光栅化、着色

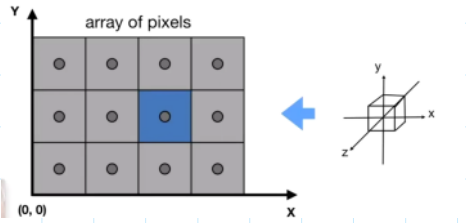
2024年7月6日 13:52

以三角形为例，学习了采样光栅化的步骤，了解了采样率不足造成的走样问题。然后从频域中高频信号重叠的角度理解了走样的成因，进而理解了为什么先对图像进行低通滤波（或者说卷积/平均）再采样能够实现反走样。了解了MSAA，TAA等抗锯齿算法的原理。
学习了深度缓存算法，其通过维护每个像素的最小深度，得到模型的遮挡关系。学习了Blinn-Phong模型，理解了漫反射项、高光项、环境光项对着色的贡献。了解三种不同的着色频率（逐多边形、逐顶点、逐像素着色）对应的着色方法。学习了常见的渲染管线。
要在模型表面贴上不同的图案，需要进行纹理映射，根据其映射到UV图中的点，获得像素中心点的漫反射系数K_d再进行着色。一般来说，多边形顶点对应的UV坐标已知，要求多边形内部点的UV坐标需要插值。对三角形通常使用重心坐标插值法。
进行逐像素着色时，需要求像素中心点的法线和漫反射系数。法线容易通过插值法求出。当纹理的分辨率过低，可能多个像素点映射到同一个纹素当中，导致出现锯齿，可以通过双线性插值等方法解决这种纹理放大的问题；当纹理的分辨率过高，一个像素点包含多个纹素，则出现对纹理采样率不足的问题，产生摩尔纹，可以通过mipmap解决。mipmap是一种快速的、近似的、方形的范围查询方法，构建了一系列分辨率更小的纹理，据此通过三线性插值计算像素漫反射系数的平均值。

光栅化

- **屏幕的简单定义：**
  由像素的阵列组成屏幕空间。像素可以简单理解为单一颜色的小方块，可由RGB值代表其颜色。屏幕分辨率即指像素阵列的大小。在屏幕空间中，每个像素都可用坐标表示。

- **光栅化步骤：**



一：
将先前通过MVP变换得到的[-1，1]^{3}的立方体映射到屏幕空间(width×height)。即不管Z方向，先通过进行拉伸和平移变换，将XY平面映射到屏幕空间上。

$$M_{viewport} = \begin{pmatrix} \frac{width}{2} & 0 & 0 & \frac{width}{2} \\ 0 & \frac{height}{2} & 0 & \frac{height}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

二：
通过采样进行光栅化：利用像素中心对三角形可见性函数inside(tri,x,y)进行采样，即判断像素中心点是否在多边形内（一般是三角形），如果是，则认为该像素在此多边形内。

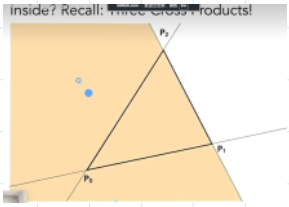
（通过判断像素中心点是否在三角形内，得到每个像素的值）

```
for (int x = 0; x < xmax; ++x)
  for (int y = 0; y < ymax; ++y)
    image[x][y] = inside(tri,
                         x + 0.5,
                         y + 0.5);
```

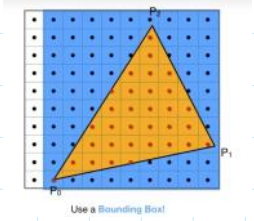
问题：
由于采样率不够造成锯齿。

- **补充：**

1.判断点是否在三角形内：利用叉乘



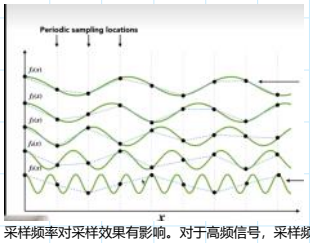
2.加速三角形采样过程的方法：
只需要计算围合Bounding Box中的像素中心点是否在三角形内



