OpenGL学习笔记

**Date :2017.12.9**

***Author： 吕宏宇***

**Void GlutIdleFunc（void （\*func）（void）；**

Func：在程序空闲的时候就会被调用的函数的函数名。

渲染(rendering)：是计算机根据模型创建图像的过程

模型(mode)： 是根据几何图元创建的，也称物体

几何图元，点，线和多边形等，他们是通过顶点指定的。

main(){

InitializeAwindowPlease(); //占位符，是特定窗口系统的函数

glClearColor(0.0,0.0,0.0,0.0);//OpenGL把窗口清除为什么颜色

glClear(GL\_COLOR\_BUFFER\_BIT);//执行清除任务

glColor3f(1.0,1.0,1.0);//确定位置物体时所使用的颜色，除非再次调用这个函数更改绘图颜色

glOrtho(0.0,1.0,0.0,1.0,-1.0,1.0);//制定Opengl在绘制最终图像时所使用的坐标系统

glBegin(GL\_POLYGON);//与glEnd()为一函数，在他们调用之间定义了要绘制的物体;

glVertex3f(0.25,0.25,0.0);

glVertex3f(0.75,0.25,0.0);

glVertex3f(0.75,0.75,0.0);

glVertex3f(0.25,0.75,0.0);

//定义了4个顶点

glEnd();

