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 Textrues
  - Texture Magnification
    - Easy Case: insufficient texture resolution
    - Bilinear Interpolation
    - Hard Case: the textrue is too large
    - Mipmap
  - Environment Map
    - Spherical Map
    - Cube Map
  - Bump Mapping
  - Displacement Mapping

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

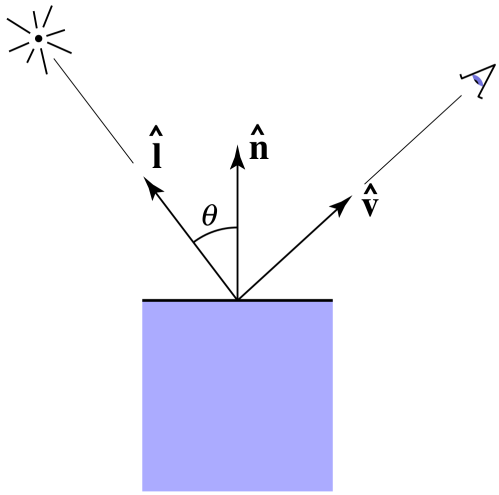
```
for (each triangle T)
    for (each sample (x,y,z) in T)
        if (z < zbuffer[x,y])
            framebuffer[x,y] = rgb
            zbuffer[x,y] = z
```

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: 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 coeffient

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 $N_v = \frac{\sum_i N_i}{\|\sum_i N_i\|}$

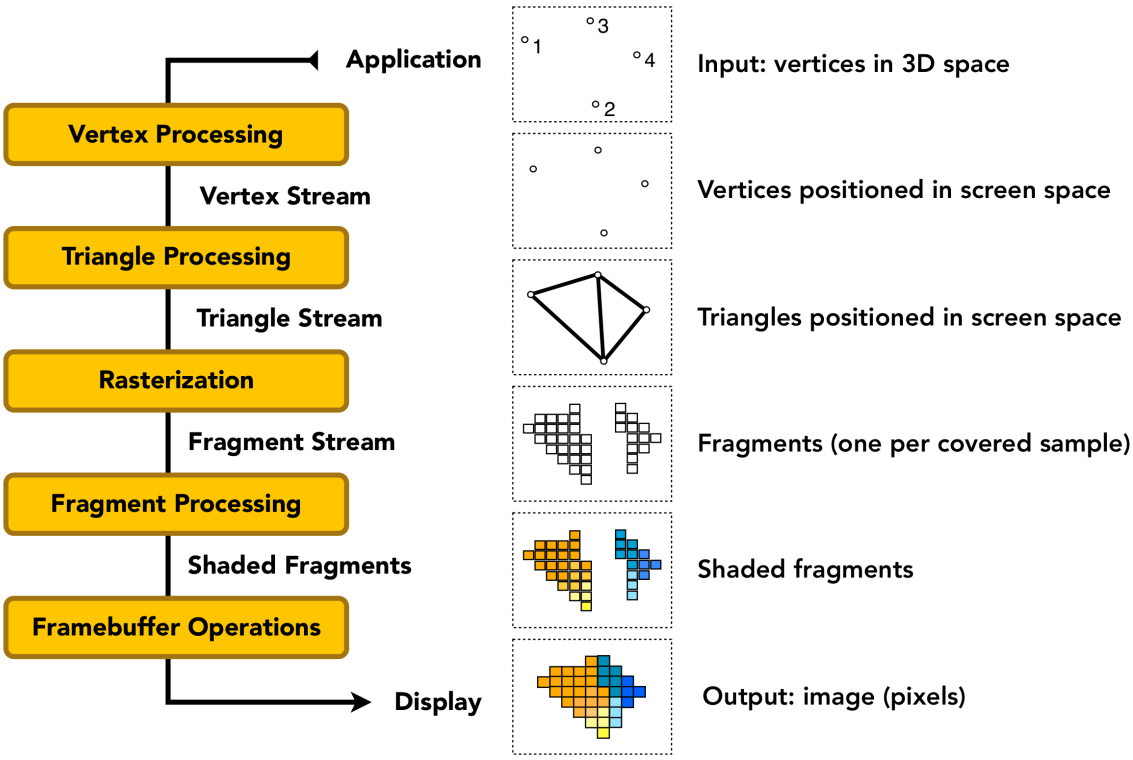
Phong Shading

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)

