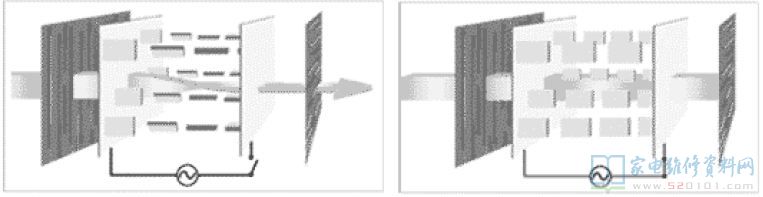
1. 液晶显示器的显示原理

Liquid Crystal Display，简称LCD

液晶的电光效应：当液晶分子的某种排列状态在电场作用下变为另一种排列状态时，液晶的光学性质随之改变。液晶显示的机理是通过能阻塞或传递光的液晶材料，传递来自周围的或内部光源的偏振光

液晶显示器(LCD/Liquid Crystal Display)的显像原理，是将液晶置于两片导电玻璃之间，靠两个电极间电场的驱动，引起液晶分子扭曲向列的电场效应，以控制光源透射或遮蔽功能，在电源关开之间产生明暗变化，从而将影像显示出来。若加上彩色滤光片，则可显示彩色影像。在两片玻璃基板上装有配向膜，所以液晶会沿着沟槽配向，由于玻璃基板配向膜沟槽偏离90度，所以液晶分子成为扭转型，当玻璃基板没有加入电场时，光线透过偏光板跟着液晶做90度扭转，通过下方偏光板，液晶面板显示白色（如图1－3－1左）；当玻璃基板加入电场时，液晶分子产生配列变化，光线通过液晶分子空隙维持原方向，被下方偏光板遮蔽，光线被吸收无法透出，液晶面板显示黑色（如图1－3－1右））。液晶显示器便是根据此电压有无，使面板达到显示效果。



1. 可编程渲染管线的流程

流水线中的三个概念阶段:

应用阶段->几何阶段->光栅化阶段

应用阶段: 将需要在屏幕上显示出来绘制的几何体，也就是绘制图元，比如点、线、矩形等输入到绘制管线的下一个阶段。具体包括图元的顶点数据、摄像机位置、光照纹理等参数。

几何阶段: 几何阶段需要将顶点数据最终进行屏幕映射。

这其中需要：

将各个图元放入到世界坐标系中，也就是进行模型变换；

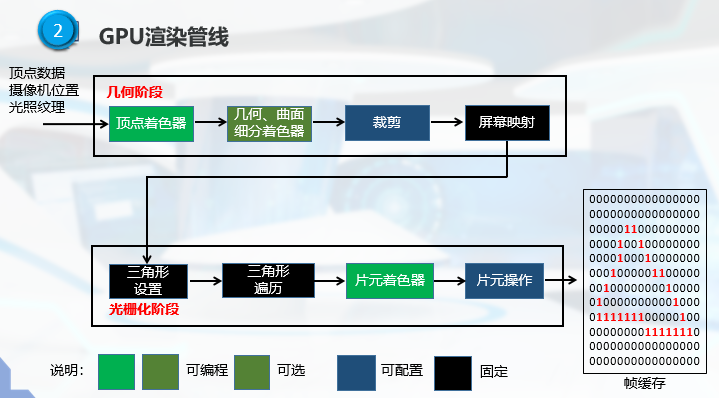
根据光照纹理等计算顶点处材质的光照着色效果；

根据摄像机的位置、取景范围进行观察变换和裁剪；

最后进行屏幕映射，也就是把三维模型转换到屏幕坐标系中。

光栅化阶段: 光栅化部分的输入是经过变换和投影后的顶点、颜色以及纹理坐标，它的工作是给每个像素正确配色，以便绘制整幅图形。由于输入的是三角形顶点，所以需要根据三角形表面的差异，逐个遍历三角形计算各个像素的颜色值。之后根据其可见性等进行合并得到最后的输出。





**GLSL在OpenGL中使用着色器的流程：创建着色器对象->源码关联到着色器对象->编译着色器->创建一个程序对象->创建一个程序对象**

**EBO:EBO(Element Buffer Object，也叫IBO:Index Buffer Object)索引缓冲区对象，这个缓冲区主要用来存储顶点的索引信息。**

**VBO(Vertex Buffer Object )顶点缓冲区对象，主要用来存储顶点的各种信息。好处：模型的顶点信息放进VBO，这样每次画模型时， 数据不用再从CPU的势力范围内存里取，而是直接从GPU的显存里取，从而提高效率。**

VAO(Vertex Arrary Object)顶点数组对象VAO是一个保存了所有顶点数据属性的状态结合，它存储了顶点数据的格式以及顶点数据所需的VBO对象的引用。VAO本身并没有存储顶点的相关属性数据，这些信息是存储在VBO中的，VAO相当于是对很多个VBO的引用，把一些VBO组合在一起作为一个对象统一管理。

1. 椭圆中点算法的推导，要求写出推导的过程

见椭圆中点算法.ppt

1. 课件给出的规则和非规则造型技术的概念和举例

规则对象:指能用欧氏几何进行描述的形体，如点、直线、曲线、平面、曲面或实体等。规则对象的造型又称为几何造型。在几何造型中，所描述的形体都是规则物体，统称为几何模型。

