国际会议

- 1967 年成立计算机图形图象专业委员会，简称 SIGGRAPH（the Special Interest Group on Computer Graphics ）
- 1974 年开始每年在美国召开（2011，2014 年在温哥华举行）
- 会议录用论文代表图形学研究的主流方向
- 2008 年 SIGGRAPH Asia

1.1 发展历史

SAGE（Semi Automatic ground Environment System ）—麻省理工学院负责

1950 年　图形显示器
1952 年　数控铣床
1958 年　GerBer 平台式绘图机、Calcomp 滚筒式绘图机

图形学创始人 Ivan E.Sutherland 1963 年　MIT 的博士学位
博士论文 《Sketchpad: A Man-machine Graphical Communications System 》
3-D 计算机建模，可视化模拟， CAD，虚拟现实等概念
1988 年,获得图灵奖（以英国科学家阿兰.图灵的名字命名）

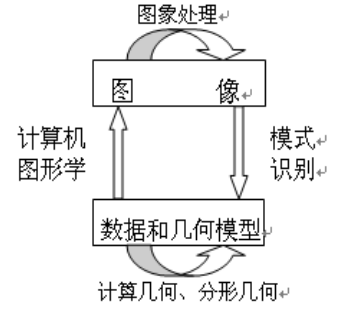
William Fetter ： 1960 年，Boeing 工作时提出

1964 年　Steve Coons　小块曲面表示自由型曲面，使曲面片边界达到任意阶连续
70 年代　Integrated Circuit　RAM(Random Access Memory)
1971 年 Pierre Bezier　UNISERFY

五、六十年代　准备和酝酿时期
六十年代　学科建立和进入应用时期
七十年代　蓬勃发展和广泛应用时期
八十年代以后　突飞猛进和成熟化、标准化时期
九十年代以后 集成化、智能化

20 世纪 80 年代末　萌芽阶段　科研院所开始对计算机图形学技术的研究
20 世纪 90 年代　发展阶段 曲线曲面造型，真实感图形显示方面取得一定的成果
2000 年 至今　有文章入选 SIGGRAPH，作者单位：微软亚洲研究院，中科院深圳先进技术研究院，清华大学，浙江大学

计算机图形学是研究怎样用计算机***生成***、***处理*** 和***显示*** 图形的一门新兴学科。
计算机图形学是由很多交叉学科形成的。



图像处理（Image Processing）

- 图像的数字化
- 图像的增强
- 图像恢复
- 图像编码
- 图像重建
- 图像分析

图形与图像

- 两个概念间的区别越来越模糊
- 区别：
  - 图象纯指计算机内以位图(Bitmap)形式存在的灰度信息
  - 图形含有几何属性，或者说更强调场景的几何表示，是由场景的几何模型和景物的物理属性共同组成的
  - 更详细的介绍见数字图像部分。。。

模式识别（Pattern Recognition）

研究怎样分析和识别输入的图像，找出其中蕴含的内在联系或抽象模型

计算几何（Computational Geometry）

计算几何是一门研究几何模型和数据处理的学科。着重讨论形体的计算机表示、几何模型的建立、模型数据的存储和管理

计算机图形学的研究内容：

- 图形的输入
- 图形的生成和输出
- 图形结果的处理

计算机图形学的应用

- 计算机辅助设计与制造（CAD/CAM）
- 作战指挥和军事训练
- 计算机动画和艺术
- 地理信息系统（GIS）

CAD/CAM

- CAD/CAM 是计算机图形学应用的一个最广泛、最活跃的领域。
- 美国波音公司 1990 年 10 月开始波音 777 的全新设计，全部设计工作在计算机终端和图形工作站上进行，在全世界范围内首次实现了“无纸设计”。整个设计、生产、试飞周期 4 年半。
- 中国大飞机计划：
  C919　　08 年开始，2015 年 11 月首架下线
- 驾驶模拟训练
  驾驶训练模拟器，也叫“汽车模拟仿真器”。该系统集计算机技术（高端 PC 工作站）、虚拟现实技术（VR）、自动化技术、多媒体技术为一体，使学员从视觉、听觉和操作感觉上都能体会出与操纵实车一样的感觉

计算机动画和艺术

- 侏罗纪公园　玩具总动员　泰坦尼克号……
- 西游记之大圣归来
- 美国 Autodesk 3D Max

加拿大　Alias　（被 SGI 公司收购）

美国 SGI　Maya……

GIS

- 1962 年 Roger Tomlinson
- 1998 年 戈尔 Open GIS Consortium

数字地球是指以地球坐标为依据的，具有多分辨率的，由海量数据组成的，能立体

表达的虚拟地球。

- 2010 年 IBM 智慧城市
- 实时化数据
- 地理数据：土地利用，路网，POI
- 交通数据：地感线圈、监控摄像，浮动车
- 移动电话：通话数据，短信数据
- 通勤数据：羊城通，咪表
- 环境监测数据：CO2，PM2.5， 噪声
- 社会网络数据: Twitter, Weibo
- ……

可视化

- 将科学计算过程中及计算结果的数据转换成图形或图像在屏幕上显示出来，并进行交互式处理的理论、方法和技术。
- 原始数据和计算结果的可视化
- 地图数据的可视化趋势：2d-> 3d, pc ->Web

Visualising Spatial and Social Media , CASA Working paper

逆向工程

- 是相对于“设计思路→产品”的一般产品开发而言。
- 分为三个层次： 原型复制（copy）→改进（redesign）→仿真（simulation）
- 关键技术：数据获取、产品建模
- 原型表面数字化→数据点网格化→几何特征提取→重建 CAD 模型

虚拟人

- 美国 1989 年 “可视人计划”
  1991 年 人体切片数据库建立
  1994 年, 美国科学家将一具男尸切成 1878 多片, 每片厚度 1 毫米, 数据量 15G；
  1998 年，又将一具女尸切成 5190 多片，每片厚度 0.33 毫米，数据量 30G
  2000 年 8 月提供人体数据集
- 韩国 2000 年 开始可视人 5 年计划现已经提前完成
- 中国 2001 年 11 月 “863”项目 中国科学院、首都医科大学、华中科技大学、第一军医大学
  - 虚拟人女 I 号 切片厚度 0.2 毫米
  - 虚拟人男 I 号 切片厚度 0.1 毫米
  - 钟世镇 广东省科学技术突出贡献奖 （2008）
- 数字可视人→数字物理人→数字生理人→数字智能人

