第一章　绪论

?计算机图形学 是研究怎样用计算机生成、处理和显示图形的一门学科。

生成：在计算机内表示客观世界物体的模型，即图形建模；

显示：模型对象在计算机显示设备或其他输出设备上的显示；

处理：利用计算机实现客观世界、 对象模型和输出图形这三者之间映射的一系列操作和处理过程。

?1.点阵法 ：枚举出图形中所有的点来表示，强调图形由点及其点的属性（颜色）

构成：像素图、位图或图像。一般地，一个图像就是一个矩阵，该矩阵的每一个元素都表示图像某行某列一个点的颜色值，矩阵的维数就是图像的宽度和高度

缺点：点阵图形需要大量的存储空间；对点阵图形进行编辑、修改较困难；

点阵图的放大操作会使图形失真； JPEG, BMP, Tif, GIF, PNG

2.参数法 ：由图形的形状参数和属性参数来表示图形（矢量图、图形）

形状参数 (必须有)：几何，方程或分析表达式的系数，线段的端点坐标等

属性参数 (可选)：非几何，颜色、线形等 DXF, OBJ, 3DS

?几何要素 ：刻画对象的轮廓、形状、几何元素组成等。

非几何要素 ：刻画对象的颜色、材质、纹理等。

?图像：图像一定是二维的。基本单位是像素：组成图像的颜色点（或亮度点） ，

是数字图像的最小信 息单位，通常是一个整数，其大小称为像素值。

基本参数•图幅参数：也称图像空间分辨率，通常用“行数 列数”表示

?灰度级分辨率、颜色分辨率

图像分为两色图（黑白） 、灰度图、彩色图、真彩色图

图形： 图形可以是二维的、或者三维的，图形的基本信息包括它的基本几何元素（必须），拓扑关系，以及颜色、 材质、纹理等可选要素

第二章　图形系统与图像生成

?计算机图形系统 是进行图形处理的计算机系统，是计算机图形硬件和图形软件的集合。

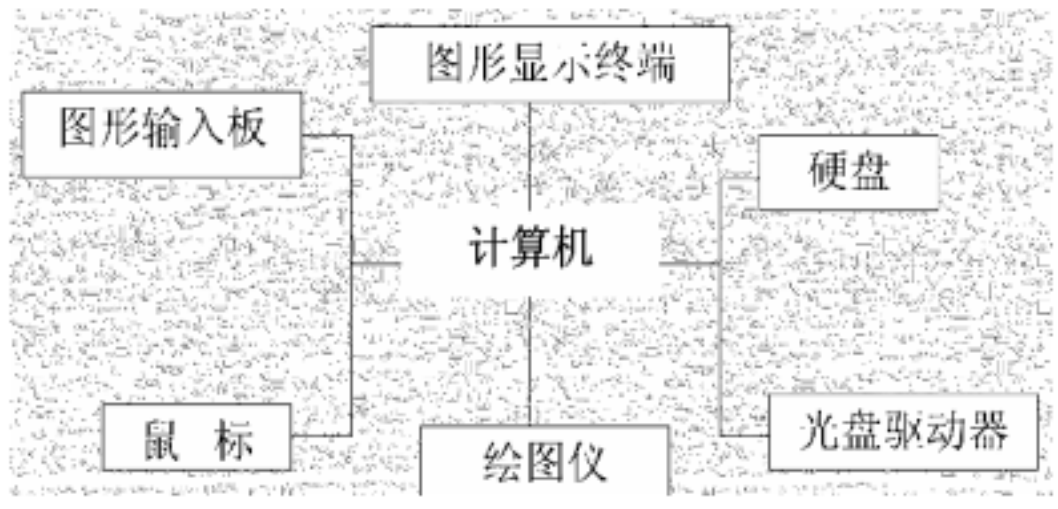
图形硬件包括具有图形处理能力的计算机主机、 图形显示器以及鼠标和键盘等基本交互工具， 还有图形输入板、 绘图仪、图形打印机等输入输出设备， 以及磁盘、光盘等图形存储设备。

图形软件分为图形数据模型、 图形应用软件和图形支撑软件三部分。 涵盖了计算机系统软件、高级语言和专业应用软件等方面。

?一个计算机图形系统至少应当具有 计算、存储、对话、输入、输出 五个方面的基本功能

?图形系统的 硬件 就是指执行以上不同功能的各种设备，如计算机、鼠标、扫描仪、显示器、 硬盘、绘图仪等。 根据具体的业务需求， 组成系统的设备是可选的。在系统中，计算机处于核心地位，其他设备与其直接相连。

星型设备，其他可选



?图形输入设备 从逻辑上分为六种，但实际的往往是某些逻辑输入功能的组合

名称	相应典型设备	基本功能
定位	叉丝、鼠标	输入一个点的坐标
笔划	图形输入板	输入一系列点的坐标
数值	键盘	输入一个整数或实数
选择	功能键、光笔、鼠标	由一个整数得到某种选择
拾取	光笔、鼠标、叉丝	通过一种拾取状态来判断一个显示的图形
字符串	键盘	输入字符串

?图形处理器 ：在图形系统硬件中，为了减轻主机负担，加快图形处理速度，一般都有两个以上的处理器部件， 采用流水线、 并行处理等技术。 除了中央处理器（CPU）之外，还有一个专用的显示处理机（ DPU）,它与 CPU 协同工作，并控制显示设备的操作。

图形处理器是图形系统结构的重要元件，是连接计算机和显示终端的纽带；

早期的图形处理器只包含简单的存储器和帧缓冲区， 它们实际上只起了一个图形的存储和传递作用，一切操作都必须有 CPU 来控制；

现在的图形处理器不单单存储图形， 而且能完成大部分图形函数， 专业的图形卡已经具有很强的 3D 处理能力，大大减轻了 CPU 的负担，提高了显示质量和显示速度。

?图形卡 将用于图形显示的处理器（ DPU）、视频处理控制器、显示处理存储器以及接口电路等集成在一起， 单独做成一块板卡， 称为图形显示适配器 (简称显卡)。

?工作站的特点 ：具有高速的科学计算、丰富的图形功能处理、灵活的窗口界面及网络管理功能的交互式计算机系统。比微机高一个等级的计算机。

1.具有 32 位或 64 位字长的 CPU，广泛采用精简指令系统（ RISC）；

2.配备大容量的内存和外存，运算速度很高，可达 20MIPS 和 5MFLOPS 以上；

3.一般采用 UNIX 及类似的操作系统，配有高性能的窗口管理系统，如 Motif 或 OpenLook 等；

4.具有很强的图形图像处理功能，配有专用的图形图像处理器，大尺寸高分辨率的显示器，如 19 英寸或 21 英寸， 1280X1024 以上的分辨率，颜色深度可达 100 个位面以上；

