单选：2×10；填空：1×9；简答：6×4；应用：9×3；综述：20

第一章：计算机图形学概述

1.基本概念
广义图形：凡是能够在人的视觉系统中形成视觉印象的客观对象都称为图形
计算机图形：由点、线、面、体等**几何要素**和**明暗、灰度（亮度）、色彩**等**非几何要素**构成的，从现实世界中**抽象**出来的带有灰度、色彩以及形状的图或形
2.计算机图形学
①概念：是研究如何在计算机环境下生成、处理和显示图形的一门学科
②研究内容：在计算机环境下景物的**几何建模方法**、对模型的**处理方法**、几何模型的**绘制技术**、图形输入和控制的**人机交互界面**
③解决的问题：利用计算机产生令人赏心悦目的真实感图形
④应用：计算机辅助设计与制造、地理信息系统、科学计算可视化、计算机动画与艺术、计算机模拟和仿真、虚拟现实
3.计算机图形的完整绘制流程
三个矛盾：①计算机不“认识”实际物体；②三维实体“撞上”二维屏幕；③连续性“碰到”像素小格
四个步骤（任务）：①用数学方法建立所需三维场景的几何描述，并将他们输入计算机；②将三维几何描述为二维透视图（投影、变换、裁剪等）；③将二维对象进行光栅化绘制；④进行显示（计算场景的光照、计算几何实体的颜色、材质和纹理，完成真实感图形绘制）
4.坐标系统【与第四章一起看】【简答题】【综述题】（理解）
①**模型坐标系**（局部坐标系）：构造单个对象的数字模型时，将其置于特定的坐标系下
②**世界坐标系**：为描述图形场景中所有图形之间的空间关系，将它们置于一个统一的坐标系中
③**设备坐标系**：在输出设备上建立的坐标系
④**观察坐标系**：根据视图所在的平面建立的坐标系
实际应用中，往往是对场景中的各个物体都建立自己的三维坐标系，以此建模，再把它们加入到统一的场景中，以简化场景的构造过程。
5.变换的模式
①全局模式（固定坐标系模式，图形模式）：每一次变换均可看成是相对于原始坐标系执行的，不会因为旋转而改变坐标系
矩阵合并角度：先调用的变换矩阵放在右边，后调用的变换矩阵放在左边
②局部模式（活动坐标系模式，空间模式）：连续执行几个变换时，每一次变换均可看成是在上一次变换所形成的新坐标系中进行的
矩阵合并角度：与全局模式相反，后调用的矩阵要乘在右边
6.OpenGL 的 glRotatef 旋转变换参数：以 glRotatef(90,0,0,1)为例，由(0,0,0)向(0,0,1)引一条直线，用右手握住这条直线（大拇指指向(0,0,1)），四指弯曲的方向即为旋转方向，旋转角度为 90°

第二章：计算机图形系统及硬件基础

1.计算机图形系统的构成
（1）计算机图形系统：用来生成、处理和显示图形
（2）构成：图形输入设备、中央处理器(CPU)、图形输出设备
①图形输入设备：键盘、鼠标、图形扫描仪等
②中央处理器：完成对图形的描述、建立、修改等各种计算，并对图形实现有效存储
③图形输出设备：
(i)图形显示设备：用于在屏幕上输出图形，液晶显示器、等离子显示器等
(ii)图形绘制设备：用于把图形画在纸上，打印机、绘图仪等
（3）基本功能：计算、存储、输入、输出、交互

