弹出当前矩阵堆栈	65
4.5 glPushAttrib, glPopAttrib —— 压入和弹出属性堆栈	68
4.6 glPushClientAttrib, glPopClientAttrib —— 在客户属性堆栈中保存 和恢复客户状态变量组	73
4.7 glPushName, glPopName —— 压入和弹出名称堆栈	74
4.8 glInitNames —— 初始名称堆栈	75
4.9 glLoadName —— 向名称堆栈中装载名称	75
第五章 显示列表	79
5.1 glNewList, glEndList —— 创建或替换一个显示列表	79
5.2 glCallList —— 执行一个显示列表	80
5.3 glCallLists —— 执行一系列显示列表	81
5.4 glGenLists —— 生成一组空的相邻的显示列表	84
5.5 glDeleteLists —— 删除一组相邻的显示列表	85
5.6 glIsList —— 检验显示列表的存在	85
第六章 使用光照和材质	87
6.1 glNormal —— 设置当前的法向量	87
6.2 glNormalPointer —— 定义法向量数组	88
6.3 glLight —— 设置光源参数	93
6.4 glLightModel —— 设置光照模型参数	98
6.5 glMaterial —— 为光照模型指定材质参数	100
6.6 glColorMaterial —— 使材质颜色跟踪当前颜色	106
第七章 像素操作	110
7.1 glRasterPos —— 为像素操作指定光栅位置	110
7.2 glBitmap —— 绘制位图	113
7.3 glReadPixels —— 从帧缓存中读取一块像素	116
7.4 glDrawPixels —— 将一个像素块写入帧缓存	119
7.5 glCopyPixels —— 在帧缓存中拷贝像素	125
7.6 glCopyTexImage1D —— 将像素从帧缓存拷贝到一维纹理图像中	130
7.7 glCopyTexImage2D —— 把像素从帧缓存拷贝到二维纹理图像中	132
7.8 glCopyTexSubImage1D —— 从帧缓存中拷贝一维纹理图像的子图像	135
7.9 glCopyTexSubImage2D —— 从帧缓存中拷贝二维纹理图像的子图像	136
7.10 glPixelZoom —— 指定像素缩放因子	138
7.11 glPixelStore —— 设置像素存储模式	141
7.12 glPixelTransfer —— 设置像素传输模式	146
7.13 glPixelMap —— 设置像素传输映射表	149

第八章 纹理映射	153
8.1 glTexImage1D —— 指定一维纹理图像	153
8.2 glTexImage2D —— 指定二维纹理映射	156
8.3 glTexParameter —— 设置纹理参数	159
8.4 glTexSubImage1D —— 指定已存在的一维纹理图像的一部分	164
8.5 glTexSubImage2D —— 指定已存在的二维纹理图像的一部分	167
8.6 glTexEnv —— 设置纹理环境参数	169
8.7 glTexCoord —— 设置当前纹理坐标	171
8.8 glTexGen —— 控制纹理坐标的生成	178
8.9 glTexCoordPointer —— 定义纹理坐标数组	183
8.10 glDeleteTextures —— 删除命名的纹理	184
第九章 特殊效果操作	186
9.1 glBlendFunc —— 指定像素的数学算法	186
9.2 glHint —— 指定由实现确定的控制行为	190
9.3 glFog —— 指定雾化参数	193
第十章 帧缓存操作	199
10.1 glClear —— 将缓存清除为预先的设置值	199
10.2 glClearAccum —— 设置累加缓存的清除值	200
10.3 glClearColor —— 设置颜色缓存的清除值	201
10.4 glClearDepth —— 设置深度缓存的清除值	201
10.5 glClearIndex —— 设置颜色索引缓存的清除值	202
10.6 glClearStencil —— 设置模板缓存的清除值	202
10.7 glDrawBuffer —— 指定绘制的颜色缓存	203
10.8 glIndexMask —— 控制颜色索引缓存中单个位的写操作	204
10.9 glColorMask —— 激活或关闭帧缓存颜色分量的写操作	205
10.10 glDepthMask —— 激活或关闭对深度缓存的写操作	206
10.11 glStencilMask —— 控制模板平面中单个位的写操作	206
10.12 glAlphaFunc —— 指定 alpha 检验函数	207
10.13 glStencilFunc —— 设置模板检验函数和参考值	208
10.14 glStencilOp —— 设置模板检验操作	210
10.15 glDepthFunc —— 指定深度比较中使用的数值	214
10.16 glDepthRange —— 指定从单位化的设备坐标到窗口坐标的 z 值映射	215
10.17 glLogicOp —— 为颜色索引绘制指定逻辑像素操作	216
10.18 glAccum —— 对累加缓存进行操作	217
第十一章 绘制曲线和曲面	223
11.1 glEvalCoord —— 求取激活的一维和二维纹理图	223
11.2 glMap1 —— 定义一维求值器	225
11.3 glMap2 —— 定义二维求值器	230
11.4 glMapGrid —— 定义一维或二维网格	236

