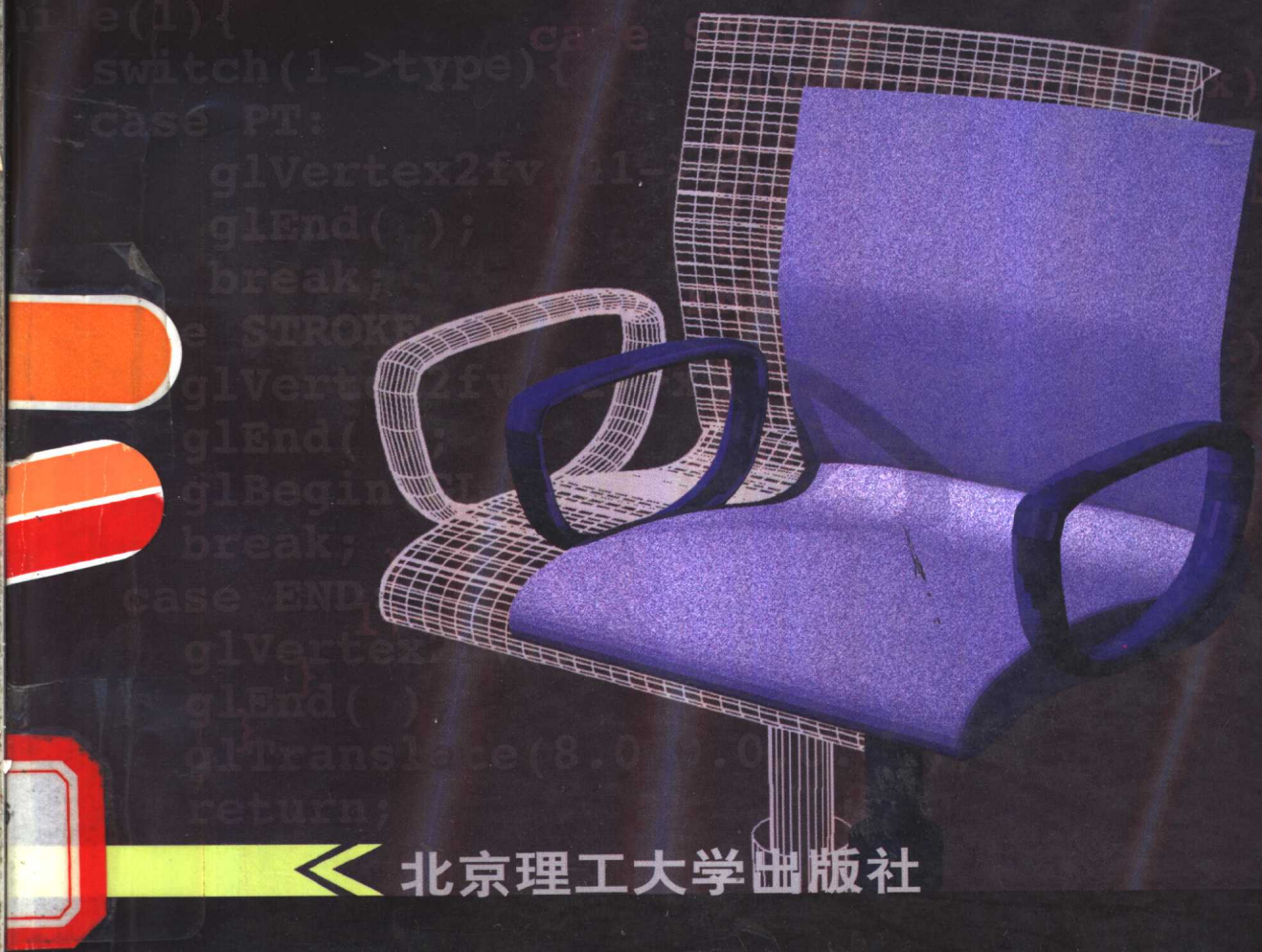
软件开发伴侣丛书

如何使用 OpenGL 开发程序

荷塘月色创作组



北京理工大学出版社

软件开发伴侣丛书

如何使用OpenGL 开发程序

荷塘月色创作组

北京理工大学出版社

内 容 简 介

OpenGL 作为事实上的计算机三维图形技术的标准，在众多的计算机平台上获得了广泛的应用，并且由 Microsoft 将 OpenGL 引入 Windows 95 和 Windows NT 操作系统。在 PC 上，新一代的 3D 图形加速卡都支持 OpenGL，Java 语言的三维图形技术的内核也将采用 OpenGL 标准。本书从 OpenGL 建模、OpenGL 三维空间设定和 OpenGL 环境空间设定等角度，力图细致地阐述利用 OpenGL 在 Windows 95/Windows NT 平台上建立三维图形应用的步骤。计算机图形编程爱好者和专业编程人员都能通过本书有所收获。

图书在版编目(CIP)数据

如何使用 OpenGL 开发程序/荷塘月色创作组编著．—北京：北京理工大学出版社，1999.1
(软件开发伴侣丛书)
ISBN 7-81045-488-9

Ⅰ．如… Ⅱ．荷… Ⅲ．图像处理-标程程序，OpenGL Ⅳ．TP391.4

中国版本图书馆 CIP 数据核字(98)第 26851 号

责任印制：李绍英　　责任校对：李　军

北京理工大学出版社出版发行
(北京市海淀区白石桥路 7 号)
邮政编码 100081　电话 (010)68912824
各地新华书店经售
北京房山先锋印刷厂印刷

*

787 毫米×1092 毫米　16 开本　17.75 印张　432 千字
1999 年 1 月第 1 版　1999 年 1 月第 1 次印刷