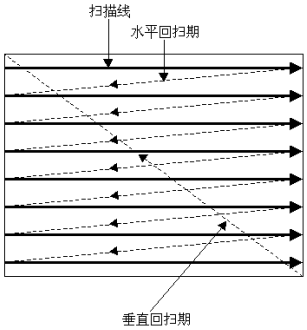
输入设备

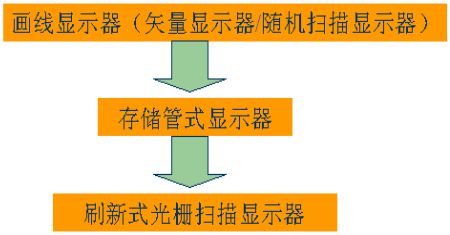
- 第一阶段：控制开关、穿孔纸等等
- 第二阶段：键盘
- 第三阶段：二维定位设备，如鼠标、光笔、图形输入板、触摸屏等等，语音
- 第四阶段：三维输入设备（如空间球、数据手套、数据衣），用户的手势、表情等等
- 第五阶段：用户的思维

显示设备的发展

随机扫描显示器：根据需要，电子束在屏幕上按任意位置扫描，扫描方向和顺序不受限制，故称随机扫描。随机扫描得到的图形只能是线条图，所以又称为矢量扫描。

光栅扫描显





硬拷贝设备：打印机

- 绘图仪

  平板式　　速度慢，精度高

  滚筒式　　速度快，精度低

- 三种类型的图形软件系统
- 用某种语言写成的子程序包

如:　GKS (Graphics Kernel System)

PHIGS(Programmer's Hierarchical Interactive Graphics system)

GL (Graphic Library)

便于移植和推广、但执行速度相对较慢，效率低

- 扩充计算机语言，使其具有图形生成和处理的功能

如：Turbo Pascal、Turbo C，AutoLisp 等。

简练、紧凑、执行速度快，但不可移植

- 专用图形系统：效率高，但系统开发量大，可移植性差。

软件标准

通用的、与设备无关的图形包，图形标准

- GKS (Graphics Kernel System) (第一个官方标准)
- PHIGS(Programmer's Hierarchical Interactive Graphics system)
- IGES (Initial Graphics Exchange Specification)
- STEP (Standard for the Exchange of Product Model Data)

一些非官方图形软件，广泛应用于工业界，成为事实上的标准

- DirectX (MS)
- Xlib(X-Window 系统)
- OpenGL(SGI)
- Adobe 公司 Postscript

OpenGL(Open Graphic Library)

- 1992 年 SGI 公司推出 是一套独立于操作系统和硬件环境的三维图形库。
- Microsoft、IBM、DEC、SUN 等公司将其作为图形标准。
- Visual C++2.0 以上版本内置的 OpenGL

光栅图形学

- 显示器是由离散像素组成的矩阵，在绘制具有连续性质的直线、曲线或区域等基本图形时，需要确定最佳逼近它们的像素，这个过程称为光栅化。
- 当光栅化按照扫描线的顺序进行时，它被称为扫描转换。

生成直线的算法

1.一般直线 $y = kx + b$，从起点到终点，x 每次增加(或减少)1，用直线方程计算对应的 y 值，再用 SetPixel(x, int(y+0.5),color)输出该像素。

复杂度：乘法+加法+取整

2.DDA 算法(Digital Differential Analyzer)

假设直线的端点为：(x_1,y_1) 、(x_2,y_2)，则其微分方程为：$\frac{\Delta y}{\Delta x}=\frac{y_2-y_1}{x_2-x_1}=m$

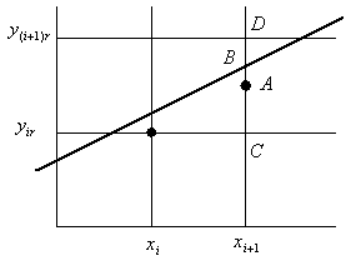
解为：$y_{i+1}=y_i+\Delta y \quad x_{i+1}=x_i+\Delta x$

当$|m|\le 1$时 $y_{i+1}=y_i+m\cdot\Delta x$　　当$|m|>1$时 $x_{i+1}=x_i+\frac{\Delta y}{m}$

减少了浮点乘法，提高了效率。但是 x 与 dx、y 与 dy 用浮点数表示，每一步要进行四舍五入后取整 round().

3.Bresenham 算法

设 xi 列上已经用（xi,yir）表示直线上的点，又设 B 点是直线上的点，其坐标为（xi+1,yi+1),显然下一个表示直线的点只能在 C 点或 D 点中选。设 A 点为 CD 边的中点，则 B 在 A 的上面则应取 D 点，否则取 C 点



$$\varepsilon(x_{i+1})=y_{i+1}-y_{ir}-0.5$$

$$y_{i+1,r}=y_{ir}+1 \qquad \varepsilon(x_{i+1})\ge 0$$

$$y_{i+1,r}=y_{ir} \qquad \varepsilon(x_{i+1})<0$$

$$\varepsilon(x_{i+2})=\begin{cases} y_{i+1}-y_{ir}-0.5+m-1, \text{当}\ \varepsilon(x_{i+1})\ge 0 \\ y_{i+1}-y_{ir}-0.5+m \text{.....当}\ \varepsilon(x_{i+1})<0 \end{cases}$$

$$=\begin{cases} \varepsilon(x_{i+1})+m-1, \text{当}\ \varepsilon(x_{i+1})\ge 0 \\ \varepsilon(x_{i+1})+m \text{.....当}\ \varepsilon(x_{i+1})<0 \end{cases}$$

$$\varepsilon(x_{i+2})=y_{i+2}-y_{(i+1)\cdot r}-0.5$$

$$=y_{i+1}+m-y_{(i+1)\cdot r}-0.5$$

$$=\begin{cases} y_{i+1}+m-(y_{i\cdot r}+1)-0.5 & \varepsilon(x_{i+1})\ge 0 \\ y_{i+1}+m-(y_{i\cdot r})-0.5 & \varepsilon(x_{i+1})<0 \end{cases}$$

$$=\begin{cases} \varepsilon(x_{i+1})+m-1 & \varepsilon(x_{i+1})\ge 0 \\ \varepsilon(x_{i+1})+m & \varepsilon(x_{i+1})<0 \end{cases}$$

