OpenGL 函数库

开发基于 OpenGL 的应用程序，必须先了解 OpenGL 的库函数。它采用 C 语言风格，提供大量的函数来进行图形的处理和显示。OpenGL 图形库一共有 100 多个函数，它们分别属于 OpenGL 的基本库、实用库、辅助库等不同的库。

1、核心库

包含的函数有 115 个，它们是最基本的函数，其前缀是 gl；这部分函数用于常规的、核心的图形处理，由 gl.dll 来负责解释执行。核心库中的函数可以进一步分为以下几类函数。

（1）绘制基本几何图元的函数。

glBegain()、glEnd()、glNormal*()、glVertex*()。

（2）矩阵操作、几何变换和投影变换的函数。

矩阵入栈函数 glPushMatrix()，矩阵出栈函数 glPopMatrix()，装载矩阵函数 glLoadMatrix()，
矩阵相乘函数 glMultMatrix()，当前矩阵函数 glMatrixMode()和矩阵标准化函数 glLoadIdentity()，
几何变换函数 glTranslate*()、glRotate*()和 glScale*()，
投影变换函数 glOrtho()、glFrustum()和
视口变换函数 glViewport()等等。

（3）颜色、光照和材质的函数。

如设置颜色模式函数 glColor*()、glIndex*()，
设置光照效果的函数 glLight*() 、glLightModel*()
设置材质效果函数 glMaterial()等等。

（4）显示列表函数。

主要有创建、结束、生成、删除和调用显示列表的函数
glNewList()、 glEndList()、glGenLists()、glCallList()和 glDeleteLists()等。

（5）纹理映射函数。

主要有一维纹理函数 glTexImage1D()、二维纹理函数 glTexImage2D()、
设置纹理参数、纹理环境和纹理坐标的函数 glTexParameter*()、glTexEnv*()和 glTetCoord*()等。

（6）特殊效果函数。

融合函数 glBlendFunc()、反走样函数 glHint()和雾化效果 glFog*()。

（7）光栅化、象素操作函数。

像素位置 glRasterPos*()、线型宽度 glLineWidth()、多边形绘制模式 glPolygonMode()，
读取象素 glReadPixel()、复制象素 glCopyPixel()等。

（8）选择与反馈函数。

主要有渲染模式 glRenderMode()、选择缓冲区 glSelectBuffer()和反馈缓冲区 glFeedbackBuffer()等。

（9）曲线与曲面的绘制函数。

生成曲线或曲面的函数 glMap*()、glMapGrid*()，
求值器的函数 glEvalCoord*() glEvalMesh*()。

（10）状态设置与查询函数。

主要有 glGet*()、glEnable()、glGetError()等。

2、实用库（OpenGL utility library，GLU）

包含的函数功能更高一些，如绘制复杂的曲线曲面、高级坐标变换、多边形分割等，共有 43 个，前缀为 glu。Glu 函数通过调用核心库的函数，为开发者提供相对简单的用法，实现一些较为复杂的操作。此类函数由 glu.dll 来负责解释执行。主要包括了以下几种：

（1）辅助纹理贴图函数。

有 gluScaleImage() 、gluBuild1Dmipmaps()、gluBuild2Dmipmaps()等。

（2）坐标转换和投影变换函数。

定义投影方式函数 gluPerspective()、gluOrtho2D() 、gluLookAt()，
拾取投影视景体函数 gluPickMatrix()，投影矩阵计算 gluProject()和 gluUnProject()等。

（3）多边形镶嵌工具。

有 gluNewTess()、 gluDeleteTess()、gluTessCallback()、gluBeginPolygon() gluTessVertex()、
gluNextContour()、gluEndPolygon()等。

（4）二次曲面绘制工具。

主要有绘制球面、锥面、柱面、圆环面
gluNewQuadric()、gluSphere()、gluCylinder()、gluDisk()、gluPartialDisk()、gluDeleteQuadric()等等。

（5）非均匀有理 B 样条绘制工具。

主要用来定义和绘制 Nurbs 曲线和曲面，包括 gluNewNurbsRenderer()、gluNurbsCurve()、 gluBeginSurface()、

gluEndSurface()、gluBeginCurve()、gluNurbsProperty()等函数。

（6）错误反馈工具。

获取出错信息的字符串 gluErrorString()等。

3、OpenGL 辅助库（OpenGL auxiliarylibrary，GLAUX）

包括简单的窗口管理、输入事件处理、某些复杂三维物体绘制等函数，共有 31 个，前缀为 aux。此类函数由 glaux.dll 来负责解释执行。辅助库函数主要包括以下几类。

（1）窗口初始化和退出函数。

auxInitDisplayMode()和 auxInitPosition()。

（2）窗口处理和时间输入函数。

auxReshapeFunc()、auxKeyFunc()和 auxMouseFunc()。

（3）颜色索引装入函数。

auxSetOneColor()。

（4）三维物体绘制函数。

包括了两种形式网状体和实心体，如绘制立方体 auxWireCube()和 auxSolidCube()。

这里以网状体为例，长方体 auxWireBox()、环形圆纹面 auxWireTorus()、圆柱 auxWireCylinder()、二十面体 auxWireIcosahedron()、八面体 auxWireOctahedron()、四面体 auxWireTetrahedron()、十二面体 auxWireDodecahedron()、圆锥体 auxWireCone()和茶壶 auxWireTeapot()。

绘制实心体只要将上述函数中的确"Wire"更换成"Solid"就可以了。

（5）其他。

背景过程管理函数 auxIdleFunc()；程序运行函数 auxMainLoop()。

4、OpenGL 工具库（penGL Utility Toolkit）

包含大约 30 多个函数，函数名前缀为 glut，此函数由 glut.dll 来负责解释执行。这部分函数主要包括：

（1）窗口操作函数

窗口初始化、窗口大小、窗口位置等函数

glutInit() glutInitDisplayMode()、glutInitWindowSize() glutInitWindowPosition()等。

（2）回调函数。

响应刷新消息、键盘消息、鼠标消息、定时器函数等，

GlutDisplayFunc()、glutPostRedisplay()、 glutReshapeFunc()、glutTimerFunc()、glutKeyboardFunc()、glutMouseFunc()。

