GAMES101

All rights reserved by [CWHer]

目录

- [x] 【1】：计算机图形学概述
- [x] 【2】：向量与线性代数
- [x] 【3】：变换（二维与三维）
- [x] 【4】：变换（模型、视图、投影）
- [x] 【5】：光栅化（三角形的离散化）
- [x] 【6】：光栅化（深度测试与抗锯齿）
- [x] 【7】：着色（光照与基本着色模型）
- [x] 【8】：着色（着色频率、图形管线、纹理映射）
- [x] 【9】：着色（插值、高级纹理映射）
- [x] 【10】：几何（基本表示方法）
- [x] 【11】：几何（曲线与曲面）
- [x] 【12】：几何（网格处理）、阴影图
- [x] 【13】：光线追踪（基本原理）
- [x] 【14】：光线追踪（加速结构）
- [x] 【15】：光线追踪（辐射度量学、渲染方程与全局光照）
- [x] 【16】：光线追踪（蒙特卡洛积分与路径追踪）
- [x] 【17】：材质与外观
- [x] 【18】：高级光线传播与复杂外观建模
- [x] 【19】：相机与透镜
- [x] 【20】：光场、颜色与感知
- [x] 【21】：动画与模拟（基本概念、质点弹簧系统、运动学）
- [x] 【22】：动画与模拟（求解常微分方程，刚体与流体）

- **GAMES101**
  - 目录
  - 计算机图形学概述
    - What is CG?
    - Why study CG?
      - Applications
      - Fundamental Intellectual Challenges
      - Technical Challenges
    - Course Topics
  - 向量与线性代数
    - 向量
    - 矩阵
  - 变换（二维与三维）

- Why study transformation
- 2D transformations: rotation, scale, shear
  - Scale
  - Reflection
  - Shear
  - Rotate
- Homogeneous coordinates (齐次坐标)
  - Affine(仿射) Transformations
  - Scale
  - Rotation
  - Translation
- Other transforms
  - Inverse transforms
- Composing transforms
- Decomposing Complex Transforms
- 3D transformations

变换（模型、视图、投影）

- 3D transformations
  - Scale
  - Translation
  - Rotation
- Viewing (观测) transformation
  - View (视图) / Camera transformation
  - Projection transformation
    - Orthographic (正交) projection
    - Perspective (透视) projection

光栅化（三角形的离散化）

- Finishing up Viewing
  - Perspective Projection
  - Viewport (视口) transformation
- Rasterization
  - Different raster displays
  - Rasterizing a triangle

光栅化（深度测试与抗锯齿）

- Antialiasing (走样)
  - Sampling theory
  - Antialiasing in practice

着色（光照与基本着色模型）

- Visibility / occlusion
  - Z-buffering
- Shading (着色)
  - Illumination & Shading
    - Blinn-Phong Reflectance Model
      - Diffuse Reflection

着色（着色频率、图形管线、纹理映射）

- Shading
  - Blinn-Phong reflectance model
    - Diffuse Term
    - Specular Term
    - Ambient Term (环境光)
  - Shading frequencies
    - Shade each triangle (flat shading)
    - Shade each vertex (Gouraud shading)
    - Shade each pixel (Phong shading)
  - Graphics (Real-time Rendering) Pipeline
  - Texture mapping

着色（插值、高级纹理映射）

- Shading

- Barycentric coordinates
- Texture queries
  - Mipmap

几何（基本表示方法）

- Shading
  - Applications of textures
    - Many applications
- Introduction to geometry
  - Various representations of geometry
    - “Implicit” Representations of Geometry
    - “Explicit” Representations of Geometry
    - More Implicit Representations

几何（曲线与曲面）

- Various representations of geometry
  - Explicit Representations
- Curves
  - Bezier curves (贝塞尔曲线)
    - De Casteljau’s algorithm
    - Algebraic Formula
    - Properties of Bezier Curves
    - Piecewise (逐段) Bezier Curves
  - B-splines
- Surfaces
  - Bezier surfaces
    - Evaluating Bezier Surfaces
  - Mesh Operations: Geometry Processing

几何（网格处理）、阴影图

- Mesh Operations: Geometry Processing
  - Mesh subdivision
    - Loop Subdivision
    - Catmull-Clark Subdivision
  - Mesh simplification
    - Edge Collapse (边坍缩)
  - Mesh regularization
- Shadow Mapping
  - Pass 1: Render from Light
  - Pass 2A: Render from Eye
  - Pass 2B: Project to light

光线追踪（基本原理）

- Why Ray Tracing?
- Basic Ray-Tracing Algorithm
  - Overview
    - Light Rays
    - Ray Casting
    - Recursive (Whitted-Style) Ray Tracing
  - Details
    - Ray-Surface Intersection (交点)
    - Accelerating Ray-Surface Intersection

光线追踪（加速结构）

- Using AABBs to accelerate ray tracing
  - Uniform grids
  - Spatial partitions
    - KD-Tree
  - Object Partitions & Bounding Volume Hierarchy (BVH)
- Basic radiometry (辐射度量学)
  - Motivation
  - Radiant Energy and Flux (Power)
    - Radiant Intensity

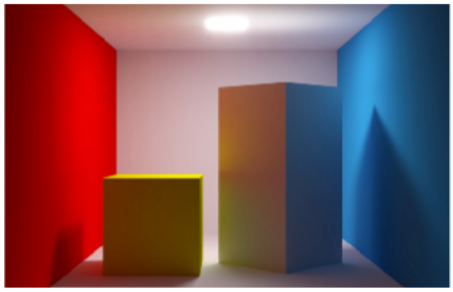
- 光线追踪（辐射度量学、渲染方程与全局光照）
  - Radiometry
    - Irradiance
    - Radiance
    - Irradiance vs. Radiance
  - Light transport
    - Bidirectional Reflectance Distribution Function (BRDF)
    - The Reflection Equation
    - The Rendering (渲染) Equation
  - Global illumination
    - Understanding the rendering equation
  - Probability review
    - Function of a Random Variable
- 光线追踪（蒙特卡洛积分与路径追踪）
  - Monte Carlo Integration
  - Path Tracing
    - A Simple Monte Carlo Solution
    - Introducing Global Illumination
      - Problem 1: Explosion of #rays as #bounces go up
      - Problem 2: The recursive algorithm will never stop!
      - Sampling the Light (importance sampling)
  - Some Side Notes
    - Ray tracing: Previous vs. Modern Concepts
    - Further Topic
- 材质与外观
  - Material == BRDF
    - Diffuse / Lambertian Material
    - Glossy material
    - Ideal reflective / refractive material
    - Microfacet (微表面) Material
    - Isotropic / Anisotropic Materials (BRDFs)
  - Properties of BRDFs
  - Measuring BRDFs
- 高级光线传播与复杂外观建模
  - Advanced Light Transport
    - Biased vs. Unbiased Monte Carlo Estimators
    - Bidirectional Path Tracing (BDPT)
    - Metropolis Light Transport (MLT)
    - Photon Mapping
    - Vertex Connection and Merging
    - Instant Radiosity (IR)
  - Advanced Appearance Modeling
    - Participating Media
    - Hair Appearance
    - Fur Appearance
    - Granular Material
    - Translucent Material
    - Cloth
    - Detailed material (non-statistical BRDF)
    - Procedural Appearance (程序生成)
- 相机与透镜
  - 相机
    - Pinhole Image Formation
    - Field of View (FOV) (视场)
    - Exposure (曝光度)
      - ISO (Gain)
      - F-Number (F-Stop): Exposure Levels
      - Shutter Speed

- 透镜
  - Thin Lens Approximation
  - The Thin Lens Equation
    - Defocus Blur
    - Ray Tracing Ideal Thin Lenses
    - Depth of Field
- 光场、颜色与感知
  - Light Field / Lumigraph (光场)
    - The Plenoptic Function (全光函数)
    - Light Field / Lumigraph (光场)
    - Light Field Camera
  - Color
    - Physical Basis of Color
    - What is color
    - Biological Basis of Color
    - Tristimulus Theory of Color
    - Metamerism (同色异谱)
    - Color Reproduction / Matching
    - Color space
    - Perceptually Organized Color Spaces
- 动画与模拟（基本概念、质点弹簧系统、运动学）
  - Introduction to Computer Animation
    - Keyframe (关键帧) animation
    - Physical simulation
      - Mass Spring System (质点弹簧系统)
      - Particle Systems (粒子系统)
    - Kinematics (运动学)
      - Forward Kinematics
      - Inverse Kinematics
    - Rigging
    - The Production Pipeline
- 动画与模拟（求解常微分方程，刚体与流体）
  - Single particle simulation
    - Explicit Euler method
    - Instability and improvements
      - Midpoint Method
      - Adaptive Step Size
      - Implicit Euler Method
      - Position-Based / Verlet Integration
  - Rigid body simulation
  - Fluid simulation
    - Eulerian vs. Lagrangian
    - Material Point Method (MPM)
- Advanced Topics

计算机图形学概述

Rasterization

Geometry



Light Transport

Animation / simulation

What is CG?

合成和操作视觉信息

Why study CG?

Applications

- Video Games

  怎么辨别好的画面：画面是不是足够亮（全局光照）
- Movies

  特效其实是CG里较为简单的部分，平常见的少，不会感觉不真实
- Animations
- Design (CAD)
- Visualization
- Virtual Reality

  虚拟现实纯虚拟，增强现实在现实里加入新的东西
- Digital Illustration
- Simulation

  物理引擎
- GUI
- Typography (字体)

  The Quick Brown Fox Jumps Over The Lazy Dog (包含26个字母)

Fundamental Intellectual Challenges

- Creates and interacts with realistic virtual world

  Require understanding of physical world (有足够的输入)

- New computing methods, displays, technologies

  形成新的东西，并且显示出来

Technical Challenges

- Math of (perspective) projections, curves, surfaces
- Physics of lighting and shading
- Representing / operating shapes in 3D
- Animation / simulation

Course Topics

- Rasterization (光栅化)

  Project geometry primitives (3D triangles / polygons) onto the screen

  广泛应用在实时图形学（30帧以上，否则叫offline）

- Curves and Meshes

  How to represent geometry, How to represent complicated geometry, How to handle transformation

- Ray Tracing

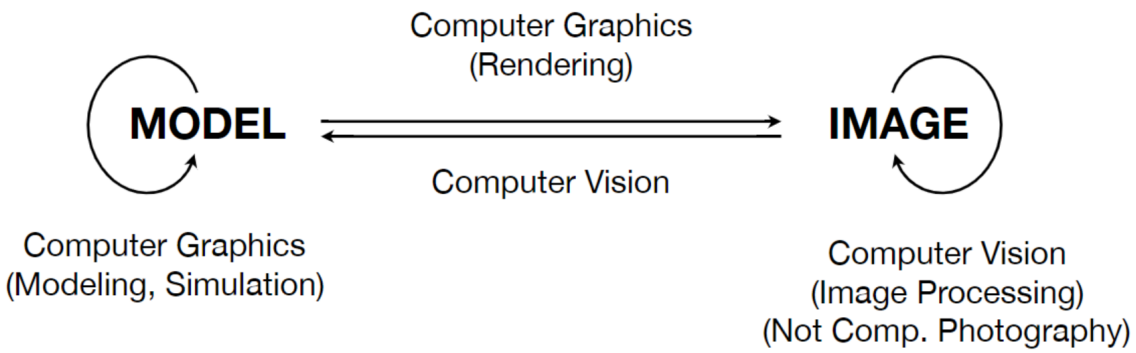
  广泛应用在动画和电影，tradeoff: 优异的效果，但是慢

  新技术：实时光线追踪

- Animation / Simulation

NOT about

- CG APIs
- 3D modeling, game development
- CV (猜测和理解图片)



But no clear boundaries.

向量与线性代数

Graphics' Dependencies

- Basic mathematics

  Linear algebra, calculus, statistics
- Basic physics

  Optics, Mechanics
- Misc

  Signal processing, Numerical analysis

向量

向量（默认为列向量）

符号$\boldsymbol{a}$，有方向和长度，平移不变

长度：$\|\boldsymbol{a}\|$

单位向量：$\hat{\boldsymbol{a}} = \boldsymbol{a}/\|\boldsymbol{a}\|$，表示方向不关心长度

向量加法

向量乘法

- Dot (scalar) product
  - 用来求向量夹角
  - 求一个向量在另一个向量的投影

$$\vec{b}_{\perp} = \|b\| \cos\theta \cdot \vec{a}$$
$$\vec{b}_{\|} = \vec{b} - \vec{b}_{\perp}$$

  - 计算两个方向接近程度
  - 判断前后
- Cross (vector) product

$$\vec{a} \times \vec{b} = A^{\star} b = \begin{pmatrix} 0 & -z_a & y_a \\ z_a & 0 & -x_a \\ -y_a & x_a & 0 \end{pmatrix} \vec{b}$$

  $A^{\star}$ is dual matrix of $\vec{a}$
  - 用于建立坐标系
  - 判断左/右
  - 判断内/外

    点和三角形的关系：假设ΔABC逆时针，P点分别在$\overrightarrow{AB}, \overrightarrow{BC}, \overrightarrow{CA}$左侧（可以推广到凸包）

正交坐标系

$$\|\vec{u}\| = \|\vec{v}\| = \|\vec{w}\| = 1$$
$$\vec{u} \cdot \vec{v} = \cdots = 0$$
$$\vec{w} = \vec{u} \times \vec{v} \text{ (right-handed)}$$
$$\vec{p} = (\vec{p} \cdot \vec{u})\vec{u} + (\vec{p} \cdot \vec{v})\vec{v} + (\vec{p} \cdot \vec{w})\vec{w}$$

矩阵

$A\vec{x}$表示变换 (transformation)，没有交换律

变换（二维与三维）

Why study transformation

- modeling 模型变换

  translation, rotation, scaling
- viewing 视图变换

  projection

2D transformations: rotation, scale, shear

note: 找变换前后的坐标对映关系（例如找(1,0), (0,1)变换之后的坐标）

Scale

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Reflection

horizontal reflection

$$\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$

Shear

Horizontal shift is 0 at y=0, Horizontal shift is a at y=1, Vertical shift is always 0

沿水平方向移动

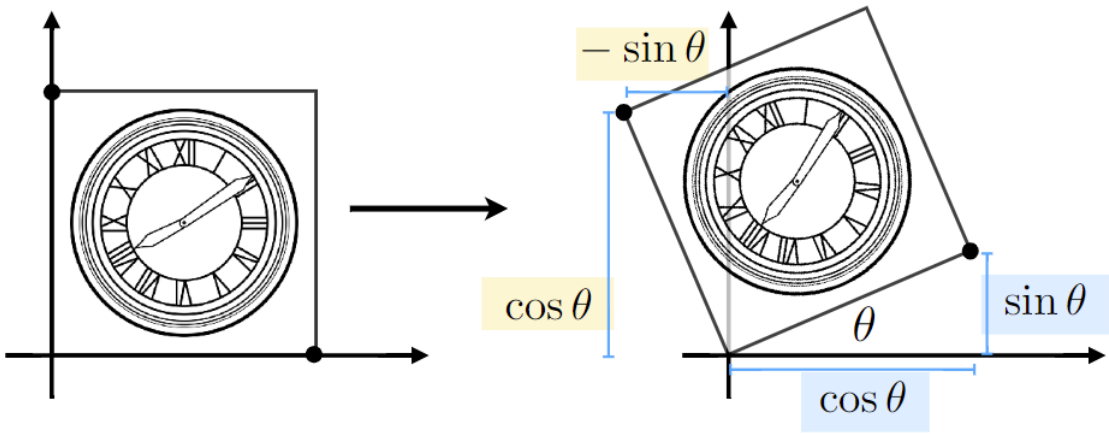
$$\begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix}$$

Rotate

关于原点，默认逆时针

$$(1,0)^\top \to (\cos\theta, \sin\theta)$$
$$(0,1)^\top \to (-\sin\theta, \cos\theta)$$

Rotation Matrix



$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$R_{-\theta} = R_\theta^{-1} = R_\theta^\top$$

Linear Transforms

$$\mathbf{x}' = \mathbf{M}\mathbf{x}$$

Homogeneous coordinates (齐次坐标)

Translation: 不是线性变换

Q: Is there a unified way to represent all transformations?(and what's the cost?)

Solution: Homogenous Coordinates

point = $(x, y, 1)^\top$, vector = $(x, y, 0)^\top$

Matrix representation of translations

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + t_x \\ y + t_y \\ 1 \end{pmatrix}$$

note: 向量具有平移不变性

Valid operation if w-coordinate of result is 1 or 0

- vector + vector = vector
- point – point = vector
- point + vector = point
- point + point = ??

In homogeneous coordinates,

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} \text{ is the 2D point } \begin{pmatrix} x/w \\ y/w \\ 1 \end{pmatrix}, \ w \neq 0$$

note: point + point = 中点

Affine(仿射) Transformations

Affine map = linear map + translation

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

note: 最下面必然是$(0, 0, 1)$

Scale

$$\mathbf{S}\left(s_x, s_y\right) = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Rotation

$$\mathbf{R}(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Translation

$$\mathbf{T}\left(t_x, t_y\right) = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

Other transforms

Inverse transforms

$\mathbf{M}^{-1}$ is the inverse of transform $\mathbf{M}$ in both a matrix and geometric sense

Composing transforms

note: Transform Ordering Matters! (矩阵乘法不可交换)

note: matrices are applied right to left (变换顺序从右到左)

$$A_n\left(\ldots A_2\left(A_1(\mathbf{x})\right)\right)=\mathbf{A}_n \cdots \mathbf{A}_2 \cdot \mathbf{A}_1 \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$\mathbf{A}=\mathbf{A}_n \cdots \mathbf{A}_2 \cdot \mathbf{A}_1$$

obtain a single matrix representing combined transform

Decomposing Complex Transforms

How to rotate around a given point c?

1. Translate center to origin
2. Rotate
3. Translate back

$$\mathbf{T}(\mathbf{c}) \cdot \mathbf{R}(\alpha) \cdot \mathbf{T}(-\mathbf{c})$$

3D transformations

类比2D变换

point = $(x, y, z, 1)^\top$, vector = $(x, y, z, 0)^\top$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}=\begin{pmatrix} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

note: 最下面是$(0, 0, 0, 1)$

note: 上述形式表示的是先线性变换后平移

变换（模型、视图、投影）

3D transformations

Scale

$$\mathbf{S}\left(s_x, s_y, s_z\right)=\begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

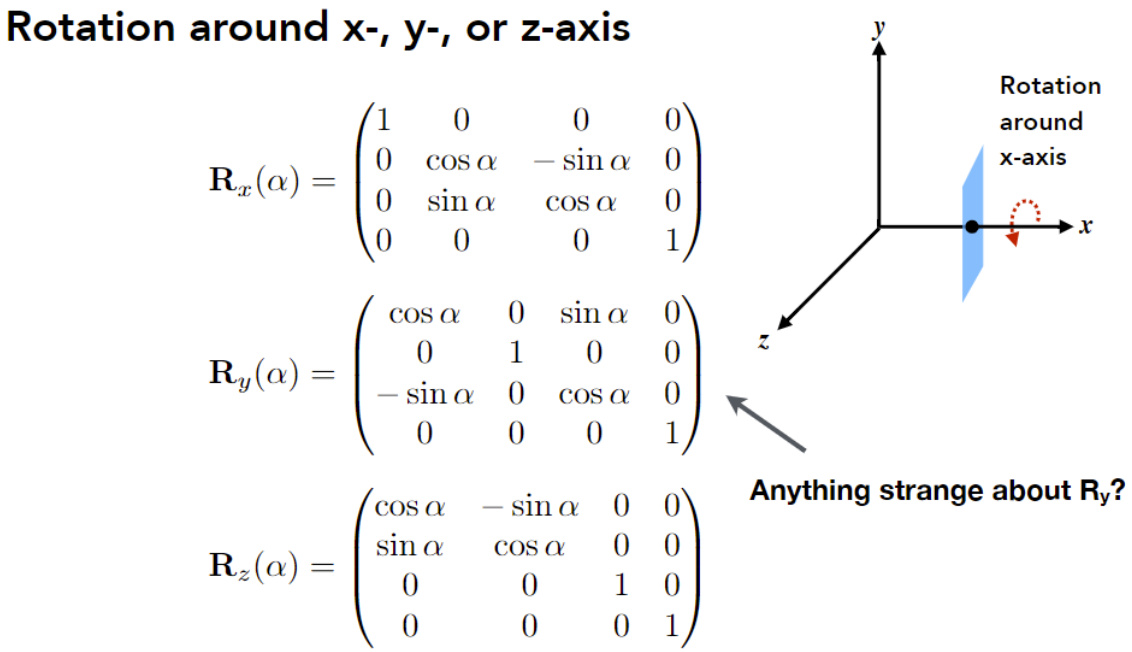
Translation

$$\mathbf{T}\left(t_x, t_y, t_z\right)=\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation

绕坐标轴旋转

note: $\vec{z} \times \vec{x} = \vec{y}$，把z当作2D情况的x，把x当作2D情况的y，所以结果是反的



一般的旋转

简单的旋转可以组合成复杂的旋转

$$R_{xyz}(\alpha,\beta,\gamma)=R_x(\alpha)R_y(\beta)R_z(\gamma)$$

其中α,β,γ称为欧拉角

Rodrigues' Rotation Formula

Rotation by angle α around axis $\mathbf{n}$ (这里默认axis过原点)

$$\mathbf{R}(\mathbf{n},\alpha)=\cos(\alpha)\mathbf{I}+(1-\cos(\alpha))\mathbf{n}\mathbf{n}^T+\sin(\alpha)\underbrace{\begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix}}_{\mathbf{N}}$$

说明：

1. 设要旋转的向量为$\vec{s}$，将$\vec{s}$对于$\vec{n}$分解为$\vec{s}_\perp, \vec{s}_{||}$

2. 将$\vec{n}, \vec{s}_\perp, \vec{n} \times \vec{s}_\perp$单位化后建立新的坐标系，记为$\vec{a}, \vec{b}, \vec{c}$
3. $\vec{s}_\perp$在旋转后会发生变化，而$\vec{s}_\parallel$不会

   $\vec{s}_\perp$的旋转完全在$\vec{b}, \vec{c}$平面内

$$\mathbf{s}_\perp^{\text{rot}} = \left\|\vec{s}_\perp\right\| \cos\alpha \cdot \vec{b} + \left\|\vec{s}_\perp\right\| \sin\alpha \cdot \vec{c}$$
$$\mathbf{s} = \mathbf{s}_\perp^{\text{rot}} + \mathbf{s}_\parallel$$

沿任意轴：平移回原点，旋转，重新平移回去

note: 四元数可以用于旋转之间插值（例如旋转15°和旋转25°中间插值，20°不等于矩阵求平均）

Viewing (观测) transformation

最终目的：3D物体变成2D照片

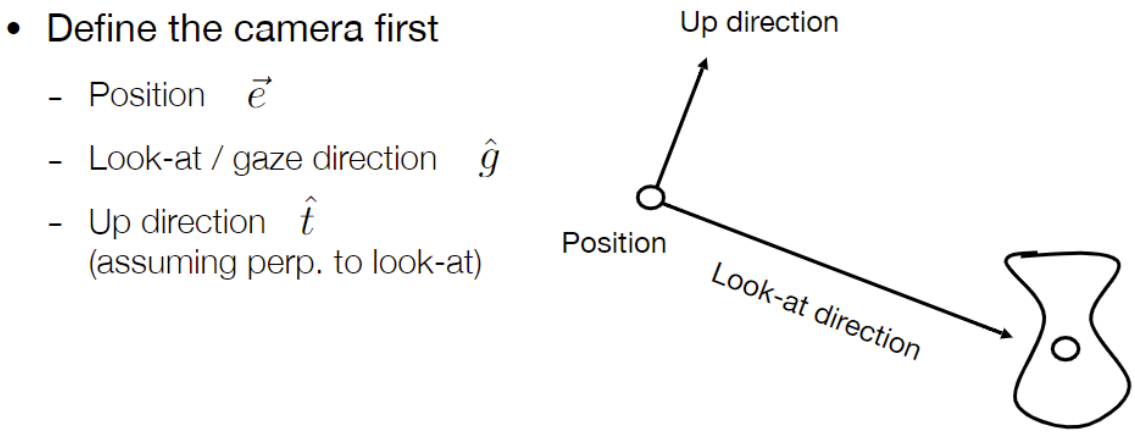
A: Think about how to take a photo

- Find a good place and arrange people (model transformation, 把场景建好)
- Find a good “angle” to put the camera (view transformation, 改变相机属性)
- Cheese! (projection transformation)

note: MVP变换

View (视图) / Camera transformation

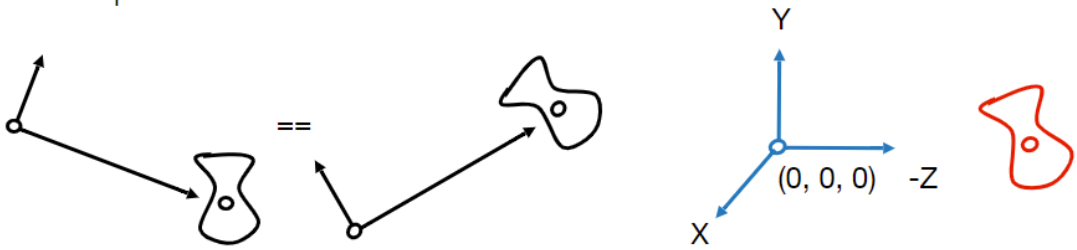
摆照相机



相机和物体的位置是相对的。不妨把相机固定在原点，朝向-z方向，以y轴为向上方向，而对其它所有物体做变换

Transform objects together with the camera

- Key observation
  - If the camera and all objects move together, the “photo” will be the same



- How about that we always transform the camera to
  - The origin, up at Y, look at -Z
  - And transform the objects along with the camera

如何把相机变换到标准位置?

1. Translates e to origin
2. Rotates g to -Z
3. Rotates t to Y

如何写成矩阵形式?

- M_{view} in math?
  - Let’s write $M_{view} = R_{view}T_{view}$
  - Translate e to origin

$$T_{view} = \begin{bmatrix} 1 & 0 & 0 & -x_e \\ 0 & 1 & 0 & -y_e \\ 0 & 0 & 1 & -z_e \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  - Rotate g to -Z, t to Y, (g x t) To X
  - Consider its inverse rotation: X to (g x t), Y to t, Z to -g

$$R_{view}^{-1} = \begin{bmatrix} x_{\hat{g}\times\hat{t}} & x_t & x_{-g} & 0 \\ y_{\hat{g}\times\hat{t}} & y_t & y_{-g} & 0 \\ z_{\hat{g}\times\hat{t}} & z_t & z_{-g} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \xrightarrow{\text{WHY?}} R_{view} = \begin{bmatrix} x_{\hat{g}\times\hat{t}} & y_{\hat{g}\times\hat{t}} & z_{\hat{g}\times\hat{t}} & 0 \\ x_t & y_t & z_t & 0 \\ x_{-g} & y_{-g} & z_{-g} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

note: 把$(1,0,0),(0,1,0),(0,0,1)$分别带入R^{-1}可以检验正确性

note: R_{view}^{-1}是正交矩阵（正交变换）

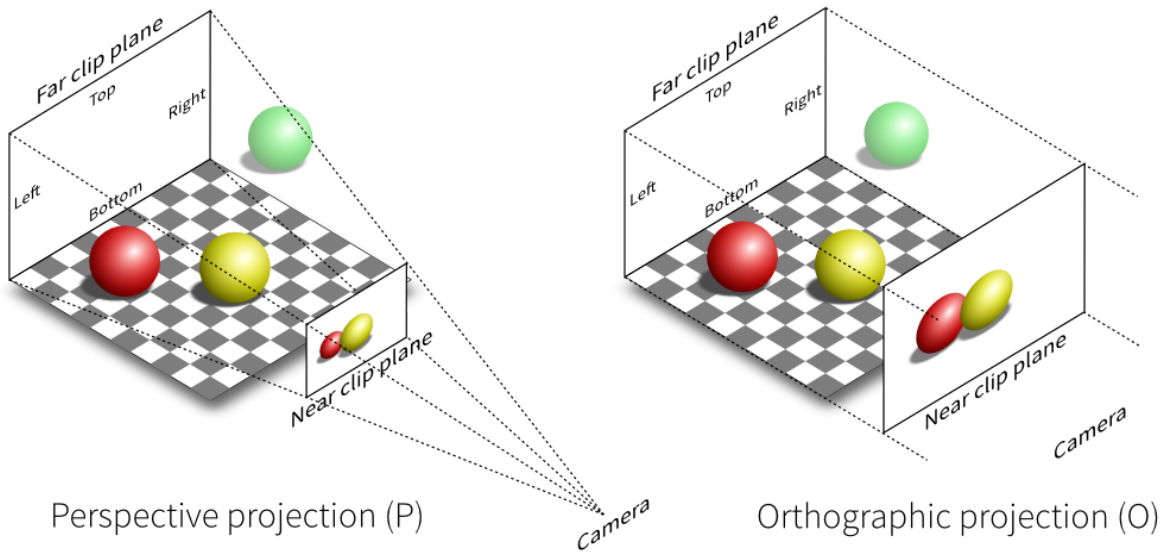
note: 和model transformation类似，M_{view}也应用在物体上

Projection transformation

正交投影后，平行线还是平行的

透视投影带来近大远小的现象

正交投影可理解为把点摄像机移到无穷远处



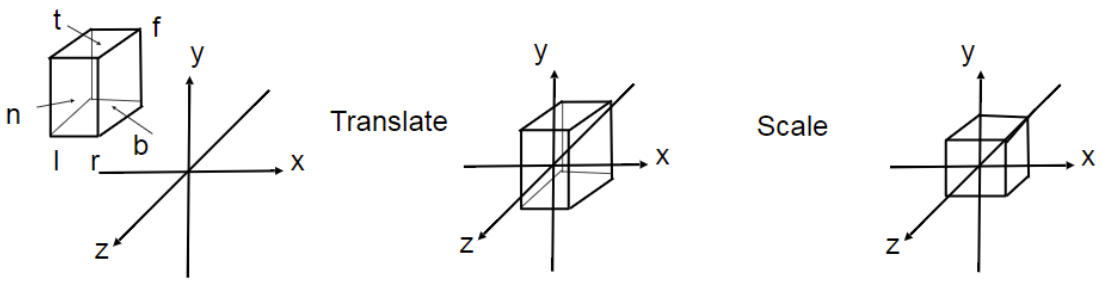
Orthographic (正交) projection

A simple way of understanding

- Camera located at origin, looking at -Z, up at Y
- **Drop** Z coordinate (Q：丢掉z后怎么区分物体的前后？)
- Translate and scale the resulting rectangle to $[-1, 1]^2$ (约定俗成做法)

实际做法

- In general
  - We want to map a cuboid [l, r] x [b, t] x [**f, n**] to the "canonical (正则、规范、标准)" cube $[-1, 1]^3$



note: left/right, bottom/top, near/far

note: 沿-z方向看，数值上n>f (因此某些API会使用左手系)

先平移中心到原点，后缩放

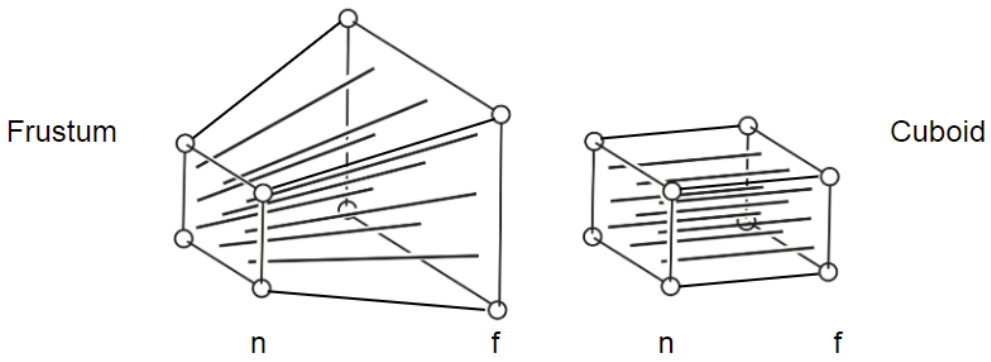
$$M_{\text{ortho}} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & \frac{2}{n-f} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -\frac{r+l}{2} \\ 0 & 1 & 0 & -\frac{t+b}{2} \\ 0 & 0 & 1 & -\frac{n+f}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Q：考虑旋转?

Perspective (透视) projection

Most common in Computer Graphics, art, visual system

- How to do perspective projection
  - First “squish” the frustum into a cuboid (n -> n, f -> f) ($M_{\text{persp->ortho}}$)
  - Do orthographic projection (M_{ortho}, already known!)



note: projection就是把线（光线）在f的点投影到在n的点，而透视投影中线会收缩

squish过程的规定：n平面不变，f平面z值不变，f平面中心点不变

In order to find a transformation

Find the relationship between transformed points (x′, y′, z′) and the original points (x, y, z)

- 先考虑x和y

  从侧面看

$$x = \frac{x_0}{n} z,\ x' = x_0 \Rightarrow x' = \frac{n}{z} x$$

$$y' = \frac{n}{z} y \qquad x' = \frac{n}{z} x$$

- In homogeneous coordinates,

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \Rightarrow \begin{pmatrix} nx/z \\ ny/z \\ \text{unknown} \\ 1 \end{pmatrix} \overset{\text{mult. by z}}{==} \begin{pmatrix} nx \\ ny \\ \text{still unknown} \\ z \end{pmatrix}$$

- So the “squish” (persp to ortho) projection does this

$$M_{persp\to ortho}^{(4\times 4)}\begin{pmatrix}x\\y\\z\\1\end{pmatrix}=\begin{pmatrix}nx\\ny\\\text{unknown}\\z\end{pmatrix}$$

- Already good enough to figure out part of $M_{\text{persp->ortho}}$

$$M_{persp\to ortho}=\begin{pmatrix}n&0&0&0\\0&n&0&0\\?&?&?&?\\0&0&1&0\end{pmatrix}$$

WHY?

- 考虑z坐标
  - Any point on the near plane will not change
  - Any point’s z on the far plane will not change

- Any point on the near plane will not change

$$M_{persp\to ortho}^{(4\times 4)}\begin{pmatrix}x\\y\\z\\1\end{pmatrix}=\begin{pmatrix}nx\\ny\\\text{unknown}\\z\end{pmatrix}\xrightarrow{\text{replace z with n}}\begin{pmatrix}x\\y\\n\\1\end{pmatrix}\Rightarrow\begin{pmatrix}x\\y\\n\\1\end{pmatrix}==\begin{pmatrix}nx\\ny\\n^2\\n\end{pmatrix}$$

- So the third row must be of the form (0 0 A B)

$$\begin{pmatrix}0&0&A&B\end{pmatrix}\begin{pmatrix}x\\y\\n\\1\end{pmatrix}=n^2$$

n^2 has nothing to do with x and y

带入两个条件后可得，

$$An+B=n^2$$
$$Af+B=f^2$$

解得$A=n+f, B=-nf$

$M_{\text{persp}}=M_{\text{ortho}}M_{\text{persp}\to\text{ortho}}$

note: 在squish过程中，中间物体的z坐标会发生变化（变远）

Q：为什么中间物体会变远?