glFlush();//既保证绘图命令将实际执行，而不是存储在缓冲区等待其他的OpenGL命令

UpdateTheWindowAndCheckForEvents();//而是一个占位符函数，它管理窗口内容，并开始进行事件处理

}

函数后缀和参数类型

b 8 为整数 signed char GLbyte

s 16为整数 short GLshort

i 32为整数 int 或 long GLint,GLsizei

f 32为浮点型 float GLfloat,GLclampf

d 64为浮点型 double GLdouble,GLclampd

ub 8 位无符号整数 unsigned char GLubyte,GLboolean

us 16位无符号整数 unsigned short GLushort

ui 32位无符号整数 unsigned int 或 unsigned long GLuint,GLenum,GLbitfield

显示列表

任何数据，不论它描述的是几何图元还是像素，都可以保存在显示列表中，供当前或以后使用。也可以不保存在显示列表中，

而是立即对数据进行处理，这种模式也叫做立即模式。

求值器

所有的几何图元最终都要通过顶点来描述，参数化曲线和表面最初可能是通过控制点以及叫做基函数的多显示函数进行描述。求职器提供了一种方法，

根据控制点计算表示表面的定点，这种方法是一种多项式映射，它可以根据控制点产生表面法线、纹理坐标、颜色以及空间做标值。

基于顶点的操作

图元装配

图元装配的一个主要内容就是裁剪，它的任务是消除位于半空间之外的那部分几何图元，这个半控件是由一个平面所定义的

像素操作

就是进行像素的裁剪、变换、移动等。

纹理装配

在物体上应用纹理图像，是他们更逼真，如果徐瑶使用多幅纹理图像，把它们放在 纹理对象 中是一种明智的选择，

这样可以很方便地在他们之间进行切换。

光栅化

光栅化就是把几何数据和像素转换为片段的过程。每个片断方块对应帧缓冲区中的一个像素。把顶点连接起来形成或者计算填充多边形的内部像素时，

需要考虑直线和多边形的画点模式、直线宽度、点的大小、着色模型以及用于支持抗锯齿处理的翻盖计算。

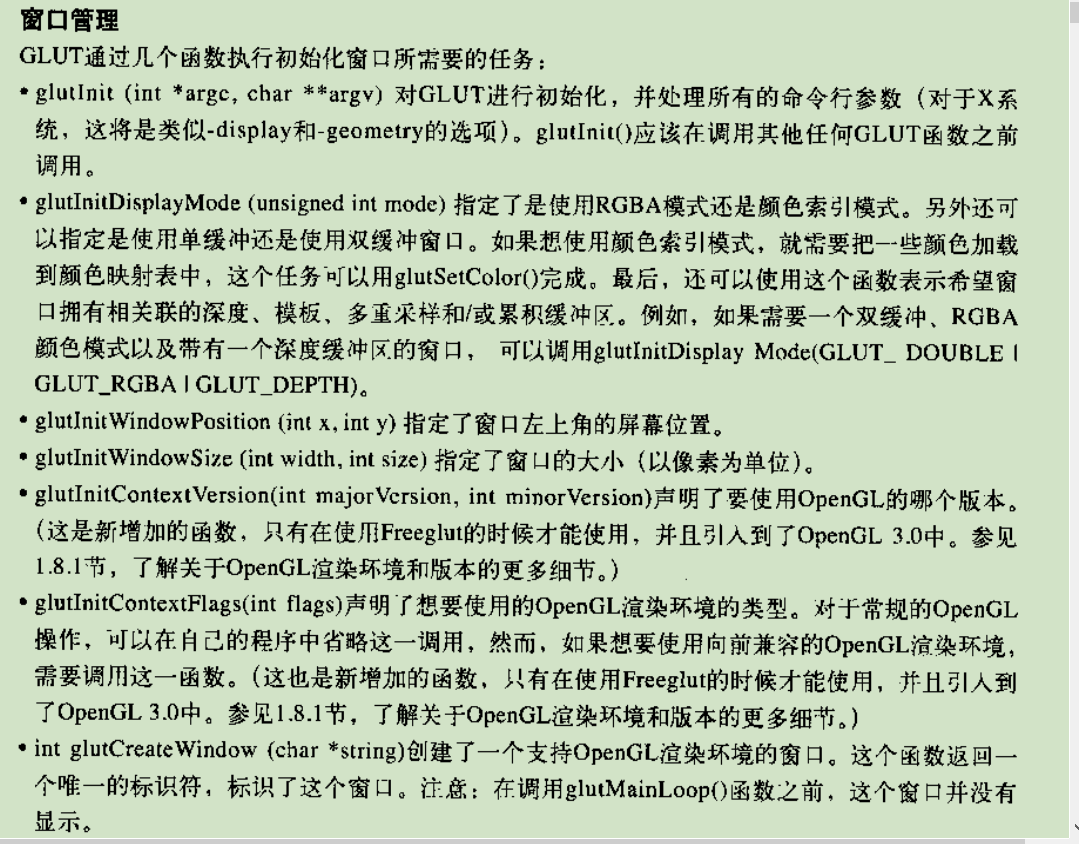
片段操作

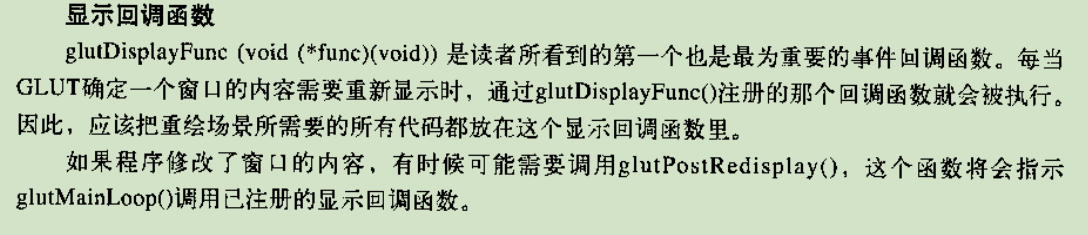
在数据实际存储到帧缓冲区之前，要执行一系列的操作。这些操作可能会修改甚至丢弃这些片断，所以有这些猜操作都可以启用或禁用。第一个可能执行的操作是纹理处理。在纹理内存中为每个片段生成一个纹理单元，并应用到这个片段上。接下来，组合主颜色和辅助颜色。

#include<GL/gl.h>

#include<GL/glu.h>

!!:Microsoft Windows 要求在gl.h/glu.h之前包含windows.h头文件。





Int main(int argc ,char \*\*argv){

glutInit(&argc,argv);

glutInitDisplayerMode(GLUT\_SINCLE|GLUT\_RGB);

glutInitWindowsSize(250,250);

glutInitWindowsPosition(100,100);

glutCreateWindow(“hello”);

init();

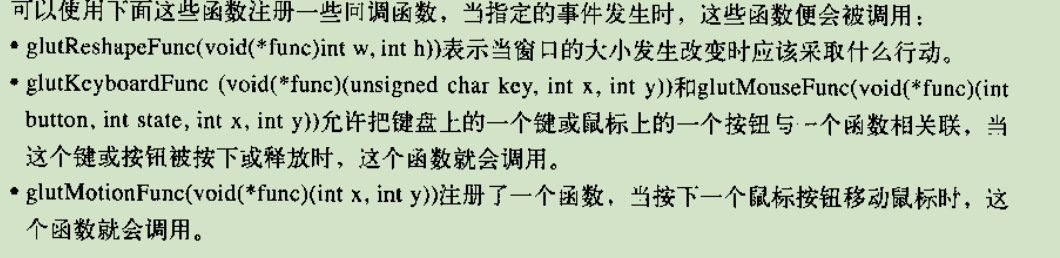
glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**重点处理输入事件：**



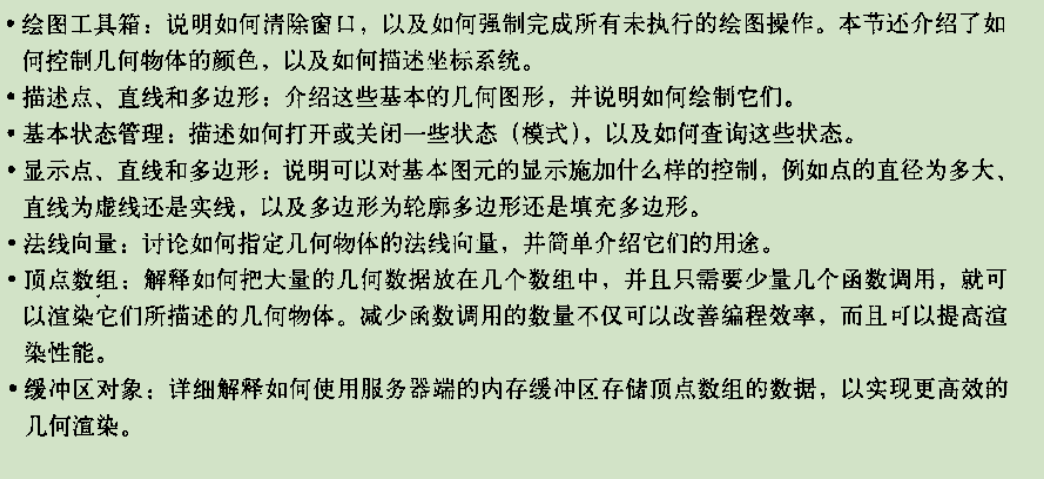
**绘制三维物体：**

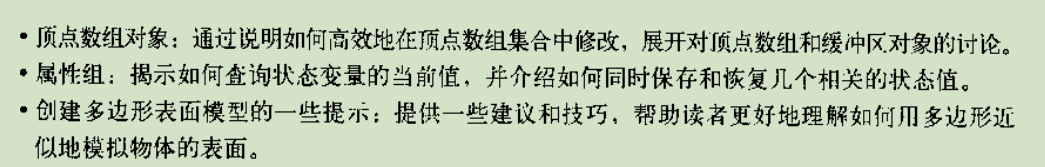
void glutWrieCube(GLdouble size); //扭曲的立方体