抗锯齿

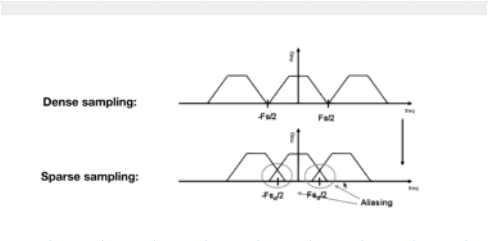
方法：先模糊处理（滤波）再采样

- **原理：**
  锯齿是走样的一种，本质是采样率不够

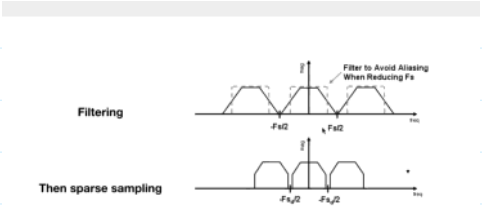


采样频率对采样效果有影响。对于高频信号，采样频率低时，出现走样（即同样的采样方法采样不同频率的函数，得到的采样结果无法区分）

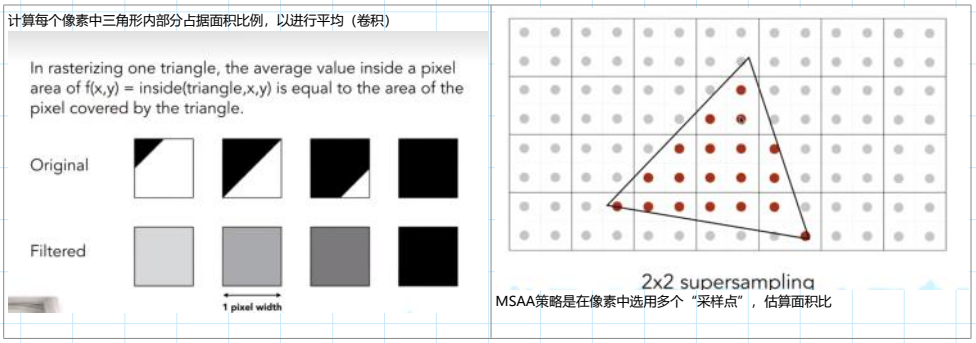
因此先对信号进行低通滤波（或者说卷积/平均），即对图像进行模糊处理，再采样，可以实现抗锯齿。反映到频域上如图所示：



反映到频域上，当采样率较低时，信号的高频部分会发生重叠，即发生走样。



- **MSAA**



计算每个像素中三角形内部分占据面积比例，以进行平均（卷积）

MSAA策略是在像素中选用多个“采样点”，估算面积比

注意，MSAA在每个像素内选用N个采样点，近似计算面积比，做平均，再做采样。是反走样的近似，但并没有真的提高采样率（没有提高像素的个数）

- **FXAA**
  对已经采样光栅化后的图像，进行图像处理，消除锯齿
- **TAA**
  对于静态的物体，可以复用上一帧的信息进行抗锯齿。

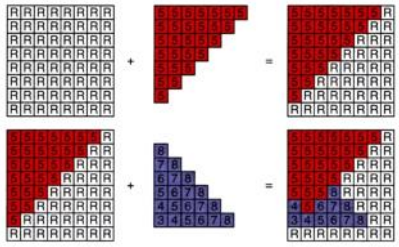
补充：超采样
通过深度学习等方法提高分辨率（如DLSS）
和反走样的本质相同，都是解决采样率不够的问题

深度缓存

深度缓存算法：
在每个多边形都已经光栅化之后，为了区分遮挡关系，使用深度缓存（depth buffer）暂存每个像素的最小深度值，遍历各多边形的像素，更新深度缓存。