注意z为负数

$$z'=n+f-\frac{nf}{z}\le z\quad(z\in[n,f])$$

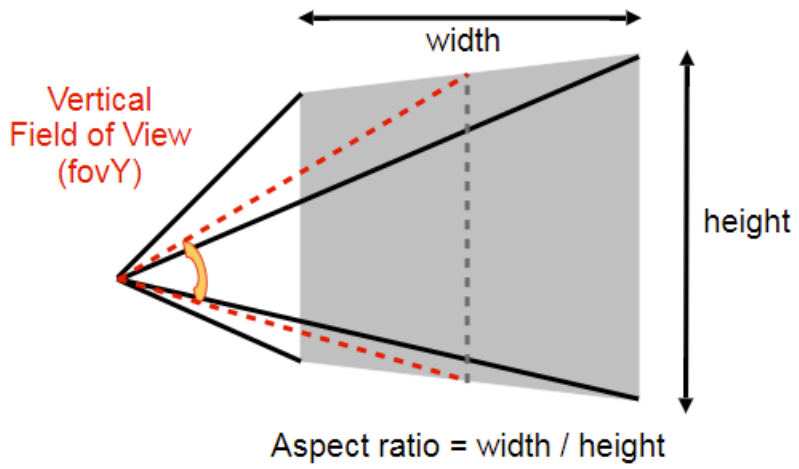
光栅化（三角形的离散化）

Finishing up Viewing

Perspective Projection

Q：如何定义视锥?

可以使用vertical field-of-view (fovY)和aspect ratio来表示 (assume symmetry i.e. l = -r, b = -t)



$$\tan\frac{fovY}{2} = \frac{t}{|n|}$$

$$aspect = \frac{r}{t}$$

Viewport (视口) transformation

观测变换的下一步

What's after MVP?

- Model transformation (placing objects)
- View transformation (placing camera)
- Projection transformation
  - Orthographic projection (cuboid to "canonical" cube $[-1, 1]^3$)
  - Perspective projection (frustum to "canonical" cube)

Canonical Cube to Screen

What is a **screen**?

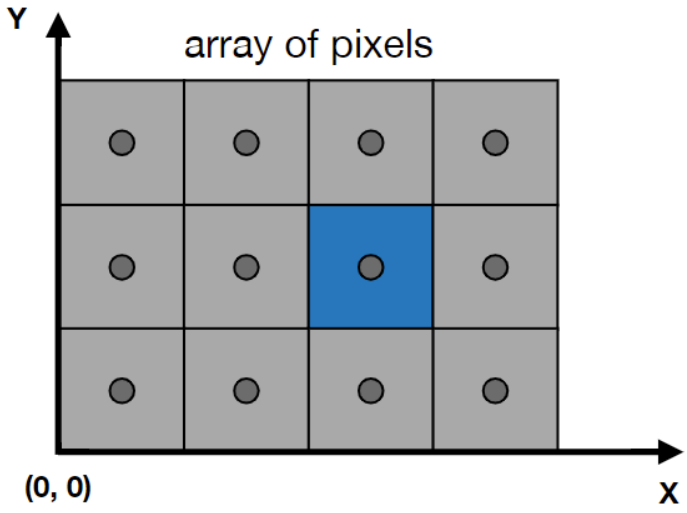
- An array of pixels
- Size of the array: resolution
- A typical kind of raster display (光栅成像)

Pixel: For now, a pixel is a little square with uniform color (目前假设为最小单位)

screen space

在屏幕建立坐标系

- Defining the screen space
  - Slightly different from the “tiger book”



Pixels’ indices are in the form of (x, y), where both x and y are integers

Pixels’ indices are from (0, 0) to (width - 1, height - 1)

Pixel (x, y) is centered at (x + 0.5, y + 0.5)

The screen covers range (0, 0) to (width, height)

Canonical Cube to Screen

- Irrelevant to z (Q：之后如何处理z？)
- Transform in xy plane: $[-1, 1]^2$ to [0, width] x [0, height]
- Viewport transform matrix

  缩放后平移到第一象限

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

现在已经可以得到照片了，下一步把结果变成像素显示出来

Rasterization

Drawing to Raster Displays

Different raster displays

Frame Buffer: memory for a raster display

- Oscilloscope (示波器)
- Cathode Ray Tube (显像管)

  隔行扫描：一次只画奇数行，一次只画偶数行（视觉暂留）

  可以应用于压缩，但会造成画面撕裂
- Flat Panel Displays

  LCD (液晶)：液晶的旋光性
- Electrophoretic (Electronic Ink) Display

  不同电压控制颜色，刷新率很低

Rasterizing a triangle

Triangles - Fundamental Shape Primitives

- Most basic polygon
  - Break up other polygons
- Unique properties
  - Guaranteed to be planar (四边形不一定是平面的)
  - Well-defined interior (里面没有洞，一定是凸的)
  - Well-defined method for interpolating values at vertices over triangle (barycentric interpolation)

    只要定义顶点的属性，其它任何点都可以插值

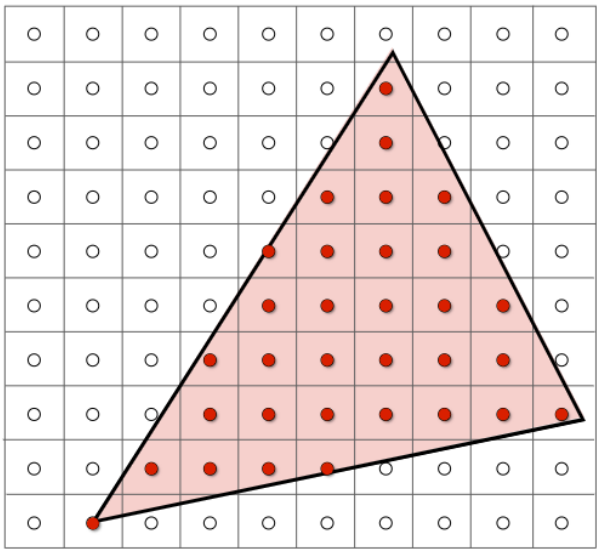
What Pixel Values Approximate a Triangle?

A Simple Approach: **Sampling**

Evaluating a function at a point is sampling. (函数离散化)

用像素中心对屏幕空间采样

Sample If Each Pixel Center Is Inside Triangle



$$\text{inside}(tri, x, y) = \begin{cases} 1 & (x, y) \in tri \\ 0 & otherwise \end{cases}$$

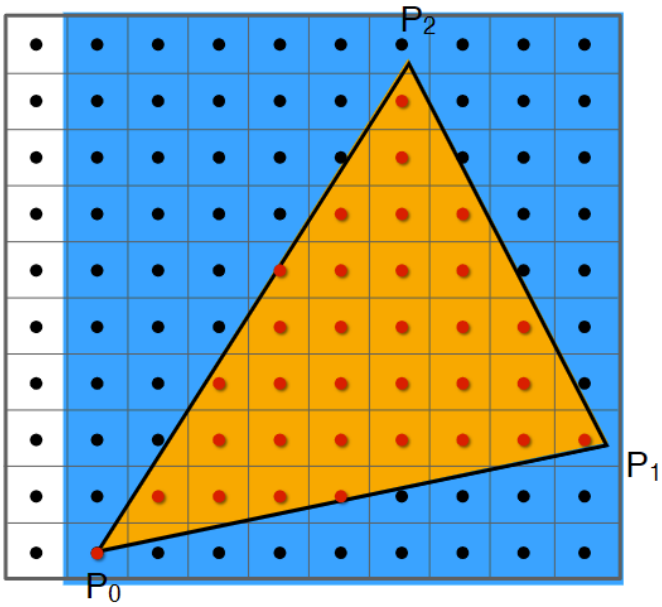
Evaluating inside(tri, x, y)

Inside? Recall: Three Cross Products!

Edge Cases: 可以自己定义

通过bounding box减少计算量

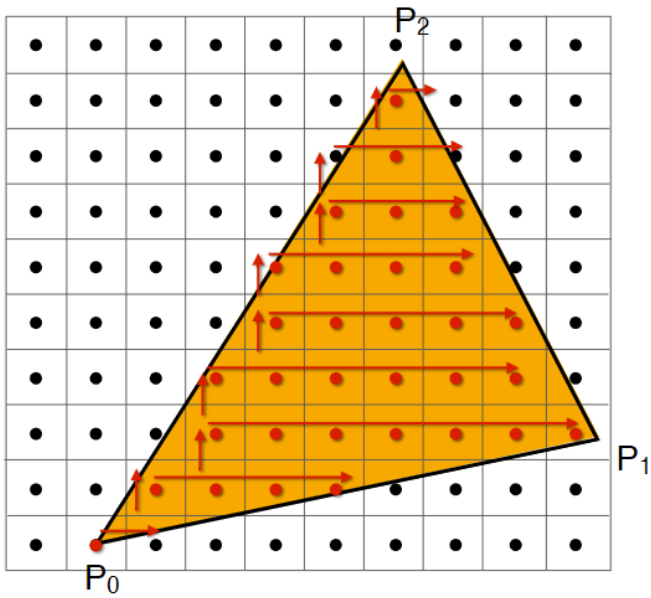
Checking All Pixels on the Screen?



Use a **Bounding Box!**

当三角形很瘦，并且呈45°角的情况下，AABB效果不理想

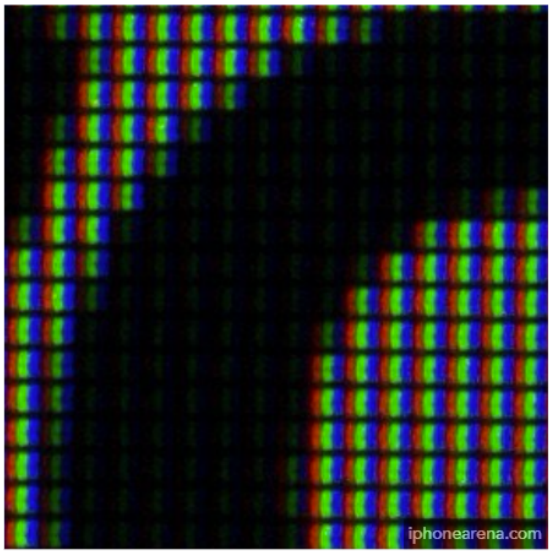
Incremental Triangle Traversal (Faster?)



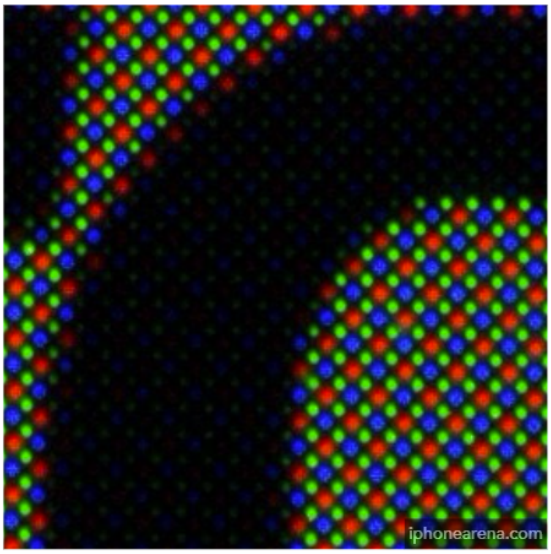
suitable for thin and rotated triangles

Rasterization on Real Displays

Real LCD Screen Pixels (Closeup)



iPhone 6S



Galaxy S5

Notice R,G,B pixel geometry! But in this class, we will assume a colored square full-color pixel.

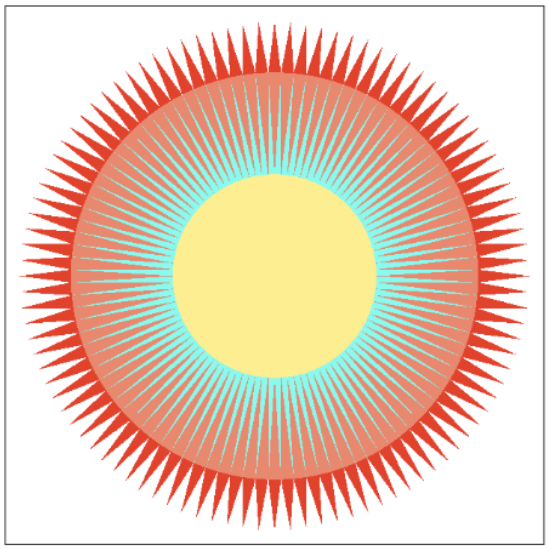
LCD pixels do not actually emit light in a square of uniform color.

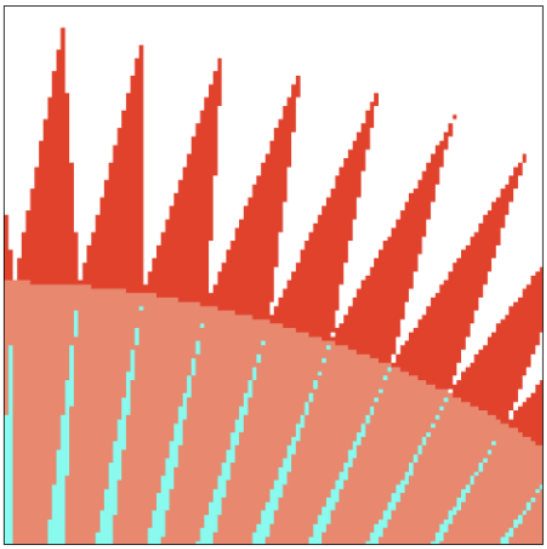
右侧显示可以看到绿点更多（人眼对绿色更敏感）

Aliasing (走样)

像素本身有一定大小，采样率对信号来说不够高，从而出现锯齿

Aliasing (Jaggies)





光栅化（深度测试与抗锯齿）

Antialiasing (走样)

一个三角形是某种连续的在屏幕空间的信号

Testing in/out △ at pixels' center

Pixels are uniformly-colored squares

Sampling theory

Sampling is Ubiquitous (广泛存在) in Computer Graphics

- Rasterization = Sample 2D Positions

  sampling in space
- Photograph = Sample Image Sensor Plane
- Video = Sample Time

Sampling Artifacts(Errors / Mistakes / Inaccuracies) in Computer Graphics

- Jaggies (锯齿) (Staircase Pattern)
- Moiré Patterns (摩尔纹) in Imaging

  cause: Skip odd rows and columns. e.g. 手机拍屏

  undersampling images
- Wagon Wheel Illusion (False Motion)

  sampling in time

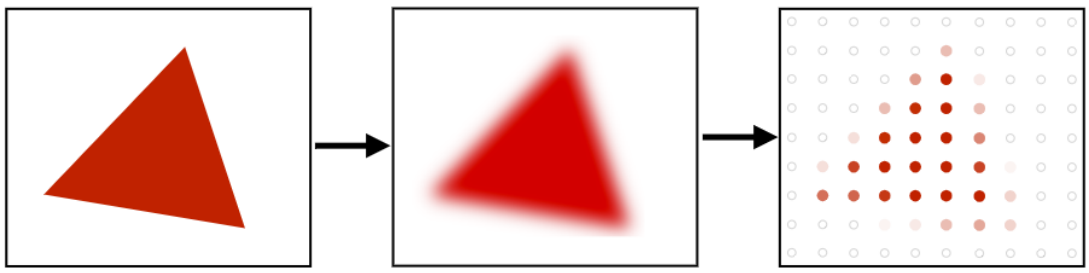
  人眼在时间中采样频率不够

Behind the Aliasing Artifacts

Signals are <u>changing too fast</u> (high frequency), but <u>sampled too slowly</u>

先给出一个结果

Antialiasing Idea: Blurring (Pre-Filtering) Before Sampling



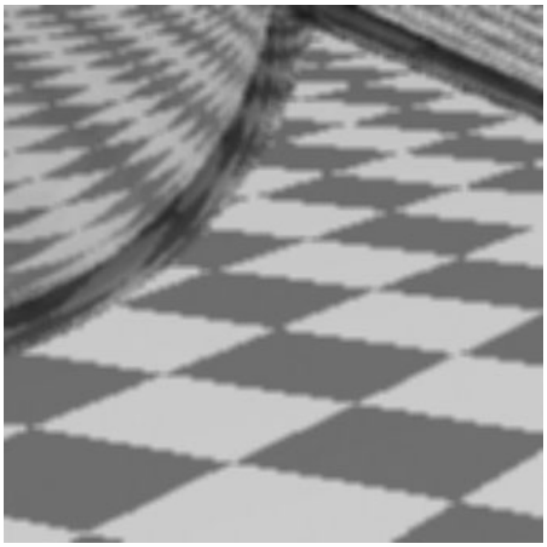
Pre-Filter

(remove frequencies above Nyquist) (?)

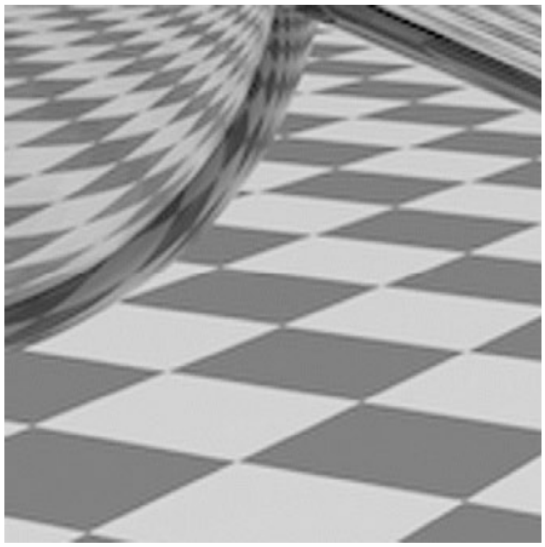
Sample

Note antialiased edges in rasterized triangle
where pixel values take intermediate values

Antialiasing vs Blurred Aliasing



(Sample then filter, WRONG!)



(Filter then sample)

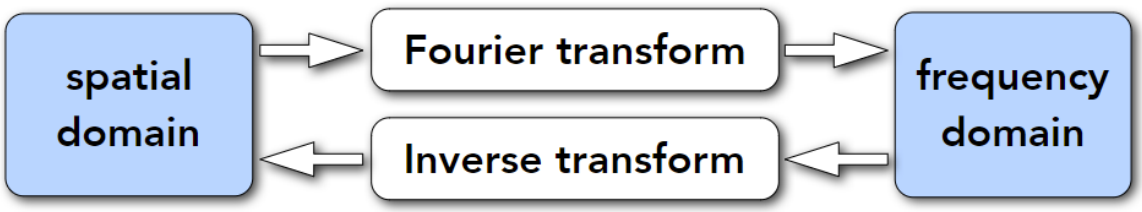
1. Why undersampling introduces aliasing?
2. Why pre-filtering then sampling can do antialiasing?
3. Why sample then filter can not work?

Frequency Domain

Fourier Transform: Represent a function as a weighted sum of sines and cosines

Fourier Transform Decomposes A Signal Into Frequencies

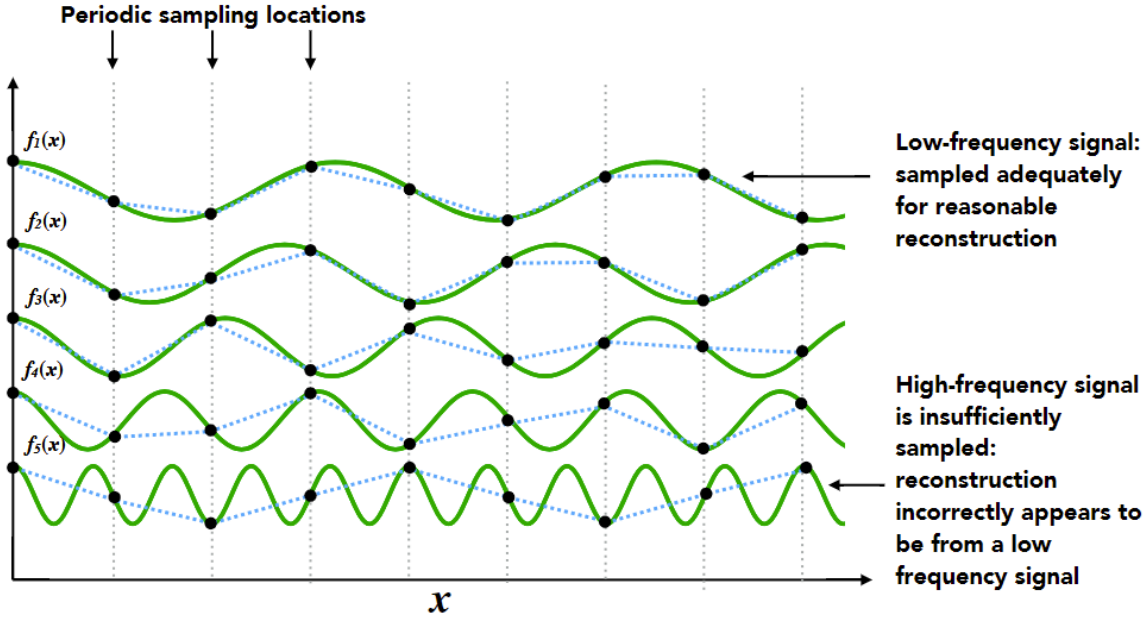
$$f(x) \qquad F(\omega) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i \omega x} dx \qquad F(\omega)$$



$$f(x) = \int_{-\infty}^{\infty} F(\omega) e^{2\pi i \omega x} d\omega$$

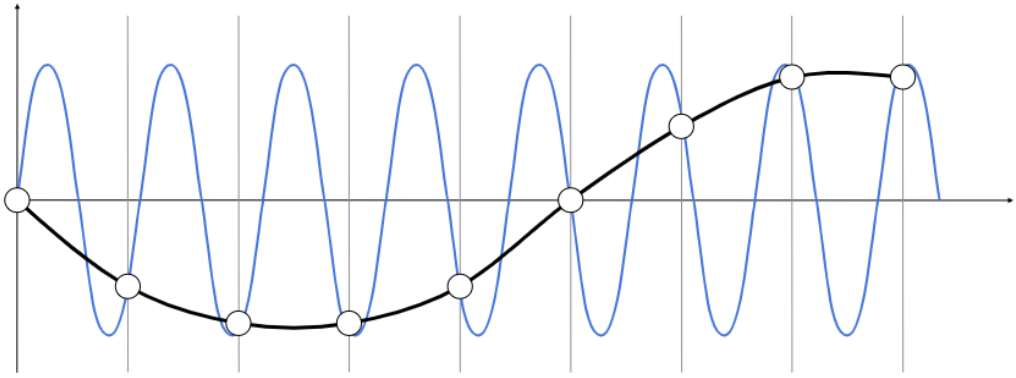
Recall $e^{ix} = \cos x + i \sin x$

Higher Frequencies Need Faster Sampling



不同频率的信号需要不同的采样频率，低频采样无法恢复出高频信号

Undersampling Creates Frequency Aliases



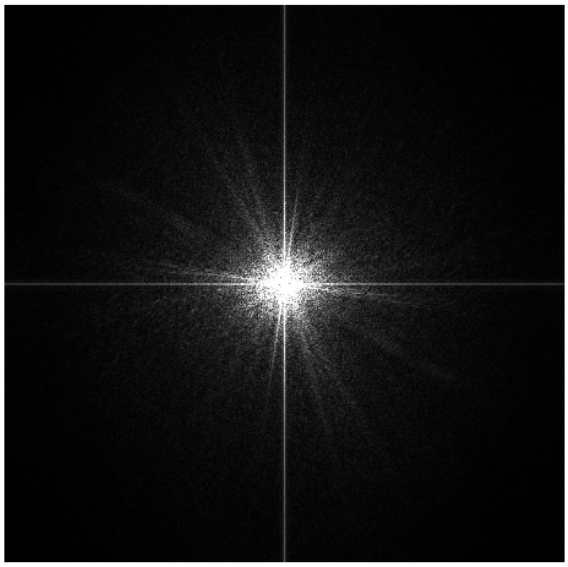
High-frequency signal is insufficiently sampled: samples erroneously appear to be from a low-frequency signal

Two frequencies that are indistinguishable at a given sampling rate are called “aliases”

走样：有两个频率完全不一样的函数，在低频采样后结果一致

Filtering (滤波) = Getting rid of certain frequency contents

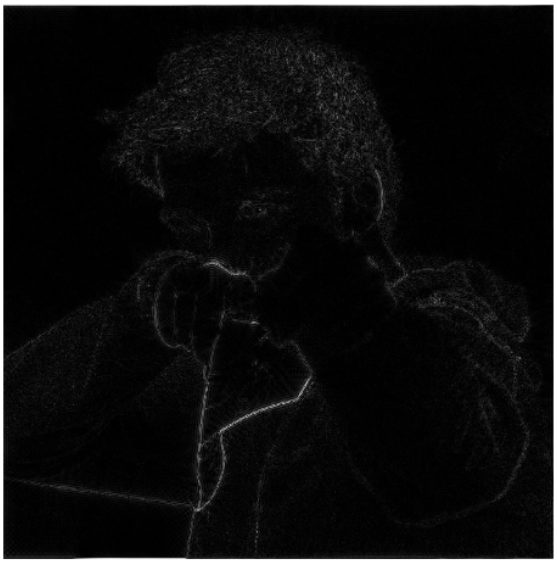
Visualizing Image Frequency Content

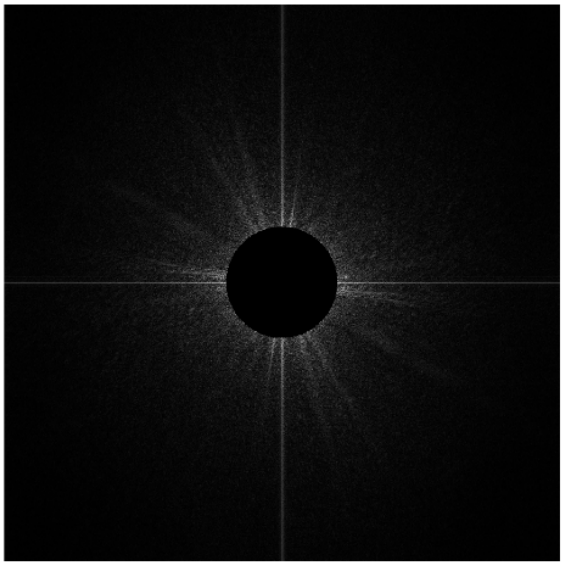


说明：

1. 图像的空间不同位置当作时间信息
2. 频率空间：中心低频，周围高频；亮度表示信息量
3. 自然图片一般大部分是低频信号
4. 水平和竖直线：图片没有周期性，把图片重复摆放形成周期信号，在相邻图片的边界上会有剧烈的信号变化

Filter Out Low Frequencies Only (Edges)





High-pass filter

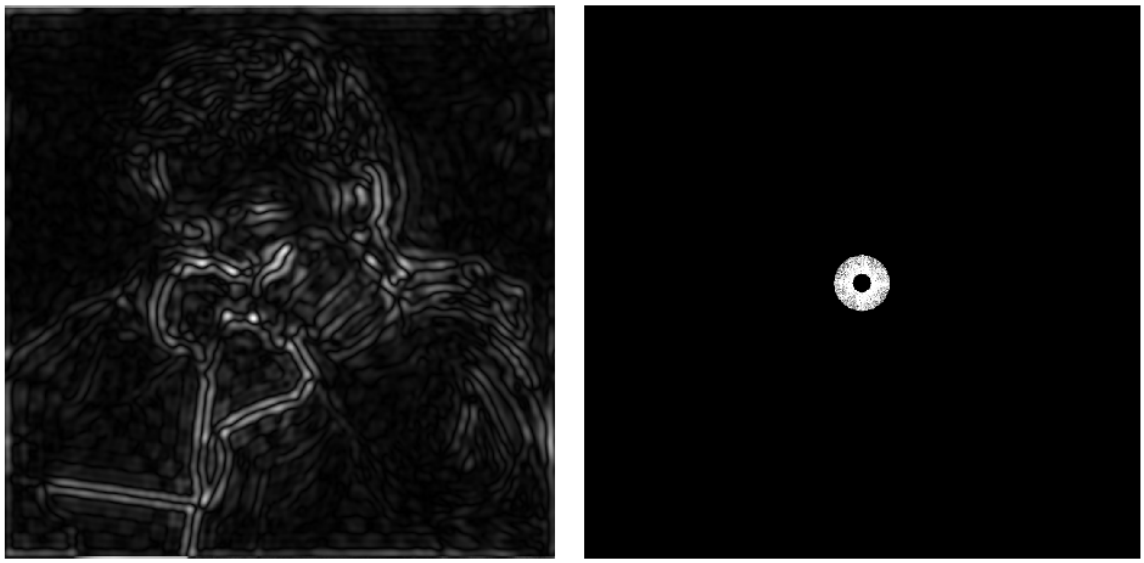
高通滤波：剩下高频信息，留下了图像的边界（边界发生突变）

Filter Out High Frequencies (Blur)

Low-pass filter

低通滤波：图像变模糊（去掉了边界）

Filter Out Low and High Frequencies



去掉高频和低频，提取到不是特别明显的边界（最明显的边界特征被去掉了，低频内容也被去掉了）

Filtering = **Convolution** (= Averaging)

Convolution: Point-wise local averaging in a “sliding window”

Convolution Theorem

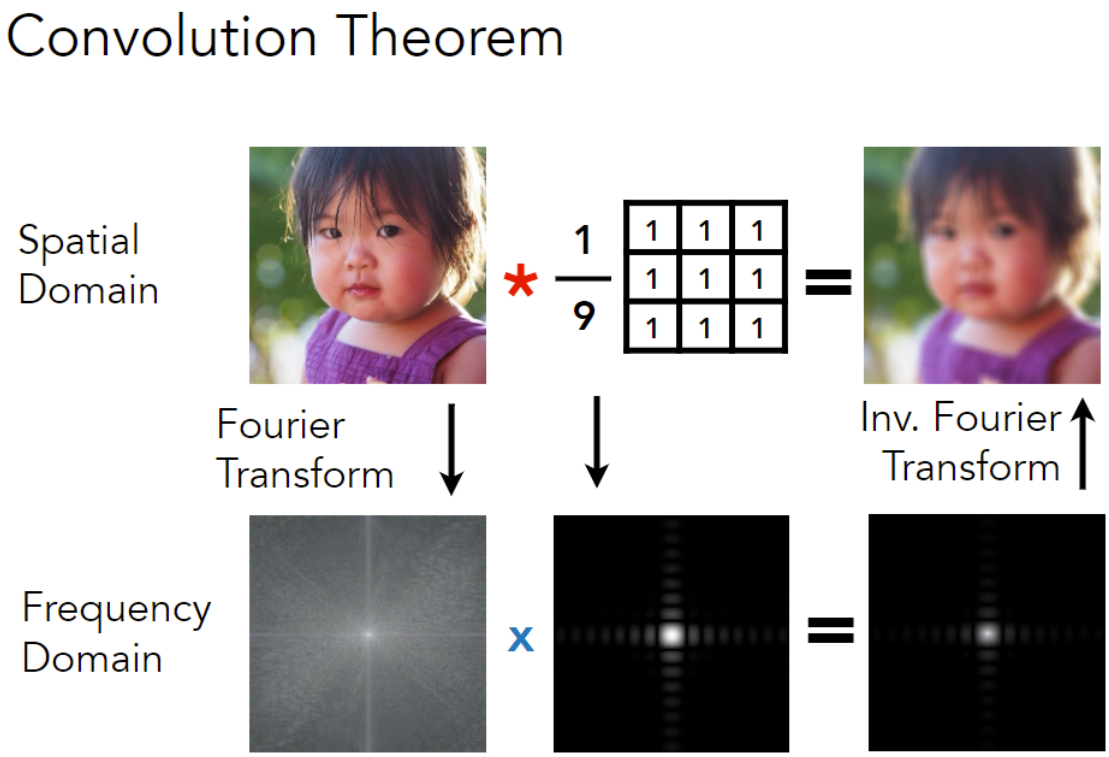
Convolution in the spatial domain is equal to multiplication in the frequency domain, and vice versa

Option 1:

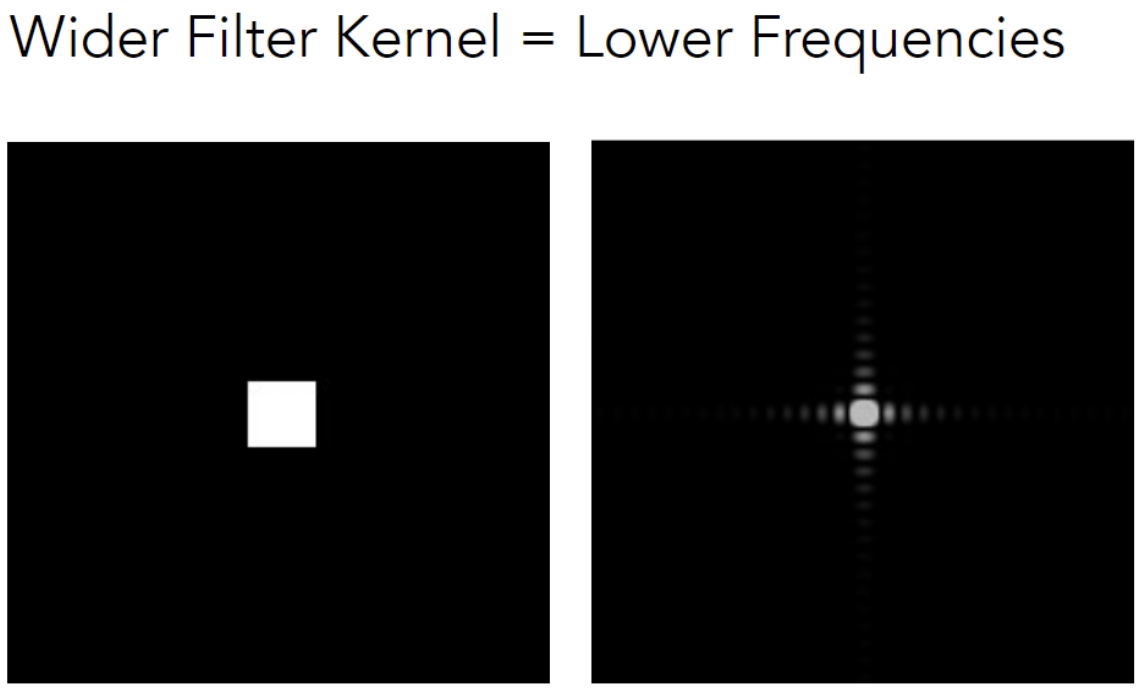
- Filter by convolution in the spatial domain