初始 e=m-0.5

整数 Bresenham 算法

根据误差项 d 的值来决定是否增 1 的过程如下：

$$d_1=y-y_i=(k(x_i+1)+b)-y_i$$

$$d_2=(y_i+1)-y=y_i+1-(k(x_i+1)+b)$$

$$d_1-d_2=2k(x_i+1)-2y_i+2b-1$$

设$\Delta y=y_1-y_0$，$\Delta x=x_1-x_0$，则 $k=\Delta y/\Delta x$，代入上式，得；$\Delta x(d_1-d_2)=2\cdot\Delta y\cdot x_i-2\cdot\Delta x\cdot y_i+c$

$c=2\Delta y+\Delta x(2b-1)$ 是常量，与像素位置无关

令 $d_i=\Delta x(d_1-d_2)$

di 的符号与(d1-d2)的符号相同。

- 当 di<0 时，直线上理想位置与像素(xi + 1，yi)更接近，应取右方像素；
- 当 di>0 时，像素(xi + 1，yi + 1)与直线上理想位置更接近；
- 当 di=0 时，两个像素与直线上理想位置一样接近，可约定取(xi + 1，yi + 1)。

$$d_i = \Delta x(d_1 - d_2) = 2\cdot\Delta y\cdot x_i - 2\cdot\Delta x\cdot y_i + c$$

对于k+1步，误差参数为：

$$d_{i+1} = 2\cdot\Delta y\cdot x_{i+1} - 2\cdot\Delta x\cdot y_{i+1} + c$$

$$d_{i+1} - d_i = 2\cdot\Delta y\cdot(x_{i+1} - x_i) - 2\cdot\Delta x\cdot(y_{i+1} - y_i)$$

此时参数C已经消去，且$x_{i+1}=x_i+1$,得：

$$d_{i+1} = d_i + 2\cdot\Delta y - 2\cdot\Delta x\cdot(y_{i+1} - y_i)$$

初始 d=2Δy-Δx，若 $y_{i+1}=y_i+1$，则 $d_{i+1}=d_i+2\Delta y-2\Delta x$；若 $y_{i+1}=y_i$，则 $d_{i+1}=d_i+2\Delta y$

圆弧 $x^2 + y^2 = R^2$

1.根据圆的基本方程，可以沿 x 轴，x 从 0 到$\frac{\sqrt{2}}{2}R$，以单位步长计算对应的 y 值来得到圆周上每点的位置：$y = \sqrt{R^2 - x^2}$　　（只要实现 1/8 圆）

像素间间隔不一，随着 x 的增加，间隔越来越大

2.使用极坐标来计算沿圆周的点，此时，圆使用参数方程来表示：$\begin{cases} x=r\cos\theta \\ y=r\sin\theta \end{cases} \theta \in [0, \Pi/4]$

像素间间隔不一，随着 x 的增加，间隔越来越大

3.中点法

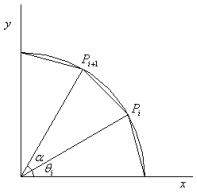
以从(0，R)到($\frac{\sqrt{2}}{2}R, \frac{\sqrt{2}}{2}R$)的 1/8 圆为例；

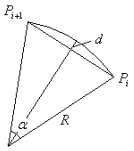
假定当前已确定了圆弧上的一个像素点为 P(xp,yp)，那么下一个像素只能是其右方、右下方的点 P1、P2，记 M 为 P1、P2 中点

$F(x,y)=x^2+y^2+-R^2$ ，while(x<=y)

- 若 F(M)<0，M 在圆内，P1 点离圆弧更近，取 P1 为下一个像素；
- 若 F(M)>0，M 在圆外，P2 点离圆弧更近，取 P2 为下一个像素；
- 若 F(M)=0，M 在圆上，P1、P2 可任取，这里约定取 P2 ；

4.多边形逼近





$$\begin{cases} x_i = R\cos\theta_i \\ y_i = R\sin\theta_i \end{cases} \quad \begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} R\cos(\theta_i+\alpha) \\ R\sin(\theta_i+\alpha) \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

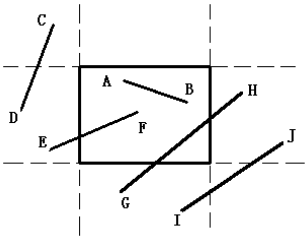
误差 $d = R - R\cos\frac{\alpha}{2}$，限差 Delta，$\cos\frac{\alpha}{2} \geq \frac{R-Delta}{R}$

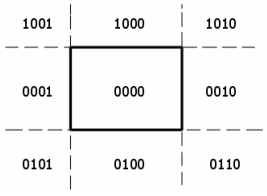
$\alpha \leq 2\arccos\frac{R-Delta}{R}$，$n = 360^0 / \alpha$

裁剪：按预先设置的窗口参数，沿窗口边框线对图形进行裁剪，保留窗口内部图形，裁剪掉窗口外图形的方法

空间任意一点(x,y)是否在窗口内的充分必要条件：$\begin{cases} XL \leq x \leq XR \\ YB \leq y \leq YT \end{cases}$

直线	起点	终点	逻辑与	性质
AB	0000	0000	0000	完全可见
CD	1001	0001	0001	完全不可见
EF	0001	0000	0000	部分可见
GH	0100	0010	0000	部分可见
IJ	0100	0010	0000	完全不可见





1.Cohen-Sutherland 算法

关键技术：在于总是要让直线段的一个顶点处于窗口之外，例如 P_0 点。这样 P_0 点到交点 P 的直线段必然不可见，故可以直接抛弃

- 输入直线段的两端点 P1、P2 坐标，以及窗口的参数。
- 对 P1、P2 进行编码；
- Code1、Code2 都为 0000，则此直线为可见；
- Code1、Code2 经位与运算不等于零，则此线段位于窗口的同一侧，为不可见；
- 算出直线与窗口的交点，将直线分割，舍去交点之外的一段，再对另一段重复上述处理。

裁剪直线段时，一般按固定顺序，如左（$x=w_{xl}$），右（$x=w_{xr}$）、下（$y=w_{yb}$）、上（$y=w_{yt}$）求解窗口边界与直线段的交点。

