复习

目录

- 向量表示和变换
  - 齐次坐标表示
  - 仿射变换矩阵
  - 投影变换
- 图形绘制与渲染
  - 光栅化绘制
    - 直线生成算法
    - 三角形填充
    - 剪裁
    - 隐藏面消除
  - 光照模型
  - 明暗着色
  - 阴影计算
  - 纹理映射
    - 双线性插值
    - 纹理反走样
- 几何表示与处理
  - 显式VS.隐式表示
  - 网格细分算法
  - 网格简化算法
- 曲线曲面理论
  - 样条函数
  - 分形

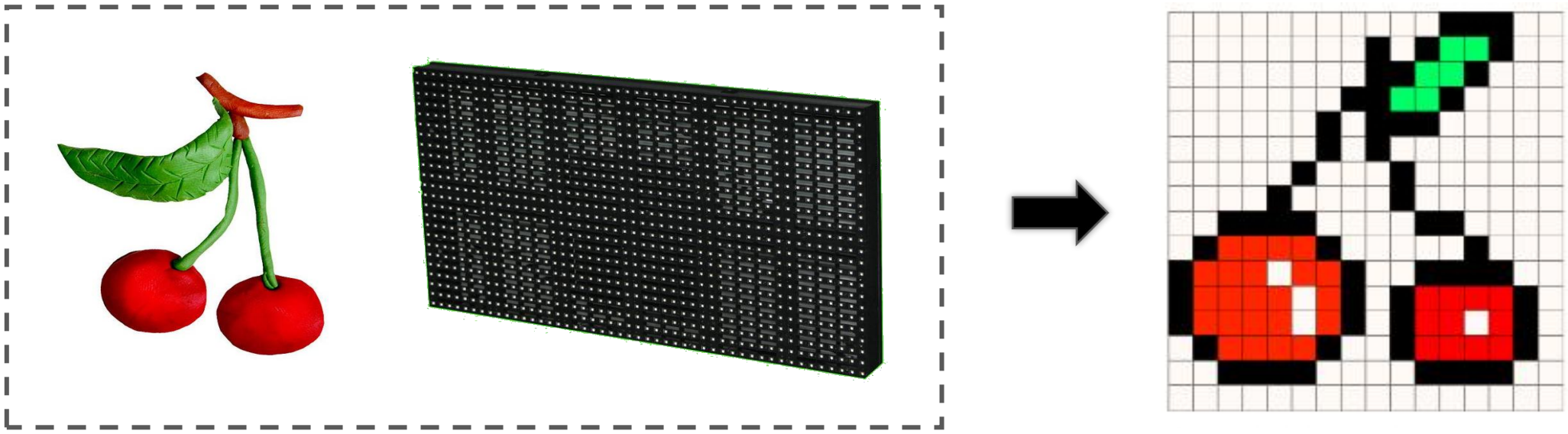
目录

- 向量表示和变换
  - 齐次坐标表示
  - 仿射变换矩阵
  - 投影变换
- 图形绘制与渲染
  - 光栅化绘制
    - 直线生成算法
    - 三角形填充
    - 剪裁
    - 隐藏面消除
    - 光照模型
    - 明暗着色
    - 阴影计算
  - 纹理映射
    - 双线性插值
    - 纹理反走样
- 几何表示与处理
  - 显式vs.隐式表示
  - 网格细分算法
  - 网格简化算法
- 曲线曲面理论
  - 样条函数
  - 分形

4.1 光栅化绘制

什么是光栅化绘制?

- 光栅化的目的是将图形绘制在光栅显示器屏幕上。
- 图形表示：用代数曲面或者三维网格表示（“连续表示”）
- 光栅显示器：光栅扫描式图形显示器的简称，是画点设备，可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（“离散表示”）



4.1.1 直线生成算法

DDA算法：用参数方程控制像素绘制

- 利用Δt控制步长来计算线段上的点

$$\begin{cases} \boldsymbol{x = x_s + \Delta x \cdot \Delta t} \\ \boldsymbol{y = y_s + \Delta y \cdot \Delta t} \end{cases}$$

- 若直线斜率小于1$(\Delta x \geq \Delta y \geq 0)$，$x$方向每次增加1，$\Delta t = 1/\Delta x$
- 若直线斜率大于1$(\Delta y \geq \Delta x \geq 0)$，$y$方向每次增加1，$\Delta t = 1/\Delta y$
- 所以