Option 2:

- Transform to frequency domain (Fourier transform)
- Multiply by Fourier transform of convolution kernel
- Transform back to spatial domain (inverse Fourier)

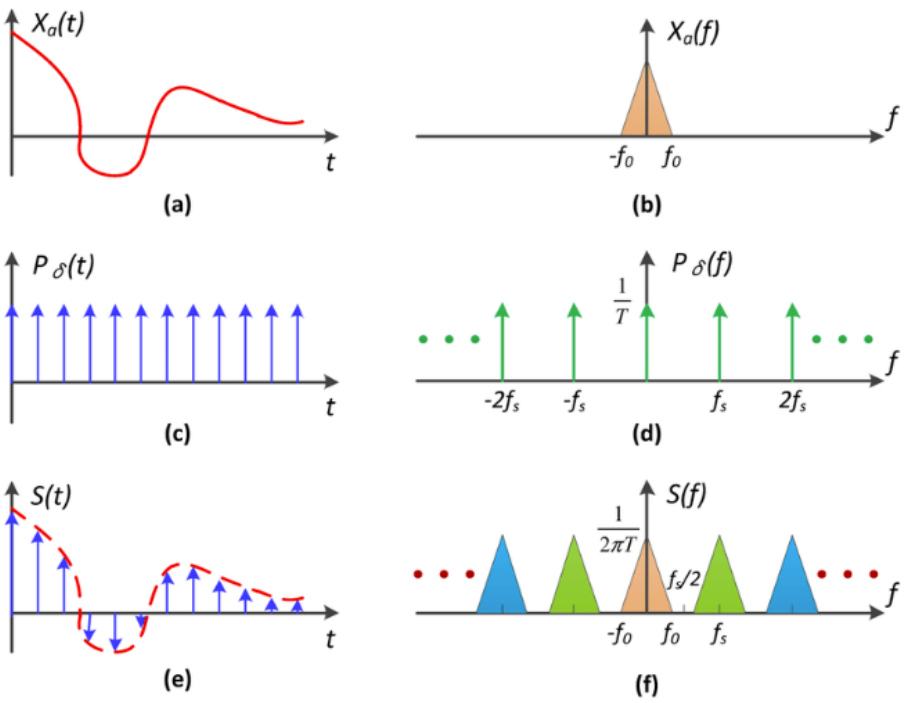


平均卷积核大部分集中在低频（周围有一点artifact），做平均卷积约等于低通滤波



更大的box filter，在频域反而变小了（留下更低的频率，更模糊）

Sampling = Repeating Frequency Contents

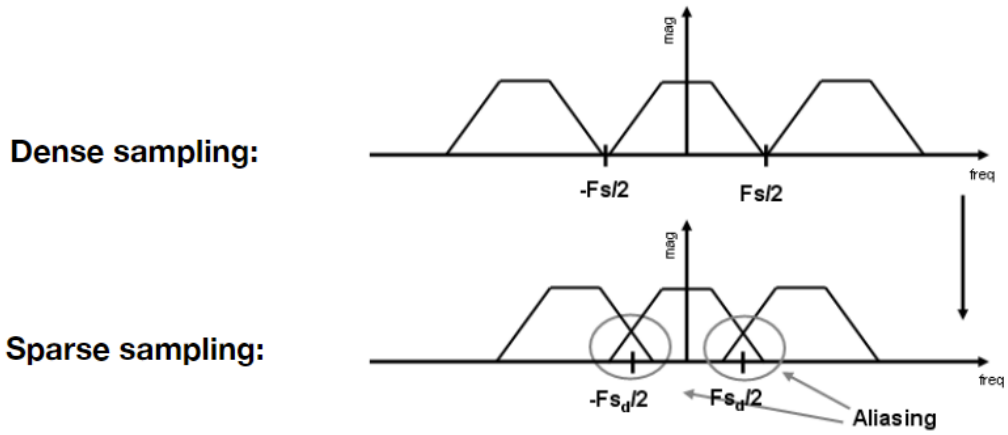


左侧为时域，右侧为频域。其中b,d分别为a,c在频域的相应的函数

采样 = 原函数和0-1指示函数（冲击函数）做乘积，等于在频域做卷积，f为b和d卷积的结果

采样在频域的操作：把原始函数在频域的像复制粘贴

Aliasing = Mixed Frequency Contents



采样不够快，导致原始信号复制粘贴间隔变小（时域和频域相反）

走样：频谱在复制粘贴后发生混叠

Antialiasing in practice

Option 1: Increase sampling rate

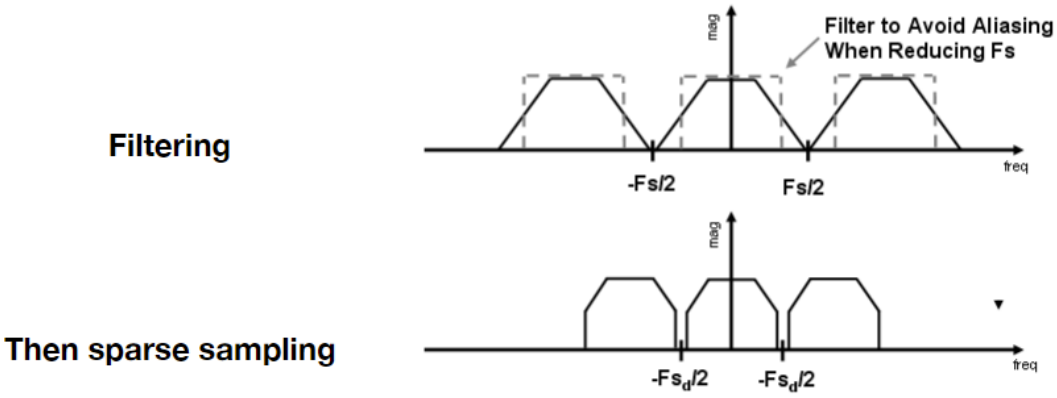
分辨率高像素多等价于采样频率高

- Essentially increasing the distance between replicas in the Fourier domain
- Higher resolution displays, sensors, framebuffers...
- But: costly & may need very high resolution

Option 2: Antialiasing

- Making Fourier contents “narrower” before repeating
- i.e. Filtering out high frequencies before sampling (先模糊后采样)

Antialiasing = Limiting, then repeating

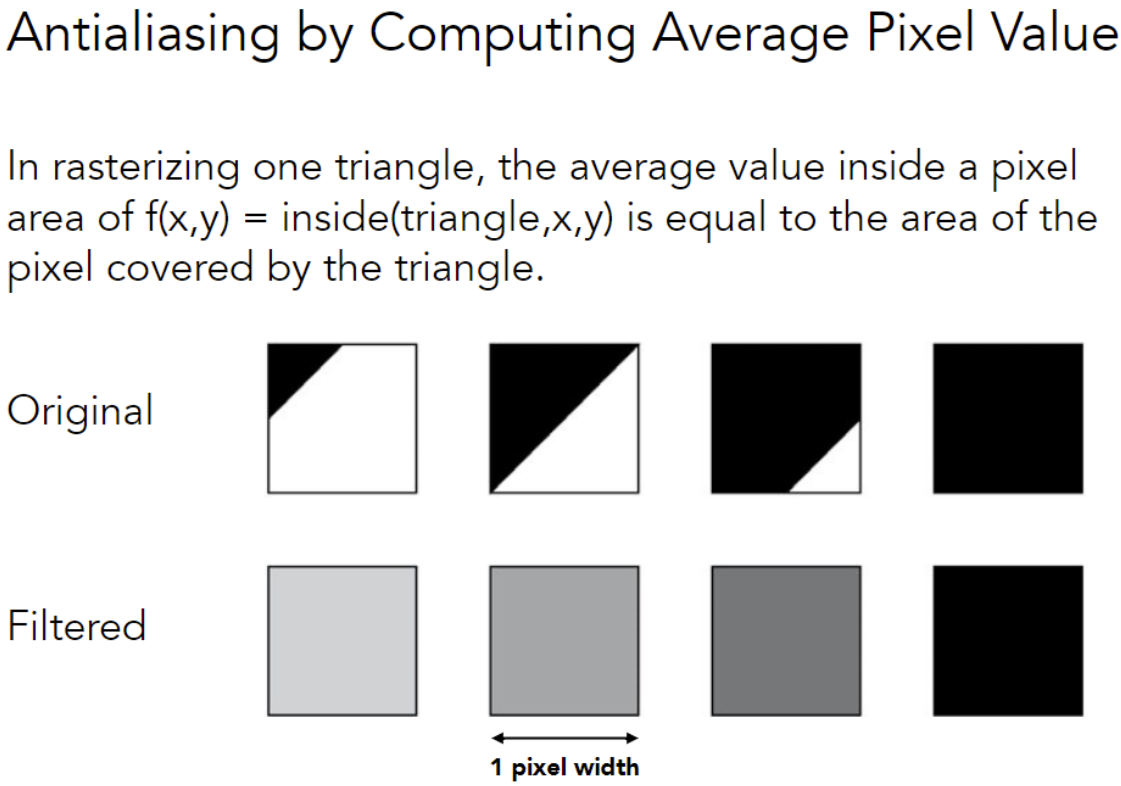


去掉高频信号后，剩下的信号不容易混叠

A Practical Pre-Filter: A 1 pixel-width box filter (low pass, blurring)

Solution:

1. Convolve f(x,y) by a 1-pixel box-blur
2. Then sample at every pixel’s center (用average即可)



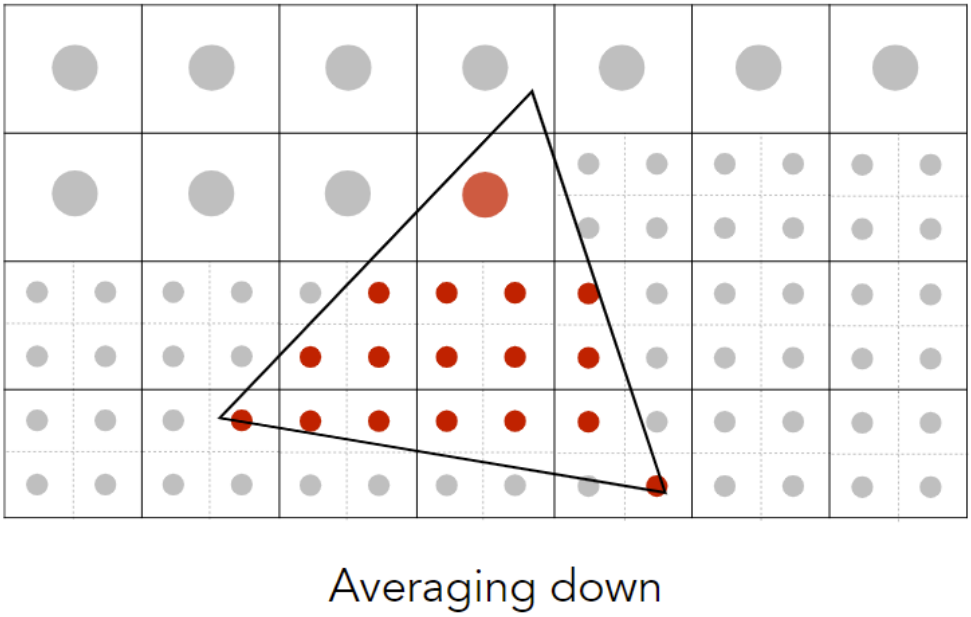
在每个pixel内对覆盖面积求平均

Antialiasing By Supersampling (MSAA)

Q：如何在每个pixel内求平均?

MSAA是通过更多采样来近似blur操作，采样隐含在过程中

Supersampling: 把每个原来的pixel划分成更小的pixel，在更小的pixel上采样后平均（在原来的每个pixel内sample多次）



注：MSAA不是靠提升分辨率（提高采样率）直接解决走样（超采样只是为了更好的模糊）

No free lunch! MSAA costs more time.

idea: random sampling, importance sampling

Milestones

- FXAA (Fast Approximate AA)

  有锯齿的图像后期处理，把锯齿边界找到并替换成没有锯齿的
- TAA (Temporal AA)

  找上一帧信息，每一帧使用不同的位置采样，复用上一帧的结果

  相当于把MSAA的样本分布在时间上

Super resolution (超分辨率) / super sampling

- From low resolution to high resolution
- Essentially still “not enough samples” problem
- DLSS (Deep Learning Super Sampling)

着色（光照与基本着色模型）

Q：为什么先采样再模糊是错的?

在信号混叠之后截断，但混叠的信号还是混叠的

Visibility / occlusion

绘制多个空间位置不同的三角形

Painter's Algorithm

解决绘制顺序

Inspired by how painters paint

Paint from back to front, overwrite in the framebuffer

Can have unresolvable depth order (不一定总是能够排序)

Z-buffering

把物体拆分成多块（拆分成pixel）

对每个pixel能看到的部分排序

Idea:

- Store current min. z-value for each sample (pixel)
- Needs an additional buffer for depth values
    - frame buffer stores color values
    - depth buffer (z-buffer) stores depth

IMPORTANT: For simplicity we suppose z is always positive (smaller z -> closer, larger z -> further)

```
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
```

Most important visibility algorithm. Implemented in hardware for all GPUs

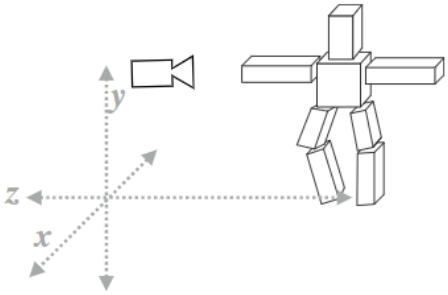
in MSAA: 对每个sample做z-buffer

note: 无法处理透明物体

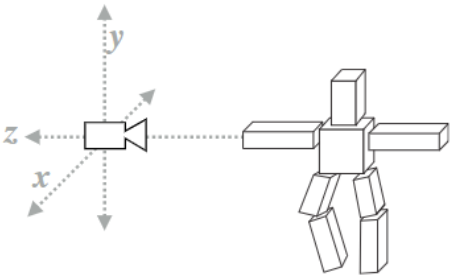
Shading (着色)

What We've Covered So Far

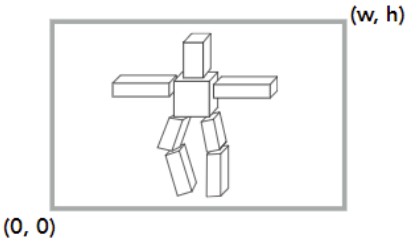
MVP + viewport + rasterization



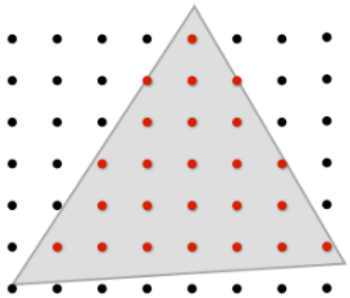
Position objects and the camera in the world



Compute position of objects relative to the camera



Project objects onto the screen



Sample triangle coverage

What Else Are We Missing?

无法反映出颜色根据光照的变化

Illumination & Shading

shading

The process of applying a material (材质) to an object.

不同材质和光线作用效果不同

Blinn-Phong Reflectance Model

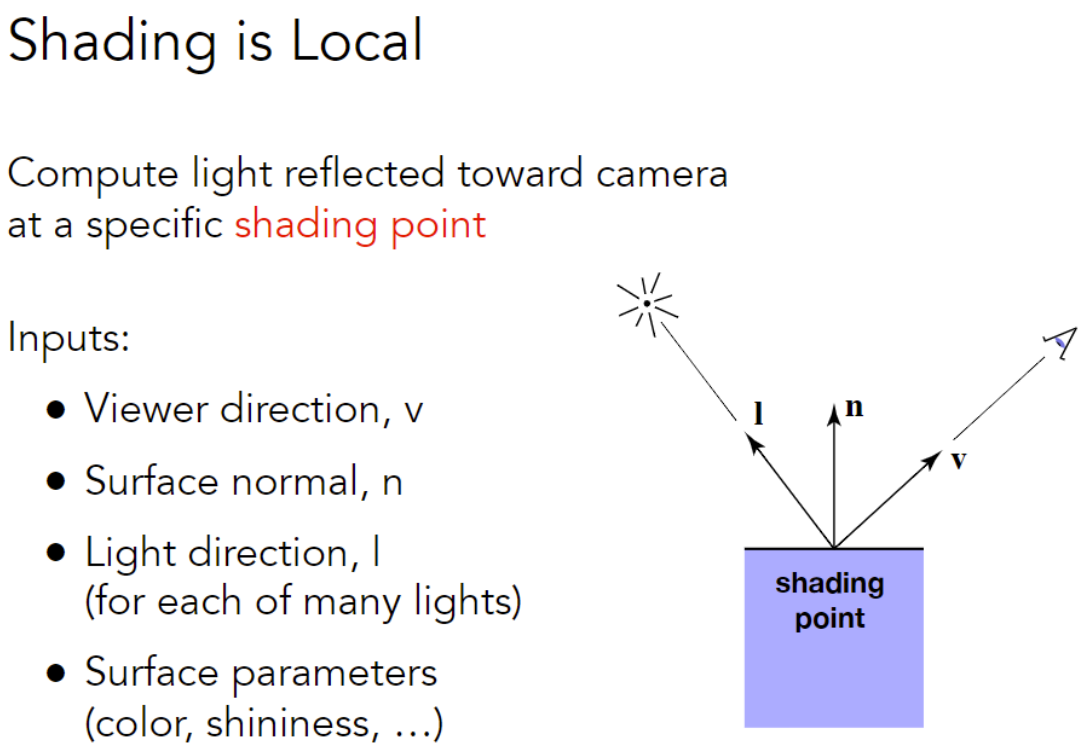
经验模型，不完全符合物理规律

Specular highlights (高光，镜面反射) + Diffuse reflection (漫反射) + Ambient lighting (环境光)

注：这里先假设每个点的Ambient lighting都是常量

把三部分光照分别计算

shading inputs



- 考虑某一个点(shading point)的着色结果
- $\vec{n}, \vec{v}, \vec{l}$仅表示方向，均为单位向量
- shading point有自己的属性

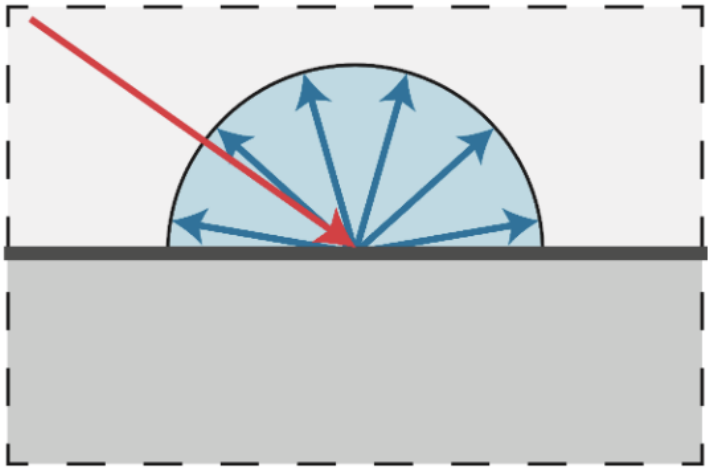
Shading is Local (局部性)

考虑点的着色仅考虑自己，不考虑光线被其它物体遮挡

No shadows will be generated! (shading ≠ shadow)

Diffuse Reflection

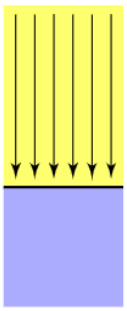
- Light is scattered uniformly in all directions
  - Surface color is the same for all viewing directions



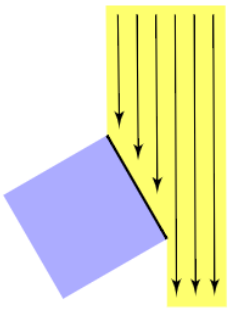
物体如何接收光

假设每根光线具有相同的能量

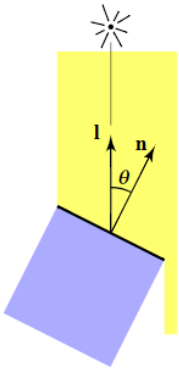
- But how much light (energy) is received?
  - Lambert's cosine law



Top face of cube receives a certain amount of light

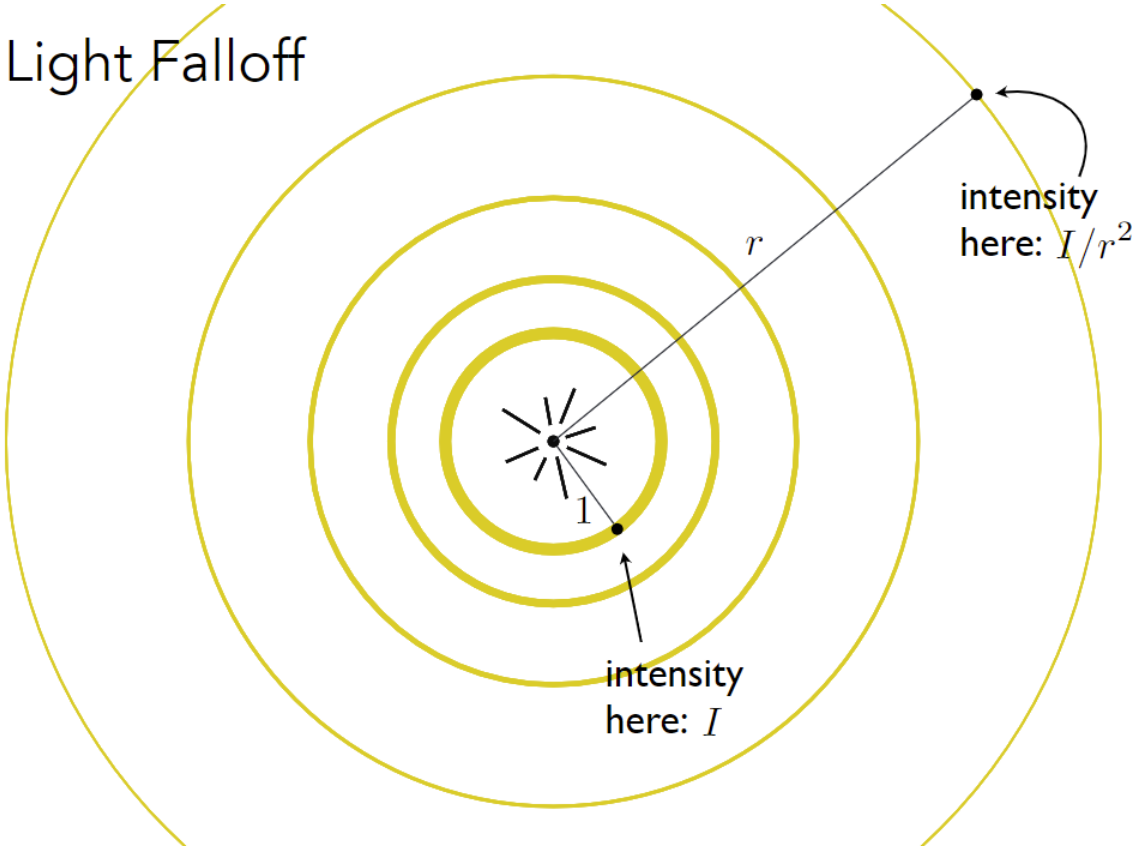


Top face of 60° rotated cube intercepts half the light



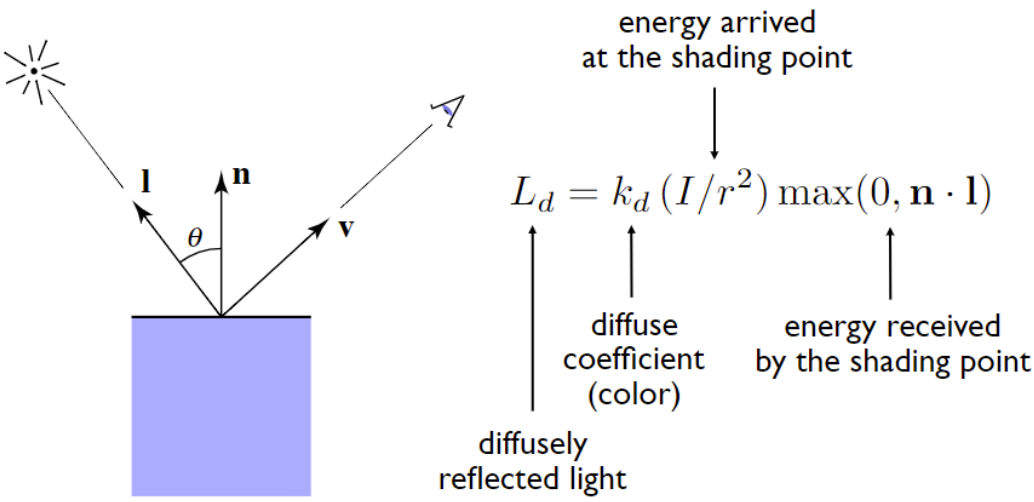
In general, light per unit area is proportional to $\cos\theta = l \cdot n$

光源如何发射光



Lambertian (Diffuse) Shading

Shading independent of view direction



- $\max(0, \vec{n}\cdot\vec{l})$去掉反方向的光
- k_d是shading point自身的参数，$k_d = 1$完全不吸收能量
- L_d和观测方向无关

着色（着色频率、图形管线、纹理映射）

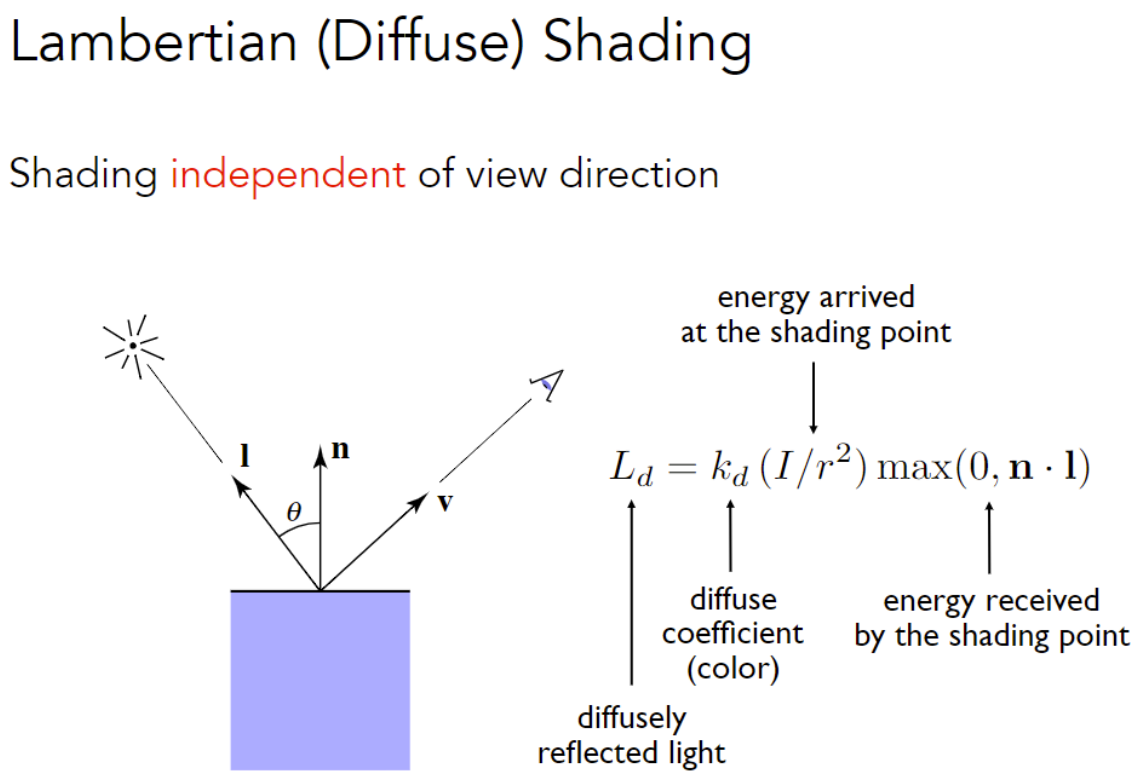
Shading

Blinn-Phong reflectance model

note: shading at a specific shading point (需要对所有的点进行操作)

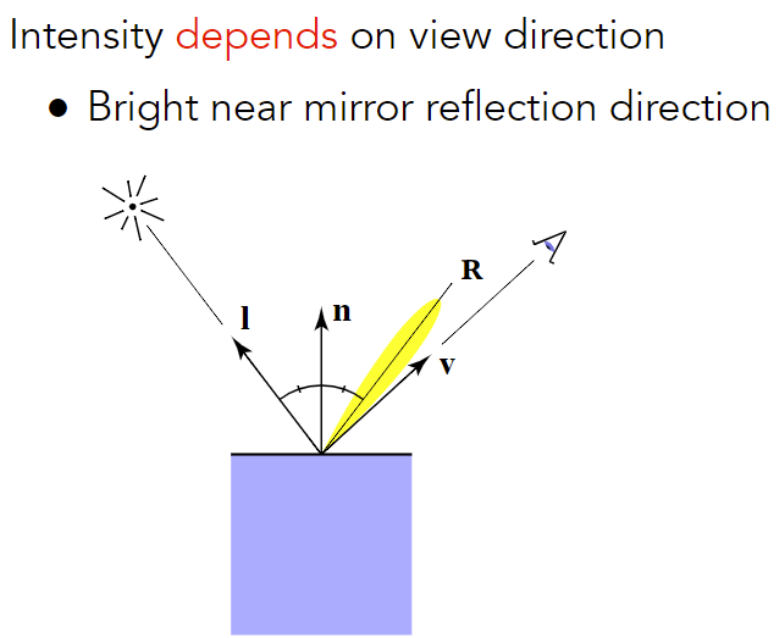
note: $\vec{l}, \vec{n}, \vec{v}$均为单位向量

Diffuse Term



Specular Term

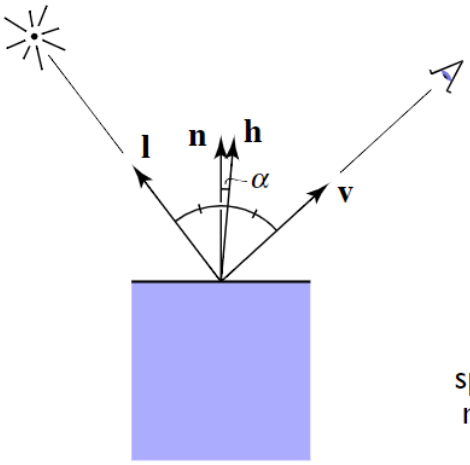
类似镜面反射，观测方向和镜面反射方向足够接近时产生高光



V close to mirror direction ⇔ half vector near normal

V close to mirror direction ⇔ **half vector** near normal

- Measure "near" by dot product of unit vectors



$$\mathbf{h} = \text{bisector}(\mathbf{v}, \mathbf{l})$$

(半程向量)

$$= \frac{\mathbf{v} + \mathbf{l}}{\|\mathbf{v} + \mathbf{l}\|}$$

$$L_s = k_s \, (I/r^2) \max(0, \cos\alpha)^p$$

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

specularly reflected light

specular coefficient

- 通常认为高光k_s是白的
- 简化模型，不考虑物体对光的吸收
- 半程向量比反射方向计算简单
- Increasing p narrows the reflection lobe (cos衰减太慢，一般p取100~200)

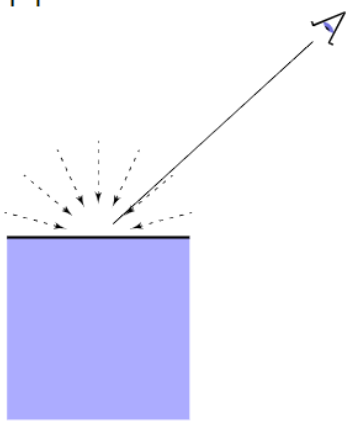
Ambient Term (环境光)

假设任何一个点接收到环境光是相同的

环境光与光照方向和观测方向无关，保证没有地方完全是黑的

Shading that does not depend on anything

- Add constant color to account for disregarded illumination and fill in black shadows
- This is approximate / fake!

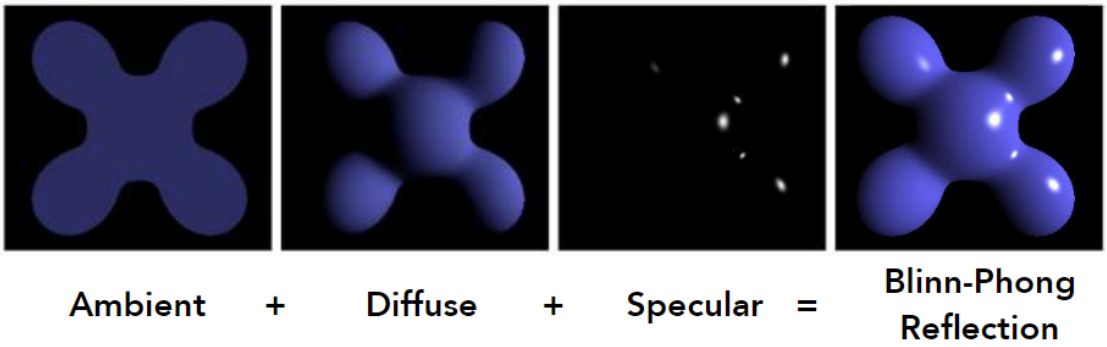


$$L_a = k_a \, I_a$$

ambient coefficient

reflected ambient light

Blinn-Phong Reflection Model



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

Shading frequencies

What caused the shading difference?

每个面着色 vs 每个顶点着色 + 插值 vs normal方向插值 + 每个pixel着色

Shade each triangle (flat shading)

平面着色

Flat shading

- Triangle face is flat — one normal vector
- Not good for smooth surfaces

Shade each vertex (Gouraud shading)

Q：怎么求顶点法线?

每个顶点着色，中间插值

Gouraud shading

- Interpolate colors from vertices across triangle
- Each vertex has a normal vector (how?)