（4）基本处理流程：表示→处理→显示
2.相关概念
①刷新：电子束从左到右从上到下将荧光屏扫描一次。
要保持屏幕上有稳定的图像就必须不断地发射电子束。
②隔行扫描：每一帧分为两个场显示，每个场只包含一半画面；两个场是交错的，一个场包含所有的奇数扫描行，另一个场包含所有的偶数扫描行
③帧缓存：是一块连续的存储空间，与屏幕上的像素一一对应，存储该像素的属性值（颜色、亮度）
④分辨率：缓冲存储器的容量越大，显示分辨率越高。分辨率越高，显示的字符或图像越清晰；显示器能支持的最大分辨率和显示器的“点距”有关。
⑤显示速度：显示字符、图形，特别是动态图像的速度
可用最大带宽表示：水平像素数*垂直像素数*最大刷新率
3.帧缓存的使用【简答题】（简单计算）
光栅中的每个像素在帧缓存中至少有 1 位，每个像素 1 位的存储容量称为位面；图形在计算机中是一位一位产生的，每个存储位只有 0 或 1 两个状态，因此一个位面的帧缓存只能产生黑白图形
【例】显示器分辨率为 m×n，计算 k 位位图需要的帧缓冲内存
帧缓冲内存=m*n*k bit/(8B/bit)
4.三种显示器的基本工作原理及系统组成【简答题】
①阴极射线管显示器(CRT)：
组成：阴极、电平控制器、聚焦系统、加速系统、偏转系统、阳极荧光粉涂层
原理：阴极被灯丝加热后，会发出电子并形成发散的电子云，由聚焦系统将电子聚焦成束，并由电平控制器控制电子束的强弱；聚焦后的电子束通过加速系统，轰击带正电的荧光粉涂层，产生能量，能量以光的形式释放出来，利用偏转系统精确定位，将需要的图形显示在屏幕上。
②液晶显示器(LCD)：
液晶：液态晶体，具有线性结晶结构的分子，加热可像液体那样流动，冷却则成结晶颗粒的固体状态
组成：玻璃板、偏光板（滤光片）、电极、背光板
原理：在液晶显示器中，液晶是灌入两个列有互相垂直的沟槽的上下夹层之间的，液晶分子被迫处于一种 90°扭转的状态；由于光线顺着分子的排列方向传播，所以透过偏光板的光线经过液晶时也被扭转 90°，光线通过；但加上电压时，分子变成竖立的状态，光线不扭转无法通过。即：不加电压，光线通过，显示屏上出现白色；加电压，光线不通过，显示屏上出现黑色。
彩色 LCD 原理：具备专门处理彩色显示的色彩过滤层，每一个像素都是由 3 个液晶单元格构成，其中每一个单元格前面都分别有红色、绿色或蓝色的过滤器；这样，通过不同单元格的光线，就可以在屏幕上显示不同的颜色。
③等离子显示器(PDP)：
组成：阴极、玻璃、电介质隔板、等离子管、透明阳极、气泡
原理：等离子显示器由密封在玻璃膜夹层中的晶格矩阵（光栅）组成，每个晶格充有低压气体；在高电压作用下，气体会电离解；当电子又重新与原子结合在一起时，能量就会以光子的形式释放出来，这时气体就会释放出具有特征的辉光。要点亮某个地址的气泡，开始要在相应行上加较高的电压，气泡点亮后，可用低电压维持气泡的亮度；要关掉某个气泡，只要将相应的电压降低；改变控制电压，可使等离子板显示不同灰度的图形。

【重点】第三章：基本光栅图形算法

1.直线生成算法：DDA 算法、Bresenham 算法【应用题】
（1）DDA 算法
①思想：舍入法求解最佳逼近，利用微分思想，即每一个点的坐标都可以由前一个坐标变化一个增量得到
②公式：设(x_i, y_i)是第 i 步得到的直线上的点，则直线上的第 i+1 个点是(x_{i+1}, y_{i+1})，其中：

$$x_{i+1} = x_i + dx \qquad y_{i+1} = y_i + dy$$

当 k≤1 时，x+1，y+k；当 k>1 时，y+1，x+1/k；取整方法：+0.5 以后强制类型转换为 int

③代码：

```
void DDA(Graphics g,int x1,int x2,int y1,int y2) {
    int k;
    float x,y,dx,dy;
    k=Math.abs(x2-x1);
    if(Math.abs(y2-y1)>k) k=Math.abs(y2-y1);
    dx=(float)(x2-x1)/k;dy=(float)(y2-y1)/k;
    x=(float)x1;y=(float)y1;
    for(int i=0;i<k;i++) {
        g.drawLine((int)(x+.5f),(int)(y+.5f),(int)(x+.5f),(int)(y+.5f));
        x=x+dx;y=y+dy;
    }
}
```