11.5	glEvalMesh —— 计算一维或二维点网格或线网格	238
11.6	glEvalPoint —— 生成并求取网格中的单个点	242
第十二章 查询函数		244
12.1	glGet —— 返回所选择的参数值	244
12.2	glGetClipPlane —— 返回指定的切平面系数	252
12.3	glGetColorTableEXT —— 获得当前目标纹理调色板的颜色表数据	253
12.4	glGetColorTableParameterfvEXT, glGetColorTableParameterivEXT —— 从颜色表中获得调色板参数	256
12.5	glGetError —— 返回错误信息	258
12.6	glGetLight —— 返回光源参数值	259
12.7	glGetMap —— 返回求值器参数	260
12.8	glGetMaterial —— 返回材质参数	262
12.9	glGetPixelMap —— 返回指定的像素映像	263
12.10	glGetPointerv —— 返回顶点数据数组地址	265
12.11	glGetPolygonStipple —— 返回多边形点划图	265
12.12	glGetString —— 返回描述当前 OpenGL 连接的字符串	266
12.13	glGetTexEnv —— 返回纹理环境参数	267
12.14	glGetTexGen —— 返回纹理坐标生成参数	268
12.15	glGetTexImage —— 返回纹理图像	269
12.16	glGetTexLevelParameter —— 返回指定细节水平的纹理参数值	271
12.17	glGetTexParameter —— 返回纹理参数值	272

第二篇 GLU 库函数

第一章 绘制 NURBS 曲线和曲面	275
1.1 gluNewNurbsRenderer —— 创建一个 NURBS 对像	275
1.2 gluNurbsProperty —— 设置 NURBS 属性	275
1.3 gluNurbsCallback —— 为 NURBS 对像定义回调函数	277
1.4 gluBeginCurve, gluEndCurve —— 限定 NURBS 曲线的定义	278
1.5 gluNurbsCurve —— 定义 NURBS 曲线的形状	278
1.6 gluDeleteNurbsRenderer —— 删除 NURBS 对像	281
1.7 gluBeginSurface, gluEndSurface —— 限定 NURBS 曲面的定义	282
1.8 gluNurbsSurface —— 定义 NURBS 曲面的形状	282
1.9 gluBeginTrim, gluEndTrim —— 限定 NURBS 裁剪环的定义	287
1.10 gluPwlCurve —— 描述分段线性 NURBS 裁剪曲线	288
1.11 gluBeginPolygon, gluEndPolygon —— 限定多边形的定义	292
1.12 gluPickMatrix —— 定义拾取区域	292