Shade each pixel (Phong shading)

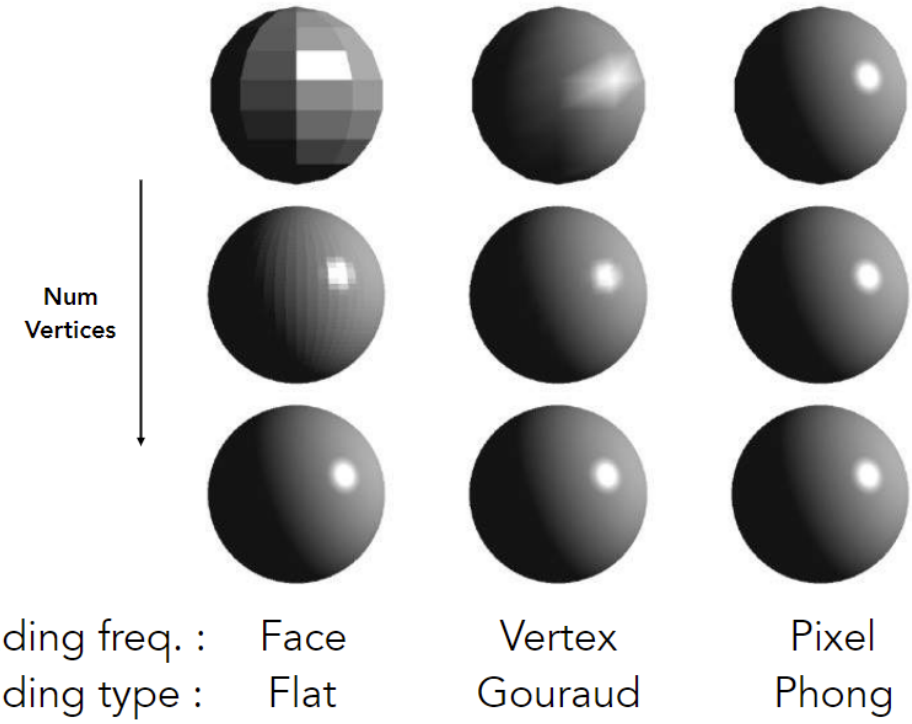
求出顶点法线，在内部插值得到每个pixel的法线，对每个pixel着色

Phong shading

- Interpolate normal vectors across each triangle
- Compute full shading model at each pixel
- Not the **Blinn-Phong Reflectance Model**

着色频率也取决于具体模型

Shading Frequency: Face, Vertex or Pixel



- 模型足够复杂时，可以采用相对简单的着色频率
- 模型足够复杂时，Phong shading开销不一定更大（三角形数量超过pixel数量）

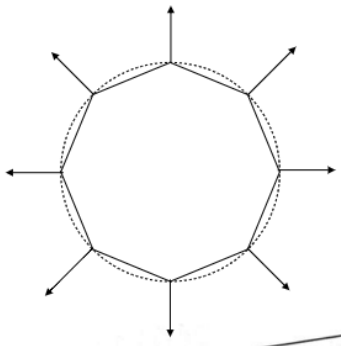
Defining Per-Pixel Normal Vectors

理想情况：根据试图表示的模型得到法线

一般情况：把相邻面法线做平均（也可根据面积加权平均）

Best to get vertex normals from the underlying geometry

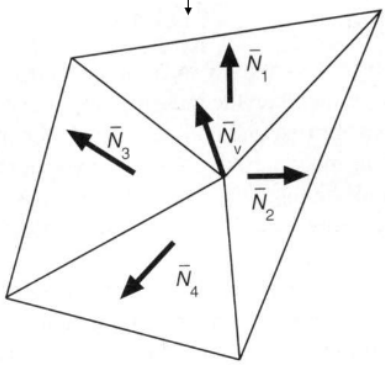
- e.g. consider a sphere



Otherwise have to infer vertex normals from triangle faces

- Simple scheme: **average surrounding face normals**

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

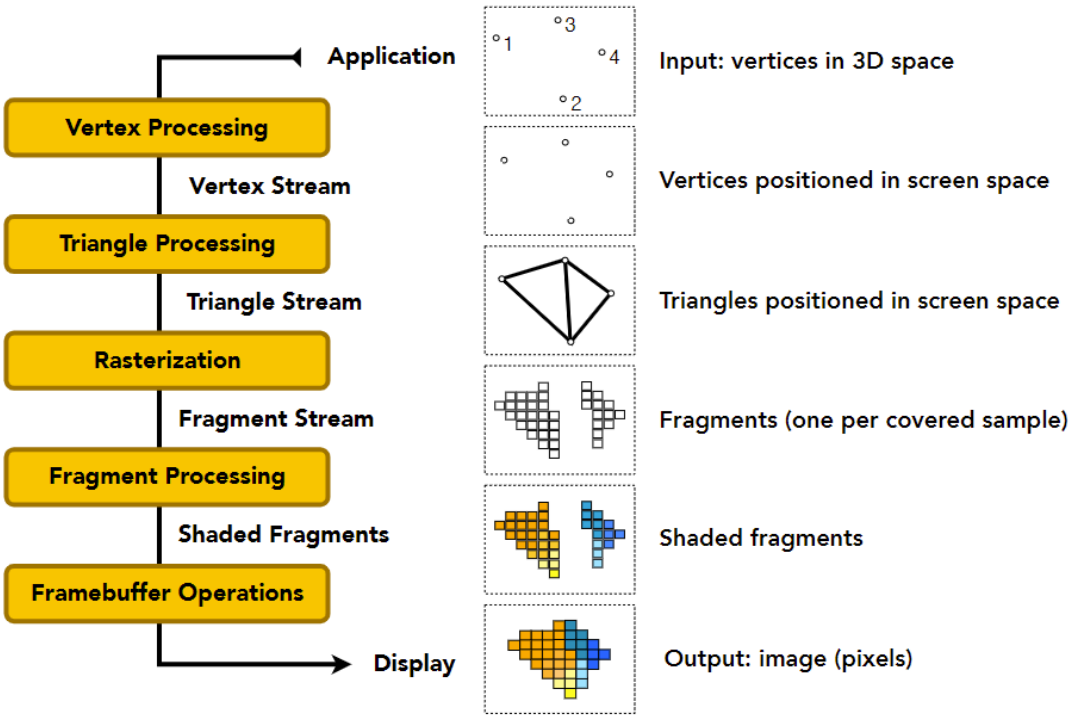


Q：如何在三角形内部得到平滑过渡的法线？

Barycentric interpolation of vertex normal

Graphics (Real-time Rendering) Pipeline

如何从场景到输出图像



注：记录不同的三角形，再MVP ⇔ 顶点MVP + 记录哪些顶点构成三角形

说明：

- MVP发生在Vertex Processing
- Sampling triangle发生在Rasterization
- Z-Buffer Visibility Tests发生在Fragment Processing
- Shading发生在Vertex Processing, Fragment Processing
  - 顶点着色可以发生在Vertex Processing
  - Phong shading发生在Fragment Processing
- Texture mapping (贴图) 发生在Vertex Processing, Fragment Processing

Shader Programs

GPU允许编程vertex and fragment processing阶段

Shader

Program vertex and fragment processing stages

- Describe operation on a single vertex (or fragment)

  每一个vertex (or fragment)都会执行，不需要循环
- 根据对象不同分为vertex shader和fragment shader

- Program vertex and fragment processing stages
- Describe operation on a single vertex (or fragment)

Example GLSL fragment shader program

```
uniform sampler2D myTexture;
uniform vec3 lightDir;
varying vec2 uv;
varying vec3 norm;

void diffuseShader()
{
  vec3 kd;
  kd = texture2d(myTexture, uv);
  kd *= clamp(dot(–lightDir, norm), 0.0, 1.0);
  gl_FragColor = vec4(kd, 1.0);
}
```

- **Shader function executes once per fragment.**
- **Outputs color of surface at the current fragment's screen sample position.**
- **This shader performs a texture lookup to obtain the surface's material color at this point, then performs a diffuse lighting calculation.**

推荐链接：[ShaderToy]

现代图形学发展方向

Goal: Highly Complex 3D Scenes in Realtime

Graphics Pipeline Implementation: GPUs

new shader: geometry shader, compute shader (通用计算)

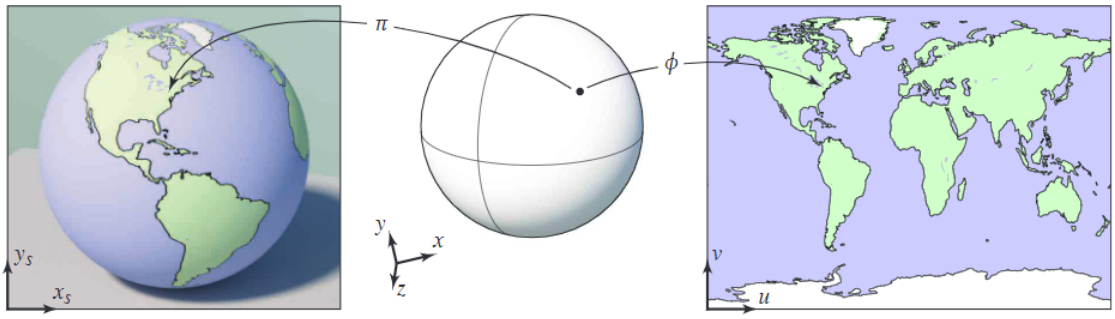
Texture mapping

希望在物体表面不同位置定义不同的属性

Surfaces are 2D

Surface lives in 3D world space

Every 3D surface point also has a place where it goes in the 2D image (**texture**).

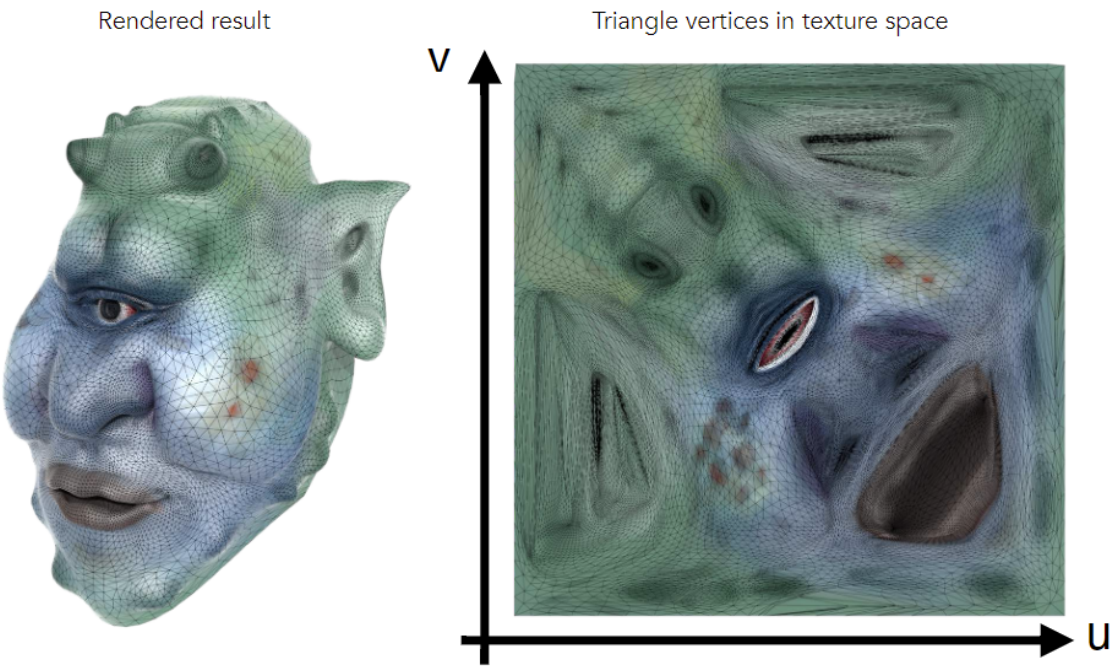


Screen space World space Texture space

纹理映射：把图贴在3D物体的表面；将纹理空间三角形映射到物体表面三角形上

Each triangle “copies” a piece of the texture image to the surface.

纹理坐标通常用(u,v)表示，u,v范围通常认为在0~1



纹理可以重复使用（最好可以使纹理无缝衔接）

着色（插值、高级纹理映射）

Shading

Barycentric coordinates

Why do we want to interpolate?

- Specify values at vertices
- Obtain smoothly varying values across triangles

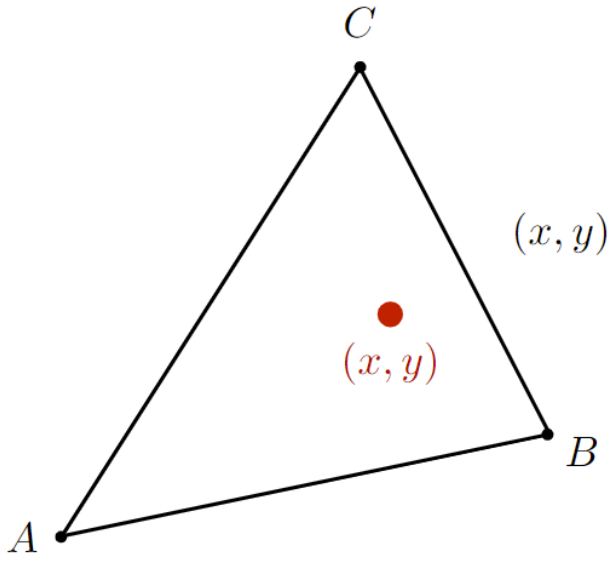
What do we want to interpolate?

- Texture coordinates, colors, normal vectors, ...

Interpolation Across Triangles: Barycentric Coordinates(重心坐标)

Barycentric Coordinates

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



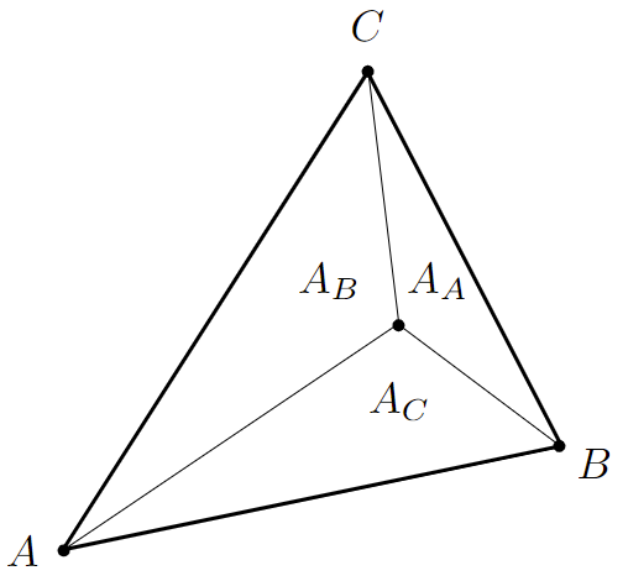
$$(x, y) = \alpha\, A + \beta\, B + \gamma\, C$$
$$\alpha + \beta + \gamma = 1$$

Inside the triangle if all three coordinates are non-negative

- 每个三角形有自己的重心坐标
- 平面内的任意一个点都可以表示成$(x, y) = \alpha A + \beta B + \gamma C \quad (\alpha + \beta + \gamma = 1)$

  若$\alpha + \beta + \gamma \neq 1$，得到的点不在平面内

Geometric viewpoint — proportional areas

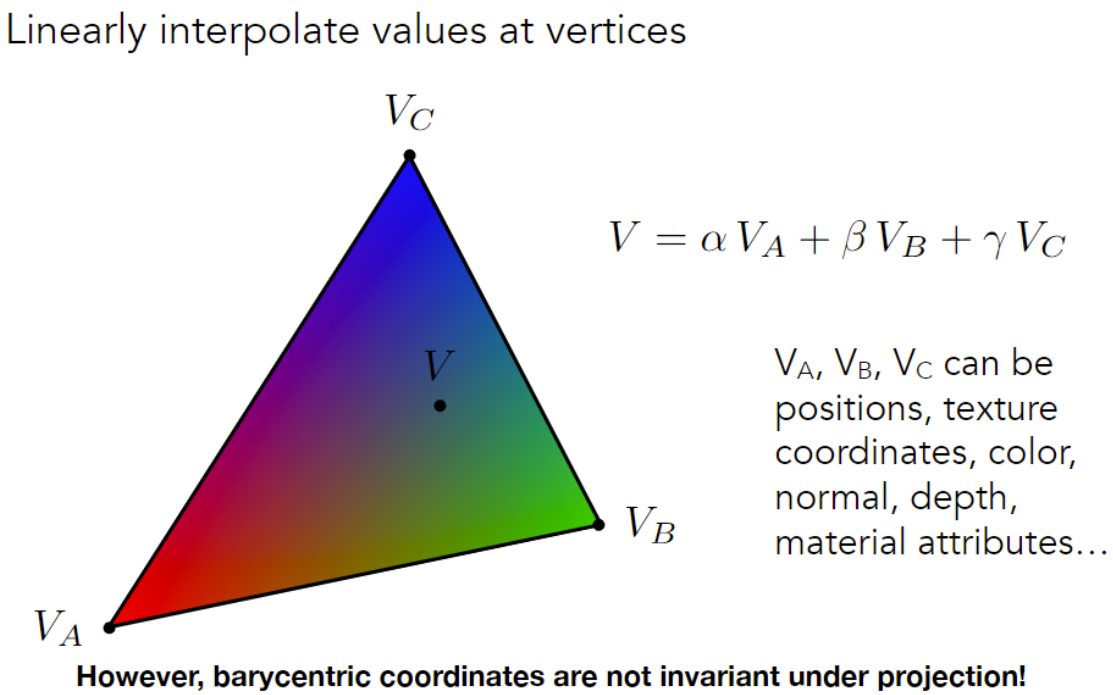


$$\alpha = \frac{A_A}{A_A + A_B + A_C}$$
$$\beta = \frac{A_B}{A_A + A_B + A_C}$$
$$\gamma = \frac{A_C}{A_A + A_B + A_C}$$

Formulas

$$\alpha = \frac{-(x - x_B)(y_C - y_B) + (y - y_B)(x_C - x_B)}{-(x_A - x_B)(y_C - y_B) + (y_A - y_B)(x_C - x_B)}$$
$$\beta = \frac{-(x - x_C)(y_A - y_C) + (y - y_C)(x_A - x_C)}{-(x_B - x_C)(y_A - y_C) + (y_B - y_C)(x_A - x_C)}$$
$$\gamma = 1 - \alpha - \beta$$

Using Barycentric Coordinates

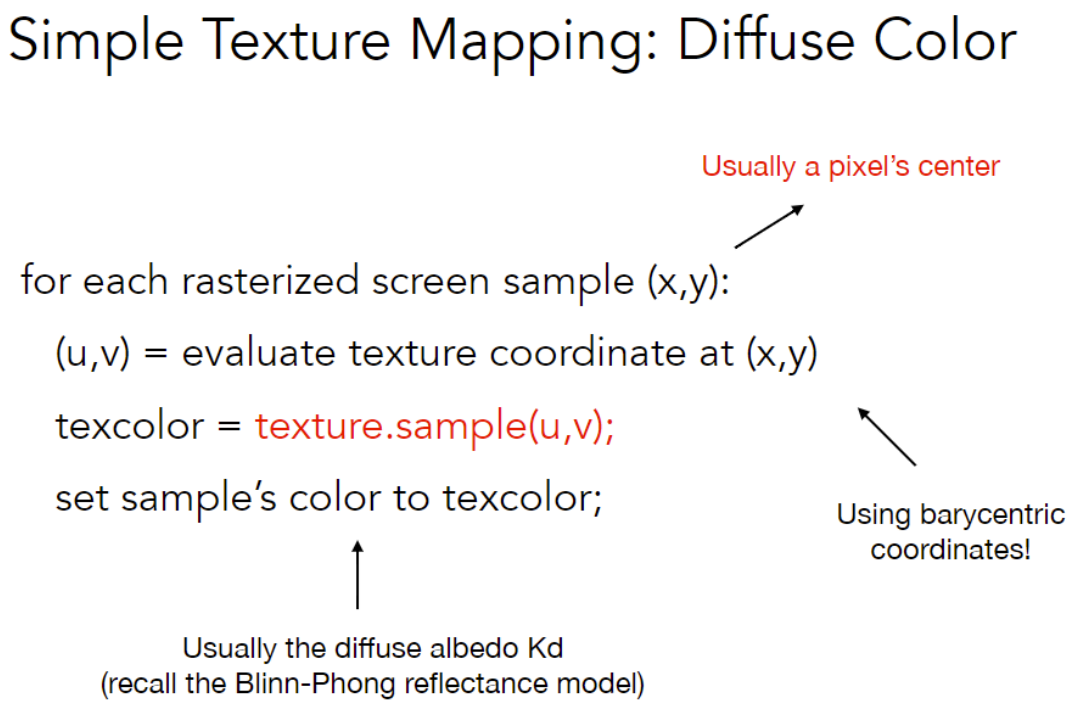


注：投影完重心坐标会改变，三维空间属性插值应该在投影之前完成

例如，插值z应该在原来的三维坐标中进行，而不是在投影后的二维坐标中计算

Texture queries

Applying Textures



Texture Magnification (Easy Case)

What if the texture is too small?

Texture Magnification - Easy Case

Generally don't want this — insufficient texture resolution

A pixel on a texture — a **texel** (纹理元素、纹素)

Nearest **Bilinear** **Bicubic**

Nearest: 对于pixel映射的texel做round操作（在一定范围内texel可能是一样的）

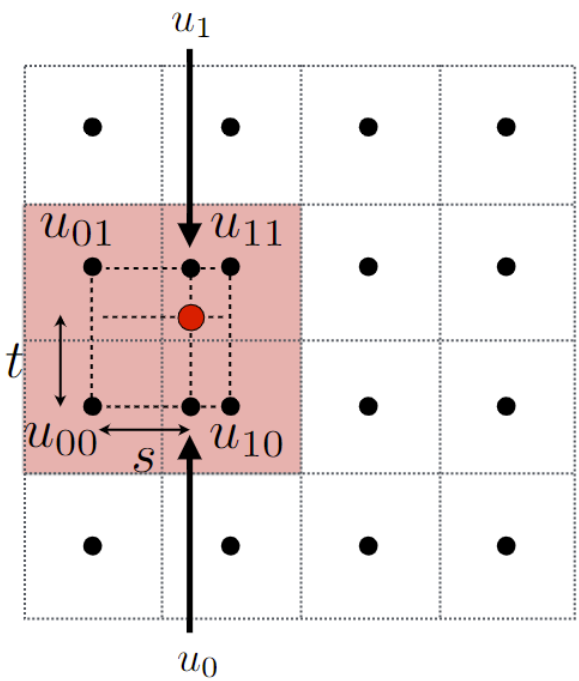
Bilinear Interpolation

Bilinear = horizontal linear + vertical linear

Black points indicate texture sample locations

Want to sample texture value f(x,y) at red point

- Take 4 nearest sample locations, with texture values as labeled.
- And fractional offsets, (s,t) as shown
- Do Linear interpolation (1D), get Two helper lerps (horizontal)
- Final vertical lerp, to get result



Linear interpolation (1D)

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

Two helper lerps

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

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

Final vertical lerp, to get result:

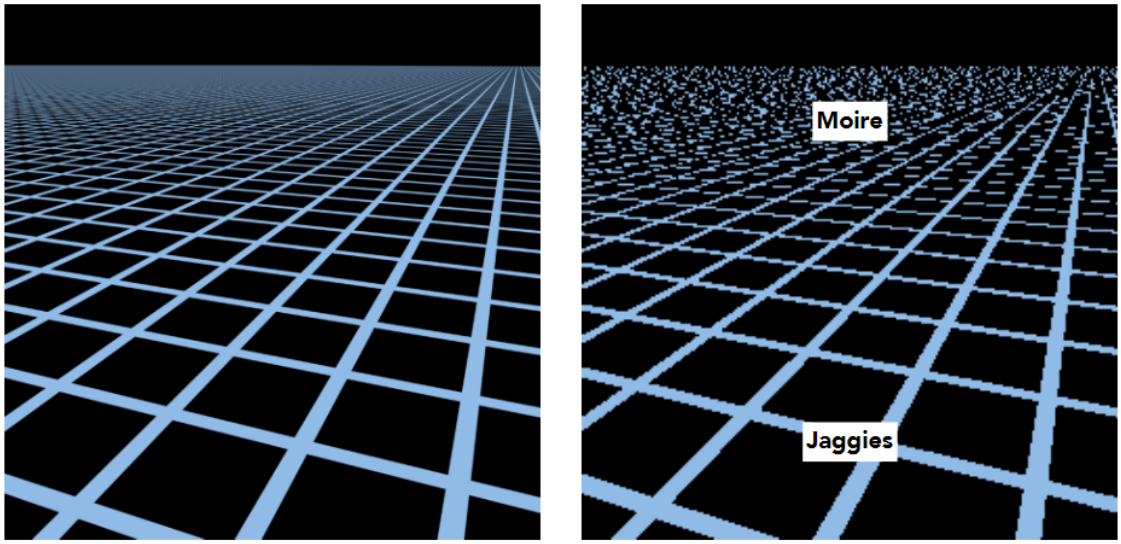
$$f(x, y) = \mathrm{lerp}(t, u_0, u_1)$$

Bilinear interpolation usually gives pretty good results at reasonable costs

Texture Magnification (hard case)

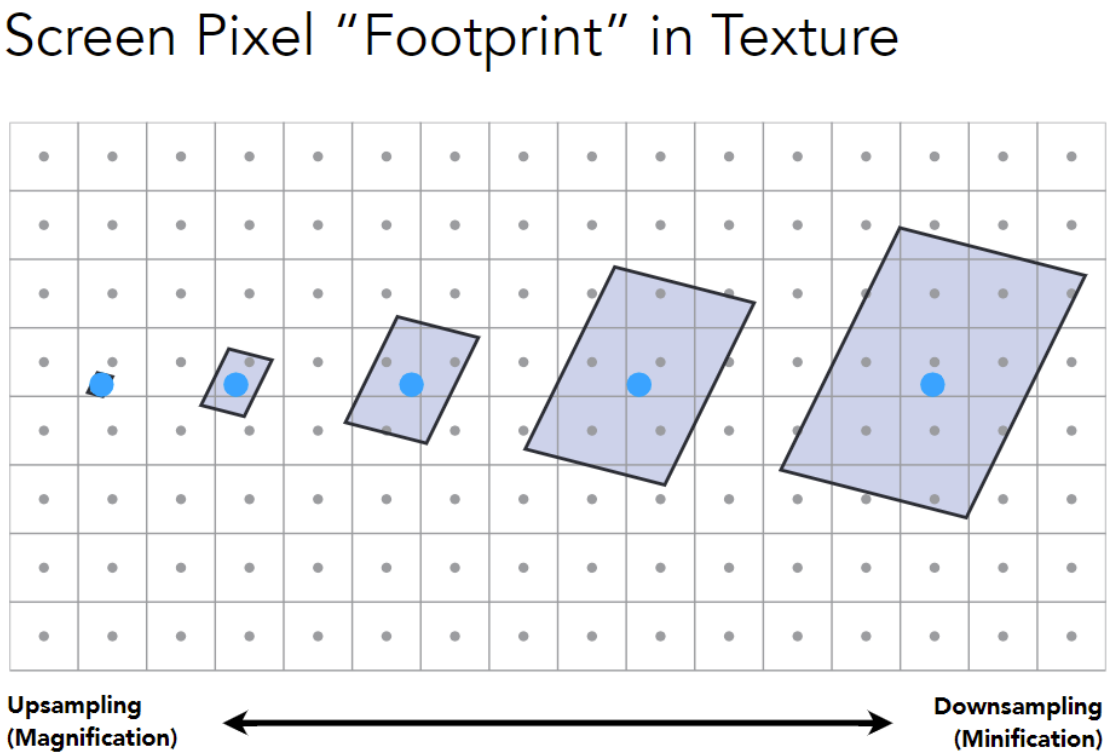
What if the texture is too large?

对纹理的采样率不够高



Reference **Point sampled**

灰色点为纹理的sample locations



在近处，每个pixel覆盖的纹理区域较小；在远处，每个pixel则会覆盖较大区域

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

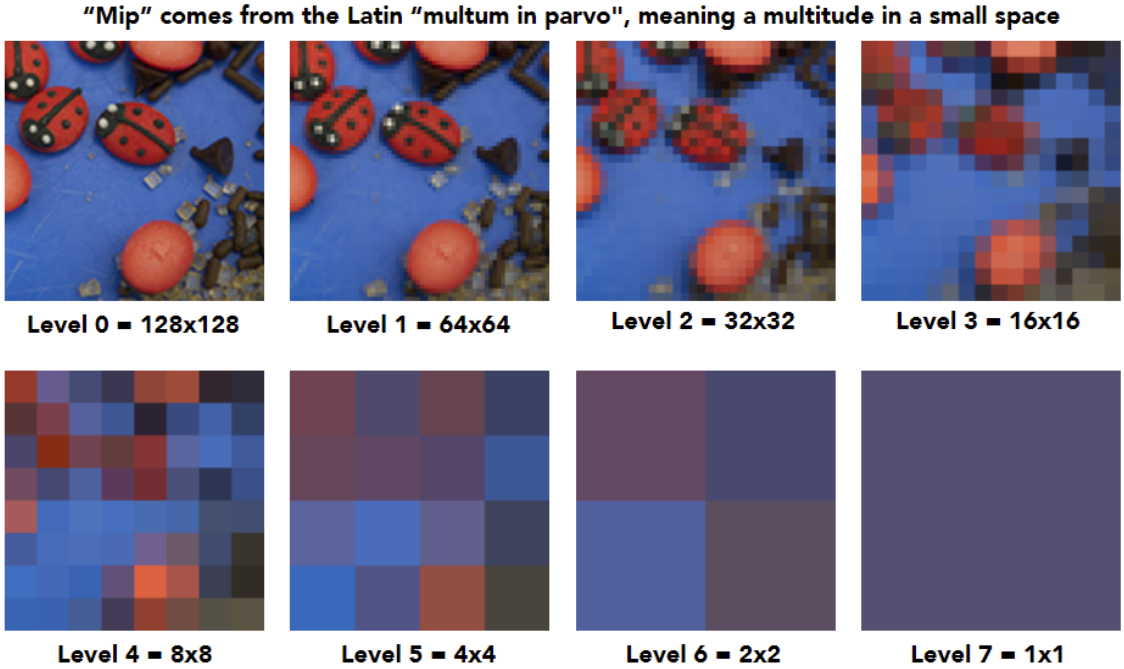
采样引起走样，直接避免采样

- What if we don't sample?
- Just need to get the average value within a range! (区间查询问题，需要支持任意大小的查询)

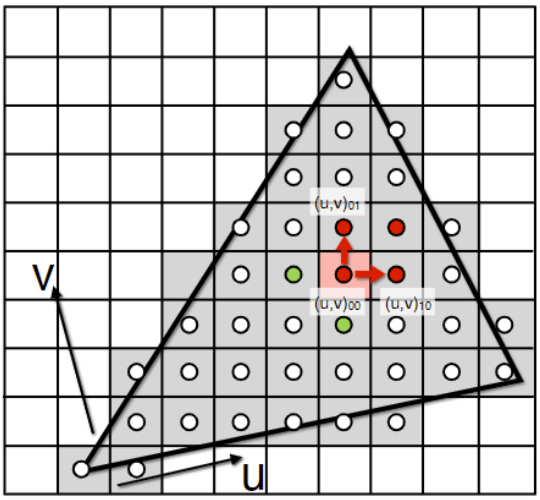
Mipmap

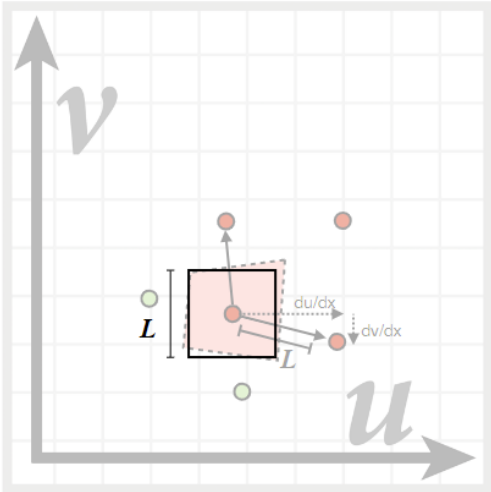
Allowing (fast, approx., square) range queries

根据原图生成一系列图（类似二维ST表）



如何得到texture footprint

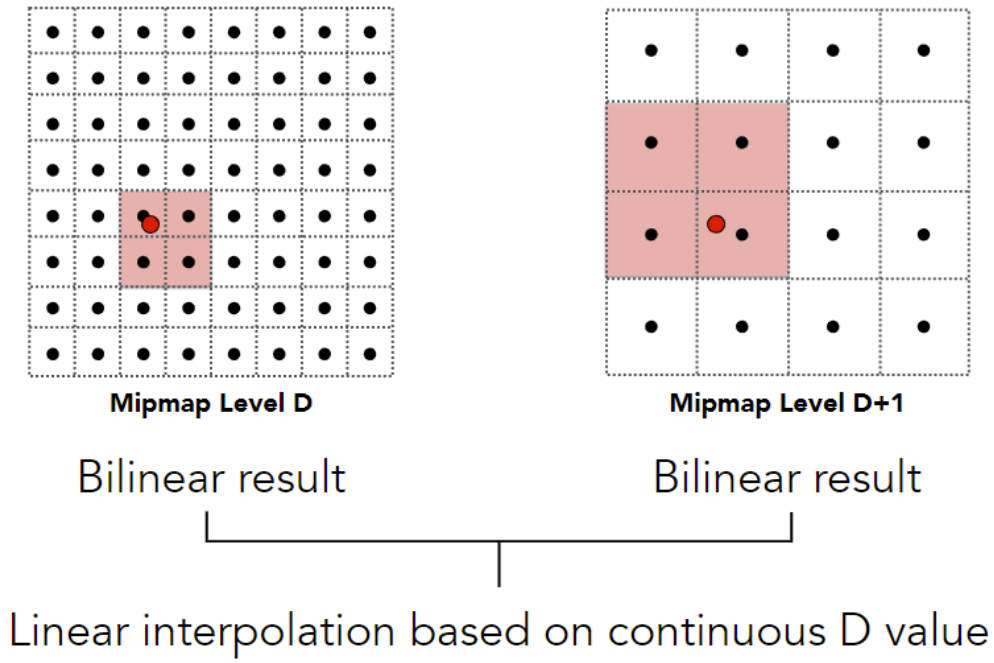




$$D = \log_2 L \qquad L = \max\left(\sqrt{\left(\frac{du}{dx}\right)^2 + \left(\frac{dv}{dx}\right)^2}, \sqrt{\left(\frac{du}{dy}\right)^2 + \left(\frac{dv}{dy}\right)^2}\right)$$

- Estimate texture footprint using texture coordinates of neighboring screen samples
- 求出相邻点映射后之间的长度L，用来近似正方形区域的边长（有限增量公式）
- 计算$D = \log_2 L$，对于整数D直接查询，对于小数的D进行插值 (Trilinear Interpolation)

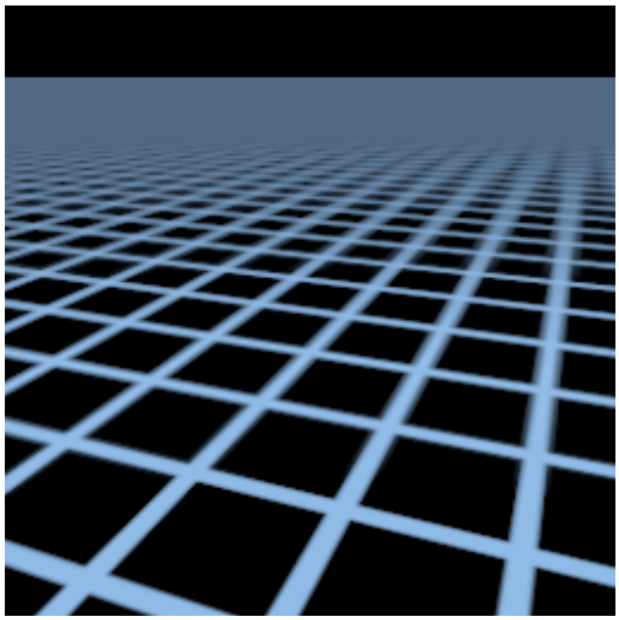
Trilinear Interpolation



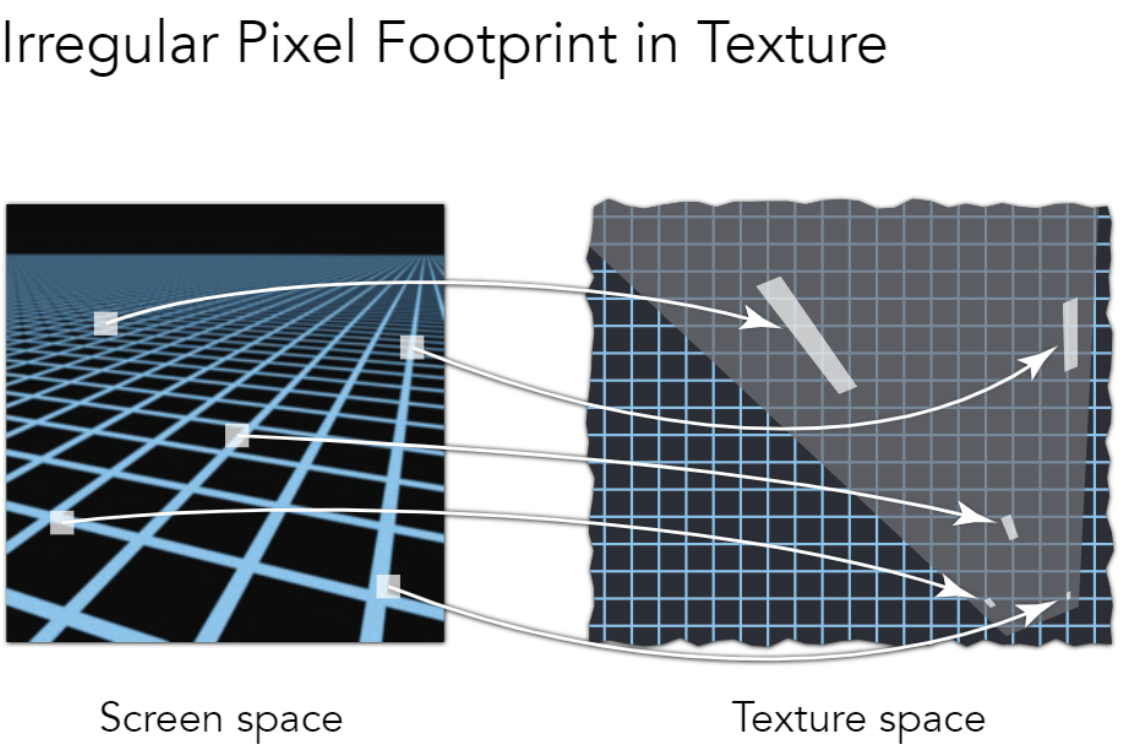
- 分别在level D和level D+1内先Bilinear插值，在D和D+1之间Linear插值
- 在层与层之间和层内部都是连续的

Mipmap Limitations

Overblur
Why?