（2）Bresenham 算法

①思想：看精确值 y 与 y_i 及 y_{i+1} 的距离 d_1 及 d_2 的大小而定

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

③代码：

```
m=(double)dy/(double)dx;
e=m-0.5;
for(int i=0;i<dx;i++) {
    g.drawLine(x,y);
    if(e>=0) {y=y+1;e=e-1;}
    x=x+1;e=e+m;
}
```

为了消去除法和浮点数，令：f=2*e*dx，m'=2*dx*m；则有：m=2*dy，e=e-2*dx，代码变为：

```
void Bresenham(Graphics g,int xs,int ys,int xe,int ye) {
    int dx=xe-xs,dy=ye-ys;
    int e=2*dy-dx;
    int x=xs,y=ys;
    for(int i=0;i<dx;i++) {
        g.drawLine(x,y,x,y);
        if(e>=0) {y=y+1;e=e-2*dx;}
        x=x+1;e=e+2*dy;
    }
}
```

2.圆弧生成算法：正负法、Bresenham 算法、多边形迫近法【应用题】

判定式如何定义？迭代问题

（1）正负法

①思想：假设已选取 P_{i-1} 位第 i-1 个像素，则如果 P_{i-1} 在圆内，就要向圆外方向走一步；若已在圆外，就要向圆内走一步；总之，尽量贴近圆的轮廓线

②公式：$F(x,y) = x^2 + y^2 - R^2$

③代码：

```
void pnarc(Graphics g,int r) {
    int x=0,y=0+r,f=0;
    while(y>0) {
```

```
        g.drawLine(x,y,x,y);
        if(y>0) {f=f-2*y+1;y--;}
        else {f=f+2*x+1;x++;}
    }
    if(y==0) g.drawLine(x,y,x,y);
}
```

（2）Bresenham 算法

①思想：在候选的两个像素中，总数选定离圆弧最近的像素为圆弧的一个近似点

②公式：$d_i = D(H_i) + D(L_i) = (x_{i-1}+1)^2 + y_{i-1}^2 - R^2 + (x_{i-1}+1)^2 + (y_{i-1}-1)^2 - R^2$

③代码：

```
void Bresenham(Graphics g,int r) {
    int x=0,y=r,d=3-2*r;
    while(x<y) {
        g.drawLine(x,y,x,y);
        if(d<0) d=d+4*x+6;
        else {d=d+4*(x-y)+10;y--;}
        x++;
    }
    if(x==y) g.drawLine(x,y,x,y);
}
```

（3）多边形迫近法

①思想：将整个圆弧等分成一段段的短直线，用这些短直线形成的折线来逼近圆弧

②公式：设圆心为 C(0,0)，半径为 R，圆弧的起始角和终止角分别为 α_0、α_1，把圆弧分割成 n 份，则两个顶点的夹角为 $\alpha=(\alpha_0-\alpha_1)/n$。设内接正多边形的一个顶点为 $P_i(x_i,y_i)$，CP_i 的幅角为 θ_i，则：

$$x_i = R\cos\theta_i \quad y_i = R\sin\theta_i$$

顶点 P_{i+1} 的坐标为：

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

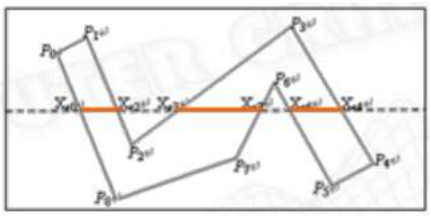
3.多边形填充【与第七章的 Z-buffer 扫描线算法一起看】【简答题】【综述题】

（1）多边形的扫描转换

①多边形的两种表示方法：顶点表示、点阵表示

②扫描转换：把多边形的顶点表示转换为点阵表示，也就是从多边形给定的边界出发，求出位于其内部的各个像素，并为帧缓存内的各个对应元素设置相应的灰度或颜色

（2）扫描线算法



①区域的连续性：多边形内部、多边形外部相间排列

②扫描线的连续性：多边形内的线段、多边形外的线段相间排列

③边的连续性：y 的值每增加 1，x 的值增加斜率分之一

④奇点的处理：极值点视为两个交点（如右图 P_6）

• ⑤数据结构

(i)活性边表(AEL)