不规则对象:指不能用欧氏几何加以描述的对象，如山、水、树、草、云、烟等自然界丰富多彩的对象。在不规则对象的造型系统中，大多采用过程式模拟，即用一个简单的模型以及少量的易于调节的参数来表示一类对象。

1. 几何阶段：视图变换的过程

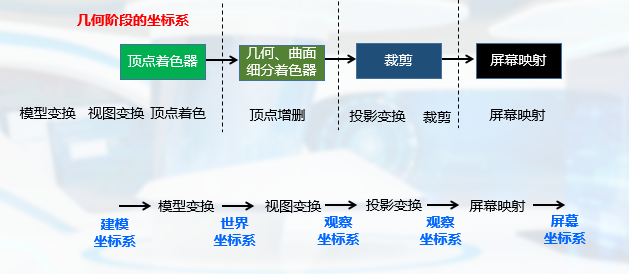
1、从不同的位置去观察它（人运动，选定某个位置去看）。（视图变换）

2、移动或者旋转它，当然了，如果它只是计算机里面的物体，我们还可以放大或缩小它（物体运动，让人看它的不同部分）。（模型变换）

3、如果把物体画下来，我们可以选择：是否需要一种“近大远小”的透视效果。另外，我们可能只希望看到物体的一部分，而不是全部（指定看的范围）。（投影变换）

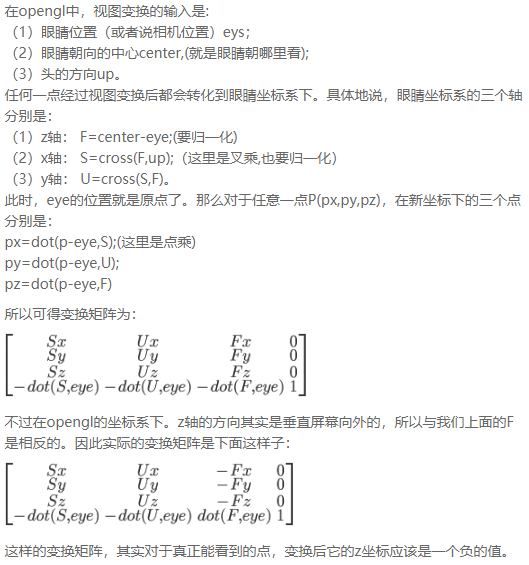
4、我们可能希望把整个看到的图形画下来，但它只占据纸张的一部分，而不是全部（指定在显示器窗口的那个位置显示）。（视口变换）

视图变换，这个过程是调整Camera到合适的位置以便拍摄, 设置摄像机的位置，gluLookAt(摄像机位置，镜头瞄准，上方向)，其中上方向的理解就类似是一个放好的瓶子，一般指定垂直向上是瓶盖的方向。默认摄像机位于原点，指向z轴的负方向，朝上向量为(0, 1, 0).





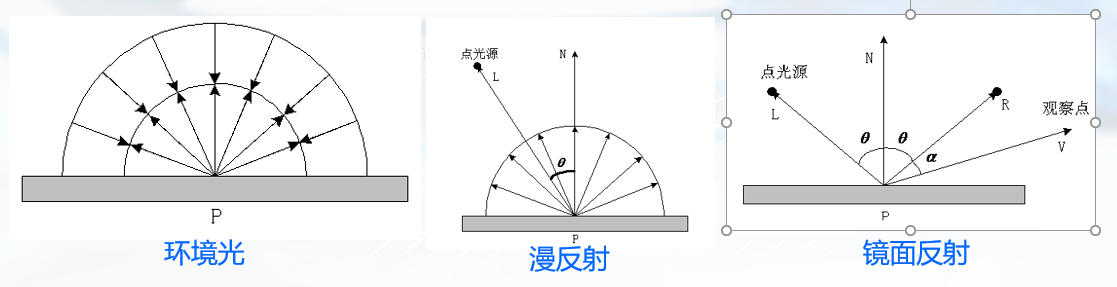
视图变换（ VIEW Transformation ）：它类似将照相机指向物体，即确定视点（观察点）的位置和观察方向。一般用的函数为 glu 封装的函数：void gluLookAt(GLdouble eyex, GLdouble eyey, GLdouble eyez, --------- 观察点GLdouble centrex, GLdouble centrey, GLdouble centrez, -- 视线方向：从 eye 指向 centreGLdouble upx, GLdouble upy, GLdouble upz ------------ 视图体自下而上的方向)这个函数会产生一个视图矩阵，并右乘到当前矩阵上。模型变换通常发生在模型变换之前。其实，视图变换也是通过平移和旋转得到的，观察位置与物体位置之间是个相对的状态，我们也把视图变换和模型变换统一成一个变换，产生一个矩阵：模型视图变换矩阵。



1. Phong模型包括那几个部分，程序中的光照计算，Blinn-Phong对哪些部分处理了近似，要求绘图

Phong模型

几部分：环境光(Ambient Light)漫反射(Diffuse Reflection)镜面反射(Specular Reflection)，下面是一些概念理解，及公式（应该只需图及公式）