$$\Delta t = min\{1/\Delta x, 1/\Delta y\}$$

缺点：需要除法计算Δt；每次计算需两次加法和取整

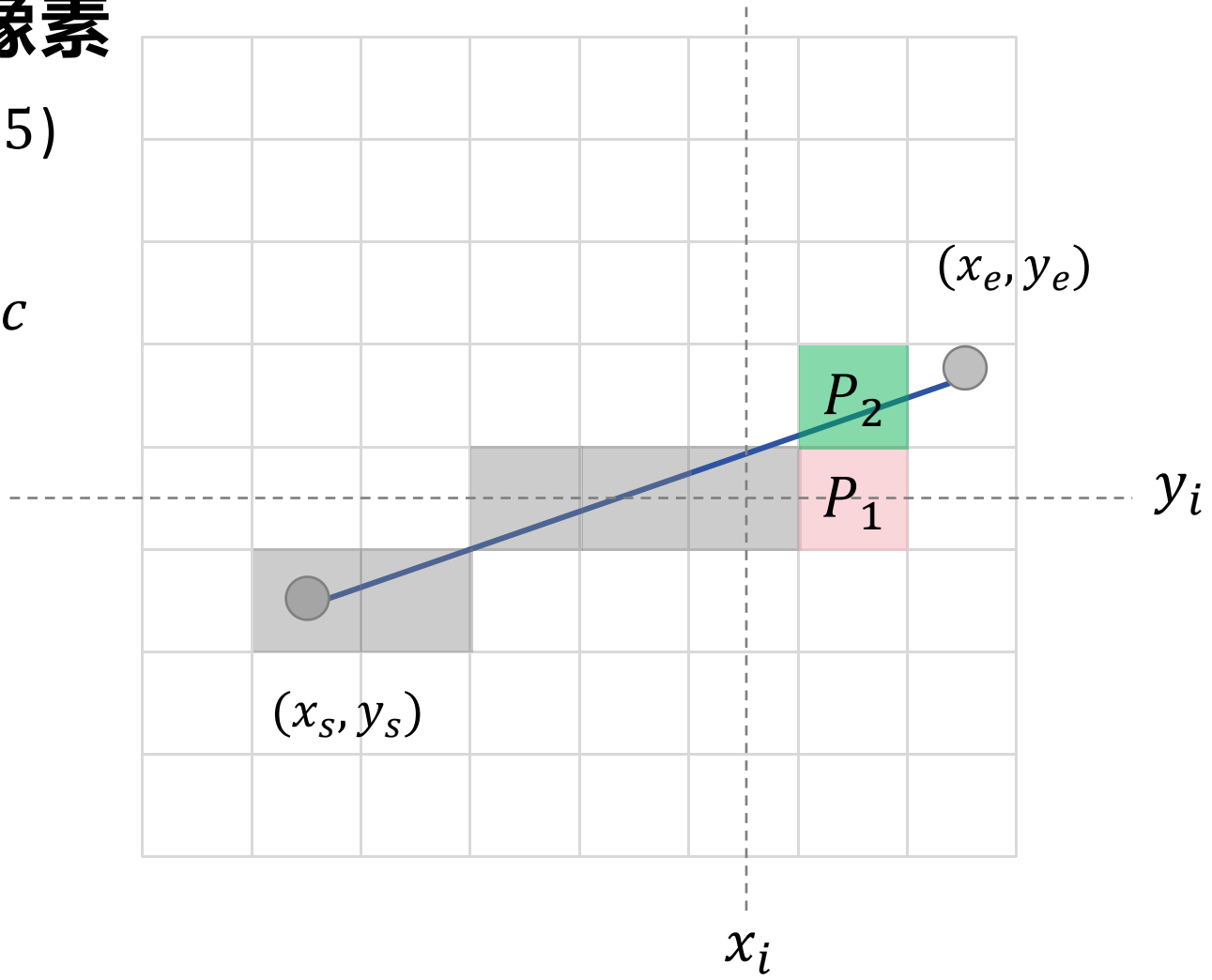
4.1.1 直线生成算法

正负法：根据正负判断下一个像素

- 找出P_1P_2中点坐标$(x_p + 1, y_p + 0.5)$
- 利用直线方程计算正负：

$F(M) = a(x_p + 1) + b(y_p + 0.5) + c$

- 根据正负判断下次要绘制的像素



4.1.1 直线生成算法

Bresenham算法：

与DDA算法类似，但不再采用四舍五入的办法；与中点画线法类似，由符号来决定下一个像素点的位置。

若K在0—1之间，在x方向上每增加一个单位长度，y方向是否增加一个单位长度是根据计算误差项的符号P确定的。

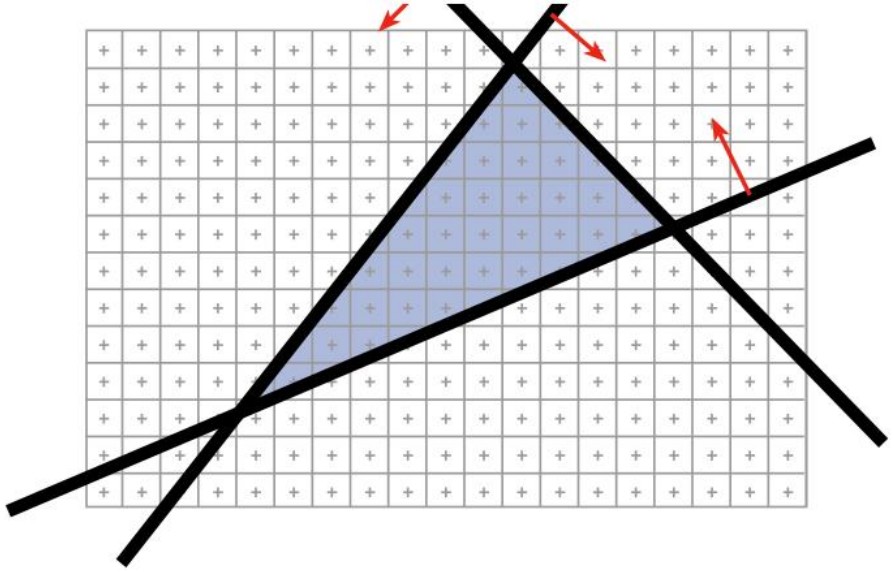
增量算法：在一个迭代算法中，如果每走一步x，y值是用前一步的值加上一个增量来获得，则称为增量算法。

4.1.2 三角形填充

区域填充：即给出一个区域的边界，要求对边界范围内的所有像素单元赋予指定的颜色代码。区域填充中最常见的是多边形填色。

三角形填充：

- 三角形三条边投影到图像平面，形成三条线段。
- 同时位于这三条线段“内侧”的区域即为三角形填充区域。



$$E_i(x,y) = a_i x + b_i y + c_i$$

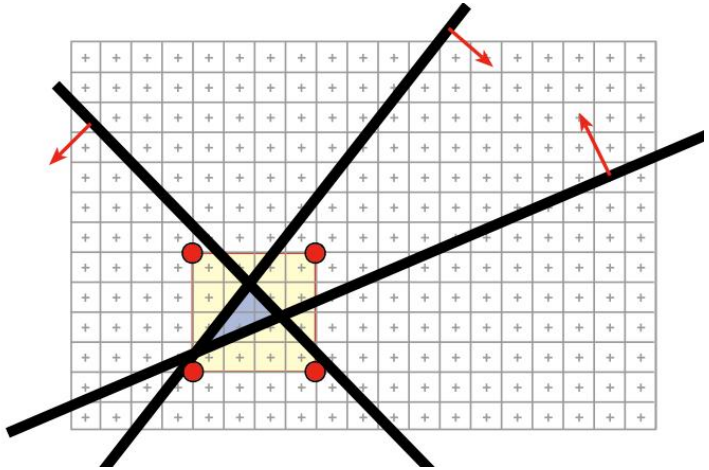
$$(x,y) \text{ within triangle} \Leftrightarrow E_i(x,y) \geq 0, \forall i = 1,2,3$$