对每一条扫描线，仅对与它相交的多边形的边求交运算，将这些边称为活性边

把活性边与扫描线的交点按 x 坐标递增的顺序放在一个链表中，即为活性边表

AEL 中的节点包含 4 个域：

y_{max}：边的上端点的 y 坐标，即与该边相交的最高扫描线号

x：边与扫描线的交点的 x 坐标

Δx：从当前扫描线到下一条扫描线间的 x 坐标的增量，即斜率的倒数

$Next$：指向下一条边的指针

(ii)新边表(NEL)

为方便 AEL 的建立与更新，要为每一条边建立一个新边表，存放在该扫描线上第一次出现的边
如果某边的较低端点为 y_{min}，则该边就放在扫描线 y_{min} 的新边表中，水平边不妨到任何扫描线的 NEL 中
NEL 的节点结构与 AEL 相同，只是 x 不再表示交点，而是表示该边较低端点的 x 坐标值
⑥基本思想/实现步骤：
计算扫描线与多边形各边的交点，设交点个数为 n；
把所有的交点按 x 值递增的顺序进行排列；
将排序后的第 1 个与第 2 个交点、第 3 个与第 4 个交点、…、第 n-1 个与第 n 个交点配对，每对交点就代表扫描线与多边形的一个相交区间；
把相交区间内的像素置成多边形的颜色，相交区间外的像素置成背景色。
（3）边缘填充算法：对图像逐位求补；偶数次求补后颜色不变，奇数次求补后颜色改变
• （4）边界标志算法
①基本原理：首先用特殊颜色在帧缓存中将多边形边界（水平边界除外）勾画出来，然后再把位于多边形内的各个像素着上所需的颜色
②实现步骤：
(i)以值为 boundary_color 的特殊颜色勾画多边形 P 的边界。设多边形顶点为 $P_i=(x,y),0\leqslant i\leqslant S_n$，$x_i$、$y_i$ 均为整数，置 $P_{n+1}=P_0$。每一条扫描线上着上这种特殊颜色的点的个数必定是偶数（包括 0）。
(ii)设 interior_point 是一布尔变量。对每一条扫描线从左到右进行搜索，如果当前是像素位于多边形 P 内，则 interior_point=true，需要填上值为 polygon_color 的颜色；否则该像素在多边形 P 外，需要填上值为 background_color 的颜色。
4.区域填充
（1）相关概念
①区域：已经表示成点阵形式的像素集合
②区域的表示方法：内点表示法、边界表示法
③区域的连通性：4 连通、8 连通
（2）简单种子填充算法（递归算法）
基本思想/步骤：给定区域 G 的一个种子点(x,y)，首先判断该点是否是区域内的一点，如果是，则将该点填充为新的颜色，然后将该点周围的四个点（四连通）或八个点（八连通）作为新的种子点进行同样的处理，通过这种扩散完成对整个区域的填充。
（3）扫描线种子填充算法
①基本思想：从给定的种子点开始，填充当前扫描线上种子点所在的一区段，然后确定与这一段相邻的上下两条扫描线上位于区域内的区段（需要填充的区间），从这些区间上各取一个种子点依次把它们存起来，作为下次填充的种子点；反复进行这过程，直到所保存的各区段都填充完毕。
②步骤：
(i)初始化：将算法设置的堆栈置为空，将给定的种子点(x,y)压入堆栈；
(ii)出栈：如果堆栈为空，算法结束；否则取栈顶元素(x,y)作为种子点；
(iii)区段填充：从种子点(x,y)开始，沿纵坐标为 y 的当前扫描线向左右两个方向逐个像素用新的颜色值进行填充，直到边界为止，即像素颜色等于边界色；设区间两边界的横坐标分别为 xleft 和 xright；
(iv)在与当前扫描线相邻的上下两条扫描线上，以区间[xleft,xright]为搜索范围，求出需要填充的各小区间，把各小区间中最右边的点并作为种子点压入堆栈，转到步骤(ii)