印数：1—4000 册　　定价：27.00 元

前　言

值得高兴的是，历时半年，《软件开发伴侣丛书》终于和读者见面了。本丛书是清华大学数名从事软件开发的博士生和硕士生在完成繁重的科研任务同时，利用工作之余的休息时间完成的。我们所想的是利用自己的一些专业知识，为中国的计算机软件产业做出更多的贡献。

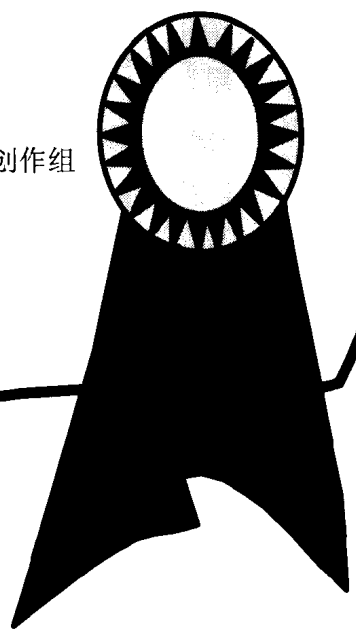
想写好一套关于软件开发的技术性较强的丛书是一件艰苦的工作。这套丛书从选题，到丛书的风格设计，每本书的技术水平的定位，实例的选择等，都浸透了作者们的大量心血。但是，由于时间较为仓促，篇幅有限，作者的水平也有待提高，所以难免出现各种各样的错误，希望读者能够指出。这也是对我们工作的一种支持。

另外，我们采用的制作方式也是独特的，丛书中的每一本书，都由策划，执笔两个层次构成。我们希望，采用这种方式可以提供给读者的是一套风格统一，质量较高的精品丛书。

《如何使用 OpenGL 开发程序》是由翟林、郭庆、杨磊、袁波等人共同执笔。由杨磊策划和创意。OpenGL 作为事实上的计算机三维图形技术的标准，在众多的计算机平台上获得了广泛地应用，并且由 Microsoft 将 OpenGL 引入 Windows 95 和 Windows NT 操作系统。在 PC 上，新一代的 3D 图形加速卡都支持 OpenGL，Java 语言的三维图形技术的内核也将采用 OpenGL 标准。本书从 OpenGL 的建模、OpenGL 三维空间设定和 OpenGL 环境空间设定等角度，力图细致地阐述利用 OpenGL 在 Windows95/Windows NT 平台上建立三维图形应用的步骤。计算机图形编程爱好者和专业编程人员都能通过本书有所收获。

