

SHADING 渲染

SHADING 渲染

- Visibility/Occlusion
- Z-Buffer
- Blinn-Phong Reflectance Model
 - Specular Highlights
 - Diffuse Reflection
 - Ambient Lighting
- Shading Frequencies
 - Flat Shading
 - Gouraud Shading
 - Phong Shading
- Graphics(Real-time Rendering) Pipeline
- Texture Mapping
 - Barycentric Coordinates
 - linearly interpolate values at vertices
 - Applying Textures
 - Texture Magnification
 - Easy Case: insufficient texture resolution
 - Bilinear Interpolation
 - Hard Case: the texture is too large
 - Mipmap

definition: The process of applying a material to an object

remark: Shading ≠ shadow

Visibility/Occlusion

1. 画家算法: 从远到近渲染物体, 近的物体可能会遮蔽远的物体。但是可能只是部分遮挡, 因此实际中不采用此算法
2. **Z-Buffer:** 以像素为单位进行画家算法

Z-Buffer

idea

- 对于每一个像素存储当前最近的 `z-value`
- 需要额外的缓存结构存储深度信息
 1. `frame buffer` 存储颜色
 2. `depth buffer(Z-Buffer)` 存储深度

algorithm

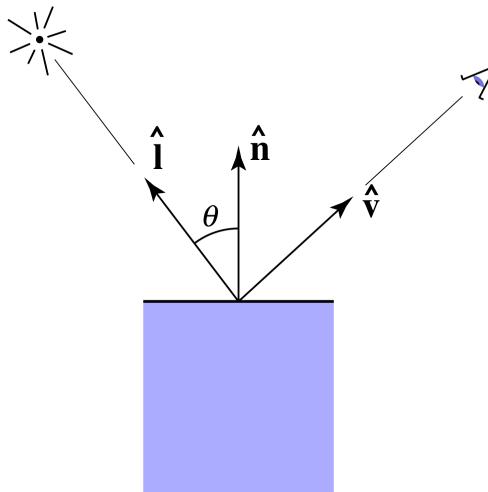
```
for (each triangle T)
    for (each sample (x,y,z) in T)
        if (z < zbuffer[x,y])
            framebuffer[x,y] = rgb
            zbuffer[x,y] = z
```

Blinn-Phong Reflectance Model

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

perceptual observations

- specular highlights 高光
- diffuse reflection 漫反射
- ambient lighting 环境照明



Specular Highlights

idea: \mathbf{V} close to mirror direction \iff half vector near normal

$$L_s = k_s (I/r^2) \max(0, \cos \alpha)^p = k_s (I/r^2) \max(0, \mathbf{n} \cdot \mathbf{h})^p$$

- L_s specularly reflected light
- k_s specular coefficient
- 半程向量 $\mathbf{h} = \frac{\mathbf{v}+\mathbf{l}}{\|\mathbf{v}+\mathbf{l}\|}$, 使用半程向量方便计算
- p increasing p narrows the reflection lobe, usually $p \in [100, 200]$

remark: intensity depends of view direction

Diffuse Reflection

$$L_d = k_d (I/r^2) \max(0, \mathbf{n} \cdot \mathbf{l})$$

- L_d diffusely reflected light
- k_d diffuse coefficient(color)
- I/r^2 energy arrived at the shading point
- $\mathbf{n} \cdot \mathbf{l}$ energy received by the shading point