（3）创建复杂的三维物体。

这些和 aux 库的函数功能相同。创建网状体和实心体。如 glutSolidSphere()、glutWireSphere()等。

（4）菜单函数

创建添加菜单的函数

GlutCreateMenu()、glutSetMenu()、glutAddMenuEntry()、glutAddSubMenu() 和 glutAttachMenu()。

（5）程序运行函数。

glutMainLoop()。

5、16 个 WGL 函数

专门用于 OpenGL 和 Windows 窗口系统的联接，其前缀为 wgl，主要用于创建和选择图形操作描述表（renderingcontexts）以及在窗口内任一位置显示字符位图。这类函数主要包括以下几类

（1）绘图上下文相关函数。

wglCreateContext()、wglDeleteContext()、wglGetCurrentContent()、wglGetCurrentDC() wglDeleteContent()等。

（2）文字和文本处理函数。

wglUseFontBitmaps()、wglUseFontOutlines()。

（3）覆盖层、地层和主平面层处理函数。

wglCopyContext()、wglCreateLayerPlane()、 wglDescribeLayerPlane()、wglReakizeLayerPlatte()等。

（4）其他函数。

wglShareLists()、wglGetProcAddress()等。

6、另外

还有五个 Win32 函数用来处理像素格式（pixel formats）和双缓存。由于它们是对 Win32 系统的扩展，因此不能应用在其它 OpenGL 平台上。

OpenGl 核心函数库

glAccum	操作累加缓冲区
glAddSwapHintRectWIN	定义一组被 SwapBuffers 拷贝的三角形
glAlphaFunc	允许设置 alpha 检测功能
glAreTexturesResident	决定特定的纹理对象是否常驻在纹理内存中
glArrayElement	定义一个被用于顶点渲染的数组成分
glBegin,glEnd	定义一个或一组原始的顶点
glBindTexture	允许建立一个绑定到目标纹理的有名称的纹理
glBitmap	绘制一个位图
glBlendFunc	特殊的像素算法
glCallList	执行一个显示列表
glCallLists	执行一列显示列表
glClear	用当前值清除缓冲区
GlClearAccum	为累加缓冲区指定用于清除的值
glClearColor	为色彩缓冲区指定用于清除的值
glClearDepth	为深度缓冲区指定用于清除的值
glClearStencil	为模板缓冲区指定用于清除的值
glClipPlane	定义被裁剪的一个平面几何体
glColor	设置当前色彩
glColorMask	允许或不允许写色彩组件帧缓冲区
glColorMaterial	使一个材质色彩指向当前的色彩
glColorPointer	定义一列色彩
glColorTableEXT	定义目的一个调色板纹理的调色板的格式和尺寸
glColorSubTableEXT	定义目的纹理的调色板的一部分被替换
glCopyPixels	拷贝帧缓冲区里的像素
glCopyTexImage1D	将像素从帧缓冲区拷贝到一个单空间纹理图象中
glCopyTexImage2D	将像素从帧缓冲区拷贝到一个双空间纹理图象中
glCopyTexSubImage1D	从帧缓冲区拷贝一个单空间纹理的子图象
glCopyTexSubImage2D	从帧缓冲区拷贝一个双空间纹理的子图象
glCullFace	定义前面或后面是否能被精选
glDeleteLists	删除相邻一组显示列表
glDeleteTextures	删除命名的纹理
glDepthFunc	定义用于深度缓冲区对照的数据
glDepthMask	允许或不允许写入深度缓冲区
glDepthRange	定义 z 值从标准的设备坐标映射到窗口坐标
glDrawArrays	定义渲染多个图元
glDrawBuffer	定义选择哪个色彩缓冲区被绘制
glDrawElements	渲染数组数据中的图元
glDrawPixels	将一组像素写入帧缓冲区
glEdgeFlag	定义一个边缘标志数组
glEdgeFlagPointer	定义一个边缘标志数组
glEnable, glDisable	打开或关闭 OpenGL 的特殊功能
glEnableClientState,glDisableClientState	分别打开或关闭数组
glEvalCoord	求解一维和二维贴图
glEvalMesh1,glEvalMesh2	求解一维和二维点或线的网格
glEvalPoint1,glEvalPoint2	生成及求解一个网格中的单点
glFeedbackBuffer	控制反馈模式
glFinish	等待直到 OpenGL 执行结束
glFlush	在有限的时间里强制 OpenGL 的执行
glFogf,glFogi,glFogfv,glFogiv	定义雾参数
glFrontFace	定义多边形的前面和背面
glFrustum	当前矩阵乘上透视矩阵
glGenLists	生成一组空的连续的显示列表
glGenTextures	生成纹理名称
glGetBooleanv,glGetDoublev,glGetFloatv,glGetIntegerv	返回值或所选参数值