void glutSolidCube(GLdouble size); //固定的立方体

void glutWrieSphere(GLdouble radius,Glint slices,Glint stacks); //扭曲的圆

void glutSolidSphere(GLdouble radius,Glint slices,Glint stacks); //固定的圆





RGBA: R red ,G green B blue A alpha 透明度0-100

Void glClearColor（GLclampf red,GLclampf green,GLclampf blue,GLclampf alpha）；

设置当前清除颜色，用于清除RGBA模式下的颜色缓冲区

Void glClear(GLbitfield mask);

Mask ：

GL\_COLOR\_BUFFER\_BIT 颜色缓冲区

GL\_DEPTH\_BUFFER\_BIT 深度缓冲区

GL\_ACCUM\_BUFFER\_BIT 累积缓冲区

GL\_STENCIL\_BUFFER\_BIT 模板缓冲区

用当前的缓冲区清除值清楚制定的缓冲区。

Void glFlush（void）；

强制以前发生的OpenGL命令执行，保证他们能够在有限的时间内完成。

Void glFinish（void）；

强制以前发出的OpenGL命令完成执行。

矩形：

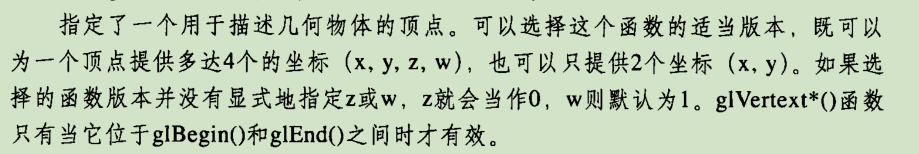
Void glRect{sifd}(TYPE x1,TYPE y1,TYPE x2,TYPE y2);

Void glRect{sifd}(const TYPE \*v1,const TYPE \*v2);

制定顶点：

Void glVertex[234]{sifd}(TYPE coords);

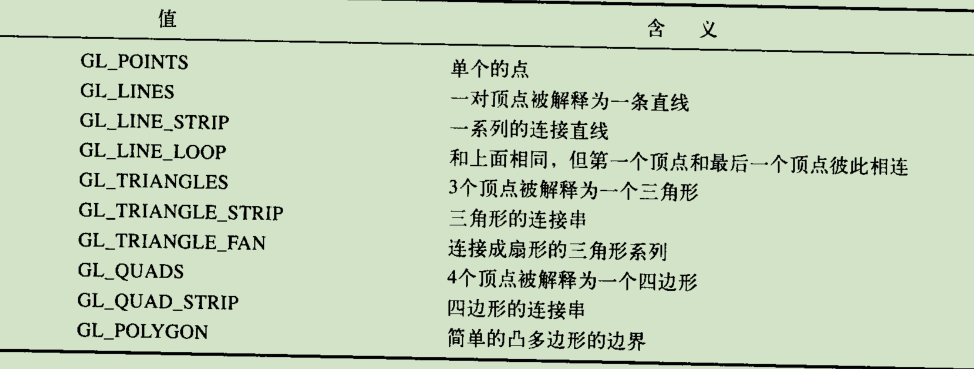
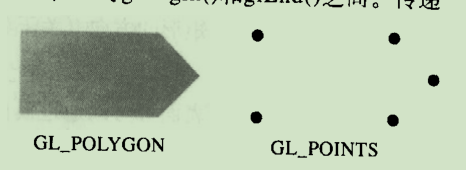
Void glVertex[234]{sifd}(const TYPE \*coords);



Void glBegin(GLenum mode);