第二章 绘制二次几何物体	294
2.1 gluNewQuadric —— 创建一个二次对象	294
2.2 gluQuadricDrawStyle —— 指定二次对象的绘制方式	294
2.3 gluQuadricNormals —— 指定二次对象使用的法向量类型	295
2.4 gluQuadricOrientation —— 指定二次对象的内侧面或外侧面方向	295
2.5 gluCylinder —— 绘制圆柱体	296
2.6 gluSphere —— 绘制球体	297
2.7 gluDisk —— 绘制圆盘	298
2.8 gluPartialDisk —— 绘制部分圆盘	298
2.9 gluDeleteQuadric —— 删除二次对象	299
2.10 gluQuadricTexture —— 指定是否为二次对象使用纹理	303
2.11 gluQuadricCallback —— 为二次对象定义回调	304
第三章 网格化	305
3.1 gluNewTess —— 创建一个网格化对象	305
3.2 gluTessVertex —— 在多边形上指定顶点	305
3.3 gluTessCallback —— 为网格化对象定义回调	306
3.4 gluTessBeginPolygon, gluTessEndPolygon —— 限定多边形的描述	314
3.5 gluTessBeginContour, gluTessEndContour —— 限定多边形 轮廓线的定义	315
3.6 gluTessProperty —— 设置网格化对象的属性	315
3.7 gluNextContour —— 标记开始绘制另一个轮廓线	317
3.8 gluTessNormal —— 为多边形指定法向量	318
3.9 gluDeleteTess —— 删除网格化对象	319
第四章 坐标变换	323
4.1 gluOrtho2D —— 定义二维正视投影矩阵	323
4.2 gluPerspective —— 创建透视投影矩阵	323
4.3 gluLookAt —— 定义视景转换	324
4.4 gluProject —— 将物体坐标映射为窗口坐标	325
4.5 gluUnProject —— 将窗口坐标映射为物体坐标	325
第五章 多重映射	327
5.1 gluBuild1DMipmaps —— 创建一维多重映射	327
5.2 gluBuild2DMipmaps —— 创建二维多重映射	328
5.3 gluScaleImage —— 将图像缩放到任意尺寸	333
第六章 查询函数	335
6.1 gluErrorString —— 从 OpenGL 或 GLU 错误代码中生成错误字符串	335
6.2 gluGetNurbsProperty —— 获得 NURBS 属性	335
6.3 gluGetString —— 获得描述 GLU 版本号或支持 GLU 扩展调用的 字符串	336
6.4 gluGetTessProperty —— 获得网格化对象的属性	336

第三篇 GLUT 库函数

● 使用说明	339
● 专用术语	339
● 分类	340
第一章 初始化和启动事件处理	342
1.1 glutInit —— 初始化 GLUT 库	342
1.2 glutInitWindowPosition —— 设置初始窗口位置	342
1.3 glutInitWindowSize —— 设置初始窗口大小	343
1.4 glutInitDisplayMode —— 设置初始显示模式	343
1.5 glutMainLoop —— 进入 GLUT 事件处理循环	345
第二章 窗口管理	347
2.1 glutCreateWindow —— 创建顶层窗口	347
2.2 glutCreateSubWindow —— 创建子窗口	349
2.3 glutHideWindow —— 隐藏当前窗口的显示状态	349
2.4 glutShowWindow —— 改变当前窗口的显示状态,使其显示	350
2.5 glutSetWindowTitle —— 设置当前顶层窗口的窗口标题	350
2.6 glutSetIconTitle —— 设置当前顶层窗口的图标标题	350
2.7 glutPostRedisplay —— 标记当前窗口需要重新绘制	354
2.8 glutSwapBuffers —— 交换当前窗口的缓存	354
2.9 glutFullScreen —— 关闭全屏显示	357
2.10 glutPositionWindow —— 申请改变当前窗口的位置	358
2.11 glutReshapeWindow —— 申请改变当前窗口的大小	358
2.12 glutSetWindow —— 设置当前窗口	361
2.13 glutGetWindow —— 获得当前窗口的标识符	362
2.14 glutPopWindow —— 改变当前窗口的位置,使其前移	362
2.15 glutPushWindow —— 改变当前窗口的位置,使其后移	362
2.16 glutDestroyWindow —— 销毁指定的窗口	371
2.17 glutIconifyWindow —— 使当前窗口图标化显示	371
2.18 glutSetCursor —— 设置当前窗口的鼠标形状	371
第三章 重叠层管理	377
3.1 glutEstablishOverlay —— 创建当前窗口的重叠层	377
3.2 glutUseLayer —— 改变当前窗口的使用层	377
3.3 glutRemoveOverlay —— 删除当前窗口的重叠层	378
3.4 glutPostOverlayRedisplay —— 标记当前窗口的重叠层 需要重新绘制	378
3.5 glutShowOverlay —— 显示当前窗口的重叠层	379
3.6 glutHideOverlay —— 隐藏当前窗口的重叠层	379