Mipmap trilinear sampling

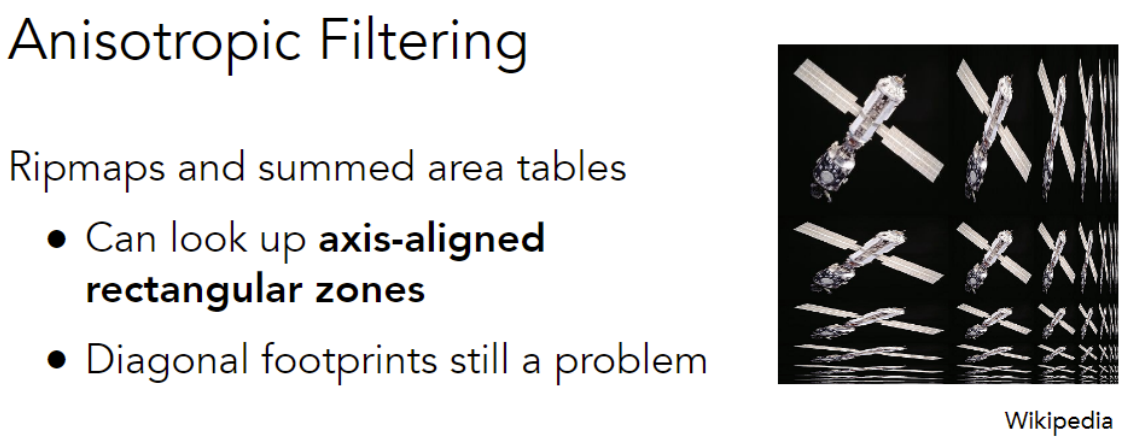


mipmap只能查询正方形，且引入了较多的近似

Anisotropic Filtering (各向异性过滤)

引入了不均匀的水平和竖直压缩，可以解决矩形区域情况，无法解决斜矩形的情况

最多引入3倍的额外存储量，kx各向异性过滤即引入k层压缩



其它过滤方法

EWA filtering

用多个圆形覆盖区域

- Use multiple lookups
- Weighted average
- Mipmap hierarchy still helps
- Can handle irregular footprints

几何（基本表示方法）

Shading

Applications of textures

In modern GPUs, texture = memory + range query (filtering)

- General method to bring data to fragment calculations

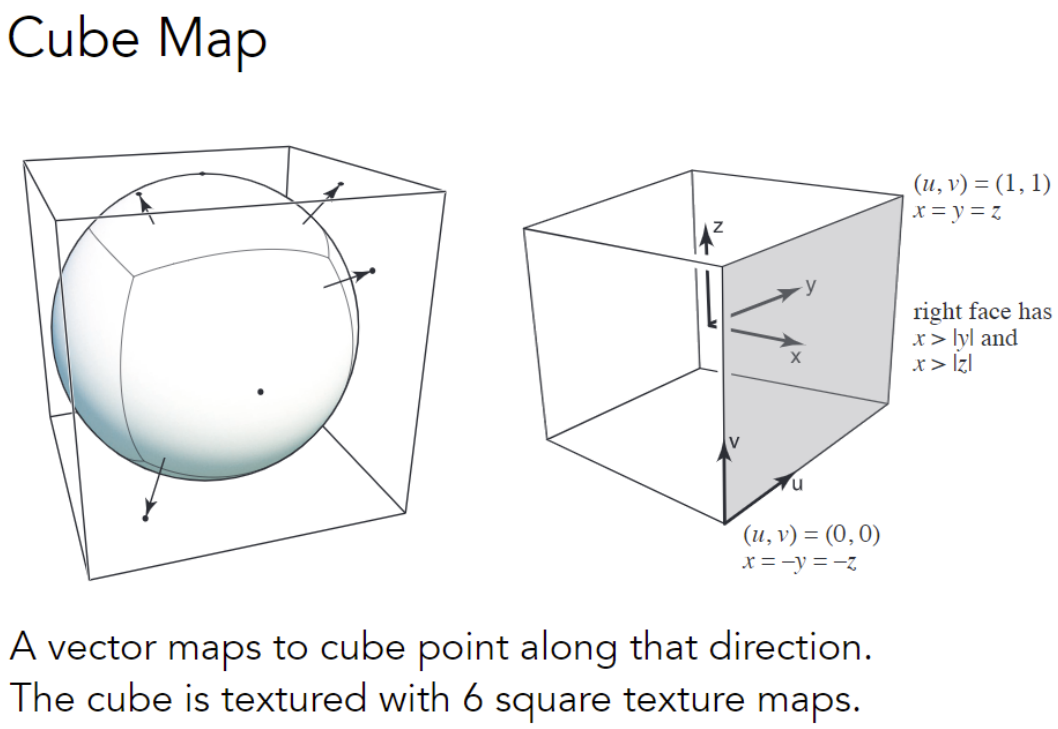
Many applications

- Environment lighting (环境贴图)

  把环境光记录下来做成纹理（假设环境光无限远，不考虑不同位置环境光不同）

将环境光记录在球面上，再展开

Problem: 上下部分会被很大地扭曲，不是均匀的表示



考虑球的立方体包围盒，环境光映射到矢量方向的立方体点上

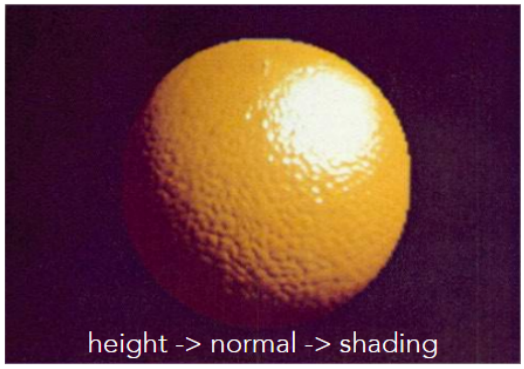
用立方体记录，不容易扭曲，但是计算稍微复杂一些

- Store microgeometry

  Textures can affect shading!

  - Textures doesn't have to only represent colors
    - What if it stores the height / normal?
    - Bump / normal mapping
    - **Fake** the detailed geometry

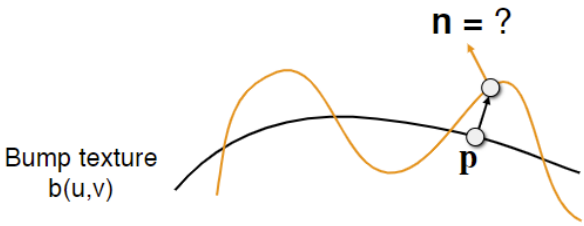
Relative height to the underlying surface



Bump Mapping

Adding surface detail without adding more triangles

- Perturb surface normal per pixel
  (for shading computations only)
- "Height shift" per texel defined by a texture
- How to modify normal vector?



- 凹凸贴图定义相对高度，从而改变normal方向
- 定义复杂纹理但不改变几何信息
- 给出假的法线以产生假的着色结果，以假乱真

How to perturb the normal ?

Note that this is in local coordinate!

认为局部法线是(0, 0, 1)，计算出新法线后变换回原来的坐标

in flatland

1. Original surface normal n(p) = (0, 1)
2. 求出切线方向 dp = c * [h(p+1) - h(p)] (c控制贴图的影响)

3. n(p) = (-dp, 1).normalized()

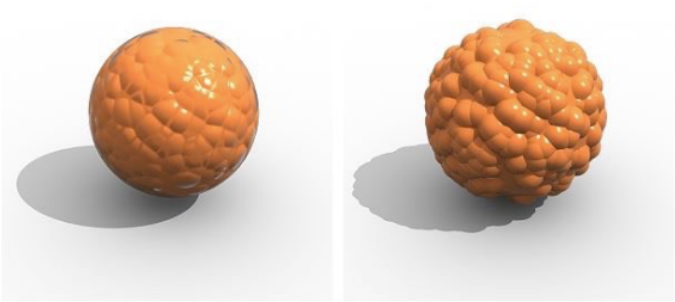
in 3D

1. Original surface normal n(p) = (0, 0, 1)
2. 求出方向导数

   dp/du = c1 * [h(u+1) - h(u)]

   dp/dv = c2 * [h(v+1) - h(v)]
3. n = (-dp/du, -dp/dv, 1).normalized() (切平面的法线)

- **Displacement mapping** — a more advanced approach
  - Uses the same texture as in bumping mapping
  - Actually **moves the vertices**



Bump / **Normal** mapping Displacement mapping

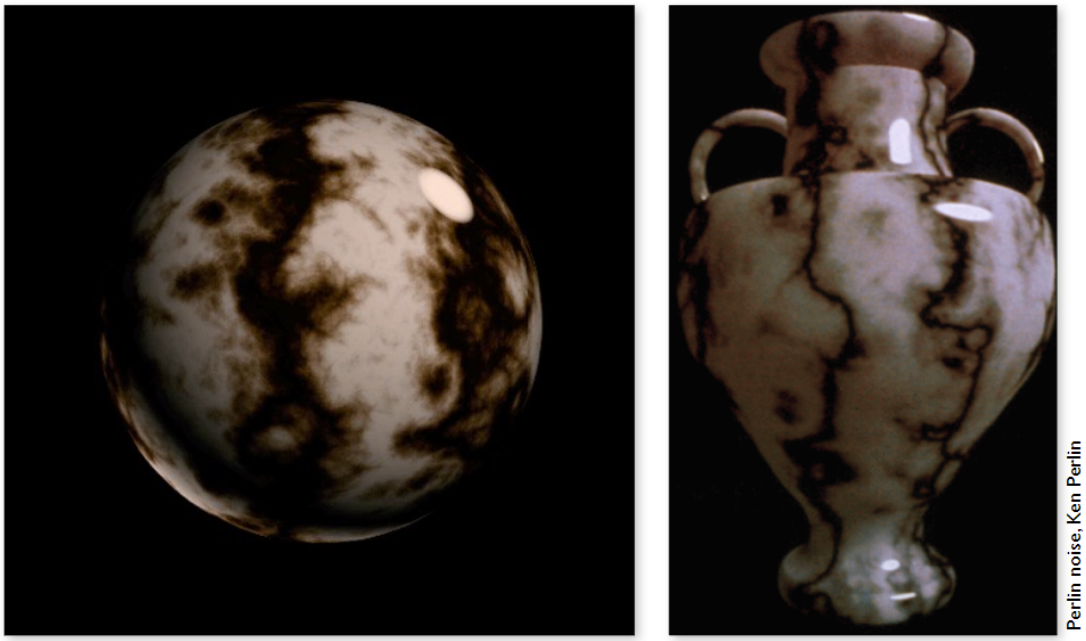
- 凹凸贴图在处理边缘和自身阴影时不真实
- 位移贴图会实际改变几何体，要求三角形足够多

  DirectX动态曲面细分，根据需要划分更多的几何体
- 其它: Parallax mapping, relief mapping

- Procedural textures + Solid modeling

  定义三维空间函数，可以看到内部纹理

3D Procedural Noise + Solid Modeling



- Provide Precomputed Shading

  例如，把阴影计算好记录进纹理
- Volume rendering

  将3D信息记录到纹理

Introduction to geometry

2nd part of this course!

Various representations of geometry

“Implicit” Representations of Geometry

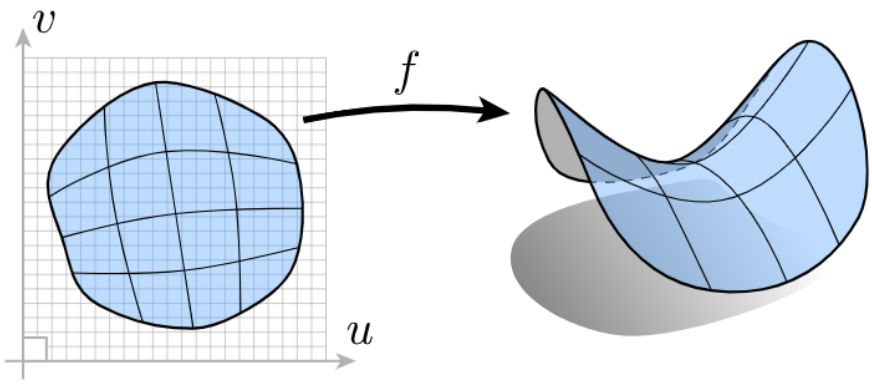
给出点满足的方程，$f(x, y, z) = 0$

- 难以看出整体结构
- Inside/Outside Tests Easy

“Explicit” Representations of Geometry

All points are given directly (e.g. 给出三角形顶点) or via parameter mapping (参数方程)

Generally: $f : \mathbb{R}^2 \to \mathbb{R}^3; (u, v) \mapsto (x, y, z)$



- 带入u,v即可得到整体结构
- Inside/Outside Test Hard

No “Best” Representation – Geometry is Hard!

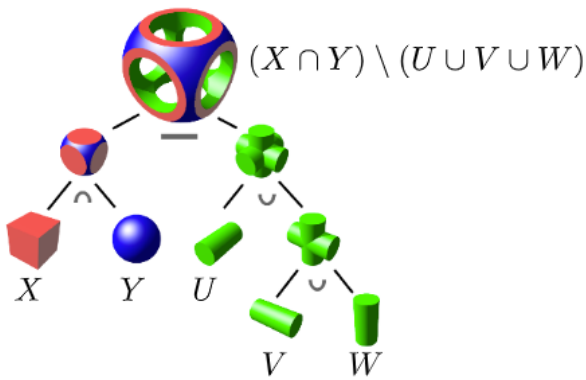
Best Representation Depends on the Task!

More Implicit Representations

- Algebraic Surfaces

  $f(x, y, z) = 0$
- Constructive Solid Geometry (CSG)

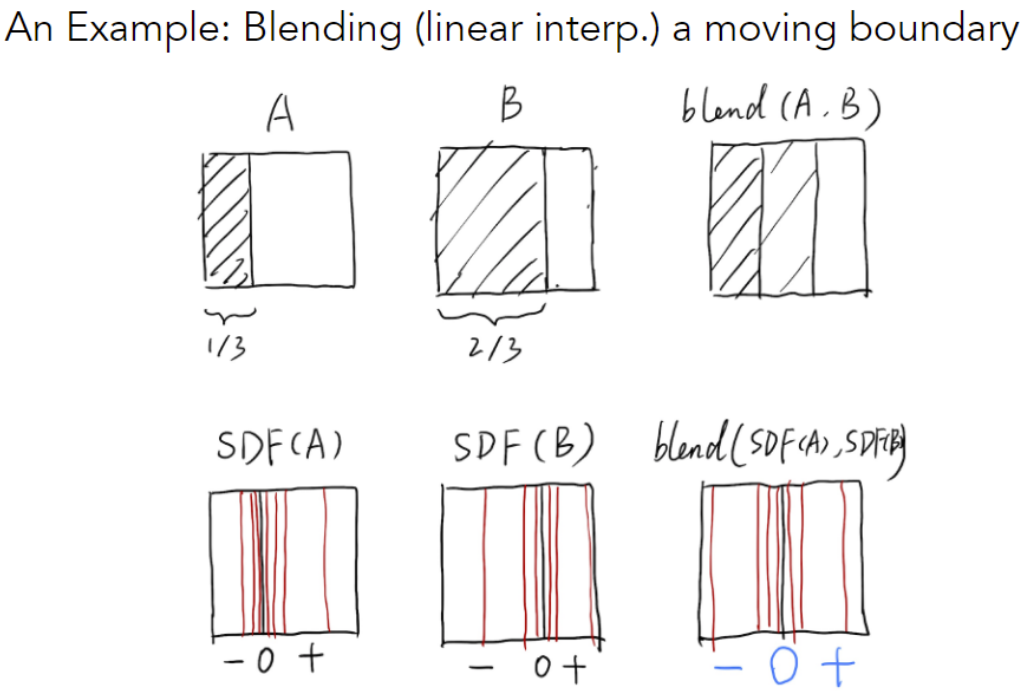
  基本几何 + 基本运算$\cap \cup \setminus$



- Distance Functions

  Distance functions: giving minimum distance (could be signed distance) from anywhere to object

  blend any distance functions



求出A到B的运动的中间状态。直接blend中间效果不理想；理想结果左边是黑的，右边是白的

Signed Distance Fields (SDF)，有非常强的表示能力

- 分别求出A和B的SDF（可以看到+-0三部分）
- blend SDF，得到中间是0，即边界

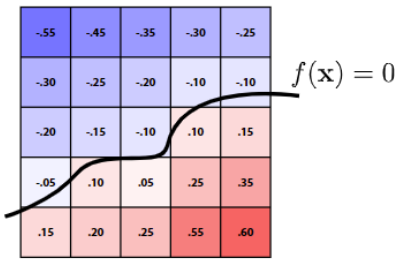
SDF阅读材料: link

- Level Set Methods (水平集)

  Distance functions难以给出解析式情况，用mesh grid记录Distance functions

  Closed-form equations are hard to describe complex shapes

  Alternative: store a grid of values approximating function



  Surface is found where interpolated values equal zero

  Provides much more explicit control over shape (like a texture)

  3D level set: 找等高线

- Fractals (分形)

  Exhibit self-similarity, detail at all scales

Pros:

- compact description (e.g., a function)
- certain queries easy (inside object, distance to surface)
- good for ray-to-surface intersection (more later)
- for simple shapes, exact description (严格描述) / no sampling error

- easy to handle changes in topology (e.g., fluid)

Cons:

- difficult to model complex shapes

几何（曲线与曲面）

Various representations of geometry

Explicit Representations

- Point Cloud

  把表面的点表示得足够细（e.g. 扫描后的结果）

  Easiest representation: list of points (x,y,z)

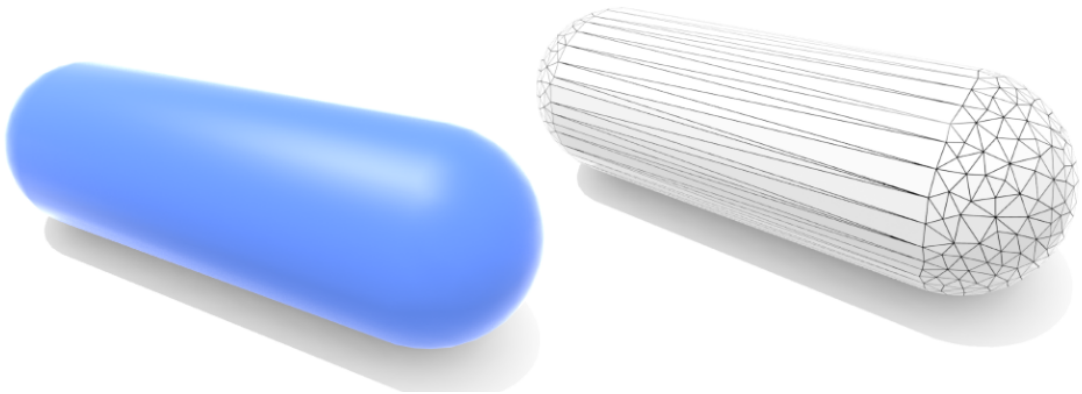
  Easily represent any kind of geometry

  Often converted into polygon mesh

- Polygon Mesh

  Store vertices & polygons (often triangles or quads)

  Easier to do processing / simulation, adaptive sampling

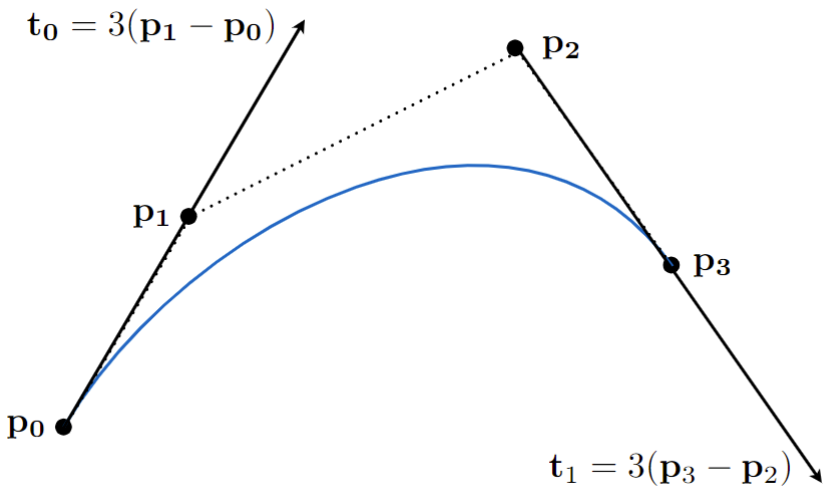


The Wavefront Object File (.obj) Format: stores vertices, normals, texture coordinates + their connectivities

Curves

Bezier curves (贝塞尔曲线)

用一系列控制点定义曲线

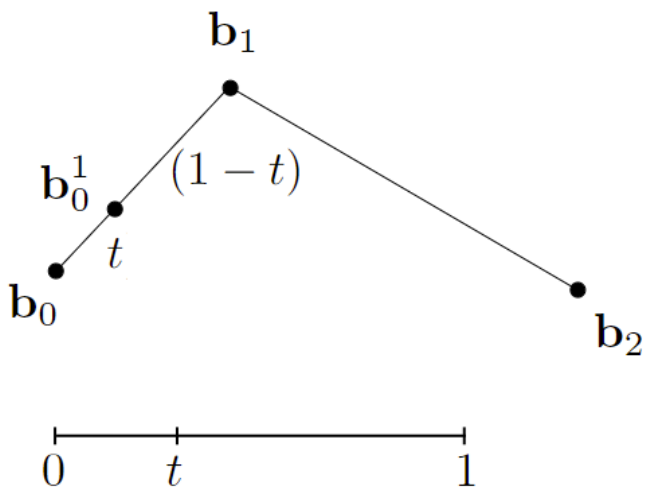


De Casteljau's algorithm

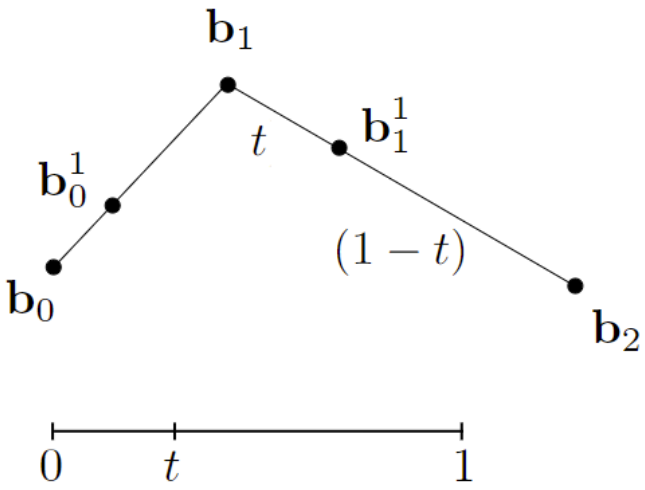
给出任意时间t，找到该时刻曲线上的点

Consider three points (quadratic Bezier)

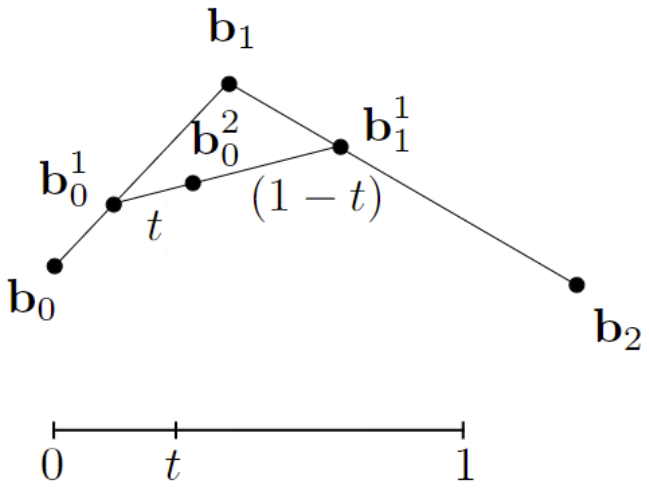
Insert a point using linear interpolation



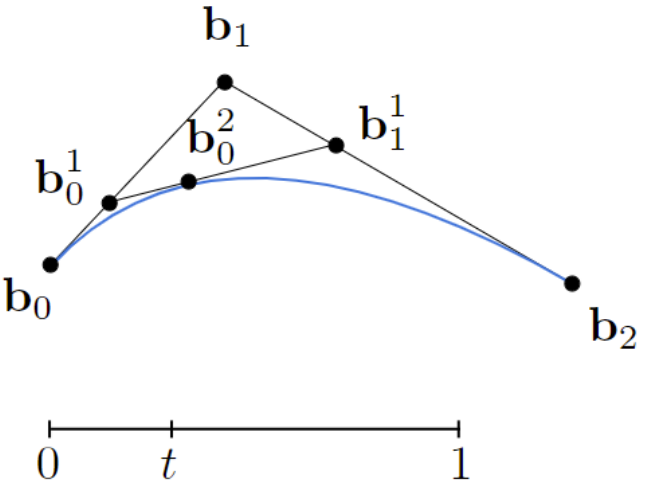
Insert on both edges

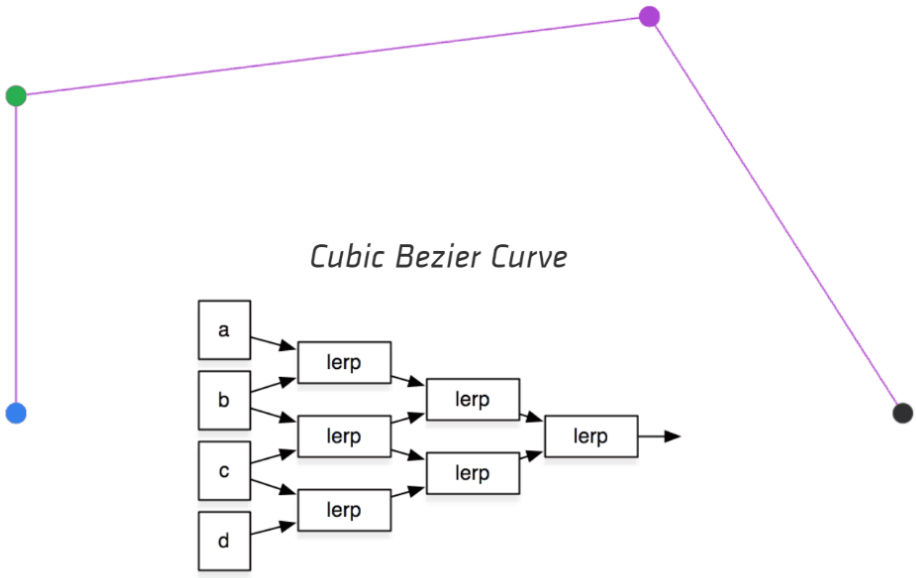


Repeat recursively



Run the same algorithm for every t in [0,1]

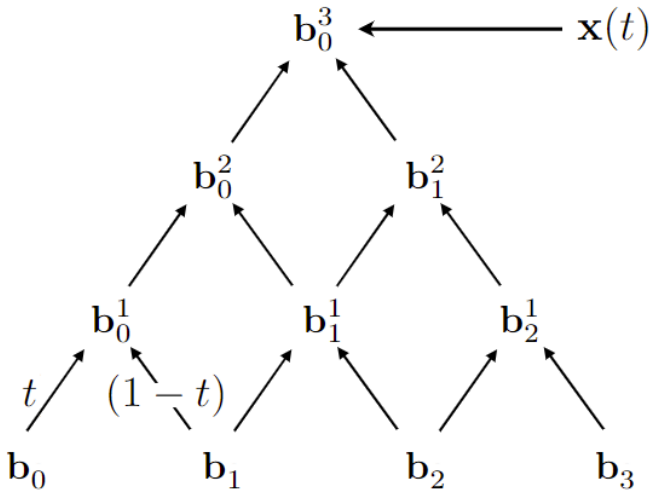


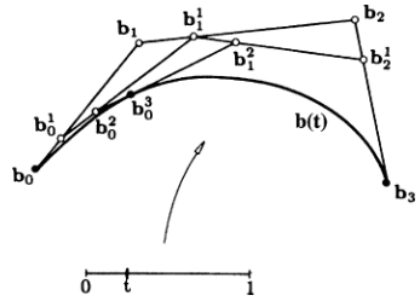


参考链接：link

Algebraic Formula

de Casteljau algorithm gives a pyramid of coefficients





Every rightward arrow is multiplication by t,
Every leftward arrow by (1–t)

Example: quadratic Bezier curve from three points

$$\mathbf{b}_0^1(t) = (1-t)\mathbf{b}_0 + t\mathbf{b}_1$$
$$\mathbf{b}_1^1(t) = (1-t)\mathbf{b}_1 + t\mathbf{b}_2$$

$$\mathbf{b}_0^2(t) = (1-t)\mathbf{b}_0^1 + t\mathbf{b}_1^1$$

$$\mathbf{b}_0^2(t) = (1-t)^2\mathbf{b}_0 + 2t(1-t)\mathbf{b}_1 + t^2\mathbf{b}_2$$

用二项式定理化简计算

Bézier Curve – General Algebraic Formula

Bernstein form of a Bézier curve of order n:

$$\mathbf{b}^n(t) = \mathbf{b}_0^n(t) = \sum_{j=0}^{n} \mathbf{b}_j B_j^n(t)$$

Bézier curve order n
(vector polynomial of degree n)

Bernstein polynomial
(scalar polynomial of degree n)

Bézier control points
(vector in R^N)

Bernstein polynomials:

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

$$\sum_{i=0}^{n} B_i^n(t) = 1$$

可以理解为：定义和时间有关的多项式，对控制点进行插值

Properties of Bezier Curves

Interpolates endpoints

- For cubic Bezier: $\mathbf{b}(0) = \mathbf{b}_0 \quad \mathbf{b}(1) = \mathbf{b}_3$

Tangent to end segments (仅对于cubic case)

- Cubic case: $\mathbf{b}'(0) = 3\left(\mathbf{b}_1 - \mathbf{b}_0\right) \quad \mathbf{b}'(1) = 3\left(\mathbf{b}_3 - \mathbf{b}_2\right)$

Affine transformation property

- Transform curve by transforming control points

  画出曲线再仿射变换 = 仿射变换控制点再画出曲线

  投影变换不行

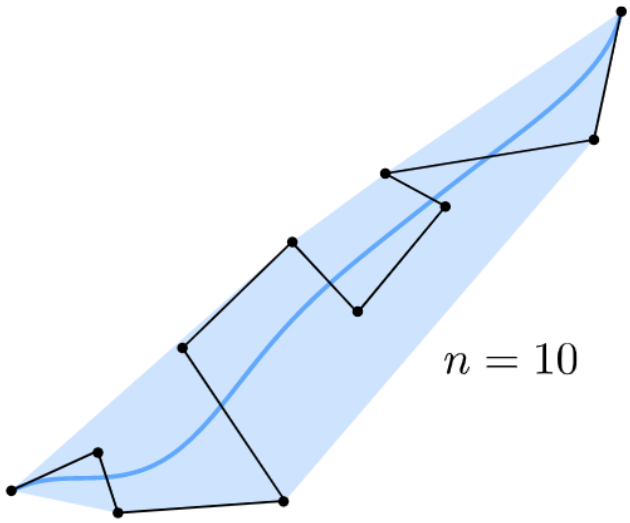
Convex hull (凸包) property

- Curve is <u>within</u> convex hull of control points

  简单应用：控制点排列在直线上

Piecewise (逐段) Bezier Curves

Higher-Order Bezier Curves?