remark: shading independent of view direction

Ambient Lighting

$$L_a = k_a I_a$$

- L_a reflected ambient light
- k_a ambient coefficient

Shading Frequencies

以下3种着色方式并没有完全的优劣之分

Flat Shading

shade each triangle

Gouraud Shading

shade each vertex

Per-Vertex Normal Vectors

- simple scheme $N_v = \frac{\sum_i N_i}{\|\sum_i N_i\|}$

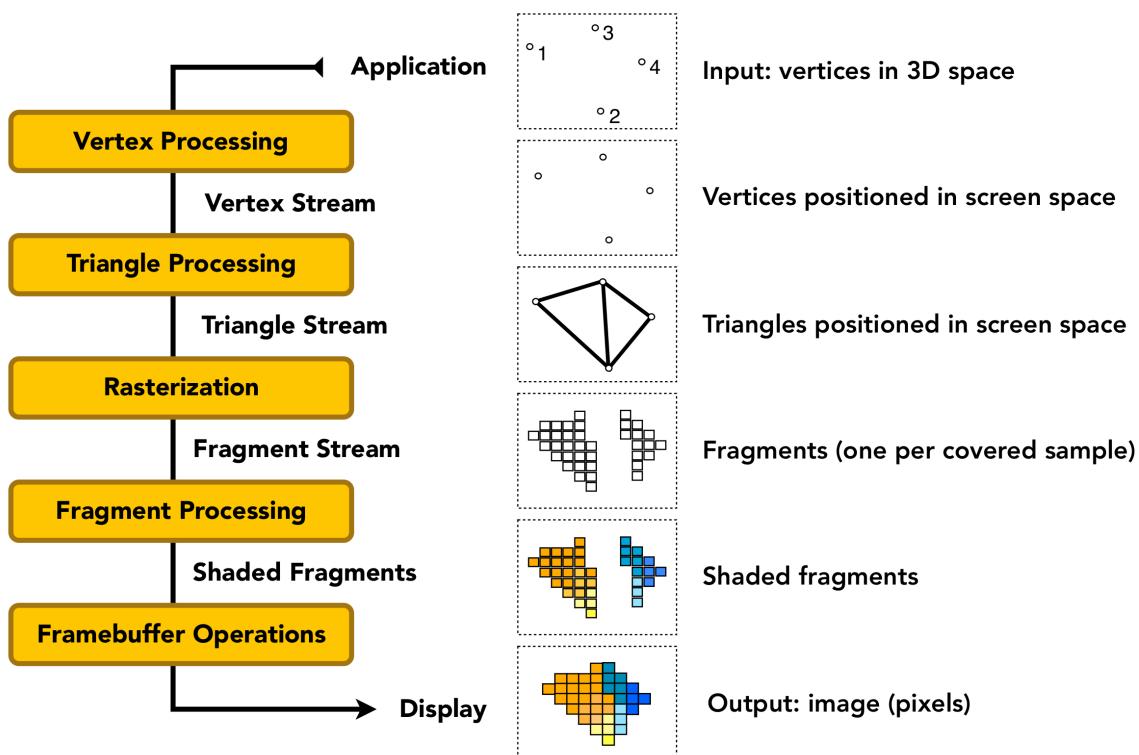
Phong Shading

shade each pixel

Per-Pixel Normal Vectors

remark: not the Blinn-Phong Reflectance Model

Graphics(Real-time Rendering) Pipeline



Texture Mapping

definition: Every 3D surface point has a place where it goes in the 2D image(texture)

remark: Each triangle vertex is assigned a texture coordinate $(u, v) \in [0, 1]^2$

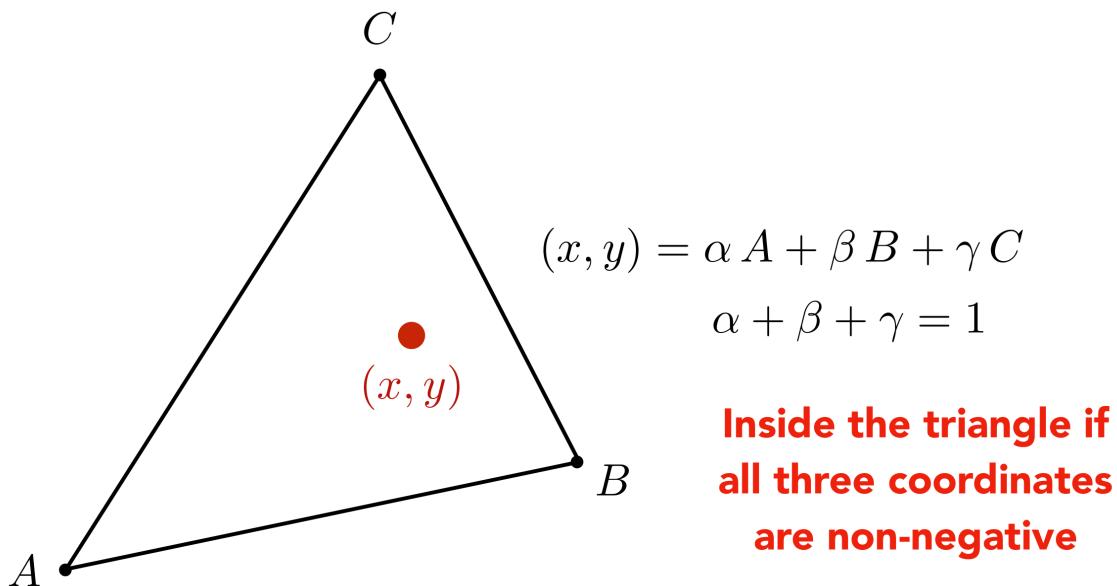
Barycentric Coordinates

why do we want to interpolate?

