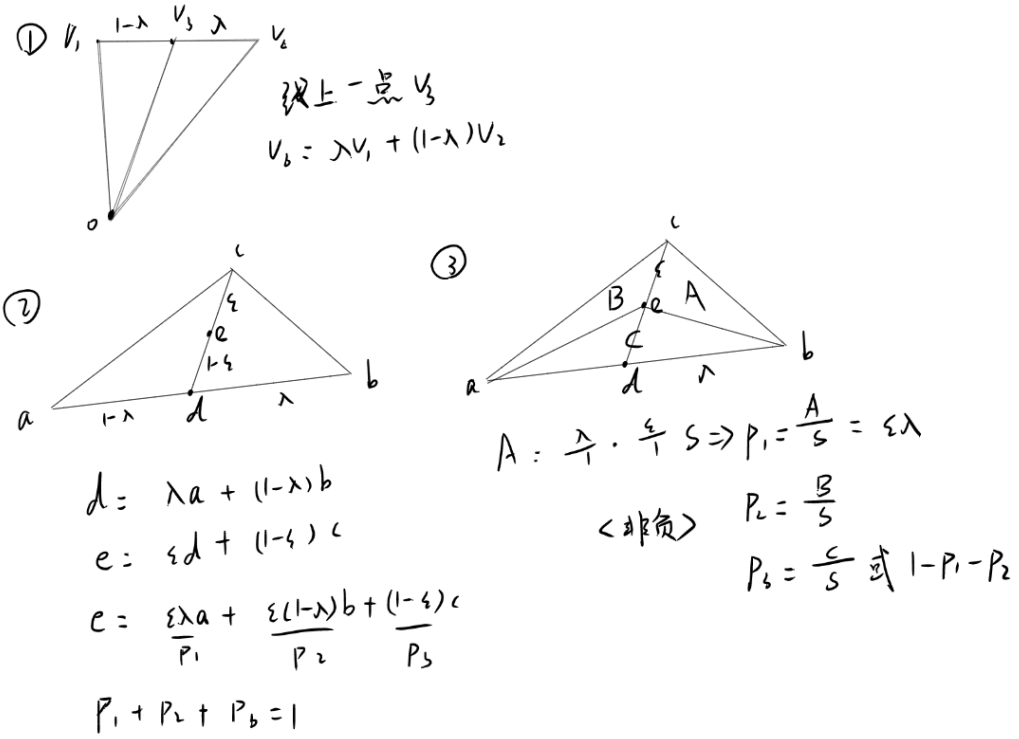
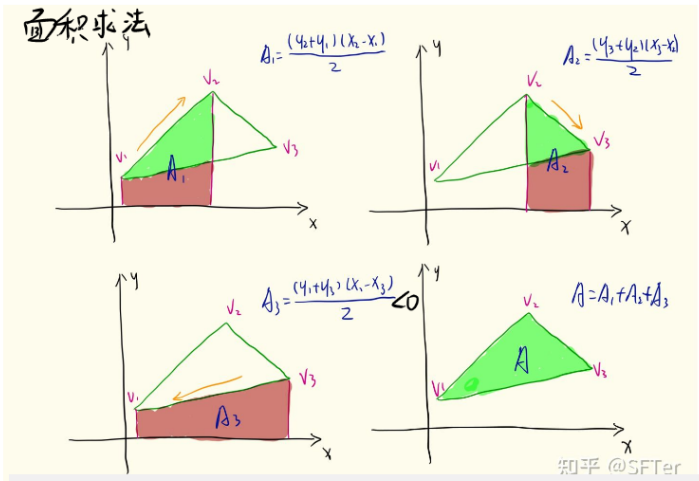
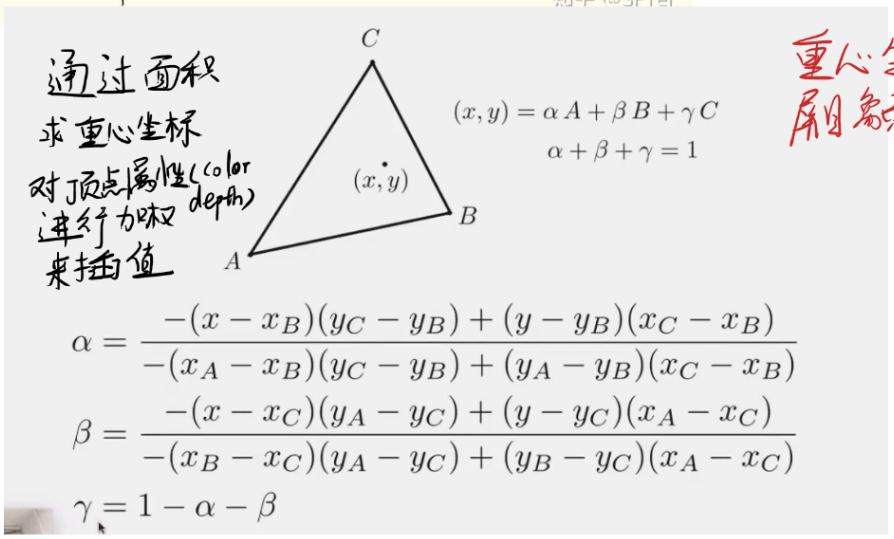
Shading 3

