图形流水线总览：

逐像素绘制

逐面片绘制：逐面片地进行投影（三维到二维）、光栅化（计算面片所覆盖的栅格）

Graphics Pipeline：

面片->Vertex operation ->Rasterization->Fragment operation->Frame buffer

Vertex operation：变换：几何变换、投影、裁剪、视口变换

Rasterization：光栅化

Fragment operation：对每一个小像素进行颜色、纹理等处理

GUP：任务相对固定，可以并行计算

CUP：灵活，实现通用计算

GUP中图形流水线的发展以及Shader编程：

固定流水线模式：流程固定，只能设置一些参数

2002年开始，可编程功能：Vertex Shader ->Vertex operation

Fragment(Pixel) Shader->Fragment operation

Vertex Shader:

可直接在（把变形的计算从CUP移到GUP）Vertex Shader中编程控制顶点的各种 属性。

Fragment Shader：

可以直接在Fragment Shader中编程控制片元、像素的各种属性。

2006:

Vertex Shader->Geometry Shader->Rasterization->Fragment Shader->Frame buffer

Open GL

一个正方体：消隐、填充颜色（光栅化）、打光（光照）、阴影、纹理。

能做什么？（主要任务用来流水线渲染，而非建模）

变换（几何变换，投影，视口变换）

光照与材质

纹理映射

图像功能（anti-aliasing（反混淆），fog，blending（混合）…）

特效（Depth Cue（深度暗示），Motion Blur（运动模糊）…）

建模（basic primitives、NURBS）

……

设置流水线的一系列状态参数，并不是用来真正执行绘制操作的

两种编程模式：

固定流水线模式—传统模式

Shader编程模式—现代方式

glutReshapeFunc

目的：当窗口尺寸改变时，图形比例不发生变化

思路：窗口宽高比改变时，通过改变窗口显示区域大小，并利用投影矩阵改变观测物体大小使之适应变化。

glutReshapeFunc(&reshape); 函数先控制窗口大小改变时的操作

glutIdelFunc(&IdleFunc); 函数再控制窗口有无事件发生时的操作

•Open GL中 观察者的坐标系是如上图所示，其中的z轴就是深度，也就是物体和我们的距离，z的正方向是指向屏幕内部，所以，离我们越远z值越大，反之亦然

•物体是3D的，屏幕显示设备是2D的，那么我们通过z值的大小来模拟现实中的3D物体，z值越大（离我们越远）在屏幕上显示的就越小，z值越小（离我们越近）在屏幕上显示的就越大，近大远小，复合眼睛看到的实际情况

•颜色缓冲区和深度缓冲区

颜色缓冲区，保存每一个像素点的颜色值，像素（key）= 颜色（value）

深度缓冲区，保存每一个像素点的颜色值，像素（key）= 深度（value）

采取keyValue的模式（个人理解）

•颜色缓冲区和深度缓冲区的关系

两者一一对应，同时更新颜色值RGBA和深度值z，因为一个像素点只能保留一个颜色和一个深度值z

•深度测试

深度缓冲区(DepthBuffer)和颜⾊色缓存区(ColorBuffer)是对应的.颜⾊色缓存区存储像素的颜⾊信息，而深度缓冲区存储像素的深度信息. 在决定是否绘制一个物体表⾯时, ⾸先要将表面对应的像素的深度值与当前深度缓冲区中的值进⾏比较. 如果⼤大于深度缓冲区中的值,则丢弃这部分.否则 利⽤用这个像素对应的深度值和颜色值.分别更新深度缓冲区和颜⾊缓存区. 这个过程称为”深度测试”

•深度测试的理解

屏幕上显示的2D物体，实际上就是每一个像素点上显示不同的颜色，如果两个都是不透明的物体，相互重叠地放在我们眼前，重叠地部分看到的一定是离我们近的物体，这是毋庸置疑的，那么对应到Open GL里面的深度z就是表示距离我们的远近，不透明的情况下，我们比较重叠部分的深度值z，哪个越小即离我们越近，重叠部分的像素点就应该渲染对应的颜色，而深度值z大的被遮挡住就不需要渲染了，这就是深度测试，深度，深度缓冲区，再次理解如下所述

•什么是深度?

深度其实就是该像素点在3D世界中距离摄像机的距离，Z值

•什么是深度缓冲区?

深度缓存区，就是⼀块内存区域,专⻔存储着每个像素点(绘制在屏幕上的)深度值，深度值(Z值)越⼤，则离摄像机就越远.

•为什么需要深度缓冲区?

在不使用深度测试的时候，如果我们先绘制一个距离比较近的物体，再绘制距离较远的物体，则距离远的位图因为后绘制，会把距离近的物体覆盖掉，有了深度缓冲区后， 绘制物体的顺序就不那么重 要的， 实际上，只要存在深度缓冲区，OpenGL 都会把像素的深度值写⼊到缓冲区中. 除⾮调⽤glDepthMask(GL\_FALSE).来禁⽌写入.

Freeglut编程的基本模式：

Main（）

{

设置显示模式；

初始化窗口；

创建窗口；

设置一系列回调函数；(由系统响应消息来调用的函数)

启动主循环；

}

消息驱动模式

glutPostRedisplay();

glutIdleFunc();

OpenGL的默认坐标系和默认视点位置：

1、右手坐标系

2、默认视点在原点

3、视线方向是z轴负方向

图形流水线：

OpenGL是一个状态机

整个绘制操作会按照固定的流程进程，如同一条流水线。

OpenGL函数只是用来设置这条流水线的一些状态参数，而并不是用来真正执行绘制操作的。

变换：

几何变换->投影->裁剪->视口变换

几何变换：

放缩：以原点为中心进行放缩，可以先将物体移到原点，放缩后再移回去

用堆栈管理矩阵：

glPushMatrix()：将栈顶状态复制一份压栈

glPopMatrix()：将栈顶弹出

glLoadIdentity()使栈顶矩阵成为单位矩阵。

用