荷塘月色创作组

1998.5.1



目　录

第一章 什么是 OpenGL

本章讲述

- OpenGL 的由来
- OpenGL 能完成什么
- OpenGL 的基本处理流程
- 本书的内容线索

1.1 3D 与 OpenGL

我们每天所面临的物质世界是一个三维的世界，其中充满各种各样的三维物体。在计算机问世以来的半个多世纪里，人们一直在试图让计算机能够描述和处理现实世界中尽可能多的对象，这当然也包括精确地描述三维物体和三维世界。

三维世界中包含的信息量，大大超过二维世界。因此，对三维世界的描述难度也非常大。需要描述的信息量和需要处理的信息量之大，都明确地反映出一种需要，即必须有一种能够完善地表现三维世界的技术。最近几年，计算机图形学发展迅猛，三维表现技术愈发完善，已经能够真实地再现三维世界中的物体，能够用三维物体及其组合来表达复杂的信息。同时随着计算机处理能力的飞速提高，计算机尤其是微机已经能够胜任对三维表现技术所描述的三维物体进行平滑处理的工作了。

三维图形技术在走向成熟的过程中，不断地被应用到现实的尖端领域，反过来也不断推动三维图形技术的进一步发展。三维图形技术主要应用在三个方面：科学可视化、计算机动画和虚拟现实。在最新的高科技影视制作、新颖的商业广告、机械设计、VR 仿真等领域，都大量地应用着三维图形技术。

在三维图形技术发展的历程中，产生了很多优秀的三维图形应用开发工具。在这中间，以 SGI（Silicon Graphics Incorporated）公司为 SGI 工作站开发的 GL（Graphics Library)三维图形库表现最为突出，赢得了众多技术人员的喜爱，并进而发展成为 OpenGL。Open 本身就有开放的含义，而 OpenGL 形成的目的就是要让应用程序能够创建出高水平的、高质量的彩色三维影像，而这一切工作都是独立于窗口系统（Windows，X-Window 等）、操作系统（Windows NT，UNIX 等）和硬件平台（Intel，Cray，Alpha，Sparc 等)的。

现在 OpenGL 已经是事实上的三维图形处理的工业标准。OpenGL ARB(Architecture Review Board)现在负责定义 OpenGL。OpenGL ARB 的成员包括：SGI、Microsoft、Intel、IBM 和 DEC（Digital Equipment Corporation，现已被 Compaq 收购）。有兴趣的读者可以访问 http://www.SGI.com/Technology/OpenGL、http://www.SGI.com/Developers和 http://www.

OpenGL.org，以获取最新的关于 OpenGL 的信息。

OpenGL 作为注册商标，其图形标志见图 1-1。

1.2 OpenGL 能干什么

作为硬件图形设备的软件接口，OpenGL 的主要意图是在帧缓冲区中精确地绘制 2D 和 3D 的物体。这些物体被描述为顶点（定义几何物体）或像素（定义影像）的序列。OpenGL 形成若干处理步骤，将这些数据转化成像素，并在帧缓冲区中形成最终的影像。

图 1-1 OpenGL 标志

OpenGL 作为独立于窗口系统、操作系统和硬件平台的三维图形处理技术，必然包括：图形处理、交互、窗口仿真、设备驱动几个方面。

1.2.1 图形处理

三维图形处理包括三维图形的绘制以及各种真实效果的处理。

- **建模**：定义三维物体的几何模型；
- **轮廓绘图**：以轮廓线的方式绘制出三维物体的外形；
- **深度轮廓**：这是一种效果处理，实现人眼观察时，远处的物体暗一些的效果；
- **反走样**：避免在绘制线段时的锯齿状台阶；
- **光源**：通过平行光和点光源来模拟现实世界的光照；
- **变换**：各种图形处理中所涉及到的变换，如：平移、旋转、剪裁等；
- **明暗处理**：模拟着色后的物体表面的明暗效果；
- **光滑处理**：更逼真地模拟物体表面光照后的效果；
- **阴影**：使得三维场景更真实和富有立体感；
- **纹理**：定义物体表面的材质，表达物体表面的质感；
- **运动模糊**：模拟人眼观察运动物体时的动感；
- **大气效果**：模拟空气中的能见度效果；
- **景深效果**：模拟照相中的景深，即在焦点附近清晰，远离焦点处模糊。

