第1章　OpenGL简介

OpenGL是图形硬件的一个软件接口。它的主要作用是将二维或三维的对象绘入一个帧缓冲区中。对象被描述为一系列的顶点（用来定义几何对象）或像素（用来定义图像）。OpenGL对数据进行几个步骤的处理从而将其转换成像素，这些像素将在帧缓冲区中形成最终需要的图形。

本章将全面地介绍OpenGL的工作原理，包括以下两个主要部分：

- OpenGL基础　主要解释基本的OpenGL概念，例如什么是几何图元以及OpenGL如何实行客户端-服务器端的执行模式。
- 基本OpenGL操作　通过一个高层的模块图来说明OpenGL在帧缓冲区中处理数据并生成相应图像的过程。

1.1　OpenGL基础

本节主要解释一些OpenGL固有的命令。

1.1.1　OpenGL图元及命令

OpenGL通过几个可选模式来绘制“图元”——点、线段或多边形。你可以对各种模式独立进行控制；也就是说，一个模式的设置并不影响其他模式的设置（尽管模式间的相互作用将影响帧缓冲区中的最后结果）。OpenGL的程序通过调用函数来指定图元、设置模式并描述其他操作。

在OpenGL中，图元由单个或多个顶点组来定义。一个顶点可以是一个点、一条线的端点或一个多边形的角。数据（由顶点坐标、颜色、法线、纹理坐标和边界标志所组成）与顶点是相对应的，并且每个顶点和与它相关的数据独立，按照次序，采用同样的方法进行操作。这里仅有一种情况例外，那就是当一组顶点必须被“剪切”从而使得某一特定的图元刚好在某一指定的区域内，则顶点数据可能会被修改并产生新的顶点。其剪切的类型由该组顶点所代表的图元决定。

虽然有些命令在生效前可能会有一段不确定的延时，但是OpenGL的所有命令都是依照其被接收的次序来执行的。也就是说，每个图元在被绘制完成之前，其后面的命令将不会有效。同时，这也意味着使用状态查询命令时它所返回的数据将只包含所有以前发布并已执行完毕的OpenGL命令。

1.1.2　OpenGL是一种过程语言

OpenGL从根本上说是一种过程语言而非描述性的语言：OpenGL提供了直接控制二维和三维几何体的基本操作。它包含了转换矩阵、光照方程系数、反走样方法和像素校正算子的描述。然而，OpenGL并不能直接描述或建模复杂的几何对象。

你所发布的OpenGL命令指定了怎样产生一个特定的结果（即接下来所应该采取的操作）而不是指定确切的结果。正是这种过程特性使我们能够了解OpenGL是如何工作的——只有明白了它的操作顺序才能对如何使用它有更深的理解。

1.1.3 OpenGL的执行模式

OpenGL使用了一种客户端-服务器端的模式来解释命令。应用程序（客户端）所发布的命令将通过OpenGL（服务器端）来编译和处理。服务器的操作既可以同客户端在同一台计算机上进行，又可以分别属于不同的机器。因此，从这个意义上讲， OpenGL是网络透明的。一个服务器可以维护数个GL*上下文*，每个上下文被封装在一个 GL状态里。服务器可以同时包含几个 GL上下文，每个上下文都被封装在一个 GL状态里。每个客户端都可以连接到这些上下文中的任何一个。所需要的网络协议可以是扩充过的已有协议（如 X Window系统）或是一个完全独立的协议。OpenGL并没有提供命令用来获取用户的输入。

窗口系统分配给帧缓冲区的资源将最终控制 OpenGL命令对帧缓冲区的影响。窗口系统将决定OpenGL帧缓冲区的哪些部分可在给定的时间内被访问并将这些部分的结构传送给 OpenGL。因此OpenGL中不存在配置帧缓冲区及初始化 OpenGL的命令。帧缓冲区的配置是在 OpenGL外，由与其相关联的窗口系统来完成的，而 OpenGL的初始化则是当窗口系统为 OpenGL绘图分配一个窗口时完成的。（GLX —— OpenGL界面的X扩展，提供了这些功能。有关细节请参阅 2.4节“对X窗口系统的OpenGL扩展”。）