glGetClipPlane 返回特定裁减面的系数
glGetColorTableEXT 从当前目标纹理调色板得到颜色表数据
glGetColorTableParameterfvEXT,glGetColorTableParameterivEXT 从颜色表中得到调色板参数
glGetError 返回错误消息
glGetLightfv,glGetLightiv 返回光源参数值
glGetMapdv,glGetMapfv,glGetMapiv 返回求值程序参数
glGetMaterialfv,glGetMaterialiv 返回材质参数
glGetPixelMapfv,glGetpixelMapuiv,glGetpixelMapusv 返回特定的像素图
glGetPointerv 返回顶点数据数组的地址
glGetPolygonStipple 返回多边形的点图案
glGetString 返回描述当前 OpenGl 连接的字符串
glGetTexEnvfv 返回纹理环境参数
glGetTexGendv,glGetTexGenfv,glGetTexGeniv 返回纹理坐标生成参数
glGetTexImage 返回一个纹理图象
glGetTexLevelParameterfv,glGetTexLevelParameteriv 返回特定的纹理参数的细节级别
glGetTexParameterfv,glGetTexParameteriv 返回纹理参数值
glHint 定义实现特殊的线索
glIndex 建立当前的色彩索引
glIndexMask 控制写色彩索引缓冲区里的单独位
GlIndexPointer 定义一个颜色索引数组
glInitName 初始化名字堆栈
glInterleavedArrays 同时定义和允许几个在一个大的数组集合里的交替数组
glIsEnabled 定义性能是否被允许
glIsList 检测显示列表的存在
glIsTexture 确定一个名字对应一个纹理
glLightf,glLighti,glLightfv,glLightiv 设置光源参数
glLightModelf,glLightModeli,glLightModelfv,glLightModeliv 设置光线模型参数
glLineStipple 设定线点绘图案
glLineWidth 设定光栅线段的宽
glListBase 为 glcallList 设定显示列表的基础
glLoadIdentity 用恒等矩阵替换当前矩阵
glLoadMatrixd,glLoadMatrif 用一个任意矩阵替换当前矩阵
glLoadName 将一个名字调入名字堆栈
glLogicOp 为色彩索引渲染定义一个逻辑像素操作
glMap1d,glMap1f 定义一个一维求值程序
glMap2d,glMap2f 定义一个二维求值程序
glMapGrid1d,glMapGrid1f,glMapgrid2d,glMapGrid2f 定义一个一维或二维网格
glMaterialf,glMateriali,glMateriafv,glMaterialiv 为光照模型定义材质参数
glMatrixMode 定义哪一个矩阵是当前矩阵
glMultMatrixd,glMultMatrixf 用当前矩阵与任意矩阵相乘
glNewList,glEndList 创建或替换一个显示列表
glNormal 设定当前顶点法向
glNormalPointer 设定一个法向数组
glOrtho 用垂直矩阵与当前矩阵相乘
glPassThrough 在反馈缓冲区做记号
glPixelMapfv,glPixelMapuiv,glPixelMapusv 设定像素交换图
glPixelStoref,glpixelStorei 设定像素存储模式
glPixelTransferf,glPixelTransferi 设定像素存储模式
glPixelZoom 设定像素缩放因数
glPointSize 设定光栅点的直径
glPolygonMode 选择一个多边形的光栅模式
glPolygonOffset 设定 OpenGL 用于计算深度值的比例和单元
glPolygonStipple 设定多边形填充图案
glPrioritizeTextures 设定纹理固定的优先级
glPushAttrib,glPopAttrib 属性堆栈的压入和弹出操作
glPushClientAttrib,glPopClientAttrib 在客户属性堆栈存储和恢复客户状态值
glPushmatrix,glPopMatrix 矩阵堆栈的压入和弹出操作

glPushName,glPopName 名字堆栈的压入和弹出操作
glRasterPos 定义像素操作的光栅位置
glreadBuffer 为像素选择一个源色彩缓冲区
glReadPixels 从帧缓冲区读取一组数据
glRectd,glRectf,glRecti,glRects,glRectdv,glRectfv,glRectiv,glRectsv 绘制一个三角形
glRenderMode 定义光栅模式
glRotated,glRotatef 将旋转矩阵与当前矩阵相乘
glScaled,glScalef 将一般的比例矩阵与当前矩阵相乘
glScissor 定义裁减框
glSelectBuffer 为选择模式值建立一个缓冲区
glShadeModel 选择平直或平滑着色
glStencilFunc 为模板测试设置功能和参照值
glStencilMask 控制在模板面写单独的位
glStencilOp 设置激活模式测试
glTexCoord 设置当前纹理坐标
glTexCoordPointer 定义一个纹理坐标数组
glTexEnvf,glTexEnvi,glTexEnvfv,glTexEnviv 设定纹理坐标环境参数
glTexGend,glTexgenf,glTexGendv,glTexGenfv,glTexGeniv 控制纹理坐标的生成
glTexImage1D 定义一个一维的纹理图象
glTexImage2D 定义一个二维的纹理图
glTexParameterf,glTexParameteri,glTexParameterfv,glTexParameteriv 设置纹理参数
glTexSubImage1D 定义一个存在的一维纹理图像的一部分,但不能定义新的纹理
glTexSubImage2D 定义一个存在的二维纹理图像的一部分,但不能定义新的纹理
glTranslated,glTranslatef 将变换矩阵与当前矩阵相乘
glVertex 定义一个顶点
glVertexPointer 设定一个顶点数据数组
glViewport 设置视窗

[OpenGL 应用函数库]

gluBeginCurve,gluEndCurve 定义一条不一至的有理的 NURBS 曲线
gluBeginPolygon,gluEndPolygon 定义一个非凸多边形
gluBeginSurface,gluEndSurface 定义一个 NURBS 曲线
gluBeginTrim,gluEndTrim 定义一个 NURBS 整理循环
gluBuild1Dmipmaps 建立一维多重映射
gluBuild2Dmipmaps 建立二维多重映射
gluCylinder 绘制一个圆柱
gluDeleteNurbsRenderer 删除一个 NURBS 对象
gluDeleQuadric 删除一个二次曲面对象
gluDeleteTess 删除一个镶嵌对象
gluDisk 绘制一个盘子
gluErrorString 根据 OpenGL 或 GLU 错误代码产生错误字符串
gluGetNutbsProperty 得到一个 NURBS 属性
gluGetString 得到一个描述 GLU 版本号或支持 GLU 扩展调用的字符串
gluGetTessProperty 得到一个镶嵌对象
gluLoadSamplingMatrices 加载 NUMRBS 例子和精选矩阵
gluLookAt 设定一个变换视点
gluNewNurbsRenderer 创建一个 NURBS 对象
gluNewQuadric 建立一个二次曲面对象
gluNewTess 建立一个镶嵌对象
gluNextContour 为其他轮廓的开始做标记
gluNurbsCallback 为 NURBS 对象设定一个回调
gluNnrbsCurve 设定一个 NuRBS 曲线的形状
gluNurbsProperty 设定一个 NURBS 属性
gluNurbsSurface 定义一个 NURBS 表面的形状
gluOrtho2D 定义一个二位正交投影矩阵
gluPartialDisk 绘制一个盘子的弧

gluPerspective 设置一个透视投影矩阵
gluPickMatrix 定义一个拾取区间
gluProject 将对象坐标映射为窗口坐标
gluPwlCurve 描述一个分段线性 NURBS 修剪曲线
gluQuadricCallback 为二次曲面对象定义一个回调
gluQuadricDrawStyle 为二次曲面设定合适的绘制风格
gluQuadricNormals 定义二次曲面所用的法向的种类
gluQuadricOrientation 定义二次曲面内部或外部方向
gluQuadricTexture 定义是否带二次其面做纹理帖图
gluScaleImage 将图象变换为任意尺寸
gluSphere 绘制一个球体
gluTessBeginContour,gluTessEndContour 划定一个边界描述
gluTessBeginPolygon,gluTessEndPolygon 划定一个多边形描述
gluTessCallback 为镶嵌对象定义一个回调
gluTessNormal 为一个多边行形定义法向
gluTessProperty 设置镶嵌对象的属性
gluTessVertex 定义在一个多边形上的顶点
gluUnProject 将窗口坐标映射为对象坐标