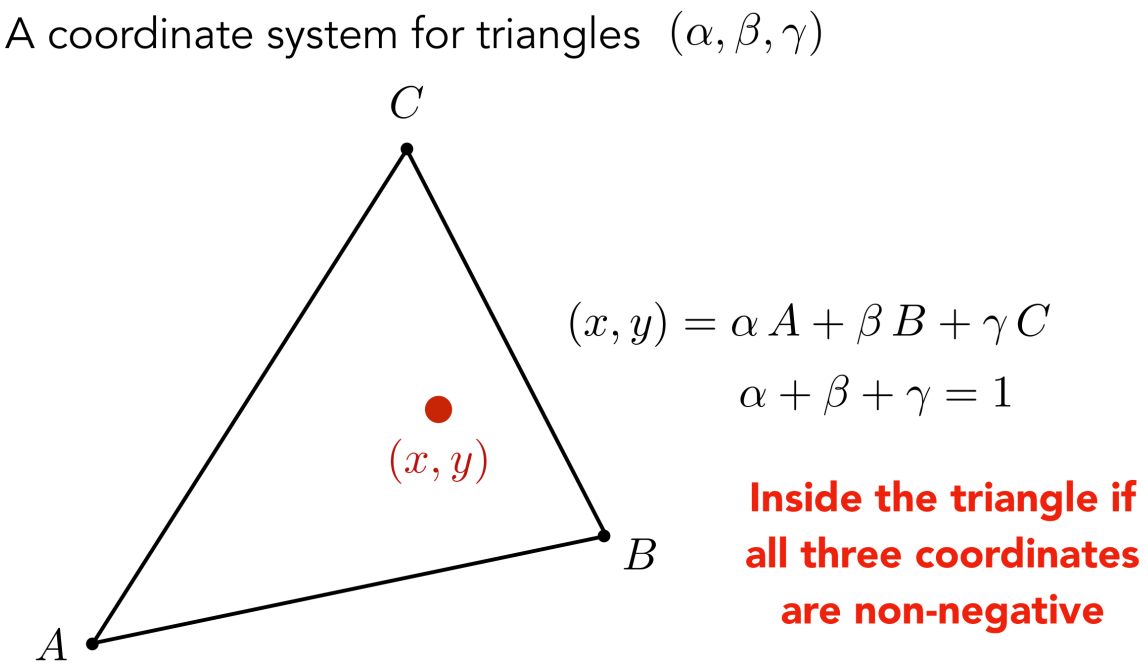
remark: Each triangle vertex is assigned a texture coordinate $(u, v) \in [0, 1]^2$

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



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 Textrues

algorithm

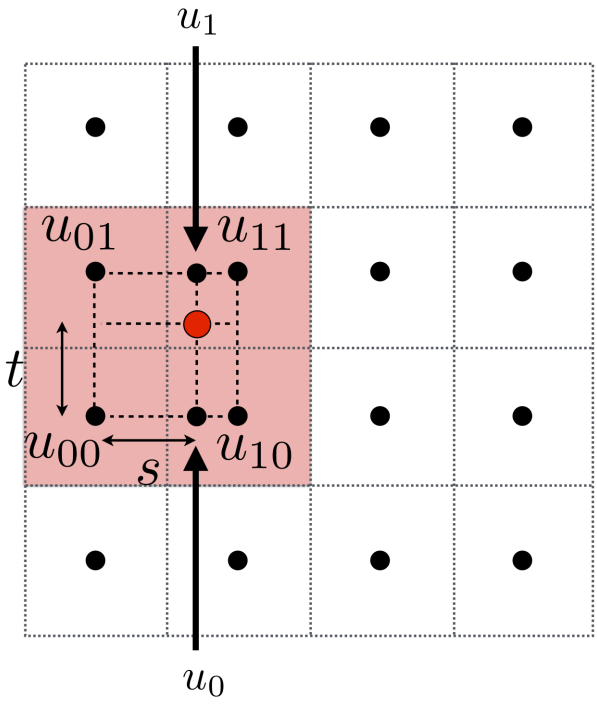
```
for each rasterized screen sample (x,y)
    (u,v) = evaluate texture coordinate at (x,y)
    texcolor = texture.sample(u,v)
    set sample's color to texcolor
```