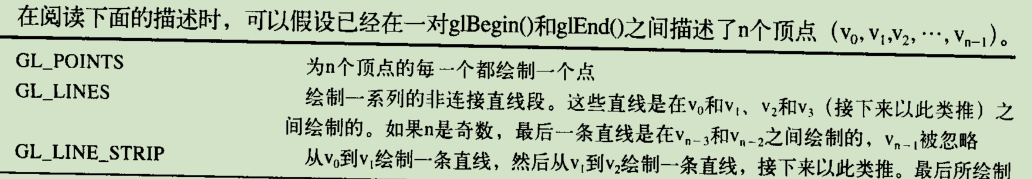
标志着一个顶点数据列表的开始，它描述了一个几何图元。

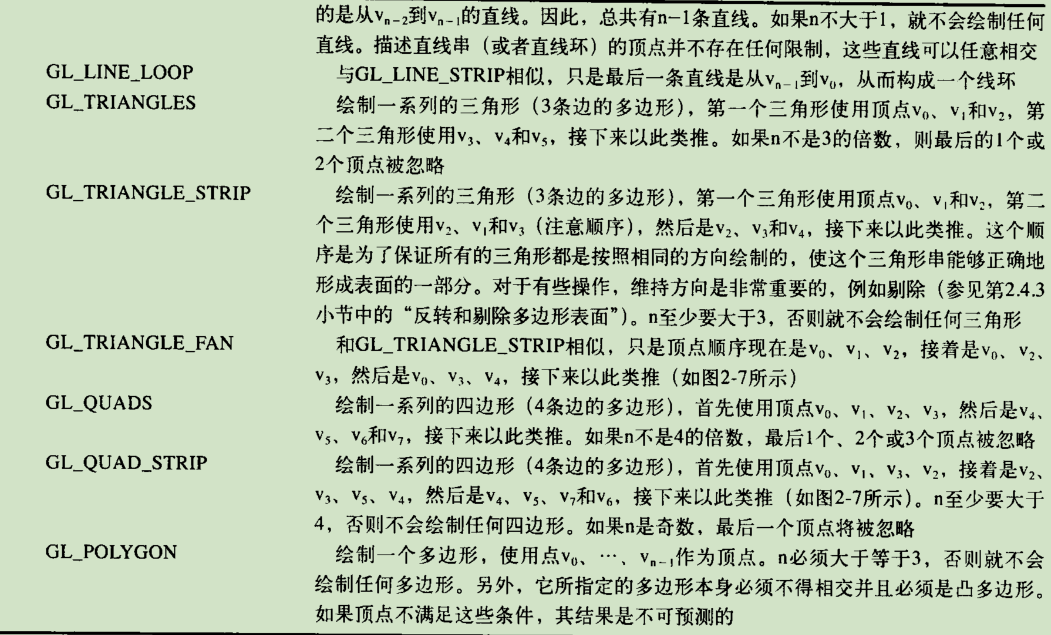
Mode：

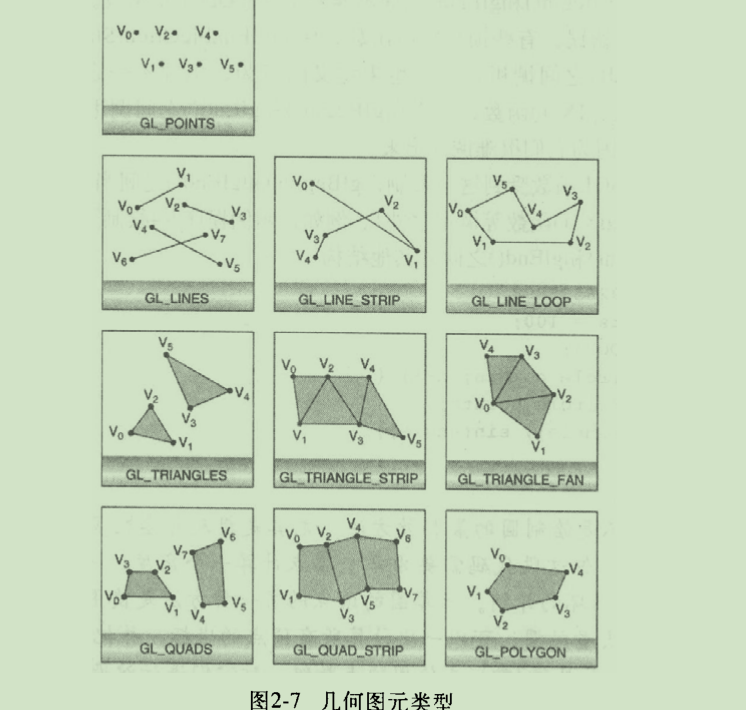
 

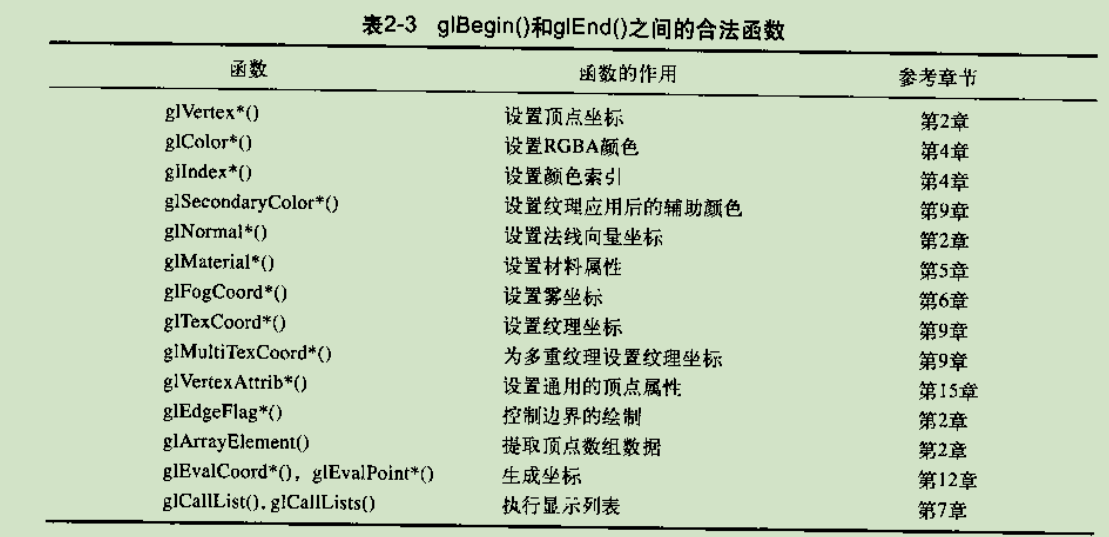
Void glEnd（void）；

标志着一个顶点数据列表结束。









**基本状态管理**

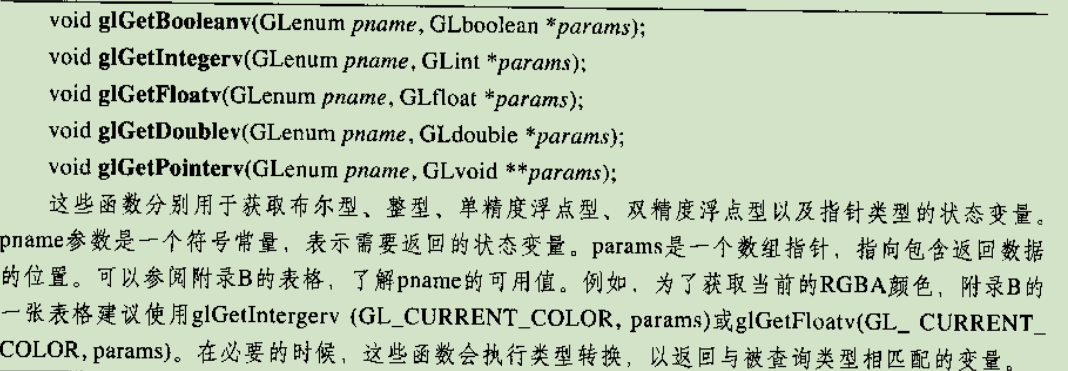
Void glEnable（GLenum capability）；

Void glDisable（GLenum capability）；

GlEnable开启一个功能，glDisable关闭一个功能

GLboolean glIsEnabled（GLenum capability）；

根据被查询的状态当前处于启用还是禁用状态，它返会GL\_TURE||GL\_FLASE



**显示点、直线多边形**

**点：**

Void glPointSize(GLfloat size);

设置被渲染点的宽度，以像素为单位。Size必须大于0，在默认状况下为1.0；

线：

Void glLineWidth（GLfloat width）；

以像素为单位设置宽度，用于直线渲染。Width参数必须大于0.0，莫默认为1

点画线：

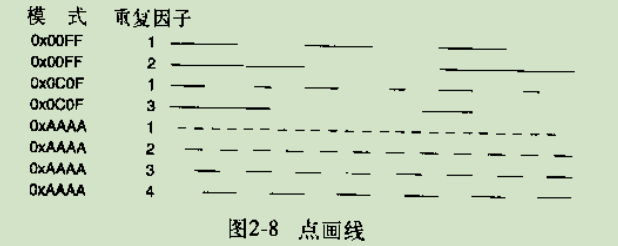
可以使用glLineStipple（）函数定义点画模式，然后使用glEnable（）函数启用直线点画功能。