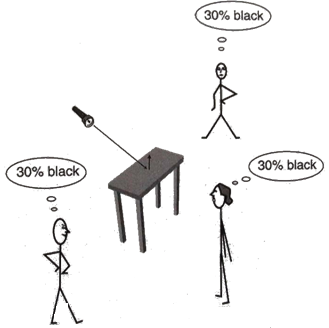


环境光（见上图环境光）：光线在场景中经过复杂的传播之后，形成了弥漫于整个空间的光线，同一环境下的环境光光强分布均匀，近似表示：=,环境光强，度物体对环境光的反射系数环境光强度

（理想漫反射）漫反射（见上图漫反射）的特点：光源来自一个方向，反射光均匀地射向各个方向，与视点无关，由Lambert余弦定律，漫反射光强为=×cosθ

其中：是与物体有关的漫反射系数，0< <1，是入射光的强度

多个点光源（m个点光源）,则直接各个光源的漫反射进行叠加



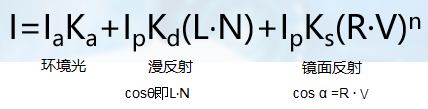
镜面反射（见上图镜面反射）：遵循反射定律，反射光位于表面法矢的两侧

=∙

cos α =R ∙ V其中 α为视点方向V与镜面反射方向R之间的夹角，且V、R均规格化为单位向量，是与物体有关的镜面反射系数，n为反射指数，反映物体表面的光泽程度，数目越大物体表面越光滑

多个点光源（m个点光源）,则直接各个光源的镜面反射进行叠加

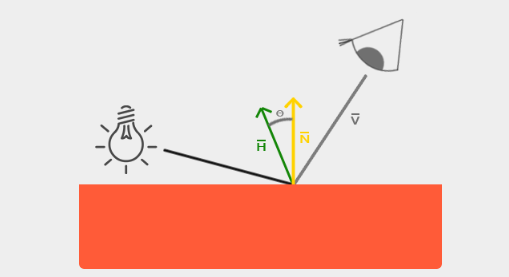
**综合表示（假定只有一个光源）**：由物体表面上一点P反射到视点的光强**I**为环境光的反射光强、理想漫反射光强和镜面反射光强的总和：



**Blinn-Phong对哪些部分处理了近似：**

****

**H = (L + V) / | L+V |，计算H比计算反射向量R更快速，这里的是向量运算（求单位向量）。**我们不是依赖于反射向量而是使用所谓的反射向量中途矢量这是一个正好在视图方向和光线方向之间的单位矢量。该中间矢量越接近表面的法向量，镜面贡献越高。下图是blinn-Phong的镜面反射图：（其他的环境光和漫反射跟上面的一样）



1. 光线追踪算法的主要思想，要求绘图

**具体见 光线追踪算法.pptx**

1. 法线贴图的主要思想，要求绘图

见 **法线贴图.pptx**

每个fragment使用了自己的法线，我们就可以让光照相信一个表面由很多微小的（垂直于法线向量的）平面所组成，物体表面的细节将会得到极大提升。这种每个fragment使用各自的法线，替代一个面上所有fragment使用同一个法线的技术叫做法线贴图。

1、（普通贴图）通过height map获得法向量信息（高度图），即对应的RGB值表示法向量的XYZ，利用这个信息计算光强，产生凹凸阴影的效果。普通贴图+高度图

2、（多个面）**切线空间的引入：将相对于某个平面的法向量变换到全局中，每个平面都有一个自己的切线空间，T：tangent 切线 B：bitangent 副切线 N：normal法线。通过三个共面且不是共线的点P1、P2和P3分别计算TBN的方向量，TBN的方向量构成TBN矩阵**

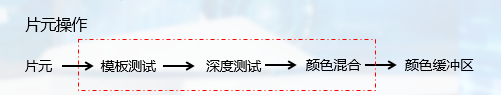




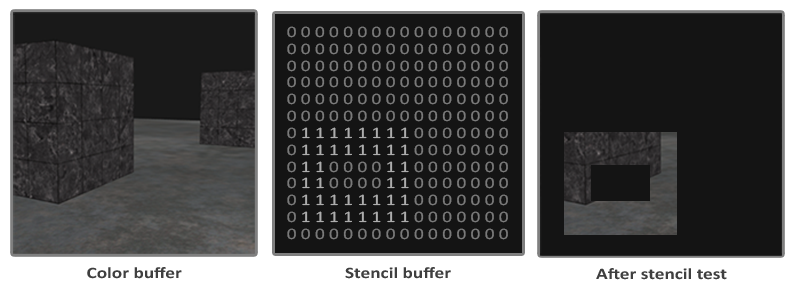
1. 阴影算法shadow mapping的主要思想，要求绘图

阴影算法.pptx

1. 片元操作的几个主要步骤



模板测试是对每一帧中的每个像素进行测试，通过和一个模板指定的对应像素进行对比决定该像素是否能通过测试，模板测试就是通过模板缓冲区中记录的模板信息完成的，首先需要初始化模板缓冲区，这可以通过渲染几何形状并制定模板缓冲区的更新方式来完成；接下来就可以使用指定的模板缓冲区中的模板信息进行其他的绘制了。模板测试原理，比如模板缓冲区如图所示，当绘制图形时设置模板缓冲区值等于1时才绘制的话，那么接下来绘制的时候就只会在模板缓冲区设置为1的部分绘制，其他位置的片元都会被丢弃。