- specify values at vertices
- obtain smoothly varying values across triangles

definition: 三角形中的任意一点都可以表示为三个顶点的凸组合，因此可以使用凸组合的系数表示三角形内的任意一点，且系数与该点分割原三角形形成的3个子三角形的面积成正比

A coordinate system for triangles (α, β, γ)



linearly interpolate values at vertices

$$V = \alpha V_A + \beta V_B + \gamma V_C$$

其中 V_A, V_B, V_C 可以是位置、纹理、坐标、颜色、法向、深度、材料属性

remark: 重心坐标在投影下不能保持不变性，因此三维空间中的坐标需要先作插值再作投影

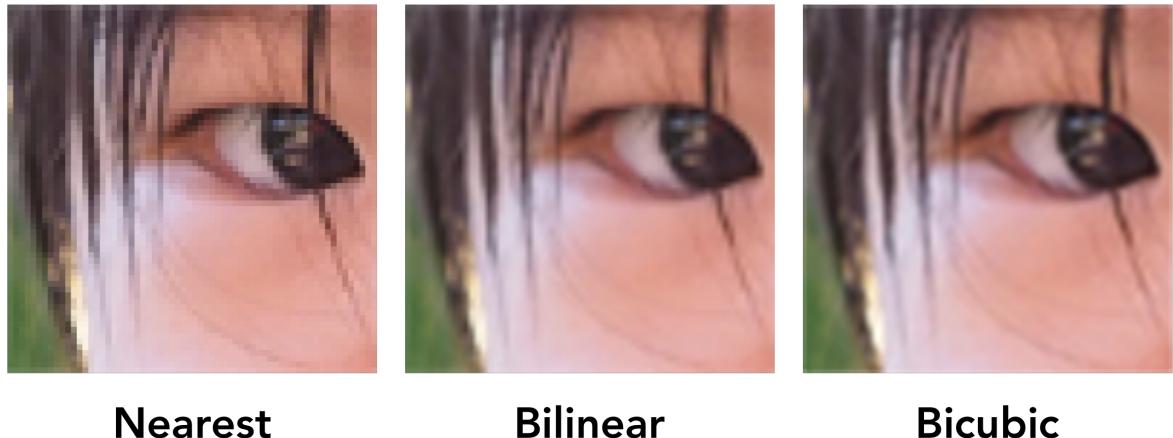
Applying Textures

algorithm

```
for each rasterized screen sample (x, y)
    (u, v) = evaluate texture coordinate at (x, y)
    texcolor = texture.sample(u, v)
    set sample's color to texcolor
```