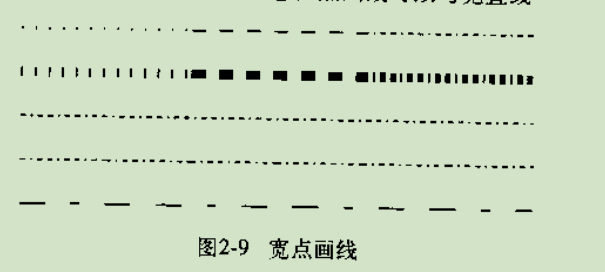
glLineStipple（1,0x3f07）;

glEnable(GL\_LINE\_STIPPLE);

Void glLineStipple(Glint factor,GLushort pattern);

设置直线的当前点化模式。Pattern参数是一个由1或0组成的16位序列他们根据徐瑶进行重复，对一条特定的直线进行点画处理。从这个模式的低位开始，一个像素一个像素地进行处理。如果模型对应的位是1，就会之这个像素，否则就不绘制。。模式可以使用factor参数（表示重复因子）进行扩展，必须调用glEnable(GL\_LINE\_STIPPLE);才能启用直线点画功能。禁止电话功能glDisable(GL\_LINE\_STIPPLE);

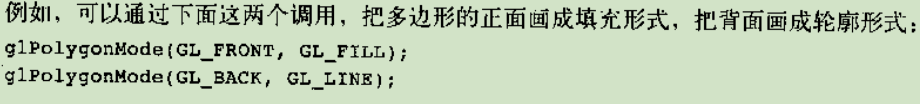




**多边形**

Void GlPolygonMode(GLenum face,GLenum mode);

控制一个多边形的正面和背面的绘图模式。Face参数可以是GL\_FRONT\_AND\_BACK、GL\_FRONT、GL\_BACK.mode参数可以是GL\_POINT、GL\_LINE或GL\_FILL，表示多边形应该被画成点、轮廓还是填充形式。在默认情况下，多边形的正面和背面都画成填充形式。



Void glFrontFace(GLenum mode)；

控制多边形的正面是如何决定的。默认情况先，mode是GL\_CCW，它表示窗口坐边上投影多边形的顶点顺序为逆时针方向的表面为正面。如果mode是GL\_CW,顶点顺序为顺时针方向的表面被认为正面。

Void glCullFace(GLenum mode);