4.1.2 三角形填充

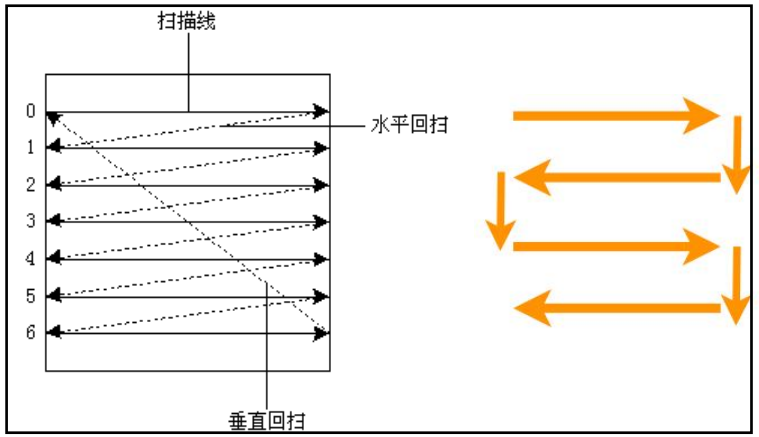
三角形填充算法：

- 对每个像素(x,y)，根据边所在的直线方程判断内外

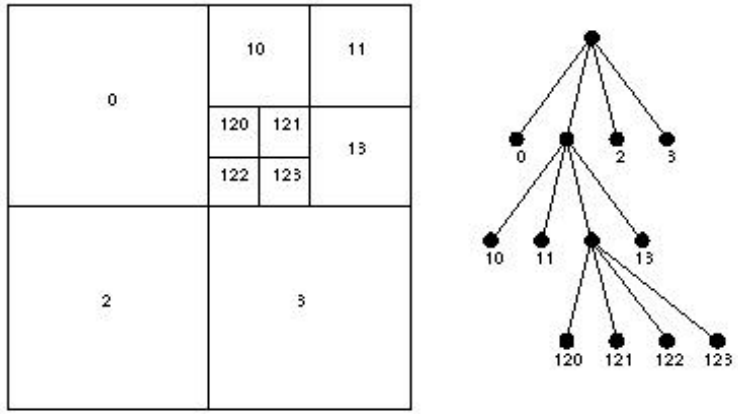
加速方法：



包围盒加速



扫描线加速



四叉树划分

4.1.2 三角形填充

三角形填充算法：

- **对每个像素(x,y)，根据边所在的直线方程判断内外**

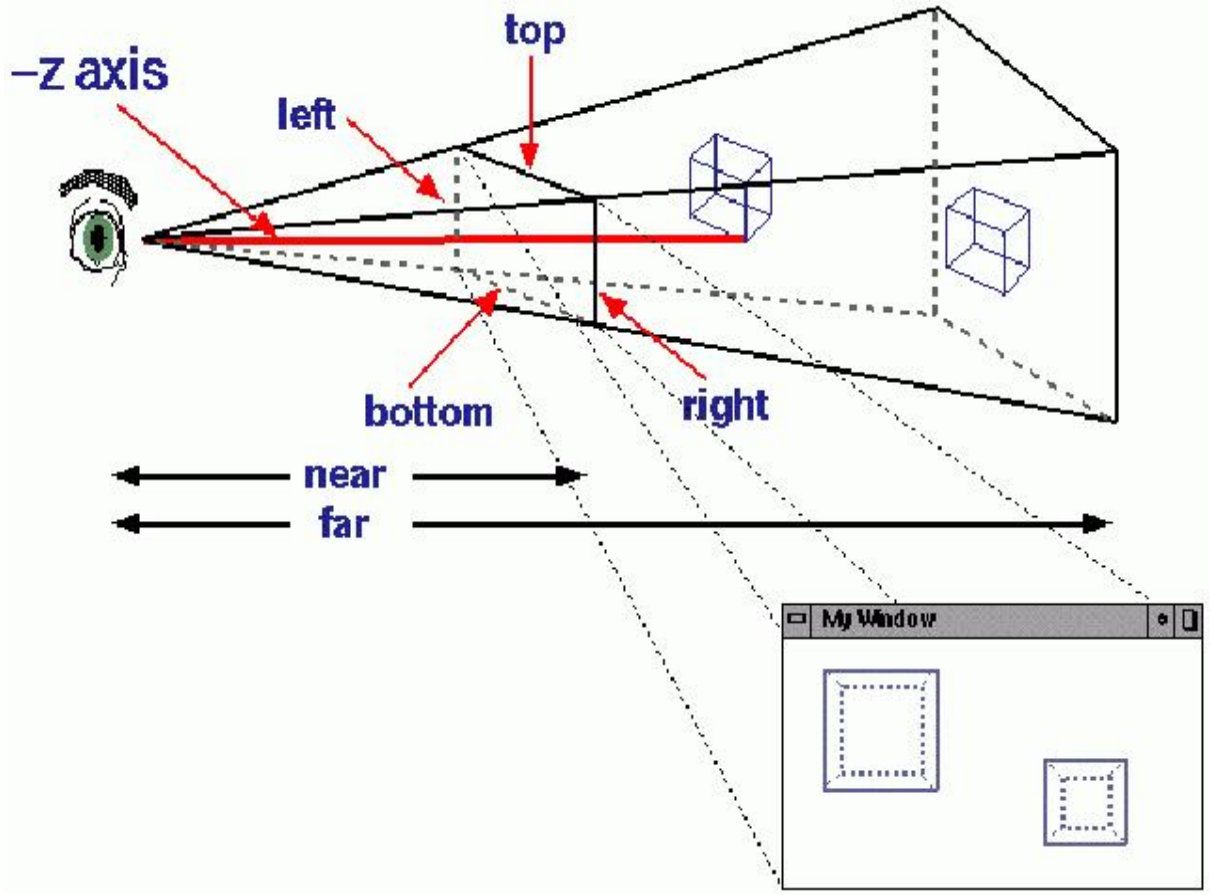
加速方法（为什么？）

- **包围盒加速**：只检测包围盒内部像素
- **扫描线加速**：沿扫描线顺序检测像素
- **四叉树划分**：根据所在节点索引检测像素

4.1.3 剪裁

视见体(View Frustum)：世界空间中将被裁剪出来并投影到视图平面的那一部分定出边界。

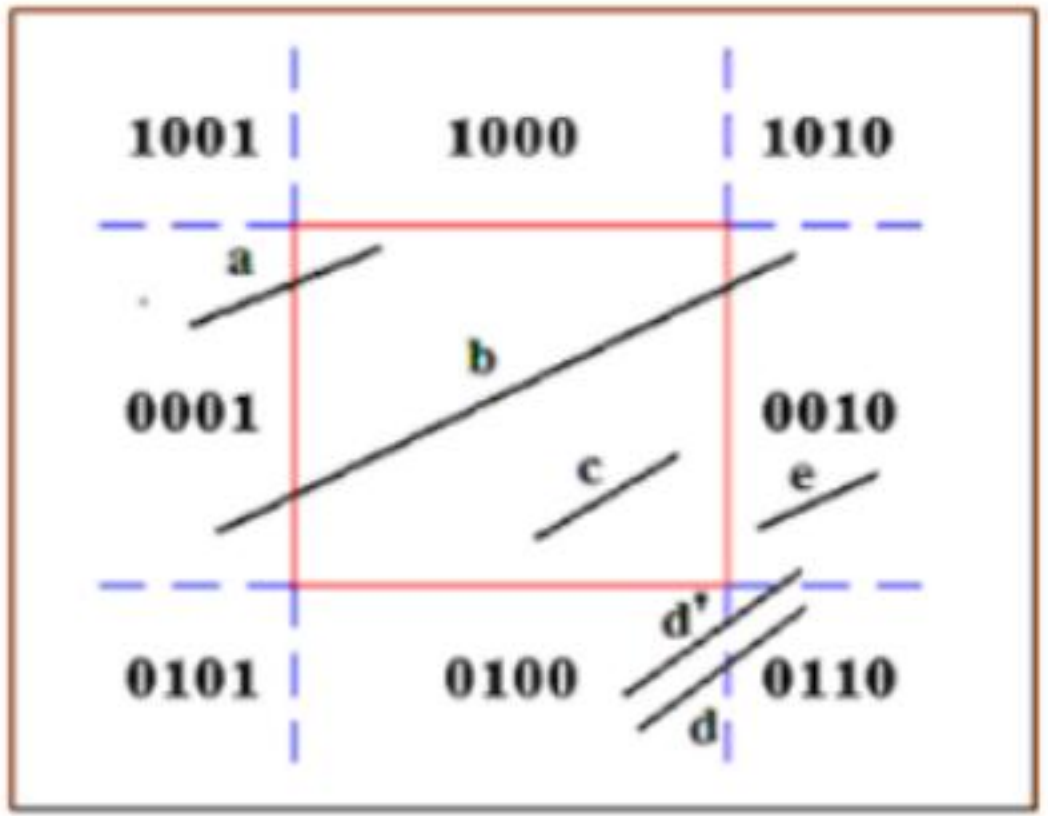
剪裁：需要确定图形中哪些部分落在显示区之内，哪些落在显示区之外，一般只显示落在显示区内部的图形，这个选择过程称为裁剪。



4.1.3 剪裁

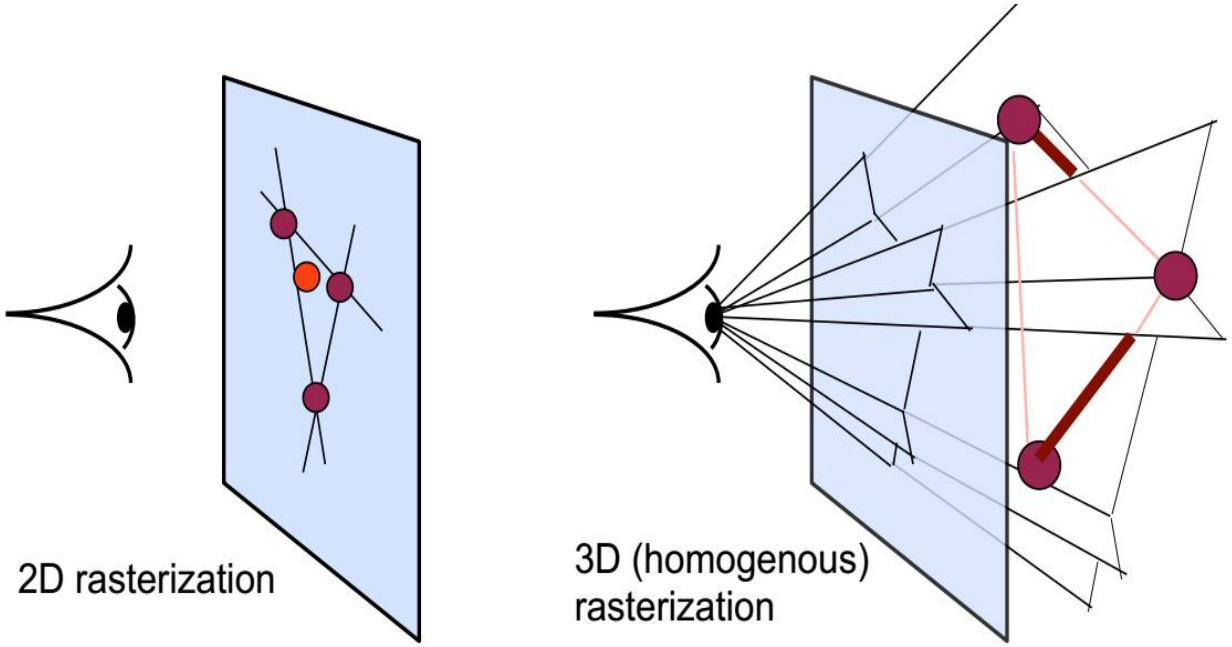
Sutherland-Cohen Algorithm算法：用于直线段剪裁

- 计算每个顶点所在区域
  - 两个顶点都是0000
  - 两个顶点的AND操作获得0000
  - 其他
- 判断线段是否在区域内部
- 显示线段在区域内部的部分

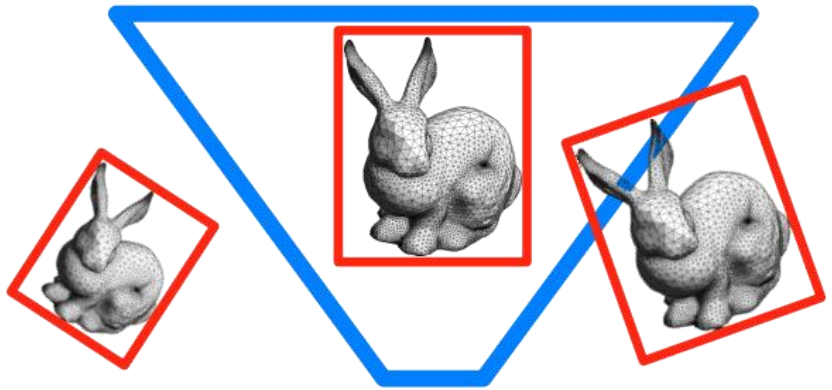


4.1.3 剪裁

- **可见体剪裁**：对每个三角形，判断其是否在可见体内部



简化为平面中的像素内外判断问题

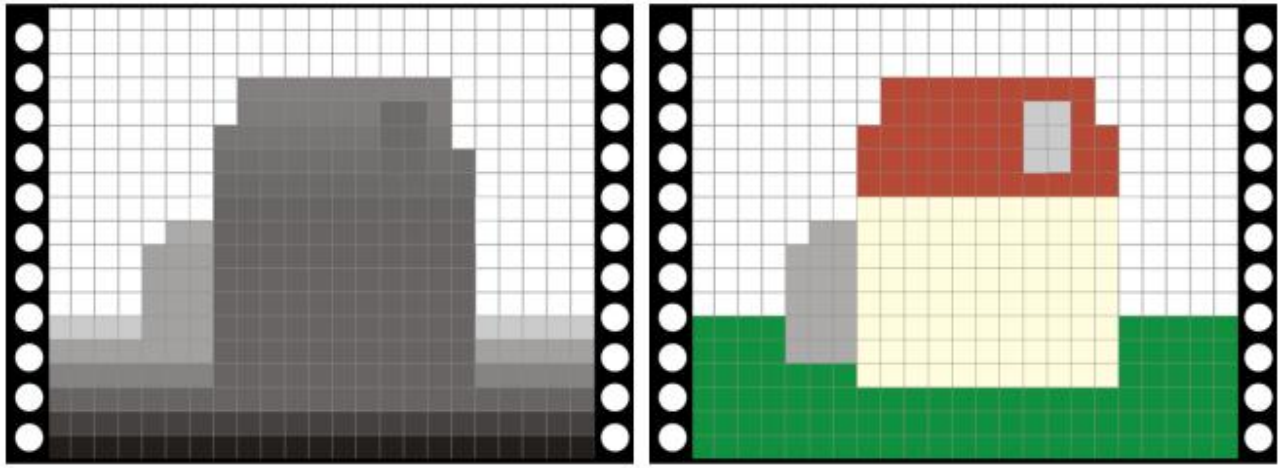


同样可以进行包围盒加速

4.1.4 隐藏面消除

隐藏面消除：又叫消隐算法。即相对于观察者，确定场景中哪些物体或部分可见，哪些物体不可见。本质是将待显示的物体沿观察方向排序。

Z-buffer：深度缓存，在z缓冲区中存储对应像素中可见点的z值。消隐过程中，计算投影到当前像素上各表面采样点的深度值，并与z-buffer中的对应存储值进行比较，只存储离视点较近的z值。