求交方法为直接代入边界值，求横纵坐标

特点：该算法用编码的方法实现了对完全可见和不可见直线段的快速接受和拒绝，这使得它在两类裁剪场合中非常高效：

（1）大窗口的场合，其中大部分线段为完全可见；

（2）小窗口的场合，其中大部分线段为完全不可见

- 但是求交计算较复杂

2.中点分割法

- 核心思想是通过二分逼近来确定直线段与窗口的交点。
- 是 Sproul 和 Sutherland 为了便于硬件实现而提出的。

线段既非完全可见，也非完全不可见。求得中点 Pm1。两半线段均不能简单接受或拒绝。不断求中点去掉确定的线段部分，最后得到交点

- 对分辩率为 2^N*2^N 的显示器，上述二分过程至多进行 N 次。
- 主要过程只用到加法和除法运算，适合硬件实现。适合并行计算。

3.Liang-Barsky 算法

- 标准矩形窗口的直线裁剪算法 $x = x_1 + u\cdot(x_2 - x_1)$
- 算法的基本出发点：直线的参数方程 $y = y_1 + u\cdot(y_2 - y_1)$ $\quad 0 \le u \le 1$

如果直线上的点（x,y）位于窗口内，则满足一下条件：$wxl \le x_1 + u\cdot(x_2 - x_1) \le wxr$，$wyb \le y_1 + u\cdot(y_2 - y_1) \le wyt$

$$u\cdot(-\Delta x) \le x_1 - wxl \qquad p_1 = -\Delta x \quad q_1 = x_1 - wxl$$
$$u\cdot\Delta x \le wxr - x_1 \qquad p_2 = \Delta x \quad q_2 = wxr - x_1$$
$$u\cdot(-\Delta y) \le y_1 - wyb \qquad p_3 = -\Delta y \quad q_3 = y_1 - wyb$$
$$u\cdot\Delta y \le wyt - y_1 \qquad p_4 = \Delta y \quad q_4 = wbt - y_1$$

- 一般位置线 $u_{\max} = \max(0, u_{k|p_k<0}, u_{k|p_k<0})$，$u_{\min} = \min(u_{k|p_k>0}, u_{k|p_k>0}, 1)$
  - 如果max>min，则直线段在窗口外
  - 否则把参数带入方程计算交点

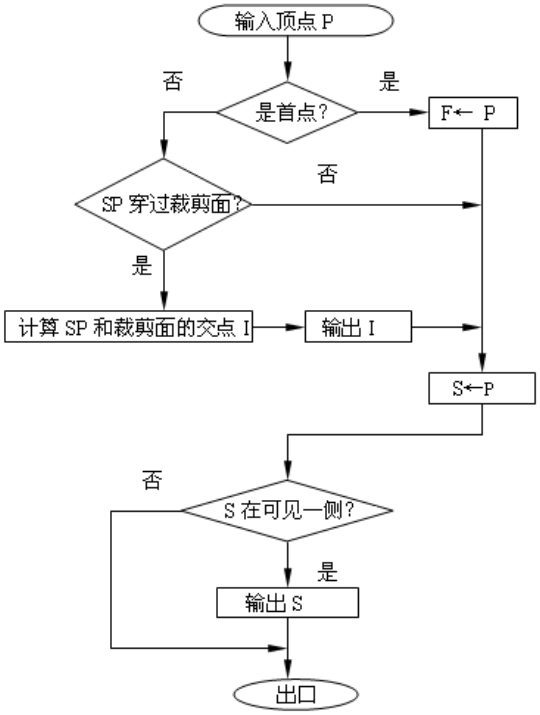
裁剪窗口的每条边界线将平面分为两个区域。定义裁剪窗口所在侧为可见侧（visible side），另一侧为不可见侧（invisible side），图中阴影线所示为边界线的可见侧。这样，裁剪窗口也可以定义为所有边界线的可见侧。

多边形裁剪

- 一个完整封闭的多边形经窗口边框裁剪后，一般不再封闭。
- 一个完整封闭的非凸多边形，常被窗口边框裁剪成几个独立的小多边形。

Sutherland_Hodgman 算法

- 该算法的基本思想是：通过对单一边的裁剪实现对多边形的裁剪。
- 用窗口的一条边框线对多边形进行裁剪，得到一个或几个新的多边形，再用第二条边框线对这些新的多边形进行裁剪。如此进行四次，从而把多边形相对于窗口的全部四条边框线进行了裁剪。



- 凸多边形用此算法可以获得正确的裁剪结果，而且可以推广到任意凸多边形窗口的情况。
- 在处理凹多边形时，

* *只能处理裁剪后仍为一个连通图的凹多边形*
* 裁剪之后产生多个分离部分的凹多边形，该算法会产生一些多余的边

二维填充

多边形的表示

- 顶点表示

是用多边形的顶点序列来刻画多边形。这种表示方法直观、几何意义强、占空间少，易于进行几何变换，被广泛地应用于各种几何造型系统中。

- 点阵表示

是用位于多边形内部的像素的集合来刻画多边形。这种表示方法虽然失去了许多重要的几何信息，但它是光栅显示系统显示时所必须的表示形式。

扫描转换多边形

- 多边形从顶点表示到点阵表示的转转，称为扫描转换。
- 常用算法有两大类
  - 扫描线算法
  - 种子填充

逐点判断算法：逐个判断绘图窗口内的像素，确定它们是否在多边形区域内部，从而求出位于多边形区域内的像素集合

包围盒：包围多边形的最小矩形。只有在包围盒内的点需要检查。

- 对于凸多边形，可以极大地减少所需检查的象素个数，但是对凹多边形，则减少的检查工作量则要少得多。

点与多边形的关系

射线法—从点发出射线与多边形的边相交，若交点的个数为奇数，则点位于多边形内；若为

偶数，则点位于多边形外。通常为了简单起见，射线为水平线或者垂直线。
注意射线通过多边形顶点的情况。
扫描线算法：扫描线上相邻象素几乎具有相同的特性，这就是所谓扫描线的连贯性。
在给定的扫描线上，象素的这种特性只有在多边形的边和该扫描线交点处才会发生变化。

- 扫描线与多边形求交
- 交点按照 x 升排序
- 交点两两配对
- 每对交点所确定的区段取多边形的填充色，交点对之间的区间则取背景色。

顶点为局部极值点，则计***两次***，否则计***一次***

种子填充：

- 假设在多边形或区域内至少有一个象素是已知的，然后设法找到区域内所有其它象素，并对它们进行填充。(4 连接、8 连接)