第四章 菜单管理	388
4.1 glutCreateMenu —— 创建一个新的弹出式菜单	388
4.2 glutAddMenuEntry —— 在当前菜单的底部增加一个菜单条目	388
4.3 glutAddSubMenu —— 在当前菜单的底部增加一个子菜单触发条目	389
4.4 glutAttachMenu —— 把当前窗口的一个鼠标按键与当前菜单的标识符联系起来	389
4.5 glutGetMenu —— 获取当前菜单的标识符	389
4.6 glutSetMenu —— 设置当前菜单	392
4.7 glutDestroyMenu —— 删除指定的菜单	393
4.8 glutChangeToMenuEntry —— 将指定的当前菜单中的菜单项更改为菜单条目	393
4.9 glutChangeToSubMenu —— 将指定的当前菜单中的菜单项更改为子菜单触发条目	394
4.10 glutRemoveMenuItem —— 删除指定的菜单项	394
4.11 glutDetachMenu —— 释放当前窗口的一个鼠标按键	395
第五章 注册回调函数	396
5.1 glutDisplayFunc —— 注册当前窗口的显示回调函数	396
5.2 glutReshapeFunc —— 注册当前窗口的形状变化回调函数	396
5.3 glutMouseFunc —— 注册当前窗口的鼠标回调函数	397
5.4 glutMotionFunc —— 设置移动回调函数	398
5.5 glutIdleFunc —— 设置全局的空闲回调函数	398
5.6 glutVisibilityFunc —— 设置当前窗口的可视回调函数	399
5.7 glutKeyboardFunc —— 注册当前窗口的键盘回调函数	399
5.8 glutSpecialFunc —— 设置当前窗口的特定键回调函数	400
5.9 glutOverlayDisplayFunc —— 注册当前窗口的重叠层显示回调函数	411
5.10 glutPassiveMotionFunc —— 设置当前窗口的被动移动回调函数	411
5.11 glutEntryFunc —— 设置当前窗口的鼠标进出回调函数	412
5.12 glutSpaceballMotionFunc —— 设置当前窗口的空间球移动回调函数	412
5.13 glutSpaceballRotateFunc —— 设置当前窗口的空间球旋转回调函数	413
5.14 glutSpaceballButtonFunc —— 设置当前窗口的空间球按键回调函数	414
5.15 glutButtonBoxFunc —— 设置当前窗口的拨号按键盒按键回调函数	414
5.16 glutDialsFunc —— 设置当前窗口的拨号按键盒拨号回调函数	415
5.17 glutTabletMotionFunc —— 设置图形板移动回调函数	415
5.18 glutTabletButtonFunc —— 设置当前窗口的图形板按键回调函数	416
5.19 glutMenuStatusFunc —— 设置全局的菜单状态回调函数	417
5.20 glutTimerFunc —— 注册按一定时间间隔触发的定时器回调函数	418
第六章 颜色索引映射表管理	419
6.1 glutSetColor —— 设置当前窗口当前层一个颜色表单元的颜色	419
6.2 glutGetColor —— 获得指定的索引颜色	420

6.3	glutCopyColormap —— 将逻辑颜色表从指定的窗口拷贝到当前窗口	420
第七章	状态检索	422
7.1	glutGet —— 检索指定的 GLUT 状态	422
7.2	glutLayerGet —— 检索属于当前窗口重叠层的 GLUT 状态	424
7.3	glutDeviceGet —— 检索 GLUT 设备信息	425
7.4	glutGetModifiers —— 返回修饰键在引起某些回调的事件发生时的状态 ...	425
7.5	glutExtensionSupported —— 判别当前 OpenGL 版本是否支持 给定的 OpenGL 扩展	426
第八章	字体绘制	430
8.1	glutBitmapCharacter —— 绘制一个位图字符	430
8.2	glutBitmapWidth —— 返回一个位图字符的宽度	431
8.3	glutStrokeCharacter —— 绘制一个笔画字符	432
8.4	glutStrokeWidth —— 返回一个笔画字体的宽度	433
第九章	几何图形绘制	439
9.1	glutSolidSphere, glutWireSphere —— 绘制实心球体和线框球体	439
9.2	glutSolidCube, glutWireCube —— 绘制实心立方体和线框立方体	439
9.3	glutSolidCone, glutWireCone —— 绘制实心圆锥体和线框圆锥体	440
9.4	glutSolidTorus, glutWireTorus —— 绘制实心圆环和线框圆环	440
9.5	glutSolidDodecahedron, glutWireDodecahedron —— 绘制实心 十二面体和线框十二面体	441
9.6	glutSolidOctahedron, glutWireOctahedron —— 绘制实心 八面体和线框八面体	441
9.7	glutSolidTetrahedron, glutWireTetrahedron —— 绘制实心四面体 和线框四面体	441
9.8	glutSolidIcosahedron, glutWireIcosahedron —— 绘制实心二十面体 和线框二十面体	442
9.9	glutSolidTeapot, glutWireTeapot —— 绘制实心茶壶和线框茶壶	442