During rasterization:

```
for (each triangle T)
    for (each sample (x,y,z) in T)
        if (z < zbuffer[x,y])              // closest sample so far
            framebuffer[x,y] = rgb;        // update color
            zbuffer[x,y] = z;              // update depth
        else
            ;                  // do nothing, this sample is occluded
```

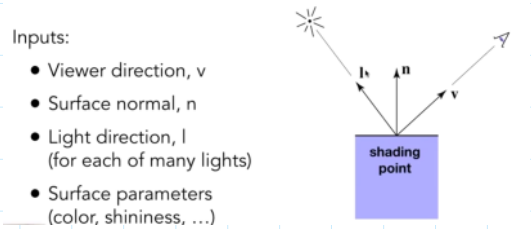


深度缓存算法需要先将每个多边形光栅化，并得到该多边形各个像素的深度信息。
时间复杂度为O(n)，n是多边形的数目。每个多边形占用像素的个数是有限的，常数个。

着色

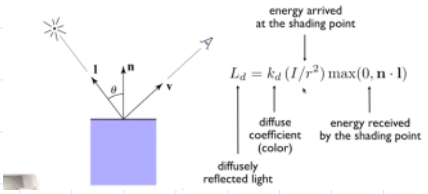
- **Blinn-Phong 反射模型**

研究对象是着色点(shading point)。
几个重要定义：（其中的向量均为单位向量）



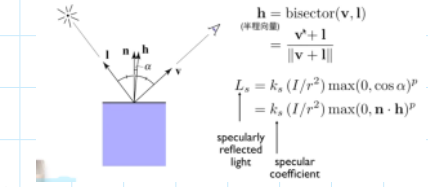
注意区分着色(shading)和阴影(shadow)，研究着色时不必考虑其他物体对光线的遮挡，也即不必考虑其他物体造成的阴影。

- 着色点光线漫反射项计算公式：

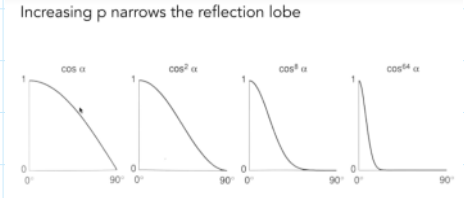


解释：
1.漫反射光线强度与观测角度没有关系，只与有多少光能量到达shading point 、法线与入射光线的夹角（有多少光的能量被吸收）、材料特征有关，光线向四面八方均匀反射。
2.Kd是与材料有关的系数，可以称为漫反射系数

- 着色点光线高光项计算公式

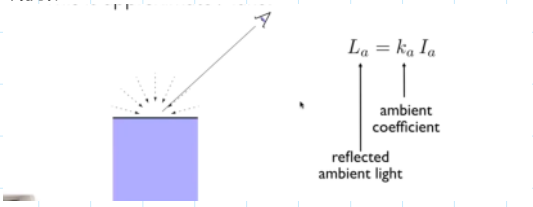


解释：
1.高光仅在观测向量v接近l反射向量时出现，可以通过半程向量h和法向量n的接近程度描述v与l反射向量间的接近程度。n*h=cosα，因此此值越小偏离得越远。
2.式子中指数p越大，则高光对α（v偏离反射角度）的容忍度越低，高光越小越集中。



如图$(\cos\alpha)^p$中指数p越大，则函数值下降越快

- 环境光项



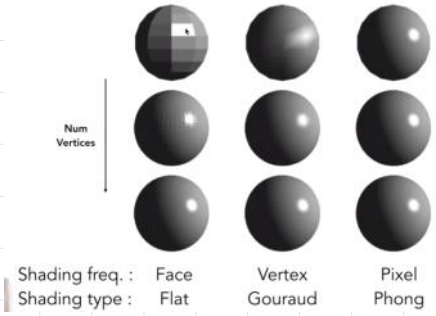
特点：
1.是一种假设，能够保证没有任何一个地方是黑的。2.假设环境光来自各个方向，并且强度一定，式子中用Ia表示其强度。