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
1. 2 glColor--设置当前颜色
1. 3 glColorPointer--定义颜色数组
1. 4 gllndex--设置当前颜色索引
1. 5 gllndexPointer--定义颜色索引数组
1. 6 glCOforTableEXT--为目标调色板纹理指定调色板的格式和大小
1. 7 glColorsubTableEXT--指定需要替代的目标纹理调色板的一部分
第二章 绘製几何图原及物体
2. 1 glVertex--指定顶点
2. 2 glVertexPointer--定义顶点数据数组
2. 3 glArrayElement--指定用来绘製顶点的数组元素
2. 4 glBegin, glEnd--限定一个或多个图原顶点的绘製
2. 5 glEdgeFlag, glEdgeFlagy--指定边界标记
2. 6 glPointsize--指定光栅化点的直径
2. 7 glLinewidth--指定光栅化直线的宽度
2. 8 glLinestipple--指定点划线
2. 9 glPolygonMode--选择多边形光栅化模式
2. 10 glFrontFace--定义正面多边形和反反面多边形
2. 11 glPolygonstipple--设置多边形点划图
2. 12 glDrawElements--从数组数据绘製图原
2. 13 glRect--绘製矩形
第三章 坐标转换
3. 1 glTranslate--用平移矩阵乘以当前矩阵
3. 2 glRotate--用旋转矩阵乘以当前矩阵
3. 3 glscale--用缩放矩阵乘以当前矩阵
3. 4 glViewport--设置机口
3. 5 glFrustum--用透视矩阵乘以当前矩阵
3. 6 glorthO--用正视矩阵乘以当前矩阵
3. 7 glClipPlane--指定切割几何物体的平面

第四章 堆栈操作
4. 1 glLoadMatrix--用任意矩阵替换当前矩阵
4. 2 glMultMatrix--用任意矩阵乘以当前矩阵
4. 3 glMatrixMode--指定哪一个矩阵是当前矩阵
4. 4 glPushMatrix, glPopMatrix--压人和弹出当前矩阵堆栈
4. 5 glPushAttrib, glPopAttrib--压人和弹出属性堆栈
4. 6 glPushClientAttrib, glPopClientAttrib--在客户属性堆栈中保存和恢复客户状态变量组
4. 7 glPushName, gPopName--压人和弹出名称堆栈
4. 8 gllnitNames--初始名称堆栈
4, 9 glLoadName--向名称堆栈中装载名称
第五章 显示列表
5. 1 glNewList, glEndList--创建或替换一个显示列表
5. 2 glCallLISt--执行一个显示列表
5. 3 glCallLISts--执行一列显示列表
5. 4 glGenLists--生成一组空的相邻的显示列表
5. 5 glDeleteLists--删除一组相邻的显示列表
5. 6 gllSLISt--检验显示列表的存在
第六章 使用光照和材质
6. 1 glNormal--设置当前的法向量
6. 2 glNormalPointer--定义法向量数组
6. 3 glLight--设置光源参数
6. 4 glLightModel--设置光照模型参数
6. 5 glMaterial--为光照模型指定材质参数
6. 6 glColorMateria--使材质颜色跟踪当前颜色
第七章 像素操作
7. 1 glRasterPos--为像素操作指定光栅位置
7. 2 glBitmap--绘製位图
7. 3 glReadPixels--从帧缓存中读取一块像素
7. 4 glDrawPixels--将一个像素块写人帧缓存
7. 5 glCopyPixels--在帧缓存中拷贝像素
7. 6 glCopyTexlmage1D--将像素从帧缓存拷贝到一维纹理图像中
7. 7 glCopyTexlmageZD--把像素从帧缓存拷贝到二维纹理图像中
7. 8 glCopyTexsublmage1D--从帧缓存中拷贝一维纹理图像的子图像
7. 9 glCopyTexsublmageZD--从帧缓存中拷贝二维纹理图像的子图像
7. 10 glPixelZoom--指定像素缩放因子
7. 11 glPixelstore--设置像素存储模式
7. 12 glPixelTransfer--设置像素传输模式
7. 13 glPixelMap--设置像素传输映射表
第八章 纹理映射
8. 1 glTexlmage1D--指定一维纹理图像
8. 2 glTexlmageZD--指定二维纹理映射
8. 3 glTexParameter--设置纹理参数
8. 4 glTexsublmage1D--指定已存在的一维纹理图像的一部分
8. 5 glTexsublmageZD--指定已存在的二维纹理图像的一部分
8. 6 glTexEnv--设置纹理环境参数
8. 7 glTexCoord--设置当前纹理坐标
8. 8 glTexGen--控制纹理坐标的生成
8. 9 glTexCoordPointer--定义纹理坐标数组
8. 10 glDeleteTextures--删除命名的纹理
第九章 特殊效果操作
9. 1 glBlendFunc--指定像素的数学算法
9. 2 glHint--指定由实现确定的控制行为
9. 3 glFOg--指定雾化参数
第十章 帧缓存操作
10. 1 glClear--将缓存清除为预先的设置值
10. 2 glClearAccum--设置累加缓存的清除值
10. 3 glClearColor--设置颜色缓存的清除值