第一篇 GL 库函数

第一章 使用颜色

1.1 glShadeModel —— 选择平面明暗模式或光滑明暗模式

C 语言描述

```
void glShadeModel(GLenum mode)
```

参数

mode 指定表示明暗模式的符号值,可以选择 GL_FLAT 和 GL_SMOOTH,缺省值为 GL_SMOOTH。

说明

OpenGL 图原可以为平面明暗模式或光滑明暗模式,缺省时,光滑明暗模式在图原光栅化时插值计算顶点的颜色,实际上为每个最终的像素片元应用不同的颜色。平面明暗模式只选择一个顶点的计算颜色,并将该颜色应用到光栅化单个图原所产生的所有像素片元上。无论哪种情况,如果激活光照,计算到的顶点颜色都是光照后的结果颜色,如果关闭光照,计算到的颜色就是指定顶点时的当前颜色。

对于点,平面明暗模式和光滑明暗模式是没有区别的,当发出 glBegin 命令时,从 1 开始计数顶点和图原的数目,在平面明暗模式下,线段 i 的颜色就是计算到的顶点 $i+1$ 的颜色。同样从 1 开始计数,平面明暗模式下多边形的颜色由下面计算到的顶点颜色来确定,如表 1-1 所示。

表 1-1 平面明暗模式下多边形颜色的确定

第 i 个多边形的图原类型	顶 点
单个多边形($i=1$)	1
相邻的三角形	$i+2$
三角形扇形	$i+2$
单独的三角形	$3i$
相邻的四边形	$2i+2$
单独的四边形	$4i$

平面明暗模式和光滑明暗模式由 `glShadeModel` 指定, `mode` 变量分别为 `GL_FLAT` 和 `GL_SMOOTH`。

下列函数用来检索与 `glShadeModel` 相关的信息:

`glGet`, 变量为 `GL_SHADE_MODEL`。

错误

如果 `mode` 的数值是 `GL_FLAT` 或 `GL_SMOOTH` 之外的任何值, 则会产生 `GL_INVALID_ENUM` 错误。

如果在 `glBegin` 和对应的 `glEnd` 语句之间调用 `glShadeModel` 函数, 则会产生 `GL_INVALID_OPERATION` 错误。

参阅

`glBegin`, `glEnd`, `glColor`, `glLight`, `glLightModel`

1.2 `glColor` —— 设置当前颜色

`glColor3b`, `glColor3d`, `glColor3f`, `glColor3i`, `glColor3s`, `glColor3ub`, `glColor3ui`, `glColor3us`, `glColor4b`, `glColor4d`, `glColor4f`, `glColor4i`, `glColor4s`, `glColor4ub`, `glColor4ui`, `glColor4us`, `glColor3bv`, `glColor3dv`, `glColor3fv`, `glColor3iv`, `glColor3sv`, `glColor3ubv`, `glColor3uiv`, `glColor3usv`, `glColor4bv`, `glColor4dv`, `glColor4fv`, `glColor4iv`, `glColor4sv`, `glColor4ubv`, `glColor4uiv`, `glColor4usv`

C 语言描述

```
void glColor3b(GLbyte red,
               GLbyte green,
               GLbyte blue)
void glColor3d(GLdouble red,
               GLdouble green,
               GLdouble blue)
void glColor3f(GLfloat red,
               GLfloat green,
               GLfloat blue)
void glColor3i(GLint red,
               GLint green,
               GLint blue)
void glColor3s(GLshort red,
               GLshort green,
               GLshort blue)
void glColor3ub(GLubyte red,
```

```
        GLubyte green ,
        GLubyte blue)
void glColor3ui(GLuint red ,
               GLuint green ,
               GLuint blue)
void glColor3us(GLushort red ,
               GLushort green ,
               GLushort blue)
void glColor4b(GLbyte red ,
               GLbyte green ,
               GLbyte blue ,
               GLbyte alpha )
void glColor4d(GLdouble red ,
               GLdouble green ,
               GLdouble blue ,
               GLdouble alpha)
void glColor4f(GLfloat red ,
               GLfloat green ,
               GLfloat blue ,
               GLfloat alpha)
void glColor4i(GLint red ,
               GLint green ,
               GLint blue ,
               GLint alpha)
void glColor4s(GLshort red ,
               GLshort green ,
               GLshort blue ,
               GLshort alpha)
void glColor4ub(GLubyte red ,
               GLubyte green ,
               GLubyte blue ,
               GLubyte alpha)
void glColor4ui(GLuint red ,
               GLuint green ,
               GLuint blue ,
               GLuint alpha)
void glColor4us(GLushort red ,
               GLushort green ,
               GLushort blue ,
```