深度就是像素点在距离摄象机的距离，对应的也有一个深度缓冲区存储着每个像素点（绘制在屏幕上的）的深度值，深度值越大，表示离摄像机越远。

深度测试的默认做法：对于每个新的输入片元，将其与视点的距离和深度缓冲区中的值进行比较，如果输入的片元的深度值小于深度缓冲区中的深度值，则表示它离视点更近，则输入片元的深度将代替深度缓冲区中的值，其颜色值将代替颜色缓冲区中的颜色值。否则，表示当前要绘制的片元在已绘制的部分物体后面，则无需绘制该图形，即丢弃。

混合就是把某一像素位置原来的颜色和将要画上去的颜色通过某种方式混在一起。 即某个片元的颜色（源片元）和颜色缓冲区中的像素颜色（目标片元中的像素颜色）进行混合。

1. PVM矩阵

opengl在画全景图时最后会调用glDrawElements或者glDrawArrays，而投影映射的shader中一般都会有一个MVP矩阵作为输入，uMVPMatrix。这个矩阵表示了画图时的视角。它由三个矩阵按PVM的顺序相乘得到： view矩阵 X model矩阵 X projection矩阵。（X乘）

projection矩阵：用于最终投影出2D输出平面上的坐标，opengl在显示时，首先通在3D空间画出整个图像，对于没有特殊变换的全景视频来说就是画一个球。而显示时显示的是一个2D的图像，这时就需要投影变换

view矩阵：用于设定视角方向，也就是把world space变换到eye space，view矩阵相当于在3D空间中放置一个照相机，摆在某个位置，并且调整相机方向对准某个向量

model矩阵：将物体的坐标从object space变换到画图时opengl用的3D空间，world space。如果本身输入就在opengl的空间，比如全景视频，则设置为单位矩阵即可，因为不需要这部变换。对于全景视频来说，是直接贴在opengl的三维空间上的球，所以不需要进行object space到world space的转换，这个矩阵直接设置为单位矩阵就可以了。

1. 片源着色器

场景渲染到显示器的过程中，有一个步骤叫光栅化（Rasterization）。由于我们现在的显示器绝大多数是基于像素的（就是由一个个非常小的红绿蓝 LED 组成的显示单元），所以“连续”的三维场景，要显示到“离散”的显示器上，需要经过的变化操作就叫光栅化。

光栅化后得到的就得到了一个个“片元”。片元和像素已经非常接近了，但两者仍是有区别的。用一种通俗的说法来解释的话，就是比如三维空间内有两个从摄像机角度看过去一前一后的三角形，它们重叠部分的显示区域，每个像素对应两个片元；不重叠的部分，像素和片元一一对应。当然，这个例子是我简化过的，真实的对应关系可能更复杂一些。

更专业的说法是，片元在成为像素之前，还会做多种测试（比如深度测试、透明度测试、模板测试）以决定其最终是否会被显示为像素。所以，严格来说，“片元”和“像素”并不是一一对应的。

同样，片元着色器就是每个片元调用一次的程序。在片元着色器中，可以访问到片元在二维屏幕上的坐标、深度信息、颜色等信息。通过改变这些值，可以实现特定的渲染效果。注意，同样是颜色信息，在顶点着色器中，得到的是顶点的颜色，而在片元着色器中，得到的是片元的颜色——也就是说，如果三角形的三个顶点颜色是不同的，片元的颜色就是根据这三个顶点的颜色进行插值后的，也可以通俗地理解为，是渐变的。

1. 裁剪

裁剪.ppt

1. **扫描转换的概念**

图形的光栅化（图形的扫描转换）分成两步：

第一步：根据图形的定义在点阵单元上确定最佳逼近于图形的象素集，逼近的过程本质可以认为是连续量向离散量的转换

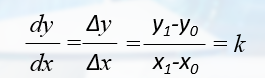
第二步：给像素指定合适的颜色值

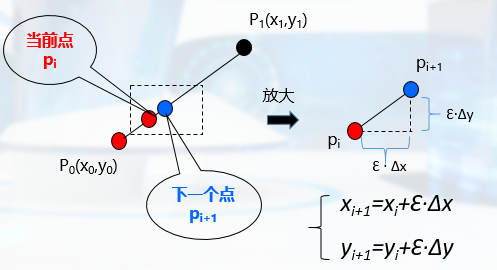
输入：三角形的定义（1）几何信息：三个顶点的坐标（2）属性信息：颜色、光照、纹理等

**点： 输入：点的坐标 输出：像素点的位置**

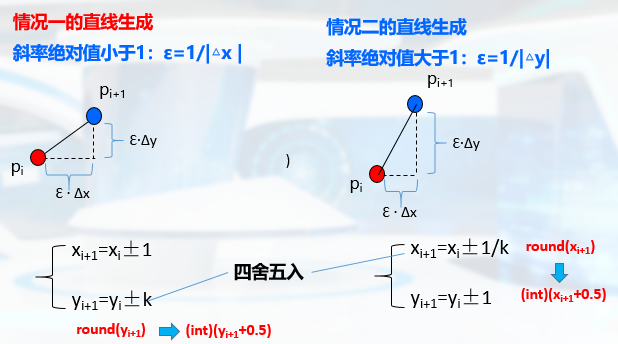
**直线： 输入：直线两个端点的坐标P0(x0,y0)和P1(x1,y1) 输出：最佳逼近这条直线的像素点集 高质量直线的要求：1）直线要直（锯齿效果小），2）直线的端点要准确，即无定向性和断裂情况（无定向性，从A到B与从B到A的直线要重合）3）直线的亮度、色泽要均匀（线宽处理，宽则亮度较弱）4）画线的速度要快，还能处理不同线宽、颜色、线型**