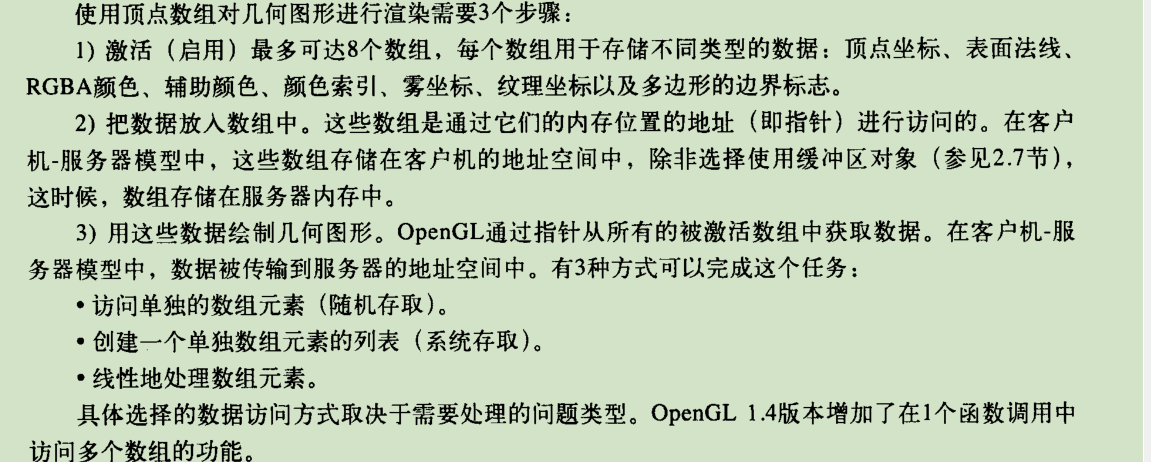
表示那些多边形在转换到屏幕坐标之前应该丢弃，mode参数可以是GL\_FRONT、GL\_BACKG或GL\_FRONT\_AND\_BACK，分别表示正面多边形、背面多边形和所有多边形。为了使提出生效，必须以GL\_CULL\_FACE为参数调用glEnable()函数来启用剔除功能。另外，可以用同一个参数调用glDisable（）函数禁用剔除功能。

Void glEdgeFlag(GLboolean flag);

Void glEdgeFlagv(const GLboolean \*flag);

表示一个顶点是否应该被认为是多边形的一条边界边的起点。如果flag是GL\_TURE，边界标志就设置为TURE，在此之后创建的所有点点都认为是边界边的起点、直到用GL\_FALSE为flag参数的值再次调用了这个函数。

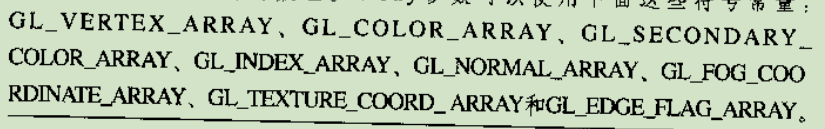
**定点数组：**



**1启用数组：**

Void glEnableClientState(GLenum array);

指定了需要启用的数组。



void glDisableClientState（GLenum array）；

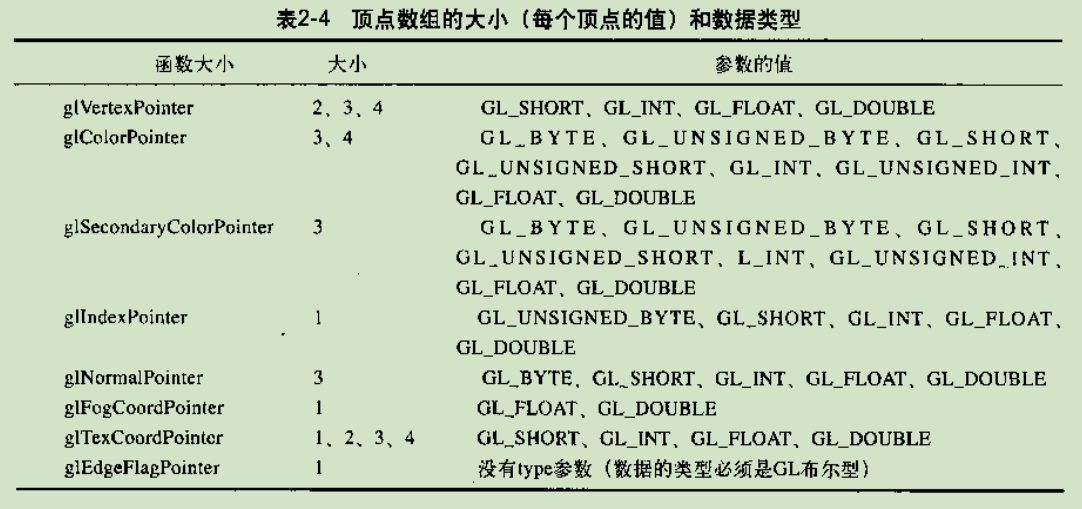
指定了需要禁用的数组。它接受的参数与glEnableClientState（）函数相同

**2指定数组数据：**

Void glVertexPointer(Glint size，GLenum type，GLsizei stride，const GLvoid \*pointer);

指定了需要访问的控件坐标数据。Pointer是数组包含的第一个顶点的第一个坐标的内存地址。Type指定了数组中的每个坐标的数据类型（GL\_SHORT，GL\_INT、GL\_FLOAT或GL\_DOUBLE）。Size是每个顶点的坐标数量，它必须是2、3、3.stride是连续顶点之间的字节偏移量。如果stride是0、数组中的顶点便是紧密相邻的。