1.种子象素入栈，当栈非空时，重复执行：

- 栈顶象素出栈；
- 将出栈象素置成多边形色；

2.四向检查象素，是否为边界象素或是否已经设置成多边形颜色。若是则忽略不计，否则把该象素压入堆栈
缺点：

- 有些象素会入栈多次，降低算法效率；栈结构占空间。
- 递归执行，算法简单，但效率不高，区域内每一象素都引起一次递归，进/出栈，费时费内存。

扫描线种子填充：
Step1：栈顶象素出栈，
Step2：沿扫描线对出栈象素的左右象素进行填充，直到遇到边界象素为止，即填充区间。
Step3：检查上下扫描线，选相应区间的最右象素作种子象素入栈。

二维图形变换
几何变换：几何变换提供了构造或修改图形的一种方法，图形在方向、尺寸和形状方面的改变都可以通过几何变换来实现。
一般来说图形的几何变换是指对图形的几何信息经过平移、比例、旋转等变换后产生新的图形。
把变换矩阵作为一个算子，作用到图形一系列顶点的位置矢量，得到新的顶点序列。
几何变换是在同一坐标系下进行的，因此，变换中坐标系是静止的，而图形是变动的。
基本几何变换都是相对于原点和坐标轴进行的几何变换，有平移、缩放和旋转等。
平移是指将 p 点沿直线路径从一个坐标位置移到另一个坐标位置的重定位过程。
齐次坐标:　用 n+1 维向量表示 n 维向量
(x,y)点对应的齐次坐标为 $x_h = hx, y_h = hy, h \neq 0$ ，标准齐次坐标为：(x,y,1)
变换具有统一表示形式的优点
用规范化齐次坐标表示的二维基本几何变换矩阵是一个 3×3 的方阵，简称为二维变换矩阵

$$T = \left[\begin{array}{cc|c} a & b & p \\ c & d & q \\ \hline l & m & s \end{array}\right]$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \xrightarrow{\text{记为}} = T(t_x, t_y) \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \xrightarrow{\text{记为}} = R(\theta) \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \xrightarrow{\text{记为}} = S(s_x, s_y) \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad SY_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad SY_y = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

复合变换

由于引入了齐次坐标，基本几何变换均可以表示成$P'=T\cdot P$的形式

复合变换时指图形作一次以上的几何变换。 $T=T_n.....T_3\cdot T_2\cdot T_1 \quad n>1$

复合变换具有同样的形式。所不同的是，此时有：$P'=T\cdot P=T_n.....T_3\cdot T_2\cdot T_1\cdot P \quad n>1$

由于矩阵的乘法满足结合律，因此，通常在计算时先求出 T，再与 P 相乘。即

$$P'=T\cdot P=(T_n.....T_3\cdot T_2\cdot T_1)\cdot P \quad n>1$$

任意参照点的放缩变换：

（1）把固定点移到坐标系的原点；

（2）把物体相对于坐标原点进行比例变换；

（3）再把固定点移回到原来的位置

窗口到视口的变换

要将窗口内的图形在视区中显示出来，必须经过将窗口到视区的变换（Window-Viewport Transformation）处理。

要将窗口内的点（xw,yw）映射到相对应的视区内的点（xv,yv）需进行以下步骤：

(1) 将窗口左下角点移至用户坐标系的坐标原点

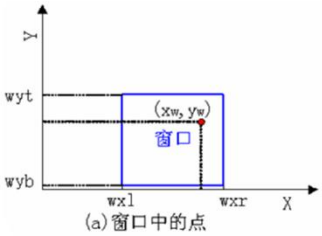
(2) 针对原点进行比例变换

(3) 进行反平移

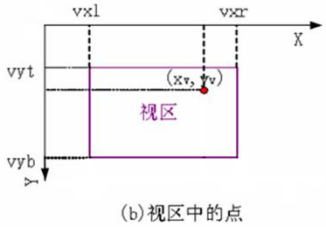
$$x_v=(x_w-wxl)*S_x+vxl$$

$$y_v=(wyt-y_w)*S_y+vyt$$

$$S_x=\frac{vxr-vxl}{wxr-wxl},\quad S_y=\frac{vyb-vyt}{wyt-wyb}$$



(a)窗口中的点



(b)视区中的点

当 Sx=Sy 时，X 和 Y 方向的变形相同，变换后图形形状保持不变。

三维图形的基本几何变换

基本几何变换都是相对于原点和坐标轴进行的几何变换，有***平移***、***缩放***和***旋转***等。在以下的讲述中，均假设用 P(x,y,z)表示三维空间上一个未被变换的点，而该点经过某种变换后得到的新点用 P'(x',y',z')表示。

$$\begin{bmatrix}x'\\y'\\z'\\1\end{bmatrix}=\begin{bmatrix}1&0&0&\Delta x\\0&1&0&\Delta y\\0&0&1&\Delta z\\0&0&0&1\end{bmatrix}\cdot\begin{bmatrix}x\\y\\z\\1\end{bmatrix}\quad\begin{bmatrix}x'\\y'\\z'\\1\end{bmatrix}=\begin{bmatrix}s_x&0&0&0\\0&s_y&0&0\\0&0&s_z&0\\0&0&0&1\end{bmatrix}\cdot\begin{bmatrix}x\\y\\z\\1\end{bmatrix}\quad\begin{bmatrix}x'\\y'\\z'\\1\end{bmatrix}=\begin{bmatrix}\cos\alpha&-\sin\alpha&0&0\\\sin\alpha&\cos\alpha&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\cdot\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

$$\begin{bmatrix}x'\\y'\\z'\\1\end{bmatrix}=\begin{bmatrix}\cos\alpha&0&\sin\alpha&0\\0&1&0&0\\-\sin\alpha&0&\cos\alpha&0\\0&0&0&1\end{bmatrix}\cdot\begin{bmatrix}x\\y\\z\\1\end{bmatrix}\quad\begin{bmatrix}x'\\y'\\z'\\1\end{bmatrix}=\begin{bmatrix}1&0&0&0\\0&\cos\alpha&-\sin\alpha&0\\0&\sin\alpha&\cos\alpha&0\\0&0&0&1\end{bmatrix}\cdot\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