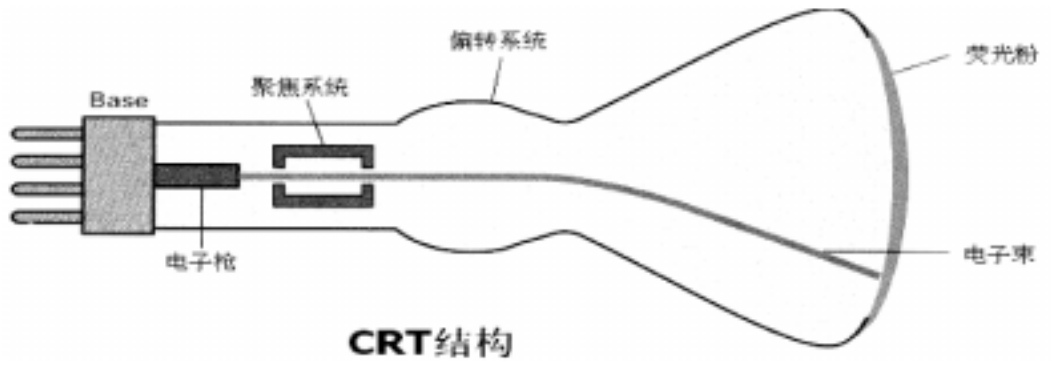
5.具有网络功能，支持 TCP/IP 协议；

6.基本用户是工程和产品的设计师，主要用于工程和产品的设计与绘图、工业模拟和艺术设计等。

从用户角度来看，除工作站比大、中、小型计算机价格便宜外，更主要的是工作站将多种功能集于一身， 体积小，通常配有高分辨率的大屏幕显示器及容量很大的内存储器和外部存储器， 并且具有较强的信息处理功能和高性能的图形、 图像处理功能以及联网功能， 为程序设计人员提供一个功能强大、 使用方便的工作环境。根据工作站本身的特点， 从使用的方便性来讲， 它更类似于 PC 机，有人说：工作站是高档的 PC 机；从功能和性能方面，它越来越多地覆盖了大、中、小型计算机的应用领域。

?**CRT** 工作原理 ：由电子枪发射电子束 (阴极射线)，通过聚焦系统（电子透镜）和偏转系统，射向涂覆荧光层的屏幕上的指定位置。在电子束冲击的每个位置，荧光层发出一个小亮点，从而产生可见图形。

?**CRT** 结构功能 CRT 主要由电子枪、 聚焦系统、偏转系统、和荧光屏四部分组成。



CRT结构

?彩色 CRT 和单色显示器的不同是由于荧光粉的缘故。目前大多使用的是荫罩式彩色 CRT，荫罩是安装在荧光屏的内侧的上面刻有 40 多万个孔的薄钢板。荫罩孔的作用在于保证三束电子共同穿过同一个荫罩孔， 准确的激发荧光粉， 使之发出红、绿、蓝三色光。 CRT 产生彩色显示有两种技术：电子束穿透法，荫罩法。

?光栅扫描原理 ：光栅扫描方式中，电子束总是不断地从左至右、从上到下反复扫描整个屏幕， 在扫描过程中， 只要在对应时刻在对应位置控制电子束的强度就能显示所要的图形。 电子束横向从左到右扫描一次称为一条扫描线， 在每条扫描线末端，电子束返回到屏幕的左边， 又开始显示下一条扫描线。 一帧图像是光栅显示系统执行一次全屏幕循环扫描（一次屏幕刷新）所产生的图像。

?光栅扫描式显示器的特点 :光栅扫描显示器是画点设备，可以看作是一个点阵单元发生器， 并可控制每个点阵单元的亮度。 它不能直接从单元阵列中的一个可编址的像素画一条直线到另一个编址的像素， 只能用尽可能靠近这条直线路径的像素点集来近似地表示这条直线。 为了得到稳定的画面， 光栅扫描显示器每秒要刷新。通常刷新频率至少应为 30 帧/每秒。随着刷新频率的降低，会出现闪烁。

?屏幕上每个光点维持发光的时间一般在毫秒或微秒数量级之间，荧光屏的亮度随着时间按指数衰减， 整个画面必须在一秒钟内重复显示， 才能得到稳定而不闪烁的图形， 所以必须重复使荧光物质发光， 即使电子束迅速地回到同一点。 这称为屏幕刷新。

?屏幕坐标 ：光栅扫描显示器的屏幕可分为 m 行扫描线， 每行扫描线有 n 个像素。这样整个屏幕就分为 m*n 个像素， m*n 就是显示器的 分辨率 。

为了得到稳定的画面，光栅扫描显示器每秒要刷新。通常刷新频率至少应为 30 帧/每秒。随着刷新频率的降低，会出现闪烁。

?要显示世界坐标系中指定对象的几何形状，就需要调整数学输入点到有限像素区域的映射；映射方法有两种：

一是按对象边界与像素区域的覆盖量来调整显示物体的尺寸，即对象与像素中心对准； 二是对象映射到像素间的屏幕位置，以使物体边界与像素边界对准。

?光栅扫描生成的图像所有像素的强度值都要存放在一块连续的存储器中，这个存储器称为帧缓冲器或刷新存储器，俗称显示存储器。

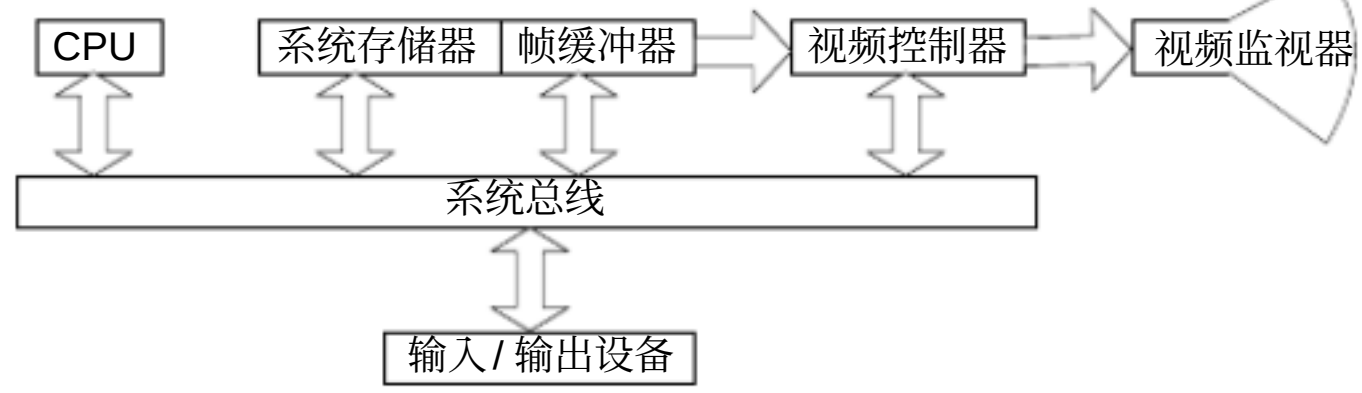
光栅扫描系统的帧缓冲器对屏幕上每一点都有存储强度/颜色信息的能力。帧缓冲器的单元个数至少与显示器能显示的像素总数相同， 且存储单元一一对应于可寻址的屏幕像素位置。

?对于黑白单灰度显示器每一象素需要一位存储器， 对一个 1024×1024 象素组成的黑白单灰度显示器所需要的最小缓存为 220，并在一个位面上。一个位面的缓存只能存储黑白图象。