Barycentric coordinates 重心坐标







重心坐标三维投影二维不同
属性插值需逆变换到三维空间

Texture queries : Applying Textures

像素在三角形内→计算对应uv→取纹理对应颜色值→设置

问题1: Texture Magnification 纹理太小怎么办 → 插值

- 纹理像素：texel
- 多个pixel映射到了同一个texel
- 解决：
    - Nearest
    - Bilinear
        - Bilinear 插值 lerp
        - 水平+竖直插值→双线性插值
        - 最近的四个点插值
    - Bicubic 双向三次插值
        - 周围16个点做三次插值
        - 运算量更大，结果更好

问题2: Texture Magnification 纹理太大怎么办

- 一个pixel对应了多个texel → 采样频率不足导致 摩尔纹+锯齿（走样）
- 解决：
    - Supersampling
        - 太浪费！
        - Just need to get the average value within a range
            - Point Query vs. **(Avg.) Range Query**
    - Mipmap：Allowing (**fast, approx., square**) range queries
        - 每一次长宽各减半 D=0,1,2,...
        - “Mip hierarchy”
        - overhead: 1/3
        - 怎么知道层数D？约为相邻pixel的映射uv之间的距离取2的对数
        - 如果计算出来需要的D是整数，就很方便
        - 如果计算出来需要的D不是整数→Trilinear Interpolation三线性插值
            - 分别在floor(D)和ceil(D)上做Bilinear Interpolation取颜色值之后再插值
        - Limitation：Overblur
            - 不是方块查询
            - Solution：各向异性过滤
    - 各向异性过滤Anisotropic Filtering
        - Ripmaps and summed area tables
            - Can look up axis-aligned rectangular zones
            - 长/宽/长和宽 各减半
        - EWA filtering 椭圆取样
            - 利用多次查询求平均值的方法来处理不规则区域
        - overhead：3
        - 多少x：压缩到多少x，显存足够的情况下开越高越好

像素覆盖的区域大小各不相同

Application of Texture

各种贴图

texture = memory + range query (filtering)

- General method to bring data to fragment calculations

Many applications

- Environment lighting - Environment Map
  - 环境光贴图
  - 例子：Utah Teapot
    - 经典：Stanford Bunny，Cornell Box
  - Spherical Environment Map
    - 球心：世界中心
    - 一个问题：拉伸，想象地球仪展开
    - 解决方法：Cube Map
  - Cube Map：立方体表面，从球心到球面的投影向外
    - 扭曲更少，但是Need dir->face computation，计算量更大
- Store microgeometry
  - Textures can affect shading! → define height/normal → Bump / Normal Map
    - 两者类似，都可以以假乱真
    - 改变表面的法线
  - Bump Mapping 凹凸贴图

    Bump Mapping的Texture记录了高度移动
    - 不改变几何信息
    - 逐像素扰动法线方向
    - 高度 offset 相对变化，从而改变法线方向
    - 计算法线方向：切线的垂直方向
  - Displacement mapping 位移贴图
    - 输入相同（Texture与Bump Mapping可共用）
    - 改变几何信息，对顶点做位移
    - 相比上更逼真，要求模型足够细致，运算量更高
    - DirectX有Dynamic的插值法，对模型做插值，使得初始不用过于细致
- Procedural textures
  - 3D Procedural Noise + Solid Modeling
    - 定义空间中任意点的颜色
    - 噪声+映射→
    - Perlin Noise
- Provide Precomputed Shading
  - Ambient occlusion texture map
    - 计算好的环境光遮蔽贴图
    - 空间换时间
- Solid modeling & Volume rendering
  - 三维渲染

Shadow mapping

光栅化下对全局光线传输、阴影的处理十分麻烦。

- draw shadows using rasterization
- An Image-space Algorithm
  - 不需要场景的几何信息
  - 有走样现象
  - 思想：the points NOT in shadow must be seen both by the light and by the camera

步骤：

- Pass 1: Render from Light
  - Depth image from light source → shadow map
- Pass 2A: Render from Eye
  - Standard image (with depth) from eye
- Pass 2B: Project to light
  - Project visible points in eye view back to light source
    - visible to light source → color
    - blocked → shadow

感觉每个光源对每个静态场景有一个shadow map

应用：

- 几乎所有3D游戏

问题：

- 走样、分辨率
- 数值精度问题
  - Involves equality comparison of floating point depth values means issues of scale, bias, tolerance
- 只能点光源、硬阴影