综上所述，我们得到了Biinn-Phong反射模型，这个模型是从某个着色点的光照情况入手的

$$L = L_a + L_d + L_s$$
$$= k_a\,I_a + k_d\,(I/r^2)\max(0, \mathbf{n}\cdot\mathbf{l}) + k_s\,(I/r^2)\max(0, \mathbf{n}\cdot\mathbf{h})^p$$

- **着色频率**

解释：根据Biinn-Phong反射模型我们能够计算每个着色点的着色情况，接下来能有下面三种有不同着色频率的着色方式，完成整个模型的着色



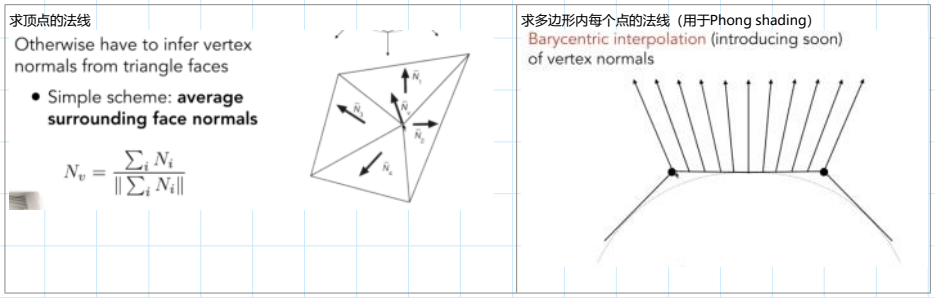
1.逐多边形着色，选择多边形内一点作为着色点，着色结果作为整个多边形的颜色。

2.逐顶点着色，计算每个顶点的着色情况，再通过插值法得到顶点围成多边形的颜色。逐顶点着色先求出顶点的法线、漫反射系数、求出着色情况，再插值求出多边形内部的像素中心点的着色情况。

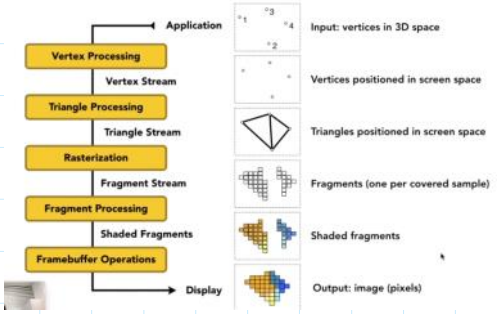
3.逐像素着色（Phong shading）。逐像素着色需要求出每个像素中心点对应的法线，并通过纹理映射求出漫反射系数，计算该点的着色情况。

着色点法线的计算方法：

求顶点的法线



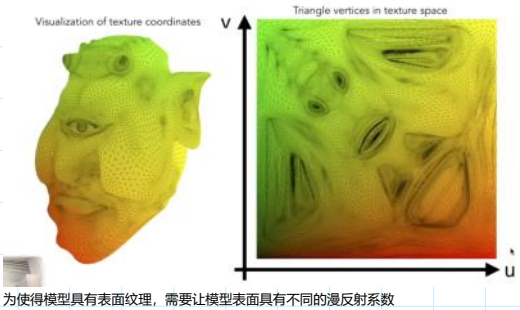
图形渲染管线



shading根据着色频率的不同，可以在顶点处理阶段完成，也可以在片段（fragment）/像素（pixel）处理阶段完成。对应vertex shader 和 fragment shader

---

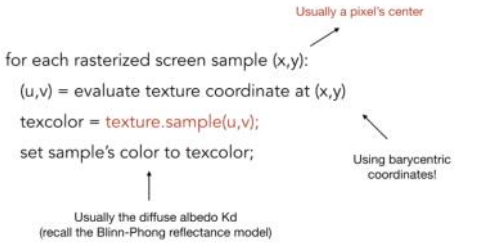
纹理映射



为使得模型具有表面纹理，需要让模型表面具有不同的漫反射系数

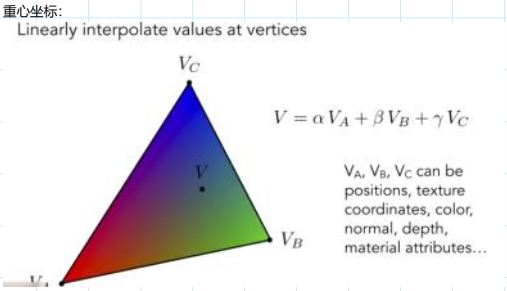
为了简化这个过程，将三维模型的表面展开成为二维平面，得到一张UV图，任何一个三维模型的三角形都可以映射到某个二维UV图的三角形，任何一个