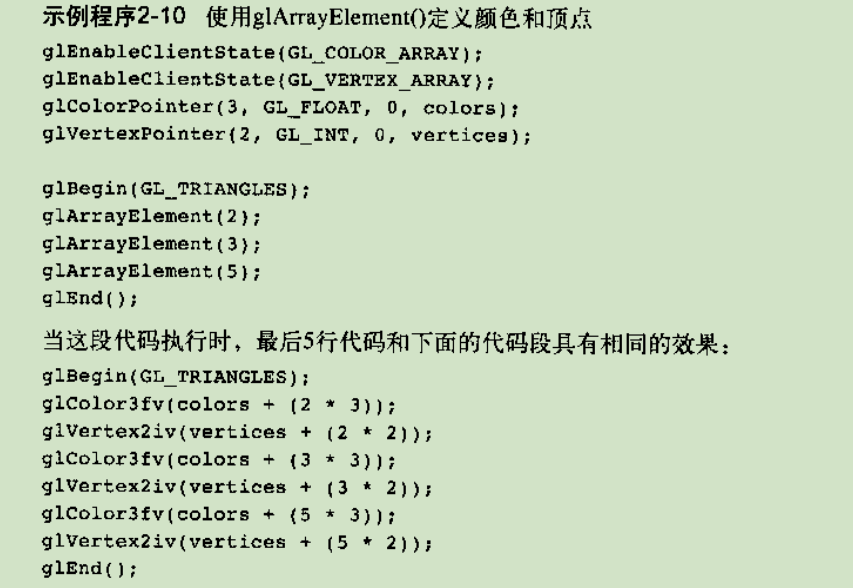




**3引用和渲染：**

Void glArrayElement（Glint ith）；

GlArrayElement()通常是在glBegin（）和glEnd()之间调用。

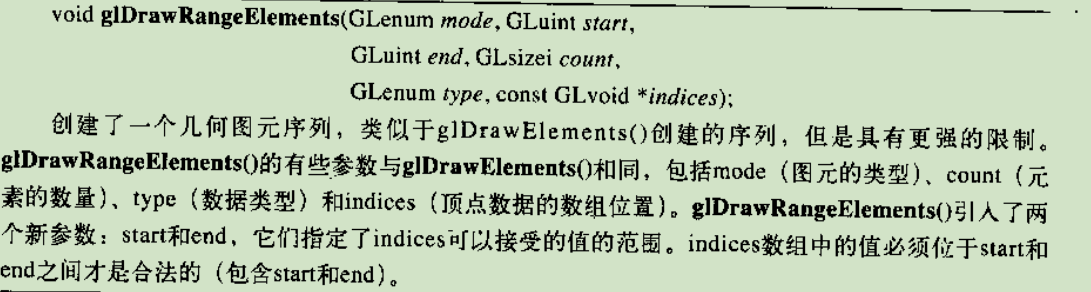


Void glDrawElement（GLenum mode，GLsizei count，GLenum type,const GLvoid \*indices）；

使用count个元素定义一个几何图元序列，这些元素的索引值保存在indices数组中。Type

必须是GL\_UNSIGNED\_BYTE、GL\_UNSIGNED\_SHORT或GL\_UNSIGNED\_INT、表示indices数

组的数据类型。Mode指定了被创建那种类型的图元，它的值和glBegin（）函数所接受的参数值相同。

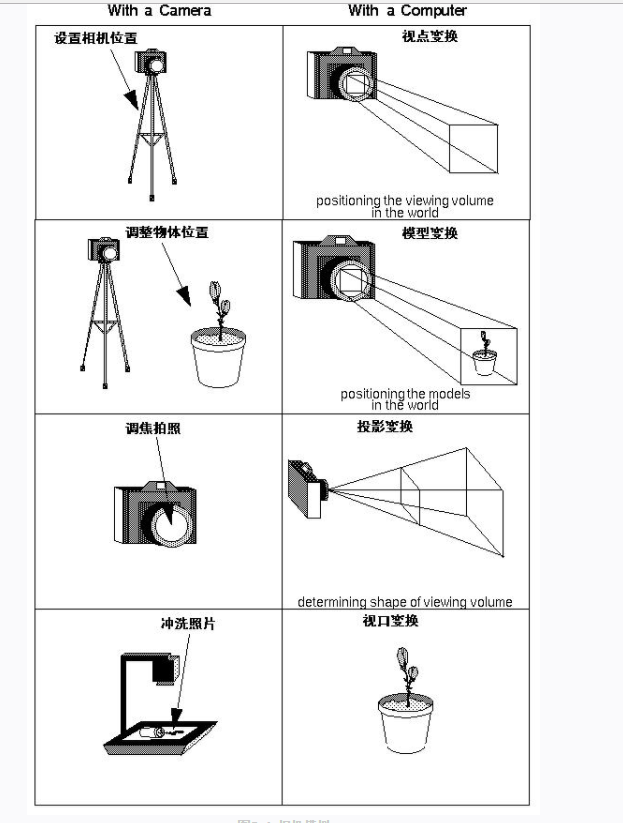


Void glPrimitiveRestartIndex（GLuint index）；

指定一个顶点数组元素索引，用来表示一个新的图元在渲染是的开始位置。

**视图**

**视图变换、模型变换、投影变换、视口变换**



Void glMatrixMode（GLenum mode）；

指定了需要修改的模型视图矩阵，投影矩阵还是纹理矩阵。Mode的值可以是GL\_MODELVIEW、GL\_PROJECTION或GL\_TEXTURE。接下来调用的变换函数将印象它指定的矩阵。注意，一次只能修改一个矩阵