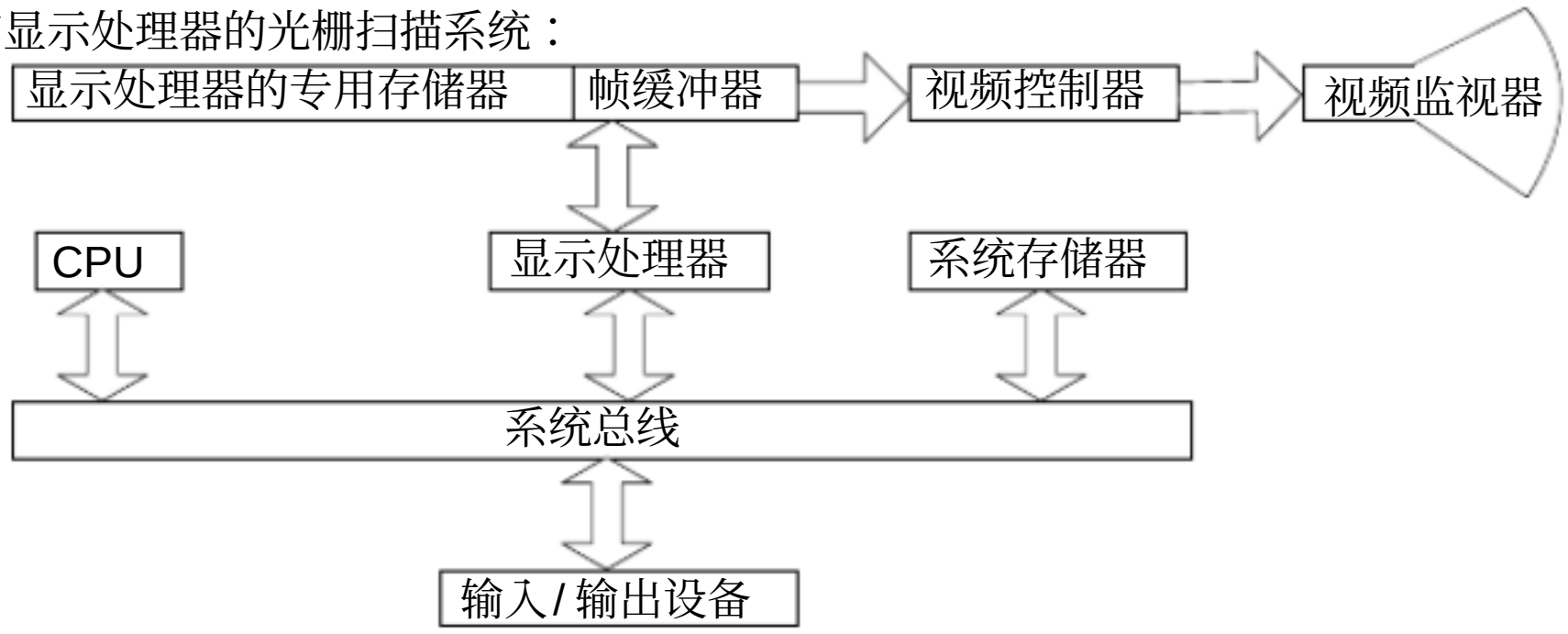
?3 个位面、分辨率 1024*1024 像素阵列的显示器，需 3*1024*1024 bit 的存储器

?每种原色电子枪有 8 个位面的帧缓存和 8 位的数模转换器，每种原色可有 256 种灰度，三种原色的组合将是 (2^8)^3=2^24，分辨率是 1024*1024 个像素阵列的显示器，帧缓存 1024*1024*24

?彩色查找表 或颜色索引技术是不断增加缓冲器存储容量而得到更多颜色的一种技术，它在帧缓冲器与显示屏的数　/模转换器之间加一个查色表。
彩色查找表可看成是一维线性表，每一项　(元素) 对应于一种颜色。　帧缓冲器中每个单元存储的是对应于某一像素颜色在彩色查找表中的地址，而不是颜色值。
　　彩色表的地址长度由帧缓冲器每个存储单元的位数决定，这确定了一幅画面能同时显示的颜色种类数。　彩色表的元素位长由帧缓冲器每个存储单元的基色数决定，这决定了显示器可选择的显示的颜色种类总数。
?光栅扫描系统：交互式光栅图形系统通常使用几个处理部件。除了中央处理器（cpu）以外还使用一个视频控制器或显示控制器来控制显示设备的操作。
帧缓冲器可在系统存储器的任意位置，视频控制器访问帧缓冲器，以刷新屏幕。
除了视频控制器，　更复杂的光栅系统运用其它处理器作协处理器和加速器，　　并执行各种图形操作。
最简单的光栅扫描系统：



帧缓冲器使用系统存储器的固定区域，　视频控制器主要用于屏幕的基本刷新操作
计算机将要显示的图形、　图像转化为位图，　经过接口电路送入帧缓存，　视频控制器控制电子束依照固定的扫描顺序，自上而下，从左到右扫描整个屏幕，同时，把一帧画面中的每个像素的值从帧缓存中读出。读出的值控制电子束能量的大小，决定像素的亮度。
具有显示处理器的光栅扫描系统：



除了系统存储器外，　还提供独立的显示处理器存储区域。　上图表示了建立包含独立显示处理器的光栅扫描系统的结构。
显示处理器的主要任务是将应用程序给出的图形定义数字化为一组像素强度值，存放在帧缓冲器，这个过程称扫描转换。
显示处理器的用途是使　cpu 从图形杂务中解脱出来。　除了系统存储器外，　还提供独立的显示处理器存储区域。上图表示了建立包含独立显示处理器的光栅扫描系统的结构。
?显示处理器 的主要任务是 将应用程序给出的图形定义数字化为一组像素强度值，存放在帧缓冲器 ，这个过程称 扫描转换 。例如，直线段的扫描转换意味着必须确定最接近于直线路径的像素位置，并把每个位置的强度值存入帧缓冲器。
?显示配置是指 显示器 和 显示卡 这两方面的内容。显示器又叫监视器。最简单显

示卡将显示器控制适配器， 它将显示处理器、 存储器、 显示控制器制作在一块板卡或者芯片上，它与显示器一起构成一个显示系统。

?从显示标准的角度说，每一种标准都包含有一种或多种 显示模式 ，每一种显示模式都规定了模式的 类型、字符尺寸、字符格式、屏幕分辨率、色彩 等指标。

PAL, NTSC

?显示卡又名显示适配器，当前显示卡主要由 显示芯片、显示内存、 **RAMDAC** 芯片、输入输出系统接口、显示器插座、连接主板总线的接口 等组成

?显示主芯片

图形处理器，显卡的核心，俗称 GPU（类似 DPU），它的主要任务是对系统输入的视频信息进行构建和渲染

视频控制器，建立帧缓存与屏幕像素之间的一一对应，负责刷新屏幕

显示缓存

用来存储将要显示的图形信息以及保存图形运算的中间数据

显存的大小和速度直接影响着主芯片性能的发挥

数字模拟转换器（ **RAMDAC** ）

它的作用就是把二进制的数字转换成为和显示器相适应的模拟信号

?显示卡的作用 **:**将 CPU 送来的图像信号经过处理后输送至显示器， 这个过程通常由四个步骤 1) CPU 将数据通过总线传送到显示芯片。

2) 显示卡上的芯片对数据进行处理，并将处理结果存放在显示卡的内存中。

3) 显示卡从内存中将数据传送到数 /模转换器进行数 /模转换。

4) 数/模转换器将模拟信号通过 VGA 接口输送到显示器。

?扫描转换

大多数图形学应用中采用点、线、面表示来建立图形对象模型，然而，对于光栅图形显示而言，点、线、面表示又不能直接用于显示，因为在光栅显示器上的任何一种图形，实际上都是一些具有一种或多种颜色的图象 (像素的集合)。

将图形对象表示转换成点阵表示， 即确定一个像素集合及其颜色， 用于显示一个图形的过程， 称为图形的 扫描转换或光栅化 。也就是说， 从图形对象的几何信息出发，求出构成图形对象的各个像素， 并将其颜色值写入帧缓冲器中相应的单元。一般基本几何元素由显示卡的显示处理器完成图形到图像的转化。