4.1.4 隐藏面消除

z-buffer算法（z越大离视点越近）：

- 将帧缓存中各像素的颜色置为背景颜色
- 将z-buffer中各像素的z值置成最小值
- 以任意顺序扫描各多边形：
  - 对多边形中每一采样点，计算其深度值z(x,y)
  - 比较z(x,y)与z-buffer中当前值zbuffer(x,y)
    - 如果z(x,y)>z-buffer(x,y)，计算该采样点(x,y)处表面的光亮度值属性，写入帧缓存相应像素；同时更新z-buffer， 取zbuffer(x,y)=z(x,y)

4.1.5 光照模型

局部光照模型

$$I = I_{ambient} + I_{diffuse} + I_{specular}$$

- **环境光：** $I_{ambient} = k_a I_{pa}$

通常简化为在各个方向都有均匀的光强度，I_{pa}为环境光亮度，k_a为物体表面的环境光反射系数$(0 \le k_a \le 1)$

- **漫反射：** $I_{diffuse} = k_d I_{pd} \cos i$

表示表面某一点向空间各方向的均匀反射，各方向反射强度相同。I_{pd}为光源垂直入射时反射光的光亮度，i为光源入射角，k_d为漫反射系数

- **镜面反射：** $I_{specular} = k_s I_{ps} cos^p \theta$

镜面反射为朝一定方向的反射光。I_{ps}为入射光的光亮度，θ为镜面反射方向和视线方向的夹角，p为镜面反射光的会聚指数（与物体表面光滑程度有关）

4.1.5 光照模型

局部光照模型

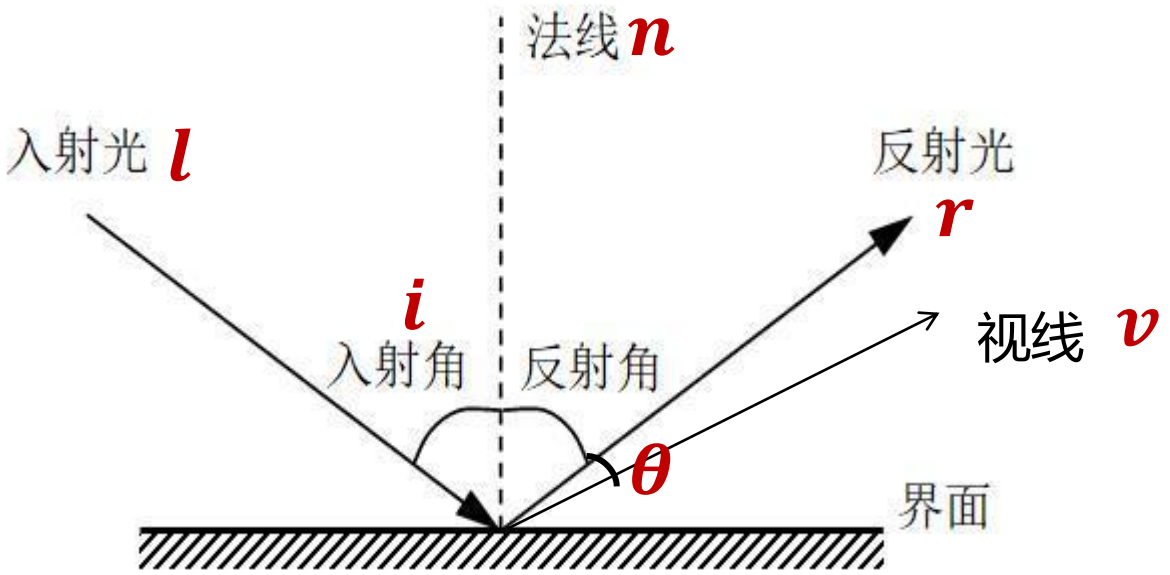
$$I = I_{ambient} + I_{diffuse} + I_{specular}$$

$\cos i = l \cdot n$

$\cos \theta = r \cdot v$

Phong Model

- **环境光：** $I_{ambient} = k_a I_{pa}$
- **漫反射：** $I_{diffuse} = k_d I_{pd} \cos i$
- **镜面反射：** $I_{specular} = k_s I_{ps} cos^p \theta$



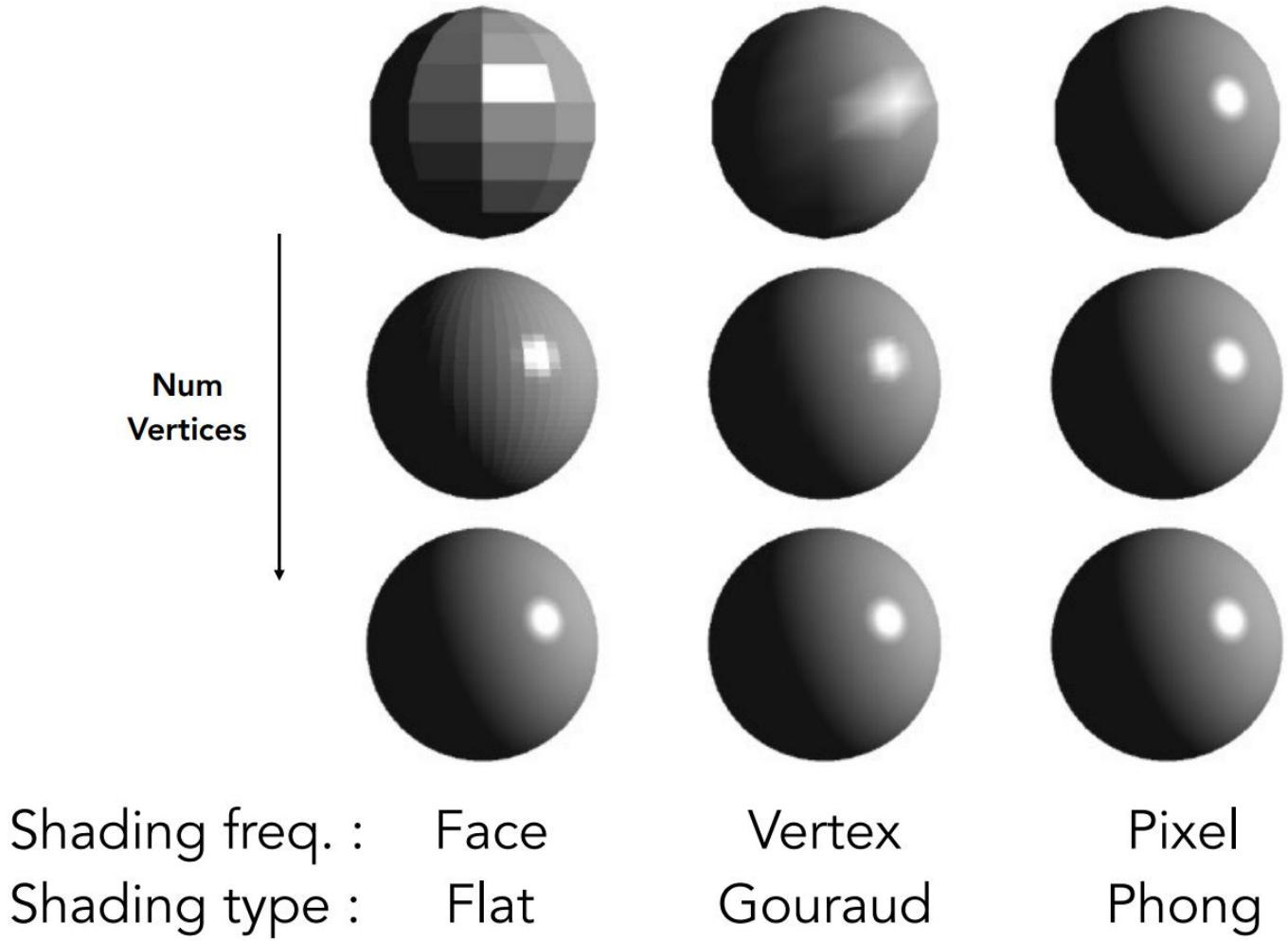
4.1.5 光照模型

全局光照 vs. 局部光照

- **局部光照模型**：仅考虑了从光源直接发出的光线对物体表面光亮度的贡献，而没有考虑光线在物体之间的相互反射。

- **全局光照模型**：从某一观察方向所观察到的物体表面某点光亮度来自3个方面：
  - 由光源直接照射引起的反射光亮度
  - 环境光在表面产生的镜面反射光
  - 沿规则透射方向入射的环境光