Texture Magnification

Easy Case: insufficient texture resolution

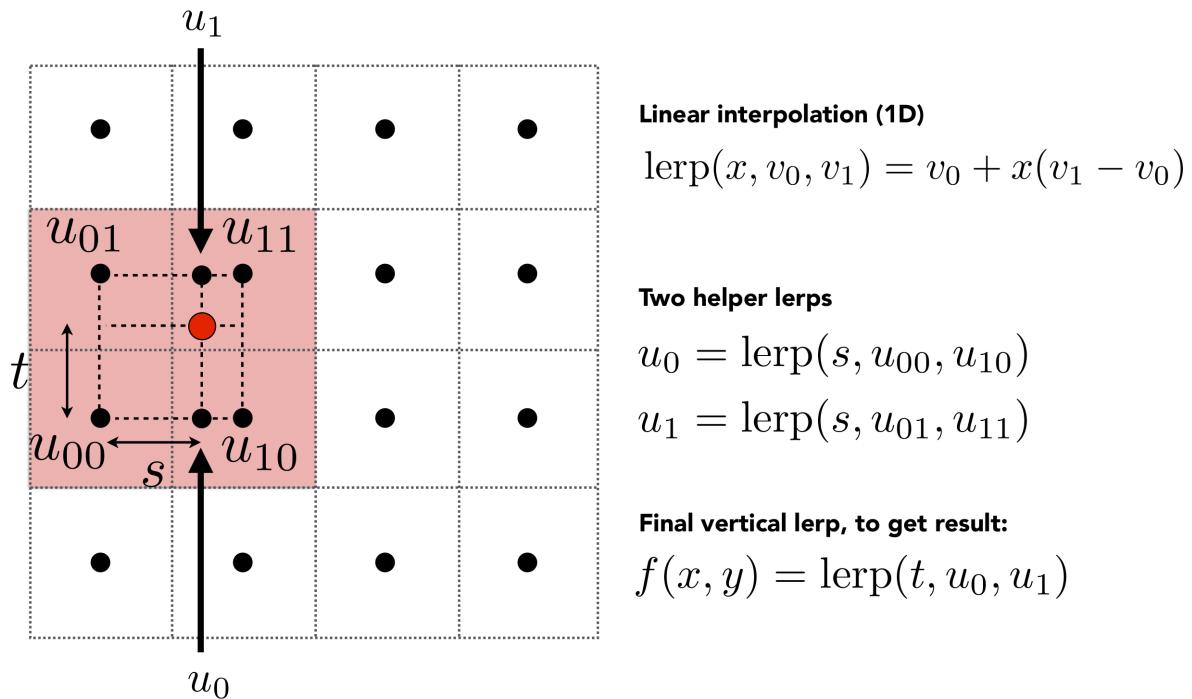


Nearest

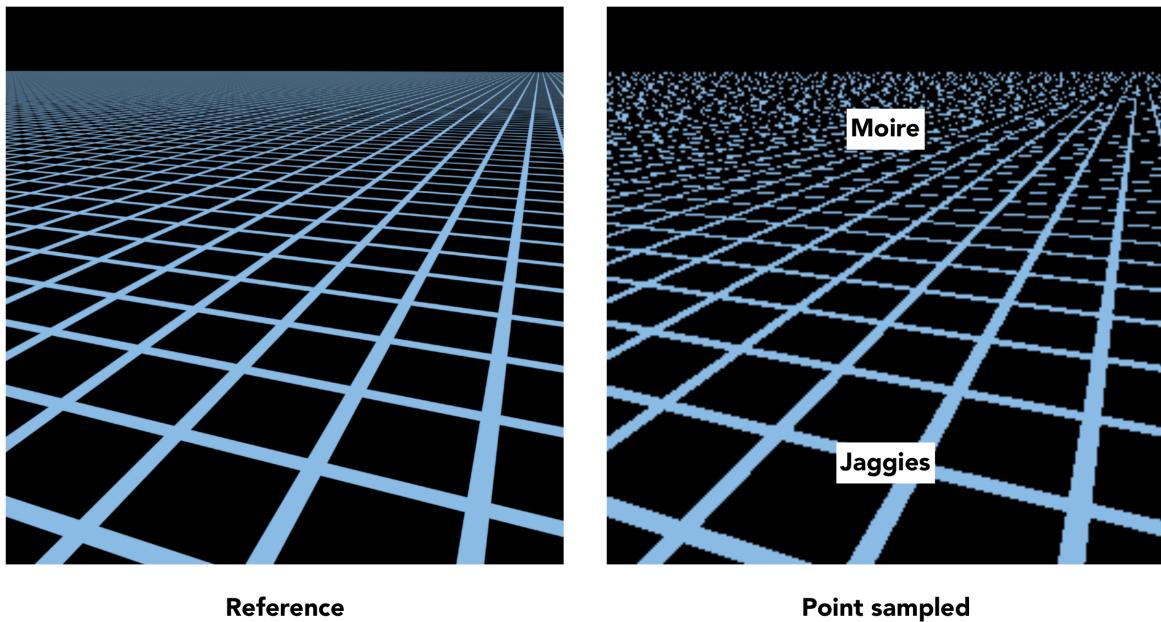
Bilinear

Bicubic

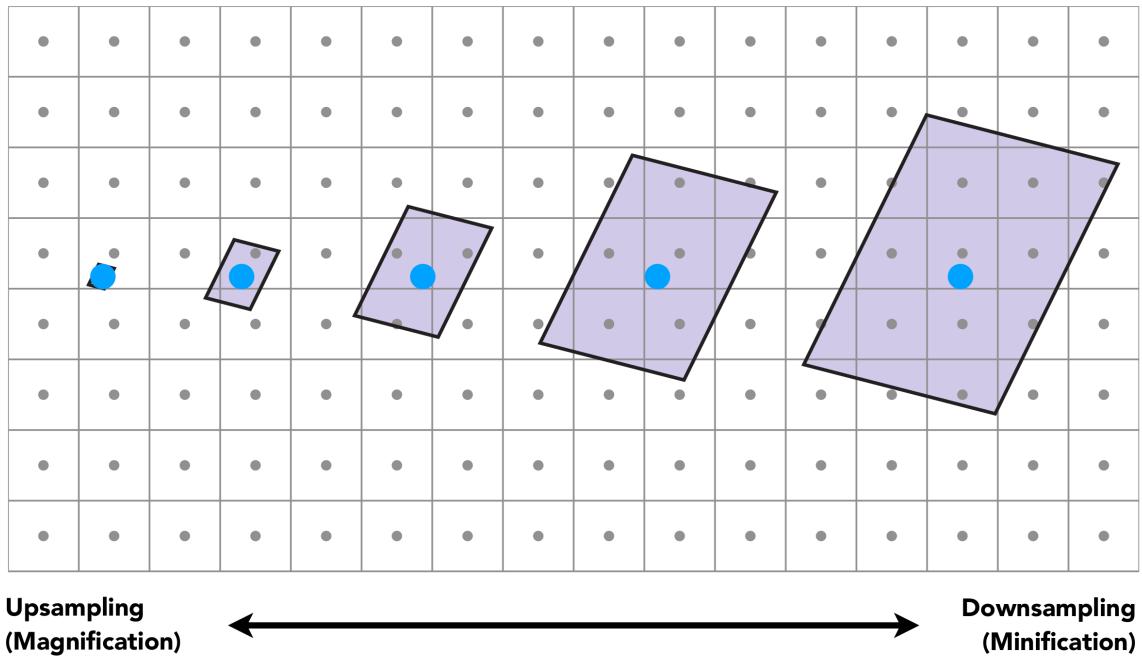
Bilinear Interpolation



Hard Case: the textrue is too large