第三章 图形编程基础

?**OpenGL** 特点

OpenGL 是由 SGI 公司设计的一套底层三维图形 API。它不是一种编程语言，而是提供了一些预封装的函数， c 语言可以调用这些函数。

OpenGL 是一个开放图形库， 目前在 Windows、MacOS、OS/2、Unix/X-Windows 等系统下均可使用， 因此具有良好的可移植性， 同时调用方法简洁明了， 深受好评，应用广泛。

OpenGL(Open Graphics Library) 是独立于操作系统和硬件环境的三维图形软件库。由于其开放性和高度的可重用性，目前已成为业界标准 ．很多优秀的软件都是以它为基础开发出来的， 象著名的产品有动画制作软件 3DMAX，Soft Image, VR 软件和 GIS 软件等等

工业标准

?**OpenGL** 功能：1.几何建模 ,2.坐标变换 ,3.颜色模式设置 ,4.光照和材质设置 ,5.图像功能 ,6.纹理映射 ,7.实时动画

?**OpenGL** 主要包括三个函数库：

OpenGL 核心库 GL：其中包含了 OpenGL 最基本的命令函数。这些函数都以`gl`为前缀。（115 个）

OpenGL 实用库 GLU：它是比 OpenGL 核心库更高一层的函数库，实用程序库中的所有函数都是由 OpenGL 的核心库函数编写。函数都以 `glu`为前缀。（43 个）

OpenGL 辅助库 AUX：提供了一些基本的窗口管理函数、事件处理函数和简单的事件函数。函数都以 ‘aux’为前缀。（31 个）

?处理窗口和输入：

void glutDisplayFunc(void(*func)(void));

该函数用于绘制当前窗口。参数 void(*func) 为绘制当前窗口时所调用的函数名。任何时候当窗口的内容需要被重新绘制，则调用该函数。

void glutReshapeFunc(void(*func)(int width,int height));

表示在窗口尺寸改变时，指定了所调用的函数。 width, height 指定了窗口新的宽度和高度。

?运行程序

void glutMainLoop(void); 这个函数开始启动主 GLUT 处理循环。事件循环包括所有的键盘、鼠标、绘制及窗口的事件等。

?glColor3f(1.0,1.0,1.0) 命令，

gl 前缀代表该函数来自 gl 库（说明该函数来自哪个库）；

根命令 Color 代表对颜色的设定；

后缀 3f 表示该函数需要 3 个浮点型的参数；（后缀中的 数字表示该函数需要的参数个数，字母表示该函数的需要的参数类型）

Glfloat color[]={1.0,1.0,1.0}

glColor3fv(color);

若函数的后缀中带有一个字母 v,表示该命令带有一个指向矢量或数组值的指针作为参数。

? 编译 OpenGL 程序需要头文件 gl.h，glu.h，GLAUX.h，库 opengl32.lib,glu32.lib, GLAUX.lib, 若使用 GLUT 还需要 glut.h 和 glut32.lib。

运行 OpenGL 程序，需要在 windows\system 目录下有动态连接库 opengl32.dll 和 glu32.dll, GLAUX.dll，若使用 GLUT 还需要 glut32.dll。

?函数 glViewport() 定义窗口内的作图区域（即视区）。函数如下：

void glViewport(GLint x,Glint y,Glsizei width,Glsizei height);

功能：设置窗口中的作图区域，用于 OpenGL 绘图。

参数说明：(x,y)为绘图区域左下角的坐标，参数 width 和 height 为绘图区宽度和高度的像素数，即视区的尺寸。

?所有的几何图最终都是通过一组有序顶点来描述的。

?OpenGL 中有十种基本图元，从空间中绘制的简单的点到任意边数的封闭多边形。用 glBegin 命令可告诉 OpenGL 开始把一组顶点解释为特定图元。然后用 glEnd 命令结束该图元的顶点列表。

void glBegin(Glenum mode);

此函数标志描述一个几何图元的顶点列表的开始。图元的类型由 mode 来决定。共有 GL_POINTS,GL_LINES,GL_LINE_STRIP 等十种图元

void glEnd(void);

此函数标志着顶点列表的结束。

?平移：将物体沿直线路径从当前坐标位置移到另一个坐标位置的重定位过程。

$$[x' \quad y' \quad 1] = [x \quad y \quad 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ tx & ty & 1 \end{bmatrix}$$

平移是不产生变形而移动物体的刚体变换，　即物体上的每一点移动相同量的坐标

?二维物体 旋转 是指将物体沿某个定点转动某个角度的重定位过程

$$[x' \quad y' \quad 1] = [x \quad y \quad 1] \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

旋转变换是不产生变形地移动物体的刚体变换。　物体上所有的点旋转相同的角度

void glRotated(GLdouble angle,GLdouble x,GLdouble y,GLdouble z);

void glRotatef(GLdouble angle, GLfloat x, GLfloat y,GLfloat z);

angle指定逆时针方向旋转的角度，　范围 0~360°，xyz 与原点相连，产生旋转轴。对于二维图形，由于只能在　xy 平面内旋转。因此只能绕　z 轴旋转，即相当于在二维平面内绕原点旋转。这时参数可取为：　x=0,y=0,z=1。

angle角度：±，旋转轴：　xyz，eg：(30,1,0,0)绕 x 轴逆时针旋转　30°

?比例 x ′= sx *x，y ′= sy*y

$$[x' \quad y' \quad 1] = [x \quad y \quad 1] \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

void glScaled(GLdouble x,GLdouble y,GLdouble z);

void glScalef(GLfloat x,GLfloat y,GLfloat z);

x,y,z 分别表示对象沿三个坐标轴缩放的比例因子。　对于二维图形　z 坐标值等于　0。

glScale(x,y,0)所生成的二维变换矩阵为　:$T = \begin{bmatrix} x & 0 & 0 \\ 0 & y & 0 \\ 0 & 0 & 1 \end{bmatrix}$

glScale(1,-1,0)　　glScale(-1,-1,0)

?对称：对称变换也称为反射变换，相对于反射轴的对称变换是通过将物体绕反射轴旋转　180°而生成的。它的基本变换包括对坐标轴、原点和　　45°线的变换。

X 轴 $[x' \quad y' \quad 1] = [x \quad y \quad 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Y 轴 $[x' \quad y' \quad 1] = [x \quad y \quad 1] \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Z 轴 $[x' \quad y' \quad 1] = [x \quad y \quad 1] \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Y=X $[x' \quad y' \quad 1] = [x \quad y \quad 1] \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Y=-X $[x' \quad y' \quad 1] = [x \quad y \quad 1] \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

?错切变换也称为错位或错移变换，变换结果将使目标图形失真。

沿 x 方向的错切： 变换的结果是使图形的 y 坐标不变， 而 x 坐标有一个增量， 图形沿 x 轴产生错切变形。 $x'=x+cy \quad y'=y$ 其中， $c=tg\,\alpha$