**数值微分法:** **通过给定直线的两端点坐标P0(x0,y0)和P1(x1,y1)，我们可以得到直线的微分方程：**





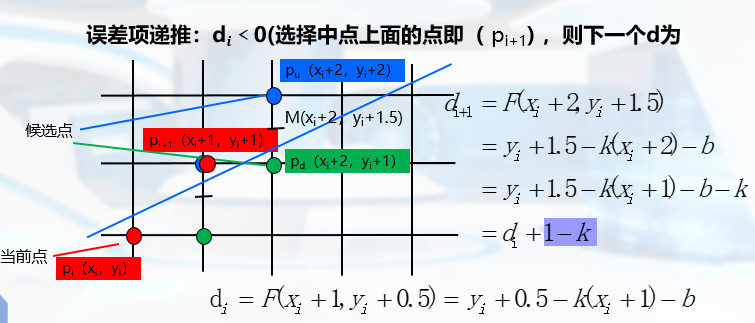
**精度无限高：Ɛ足够小,** **设备的精度是有限的,** **令ε=1/max(|△x|,|△y|)使得ε△x或ε△y中会有一个变成单位步长算法在最大位移方向上，每次总是走一步**

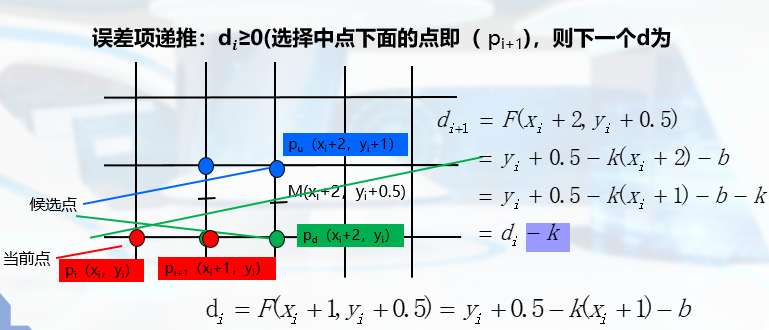


**在0≤k≤1情况下的整数的中点Bresenham算法：*F(x,y)=y-kx-b***

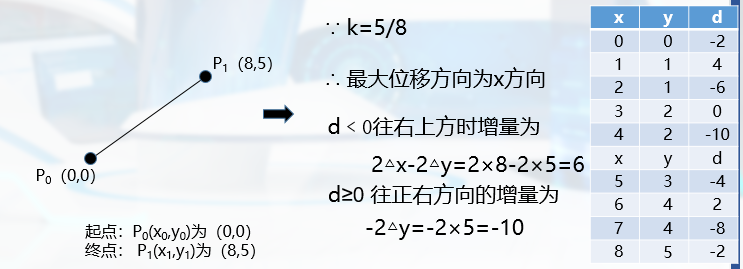
1. **输入直线的两端点P0(x0,y0)和P1(x1,y1)**
2. **计算初始值△x、△y、d=△x-2△y=（x1-x0）-2(y1-y0)、x=x0、y=y0**
3. **绘制点(x,y) 判断d的符号： 若d<0，则(x,y)更新为(x+1,y+1)，d更新为d+2△x-2△y； 否则(x,y)更新为(x+1,y), d更新为d-2△y**
4. **当直线没有画完时，重复步骤3，否则结束**

**推导：（d放大2△x倍得到上面的算法）**

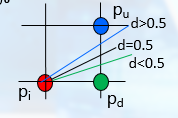




**一个例子：（d放大2△x倍）f*(x,y)=2y△x-2x△y（-2b*△x）**



**在0≤k≤1情况下改进的Bresenham算法：**



**（1）输入直线的两端点P0(x0,y0)和P1(x1,y1)。**

**（2）计算初始值△x、△y、e=-△x、x=x0、y=y0。**

**（3）绘制点(x,y)。**

**（4）e更新为e+2△y**

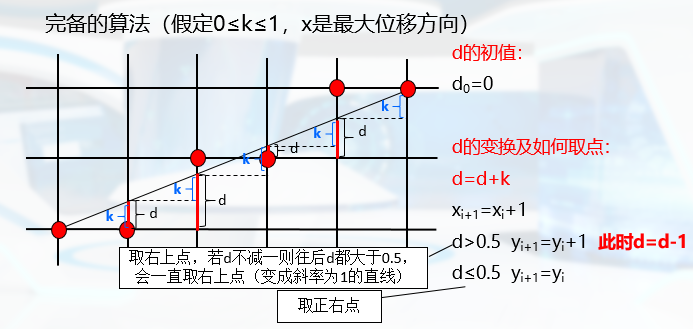
**判断e的符号**

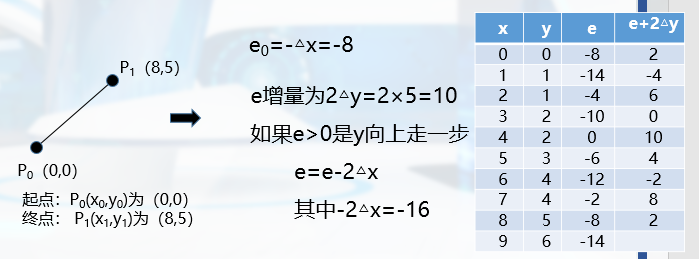
**若e>0，则(x,y)更新为(x+1,y+1)，同时将e更新为e-2△x；**