1.2 基本OpenGL操作

图1-1是抽象的高层的模块图，它展示了 OpenGL处理数据的过程。如图所示，命令由左边进入，然后经过了一个可被看作处理流程的处理过程。其中有一些命令指定了所要绘制的几何对象，而另一些命令则用于控制不同阶段中对象的处理方法。

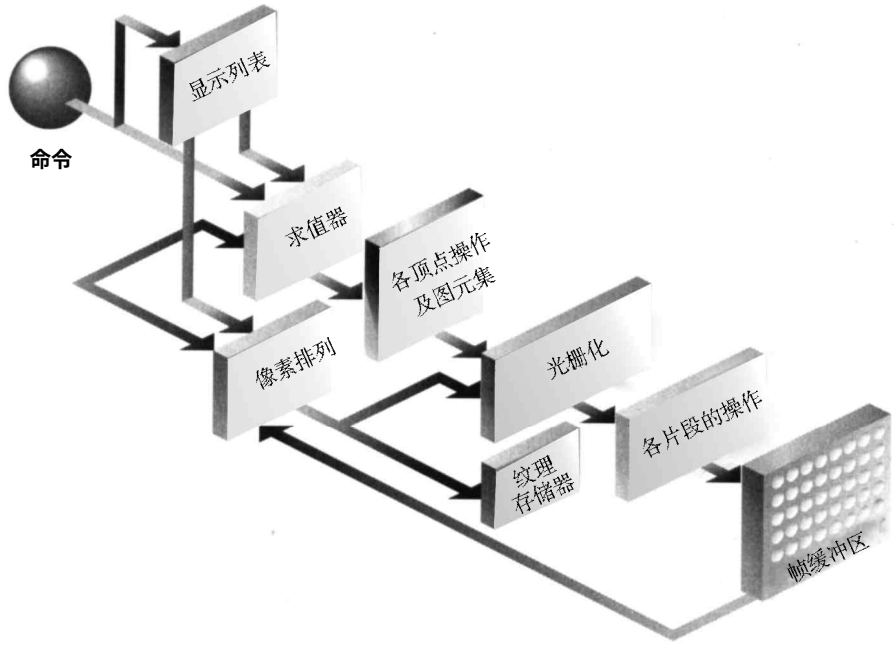

图1-1 OpenGL数据处理过程

当命令进入流程时，你可以选用两种方法对它们进行处理：一种是通过流程立即执行这些命令；另一种是将其中一些命令组织到一个“显示列表”，过一段时间再执行它们。

流程中的“求值器”阶段通过将输入值赋给多项式命令提供了一种非常有效的方法来生成几何曲线和曲面的近似值。接下来的“各顶点操作及图元集”阶段主要是处理 OpenGL的几何图元——点、线段和多边形。所有这些图元均由顶点来描述。顶点可以被转换和照亮。接下来图元被剪切到视口，为下一阶段做好了准备。

“光栅化”生成了一系列的帧缓冲区地址以及相应的用于描述点、线段或多边形的二维值。这些生成的“片断”将被送到最后一个阶段——“各片断的操作”，这一阶段是数据以像素形式存入“帧缓冲区”之前的最后操作。这些操作包括根据帧缓冲区中原有的深度值（用于深度缓存操作）与输入值而有条件地更新帧缓冲区的操作，还包括对输入的像素颜色与已存储的颜色所进行的融合操作，对像素值所进行的屏蔽操作及其他的逻辑操作。

数据是以像素形式而非顶点形式输入的。这些数据可以用来描述一个用于纹理映射的图像，它将跳过第一阶段（如前面所述），而通过“像素操作”阶段作为像素来处理。这一阶段的处理将导致两种结果：其一是被存入“纹理存储器”，以备光栅化阶段所用；其二是直接被光栅化。后者所形成的片断将被存入帧缓冲区，就好象它们是由几何数据生成的一样。

一个OpenGL应用程序可以获得OpenGL状态的所有元素，其中包括纹理存储器的内容，甚至还包括帧缓冲区的内容。