沿 x 方向的错切：当 c>0 时，如 y>0，沿 +x 方向错切； y<0，沿 -x 方向错切。

当 c<0 时，如 y>0，沿 -x 方向错切； y<0，沿 +x 方向错切。

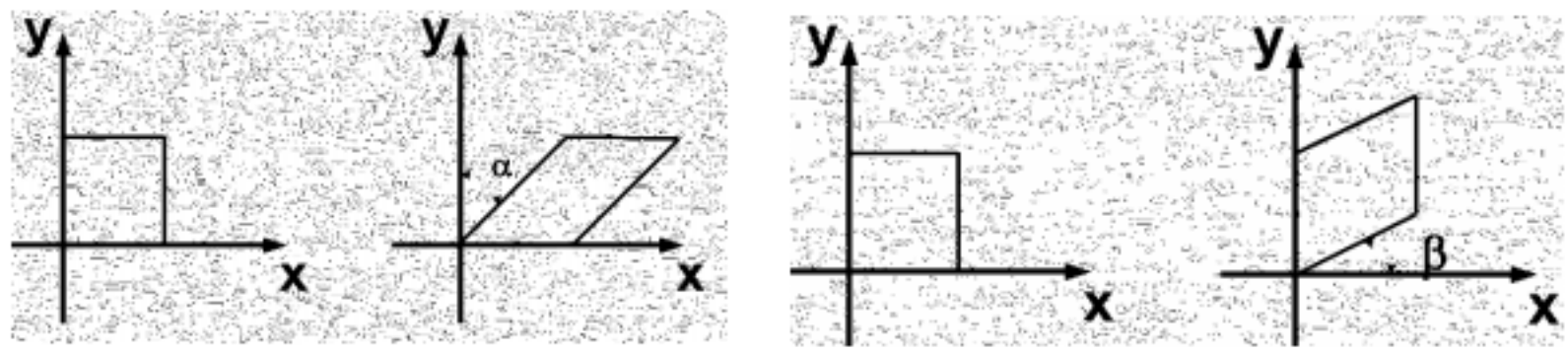
$$[x' \quad y' \quad 1] = [x \quad y \quad 1] \begin{bmatrix} 1 & 0 & 0 \\ c & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

沿 y 方向的错切： 变换的结果是使图形的 x 坐标不变， 而 y 坐标有一个增量， 图形沿 y 轴产生错切变形。 $x'=x \quad y'=bx+y$ 其中， $b=tg\,\beta$

当 b>0 时，如 x>0，沿 +y 方向错切； x<0，沿 -y 方向错切。

当 b<0 时，如 x>0，沿 -y 方向错切； x<0，沿 +y 方向错切。

$$[x' \quad y' \quad 1] = [x \quad y \quad 1] \begin{bmatrix} 1 & b & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



?复合变换 :复合变换矩阵 T 也可由一系列基本几何变换矩阵 Ti 的乘积来表示，即：

T= T1 ?T2 ? ?Tn

例 1：平面图形绕任意点 P(xp,yp) 旋转 角，可以经过以下几个步骤实现：

1.将旋转中心移到原点，变换矩阵为： $T_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_p & -y_p & 1 \end{pmatrix}$

2.将图形绕原点旋转　θ角，变换矩阵为：　$T_2 = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$

3.将旋转中心平移回原来的位置，变换矩阵为：　$T_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_p & y_p & 1 \end{pmatrix}$

例 2：平面图形相对于任意点　P(xp,yp)作比例变换可通过以下几个步骤来完成：

1.将 p 点平移到坐标原点，变换矩阵为：　$T_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_p & -y_p & 1 \end{pmatrix}$

2.关于原点作比例变换，变换矩阵为：　$T_2 = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$

3.将旋转中心平移回原来的位置，变换矩阵为：　$T_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_p & y_p & 1 \end{pmatrix}$

?图形的二维观察是通过指定一个在图形中要显示的部分以及在显示器显示位置，并执行从世界坐标系到设备坐标系的图形变换及删除位于显示区域范围以外的图形部分而实现的。

?　世界坐标系：用户用来定义图形的坐标系称为世界坐标系，也称为用户坐标系；它的定义域是实数域，二维的世界坐标系通常用　$o_w x_w y_w$ 表示。

设备坐标系：在显示器和绘图仪等图形的输出设备上也有一个自身的坐标系，该坐标系称为设备坐标系或物理坐标系，简称为　DC。设备坐标系是一个二维坐标系，对于显示器而言它的度量单位是像素。

规格化设备坐标系：它独立于设备坐标系和世界坐标系，又可容易地转变成设备坐标系的一个坐标系，是一个中间坐标系。在该坐标系中，　x 和 y 轴的坐标被规格化为 0~1 间的量。

?在世界坐标系中要显示的区域称为窗口；窗口映射到显示设备上的坐标区域称为视区。标准窗口与视区一般都采用矩形，其各边分别与坐标平行。

窗口定义了我们要显示的内容。

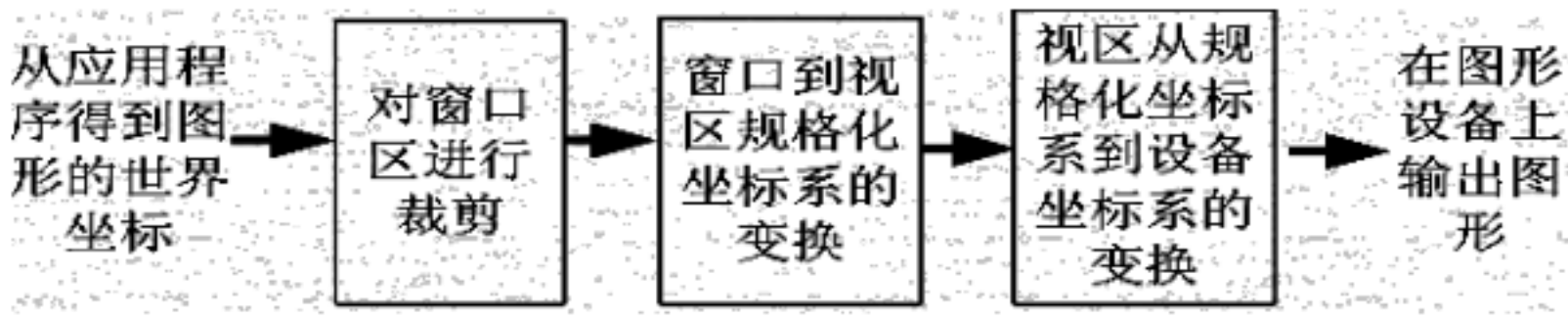
视区决定在显示设备上的显示位置，我们可以在输出设备的不同位置观察物体；也可以通过改变视区的尺寸来改变显示对象的尺寸和位置。

窗口和视区分别处在不同的坐标系内，它们所用的长度单位及大小位置等不同。因此要将窗口内的内容在视区中显示出来，　必须经过从窗口到视区的变换处理，这种变换即称为观察变换

?二维观察变换流程：①在世界坐标系中生成图形②裁剪，得到要显示的内容

③窗口到视区的变换：窗口—规格化设备坐标系中的视区—设备坐标系中的视区

④显示图形



?识别图形在指定区域内或指定区域外的过程称为裁剪。

? gluOrtho2D(0.0, 0.0, (GLdouble)w,(GLdouble)h) 该函数定义窗口的大小。

左上角 x，y； 窗口 w，h 世界坐标系

glViewport(0， 0,(GLsizei)w,(GLsizei)h) 该函数定义视口的大小 屏幕

?三维几何变换

1.平移

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix} \qquad \begin{aligned} x' &= x + t_x \\ y' &= y + t_y \\ z' &= z + t_z \end{aligned}$$

2.比例

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \begin{aligned} x' &= s_x \cdot x \\ y' &= s_y \cdot y \\ z' &= s_z \cdot z \end{aligned}$$

3.旋转

绕 z 轴旋转：

其对应的变换矩阵可表示为：

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \\ z' &= z \end{aligned}$$

绕 x 轴旋转

其对应的变换矩阵可表示为：

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \begin{aligned} y' &= y\cos\theta - z\sin\theta \\ z' &= y\sin\theta + z\cos\theta \\ x' &= x \end{aligned}$$