旋转角度正负的确定：当沿坐标轴的正向往坐标原点看过去时，**逆时针**方向旋转的角度为正向旋转角

绕任意轴的旋转

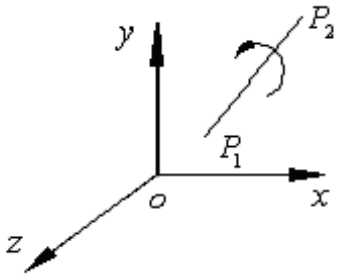
1）使 P_1 点与原点重合（图(b)）；

2）使得轴落入坐标平面内（图(c)）

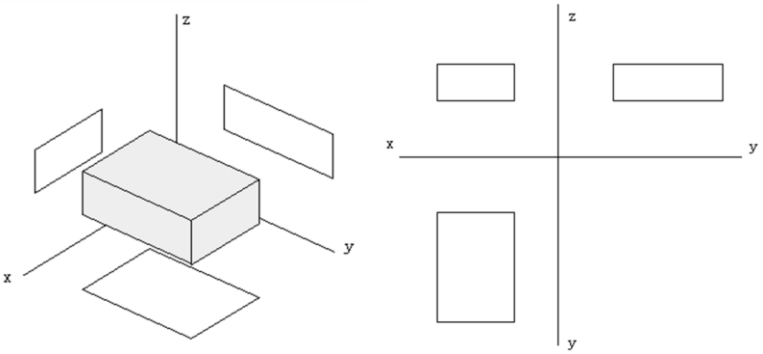
3）使与 z 轴重合（图(d)）；

4）执行绕轴的角度的旋转(图(e))；

5）作变换 3-2-1 的逆变换；



投影：投影将三维物体变换为二维图形表示的过程。
投影变换的主要元素：光源、投影方向、投影平面
平行投影可根据投影方向与投影面的夹角分为两类：正投影和斜投影。当投影方向与投影面垂直时，为正投影，否则为斜投影。



$$T_v=\begin{bmatrix}-1&0&0&0\\0&0&0&0\\0&1&0&0\\0&0&0&1\end{bmatrix}\quad T_H=\begin{bmatrix}-1&0&0&0\\0&-1&0&0\\0&0&0&0\\0&0&0&1\end{bmatrix}\quad T_W=\begin{bmatrix}0&0&0&0\\1&0&0&0\\0&1&0&0\\0&0&0&1\end{bmatrix}.$$

主视图 俯视图 左视图

曲线的生成
初等解析曲线、自由曲线
直线段的数学表示
曲线的表示方法
非参数法：显式函数形式，如 y=f(x)　隐式函数形式，如 f(x,y)=0
参数法：
$$x(t)=a_1t^3+b_1t^2+c_1t+d_1$$
$$y(t)=a_2t^3+b_2t^2+c_2t+d_2$$
$$z(t)=a_3t^3+b_3t^2+c_3t+d_3$$
插值：当用一组数据点来指定曲线的形状时，曲线精确地通过给定的数据点且形成光滑的曲线，称为曲线的插值。
逼近：当用一组控制点来指定曲线的形状时，曲线被每个控制点所吸引，但实际上并不必须经过这些控制点，称为曲线的逼近
连续性条件：通常单一的曲线段或曲面片难以表达复杂的形状，必须将一些曲线段连接成组合曲线，或将一些曲面片连接成组合曲面，才能描述复杂的形状。为了保证在连接点处光滑过渡，需要满足连续性条件
参数连续性
C^0连续，是指曲线的几何位置连续
C^1连续，是指两个相邻曲线段的方程在交点处有相同的一阶导数
C^2连续，是指两个相邻曲线段的方程在交点处有相同的一阶、二阶导数
几何连续性
G^0连续，是指曲线的几何位置连续
G^1连续，是指曲线在相邻的交点处方向相同，大小不一定相同
G^2连续，是指相邻曲线段在交点处一阶、二阶导数均成比例
样条描述:描述 n 次参数多项式曲线的方程：
$$x(t)=a_nt^n+\cdots+a_2t^2+a_1t+a_0$$
$$y(t)=b_nt^n+\cdots+b_2t^2+b_1t+b_0\quad t\in[0,1]$$
$$z(t)=c_nt^n+\cdots+c_2t^2+c_1t+c_0$$
写成矩阵形式：$p(t)=[x(t)\quad y(t)\quad z(t)]=[t^n\quad \dots\quad t\quad 1]\cdot C=T\cdot C$

矩阵 C：三次样条曲线由两个端点和端点的切矢来定义。

$$\begin{bmatrix} p(0) \\ p(1) \\ p'(0) \\ p'(1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \cdot C$$

$$p(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot C \quad p'(t) = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} \cdot C$$

$$C = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \cdot \begin{bmatrix} p(0) \\ p(1) \\ p'(0) \\ p'(1) \end{bmatrix} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} p(0) \\ p(1) \\ p'(0) \\ p'(1) \end{bmatrix}$$

Bezier 曲线：$p(t) = \sum_{i=0}^{n} P_i B_{i,n}(t)$

$$p(t) = \sum_{i=0}^{3} P_i B_{i,3}(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3$$

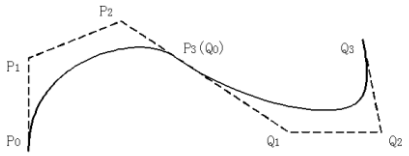
$$p(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

de Casteljau 算法：给定空间 n+1 个控制点 P_i（i=0，1，2…n）及参数 t，de Casteljau 递推算法表述为 $P_i^r(t) = (1-t) \cdot P_i^{r-1}(t) + t \cdot P_{i+1}^{r-1}(t) \quad r = 1,2,\cdots n; \ i = 0,1,\cdots n-r; \ t \in [0,1]$