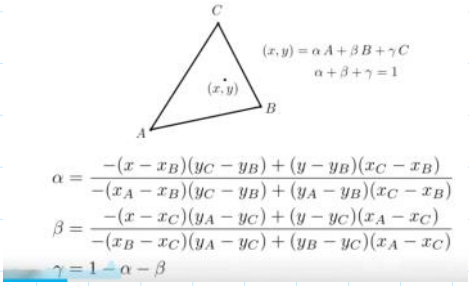
点都能映射到UV图中，这称为纹理映射。
因此找到像素点对应的纹理坐标和对应的纹理值/texture.sample，作为该点处的材料漫反射系数/texcolor



- **补充：三角形内部插值方法：重心坐标插值法**

已知顶点属性，可以通过插值实现该属性在三角形内部的平滑过渡。例如，已知顶点法线，求内部任一点法线；已知顶点着色情况，求求内部点的着色情况，已知顶点的UV坐标，求内部任一点的UV坐标。插值是一近似方法。
重心坐标：



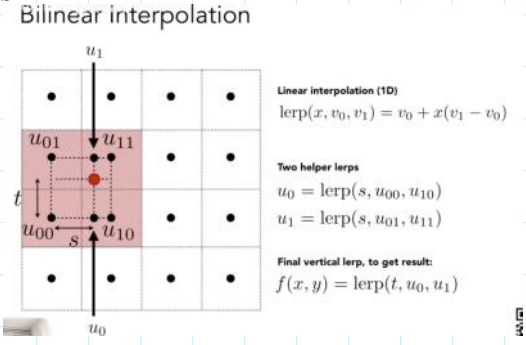


给出三角形任意一点的坐标，可以计算出α，β，γ值（设定三者之和为1），以此来计算该点的属性
注意，三维空间中的三角形某点的重心坐标值在投影到二维平面后可能发生变化。例如在深度缓存中，三角形光栅化后需要知道所占每个像素的深度值，因此使用插值法求像素中心点的深度值，要使用该像素中心点对应三维空间中的三角形的点的重心坐标。
总之在三维空间中点的属性，建议在三维空间中做插值，再将结果应用到二维空间中的点。

纹理放大(texture magnification)

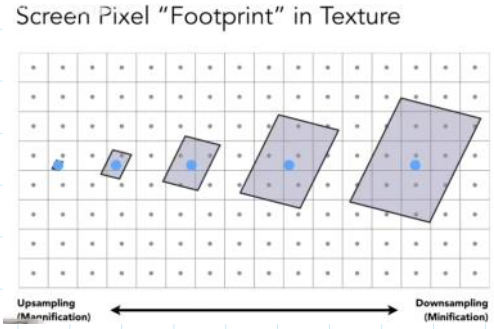
Texture Magnification（纹理放大）是计算机图形学中的一个重要概念，发生在纹理的分辨率小于物体表面分辨率，即“纹素”texel的数量远小于物体表面在屏幕上占据的像素数时。这意味着纹理需要被放大以覆盖更大的表面区域，这可能导致纹理的细节变得模糊或出现锯齿状边缘。如何有效地将纹理映射到物体表面，并避免产生视觉上的不连续或模糊效果或者锯齿是其需要解决的问题。

当物体表面分辨率小于纹理分辨率时，可能会有多个像素点映射到同一个texel中的不同位置。此时根据周围的四个texel进行双线性插值，确定该像素点的texcolor



- **纹理映射中的Downsampling：**

通常可能出现两种问题：1.物体表面分辨率大于纹理分辨率（upsampling），产生锯齿，即上述需要纹理放大时的情况 2. 物体表面分辨率小于纹理分辨率（downsampling），产生摩尔纹



mipmap

Antialiasing — Supersampling?