绕 y 轴旋转

其对应的变换矩阵可表示为：

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \begin{aligned} z' &= z\cos\theta - x\sin\theta \\ x' &= z\sin\theta + x\cos\theta \\ y' &= y \end{aligned}$$

4.对称：①关于坐标平面对称：空间中一点关于 xoy 坐标平面的对称变换：点的 x 和 y 向的坐标保持不变，只改变 z 坐标的正负号

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1]\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = [x \quad y \quad -z \quad 1]$$

②关于坐标轴对称：空间中一点关于 x 轴作对称变换时，则点的 x 坐标保持不变，只改变 y 和 z 方向的坐标的正负号

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1]\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = [x \quad -y \quad -z \quad 1]$$

5.错切：沿 x 轴错切：

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1]\begin{pmatrix} 1 & 0 & 0 & 0 \\ d & 1 & 0 & 0 \\ g & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = [x+dy+gz \quad y \quad z \quad 1]$$

?例三维物体绕平行于 x 轴的直线旋转 θ 角

将旋转轴平移至 x 轴，将物体绕旋转轴旋转 θ 角，将旋转轴平移至最初位置

?三维投影变换 ：把三维物体变为二维图形表示。投影： 平行投影、透视投影

?在平行投影中，物体的坐标位置沿平行线变换到投影平面上。

?在透视投影中，定义投影平面后的一点为投影中心，将投影中心和三维物体上的各点的连线称为投影线。投影线与投影面的交点即称为各点的投影。

?平行投影保持物体的 大小比例不变， 物体各个面的精确视图 可有平行投影得到，但无法给出三维物体的真实性表示 。

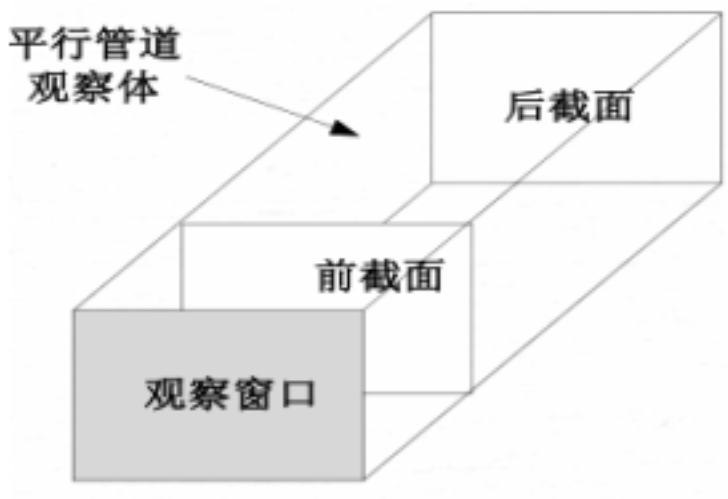
根据平行投影与投影平面的间的夹角也可分为 正平行投影 与斜平行投影 两类，当投影线垂直于投影平面时，得到的投影称为正平行投影；否则为斜平行投影。

?平行投影中，物体投影大小与物体距投影面的距离无关，与人的视觉成像不符。而在透视投影中， 离投影面近的物体较远的物体生成的图像大， 生成真实感图形。

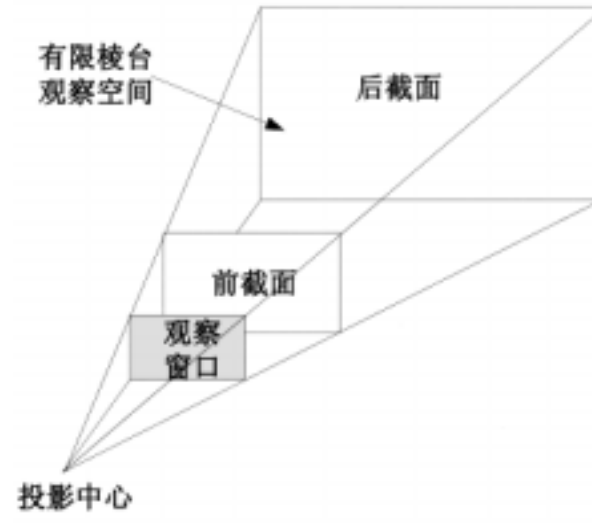
?投影中心又称为 视点，相当于观察者的眼睛。投影面置于视点与物体之间，将物体上的各点与视点相连所得的 投影线与投影面的交点 就是三维物体上相应点的透视变换结果。

?观察空间 ：定义了观察窗口的大小，可以利用窗口边界来定义观察空间。只有在观察空间中的物体才会在输出设备中显示，而其它的景物将被裁剪掉。

平行投影观察空间：无限长的长方体管道；透视投影观察空间：无限长的棱锥

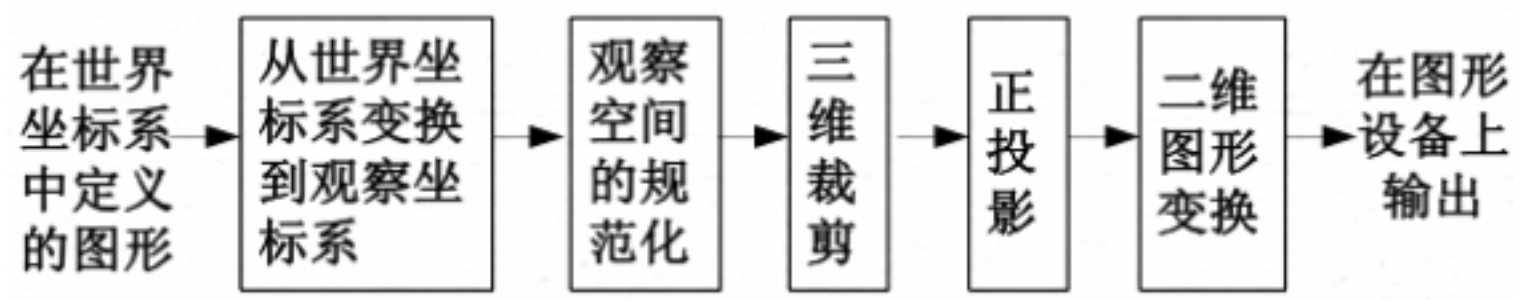


正投影的有限观察空间



透视投影的有限观察空间

?三维观察流程：



?OpenGL 中的图形变换分为四类：视点变换、模型变换、投影变换、视区变换(还有裁剪变换)这些图形变换函数都是通过矩阵操作来实现的。

?入栈函数： void glPushMatrix(void)　　功能：把当前的矩阵拷贝到栈中；

出栈函数： void glPopMatrix(void)　　最后压入栈的矩阵恢复为当前矩阵。

作用：保护当前矩阵现场。

?void glTranslate*(TYPE x,TYPE y,TYPE z);　该函数以平移矩阵乘当前矩阵

?void glRotate*(TYPE angle,TYPE x,TYPE y,TYPE z); 该函数以旋转矩阵乘当前矩阵，其中： angle 指定旋转的角度（以度为单位） 。