第四章：变换和裁剪

1.几何变换【应用题】（简单计算）
平移：

坐标关系：$\begin{cases}x'=x+\Delta x\\y'=y+\Delta y\\z'=z+\Delta z\end{cases}$　矩阵形式：$\begin{bmatrix}x'\\y'\\z'\end{bmatrix}=\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}\begin{bmatrix}x\\y\\z\end{bmatrix}+\begin{bmatrix}\Delta x\\\Delta y\\\Delta z\end{bmatrix}$

放大缩小：

坐标关系：$\begin{cases} x' = s_x x \\ y' = s_y y \\ z' = s_z z \end{cases}$ 矩阵形式：$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$

旋转：

绕 z 轴选择 α 角：$\begin{cases} x' = r\cos(\varphi + \alpha) = x\cos\alpha - y\sin\alpha \\ y' = r\sin(\varphi + \alpha) = x\sin\alpha + y\cos\alpha \\ z' = z \end{cases}$ 矩阵形式：$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$

绕 y 轴：$\begin{cases} x' = x\cos\alpha + z\sin\alpha \\ y' = y \\ z' = -x\sin\alpha + z\cos\alpha \end{cases}$ 绕 x 轴：$\begin{cases} x' = x \\ y' = y\cos\alpha - z\sin\alpha \\ z' = y\sin\alpha + z\cos\alpha \end{cases}$

A(x1,y1)绕点 O(x0,y0)旋转角度 k 之后的点 B(x2,y2)：

x2=(x1-x0)cosk+(y1-y0)sink+x0　　y2=-(x1-x0)sink+(y1-y0)cosk+y0

2.裁剪【简答题】【应用题】

（1）Sutherland-Cohen 算法（用矩形裁剪线段）

①思想：对于每条线段 P_1P_2 分为三种情况处理：

(i)若 P_1P_2 完全在窗口内（完全可见），则显示 P_1P_2

(ii)若 P_1P_2 完全在窗口外（完全不可见），则丢弃该线段

(iii)若不满足①②，则在交点处把线段分成两段，其中一段完全在窗口外，可以丢弃；另一段重复上述处理步骤

②编码规则：用窗口的四条边把整个平面分为九个区域，对每个区域编码，其中中心部分为 0000，左侧区域第 4 位为 1，右侧区域第 3 位为 1，下侧区域第 2 位为 1，上侧区域第 1 位为 1

③判断方法：对要裁剪的线段的两个端点，如果区域编码都是 0000，则完全可见；若逻辑与的结果不为 0000，则完全不可见；若并非上述情况，则求出与窗口的交点并截断，丢弃完全不可见的部分，然后再次对端点编码，重复上述判断直至完全可见

（2）Sutherland-Hodgman 算法（用矩形裁剪多边形）

①原理：只需要对多边形用窗口的四条边依次裁剪四次，即可得到裁剪后的多边形

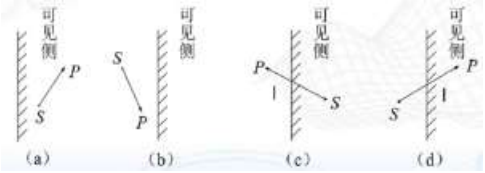
②实现：每条线段的端点 S（起点）、P（终点）与裁剪线比较后可输出 0~2 个顶点：

(i)若 S、P 都在可见的一侧，则输出 P

(ii)若 S、P 都在不可见的一侧，则不输出

(iii)若 S 在可见的一侧，P 在不可见的一侧，则输出 SP 与裁剪线的交点 I

(iv)若 S 在不可见的一侧，P 在可见的一侧，则输出 SP 与裁剪线的交点 I 和 P



（3）Cyrus-Beck 算法（用多边形裁剪线段）

①思想：对于线段 P_1P_2 的参数方程表示，如果能判断出线段进入多边形的参数 ts 和退出多边形的参数 t_e，则 t_st_e 之间的线段即为裁剪完毕后的结果