1.2.2 交互

在三维应用中，需要实时地接受用户输入的信息，这包括处理鼠标和键盘的基本输入，从而调整后续处理的步骤。鼠标和键盘的输入动作以回调函数的形式通过 OpenGL 的窗口仿真体系，传递给应用程序。

1.2.3 窗口仿真

既然 OpenGL 是面向多平台，独立于窗口系统的，那么就需要一整套仿真窗口的功能，以实现类似于窗口的效果，并能在窗口系统中顺利地运行。OpenGL 为应用程序提供一个窗口环境/界面，并通过设置回调（Callback）函数的办法，将应用程序的消息响应与窗口系统的事件循环联系在一起。其实，可以直接使用平台中的窗口系统，只不过在有移植考虑时，最好使用 OpenGL 提供的窗口函数。

1.2.4　外部设备访问

OpenGL 提供了一组通用的访问外部设备的函数，用以实现文件格式、屏幕访问等功能，从而实现独立于外部硬件设备的目的。

1.3　OpenGL 的基本处理流程

在如图 1-2 所示的 OpenGL 基本工作流程图中，描述了一个 OpenGL 命令从开始到最终形成图像的过程。

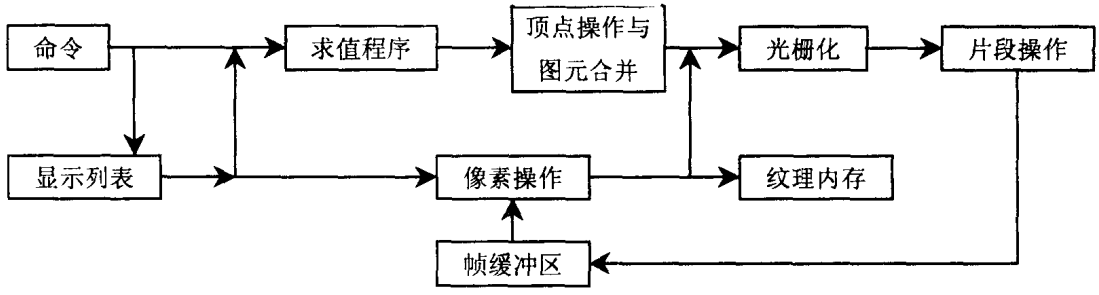


图 1-2　OpenGL 的基本工作流程图

OpenGL 的命令发出后，或者是存储在显示列表中供以后调用，或者与像素操作一起处理，通过运算、逐个顶点操作，进行光栅化处理、逐个片段操作，形成最终的影像并存入帧缓冲区。对于图像数据，像素操作的结果被存储在纹理内存中，供纹理组装后与顶点数据一起进行光栅化处理，共同形成三维影像。

对于创建一个三维图形的基本步骤，大致上包括下面三个环节。

- **建模**：描述三维物体的外部轮廓和表面信息；
- **设置视点**：描述观察者的空间位置；
- **设定环境**：描述环境的特征，如：光源、空气能见度等。

有了这些描述后，OpenGL 将根据数学描述，通过计算得到该三维物体的空间位置和色彩信息，进而转化为计算机上的像素，即光栅化过程。同时，还可能有诸如计算消隐、像素处理等操作，最后光栅化后的数据被送到帧缓冲区，三维物体就显示在屏幕上了。

1.4　本书的内容线索

本书将主要介绍在 Microsoft Windows 95/NT 平台下 OpenGL 的开发，应用的编程工具是 Microsoft Visual C++ 5.0。随后的章节中，将介绍 Microsoft Windows 95/NT 平台下 OpenGL 的构成。

本书主要面对的读者群是对 OpenGL 有一定了解的软件开发人员或软件爱好者。因此，本书将忽略一些对 OpenGL 较基本知识的论述，重点放在使用 OpenGL 实现高级应用的话题上。最后简要介绍 VRML。