10. 4 glClearDepth--设置深度缓存的清除值
10. 5 glClearIndex--设置颜色索引缓存的清除值
10. 6 glClearstencil--设置模板缓存的清除值
10. 7 glDrawBuffer--指定绘製的颜色缓存
10. 8 glIndexMask--控制颜色索引缓存中单个位的写操作
10. 9 glColorMask--激活或关闭帧缓存颜色分量的写操作
10. 10 glDepthMask--激活或关闭对深度缓存的写操作
10. 11 glstencilMask--控制模板平面中单个位的写操作
10. 12 glAlphaFunc--一指定 alpha 检验函数
10. 13 glstencilFunc--设置模板检验函数和参考值
10. 14 glstencilop--设置模板检验操作
10. 15 glDepthFunc--指定深度比较中使用的数值
10. 16 glDepthRange--指定从单位化的设备坐标到窗口坐标的 z 值映射
10. 17 glLOgiCOp--为颜色索引绘製指定逻辑像素操作
10. 18 glACCum--对累加缓存进行操作
第十一章 绘製曲线和曲面
11. l glEvalCoord--求取激活的一维和二维纹理图
11. 2 glMapl--定义一维求值器
11. 3 glMapZ--定义二维求值器
11. 4 glMapGrid--定义一维或二维网格
11. 5 glEvalMesh--计算一维或二维点网格或线网格
11. 6 glEvalPoint--生成并求取网格中的单个点
第十二章 查询函数
12. l glGet--返回所选择的参数值
12. 2 glGetClipPlane--返回指定的切平面係数
12. 3 glGetColorTableEXT--获得当前目标纹理调色板的颜色表数据
12. 4 glGetColorTableParameterfvEXT, glGetColorTableParameterlvEXT-从颜色表中获得调色板参数
12. 5 glGetError--返回错误信息
12. 6 glGetLight--返回光源参数值
12. 7 glGetMap--返回求值器参数
12. 8 glGetMaterial--返回材质参数
12. 9 glGetPixelMap--返回指定的像素映像
12. 10 glGetPointery--返回顶点数据数组地 J
12. 11 glGetPolygonstipple--返回多边形点戈
12. 12 glGetstring--返回描述当前 OpenGL
12. 13 glGetTexEnv--返回纹理环境参数
12. 14 glGetTexGen--返回纹理坐标生成参数
12. 15 glGetTexlmage--返回纹理图像
12. 16 glGetTexLevelParameter--返回指定细节水平的纹理参数值
12. 17 glGetTexParameter--返回纹理参数值
第二篇 GLU 库函数
第一章 绘製 NURBS 曲线和曲面
1. l gluNewNurbsRenderer--创建一个 NURBS 对像
1. 2 gluNurbsProperty--设置 NURBS 属性
1. 3 gluNurbsCallback--为 NURBS 对像定义回调函数
1. 4 gluBeginCurve, gluEndCurve--限定 NURBS 曲线的定义
1. 5 gluNurbsCurve--定义 NURBS 曲线的形状
1. 6 gluDeleteNurbsRenderer--删除 NURBS 对像
1. 7 gluBeglnsurface, gluEndsurface--限定 NURBS 曲面的定义
1. 8 ghiNurbssurface--定义 NURBS 曲面的形状
1. 9 gluBeginTrim, gluEndTrim--限定 NURBS 裁剪环的定义
1. 10 gluPwlCurve--描述分段线性 NURBS 裁剪曲线
1. ll gfuBeglnPolygon, gluEndPolygon--限定多边形的定义
1. 12 gluPickMatrix--定义拾取区域
第二章 绘製二次几何物体
2. l gluNewQuadric--创建一个二次对象
2. 2 gluQuadricDrawsope--指定二次对象的绘製方式

2. 13 ghitGetwindow--获得当前窗口的标识符
2. 14 glutPopwindow--改变当前窗口的位置，使其前移
2. 15 ghitPtshwilld00--改变当前窗口的位置，使其后移
2. 16 glutDestroywindow--销毁指定的窗口
2. 17 glutlconifywindow--使当前窗口图标化显示
2. 18 glutsetCursor--设置当前窗口的鼠标形状
第三章 重迭层管理
3. 1 glutEstablishoverlay--创建当前窗口的重迭层
3. 2 glutUseLayer--改变当前窗口的使用层
3. 3 glutRemoveoverlay--删除当前窗口的重迭层
3. 4 glutPostoverlayRedisplay--标记当前窗口的重迭层
需要重新绘製
3. 5 glutshowoverlay--显示当前窗口的重迭层
3. 6 glutHideoverlay--显示当前窗口的重迭层
第四章 菜单管理
4. 1 glutCreateMenu--创建一个新的弹出式菜单
4. 2 glutAddMenuEntry--在当前菜单的底部增加一个菜单条目
4. 3 glutAddsubMenu--在当前菜单的底部增加一个子菜单触发条目
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繫起来
4. 5 glutGetMenu--获取当前菜单的标识符
4. 6 glutsetMenu--设置当前菜单
4. 7 glutDestroyMenu--删除指定的菜单
4. 8 glutChangeToMenuEntry--将指定的当前菜单中的菜单项更改为菜单条目
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
4. 10 glutRemoveMenultem--删除指定的菜单项
4. 11 glutDetachMenu--释放当前窗口的一个鼠标按键
第五章 注册国调函数
5. 1 glutDispfayFunc--注册当前窗口的显示回调函数
5. 2 glutReshapeFunc--注册当前窗口的形状变化回调函数
5. 3 glutMouseFunc--注册当前窗口的鼠标回调函数
5. 4 glutMotionFunc--设置移动回调函数
5. 5 glutldleFunc--设置全局的空閒回调函数
5. 6 glutVisibilityFunc--设置当前窗口的可视回调函数
5. 7 glutKeyboardFunc--注册当前窗口的键盘回调函数
5. 8 glutspecialFunc--设置当前窗口的特定键回调函数
5. 9 glutoverlayDisplayFunc--注册当前窗口的重迭层显示回调函数
5. 10 glutPassiveMotionFunc--设置当前窗口的被动移动回调函数
5. 11 glutEntryFunc--设置当前窗口的鼠标进出回调函数
5. 12 glutspaceballMotionFunc--设置当前窗口的空间球移动回调函数
5. 13 glutspaceballRotateFunc--设置当前窗口的空间球旋转回调函数
5. 14 glutspaceballButtonFunc--设置当前窗口的空间球按键回调函数
5. 15 glutButtonBoxFunc--设置当前窗口的拨号按键盒按键回调函数
5. 16 glutDialsFunc--设置当前窗口的拨号按键盒拨号回调函数
5. 17 glutTabletMotionFunc--设置图形板移动回调函数
5. 18 glutTabletButtonFunc--设置当前窗口的图形板按键回调函数
5. 19 glutMenustatusFunc--设置全局的菜单状态回调函数
5. 20 glutTimerFunc--注册按一定时间间隔触发的定时器回调函数
第六章 颜色素引映射表管理
6. 1 glutsetColor--设置当前窗口当前层一个颜色表单元的颜色
6. 2 glutGetColor--获得指定的索引颜色
6. 3 glutCopyColormap--将逻辑颜色表从指定的窗口拷贝到当前窗口
第七章 状态检索
7. 1 glutGet--检索指定的 GLUT 状态
7. 2 glutLayerGet--检索属于当前窗口重迭层的 GLU T 状态
7. 3 glutDeviceGet--检索 GLUT 设备信息
7. 4 glutGetModifiers--返回修饰键在引起某些回调的事件发生时的状态
7. 5 glutExtensionsupported--判别当前 OpenGL 版本是否支持给定的 OpenGL 扩展