Texture Magnification

Easy Case: insufficient texture resolution

Nearest Bilinear Bicubic

Bilinear Interpolation



Linear interpolation (1D)

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

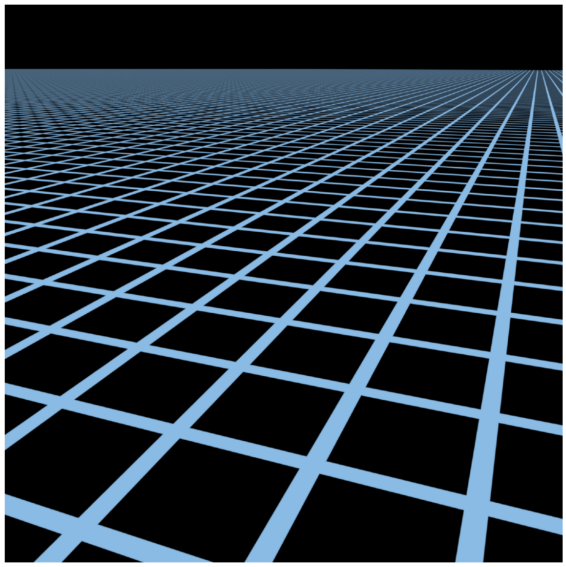
Two helper lerps

$$u_0 = \text{lerp}(s, u_{00}, u_{10})$$
$$u_1 = \text{lerp}(s, u_{01}, u_{11})$$

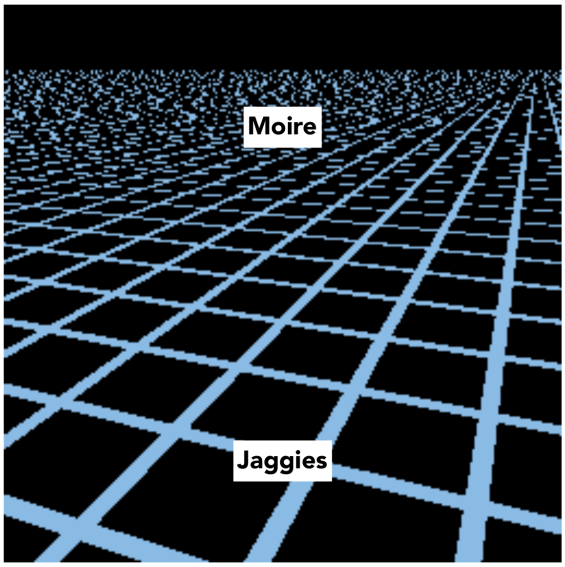
Final vertical lerp, to get result:

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

Hard Case: the textrue is too large

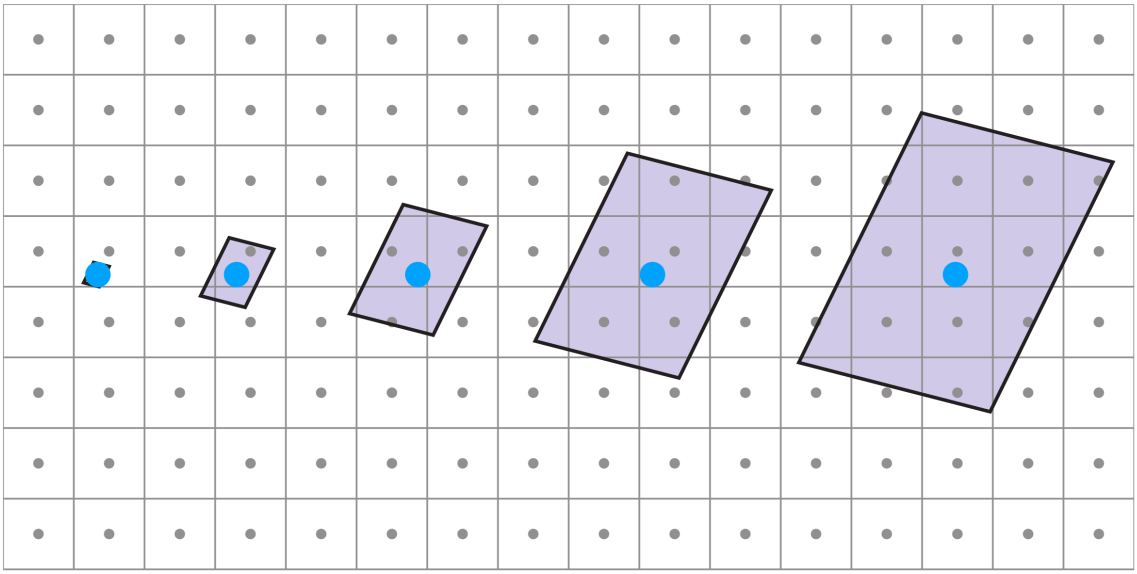


Reference



Point sampled

why aliasing?

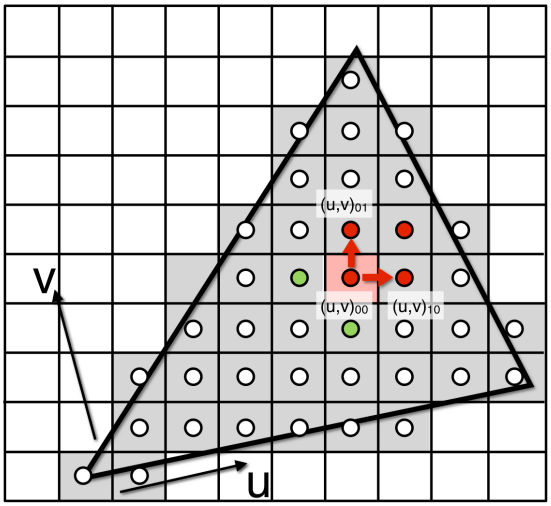


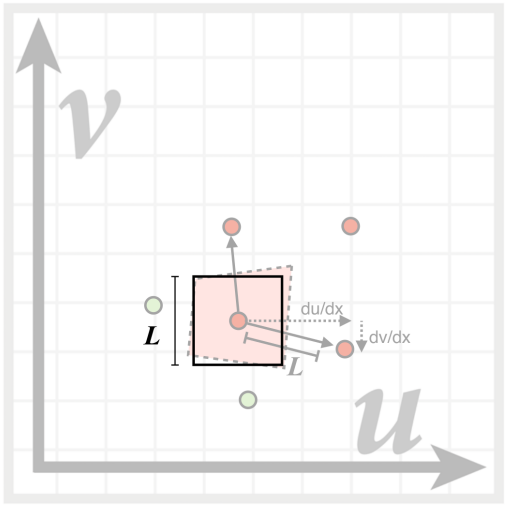
Upsampling
(Magnification)

Downsampling
(Minification)

Mipmap

Allowing(fast,approx,squra)range queries

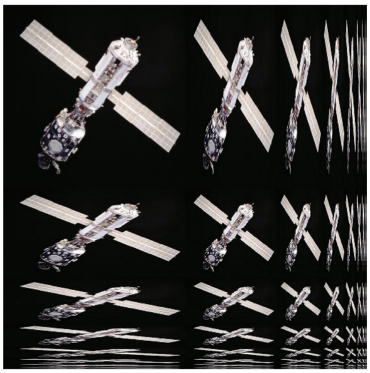




$$D = \log_2 L \qquad L = \max\left(\sqrt{\left(\frac{du}{dx}\right)^2 + \left(\frac{dv}{dx}\right)^2}, \sqrt{\left(\frac{du}{dy}\right)^2 + \left(\frac{dv}{dy}\right)^2}\right)$$