**否则(x,y)更新为(x+1,y)。**

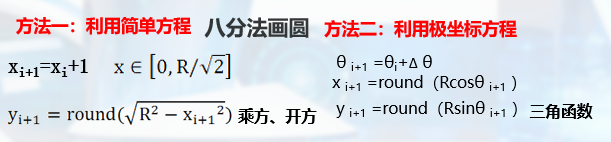
**（5）当直线没有画完时，重复步骤3和4。否则结束。**

**推导：（d放大2△x倍就变成上面的e,然后就不用跟0.5比较，变为e跟0比较）**

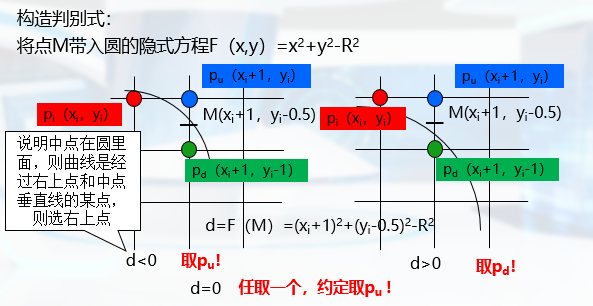


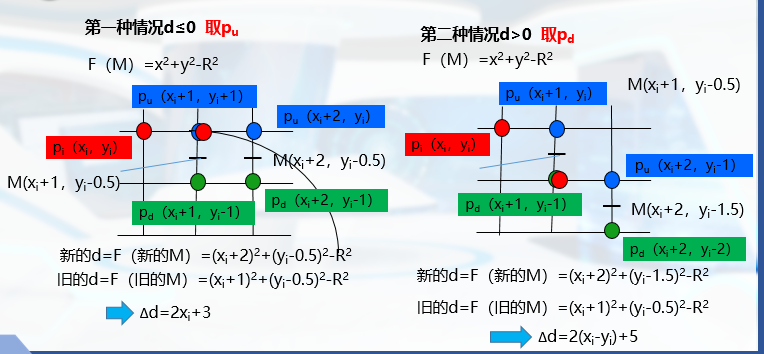


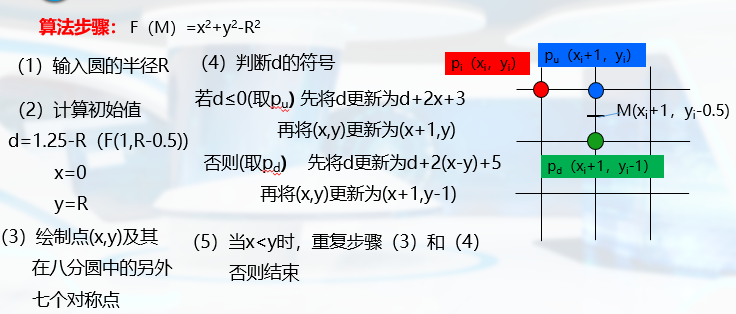
**圆：（这里只讨论0<|k|<1）**



**中点画圆法：**

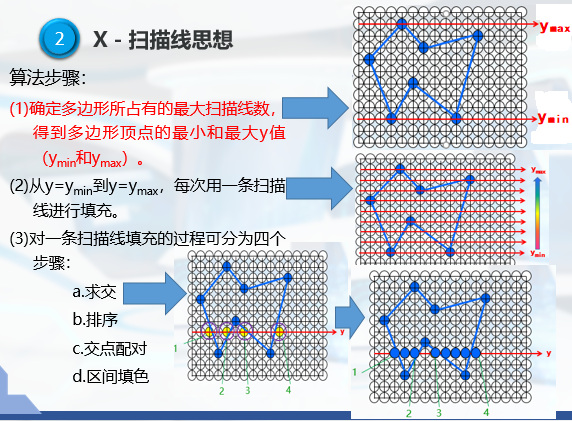


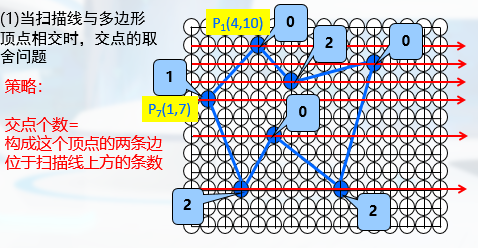


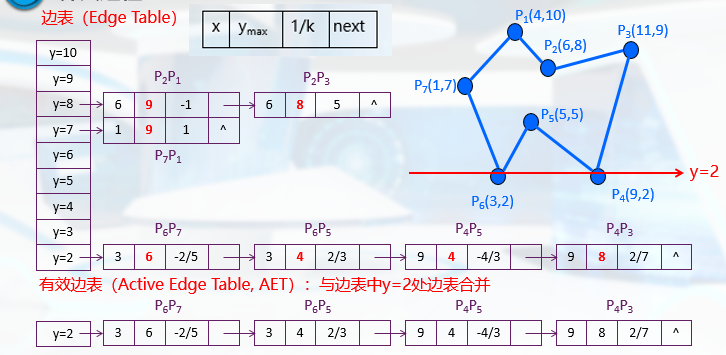


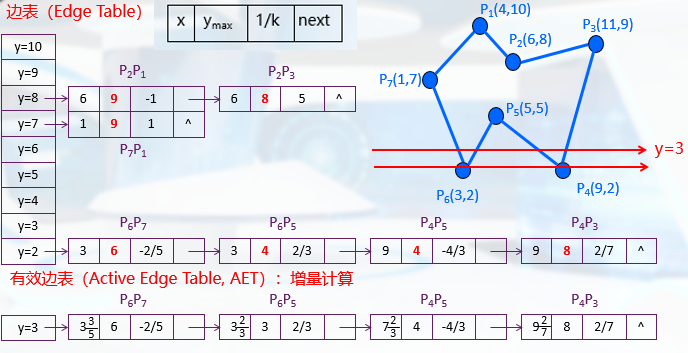
**多边形：**

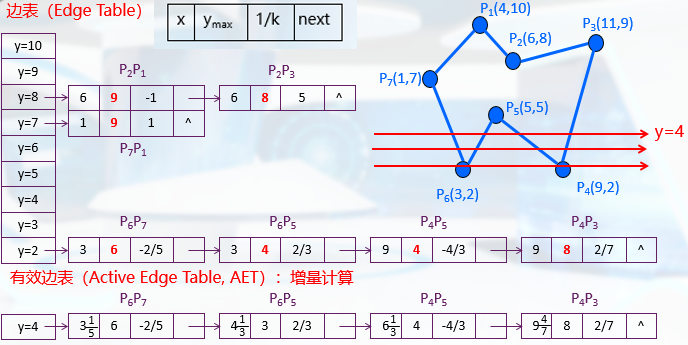
**1X扫描**

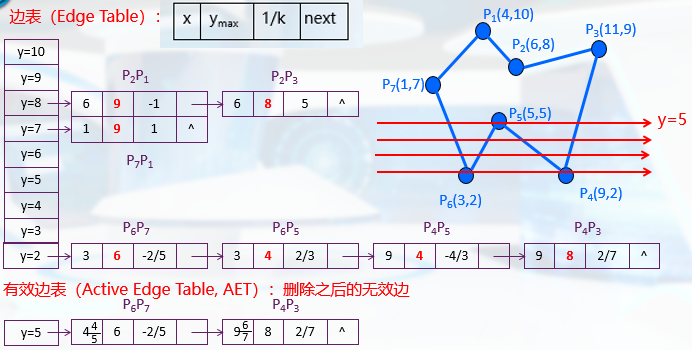
****











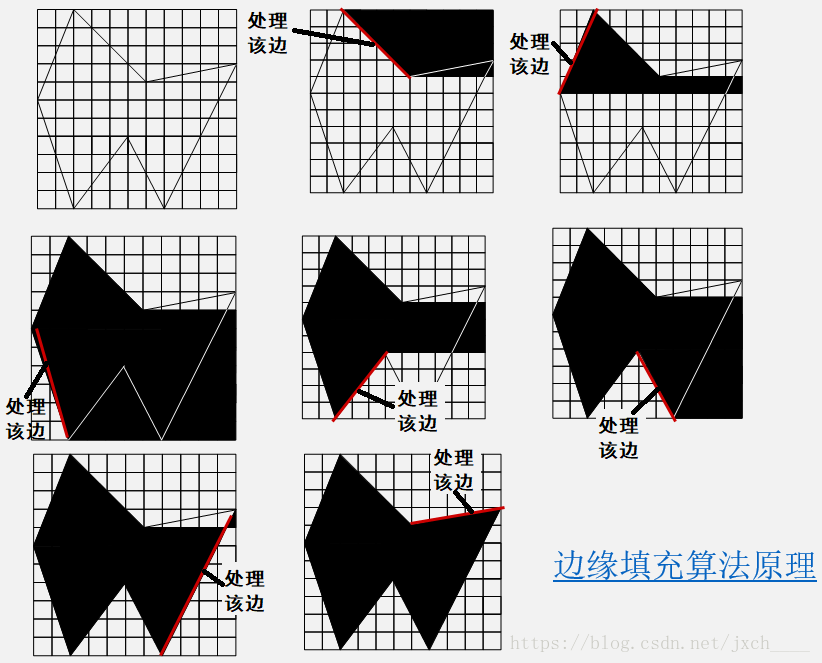
**改进的出发点：**

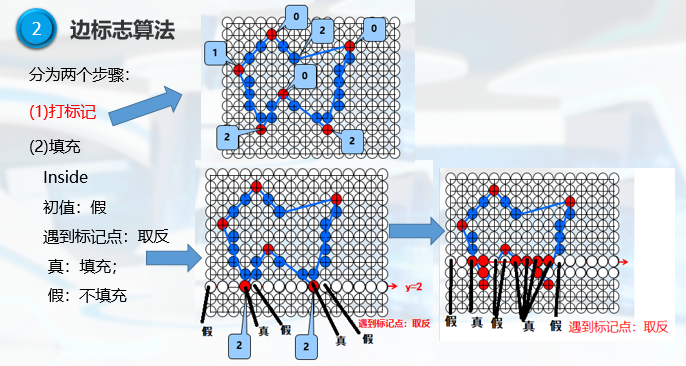
**(1)对于某一条扫描线，需要与所有的边求交吗？答：只与有效边求交！**

**(2)扫描线和直线在Y方向上都有连贯性，那么交点可以怎么求？答：增量计算！**

**(3)每次都需要排序吗？