第八章 字体绘製
8. 1 glutBltmapCharcter--绘製一个位图字符
8. 2 glutBitmapwidth--返回一个位图字符的宽度
8. 3 glutstrokeCharcter--绘製一个笔划字符
8. 4 glutstrokewidth--返回一个笔划字体的宽度
第九章 几何图形绘製
9. 1 glutSolidsphere, glutwiresphere--绘製实心球体和线框球体
9. 2 glutsolidCube, glutwireCube--绘製实心立方体和线框立方体
9. 3 glutsolidCone, glutwireCone--绘製实心圆锥体和线框圆锥体
9. 4 glutsolidTorus, glutwireTorus--绘製实心圆环和线框圆环
9. 5 glutSolidDOdeCahedroll, glLltwiFeDOdechedfotl--绘製实心
十二面体和线框十二面体
9. 6 glutSolidOctahedron, glutWireOctahedron--绘製买心八面体和线框八面体
9. 7 glutsolldTetrahedron, glutwireTetrahedron--绘製实心四面体和线框四面体
9. 8 glutSollelcosahedron, glutwirelcosahedron--绘製实心二十面体和线框二十面体
9. 9 glutsolidTeapot, glutwireTeapot--绘製实心茶壶和线框茶壶

每组三维物体包括两种形式：网状体（wire）和实心体（solid)。网状体没有平面法向，而实心体有，能进行光影计算，有光照时采用实心体模型。下面这些函数的参数都是定义物体大小的，可以改变。

功能 函数

绘制球
void auxWireSphere(GLdouble radius)
void auxSolidSphere(GLdouble radius)
绘制立方体
void auxWireCube(GLdouble size)
void auxSolidCube(GLdouble size)
绘制长方体
void auxWireBox(GLdouble width,GLdouble height,GLdouble depth)
void auxSolidBox(GLdouble width,GLdouble height,GLdouble depth)
绘制环形圆纹面
void auxWireTorus(GLdouble innerRadius,GLdouble outerRadius)
void auxSolidTorus(GLdouble innerRadius,GLdouble outerRadius)
绘制圆柱
void auxWireCylinder(GLdouble radius,GLdouble height)
void auxSolidCylinder(GLdouble radius,GLdouble height)
绘制二十面体
void auxWireIcosahedron(GLdouble radius)
void auxSolidIcosahedron(GLdouble radius)
绘制八面体
void auxWireOctahedron(GLdouble radius)
void auxSolidOctahedron(GLdouble radius)
绘制四面体
void auxWireTetrahedron(GLdouble radius)
void auxSolidTetrahedron(GLdouble radius)
绘制十二面体
void auxWireDodecahedron(GLdouble radius)
void auxSolidDodecahedron(GLdouble radius)

绘制圆锥
void auxWireCone(GLdouble radius,GLdouble height)
void auxSolidCone(GLdouble radius,GLdouble height)
绘制茶壶
void auxWireTeapot(GLdouble size)
void aucSolidTeapot(GLdouble size)

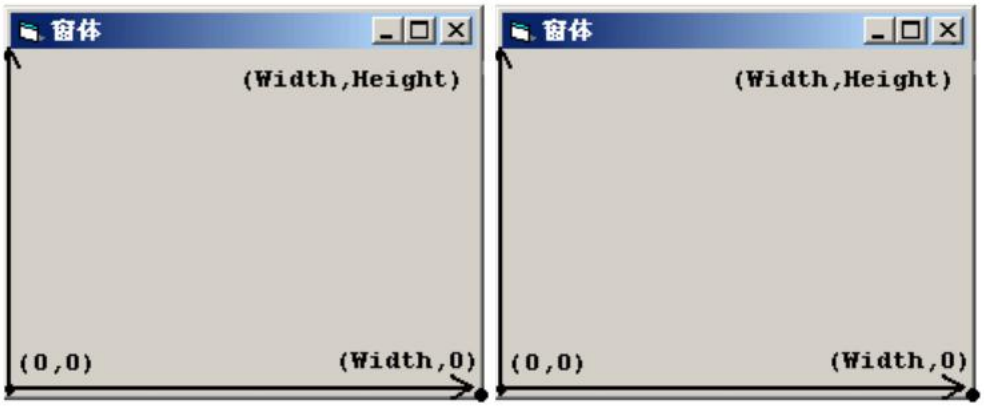
以上物体均以各自中心为原点绘制，所有坐标都已单位化，可以缩放。

glViewport()函数

在 OpenGL 初始化完成之后，我们应该进行一些视图设置。首先是设定视见区域，即告诉 OpenGL 应把渲染之后的图形绘制在窗体的哪个部位。当视见区域是整个窗体时，OpenGL 将把渲染结果绘制到整个窗口。我们调用 glViewPort 函数来决定视见区域:

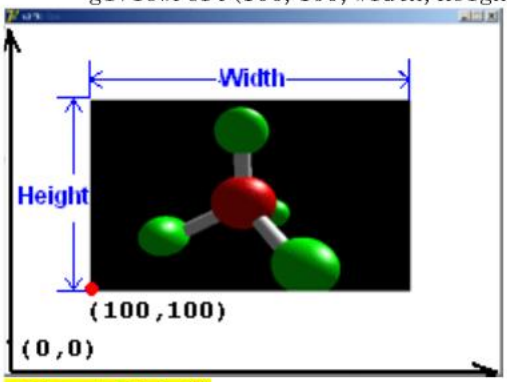
```
procedure glViewPort(x:GLInt;y:GLInt;Width:GLSizei;Height:GLSizei);
```

其中，参数 X，Y 指定了视见区域的左下角在窗口中的位置，一般情况下为（0，0），Width 和 Height 指定了视见区域的宽度和高度。注意 OpenGL 使用的窗口坐标和 WindowsGDI 使用的窗口坐标是不一样的。图 3.1-1 表示了在 WindowsGDI 中的窗口坐标，而图 3.1-2 则是 OpenGL 所定义的窗口坐标。