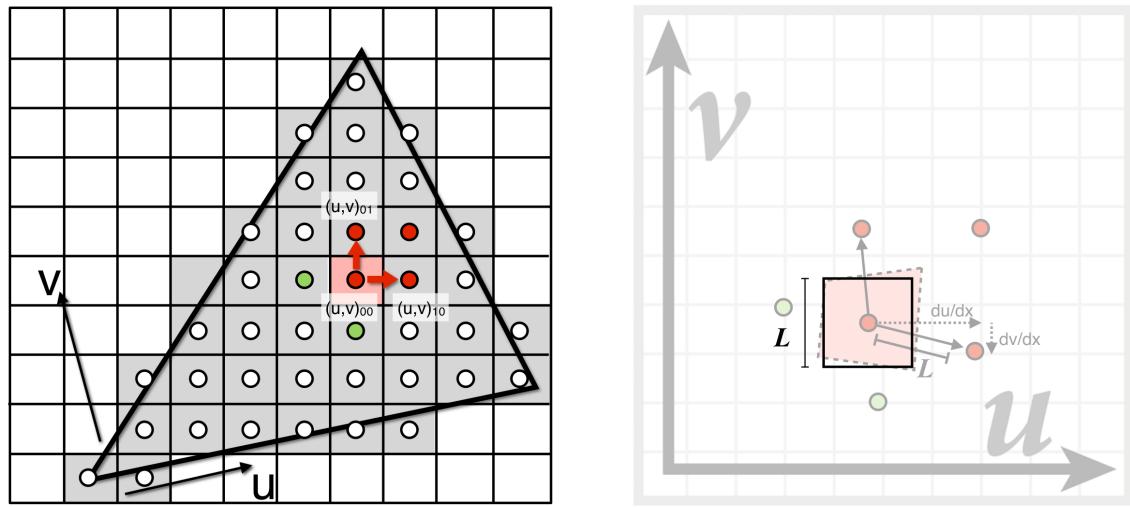


why aliasing?



Mipmap

Allowing(fast,approx,square)range queries



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

如果 L 不是整数，则可以在Mipmap的不同层之间作插值计算像素点。考虑到计算 L 需要一个双线性插值，所以此方法也被称作三线性插值。三线性插值的优点有：开销低，连续性好

缺点：Overblur(过分模糊)，可以用anisotropic filtering（各向异性过滤）