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

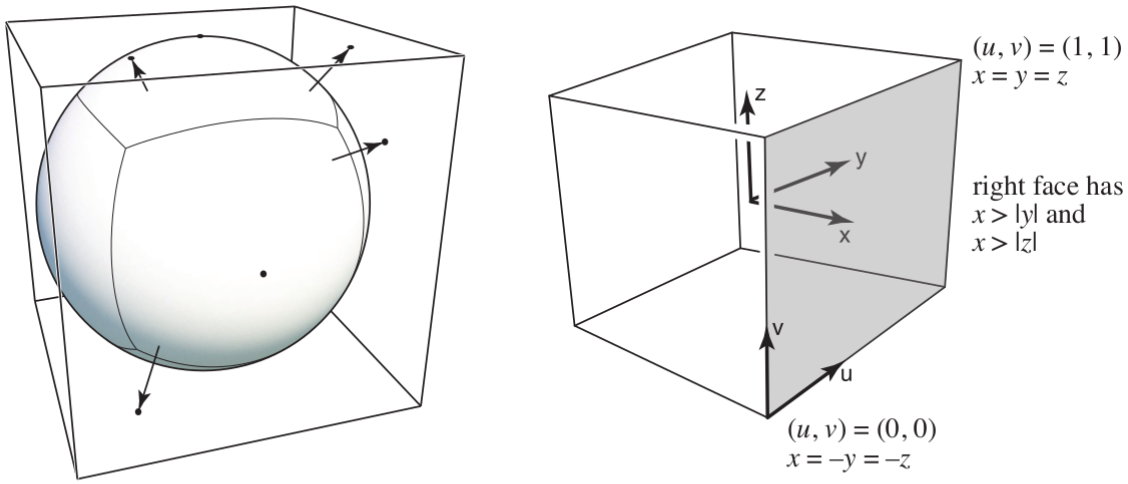
缺点： Overblur(过分模糊)，可以用anisotropic filtering（各相异性过滤）



Environment Map

Spherical Map

Cube Map



A vector maps to cube point along that direction.
The cube is textured with 6 square texture maps.

Bump Mapping

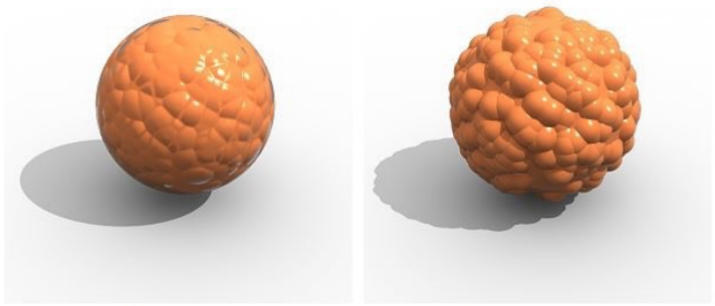
通过扰动每个像素的法线改变纹理，使其看起来凹凸不平

algorithm

- original surface normal $\mathbf{n}(p) = (0, 0, 1)$ 假设在局部坐标系中，因此法线是(0, 0, 1)
- derivatives at p are
    - $\frac{dp}{du} = c_1 \cdot [h(\mathbf{u} + 1) - h(\mathbf{u})]$
    - $\frac{dp}{dv} = c_2 \cdot [h(\mathbf{v} + 1) - h(\mathbf{v})]$
- perturbed normal is $\mathbf{n} = (-\frac{dp}{du}, -\frac{dp}{dv}, 1)$.normalized()

Displacement Mapping

相比Bump Mapping真的移动了顶点



Bump / **Normal** mapping　Displacement mapping