• ②判定方式：对于线段 P_1P_2，可用参数方程表示　$P(t) = (P_2 - P_1)t + P_1 \quad 0 \le t \le 1$

记多边形的一条边为 L，其内法矢量为 N，A 是 L 上的一点，则 P 和 N 的关系如下：

(i)若 N • (P-A)>0，则 P 在 L 内侧

(ii)若 N • (P-A)=0，则 P 在 L 或其延长线上

(iii)若 N • (P-A)<0，则 P 在 L 外侧

第五章：三维空间的观察

1.投影变换的要素：视点（投影中心）、投影平面（不经过视点的平面）、投影线（从视点向投影平面上引出的射线）、投影方向
2.投影变换的类型：
①透视投影：投影中心和投影平面之间的距离是有限的
②平行投影：投影中心和投影平面之间的距离是无限的
3.投影的计算【应用题】
①透视投影
已知：在坐标系$\bar{o}\bar{x}\bar{y}\bar{z}$中来讨论投影，假定投影平面是$\bar{z}=0$，设视点为$\bar{C}(\bar{x}_C,\bar{y}_C,\bar{z}_C)$，空间中任一点$\bar{Q}(\bar{x},\bar{y},\bar{z})$；求解：$\bar{Q}(\bar{x},\bar{y},\bar{z})$在$\bar{z}=0$平面上的投影$\bar{P}(\bar{x}_P,\bar{y}_P)$
解：由定义进行推导，视点与物体的连线与投影面的交点即为物体点的投影点，连接$\bar{Q}$、$\bar{C}$，得直线：

$$\frac{\bar{x}_P-\bar{x}_C}{\bar{x}_Q-\bar{x}_C}=\frac{\bar{y}_P-\bar{y}_C}{\bar{y}_Q-\bar{y}_C}=\frac{\bar{z}_P-\bar{z}_C}{\bar{z}_Q-\bar{z}_C}$$

与投影方程$\bar{z}_P=0$联立，求解方程组，得：

$$\bar{x}_P=\bar{x}_C+(\bar{x}-\bar{x}_C)\frac{\bar{z}_C}{\bar{z}_C-\bar{z}}\qquad \bar{y}_P=\bar{y}_C+(\bar{y}-\bar{y}_C)\frac{\bar{z}_C}{\bar{z}_C-\bar{z}}$$

②平行投影
已知：在坐标系$\bar{o}\bar{x}\bar{y}\bar{z}$中来讨论投影，假定投影平面是$\bar{z}=0$，设投影方向为$\bar{D}(\bar{x}_D,\bar{y}_D,\bar{z}_D)$，空间中任一点$\bar{Q}(\bar{x},\bar{y},\bar{z})$；求解：$\bar{Q}(\bar{x},\bar{y},\bar{z})$在$\bar{z}=0$平面上的投影$\bar{P}(\bar{x}_P,\bar{y}_P)$
解：由定义进行推导，视点与物体的连线与投影面的交点即为物体点的投影点，根据投影方向得直线：

$$\frac{\bar{x}_P-\bar{x}_Q}{\bar{x}_D}=\frac{\bar{y}_P-\bar{y}_Q}{\bar{y}_D}=\frac{\bar{z}_P-\bar{z}_Q}{\bar{z}_D}$$

与投影方程$\bar{z}_P=0$联立，求解方程组，得：

$$\bar{x}_P=\bar{x}-\frac{\bar{x}_D}{\bar{z}_D}\bar{z}\qquad \bar{y}_P=\bar{y}-\frac{\bar{y}_D}{\bar{z}_D}\bar{z}$$

第六章：人机交互绘图技术

1.基本的交互任务：定位、笔画、定值、选择、拾取、字符串
2.人机交互的输入模式：
①请求模式：指定是哪个应用程序调用的、调用的哪个输入设备
②样本模式：可同时处理多个输入设备的输入信息；可能会丢失某些输入信息
③事件模式：输入设备和程序独立运行

第七章：可见面的判定

1.可见面判断的有效技术
（1）边界盒
①概念：能够包含该物体的一个简单的几何形状，如矩形、圆、长方体等
②简单的求边界盒的方法：通过计算多边形顶点坐标的最大值和最小值得到
（2）后向面消除
①相关概念：
(i)外法向与内法向：多面体表面多边形的法向可分为两种，一种是指向多面体的外部（外法向），一种指向多面体的内部（内法向）
(ii)后向面：必然有一些多边形表面的外法向指向与观察者相背离的方向，这些多边形完全被多面体上其它多边形遮挡。这些被遮挡的多边形称为后向面。