答：有新边加入时求交**

**边缘填充算法：边缘填充算法是先求出多边形的每条边与扫描线的交点，然后将交点右侧的所有像素颜色全部取为补色（或反色）。按任意顺序处理完多边形的所有边后，就完成了多边形的填充任务。边缘填充算法利用了图像处理中的求“补”或求“反”的概念，对于黑白图像，求补就是把RGB(1,1,1)（白色）的像素置为RGB(0,0,0)（黑色），反之亦然；对于彩色图像，求补就是将背景色置为填充色，反之亦然。求补的一条基本性质是一个像素求补两次就恢复为原色。如果多边形内部的像素被求补偶数次，保持原色，如果被求补奇数次，显示填充色**

****



**边标志算法对每个象素仅访问一次。与边缘填充算法和栅栏填充算法相比，避免了对帧缓存中大量元素的多次赋值，但仍然需逐条扫描线地对帧缓存中的元素进行搜索和比较。**

**当用软件实现本算法时，速度与Y向连贯性算法相当，但本算法用硬件实现后速度会有很大提高**

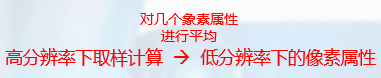
**走样：**

**走样现象：1）阶梯状（锯齿感）2）微小物体的忽略或时隐时现**

**走样的本质：用离散量表示连续量引起的失真**

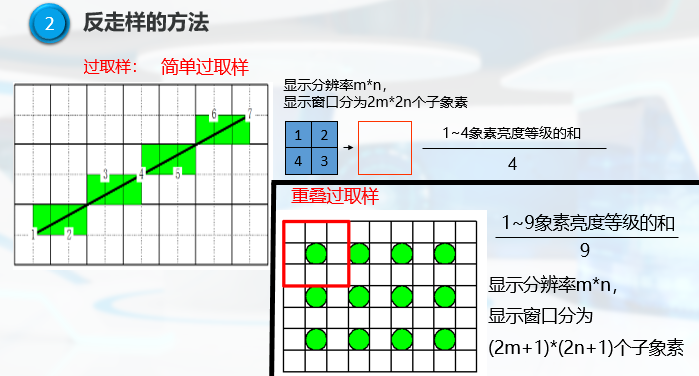
**反走样的本质：用于减少或消除这种效果的技术 最简单的方法：提高分辨率**

**一般方法：**



**过取样（supersampling），或后滤波 区域取样（area sampling），或前滤波**

**过采样：**



**区域取样：**

**面积计算：(1)将屏幕象素分割成n个更小的子象素，(2)计算中心落在直线段内的子象素的个数m， (3)m/n为线段与象素相交区域面积的近似值**

**简单的区域取样特点：1）直线段对一个象素亮度的贡献与两者重叠区域的面积成正比**

**2）相同面积的重叠区域对象素的贡献相同**

**加权区域取样原理：假想一个连续的加权曲面（或过滤函数）覆盖象素。