4.1.6 明暗着色

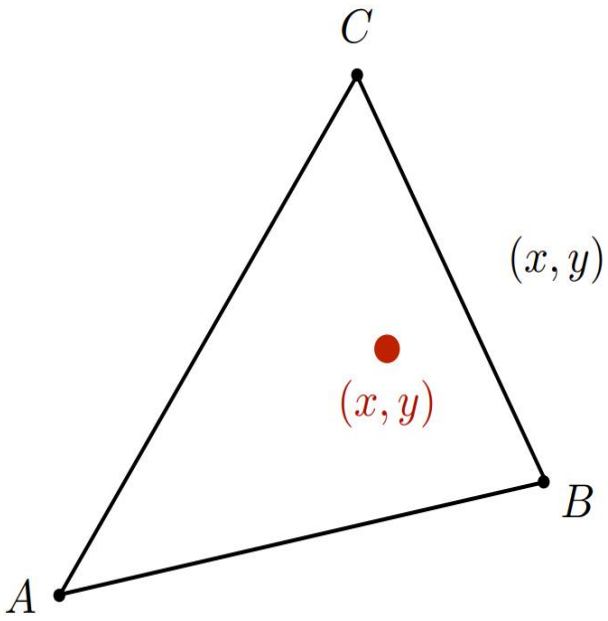

Num
Vertices
Shading freq. : Face Vertex Pixel
Shading type : Flat Gouraud Phong

4.1.6 明暗着色

- **Flat Shading**：根据局部光照模型，按每个三角形的法向计算各处光亮度
- **Gouraud Shading**：
  - 为每个顶点计算法向（邻接面法向的平均值）
  - 根据局部光照模型计算顶点光亮度
  - 顶点线性插值得到各处光亮度
- **Phong Shading**：
  - 为每个顶点计算法向（邻接面法向的平均值）
  - 顶点线性插值得到各处法向值
  - 根据局部光照模型计算各处光亮度

4.1.6 明暗着色

重心坐标插值

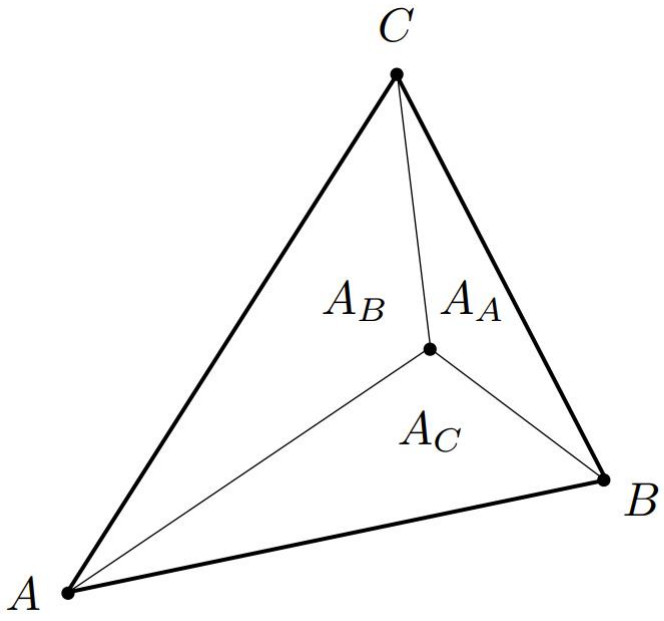


$$(x, y) = \alpha A + \beta B + \gamma C$$

$$\alpha + \beta + \gamma = 1$$

Inside the triangle if all three coordinates are non-negative

重心坐标的定义



$$\alpha = \frac{A_A}{A_A + A_B + A_C}$$

$$\beta = \frac{A_B}{A_A + A_B + A_C}$$

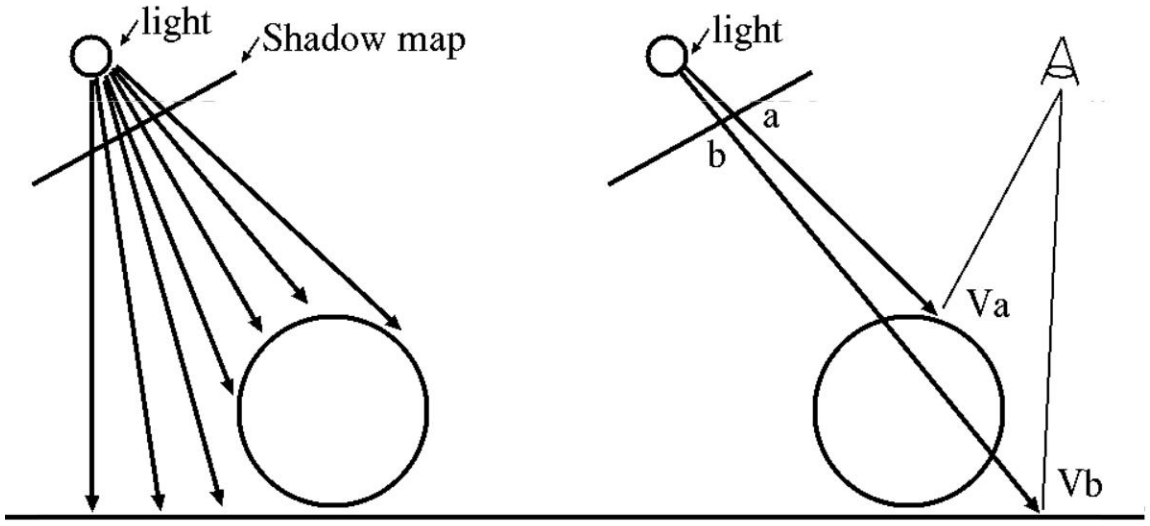
$$\gamma = \frac{A_C}{A_A + A_B + A_C}$$

重心坐标的计算

4.1.7 阴影计算

阴影缓冲方法

- 为每个光源计算一个阴影缓冲s_buffer
    - 将摄像机摆在光源位置，在该处生成一个深度缓冲，称为阴影缓冲
- 在视点位置计算深度缓冲z_buffer
- 对每个像素判断是否处于阴影中
    - 比较depth(p)和s_buffer(p)



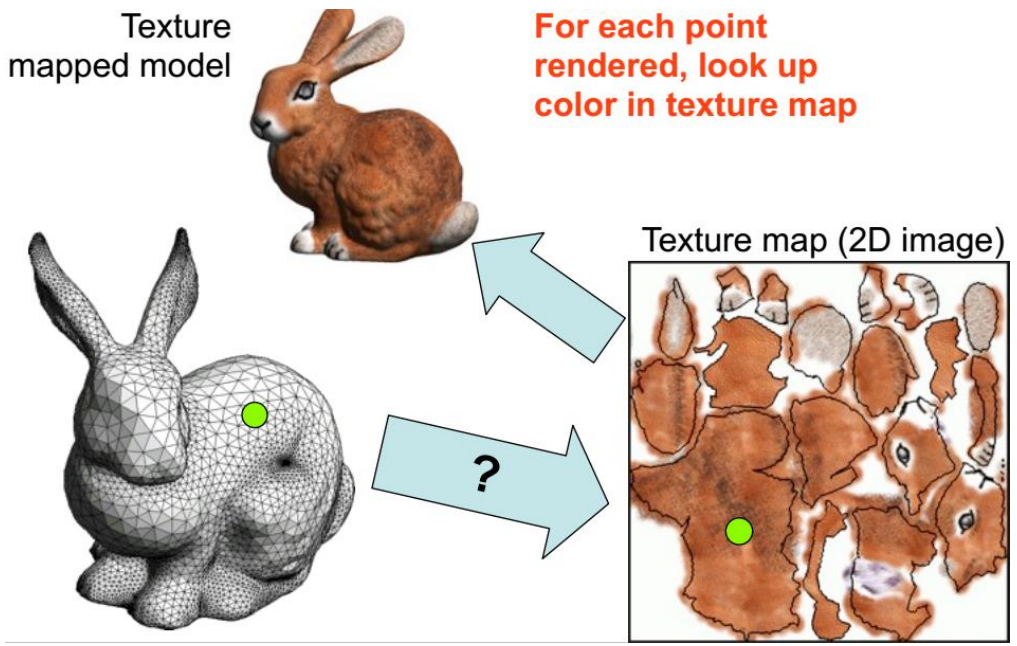
Checking whether Va,Vb is closer to the light