(iii)后向面消除：首先消除掉这些面，去除后向面的过程称为后向面消除
②思路：把显然不可见的面去掉，减少消隐过程中的直线求交数目
2.基于窗口的细分算法（从像素出发）
思想：把物体投影到全屏幕窗口上，然后递归的将窗口一分为四；如果可以确定小窗口内的多边形是否可见，则显示这些多边形；否则将小窗口细分为更小的窗口。递归地执行上述过程，每一次把矩形的窗口等分成四个相等的小矩形。
3.基于多边形的细分算法（从物体出发）
基本思想:用多边形的边界对区域作划分,其目的是尽量减少对区域划分的次数。可以利用裁剪算法。该算法是对基于窗口细分算法的改进。由于算法在景物空间中以任意指定的精度进行运算，其输出结果仍为多边形，所以算法不仅可用来处理隐藏面消除，也可用来处理多面体隐藏线消除问题。
4.z 缓冲器及其扫描线算法
（1）z 缓冲器：是一组存贮单元，其单元个数和屏幕上的像素个数相同，也和帧缓存的单元个数相同，它们之间一一对应；帧缓冲器每个单元存放对应像素的颜色值，Z 缓冲器每个单元存放对应像素的深度值（z 值）
帧缓冲器：Frame[][]；Z 缓冲器：Zdepth[][]
（2）z 缓冲器算法
①基本思想：对屏幕上每一个像素点，过像素中心做一条投影线，找到此投影线与所有多边形交点中离观察者最近的点，此点的属性（颜色或灰度）值即为这屏幕像素点的属性值。
②具体步骤：
(i)对屏幕上的每个点(x,y)，令 Zdepth[x][y]为 z 的极小值，Frame[x][y]为背景颜色
(ii)对所有多边形做如下工作：对多边形上在像素中心有投影的每一点(x,y)，计算其 z 值；若 z>Zdepth[x][y]，则将此点属性值赋给 Frame[x][y]，同时 z 缓冲器中相应单元的值也要改成这个点的 z 坐标值；否则说明此点离观察者较远，两个数组的值不用改变
③优点：简单可靠，不需要对显示物体的表面预先进行排序，有利于硬件实现
④缺点：需要一个额外的 z 缓冲器，内存庞大
•（3）扫描线 z 缓冲器算法
①算法引入：为了克服 z 缓冲器算法的缺点，可把整个平面分成若干区域，一区一区来显示，这样 z 缓冲器的单元数只需等于屏幕上一个区域的像素个数。如果把这个区域取成屏幕上一行就得到了扫描线 z 缓冲器算法，这时 z 缓冲器的单元数和一条扫描线上的像素数目相同。
②具体步骤（与 z 缓冲器算法类似）：将 z 缓冲器的单元数置为和一条扫描线上的像素数目相同，从最上面的一条扫描线开始工作，向下对每一条扫描线作如下处理：
(i)把相应的帧缓冲器单元置成底色，z 缓冲器中存放 z 的极小值
(ii)对每个多边形检查它在平面上的投影和当前的扫描线是否相交；若不相交，则不考虑该多边形；如果相交，则扫描线和多边形边界的交点是成对地出现对每对交点中间的像素计算多边形所在平面对应点的深度（即 z 值），并和 z 缓冲器中相应单元存放的深度值作比较；若大于 z 缓冲器中的值，则 z 缓冲器的相应单元内容要被求得的平面深度代替，帧缓冲器相应单元的内容也要换成该平面的属性。
③数据结构：多边形 Y 表、边表(ET)、活化多边形表(APT)、活化边对标(AET)
(i)Y 表：记录的个数和扫描线的行数相同，根据多边形顶点 y 坐标最大值来决定将多边形放入 Y 表的对应行数，根据边两端点较大的 y 坐标值来确定放入边表的对应行数
记录多边形所在屏幕方程 ax+by+cz+d=0 的 a、b、c、d 的值，和该多边形在 Oxy 平面上的投影相交的扫描线条数Δy，多边形的属性 Color 和编号 IP
(ii)边表：记录的个数和扫描线的行数相同，每条记录保存边的上端点的 x 坐标值、边在 Oxy 平面上的投影和相邻两条扫描线的交点的 x 坐标值的差Δx、边在 Oxy 平面上的投影所覆盖的扫描线的条数Δy、边所属多边形的编号 IP
(iii)APT：记录了在 Oxy 平面上的投影和当前考虑的扫描线相交的多边形，内容：a、b、c、d、IP、Δy、color
(iv)AET：存放多边形的边和当前扫描线相交的边对，边对的两条边分别称为左边、右边，信息如下：