第二章 Windows 95/NT 下的 OpenGL

在 Windows NT 4.0 及其以上版本和 Windows 95 OEM Service Release 2 及其以上版本中，开始提供对 OpenGL 的支持。在 Windows 中可以看到有一些名为三维××等屏幕保护程序，这些都是采用 OpenGL 技术编写的。

本章讲述

- 在 Windows 95/NT 中，OpenGL 的构成
- 在 Windows 95/NT 中，OpenGL 的开发方式

2.1 Windows 95/NT 下的 OpenGL 组件

在 Windows 95/NT 中，有两种 OpenGL 组件。一种是基于静态库（.LIB）的 SDK；一种是基于动态库（.DLL）的 SDK。

2.1.1 OpenGL SDK

OpenGL SDK 是由 SGI 公司（见图 2-1）提供的 C/C++开发工具，其核心文件是 GLU. H、GL. H、OpenGL. LIB 和 GLU. LIB。并提供了 13 个在 Win32 境下编写的例子源程序，约 50 个 HTML 格式的帮助文件（通过 Internet 浏览器观看）和 2 个演示程序。

图 2-1 SGI Logo

在 Microsoft Visual C++ 5.0 开发环境下，读者可以新建一个 Win32 Console 的工程，并引用上述的两个静态库文件，即可立即享受 OpenGL 带来的全新三维世界。

2.1.2 Microsoft Implementation of OpenGL

Microsoft Implementation of OpenGL 是在 Microsoft™ Windows NT®和 Windows® 95 操作系统（见图 2-2）下的基于工业标准的 OpenGL 3D 图形软件接口。在 Microsoft Visual C++ 5.0 开发环境中提供了对 OpenGL 的全面支持。

其核心是 OpenGL32.LIB、GLU32.LIB、OpenGL32.DLL、GLU32.DLL 以及一些头（.Hxx）文件。另外还包括 OpenGL.LIB、GLU.LIB、OpenGL.DLL、GLU.DLL 等开发 16 位应用系统的库文件。作为开发的辅助工具，OpenGL 还提供了 GLAux.LIB 和 GLUT.LIB 及其头文件，但是它们不与 OpenGL SDK 一起提供，需要的读者可以从以下网址免费获得：

图 2-2 Microsoft Logo

http://reality.sgi.com/mjk_asd/glut3/glut3.html
http://www.cs.utah.edu/~narobins/opengl.html

读者还能够从网址

ftp://sgigate.sgi.com/pub/opengl/contrib/samples.tar.Z，下载最新的例子和演示程序的（源）代码。

在 Microsoft Visual C++ 5.0 开发环境下，读者可以自由地应用 Microsoft Visual C++ 5.0 提供的灵活强大的工具，与 MFC 结合起来，创建丰富多彩的 3D 应用。在 Microsoft Visual C++ 5.0 中也同样提供了大量的例子源程序和帮助信息。

2.2 OpenGL 的函数

OpenGL 的函数大致上分为 6 类。

1. **OpenGL 核心库**：约 150 个函数。函数名前缀为 gl。这部分函数用于常规的、核心的图形处理。由于许多函数可以接受不同数据类型的参数，因此派生出来的函数原形多达 300 多个。
2. **OpenGL 实用库**：约 40 个函数。函数名前缀为 glu。这部分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。如：坐标变换、纹理映射、绘制椭球、茶壶等简单多边形。OpenGL 核心库和 OpenGL 实用库的函数是可以在所有 OpenGL 平台上运行的。
3. **OpenGL 辅助库**：约 30 个函数。函数名前缀为 aux。这部分函数提供窗口管理、输入输出处理以及绘制一些简单三维物体。因此，OpenGL 辅助库的函数不能在所有的 OpenGL 平台上实现。
4. **OpenGL 工具库**：约 30 个函数。函数名前缀为 glut。这部分函数主要提供基于窗口的工具，如：多窗口绘制、空消息和定时器，以及绘制一些较复杂物体的函数。由于 GLUT 的窗口管理函数是不依赖于运行环境的，因此 GLUT 可以在所有 OpenGL 平台上运行。
5. **Windows 专用**：约 16 个函数。函数名前缀为 wgl。这部分函数用于连接 OpenGL 和 Windows 95/NT，以弥补 OpenGL 在文本方面的不足。专用于 Windows 95/NT 环境。
6. **Win32API 函数**：约 6 个函数。函数名无专用前缀。这部分函数用于处理像素存储格式和双帧缓存。专用于 Windows 95/NT 环境。这 6 个函数将替换 Windows GDI 中原有的同样的函数，因此，链接库的清单中 OpenGL 的库必须放在前面。