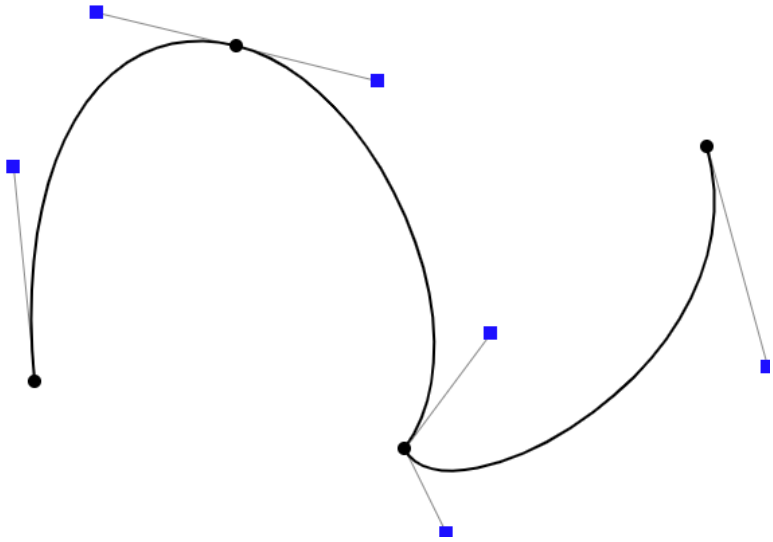
Very hard to control!

Uncommon

Instead, chain many low-order Bezier curve

<u>Piecewise cubic Bezier</u> the most common technique (一般每4个控制点分段)

Demo: link



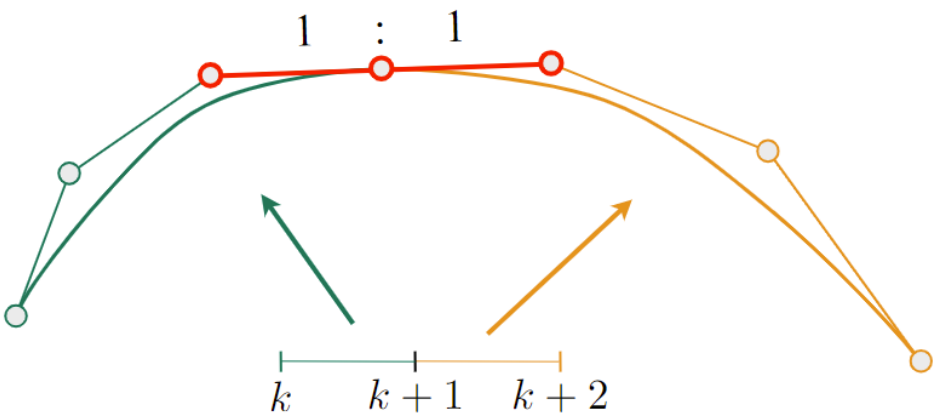
Q：怎么保证连接曲线是光滑的?

A：连接处控制杆共线等长反向

Continuity

C1连续：导数连续

$$C^1 \text{ continuity:} \qquad \mathbf{a}_n = \mathbf{b}_0 = \frac{1}{2}\left(\mathbf{a}_{n-1} + \mathbf{b}_1\right)$$



B-splines

Spline (样条)

- a continuous curve constructed so as to pass through a given set of points and have a certain number of continuous derivatives.
- In short, a curve under control

B-splines (basis splines)

e.g. Bernstein polynomials为基函数 (basis)

- Require more information than Bezier curves
- Satisfy all important properties that Bezier curves have (i.e. superset)

- 满足局部性：一个控制点改变不会影响整条曲线，只影响某个局部范围

Further study

- B-splines and NURBS
- operations on curves (e.g. increasing/decreasing orders, etc.)
- reference: [Prof. Shi-Min Hu's course]

Surfaces

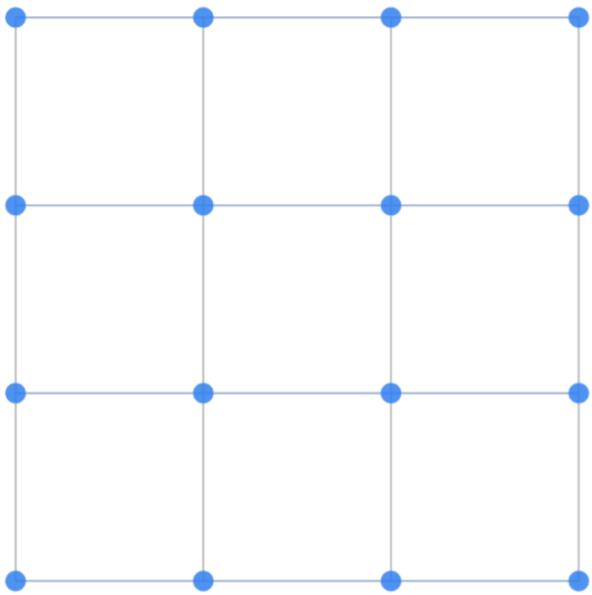
Bezier surfaces

在水平和竖直方向分别计算Bezier curves

Bicubic Bezier Surface

4 x 4 control points

1. 根据每行的4个控制点生成4条Bezier curves
2. 把第一步生成的4个点当作新的控制点



Evaluating Bezier Surfaces

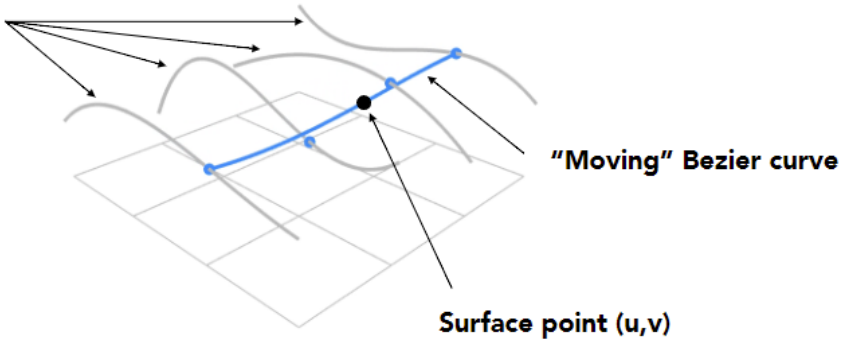
Evaluating Surface Position For Parameters (u,v)

Input: 4x4 control points (bi-cubic case)

Output is 2D surface parameterized by (u,v) in $[0,1]^2$

(u,v)-separable application of de Casteljau algorithm

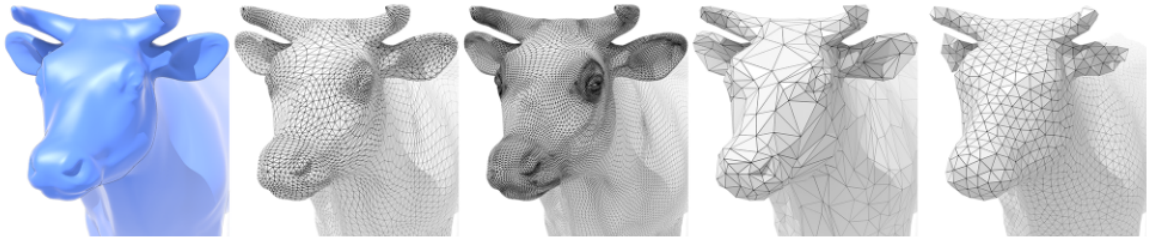
1. Use de Casteljau to evaluate point u on each of the 4 Bezier curves in u. This gives 4 control points for the "moving" Bezier curve
2. Use 1D de Casteljau to evaluate point v on the "moving" curve



Mesh Operations: Geometry Processing

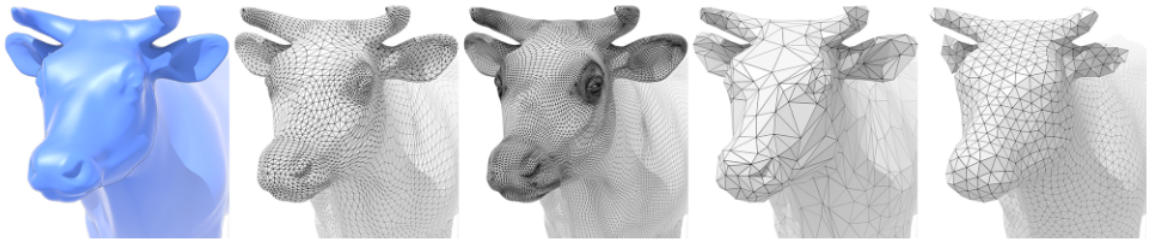
- Mesh subdivision (细分)
- Mesh simplification (简化)
- Mesh regularization

  regularization: 让三角形都变得和正三角形差不多



几何（网格处理）、阴影图

Mesh Operations: Geometry Processing



Mesh subdivision

Increase resolution, upsampling

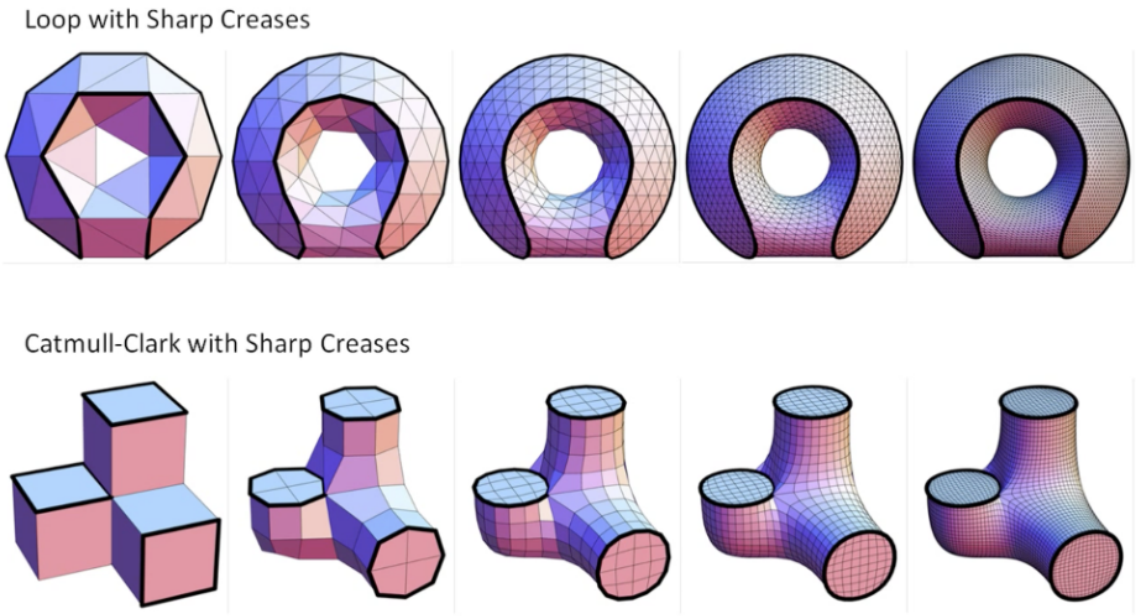
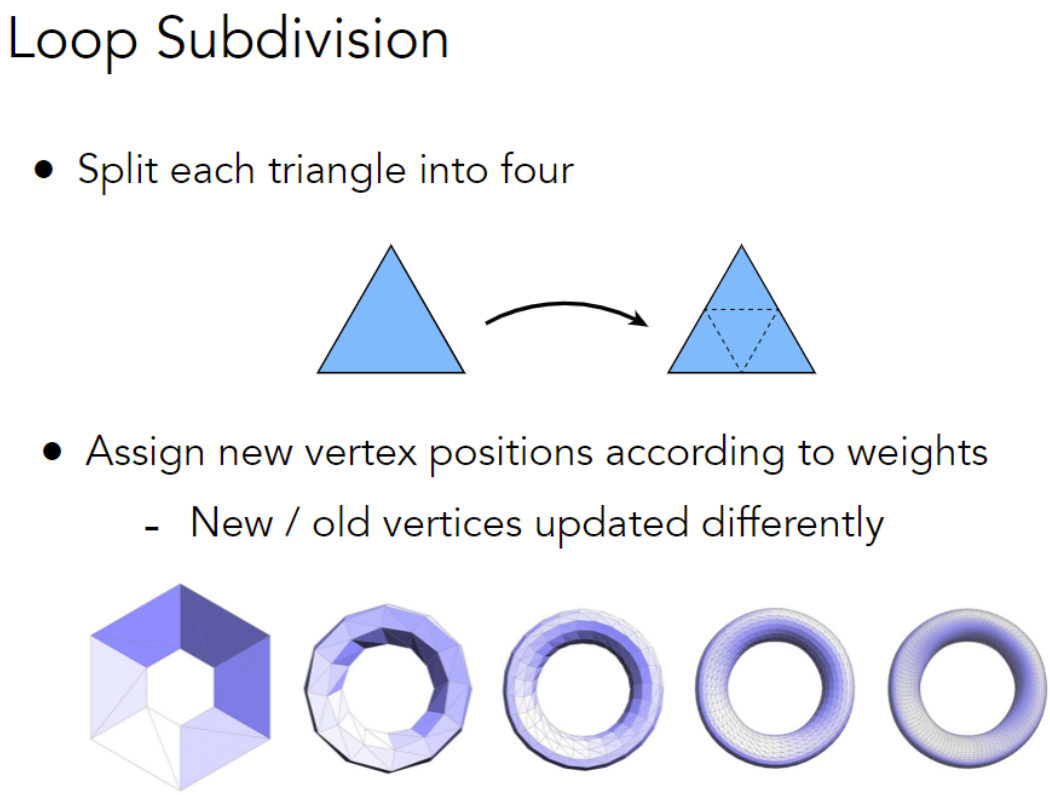


Figure from: Hakenberg et al. Volume Enclosed by Subdivision Surfaces with Sharp Creases

Loop Subdivision

Common subdivision rule for triangle meshes

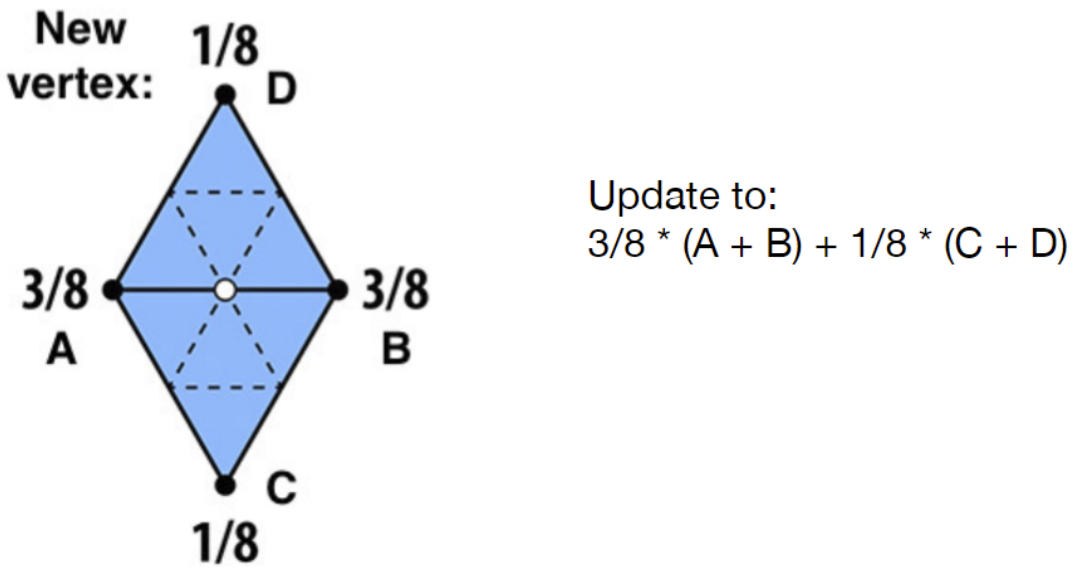
1. create more triangles (vertices)
2. tune their positions



Loop Subdivision — Update

- For new vertices

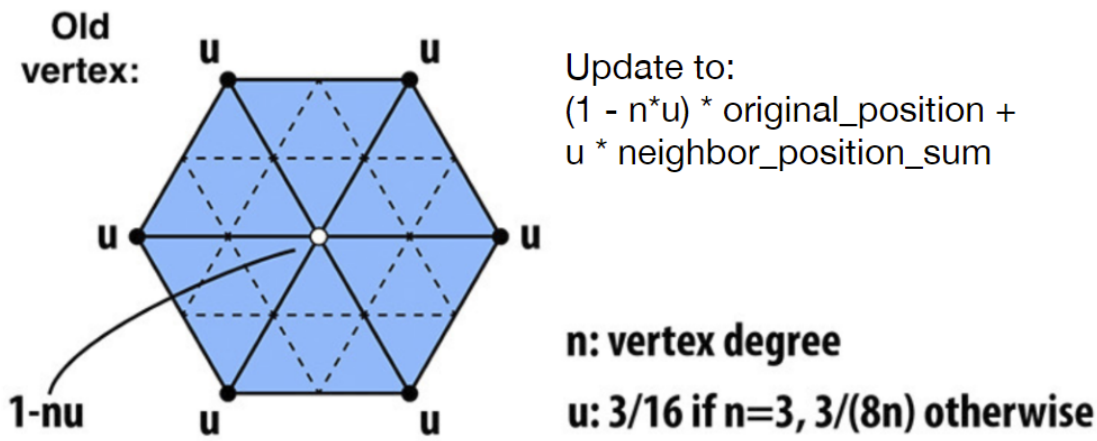
  根据周围点加权平均



- For old vertices

  e.g. degree 6 vertices here

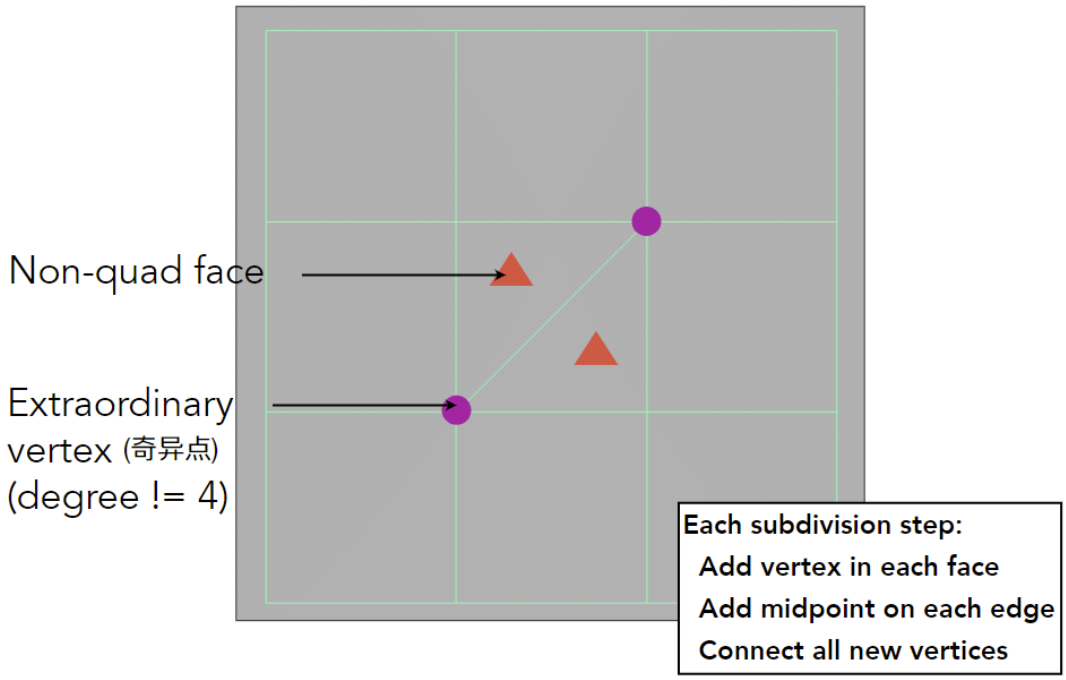
  根据自身和周围点加权平均

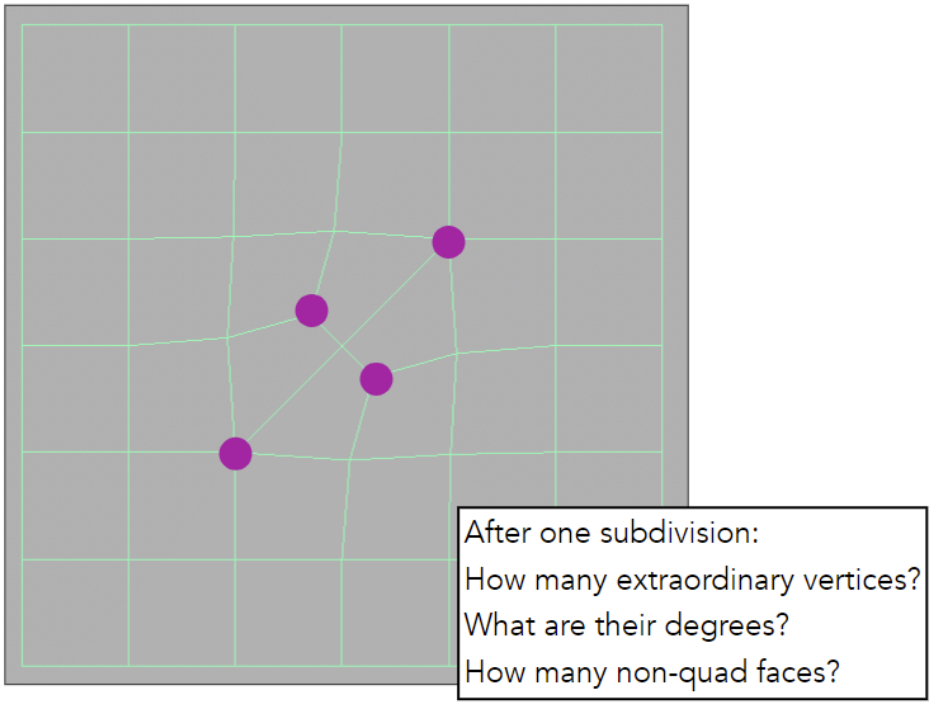


Catmull-Clark Subdivision

处理一般情况，不只是纯三角形

1. create more vertices





After one subdivision:

- How many extraordinary vertices?

  原来的奇异点 + 划分三角形得到的奇异点，之后再也不会增加

- What are their degrees?

  新的奇异点度为3

- How many non-quad faces?

  None. 所有non-quad faces在引入奇异点后消失

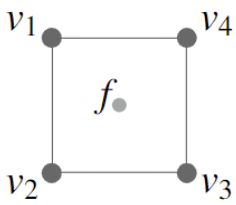
2. Catmull-Clark Vertex Update Rules

   本质上就是加权平均

   分为Face point + Edge point + Vertex point

FYI: Catmull-Clark Vertex Update Rules (Quad Mesh)

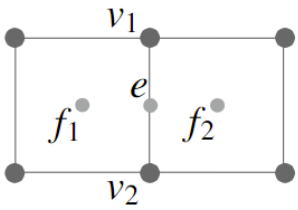
Face point

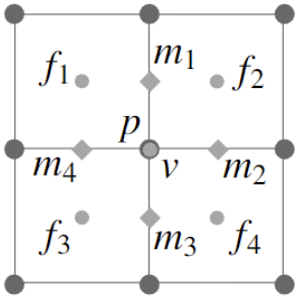


$$f = \frac{v_1 + v_2 + v_3 + v_4}{4}$$

$$e = \frac{v_1 + v_2 + f_1 + f_2}{4}$$

Edge point





Vertex point

$$v = \frac{f_1 + f_2 + f_3 + f_4 + 2(m_1 + m_2 + m_3 + m_4) + 4p}{16}$$

m midpoint of edge
p old "vertex point"

Mesh simplification

Decrease resolution, downsampling; try to preserve shape/appearance

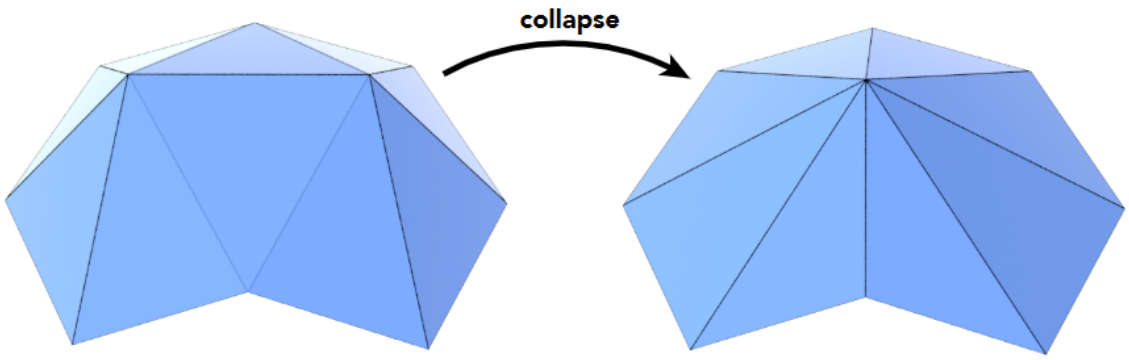
直接降低开销；不同情况的物体选用不同复杂程度的几何模型（e.g. 远处和近处）

Goal: reduce number of mesh elements
while maintaining the overall shape

How to compute?

Edge Collapse (边坍缩)

one way to simplify a mesh



把边的两点合在一起

Q：坍缩哪些边，哪些边不重要？

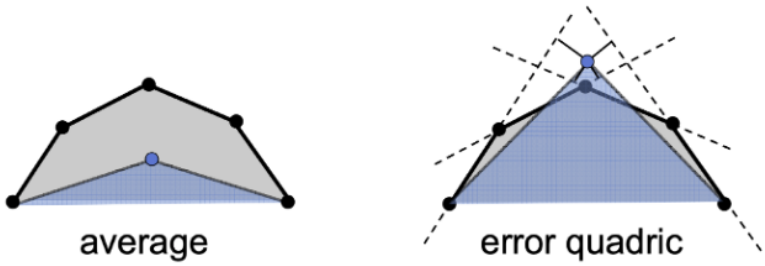
Example：把中间三个点变成一个，新的点放在哪？

引入二次误差（新的点和原来面距离的平方和），使得新的图形最小化二次误差

Quadric Error Metrics

（二次误差度量）

- How much geometric error is introduced by simplification?
- Not a good idea to perform local averaging of vertices
- Quadric error: new vertex should minimize its sum of square distance (L2 distance) to previously related triangle planes!



参考资料：link

Q: How much does it cost to collapse an edge?

Better idea: choose point that minimizes quadric error

Simplification via Quadric Error

Iteratively collapse edges

Which edges? Assign score with quadric error metric

- approximate distance to surface as sum of distances to planes containing triangles
- iteratively collapse edge with smallest score and update model
- greedy algorithm... great results!

简单的部分坍缩的多，复杂的部分坍缩的少

Mesh regularization

Modify sample distribution to improve quality, same #triangles

Shadow Mapping

光栅化遗留部分，Ray Tracing动机

光栅化难题：全局光照，阴影

An Image-space Algorithm

- no knowledge of scene's geometry during shadow computation
- must deal with aliasing artifacts

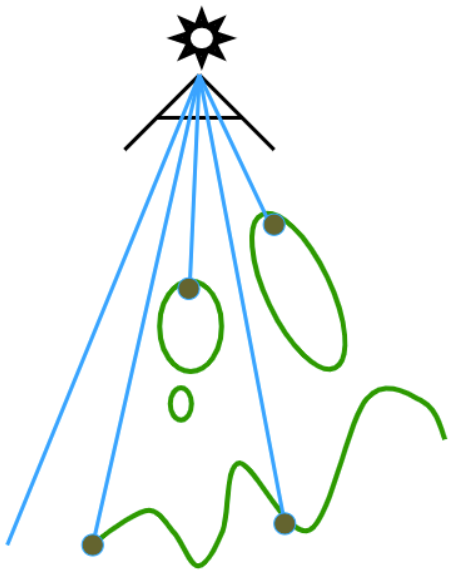
Key idea: the points NOT in shadow must be seen both **by the light** and **by the camera**

经典Shadow Mapping只能处理点光源（硬阴影：有明显阴影边界）

Pass 1: Render from Light

放置虚拟相机在光源处并做光栅化，得到光源看到点的深度图 (shadow map)

- Depth image from light source



Pass 2A: Render from Eye

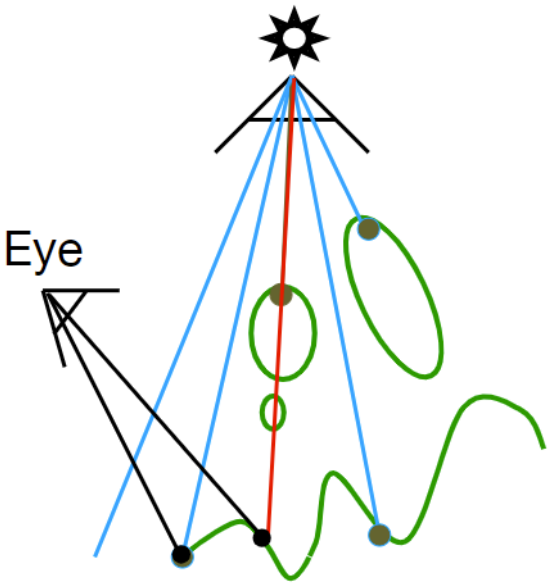
Standard image (with depth) from eye

Pass 2B: Project to light

Project visible points in eye view back to light source

reproject之后，在depth image上查询并和真实深度比较

- Project visible points in eye view back to light source



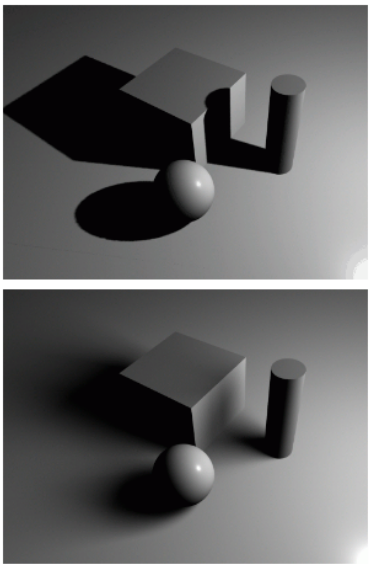
(Reprojected) depths from light and eye are not the same. BLOCKED!!

problem: 只能处理hard shadow；严重的精度问题；shadow map的分辨率过低会使得阴影走样

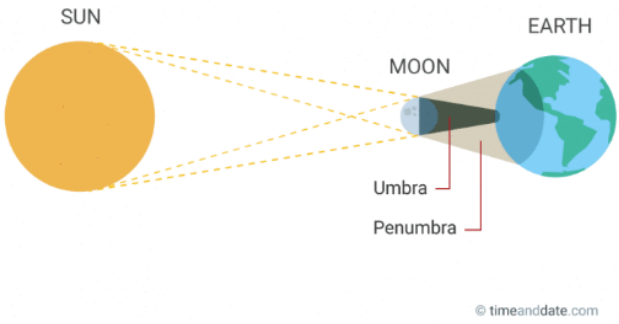
Shadow Mapping仍然是游戏的主流技术

注：游戏里的阴影质量一般指shadow map分辨率

- Hard shadows vs. soft shadows



[RenderMan]



[https://www.timeanddate.com/eclipse/umbra-shadow.html]

hard shadow边界比较锐利；soft shadow边界渐变

soft shadow = Penumbra (半影) => 光源有一定大小

umbra (本影): 完全看不到光源; penumbra: 可以看到部分光源

光线追踪（基本原理）

Why Ray Tracing?

- Rasterization couldn't (too hard) handle global effects well

  (Soft) shadows; Glossy reflection; Indirect illumination

  And especially when the light bounces more than once
- Rasterization is fast, but quality is relatively low
- Ray tracing is accurate, but is very slow

  Rasterization: **real-time**, ray tracing: **offline**

  ~10K CPU core hours to render one frame in production

Basic Ray-Tracing Algorithm

Overview

Light Rays

1. Light travels in straight lines (though this is wrong)
2. Light rays do not "collide" with each other if they cross (though this is still wrong)
3. Light rays travel from the light sources to the eye (but the physics is invariant under path reversal - **reciprocity**).