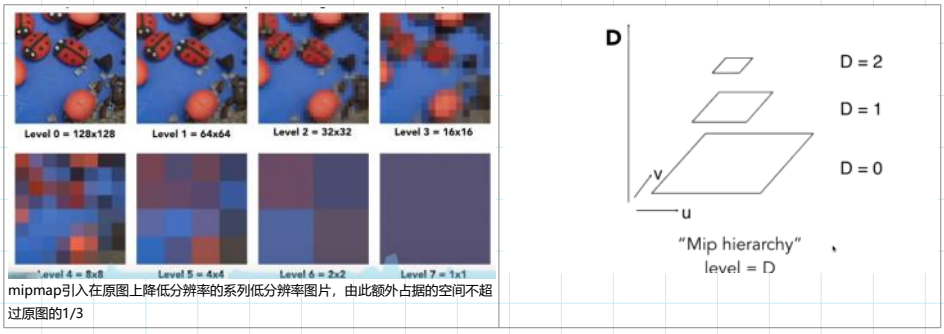
Will supersampling work?

- Yes, high quality, but costly
- When highly minified, many texels in pixel footprint
- Signal frequency too large in a pixel
- Need even higher sampling frequency

Let's understand this problem in another way

- What if we don't sample?
- Just need to **get the average value within a range**!

Mipmap可以用来解决downsampling。通常，为解决downsampling出现的问题通常可以采用超采样的方法，增加“采样点”然后做平均，典型的例子是MSAA。
而mipmap不通过超采样，直接求平均。

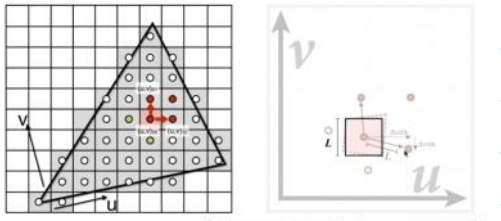


mipmap引入在原图上降低分辨率的系列低分辨率图片，由此额外占据的空间不超过原图的1/3

- mipmap实际上一种范围查询的问题，给出一个区域，希望得到其平均值是多少。当然，范围查询解决的对象不仅仅是一个区域中的平均值，与范围查询相对的是点查询，例如在解决upsampling当中的使用点查询，通常可以引入插值的方法。
- mipmap是一种快速的近似方法，进行近似时每个像素在UV图中对应一个正方形。

- mipmap方法流程

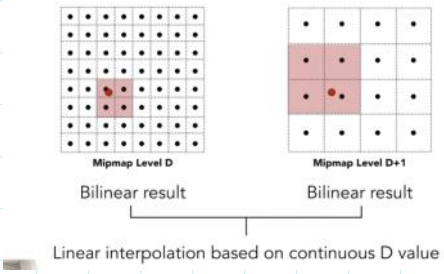
1.近似估计像素在纹理图中的大小，该像素在纹理图中近似成一个正方形。



$$D = \log_2 L \qquad L = \max\left(\sqrt{\left(\frac{du}{dx}\right)^2 + \left(\frac{dv}{dx}\right)^2}, \sqrt{\left(\frac{du}{dy}\right)^2 + \left(\frac{dv}{dy}\right)^2}\right)$$

2.根据所得的D值，找到对应的mip hierarchy level。使得该像素在纹理图中的对应正方形正好占一个texel的大小。
例如：。。。

3.将像素中心点映射到对应的level上，找到像素的texcolor。当所得的D值不是整数时，需要在双线性插值得到两个点的texcolor的基础上再进行插值，即三线性插值。



该方法的开销是两次点查询，和一次插值。