4.2 纹理映射

纹理映射过程

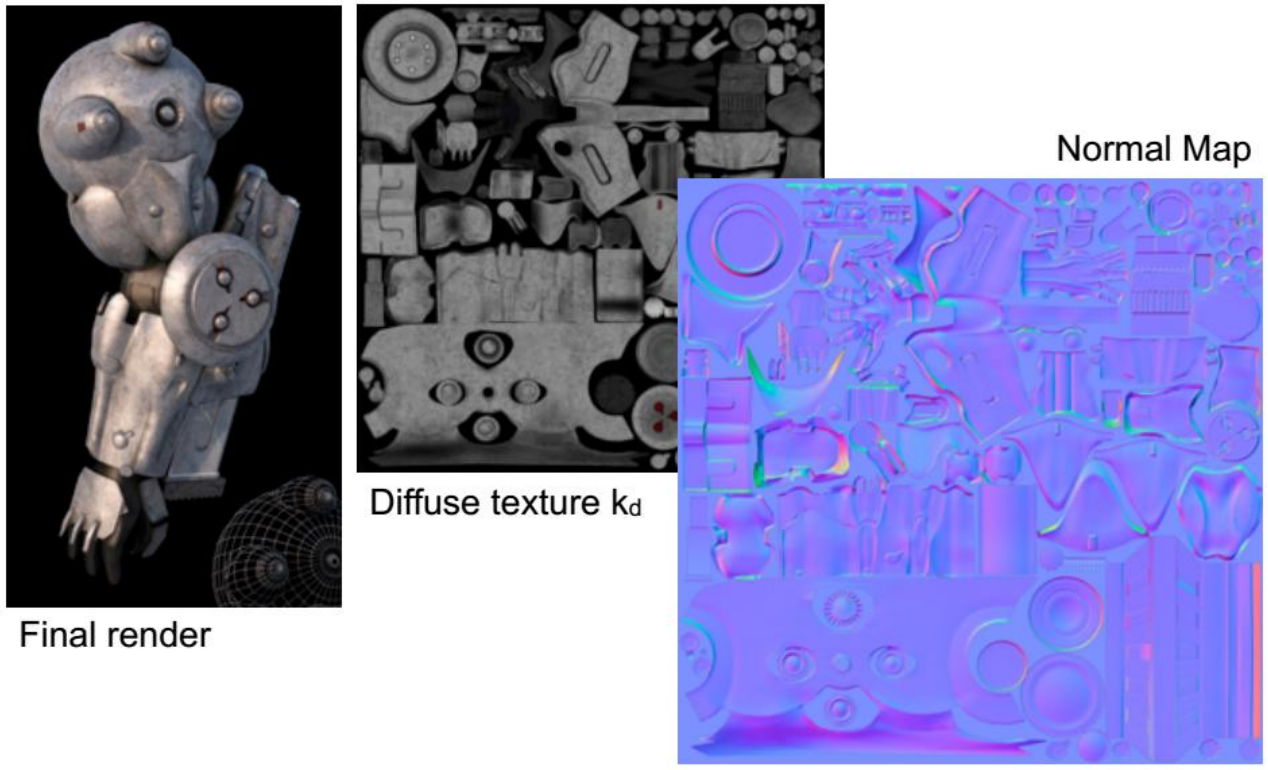
采用景物表面的参数化表示来确立纹理映射坐标

- 设景物表面的参数化表示为f(u,v)，纹理图像为T(s,t)
- 只需建立景物表面参数空间(u,v)和纹理图像参数空间(s,t)之间的一一对应关系，
- 对表面任意一点(u_p,v_p)，找出对应的(s_p,t_p)，将纹理图像在该处的属性值映射到该点处。



4.2 纹理映射

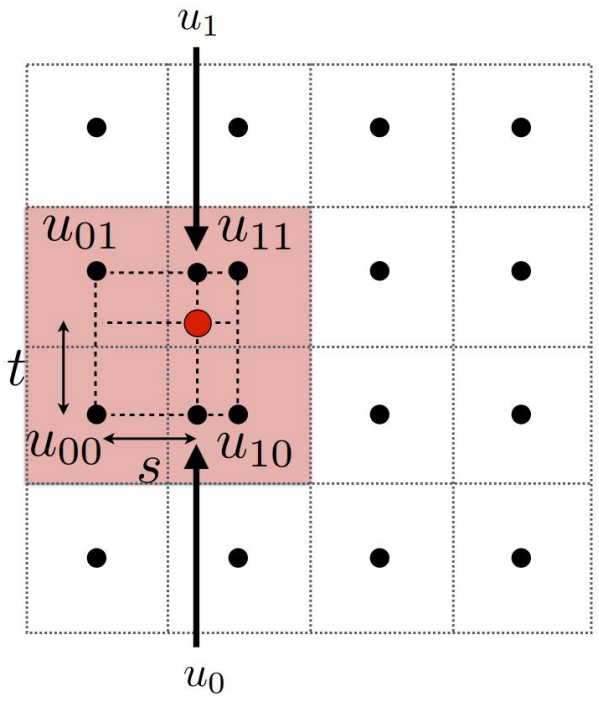
- **颜色纹理**描述了光滑表面上各点处的颜色分布。
- **法向纹理**每个点描述了该处的法向方向。



4.2.1 双线性插值

双线性插值

- 计算四个顶点的属性值
- 插值计算u0和u1处的属性值
- 插值计算目标点处的属性值



Linear interpolation (1D)

$$\text{lerp}(x, v_0, v_1) = v_0 + x(v_1 - v_0)$$

Two helper lerps

$$u_0 = \text{lerp}(s, u_{00}, u_{10})$$

$$u_1 = \text{lerp}(s, u_{01}, u_{11})$$

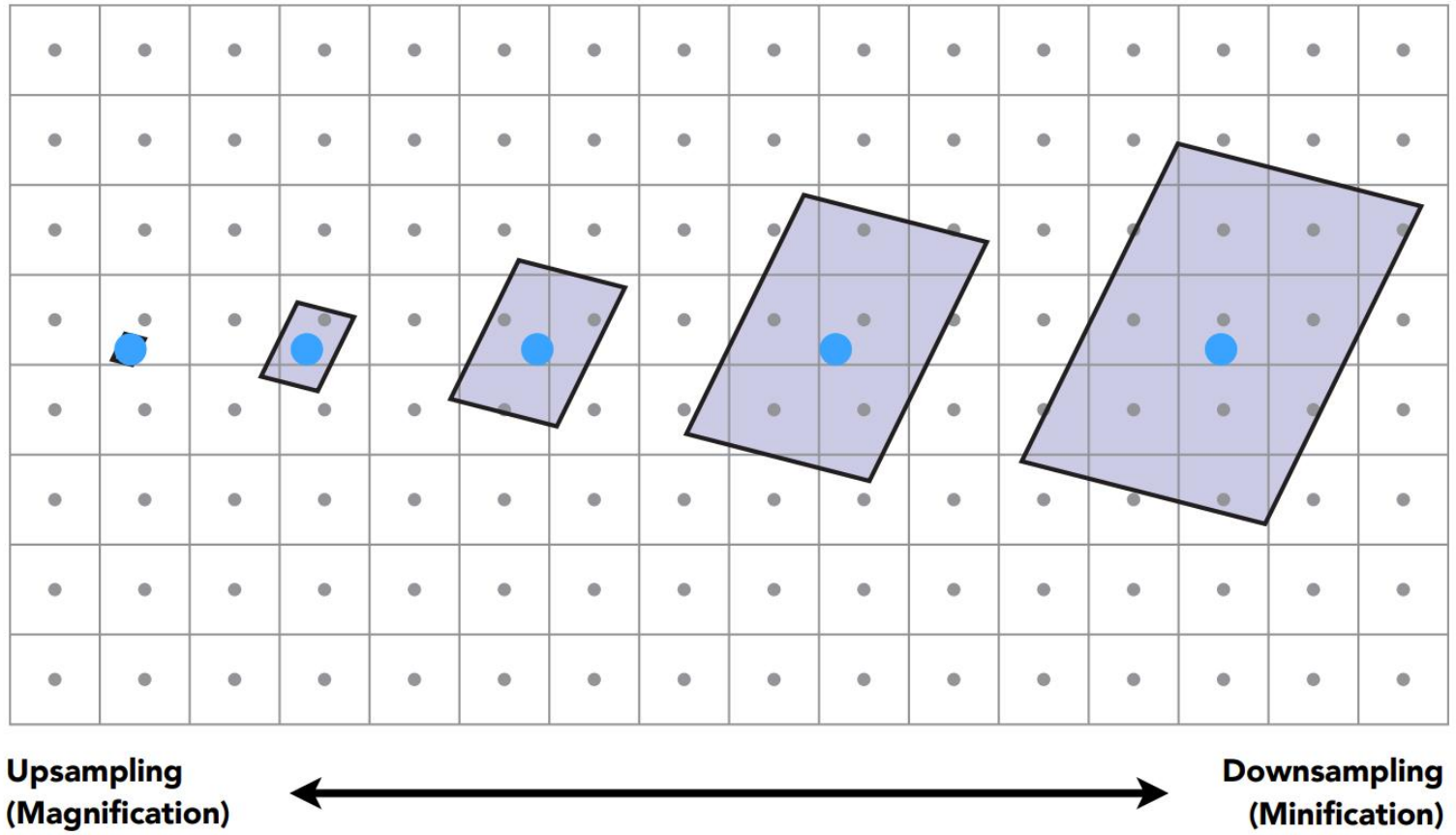
Final vertical lerp, to get result:

$$f(x, y) = \text{lerp}(t, u_0, u_1)$$

Nearest　　Bilinear　　Bicubic

4.2.2 纹理反走样

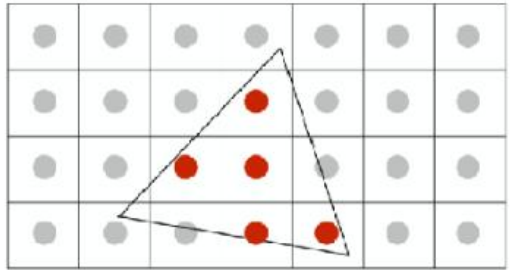
欠采样：很多纹理像素却只采样了其中一个像素的值



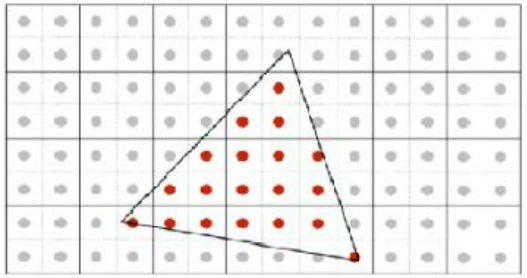
4.2.2 纹理反走样

SuperSampling反走样算法

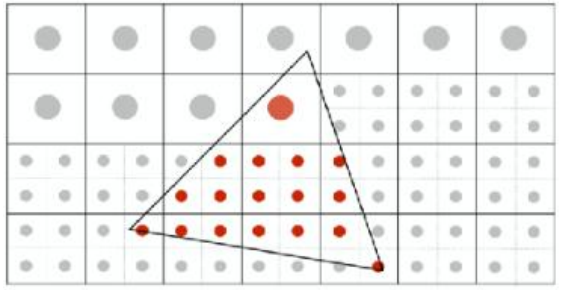
- 在一个像素内增加采样点
- 对采样点的值求平均



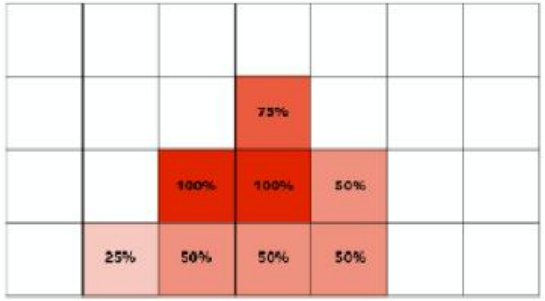
One sample per pixel



2x2 supersampling



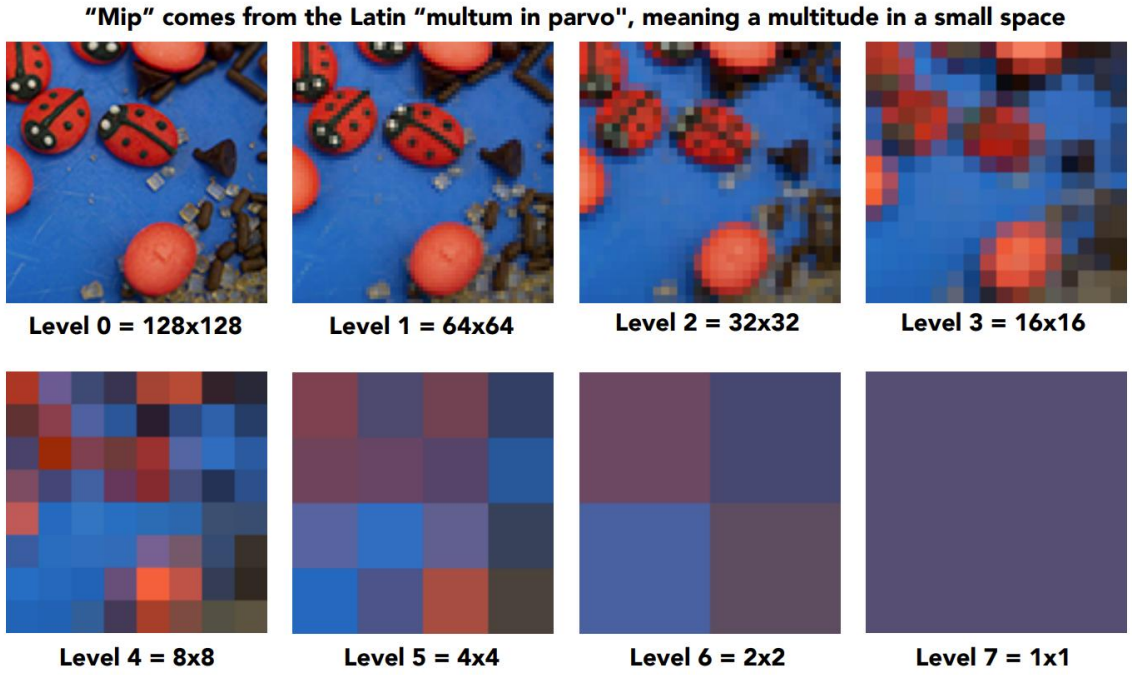
Averaging down



4.2.2 纹理反走样

MipMap算法

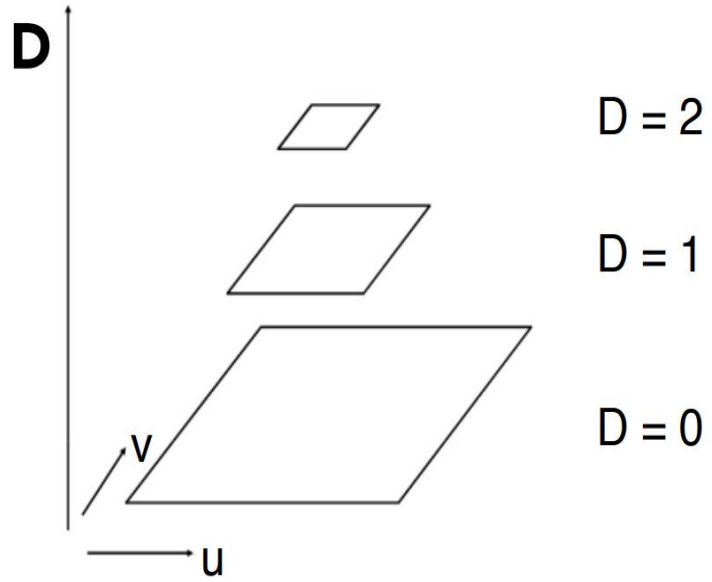
从原始纹理图像出发，首先生成一个分辨率为原始图像1/4的新纹理图像，其中每个像素值取原始图像中对应四个像素颜色值的平均值；然后类似地基于新纹理图像生成一个更低分辨率的、尺寸更小的纹理图像版本；这一过程一直持续到最后生成的纹理图像仅包含一个像素为止，从而得到一个由不同分辨率图像构成的纹理图像序列。



4.2.2 纹理反走样

如何计算MipMap纹理映射?

从纹理图像序列中找出其压缩率最接近当前纹理像素与屏幕像素比率的两个相邻纹理图像，算法在这两个纹理图像上查取当前屏幕像素映射点的纹理颜色值，并根据两个纹理图像对原始图像的压缩率在两个颜色值之间取加权平均值，作为要映射的颜色值。



"Mip hierarchy"

level = D

What is the storage overhead of a mipmap?

5.1 几何表示

显式 vs. 隐式表示

- 显式表示：几何坐标要么直接给出，要么通过参数映射给出。如点云、网格、参数曲面等。
  - 易于绘制
  - 难以确定点和几何的位置关系
- 隐式表示：不给出顶点或面的具体位置信息，而告诉点之间满足的关系。如距离函数、水平集等。
  - 很难直接判断它表示的是什么形状
  - 易于判断内外