Bezier 曲线的拼接：保证连接处具有 G1、G2 连续性

$$Q_1 - Q_0 = \alpha \cdot (P_3 - P_2)$$

$$(Q_0 - 2Q_1 + Q_2) = \beta \cdot (P_1 - 2P_2 + P_3)$$

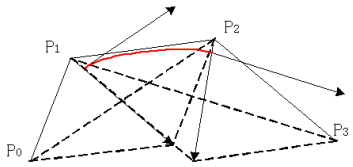


Bezier 曲线的优缺点

- 控制多边形确定曲线的形状,在设计过程中具有很强的可操作性
- 不具有局部修改性,修改一点会影响到整体，拼接复杂

三次 B 样条（G2 连续）

$$p(t) = \frac{1}{6}\begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$



消隐：消隐要解决的问题就是在给定视点和视线方向后，决定场景中哪些物体的表面是可见的，哪些是被遮挡不可见的。

消隐的目的：消除二义性，增加图形的真实感

消隐算法的分类：

1 对象空间：在对象被定义时所处的坐标系中实现。该算法精度高。

- for(场景中的每一个物体) {
  将其与场景中的其它物体比较，确定其表面的可见部分；
  显示该物体表面的可见部分；
  }

2 图象空间：在对象显示所在的屏幕坐标系中实现。一旦达到屏幕的分辨率，计算就终止。

- for(场景中的每一个像素) {
  确定与此像素对应的距离视点最近的物体；
  以该物体表面该处的颜色来显示像素；
  }

理论上讲，物空间算法的计算量少于像空间算法的计算量，但实际上物体到视点距离的排序与遮挡判别比较复杂，算法效率很大程度上取决于排序的效率。

以扫描线的方式实现像空间算法时容易利用连贯性质从而使得像空间算法更具效率。

对于凸多面体的任一个面，根据其外法矢量和视矢量的夹角θ来进行可见性检测。如果两个矢量的夹角 0°≤θ≤90°时，表示该表面可见；如果 90° < θ≤180°时，表示该表面不可见。

确定面的朝向

假设多边形面的方程为：ax+by+cz+d=0，即[x y z 1]·[a b c d]^T=0　　$[V]=\begin{bmatrix} a_1 & a_2 & \dots & a_n \\ b_1 & b_2 & \dots & b_n \\ c_1 & c_2 & \dots & c_n \\ d_1 & d_2 & \dots & d_n \end{bmatrix}$

- 凸体可由平面方程系数组成的体矩阵表示
- 有一点 S，如果在平面上，则[S]·[P]=0，如果在平面的一侧，则>0 或者<0.

对体矩阵，应该使得平面方程的系数（a,b,c）分别是指向物体内部的内法矢。

- 内法矢与视线夹角决定了多边形的朝向

Z-Buffer 算法：

1. 初始化：把 Z-Buffer 中各（x,y）单元置为 z 的最小值，而帧缓存中各(x,y)单元置为背静色。
2. 在把物体表面相应的多边形扫描转换成帧缓存中的信息时，对于多边形内的每一个采样点(x,y)进行以下几步处理：
   (a). 计算采样点(x,y)的深度 z(x,y);
   (b). 如 z(x,y)>zbuffer(x,y),则把 z(x,y)存入 z 缓存，再把多边形在 z(x,y)处的颜色存入帧缓存的(x,y)地址中。

如果平面方程为： Ax+By+Cz+D=0，若 C≠0 ， z(x,y)=(Ax+By+D)/(-C)

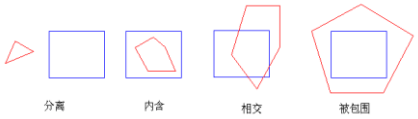
利用连贯性加速 z 值的计算：z(x+1,y)=(A(x+1)+By+D)/(-C)=z(x,y)-A/C

```
For pixcel in windows{
    将帧缓冲器的第(u,v)单元置为背景色;
    将 Z 缓冲器的第(u,v)单元置为最小深度值;
}
for（每个多边形）
    for （多边形在投影平面上的投影区域内的每个像素（u,v））{
        计算多边形在当前像素（u,v）处的深度值 d；
        if （d>Z 缓冲器的第（u,v）单元的值）{
            置帧缓冲器的第（u,v）单元值为当前背景色；
            置 Z 缓冲器的第（u,v）单元值为 d;
        }
    }
```

Warnock 算法：先考察在图象空间中的一个窗口，判定该窗口是否为空，或者该窗口所包含的画面***足够简单***可以立即显示，否则分割该窗口，直到子窗口内所包含的画面相当简单可以立即显示或者其尺寸已达到给定的分辨率。

- 太简单，则分割的层次大大增加；
- 太复杂，则失去了使用四叉树结构的意义；

使用较为复杂的分割算法和分割准则，可使算法更为有效。



多边形与窗口之间的关系：

对于每一个窗口：

若画面中所有多边形均与窗口分离，则窗口为空。用背景色填充；

若窗口中仅包含一个多边形，则窗口中多边形外区域背景色填充，多边形内前景色填充；

若窗口与一个多边形相交，则窗口中多边形外背景色填充，窗口内多边形内前景色填充；

若窗口为一个多边形所包围且窗口内无其他多边形，则窗口按包围多边形颜色填充；

若窗口至少为一个多边形所包围且此多边形离视点最近，则窗口按此多边形的颜色填充

否则分割窗口

多边形是否内含于窗口：窗口与包围盒

若多边形每一棱边均与窗口无交，则多边形与窗口分离或者包围此窗口→包围盒

真实感图形绘制：对于场景中的物体、要得到它的真实感图形，就要对它进行透视投影，并消除隐藏面，然后计算可见面的光照明暗效果。给定一个**三维场景及其光照明条件**，如何确定它在屏幕上生成的真实感图象，即确定图象每一个象素的**明暗、颜色**，是真实感图形学需要解决的问题。

1 用数学方法建立所需三维场景的几何描述

2 将三维几何描述转换为二维透视图。通过对场景的透视变换来完成；

3 隐藏面消除，确定场景中的所有可见面，将视野之外或被其它物体遮挡的不可见面消去；（以上确定形状）

4 计算场景中可见面的颜色：根据基于光学物理的光照明模型计算可见面投射到观察者眼中的 光亮度大小和色彩组成，并将它转换成适合图形设备的颜色值， 从而确定投影画面上每一象素的颜色（或者与纹理相结合），最终生成图形。（以上确定色彩）

光照计算的必要性：单纯判别物体表面的可见性，远远不能反映物体表面的真实感.