2.3 Windows 95/NT 下 OpenGL 的结构

OpenGL 命令的解释模式是客户/服务器（Client/Server）方式。图 2-3 描绘了 OpenGL 在本地或网络环境下的工作模式。

应用代码（客户）发出 OpenGL 命令，OpenGL32.DLL 将命令发送到 OpenGL 服务器（即 OpenGL 内核）。由 OpenGL 服务器解释并执行。对客户来讲，服务器可以是在同一台计算机上，也可以不在。从这个意义上说，OpenGL 是网络透明的。一个服务器可以保

持几个 OpenGL 环境，其中的每一个都是一个封装的、独立的 OpenGL 环境。一个客户可以与这些环境中的任一个连接。

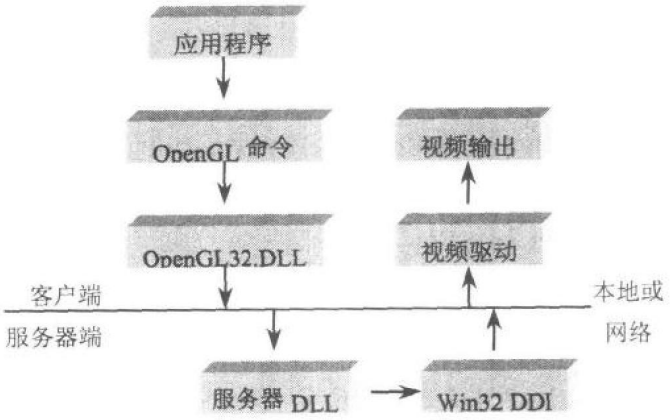


图 2-3 OpenGL 的 C/S 结构

第三章 OpenGL 基础

本章讲述

- OpenGL 的数据类型和函数名后缀
- OpenGL 的简单图元和命令
- OpenGL 的图形控制

3.1 OpenGL 的数据类型和函数名后缀

OpenGL 的数据类型主要是描述三维物体空间位置及其属性的整数和浮点数。虽然 OpenGL 的数据类型可以用其它语言的相应数据类型来表达，但是建议在 OpenGL 编程时采用 OpenGL 定义的数据类型，如：GLint、GLfloat 等。OpenGL 的数据类型与 C 语言数据类型的对照关系见表 3-1。

表 3-1 OpenGL 的数据类型

缩写	数据类型	C 中的数据类型	OpenGL 数据类型
b	8 bit integer	signed char	GLbyte
ub	8 bit unsigned integer	unsigned char	GLubyte,GLboolean
s	16 bit integer	short,int	GLshort
us	16 bit unsigned integer	unsigned short	GLushort
i	32 bit integer	long	GLint,GLsizei
ui	32 bit unsigned integer	unsigned long	GLuint,GLenum,GLbitfield
f	32 bit float-point	float	GLfloat,GLclampf
d	64 bit float-point	double	GLdouble,GLclampd

OpenGL 的库函数名称有一定的规律可寻，象函数名称的前缀，如 gl、wgl 等，前面已经做了介绍，而一个完成某个功能的函数会有若干变种，变种之间的差异只是函数入口参数的数据类型不同。如：

glVertex**2i**(3,1);　　用 **2** 维坐标（**整数**）来描述一个顶点

glVertex**3f**(2.0,3.0,4.0); 用 **3** 维坐标（**浮点数**）来描述一个顶点

由此可见，数字 2，3 代表了数据的维数，其后的字母代表了参数的类型。还有一些函数的后缀多一个字母 v，它表示函数的入口参数是一个矢量，换句话说，就是数组。例如,下面这两种设置颜色的方式是等价的。

glColor**3f**(1.0,0.5,0.5);

```
float ColorArray[]={1.0,0.5,0.5};
glColor3fv(ColorArray);
```