例如，要设置如图 3.1-3 中的视见区域，我们应该调用函数:

```
glViewPort(100,100,Width,Height);
```



glMatrixMode()

指定哪一个矩阵是当前矩阵
C 语言描述 void glMatrixMode(GLenum mode)
参数 mode 指定哪一个矩阵堆栈是下一个矩阵操作的目标，可选值:GL_MODELVIEW、GL_PROJECTION、GL_TEXTURE.
说明
glMatrixMode 设置当前矩阵模式:
GL_MODEVIEW，对模型视景矩阵堆栈应用随后的矩阵操作。
GL_PROJECTION，对投影矩阵应用随后的矩阵操作。
GLTEXTURE，对纹理矩阵堆栈应用随后的矩阵操作。

glLoadIdentity()

该函数的功能是重置当前指定的矩阵为单位矩阵。在 glLoadIdentity()之后我们为场景设置了透视图。
glMatrixMode(GL_MODELVIEW)设置当前矩阵为模型视图矩阵，模弄视图矩阵储存了有关物体的信。

gluOrtho2D()

gluOrtho2D(-5.0,5.0,-5.0,5.0);
参数分别代表（左下角 x 坐标，右上角 x 坐标，左下角 y 坐标，右上角 y 坐标）——坐标全相对于窗口左下角－－原点），near 和 far 默认为-1 和 1，此函 数决定一个平行六面体，即 View Volume！ View Volume 越大，里面的物体显得越小！如，一个点的坐标是(0,0,0)就是在平行六面体的中间，也就是在 viewport 的中间;又如一个点的坐标是(-5.0,-5.0,0),是在平行六面体的左下角，也就是在 vieport 的左下角。
注：viewport 是指视窗的大小，就好像我们的眼睛！viweport 可以用下面函数指定：glViewport(左下角 x 坐标, 左下角 y 坐标, wigth, height)； ，默认是（0，0，窗口的宽度，窗口的高度）可以用 gluOrtho 函数来指定 near 和 far，gluOrtho（Xmin,Xmax,Ymin,Ymax,near,far);

glTranslatef()

glTranslatef 的作用就是移动坐标原点。对应的 3 个参数对应着 3 个坐标轴。如果你调用一次 glTranslatef(1.0f,0.0f,0.0f)然后画一个小球，接着再调用次 glTranslatef(0.0f,1.0f,0.0f)再画一个小球。此时，两个小球中，一个在另外一个正右方。所以，如果要使两个小球分别处于 x、y 轴，则需要在第二次画之前调用 glLoadIdentity()函数，使坐标原点归位。另外，此处的坐标系为右手坐标系。

glPushMatrix() glPopMatrix()

相当于栈里的入栈和出栈。
许多人不明白的可能是入的是什么，出的又是什么。我也是自己反复做了下测试才懂的.例如你当前的坐标系原点在你电脑屏幕的左上方。现在你调用 glPushMatrix，然后再调用一堆平移、旋转代码等等，然后再画图。那些平移和旋转都是基于坐上角为原点进行变化的。而且都会改变坐标的位置，经过了这些变化后，你的坐标肯定不再左上角了。那如果想恢复怎么办呢？这时就调用 glPopMatrix 从栈里取出一个“状态”了，这个状态就是你调用 glPushMatrix 之前的那个状态。就如很多 opengl 的书上所讲：调用 glPushMatrix 其实就是把当前状态做一个副本放入堆栈之中。

glRasterPos2i()

glRasterPos2i(200, 200); 改变光栅位置
光栅（Raster）：由像素构成的一个矩形网格。要在光栅上显示的数据保存于帧缓存内。

glBitmap()

当设置了光栅位置后，就可以调用 glBitmap()函数来显示位图数据了。这个函数形式为：

```
void glBitmap( GLsizei width,GLsizei height,GLfloat xbo,GLfloat ybo,GLfloat xbi,GLfloat ybi,const GLubyte *bitmap);
```

显示由 bitmap 指定的位图，bitmap 是一个指向位图的指针。位图的原点放在 最近定义的当前光栅位置上。若当前光栅位置是无效的，则不显示此位图或其一部分，而且当前光栅位置仍然无效。参数 width 和 height 一象素为单位说 明位图的宽行高。宽度不一定是 8 的倍数。参数 xbo 和 ybo 定义位图的原点（正值时，原点向上移动；负值时，原点向下移动）。参数 xbi 和 ybi 之处在位 图光栅化后光栅位置的增量。

glReadPixels()

读取象素数据,

```
void glReadPixels(GLint x,GLint y,GLsizesi width,GLsizei height,
GLenum format,GLenum type,GLvoid *pixel);
```

函数参数(x,y)定义图像区域左下角点的坐标，width 和 height 分别是图像的高度和宽度，*pixel 是一个指针，指向存储图像数据的数组。参数 format 指出所读象素数据元素的格式（索引值或 R、G、B、A 值，如表 11-1 所示），而参数 type 指出每个元素的数据类型（见表 11-2）。

名称	象素数据类型
GL_INDEX	单个颜色索引
GL_RGB	先是红色分量，再是绿色分量，然后是蓝色分量
GL_RED	单个红色分量
GL_GREEN	单个绿色分量
GL_BLUE	单个蓝色分量
GL_ALPHA	单个 Alpha 值
GL_LUMINANCE_ALPHA	先是亮度分量，然后是 Alpha 值
GL_STENCIL_INDEX	单个的模板索引
GL_DEPTH_COMPONENT	单个深度分量

表 11-1 函数 glReadPixels()及 glDrawPixels()的象素格式

名称	数据类型

GL_UNSIGNED_BYTE　　无符号的 8 位整数
GL_BYTE　　8 位整数
GL_BITMAP　　无符号的 8 位整数数组中的单个数位
GL_UNSIGNED_SHORT　　无符号的 16 位整数
GL_SHORT　　16 位整数
GL_UNSIGNED_INT　　无符号的 32 位整数
GL_INT　　32 位整数
GL_FLOAT　　单精度浮点数
表 11-2 函数 glReadPixels()及 glDrawPixels()的象素数据类型

glDrawPixels()