Void glLoadIdentity（void）；

把当前的可修改矩阵设置为4\*4的单位矩阵

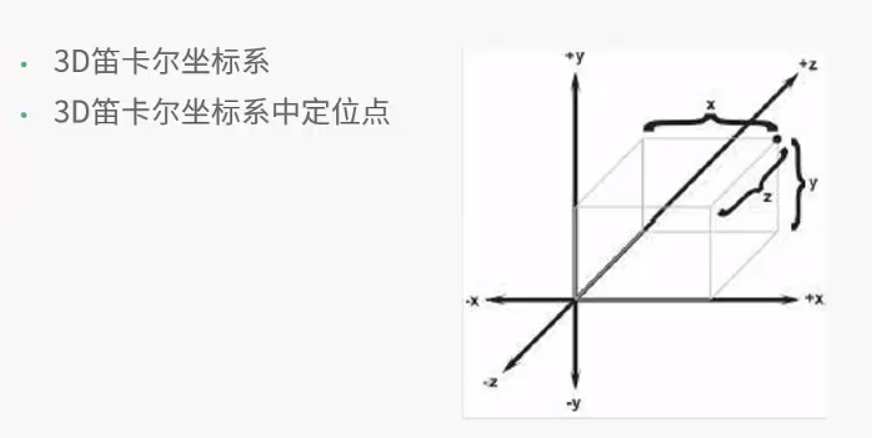
OpenGL数学基础

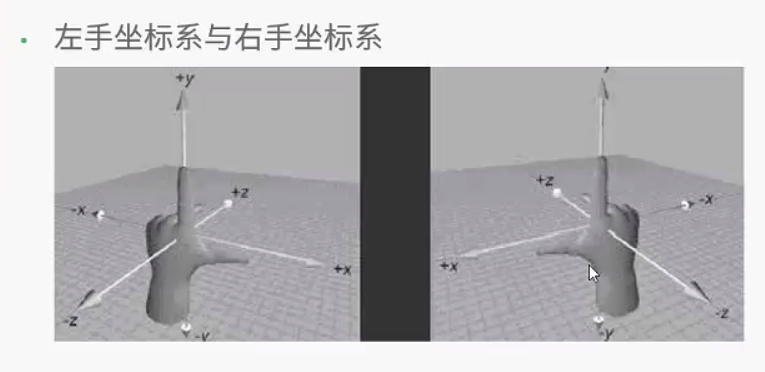
线性代数:

三角函数:

3D几何变换:

透视投影:

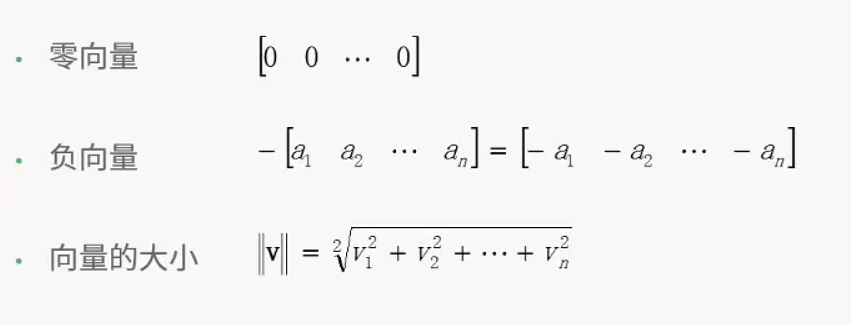


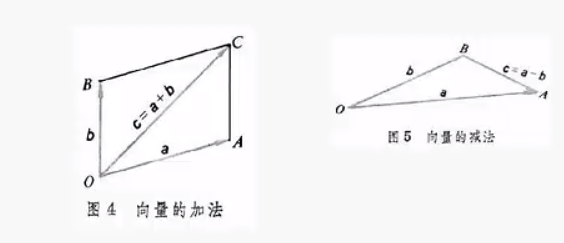


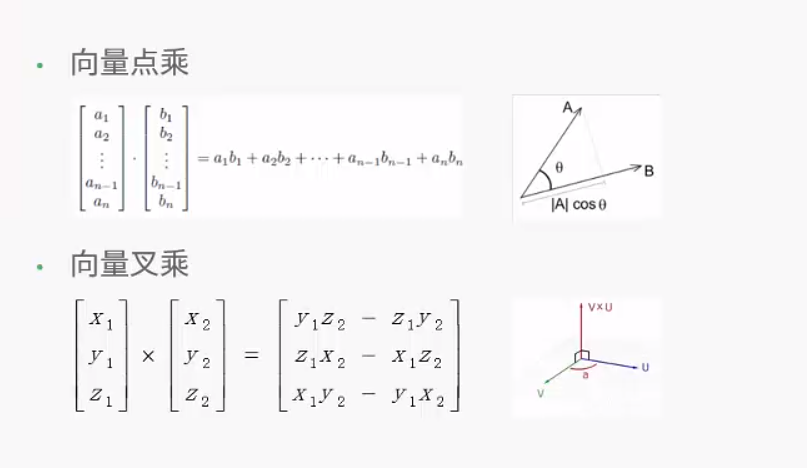
世界坐标系

向量；

向量的大小







矩阵：