除了以上的基本命名规则外，还有一种“*”的引用方式，相当于文件名中的通配符（Wild Char）。例如：glColor*（）表示所有设置颜色的各种不同变种的函数的总称；而 glVertex*v（）则表示所有使用向量方式定义顶点的函数的总称。

另外，OpenGL 也同样定义了 GLvoid 数据类型，等同于 C 语言中的 void 类型。

3.2 OpenGL 的基本图元和命令

OpenGL 的基本图元指的是点（Point）、线段（Line segment）和多边形（Polygon）。

3.2.1 点（Point）

这里的点指的是顶点，即空间中的一个三维物体的几何顶点。通常用齐次坐标表示（x，y，z，w），w 的默认值为 1.0，z 的默认值为 0.0（即退化为二维坐标）。

3.2.2 线段（Line Segment）

这里的线段与通常几何中描述的线段是一致的，即两端有端点的、有限长度的直线。在 OpenGL 的应用中，若干个线段通常一起使用，构成闭合或不闭合的图案。OpenGL 的线段应用式样见图 3-1。

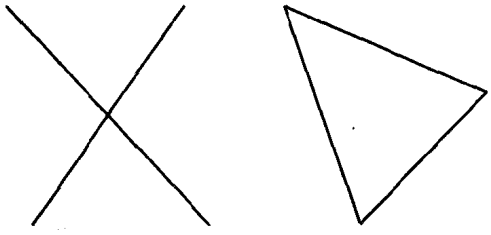


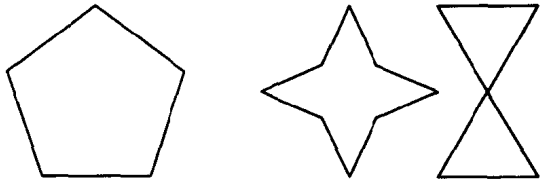
图 3-1 线段

3.2.3 多边形（Polygon）

OpenGL 中定义的多边形是由一系列线段首尾相接而成的封闭区域。它必须是几何中的凸多边形。否则不能被 OpenGL 的函数接受。

验证凸多边形的方法很简单，多边形的任何一边做无限延伸，多边形的其余各边处于该线的同侧，即为凸多边形。OpenGL 中凸多边形与凹多边形的示例见图 3-2。

3.2.4 绘制基本图元



凸多边形　　凹多边形

图 3-2 多边形

定义顶点：在 OpenGL 中，所有的几何物体都要由若干个有序的顶点集合来描述。

glVertex{234}{sifd}[v](TYPE coords);　　//定义二维、三维或齐次坐标的顶点

由此可以看出，glVertex 函数的具体实现形式是非常多样的。例如：

```
glVertex2s(3,8);          //整数定义的二维坐标
glVertex3i(1280,1024);    //长整数定义的三维坐标
glVertex4f(1.,2.,3.,4.);  //浮点数定义的齐次坐标
GLdouble aVertex[3]={2.7,3.14,1.41,1.0};
glVertex3dv(aVertex);     //双精度数组定义的三维坐标
```

构造几何图元：在 OpenGL 的实际应用中，通常是用一组有序排列的顶点集合来构造几何图元，而不是将线段、多边形组合起来构成几何图元。

在 OpenGL 中，同一个几何图元的所有被定义的顶点一起放在 glBegin()和 glEnd()函数之间，同时定义这些顶点之间的关系。例如：

```
glBegin(GL_POLYGON);
    glVertex2s(0,0);
    glVertex2s(0,12);
    glVertex2s(12,16);
    glVertex2s(16,8);
    glVertex2s(8,0);
glEnd();
```

以上这段程序定义了一个多边形的一组顶点，由于在 glBegin()函数中指定了 GL_POLYGON，这些顶点的关系就被限定为多边形的顶点（顺序是依次排列）。

```
glBegin(GLenum mode);
glEnd();
```