x,y,z 指定旋转轴向量的三个分量 (该向量位于世界坐标系中)。

?视点变换也可以称为视图变换，是指改变对象观察点的位置和方向。

视图变换改变视点的位置和方向，也就是改变视觉坐标系。

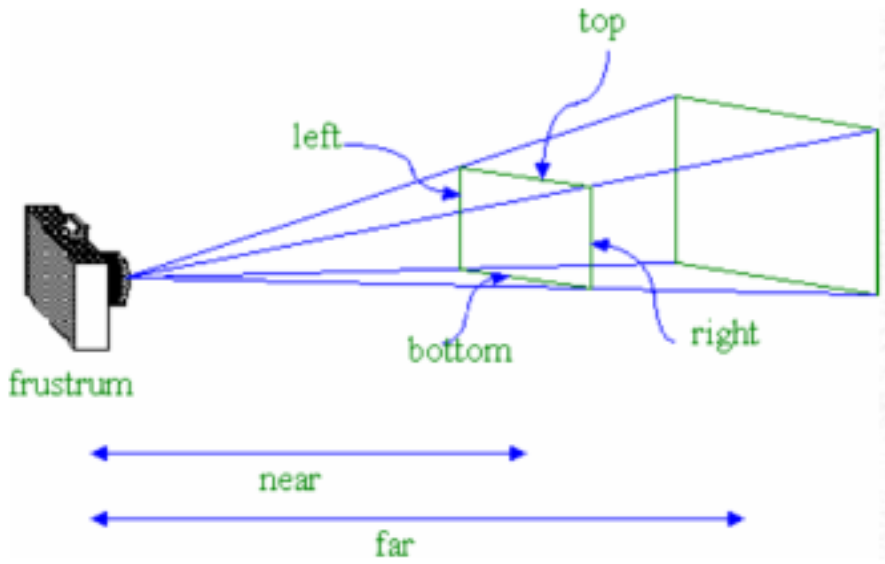
在世界坐标系中，视点和物体的位置是一个相对的关系，对物体作一些平移、旋转变换，必定可以通过对视点作相应的平移、 旋转变换来达到相同的视觉效果。

?投影变换：透视投影，正交 (平行)投影

透视投影：其取景体积是一个截头锥体，在这个体积内的物体投影到锥的顶点，用 glFrustum()函数定义这个截头锥体， 这个取景体积可以是不对称的， 计算透视投影矩阵 M，并乘以当前矩阵 C，使 C=CM。

void glFrustum(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far);　　该函数以透视矩阵乘当前矩阵

left，right 指定左右垂直裁剪面的坐标。 bottom,top 指定底和顶水平裁剪面的坐标。 near,far 指定近和远深度裁剪面的距离，两个距离一定是正的。



?视区变换：视口就是窗口中矩形绘图区。用窗口管理器在屏幕上打开一个窗口时，已经自动地把视口区设为整个窗口的大小， 可以用 glViewport 命令设定一个较小的绘图区， 利用这个命令还可以在同一窗口上同时显示多个视图， 达到分屏显示的目的。

void glViewport(Glint x,Glint y,Glsize width,Glsize height);

该函数设置视口的大小。

x,y 指定视口矩形的左下角坐标（以象素为单位） ，缺省值为（ 0,0)。

width,height 分别指定视口的宽和高。

?曲线曲面的参数化表示：

曲线的参数化表示：空间曲线上一点　P 的每个坐标被表示为某个参数　u 的函数：

$$\begin{cases} x = x(u) \\ y = y(u) \\ z = z(u) \end{cases} \qquad p(u) = [x(u) \quad y(u) \quad z(u)]$$

曲面的参数化表示：曲面可用参数　u、v 的矢函数　P=P(u,v)描述。曲面的范围常用两个参数的变化区间所表示的　uv 参数平面上的一个矩形区域 $\begin{cases} u_1 \le u \le u_2 \\ v_1 \le v \le v_2 \end{cases}$ 给出

曲面上一点 P 的每个坐标被表示为参数　u、v 的函数：$\begin{cases} x = x(u,v) \\ y = y(u,v) \\ z = z(u,v) \end{cases}$

$$p(u,v) = [x(u,v) \quad y(u,v) \quad z(u,v)]$$

?曲线曲面的基本类型：

规则曲线和规则曲面　:圆、抛物线、螺旋线等曲线和球、圆柱、圆锥等曲面都很容易用数学方程式表示出来，这类曲线和曲面分别称为规则曲线和规则曲面。

自由曲线和自由曲面　：曲线和曲面的形状相当自由又不规则，　如飞机机翼、 汽车车身、人体外形、卡通形象等，很难用数学式表示。

什么事自由曲线 /曲面？一种非解析表达的曲线　/曲面，属于参数曲线。

?自由曲线 /曲面的生成：自由曲线和自由曲面一般　通过少数分散的点生成　。常用三种类型的点：

控制点　:用来确定曲线和曲面的位置与形状，　相应的曲线和曲面不一定经过的点。

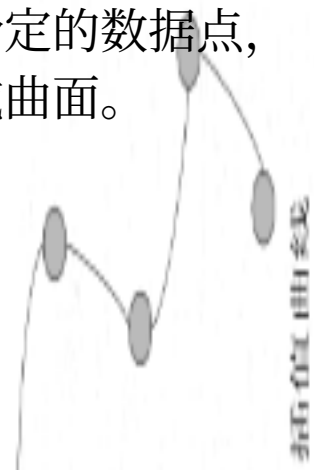
型值点　:用来确定曲线和曲面的位置与形状，　而相应的曲线和曲面一定经过的点。

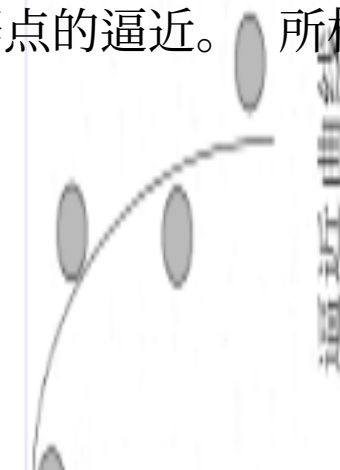
插值点　：为提高曲线和曲面的输出精度，在型值点之间插入的一系列点。

然后根据应用要求得到最贴近这些点（即　少数分散的点 ）的函数式描述

?插值曲线 /曲面　：利用数学方法构造的曲线曲面按要求通过已知的型值点，称为对这些已知型值点进行插值。　所构造的曲线或曲面称为揑值曲线或曲面。　当型值点较多时，构造的插值函数通过所有的型值点是相当困难的。

逼近曲线 /曲面　：人们往往选择一个次数较低的函数，使其在某种意义上最为接近于给定的数据点，　称之为对这些数据点的逼近。所构造的曲线或曲面称为逼近曲线或曲面。





插值和逼近统称为拟合

?三次 Hermite 样条： Hermite 插值样条是一个分段三次多项式，并在每个控制点有给定切线。 它的每个曲线段仅依赖于端点约束，　可以通过调整每段曲线两端点切矢量来控制曲线的形状。

下图所示为在控制点 pk 和 pk+1 之间的曲线段。 rk 和 rk+1 是在控制点 pk 和 pk+1 处相应的导数值（曲线的斜率） 。

Hermite 曲线的边界条件是：

$$\begin{cases} P(0) = P_k \\ P'(0) = r_k \\ P(1) = P_{k+1} \\ P'(1) = r_{k+1} \end{cases}$$