x_1：左边交点的 x 坐标值
Δx_1：左边和两相邻扫描线交点的 x 坐标之差
Δy_1：以左边的投影所覆盖的扫描线条数为初值，以后每处理一条扫描线减 1
x_r：右边交点的 x 坐标值
Δx_r：右边和两相邻扫描线交点的 x 坐标之差
Δy_r：以右边的投影所覆盖的扫描线条数为初值，以后每处理一条扫描线减 1
z_1：多边形平面在左交点处的深度值
Δz_x：沿扫描线向右走过一个像素时，多边形所在平面深度的增量，$\Delta z_x = -a/c\ (c \neq 0)$
Δz_y：沿 y 方向向下移过一根扫描线时，多边形所在平面深度的增量，$\Delta z_y = b/c\ (c \neq 0)$
IP：所在多边形的编号

第八章：光照明模型

1.基本概念
①光源：所有可以发出辐射光能的物体统称为光源，如日光灯、太阳等
②材质：物体所呈现出的颜色在很大程度上取决于物体表面的材质
2.几种颜色模型：RGB 模型、CMY 模型、HSV 模型、YIQ 模型
3.光照明模型
反射光组成：环境反射、漫反射、镜面反射
要计算某一点的光亮度，就要分别求出这三个分量：$I=I_{环}+I_{漫}+I_{镜}$
①环境反射：$I_e = k_a I_{pa}$（郎伯余弦定律）
I_e 为物体的环境光反射亮度；I_{pa} 为环境光亮度；k_a 为物体表面的环境光反射系数（$0 \leqslant k_a \leqslant 1$），表明了当光射向物体表面时，物体表面向各个方向漫反射该光线的能力
②漫反射：对于一个漫反射体，表面的反射光亮度和光源入射角（入射光线和表面法向量的夹角）的余弦成正比，即：$I_d = k_d I_{pd} \cos i$
I_d 为物体表面漫反射的光亮度；I_{pa} 为光源垂直入射时的光亮度；i 为光源入射角；k_d 为漫射系数，决定于表面材料及入射光的波长($0 \leqslant k_d \leqslant 1$)
表面对入射光在各个方向上都有强度相同的反射，因而无论从哪个角度观察，这一点的光亮度都是相同的。
③镜面反射：镜面反射光为朝一定方向的反射光，采用余弦函数的幂次来模拟一般光滑表面的镜面反射光的空间分布　　$I_s = k_s I_{ps} \cos^n \theta$
I_s 为观察者接受到的镜面反射光亮度；I_{ps} 为入射光的光亮度；k_s 为镜面反射系数（与材料性质和入射光波长有关）；n 为镜面反射光的会聚指数（与物体表面光滑度有关）；θ 为镜面反射方向和视线方向的夹角，介于 0° 到 90° 之间
④Phong 模型：三个分量求出后，相加得最后的光照结果：$I = I_{pa}k_a + I_p(k_d \cos i + k_s \cos^n \theta)$
当光源有多个时，有$I = I_{pa}k_a + \sum(I_{pd}k_d \cos i + I_{ps}k_s \cos^n \theta)$，且有$k_d + k_s = 1$
*4.光滑明暗处理技术
①Gouraud 明暗处理：将曲面表面某一点的光亮度做近似表示，近似值为该曲面各多边形顶点光亮度的双线性插值
②Phong 明暗处理：采用简单光照明模型计算光亮度，对多边形顶点处法向量做双线性插值