光照模型：在已知物体物理形态和光源性质的条件下，能够计算出场景的光照明暗效果的数学模型称为光照模型。早期的光照明模型都是基于经验的模型，只能反映光源直接照射的情况，而一些比较精确的模型，通过模拟物体之间光的相互作用，可以产生更好的真实感效果。

简单光照模型：

当光照射到物体表面时，光可能被吸收、反射和透射，被物体吸收的部分转化为热，只有反射、透射的光能够进入人眼产生视觉效果，它们决定了物体所呈现的颜色。

如果物体是不透明的，则透射光不存在，物体的颜色仅由反射光决定。

简单光照模型只考察光源直接照射下物体表面的反射情况。

光学反射模型：环境光的反射、对特定光源的漫反射和镜面反射

环境光反射：环境光(ambient light)指光源间接对物体的影响，是在物体和环境之间多次反射，最终达到平衡时的一种光。近似地认为同一环境下的环境光，其光强分布是均匀的，它在任何一个方向上的分布都相同。 物体对环境光的反射分量表示 $I = K_a I_a \quad 0 \le K_a \le 1$

漫反射：当光源来自一个方向时，漫反射光均匀向各方向传播，与视点无关，它是由表面的粗糙不平引起的，因而漫反射光的空间分布是均匀的。$I = K_d I_l \cos(\theta) \quad 0 \le \theta \le \frac{\pi}{2} \quad 0 \le K_d \le 1$

镜面反射：理想镜面，反射光集中在一个方向，并遵守反射定律。对一般的光滑表面，反射光集中在一个范围内，且由反射定律决定的反射方向光强最大 $I = K_s I_l \cos^n(\alpha) \quad 0 \le \alpha \le \frac{\pi}{2}$

镜面反射光将会在反射方向附近形成很亮的光斑，称为高光现象。

兰伯特反射光照模型：只考虑对**环境光的反射分量和对特定光源的漫反射分量**，则物体表面的反射光亮度为：$I = K_a I_a + K_d I_l \cos(\theta) \quad 0 \le \theta \le \frac{\pi}{2} \quad 0 \le K_d \le 1$

适用于**粗糙、无光泽的物体**。对于擦亮的金属、光滑的塑料等光亮物体需要计算镜面反射。

Phong 模型：考虑环境光、漫反射和镜面反射，则物体表面的反射光亮度为 ：

$I = K_a I_a + K_d I_l \cos(\theta) + K_s I_l \cos^n(\alpha) \qquad I = K_a I_a + \frac{I_l}{d^p + K}\left[K_d \cos(\theta) + K_s \cos^n(\alpha)\right]$

光亮度衰减：实际上光的**亮度与传播距离的平方成反比**，I1 为光源处的光亮度，光线抵达物体表面以及从物体表面反射进入观察者眼睛的过程中存在衰减的问题。

漫反射分量和镜面反射分量应该乘以一个衰减因子，以使得远的物体看起来暗些的效果。

当场景的投影变换采用透视投影时，Warnock 提出线性衰减因子 1/d，而 Rommey 提出衰减因子 $1/d^p$ 可以取得比较真实的效果。d 是物体上当前考察点到视点的距离

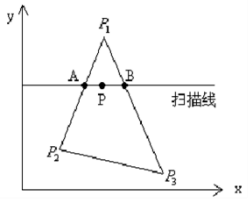
如果存在多个光源，则将效果线性相加。

明暗处理：光照计算时需要用到多边形上点的法矢量，如果多边形上点的法矢量总是取多边形的面法矢，则由于不同平面片之间法矢量不连续，最终绘制出来的图像看起来呈多面体状。

Phong 明暗处理(法矢插值)：

如图 P1、P2、P3 是多边形顶点，其法矢量视为共该点的所有多边形法矢量的平均值。

由 P1、P2 的法矢量可以线性插值计算出 A 点的法矢量，由 P1、P3 的法矢量可以线性插值计算出 B 点的法矢量，P 点的法矢量可以由 A、B 点处的法矢量线性插值计算出，计算出 P 的法矢量后应用简单光照模型可以计算出 P 点的光亮度。



对P点进行双线性插值

Gouraud 明暗处理（双线性光强插值）：先计算物体表面多边形各顶点的光强，然后用双线性插值，求出多边形内部区域中各点的光强。

Phong 模型特点：

表面漫反射和镜面反射的光亮度均被认为是**对光源入射光的直接反射**。

镜面反射光亮度由经验公式计算，高光指数来模拟景物的光滑程度

周围环境对景物表面的影响被假设为一常数

Whitted 模型：Whitted 特别考察了光在物体间往复反射、折射引起的照明效果。Whitted 认为物体表面向空间某方向 V 辐射的光亮度 I 由三部分组成： $I = I_c + K_s \cdot I_s + K_t \cdot I_t$

Ic 为简单光照模型计算结果，*Is* 为其它物体反射光，*It* 为透射光，

Ks 为物体表面的镜面反射系数，*Kt* 为物体表面的透射系数。

光线追踪：自然界光照明物理过程的的近似逆过程，光线跟踪故此得名。

光线追踪何时终止

- 如果发出的光线与所有景物无交点，则来自这条光线方向上的亮度视为 0；
- 追踪深度已经超出预先设定的深度；
- 来自发出光线方向上的亮度对显示像素亮度的贡献系数

纹理映射：采用光照模型只能生成颜色单一的光滑景物表面，计算景物表面各点处的光亮度时，仅仅考虑了表面法向的变化而假设表面反射率为一常数。实际上，景物表面有丰富的纹理细节。1974 年，Catmull 首次采用纹理技术生成景物表面。

纹理形式

- 颜色纹理：呈现在物体表面的各种花纹、图案和文字等，如大理石墙面、器皿上的图案、墙上帖的字画等；
- 几何纹理：基于物理表面微观几何形状的表面纹理，如桔子、岩石、树干表面呈现的凹凸不平的纹理细节；
- 过程纹理：表现了各种规则或不规则的动态变化的自然景象，如水波、云、火、烟雾等

在计算机中描述物体材质属性、几何形状的细节很难，如果只追求看起来象就可以了，可以通过纹理映射的方式生成物体表面的细节。

利用纹理映射可以在不增加场景描述复杂度，不显著增加计算量的前提下，大幅度地提高图形的真实感。

颜色纹理映射三个主要步骤：(1) 纹理函数定义；(2) 映射函数定义；(3) 纹理映射的实施

当真正改变物体曲表面各处的几何形状时称为**位移纹理**，有时只需表面呈现凹凸不平的状态，而不真正改变其几何性质，称为“视觉”上的**几何纹理**

过程纹理：构成纹理函数