?贝塞尔曲线由一组控制点构成的特征多边形唯一的定义。

贝塞尔曲线的起点与终点和特征多边形的起点与终点重合， 且多边形的第一条边和最后一条边表示了曲线在起点和终点处的切矢方向。

曲线的形状趋向于控制多边形的形状，可以通过调整顶点的位置来控制曲线的形状。

给定 n+1 个控制顶点的位置， 则贝塞尔曲线可表示为： $C(u) = \sum_{i=0}^{n} P_i B_{i,n}(u)$

其中 Pi 构成了该曲线的特征多边形， PiBi,n 是伯恩斯坦基函数，贝塞尔曲线是控制顶点 Pi 关于伯恩斯坦基函数的加权和。

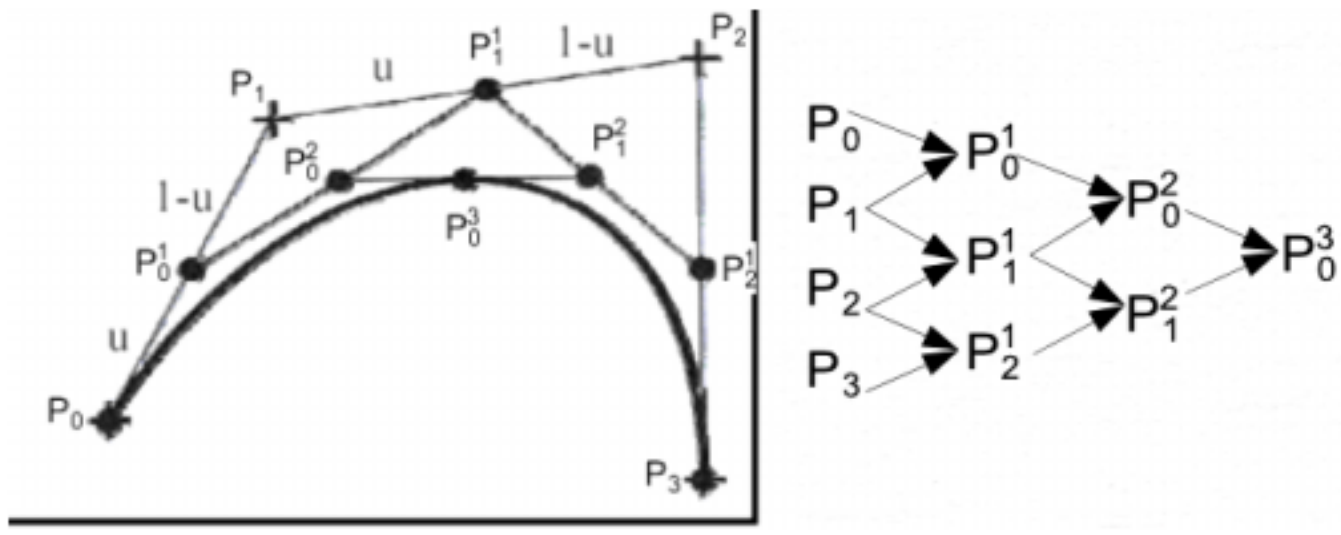
?贝塞尔曲线的递推算法： Bezier 的递推算法，即德卡斯特里奥算法

对于一给定参数 u [0,1], 已知控制点 P_i， i=0,1,⋯,n,构成 n 条边的控制多边形。

求其在 Bezier 曲线上对应的点 C(u)。

①依次对控制多边形进行定比分割， 分割比例为 u:(1-u)，得到第一级递推的中间顶点 P_i^1， i=0,1,2,...,n

②再对上述得到的这些中间顶点构成的控制多边形执行同样的定比分割， 得到第二级递推的中间顶点 P_i^2， i=0,1,2,...,n。如此继续，直到 n 次分割得到一个中间顶点 P_i^n，即为所求的 Bezier 曲线上对应的点 C(u)。



?Bezier 曲线有以下缺点：

不能作局部修改，修改某一控制顶点将影响整条曲线；

控制多边形的顶点数决定了 Bezier 曲线的阶次，即 n+1 个控制点必将产生一段 n 次的 Bezier 曲线。当控制顶点的数目 n+1 较大时，由其构成的特征多边形对曲线形状的控制也将减弱。 **Bezier** 曲线是 **B** 样条的特例

?以 B 样条基函数代替 Bernstein 基函数而获得的 B 样条曲线曲面克服了上述缺点。 B 样条仍采用控制点来定义曲线，曲线方程可写为： $C(u) = \sum_{i=0}^{n} P_i B_{i,k}(u)$

?要消除上述二义性，在显示三维图形物体时，就要决定物体上每条棱边在视图中的位置和它的可见性，　对不可见的就必须加以抹掉或区别。　这种找出并消除物体中不可见的部分的过程，　称为消隐图。 若消除的是物体上不可见的线段，　即棱边，称为 线消隐 。若消除的是物体上不可见的面，就称为　面消隐 。隐藏面消除算法也称为 可见面判别算法。

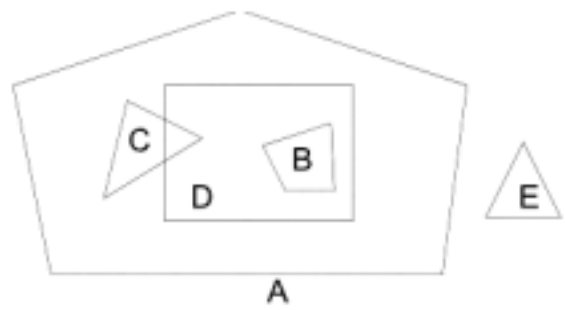
?可见面辨别算法：后向面判别，画家算法，区域细分算法

后向面判别：多面体的表面可分为内侧和外侧。对观察者而言，外侧是可见的，而内侧不可见。 我们可利用表面外法线的方向性来进行判别，　当某一表面的外法线与观察点和表面上任意一点之间的连线的夹角在　0≤ θ≤90 ° 之间时，该表面就是可见的，否则，　当 90 ° ≤ θ≤180 ° ，为不可见，该多边形表面为后向面。

?画家算法也称列表优先算法，需要先把屏幕置成背景色，再把根据距离观察点的远近构成的优先表逐个地取出多边形在屏幕上进行投影。　先投影的线或点会被后投影的线或点所取代。直到最高优先级的多边形的图形送入了帧缓冲器以后，整幅图就画好了。需要建立深度优先表。对于复杂画面，简单的深度易出错，特别是多个面交互重叠的情形，需把有关的面分割后再进行排序。

?区域细分算法是在图像空间中实现的，基本思想：把物体投影到窗口上，然后递归地对窗口进行分割，　直到窗口内的目标简单到足够可以显示为止。　其中，窗口是指整个屏幕； 同时，利用递归过程，　每一次把矩形的窗口等分成的四个相等的小矩形也称为窗口。

D 是窗口，多边形是　A、B、C 和 E。关系：内含、包围、相交和分离。



?光照：反射光，透射光，热能转化；只有透射光和反射光能够进入人眼而产生视觉效果，　并且反射光和透射光决定了物体所呈现出来的颜色。　具体说来，　反射光和透射光的强弱决定了物体表面的明暗程度，　同时，这些光所含的不同波长的光的比例则决定了物体表面的颜色。

反射光：环境光，镜面反射，漫反射。

?Phone模型：

$$E = E_e + E_d + E_s = I_a K_a + \sum_{i=1}^{n} I_{(d,i)} K_d \cos\theta_i + \sum_{i=1}^{n} I_{(s,i)} K_s \cos^n \phi_i$$

环境光：　$E_e = I_a K_a$　Ee 是物体表面上的一点由于受到环境光照明而反射出来的光能，称为环境光反射强度。　Ia 为物体在漫射照明时所受到的光能强度，称为环境光的强度。　Ka 为环境光的漫反射系数，也就是物体表面对环境光的反射系数 ,0≤ Ka≤1

镜面反射：　$E_s = I_s K_s \cos^n \phi$ 其中　Es 为镜面反射光在观察方向上的光强；　Is 为点光源的强度；　Φ 为视点方向与镜面反射方向的夹角；　Ks 为物体表面的反射率，也称为镜面反射系数；　n 是一个与物体表面光滑度有关的常数，用以模拟各