写入象素数据 void glDrawPixels(GLsizesi width,GLsizei height,GLenum format,
GLenum type,GLvoid *pixel);
函数参数 format 和 type 与 glReadPixels()有相同的意义，pixel 指向的数组包含所要画的象素数据。
注意，调用这个函数前必须先设置当前光栅位置，若当前光栅位置无效，则给出该函数时不画任何图形，
并且当前光栅位置仍然保持无效。

glCopyPixels()

象素拷贝函数是： void glCopyPixels(GLint x,GLint y,GLsizesi width,GLsizei height, GLenum type);
这个函数使用起来有点类似于先调用 glReadPixels()函数后再调用 glDrawPixels()一样，但它不需要
将数据写到内存中去，因它只将数据写到 framebuffer 里。函数功能就是拷贝 framebuffer 中左下角点
在(x,y)尺寸为 width、height 的矩形区域象素数据。数据拷贝到一个新的位置，其左下角点在当前光栅
的位置，参数 type 可以是 GL_COLOR、GL_STENCIL、GL_DEPTH。在拷贝过程中，参数 type 要按如下方式
转换成 format：
　　1）若 type 为 GL_DEPTH 或 GL_STENCIL，那么 format 应分别是 GL_DEPTH_COMPONENT 或 GL_STENCIL_INDEX；
2）若 type 为 GL_COLOR，format 则用 GL_RGB 或 GL_COLOR_INDEX，这要依赖于图形系统是处于 RGBA 方式还是处
于颜色表方式。

glPixelZoom()

一般情况下，图像的一个象素写到屏幕上时也是一个象素，但是有时也需要将图像放大或缩小，OpenGL 提供了
这个函数：
图像缩放 void glPixelZoom(GLfloat zoomx,GLfloat zoomy);
设置象素写操作沿 X 和 Y 方向的放大或缩小因子。缺省情况下，zoomx、zoomy 都是 1.0。如果它们都是 2.0，
则每个图像象素被画到 4 个屏幕象素上面。
注意：小数形式的缩放因子和负数因子都是可以的。

glutSwapBuffers()

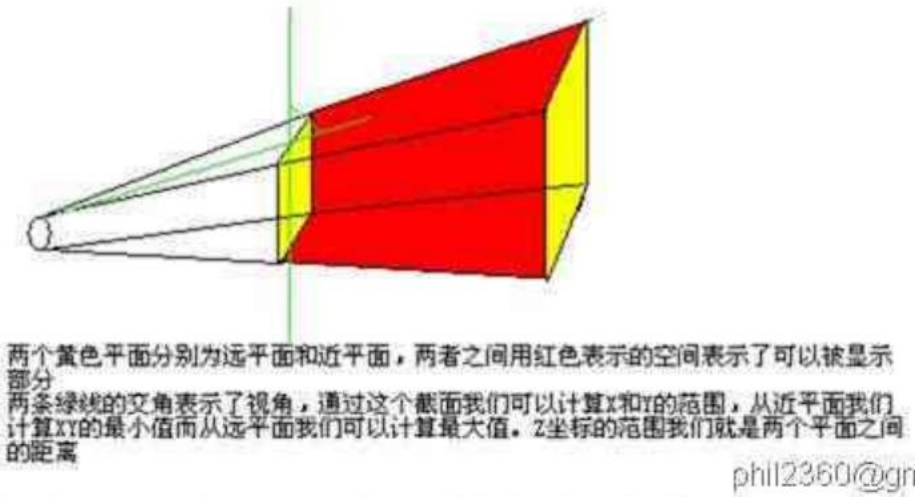
交换缓冲显存
在双缓冲模式下，游戏运行时占用 3 个图象内存区域。
其中一个是“窗口内存”。其他两个是“缓冲显存 1”和“缓冲显存 2”。
用 glBegin()进行渲染后用 glutSwapBuffers，显卡负责把“缓冲显存 1”的画面复制到“窗口内存”中。
然后再用 glBegin()进行渲染，画面就先渲染到“缓冲显存 2”上面了。这时候显卡还在忙着把“缓冲显存 1”
上的内容拷贝到“窗口内存”中呢。
所以“渲染过程”和“显示过程”两不误！

glutPostRedisplay()

如果是单线程，仅仅用 postredisplay 而不返回是走不到 display 的。
应该把这个循环拆开，循环体放在 idle 里。

glFrustum()

glFrustum(left,right,bottom,top,near,far);　参数用来确定视角边界的各个点。



我们的目的就是要通过 glFrustum 的调用来确定红色区域。首先，图片左侧的圆点就是坐标原点(0,0,0)，黄色的近平面的左下角坐标（从坐标原点往黄色平面看）用(left, bottom, -near)来指定，而近平面的右上角坐标则用(right, top,-near)来指定。为了确定红色区域，我们还缺少远平面的坐标点。实际上我们只需要知道远平面的 Z 坐标 far 就可以确定这个平面了。因为远平面的四个角的坐标可以用原点和近平面四个角的射线和远平面相交来确定。因此我们只需要设定(left,bottom, -near)、(right, top,-near)以及-far 等值就可以确定红色的可视区域了，这也就构成了 glFrustum 函数的几个参数。

glBegin()

OpenGL 通过将物体抽象为笛卡尔坐标系下点、线段、多边形的集合，再将点、线段、多边形等通过在函数 glBegin()与 glEnd()之间的一系列顶点数据，绘制出图形还原物体。OpenGL 通过 glBegin()与 glEnd()函数完成点、线、三角
形、四边形及多边形的绘制
glBegin(GLenum)
数 GLenum 有以下 10 个参数：GL_POINTS：表示将要绘制点
GL_LINES：表示函数开始绘制线
GL_LINE_STRIP：表示函数将开始绘制折线
GL_LINE_LOOP：表示函数将开始绘制闭合曲线
GL_TRIANLES：表示函数开始绘制三角形
GL_TRIANLE_STRIP：表示函数将开始绘制三角形片
GL_TRIANLE_FAN：表示函数将开始绘制三角形扇
GL_QUADS：表示函数开始绘制四边形
GL_QUAD_STRIP：表示函数开始绘制多边形片
GL_POLYGON：表示函数绘制多边形

glRotatef()

物体围绕指定向量旋转指定角度
glRotatef(GLfloat angle,GLfloat x,GLfloat y,GLfloat z)
angle 毋庸置疑就是旋转的角度，而 x,y,z 三个参数则应该看成一个整体表示一个向量，表示物体围绕向量[x,y,z]旋转。