函数 glEnd()仅仅表示顶点列表的结束，但是在代码中它必须与函数 glBegin()成对出现。在函数 glBegin()和函数 glEnd()之间，指定的是顶点的信息，包括顶点的空间位置信息及其排列顺序。

函数 glBegin()中可以使用的顶点关系类型见表 3-2。

表 3-2 基本图元定义常量

参数名称	参数说明
GL_POINTS	单个顶点集
GL_LINES	多组独立双顶点线段
GL_POLYGON	单个连线多边形
GL_TRIANGLES	多个独立连线三角形
GL_QUADS	多个独立连线四边形
GL_LINE_STRIP	不闭合折线
GL_LINE_LOOP	闭合折线
GL_TRIANGLE_STRIP	线形的连线三角形串
GL_TRIANGLE_FAN	扇形的连线三角形串
GL_QUAD_STRIP	连续的连线四边形串

在 glBegin()和 glEnd()函数之间，还可以指定顶点的颜色、法向、纹理坐标等信息，只需在该顶点的空间位置定义之前，调用相关的函数即可。可以在函数 glBegin()和 glEnd()之间调用的函数见表 3-3。

表 3-3 glBegin()和 glEnd()之间可以调用的函数

函数名称	函数功能
glCallList()	执行显示列表
glCallLists()	执行显示列表
glColor*()	设置当前颜色
glEdgeFlag*()	控制边界绘制
glEvalCoord*()	产生坐标
glIndex*()	设置当前颜色表
glMaterial*()	设置材质
glNormal*()	设置法向坐标
glTexCoord*()	设置纹理坐标
glVertex*()	设置顶点坐标

下面的这个例子将全面演示绘制图元的方法，并明确地描述函数 glBegin()入口参数的含义。这个例子重点使用 glBegin()和 glEnd()两个函数，并根据函数 glBegin()的各种入口参数一一示范其用法。

```
#include   <windows.h>
#include   <GL\gl.h>
#include   <GL\glaux.h>

void Init( void );
```

```
void CALLBACK Resize( GLsizei w,GLsizei h );
void CALLBACK Paint( void );
void DrawObject( void );

void Init( void )
{    //初始化窗口环境
     glClearColor( 0. ,0. ,0. ,0. );
     glClear( GL_COLOR_BUFFER_BIT );
     glShapeModel( GL_FLAT );
}

void CALLBACK Resize( GLsizei w,GLsizei h )
{    //调整窗口尺寸时的事件响应函数
     glViewport( 0 ,0 ,w ,h );
     glMatrixMode( GL_PROJECTION );
     glLoadIdentity();
     if (w > h) glOrtho( -20. * (GLfloat)h / (GLfloat)w ,20. * (GLfloat)h / (GLfloat)w
          ,-20. ,20. ,-50. ,50. );
     else glOrtho( -20. ,20. ,-20. *   (GLfloat)h / (GLfloat)w ,20. * (GLfloat)h
          / (GLfloat)w ,-50. ,50. );
     glMatrixMode( GL_MODELVIEW );
     glLoadIdentity();
}

void CALLBACK Paint( void )
{    //绘制窗口
     glColor3f( 1. ,1. ,0. );
     DrawObject();
     glFlush();
}

void DrawObject( void )
{
     glBegin( GL_POINTS );              //单个顶点集
          glColor3f( 1. ,0. ,0. );
          glVertex2f( -10. ,11. );
          glColor3f( 1. ,1. ,0. );
          glVertex2f( -9. ,10. );
          glColor3f( 0. ,1. ,1. );
          glVertex2f( -8. ,12. );
     glEnd();

     glBegin( GL_LINES );        //多组独立双顶点线段
          glColor3f( 1. ,1. ,0. );
          glVertex2f( -11. ,9. );
          glVertex2f( -5. ,3. );
          glColor3f( 1. ,0. ,1. );
          glVertex2f( -11. ,3. );
          glVertex2f( -5. ,9. );
     glEnd();
```