GLushort *alpha*)

参数

red, green, blue 指定当前新的红、绿、蓝颜色值。

alpha 指定当前新的 alpha 颜色值。只有在 glColor4 函数带 4 个变量时才指定此参数。

C 语言描述

```
void glColor3bv(const GLbyte * v)
void glColor3dv(const GLdouble * v)
void glColor3fv(const GLfloat * v)
void glColor3iv(const GLint * v)
void glColor3sv(const GLshort * v)
void glColor3ubv(const GLubyte * v)
void glColor3uiv(const GLuint * v)
void glColor3usv(const GLushort * v)
void glColor4bv(const GLbyte * v)
void glColor4dv(const GLdouble * v)
void glColor4fv(const GLfloat * v)
void glColor4iv(const GLint * v)
void glColor4sv(const GLshort * v)
void glColor4ubv(const GLubyte * v)
void glColor4uiv(const GLuint * v)
void glColor4usv(const GLushort * v)
```

参数

v 指定一个指向包含红、绿、蓝和 alpha 值的数组指针。

说明

OpenGL 既存储当前的单个颜色索引值,又存储当前的 4 个 RGBA 颜色值。glColor 函数设置新的 RGBA 颜色值,这个函数有两种主要的形式:glColor3 和 glColor4。glColor3 形式明确指定新的红、绿、蓝值,并设置当前的 alpha 值为 1.0。glColor4 形式指定所有的 4 个分量。

glColor3b、glColor4b、glColor3s、glColor4s、glColor3i 和 glColor4i 分别用 3 个或 4 个无符号字节、短整型或长整型数值做为它们的变量,如果在这些命令名称后再加上字符 *v* 时,那么这些颜色命令中的变量则是指向这些数值的数组指针。

当前的颜色值以浮点形式进行存储,不指定尾数和指数。如果指定的是无符号整数颜色分量,则这些值被线性映射为浮点值,最大的值映射为 1.0(全浓度),零映射为 0.0(零浓度)。当指定的是有符号整数颜色分量时,这些值也被线性映射为浮点值,最大的正

值映射为 1.0,最小的负值映射为 -1.0,浮点值则被直接映射。

在更改当前颜色之前,浮点值及有符号的整数值均不被箝位在[0,1]之间,然而,颜色分量在进行插值或写入到颜色缓存之前都被箝位在此范围之内。

下列函数可以用来检索与 glColor 相关的信息:

glGet, 变量为 GL_CURRENT_COLOR。

glGet, 变量为 GL_RGBA_MODE。

注意

可以在任何时候更改当前颜色,而且,glColor 可以在 glBegin 及对应的 glEnd 语句之间调用。

参阅

glBegin, glEnd, glShadeModel, glGet, glIndex

1.3 glColorPointer —— 定义颜色数组

C 语言描述

```
void glColorPointer (GLint size,
                    GLenum type,
                    GLsizei stride,
                    GLsizei count,
                    const GLvoid * pointer)
```

参数

- size* 每个颜色的分量数目,该值必须为 3 或 4。
- type* 在颜色数组中每个颜色分量的数据类型,可以选择的数据类型为:GL_BYTE、GL_UNSIGNED_BYTE、GL_SHORT、GL_UNSIGNED_SHORT、GL_INT、GL_UNSIGNED_INT、GL_FLOAT 或 GL_DOUBLE。
- stride* 相邻两个颜色的字节偏移量。当 *stride* 为零时,颜色值在数组中是一个接一个排列的。
- count* 静态颜色的数目,从第一个颜色开始计数。
- pointer* 指向颜色数组中第一个颜色元素中第一个分量的指针。

说明

glColorPointer 函数指定了绘制中使用的颜色分量数组的位置和数据格式,*stride* 参数指定了从一个颜色到下一个颜色之间的字节偏移量,在有些实现中,在单个数组中存储顶点属性可能比在几个数组中存储顶点属性效率更高。*count* 参数从第一个颜色数组元