Ray Casting

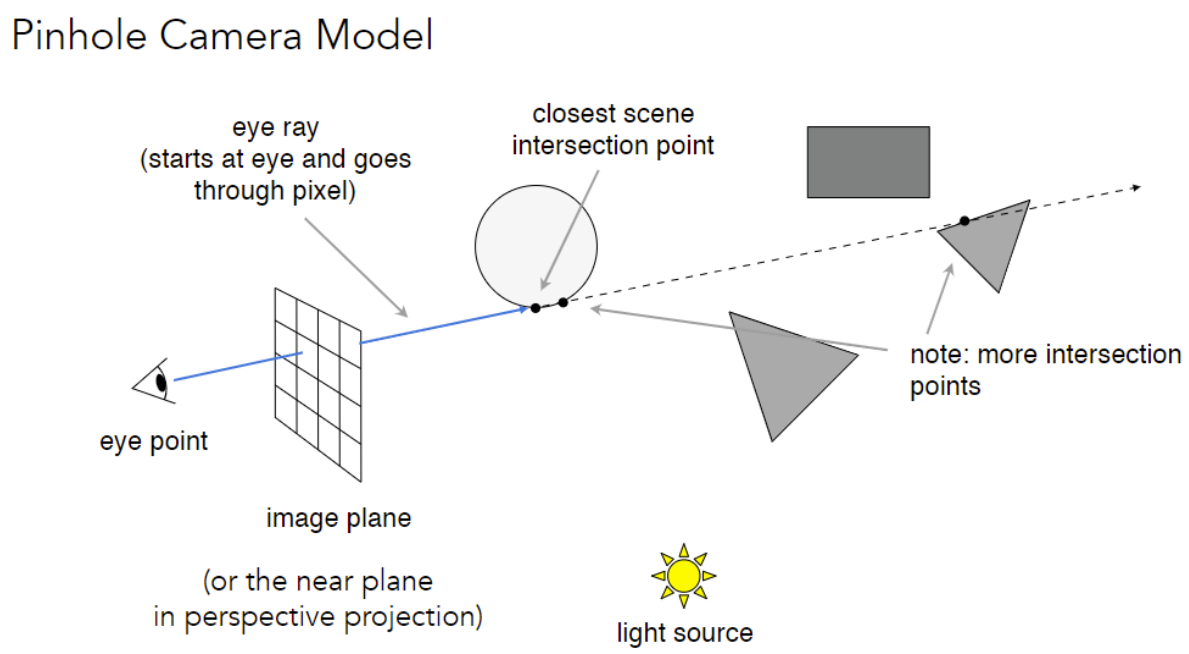
从相机开始

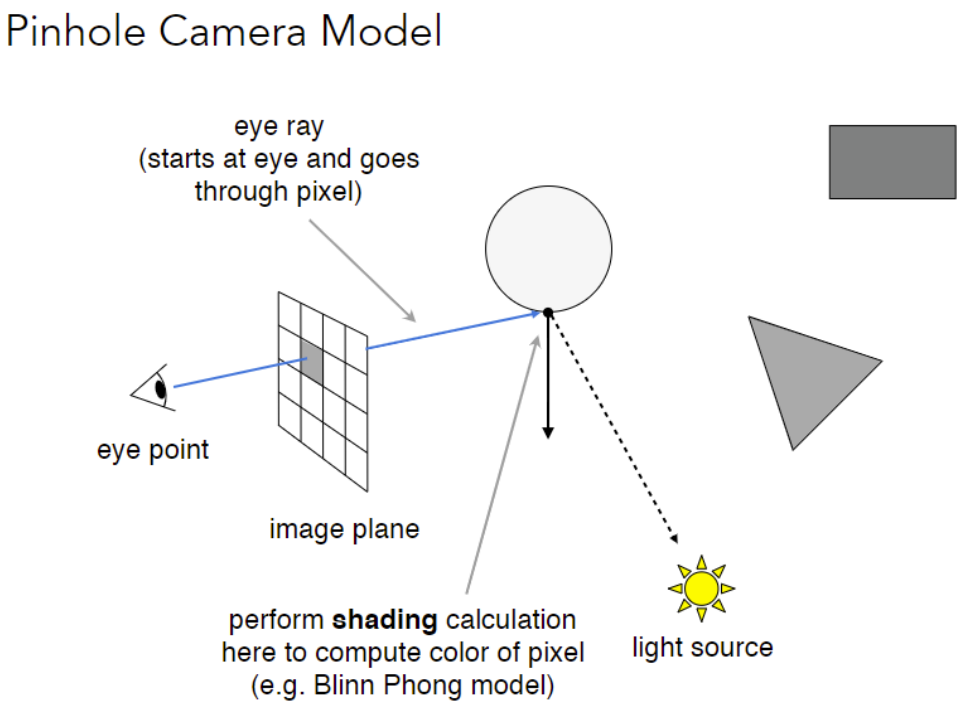
1. Generate an image by casting one ray per pixel (找closest = z-buffer)
2. Check for shadows by sending a ray to the light

假设:

- 眼睛和光源都是点
- 物体会有完美的折射和反射

效果 ≈ 光栅化

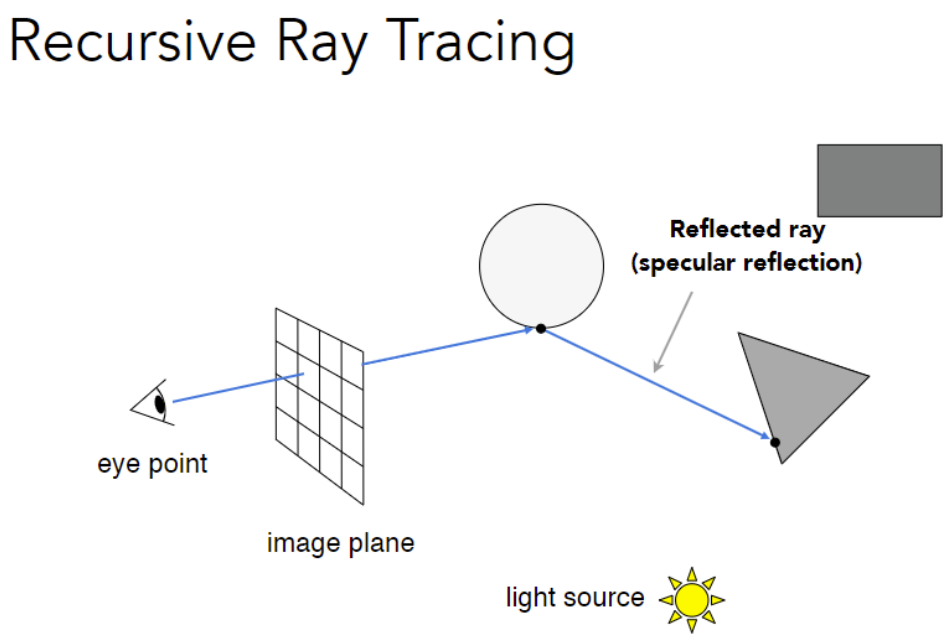


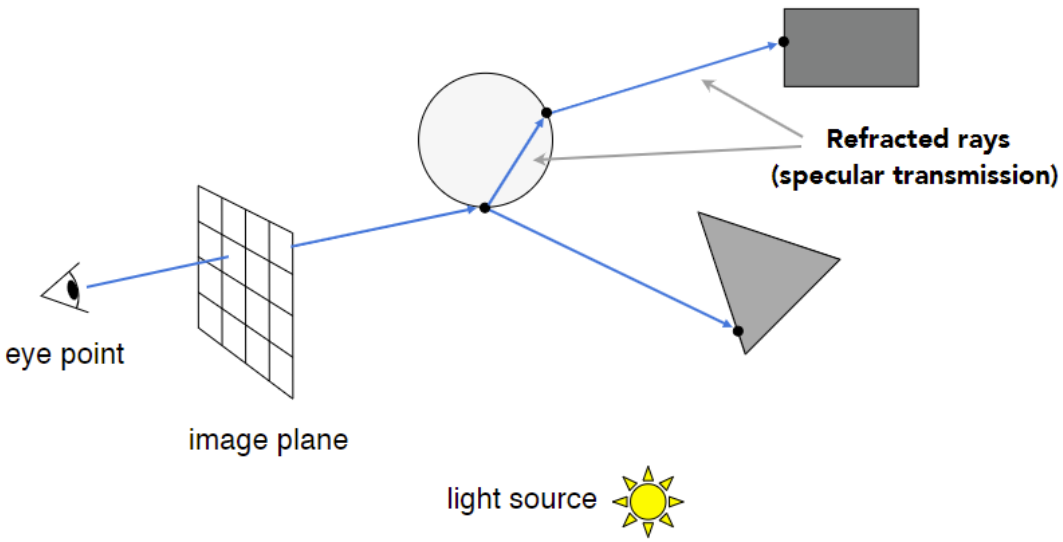


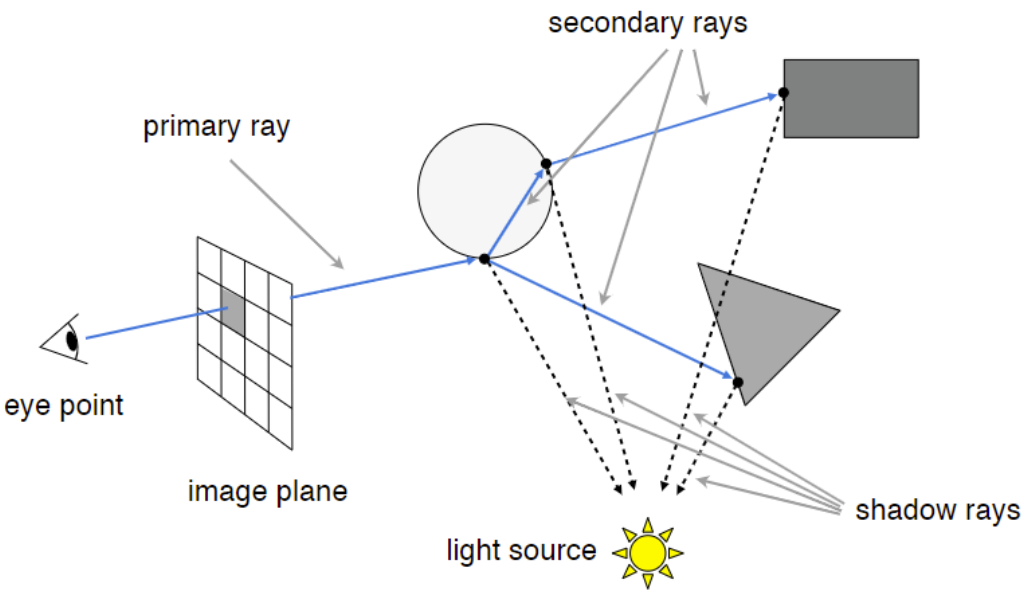
Recursive (Whitted-Style) Ray Tracing

假设：

- 眼睛和光源都是点
- 物体会有完美的折射和反射







- Always perform specular reflections / refractions
- Stop bouncing at diffuse surfaces

每一个交点都和光源连线并着色，都对pixel着色产生影响

primary ray + secondary ray + shadow ray

Details

Ray-Surface Intersection (交点)

The intersection p must satisfy both equations

Ray is defined by its origin and a direction vector

$$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d} \quad 0 \leq t < \infty$$

Ray Intersection With Sphere

$$\text{Sphere:} \quad (\mathbf{p}-\mathbf{c})^2 = R^2$$

$$(\mathbf{o}+t\mathbf{d}-\mathbf{c})^2 - R^2 = 0$$

Ray Intersection With Implicit Surface

General implicit surface: $f(\mathbf{p}) = 0$

Solve for <u>real</u>, <u>positive</u> roots: $f(\mathbf{o}+t\mathbf{d}) = 0$

Ray Intersection With Triangle Mesh (explicit)

Why?

- Rendering: visibility, shadows, lighting ...
- Geometry: inside/outside test

  点投射的光线和物体有奇数个交点，则点在物体内（物体需要是封闭的）

Simple idea: just intersect ray with <u>each</u> triangle

Simple, but slow (acceleration?)

Note: can have 0, 1 intersections with each triangle

Steps

1. Ray-<u>plane</u> intersection
2. Test if hit point is inside triangle

Ray Intersection With Plane

Plane is defined by normal vector and a point on plane

$$\mathbf{p} : \left(\mathbf{p}-\mathbf{p}'\right)\cdot\mathbf{N} = 0$$
$$ax+by+cz+d=0$$

$$\left(\mathbf{p}-\mathbf{p}'\right)\cdot\mathbf{N} = \left(\mathbf{o}+t\mathbf{d}-\mathbf{p}'\right)\cdot\mathbf{N} = 0$$
$$t = \frac{(\mathbf{p}'-\mathbf{o})\cdot\mathbf{N}}{\mathbf{d}\cdot\mathbf{N}} \qquad \text{Check: } 0 \le t < \infty$$

Möller Trumbore (MT) 算法

giving barycentric coordinate directly

使用重心坐标表示平面

$$\overrightarrow{\mathbf{O}} + t\overrightarrow{\mathbf{D}} = (1-b_1-b_2)\overrightarrow{\mathbf{P}}_0 + b_1\overrightarrow{\mathbf{P}}_1 + b_2\overrightarrow{\mathbf{P}}_2$$

解线性方程组（使用Cramer法则）

$$\begin{bmatrix} t \\ b_1 \\ b_2 \end{bmatrix} = \frac{1}{\vec{\mathbf{S}}_1 \bullet \vec{\mathbf{E}}_1} \begin{bmatrix} \vec{\mathbf{S}}_2 \bullet \vec{\mathbf{E}}_2 \\ \vec{\mathbf{S}}_1 \bullet \vec{\mathbf{S}} \\ \vec{\mathbf{S}}_2 \bullet \vec{\mathbf{D}} \end{bmatrix}$$

Cost = (1 div, 27 mul, 17 add)

Where:

$$\vec{\mathbf{E}}_1 = \vec{\mathbf{P}}_1 - \vec{\mathbf{P}}_0$$
$$\vec{\mathbf{E}}_2 = \vec{\mathbf{P}}_2 - \vec{\mathbf{P}}_0$$
$$\vec{\mathbf{S}} = \vec{\mathbf{O}} - \vec{\mathbf{P}}_0$$
$$\vec{\mathbf{S}}_1 = \vec{\mathbf{D}} \times \vec{\mathbf{E}}_2$$
$$\vec{\mathbf{S}}_2 = \vec{\mathbf{S}} \times \vec{\mathbf{E}}_1$$

判断$t \geq 0$，重心坐标非负

Accelerating Ray-Surface Intersection

Performance Challenges

Simple ray-scene intersection

- Exhaustively test ray-intersection with every triangle
- Find the closest hit (i.e. minimum t)

Naive algorithm = #pixels × # traingles (× #bounces)

For generality, we use the term objects instead of triangles later (but doesn't necessarily mean entire objects)

Bounding Volumes

bound complex object with a simple volume

- Object is fully contained in the volume
- If it doesn't hit the volume, it doesn't hit the object
- So test BVol first, then test object if it hits

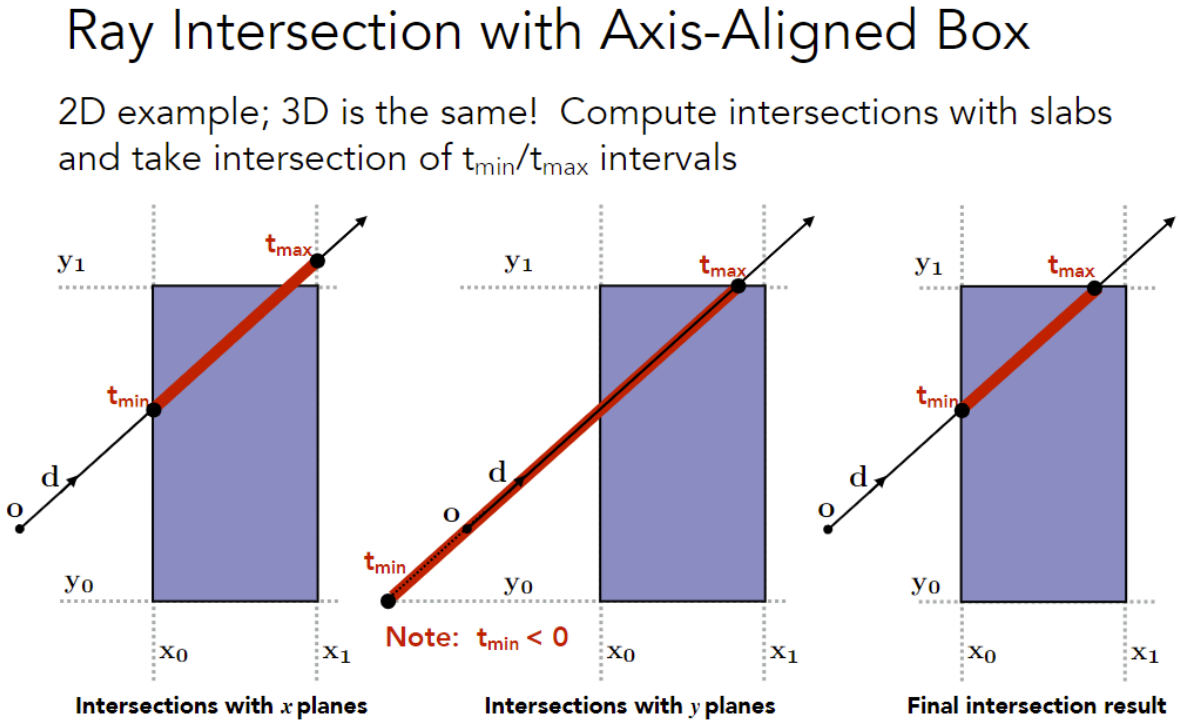
AABB (Axis-Aligned Bounding Box)

Understanding: box是三组对面空间的交集

We often use an Axis-Aligned Bounding Box (AABB)

Q: Why Axis-Aligned ?

简化线和面交点的计算，e.g. $t = (\mathbf{p}'_x - \mathbf{o}_x)/\mathbf{d}_x$，带入坐标即可



Recall: a box (3D) = three pairs of infinitely large slabs

Key ideas

- The ray enters the box <u>only when</u> it enters all pairs of slabs
- The ray exits the box <u>as long as</u> it exits any pair of slabs

For each pair, calculate the tmin and tmax (negative is fine)

For the 3D box, $t_{enter} = \max\{t_{min}\}, t_{exit} = \min\{t_{max}\}$

If $t_{enter} < t_{exit}$, we know the ray stays a while in the box(so they must intersect!)

- What if $t_{exit} < 0$?

  The box is “behind” the ray — no intersection!

- What if $t_{exit} >= 0$ and $t_{enter} < 0$?

  The ray’s origin is inside the box — have intersection!

In **summary**: ray and AABB intersect iff $t_{enter} <= t_{exit}$ && $t_{exit} >= 0$

光线追踪（加速结构）

Announcements from Prof. Yan.

DLSS 2.0: link

RTXGI: link

- Offline rendering techniques will soon become real-time
- Current real-time rendering techniques will still be useful

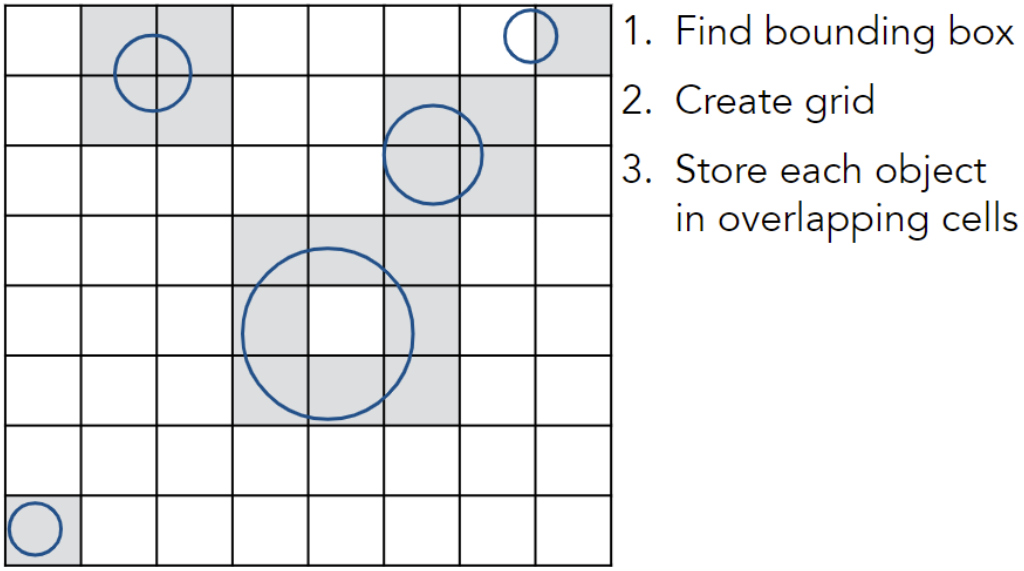
WHY, WHAT, then HOW (HOW is less important and more likely to forget)

Using AABBs to accelerate ray tracing

Uniform grids

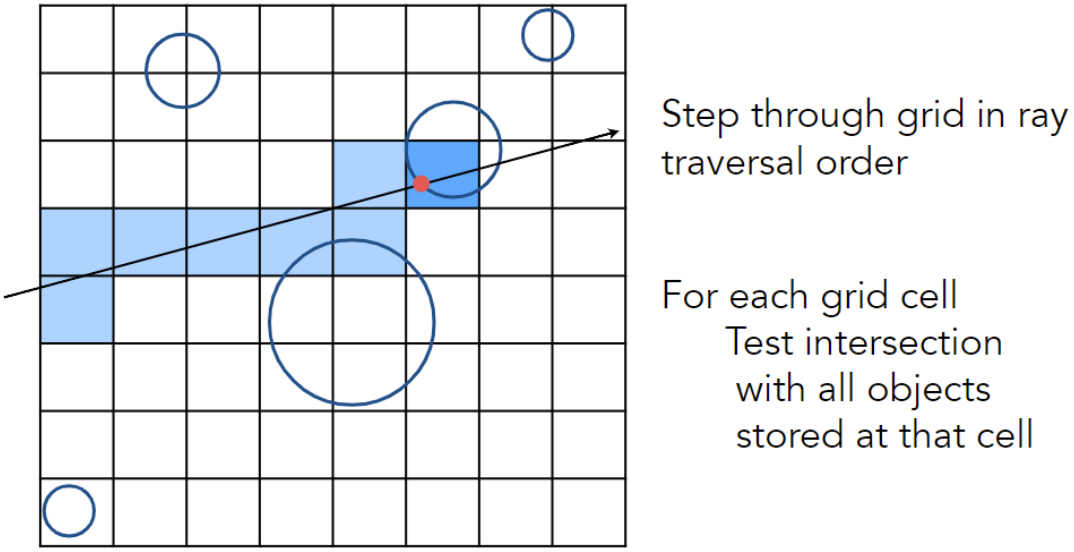
1. Build Acceleration Grid

   对场景的预处理（在RT之前）



2. Ray-Scene Intersection

   假设ray和grid求交比较快，和物体求交比较慢（实际上也是）



   Q: 如何判断光传播经过哪些grid?

   下一个grid在右边或上面（~线的光栅化）

Grid Resolution?

- One cell

  No speedup

- Too many cells

  Inefficiency due to extraneous grid traversal

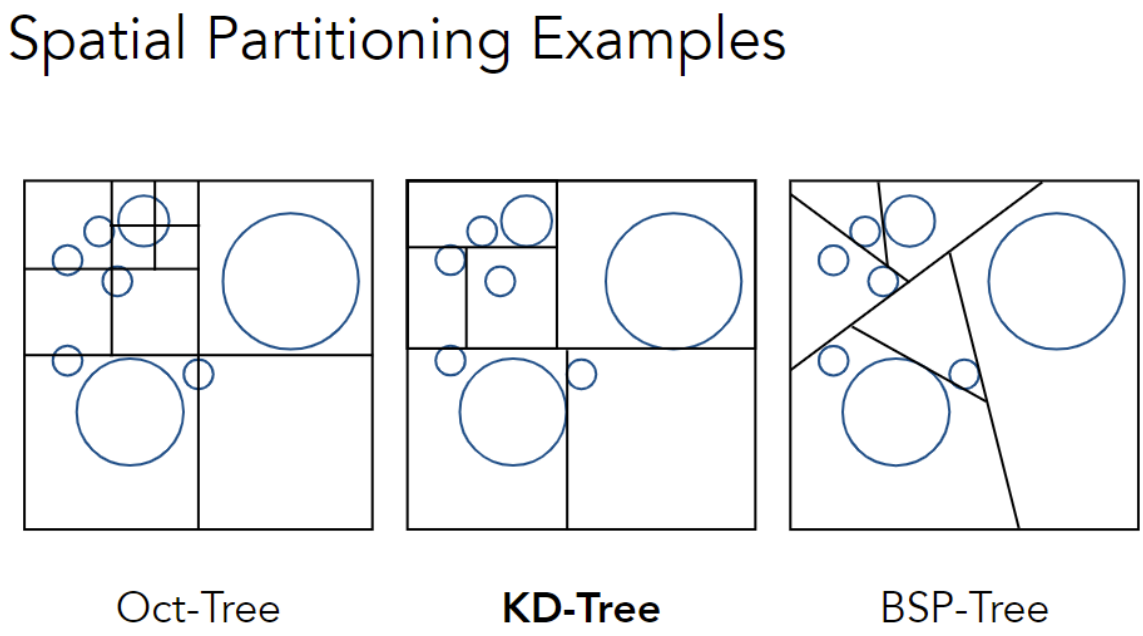
Heuristic: #cells = C * #objs, C ≈ 27 in 3D

Grids work well on large collections of objects that are distributed evenly in size and space, 处理稀疏场景效果不好

Spatial partitions

对空间做划分

对场景的预处理（在RT之前）



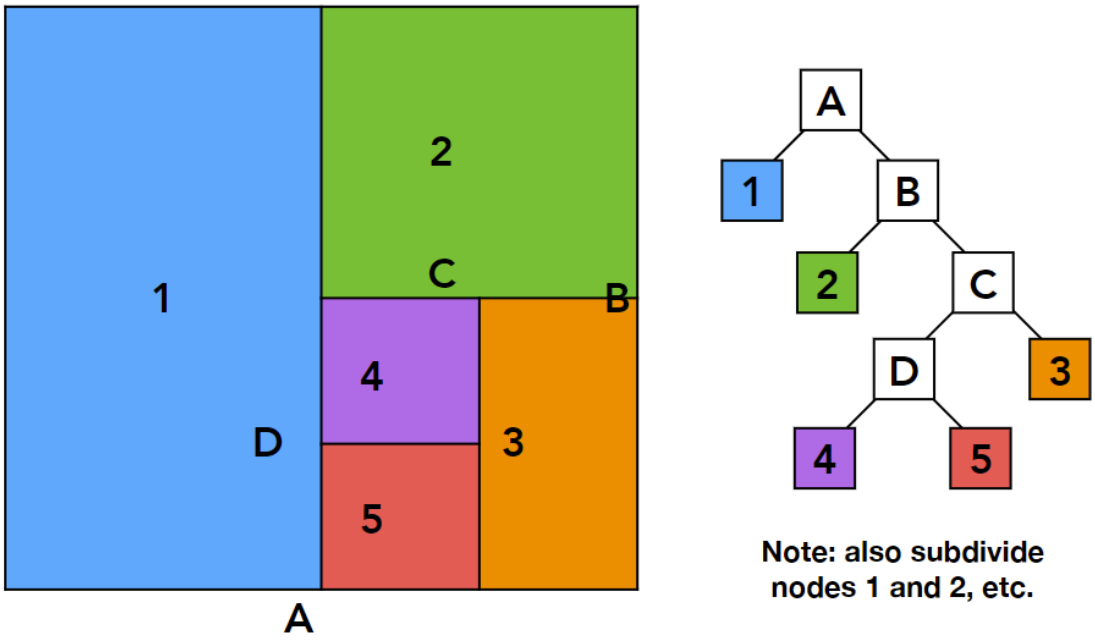
Note: you could have these in both 2D and 3D. In lecture we will illustrate principles in 2D.

Oct-Tree: 对于所有坐标轴切一刀，2^n划分

KD-Tree: 使用某一个坐标轴划分，一般交替或者随机选取

BSP-Tee: 每次对空间切一刀（非Axis-Aligned）

KD-Tree



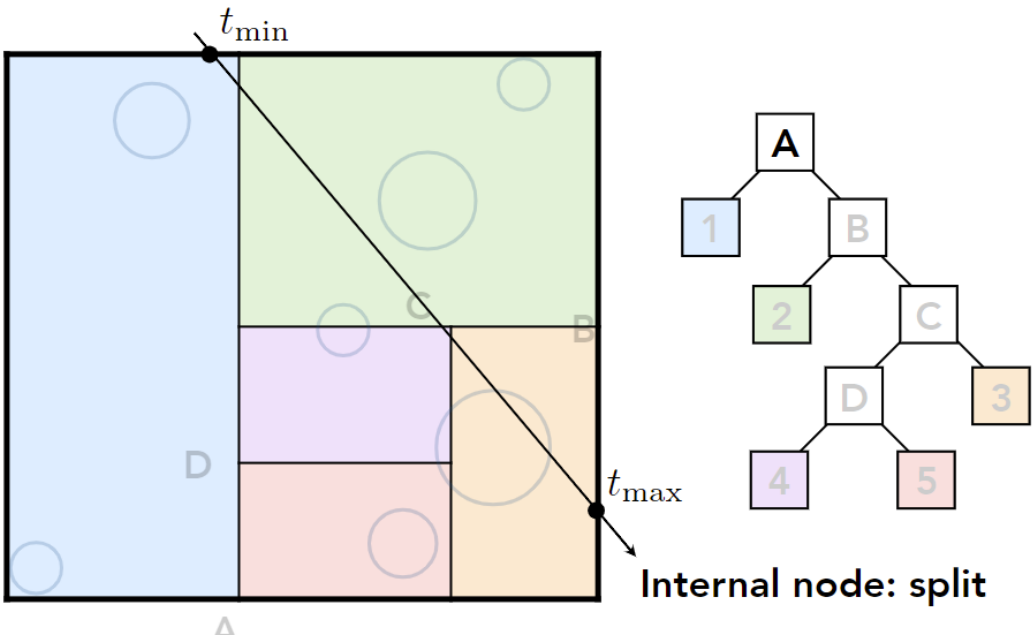
- 叶子节点存储物体
- 中间节点

  split axis, split position, children

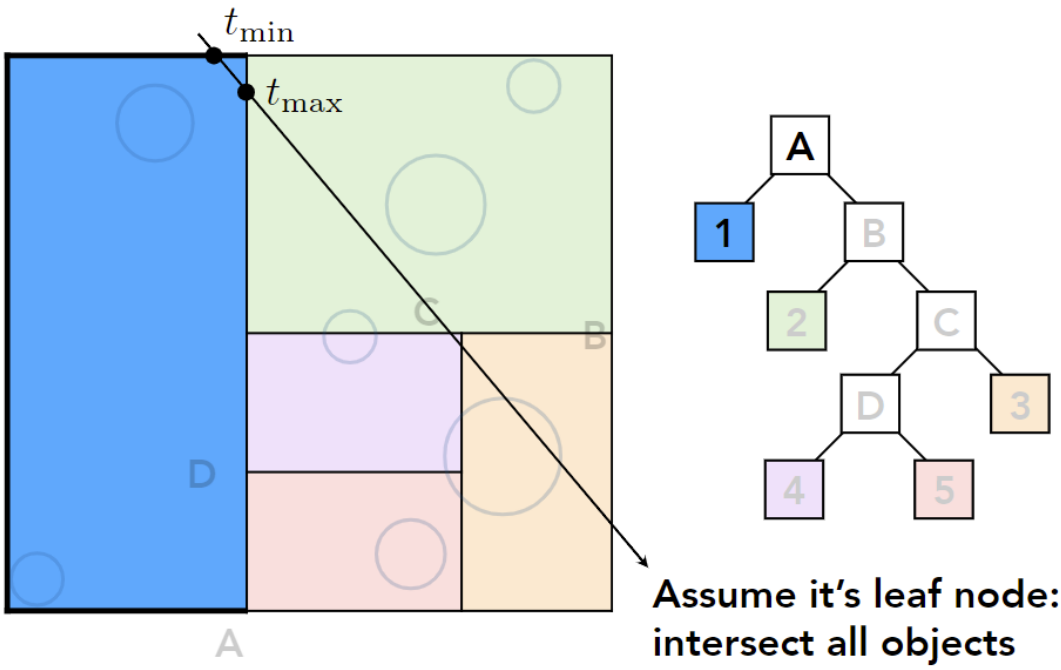
  NO objects are stored in internal nodes

Traversing a KD-Tree

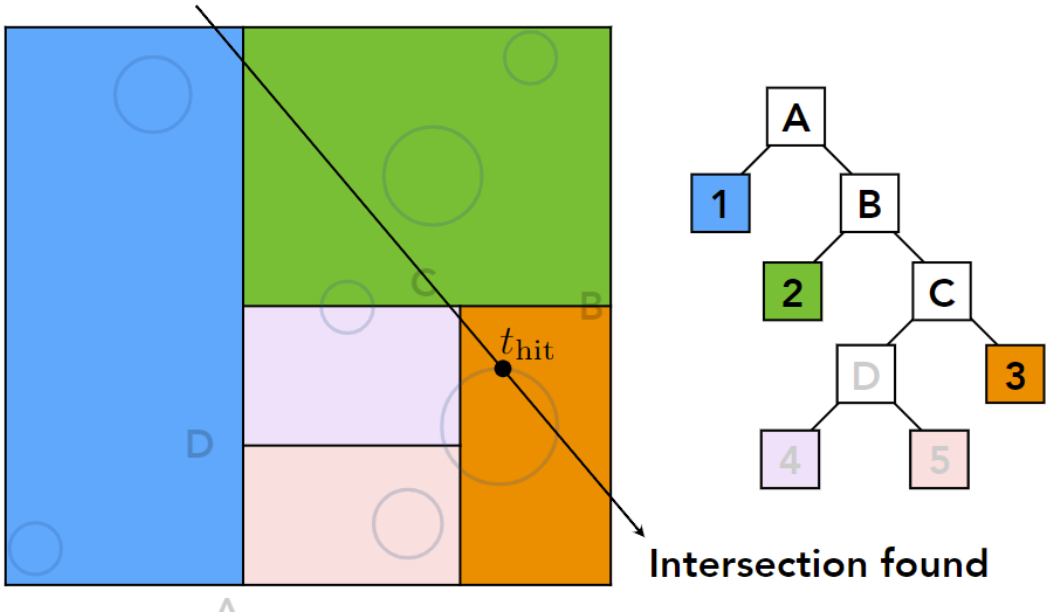
- Internal node: split



- Leaf node: intersect all objects



- repeat



过程

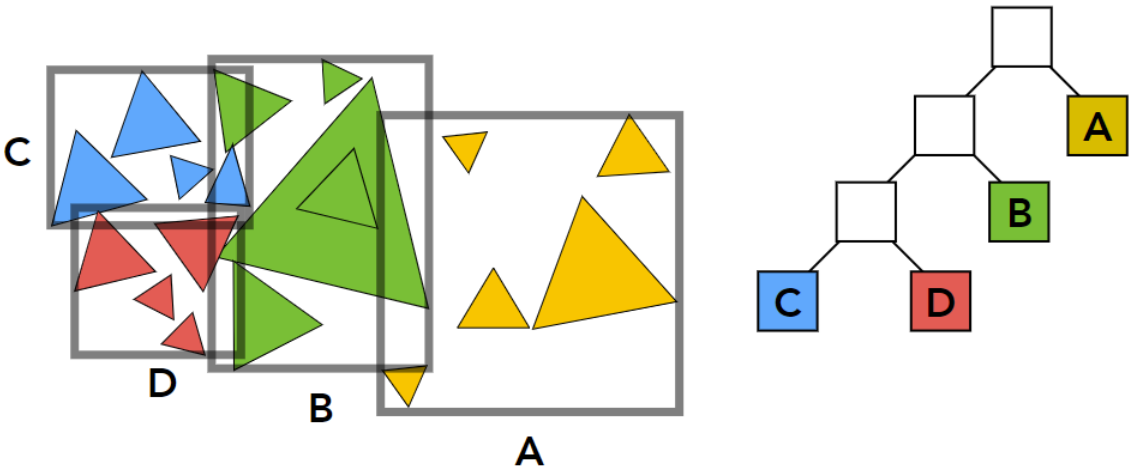
1. 和node没有交点，结束
2. 和node有交点，递归判断children (objects)

problem:

- 怎么判定物体和BB是否相交，哪些物体在BB内？
- 物体可能和不同的BB相交，需要记录在多个叶子节点

Object Partitions & Bounding Volume Hierarchy (BVH)

对物体做划分



- 一个物体只会出现在一个BB里
- BB之间可以相交

过程：

1. Find bounding box
2. Recursively split set of objects in two subsets
3. **Recompute** the bounding box of the subsets

4. Stop when necessary
5. Store objects in each leaf node

How to subdivide a node?