当直线条经过该象素时，该象素的灰度值是在二者重叠区域上对滤波器（过滤函数）进行积分的积分值。**

**加权区域取样特点：1）接近理想直线的象素将被分配更多的灰度值；**

**2）相邻两个象素的滤波器相交，有利于缩小直线条上相邻象素的灰度差。**

1. 色彩

**人们色彩感觉形成的四大要素：光源、彩色物体、眼睛、大脑**

颜色视觉原理 ：1）Young -Helmholtz的三原色学说（在人眼视网膜上存在感受红、绿、蓝色的锥状细胞，分别对红、绿、蓝三种光最敏感；一切颜色特性都由这些锥状体的响应量比例来表示；三者共同作用，使人产生不同的颜色感觉。）

2）Hering的对立颜色学说（四色学说）（由视觉现象总结出的规律；在人眼视网膜上存在三组对立视素：红-绿、黄-蓝、白-黑；所有的颜色特性都由这三组对立颜色的响应量比例来表示，绿和黄—蓝两组对立颜色响应值的组合决定其色调，黑—白响应值决定其亮度）

颜色辨别：人眼能分辨100多种颜色；对490nm的青绿色和590nm的橙黄色的光波变化最敏感；在可见光谱的两端最不敏感

颜色对比：相邻区域的不同颜色相互影响，是由是视觉暂留效应引起的（同时对比(**对比区域的颜色不同，效果不同**)、继时对比(**当看了一种色彩再看一种色彩时，会把前一种色彩的补色加到后一种色彩上**)、边界对比(**马赫带效应,** **亮度发生跃变时，可看到有一种边缘增强的感觉;明度高的越高，明度低的越低；越接近边界线，影响越强烈**) 、色相对比、明度对比(**因明度差异形成的对比，同一明度的色彩，在白底上显得暗，而在黑色背景上却显得更亮)** 、纯度对比(在纯度低的背景色上的会显得鲜艳一些在纯度高的背景色上会显得灰浊)）

颜色错觉: 同化现象、色彩的醒目性、色彩的进退、色彩的冷暖、色彩的胀缩、色彩的轻重、色彩的情绪