5.1 几何表示

点云

- 由一系列坐标点表示
- 每个点可附加属性值，如颜色、法向等

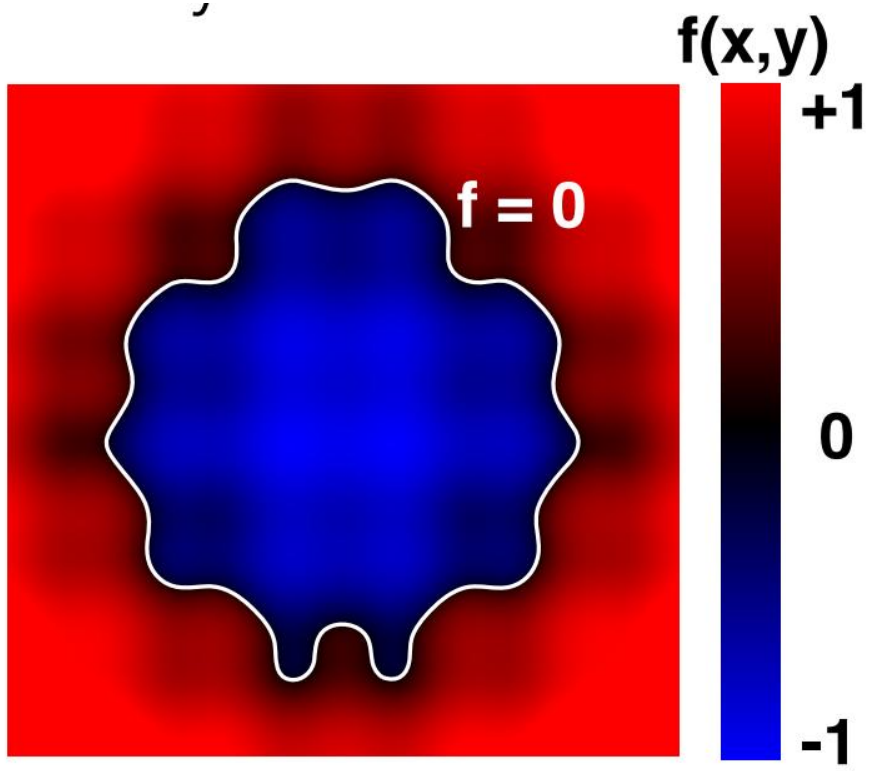
网格

- 两个数组分别表示顶点和多边形面
- 顶点数组记录每个顶点的坐标值和属性
- 多边形数组记录每个面的顶点索引值

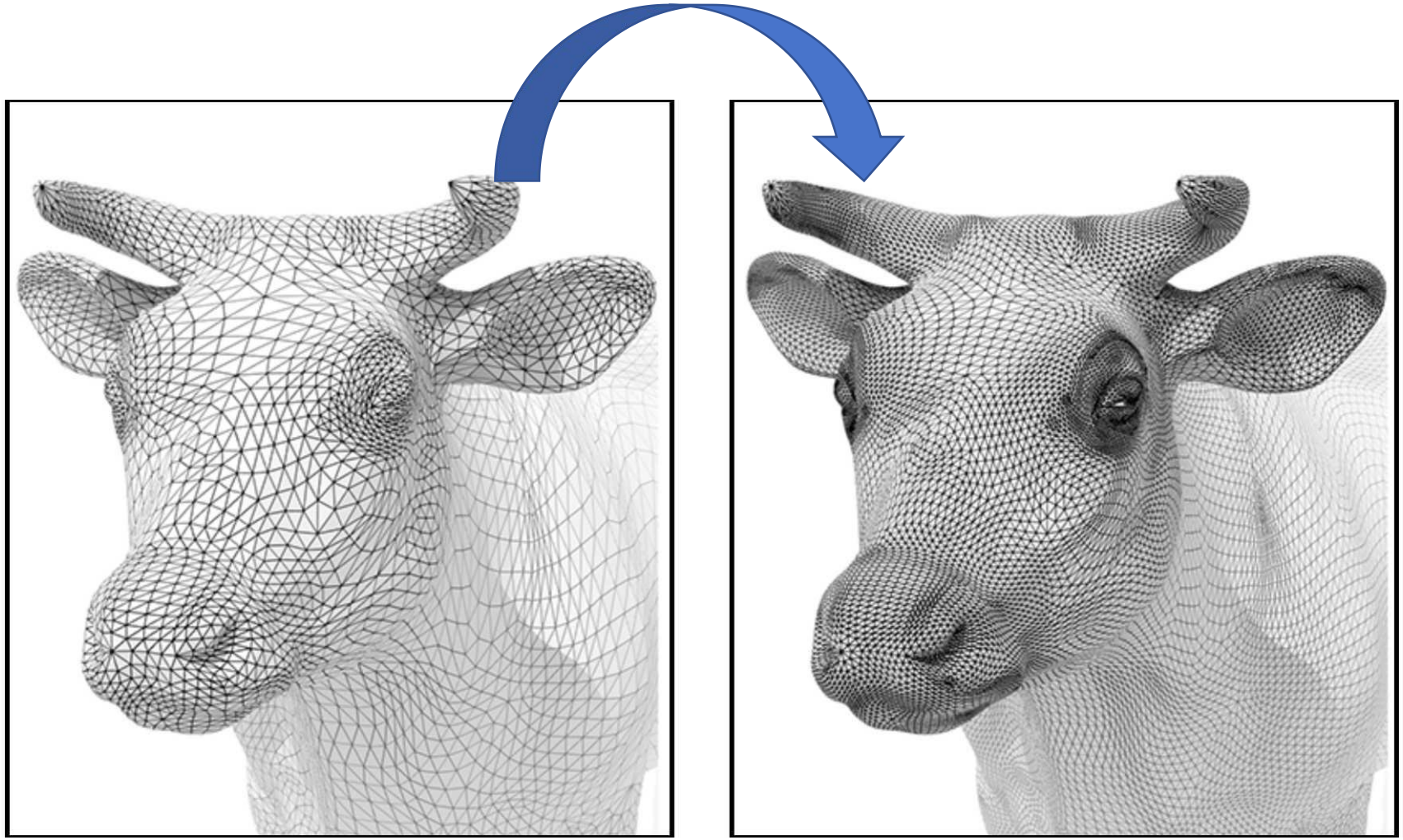
5.1 几何表示

距离场

- 定义距离函数f，对于空间中任意一点，其函数值为该点到几何表面的最近距离，符号表示该点在表面内外。



5.2 网格细分算法



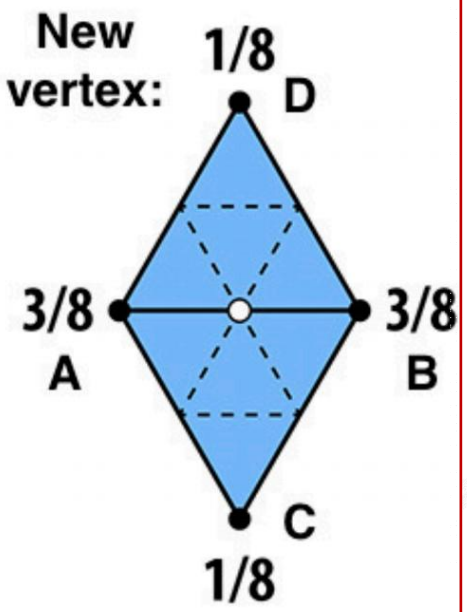
5.2 网格细分算法

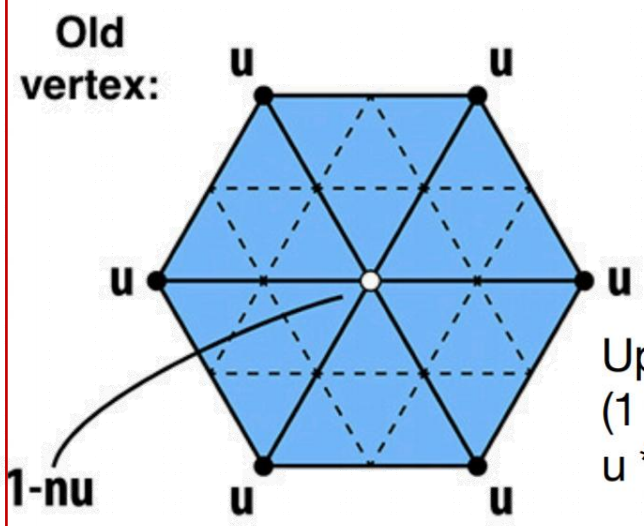
Loop三角网格细分算法

- 增加三角形的数量
  - 取三条边的中点连线就可以从一个三角形细分出4个三角形
- 调整顶点位置

Update new vertex

Update to:
3/8 * (A + B) + 1/8 * (C + D)





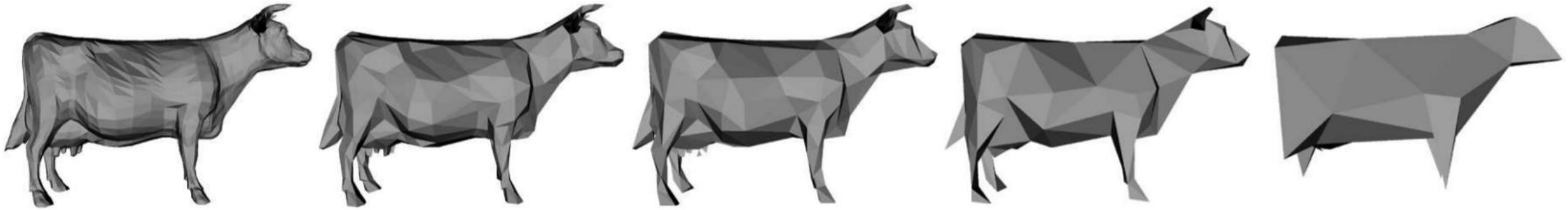
Update old vertex

Update to:
(1 - n*u) * original_position +
u * neighbor_position_sum

5.2 网格简化算法

QEM网格简化算法

- 初始化：预先计算各顶点对信息
- 每次选出简化操作代价最小的顶点对，进行单步简化操作
- 直到满足条件停止

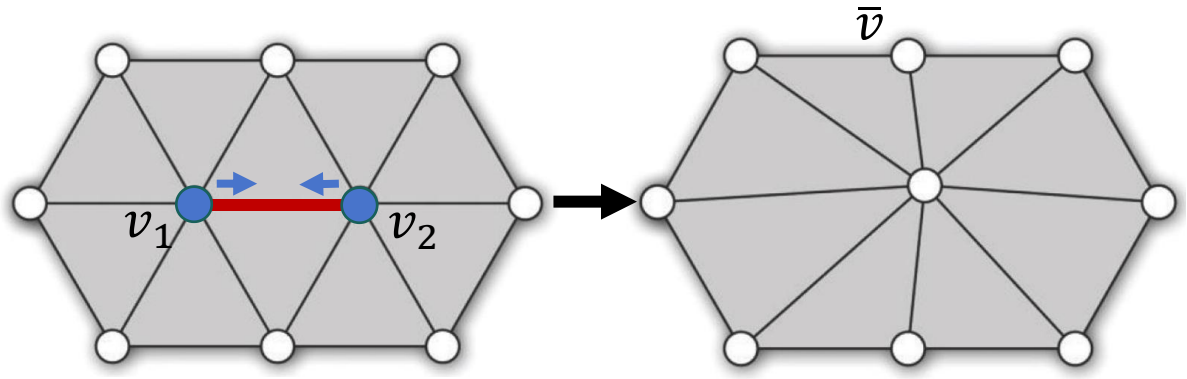


Simplification Results

5.2 网格简化算法

QEM网格简化算法

边坍缩



单步边坍缩代价：

$$\sum_{P \in plane(v_1) \cup plane(v_2)} distance(v, P)^2$$

$plane(v_1)$指顶点v_1邻接的三角面片集合，v表示更新后的顶点位置