- Choose a dimension to split

  希望分布能够更加均匀
- Heuristic #1: Always choose the longest axis in node
- Heuristic #2: Split node at location of median object (不需要排序)

Termination criteria?

- Heuristic: stop when node contains few elements (e.g. 5)

注：不支持场景修改

Data Structure for BVHs

- Internal nodes store

  Bounding box

  Children: pointers to child nodes
- Leaf nodes store

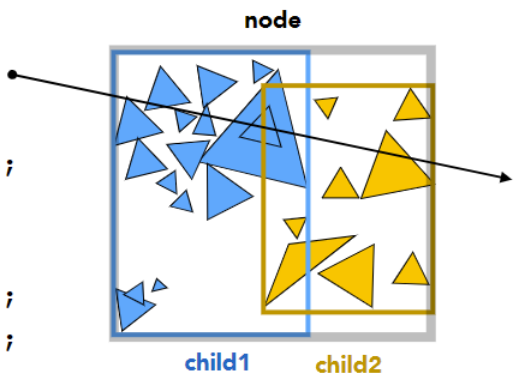
  Bounding box

  List of objects

Nodes represent subset of primitives in scene (All objects in subtree)

BVH Traversal

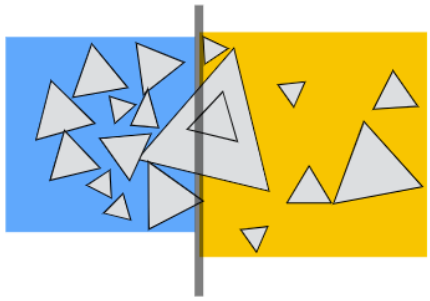
```
Intersect(Ray ray, BVH node) {
  if (ray misses node.bbox) return;

  if (node is a leaf node)
     test intersection with all objs;
     return closest intersection;

  hit1 = Intersect(ray, node.child1);
  hit2 = Intersect(ray, node.child2);

  return the closer of hit1, hit2;
}
```



Spatial vs Object Partitions

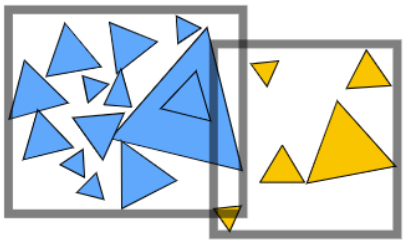
Spatial partition (e.g.KD-tree)

- Partition space into non-overlapping regions
- An object can be contained in multiple regions



Object partition (e.g. BVH)

- Partition set of objects into disjoint subsets
- Bounding boxes for each set may overlap in space



Basic radiometry (辐射度量学)

new topics from now on!

精确定义光，光如何和物体作用，光线如何传播等

Motivation

- Light intensity I is 10, for example. But 10 what?
- Do you think Whitted style ray tracing gives you CORRECT results?

  没有考虑折射能量损失，考虑反射时能量不守恒
- Blinn-Phong model is NOT accurate

Radiometry

- Measurement system and units for illumination
- Accurately measure the spatial properties of light
  - New terms: Radiant flux, intensity, irradiance, radiance

  基于几何光学，不考虑波动性
- Perform lighting calculations in a physically correct manner

Radiant Energy and Flux (Power)

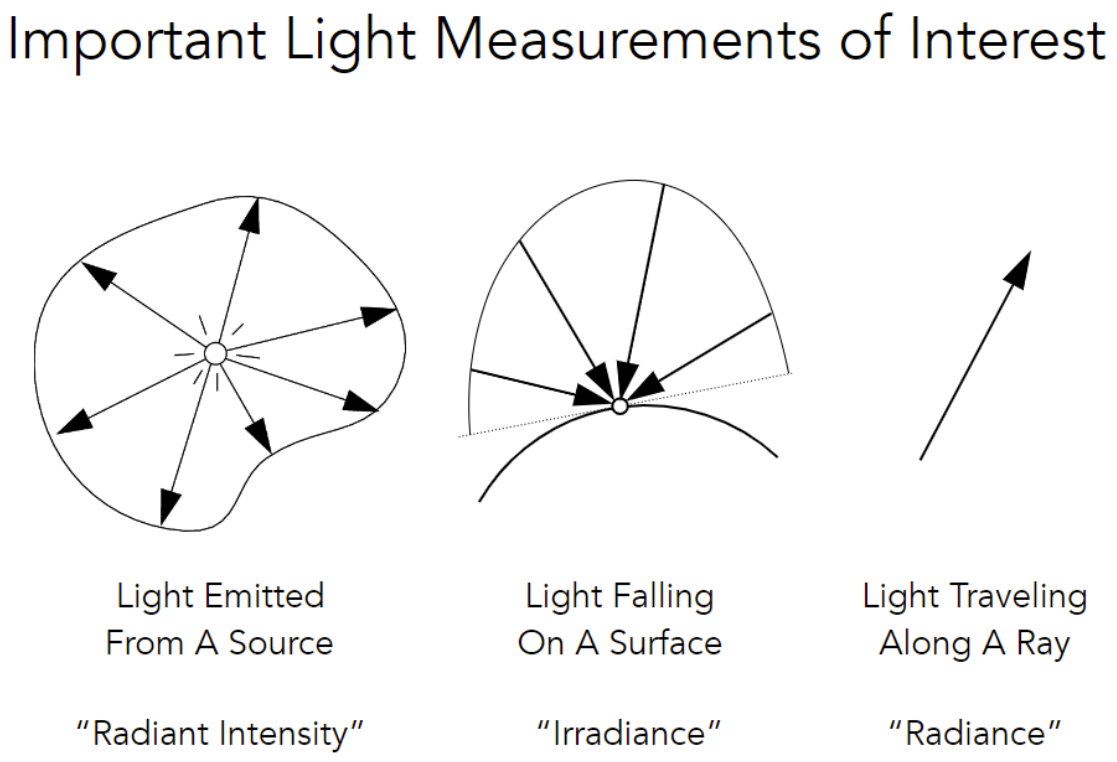
Definition: Radiant energy is the energy of electromagnetic radiation. It is measured in units of joules, and denoted by the symbol:

$$Q\,[\,\mathrm{J} = \mathrm{J_{oule}}\,]$$

Definition: Radiant flux (power) is the energy emitted, reflected, transmitted or received, per unit time. (光源有多亮)

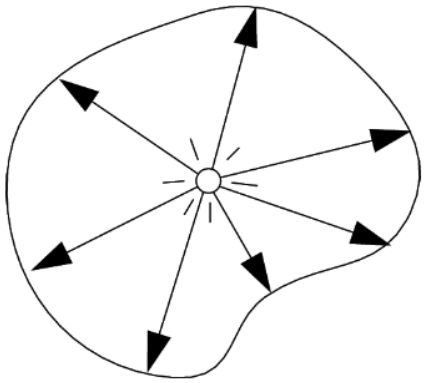
$$\Phi \equiv \frac{\mathrm{d}Q}{\mathrm{d}t}[\,\mathrm{W} = \mathrm{Watt}][\mathrm{lm} = \ \mathrm{lumen}\,]^{\star}$$

Flux – #photons (光子) flowing through a sensor in unit time



Radiant Intensity

Definition: The radiant (luminous) intensity is the power per unit solid angle (立体角) emitted by a point light source.



$$I(\omega) \equiv \frac{\mathrm{d}\Phi}{\mathrm{d}\omega}$$

$$\left[\frac{\mathrm{W}}{\mathrm{sr}}\right] \left[\frac{\mathrm{lm}}{\mathrm{sr}} = \mathrm{cd} = \mathrm{candela}\right]$$

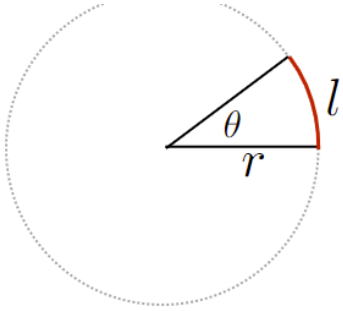
The candela is one of the seven SI base units.

球形均匀光源：$I = \Phi/4\pi$

Solid Angles

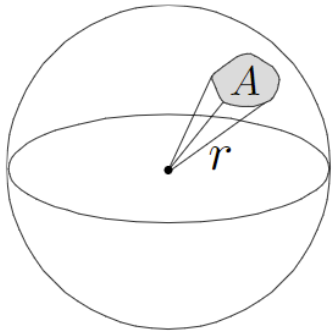
Angle: ratio of subtended arc length on circle to radius

- $\theta = \dfrac{l}{r}$
- **Circle has 2π radians**



Solid angle: ratio of subtended area on sphere to radius squared

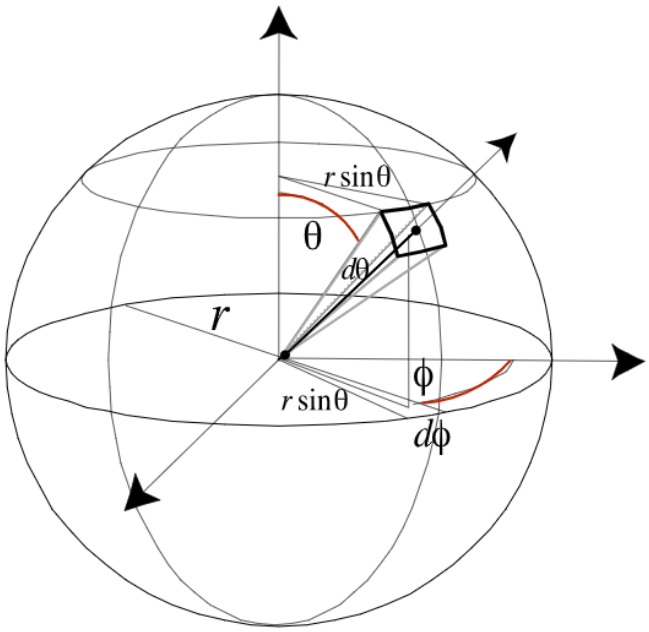
- $\Omega = \dfrac{A}{r^2}$
- **Sphere has 4π steradians**



微分立体角

$\mathrm{d}\omega$随着$\mathrm{d}\theta, \mathrm{d}\phi$的变化情况

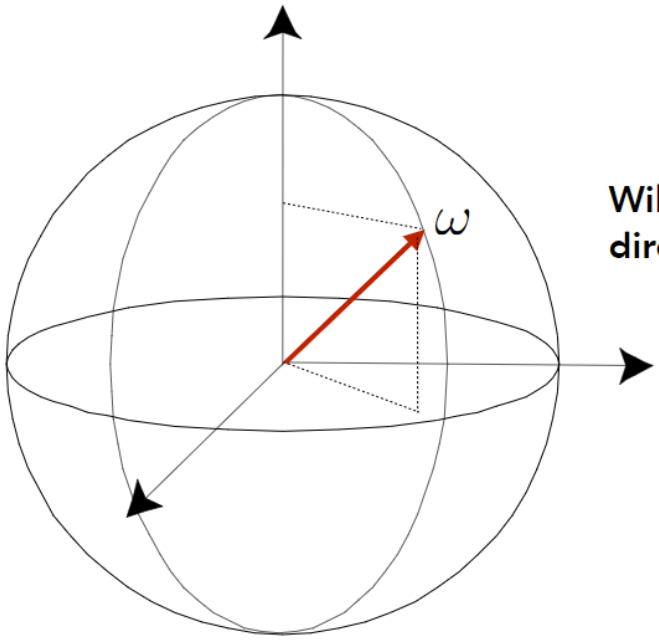
注：和$\sin\theta$有关，不均匀



$$\begin{aligned}\mathrm{d}A =&(r\,\mathrm{d}\theta)(r\sin\theta\,\mathrm{d}\phi)\\ =&r^2\sin\theta\,\mathrm{d}\theta\,\mathrm{d}\phi\end{aligned}$$

$$\mathrm{d}\omega = \frac{\mathrm{d}A}{r^2} = \sin\theta\,\mathrm{d}\theta\,\mathrm{d}\phi$$

ω as a direction vector



Will use ω to denote a direction vector (unit length)

注：dω立体角和这里的ω没有关系

光线追踪（辐射度量学、渲染方程与全局光照）

Radiometry

Radiant energy Q $[\mathrm{J} = \mathrm{Joule}]$ (barely used in CG)

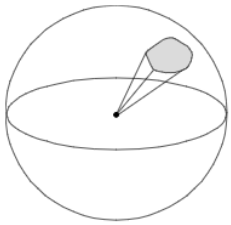
- the energy of electromagnetic radiation

Radiant flux (power) $\Phi \equiv \dfrac{\mathrm{d}Q}{\mathrm{d}t}$ $[\mathrm{W} = \mathrm{Watt}]$ $[\mathrm{lm} = \mathrm{lumen}]$

- Energy per unit time

Radiant intensity $I(\omega) \equiv \dfrac{\mathrm{d}\Phi}{\mathrm{d}\omega}$

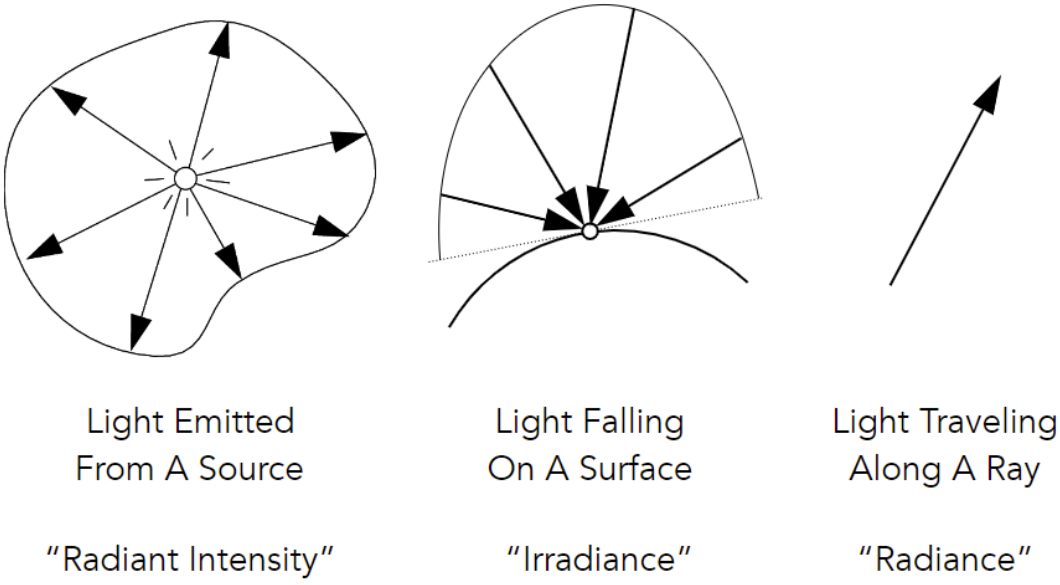
- power per unit solid angle



Solid Angle $\Omega = \dfrac{A}{r^2}$

- ratio of subtended area on sphere to radius squared

Important Light Measurements of Interest



Irradiance

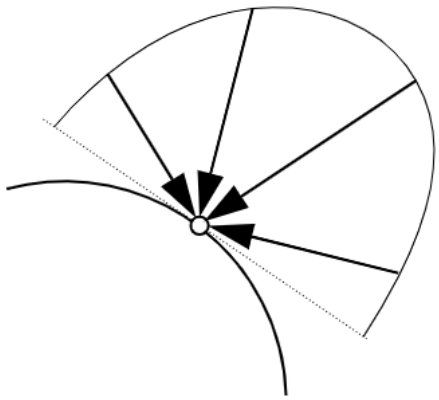
Definition: The irradiance is the power per unit area incident on a surface point.

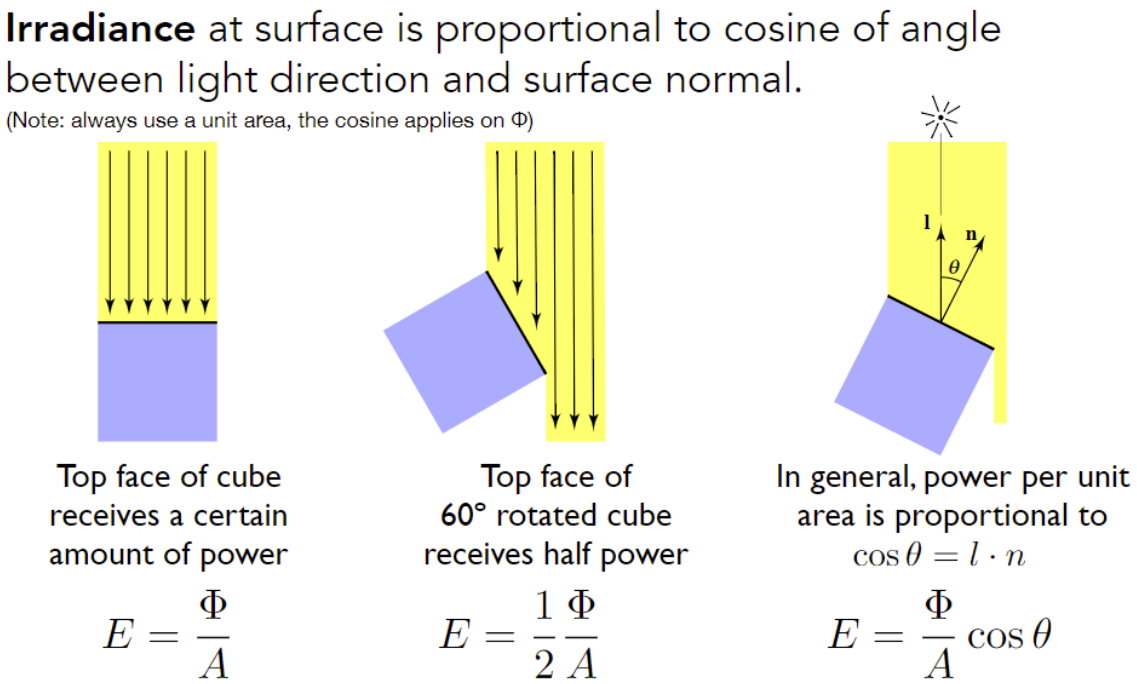
区分：Intensity是per unit solid angle

注：area需要垂直（投影）于光线

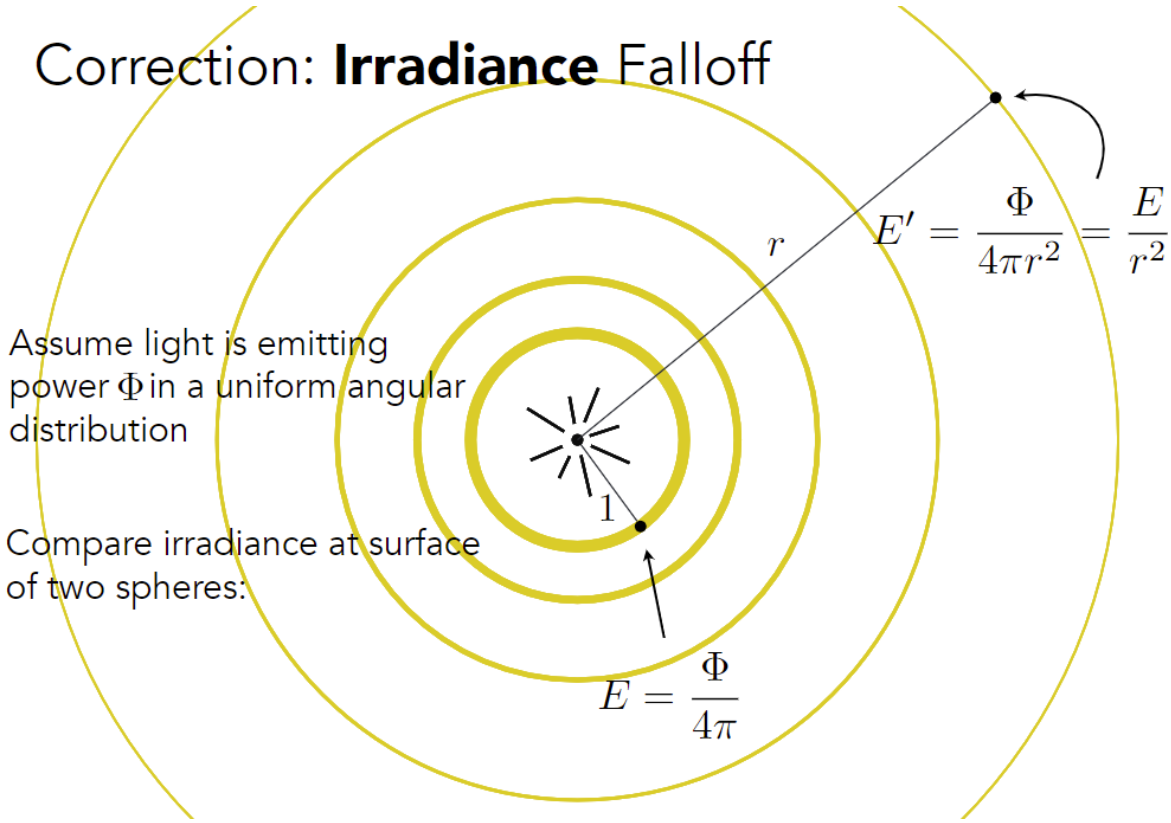
$$E(\mathbf{x}) \equiv \frac{\mathrm{d}\Phi(\mathbf{x})}{\mathrm{d}A}$$

$$\left[\frac{\mathrm{W}}{\mathrm{m}^2}\right] \quad \left[\frac{\mathrm{lm}}{\mathrm{m}^2} = \mathrm{lux}\right]$$





Lambert's Cosine Law正是反映Irradiance



注：Intensity没有衰减，只有Irradiance在衰减

Radiance

注：描述光线属性，和$\mathrm{d}\omega, \mathrm{d}A, \theta$无关

注：可以理解为每根光线携带的能量

Radiance is the fundamental field quantity that describes the distribution of light in an environment

- Radiance is the quantity associated with a ray
- Rendering is all about computing radiance Light Traveling Along A Ray

Definition: The radiance (luminance) is the power emitted, reflected, transmitted or received by a surface, per unit solid angle, per projected unit area. (两次微分)

$$L(\mathrm{p},\omega) \equiv \frac{\mathrm{d}^2\Phi(\mathrm{p},\omega)}{\mathrm{d}\omega\,\mathrm{d}A\cos\theta}$$

$\cos\theta$ accounts for projected surface area

$$\left[\frac{\mathrm{W}}{\mathrm{sr\,m}^2}\right]\left[\frac{\mathrm{cd}}{\mathrm{m}^2} = \frac{\mathrm{lm}}{\mathrm{sr\,m}^2} = \mathrm{nit}\right]$$

Recall

- Irradiance: power per projected unit area
- Intensity: power per solid angle

So

- Radiance: Irradiance per solid angle

  Irradiance = $\mathrm{d}A$面积的能量 辐射向$\mathrm{d}\omega$方向的Intensity (Irradiance的Intensity)

Exiting Radiance

Exiting surface radiance is the intensity per unit projected area leaving the surface.

$$L(\mathrm{p},\omega) = \frac{\mathrm{d}I(\mathrm{p},\omega)}{\mathrm{d}A\cos\theta}$$

e.g. for an area light it is the light emitted along a given ray (point on surface and exit direction).

- Radiance: Intensity per projected unit area

  Intensity = $\mathrm{d}\omega$方向的能量 打到$\mathrm{d}A$面积的Irradiance (Intensity的Irradiance)

Incident Radiance

Incident radiance is the irradiance per unit solid angle arriving at the surface.

$$L(\mathrm{p},\omega) = \frac{\mathrm{d}E(\mathrm{p})}{\mathrm{d}\omega\cos\theta}$$

i.e. it is the light arriving at the surface along a given ray (point on surface and incident direction).

Irradiance vs. Radiance

Irradiance有方向性，Radiance可以是双向的

$\mathrm{d}E(p,\omega)$为p点来自ω方向的Irradiance，$E(p)$为p点的总irradiance

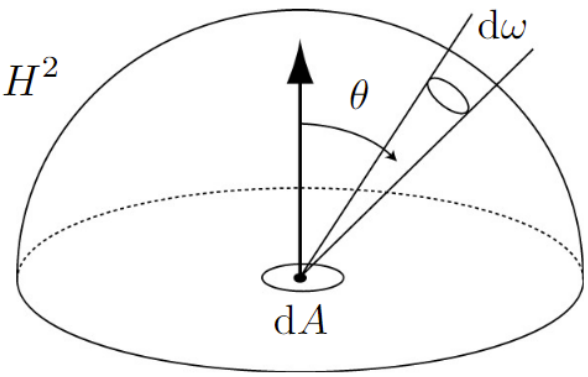
Irradiance: total power received by area dA

Radiance: power received by area dA from "direction" dω

$$dE(\mathrm{p},\omega) = L_i(\mathrm{p},\omega)\cos\theta\,\mathrm{d}\omega$$

$$E(\mathrm{p}) = \int_{H^2} L_i(\mathrm{p},\omega)\cos\theta\,\mathrm{d}\omega$$

Unit Hemisphere: H^2

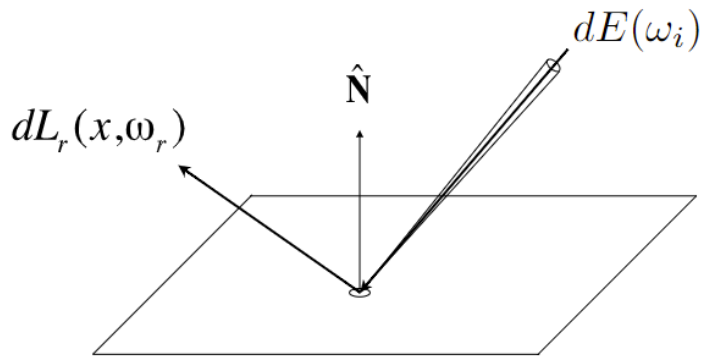


Light transport

Reflection at a Point

Radiance from direction ω_i turns into the power E that dA receives

Then power E will become the radiance to any other direction ω_o



Differential irradiance incoming: $dE(\omega_i) = L(\omega_i)\, cos\, \theta_i\, d\omega_i$

Differential radiance exiting (due to $dE(\omega_i)$): $dL_r(\omega_r)$

反射 = 入射光被物体吸收后再发射 (能量守恒)

入射radiance在dA转化成power，power再辐射到其它方向

考虑dA的irradiance

- incoming irradiance

$$dE\left(\omega_i\right) = L\left(\omega_i\right) \cos\theta_i d\omega_i$$

- radiance exiting

  irradiance被转化成radiance辐射出去

$$\mathrm{d}L_r\left(\omega_r\right)$$

  注:

$$\mathrm{d}L_r\left(\omega_r\right) \neq L\left(\omega_o\right) \cos\theta_o d\omega_o$$

$$\int \mathrm{d}L_r\left(\omega_r\right) = L_r\left(\omega_r\right)$$

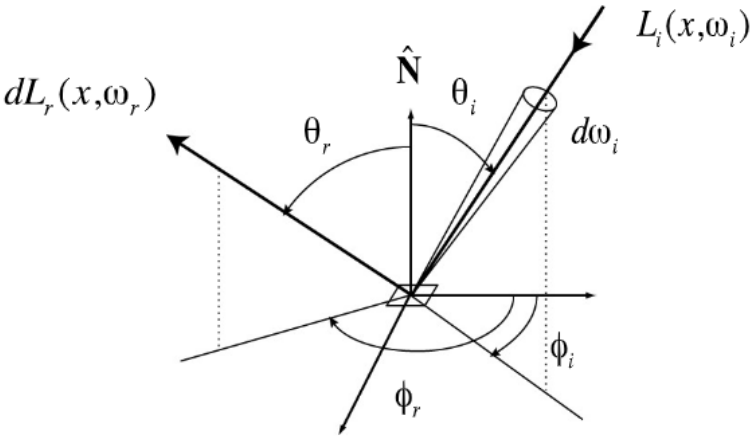
$$\int L\left(\omega_o\right) \cos\theta_o d\omega_o = \int L\left(\omega_i\right) \cos\theta_i d\omega_i$$

problem：不知道irradiance变成radiance出射的能量分布

Bidirectional Reflectance Distribution Function (BRDF)

定义往各个方向分配多少能量，描述光线和物体如何作用，定义不同材质

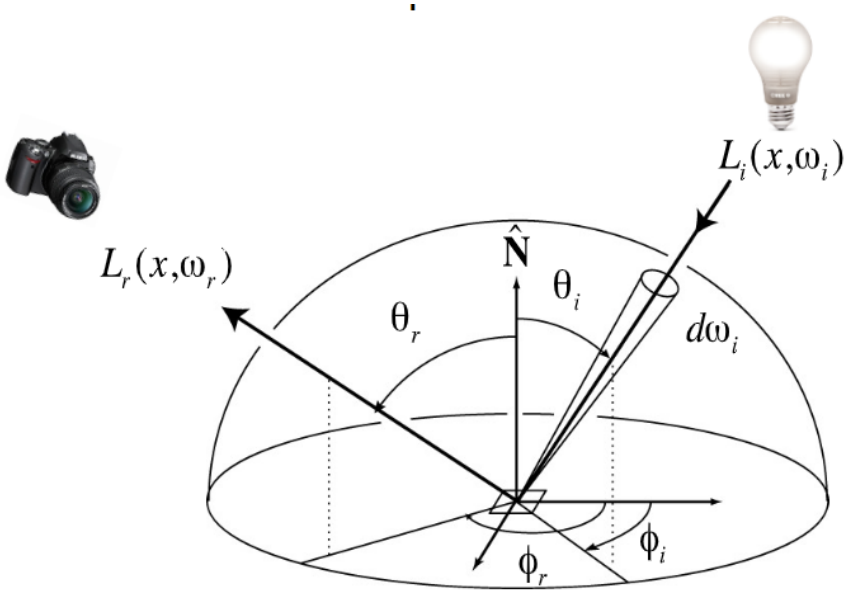
BRDF represents how much light is reflected into each outgoing direction ω_r from each incoming direction



$$f_r(\omega_i \to \omega_r) \equiv \frac{\mathrm{d}L_r(\omega_r)}{\mathrm{d}E_i(\omega_i)} = \frac{\mathrm{d}L_r(\omega_r)}{L_i(\omega_i)\cos\theta_i\,\mathrm{d}\omega_i}$$

The Reflection Equation

固定出射方向，考虑所有方向的入射光

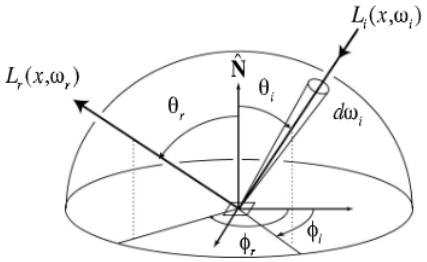


$$L_r(\mathrm{p},\omega_r) = \int_{H^2} f_r(\mathrm{p},\omega_i \to \omega_r)\, L_i(\mathrm{p},\omega_i)\, \cos\theta_i\, \mathrm{d}\omega_i$$

Challenge: Recursive Equation

Reflected radiance depends on incoming radiance

$$\boxed{L_r(\mathrm{p},\omega_r)} = \int_{H^2} f_r(\mathrm{p},\omega_i \to \omega_r)\boxed{L_i(\mathrm{p},\omega_i)}\cos\theta_i\,\mathrm{d}\omega_i$$



But incoming radiance depends on reflected radiance (at another point in the scene)

入射光可以是来自其它物体的反射光

Q：光线在某些物体循环弹射，递归死循环？

Q：不考虑循环（但是好像一定有循环），整个过程应该是始于光源的DAG？

The Rendering (渲染) Equation

adding an Emission term (物体发光)

出射光 = 自己发光 + 所有根据BRDF的反射光

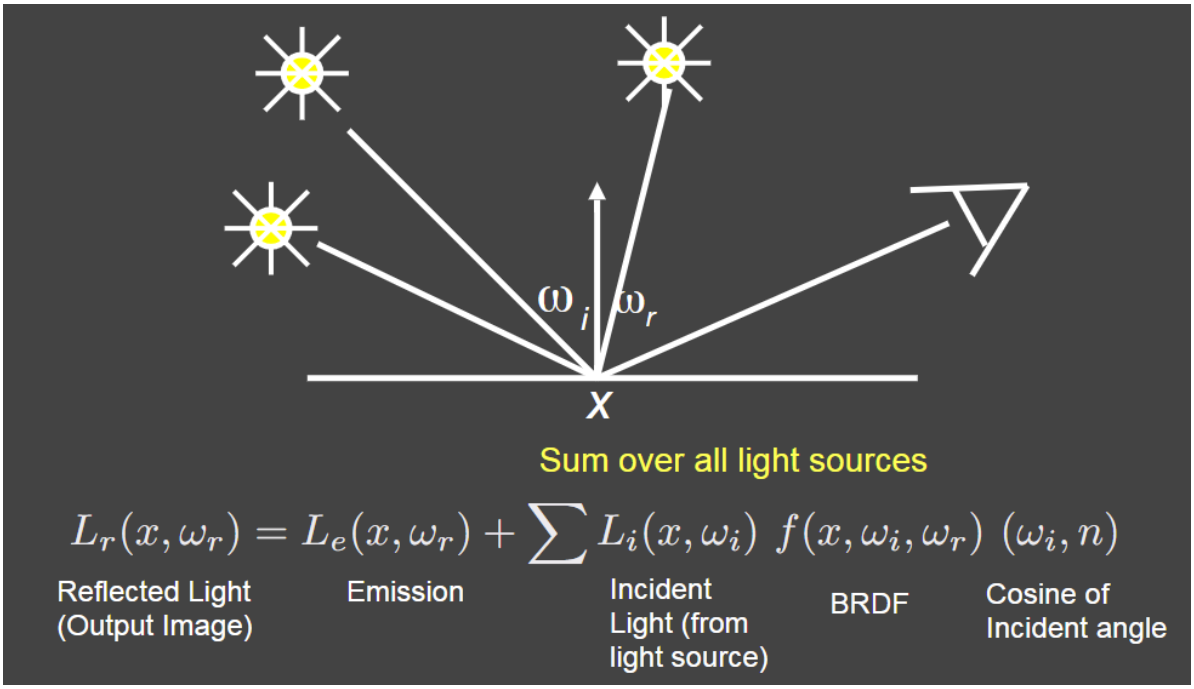
$$L_o\left(p,\omega_o\right) = L_e\left(p,\omega_o\right) + \int_{\Omega^+} L_i\left(p,\omega_i\right) f_r\left(p,\omega_i,\omega_o\right)\left(n\cdot\omega_i\right)\mathrm{d}\omega_i$$

注：假定所有方向向量都朝外；积分域为Ω^+不考虑下面来的光

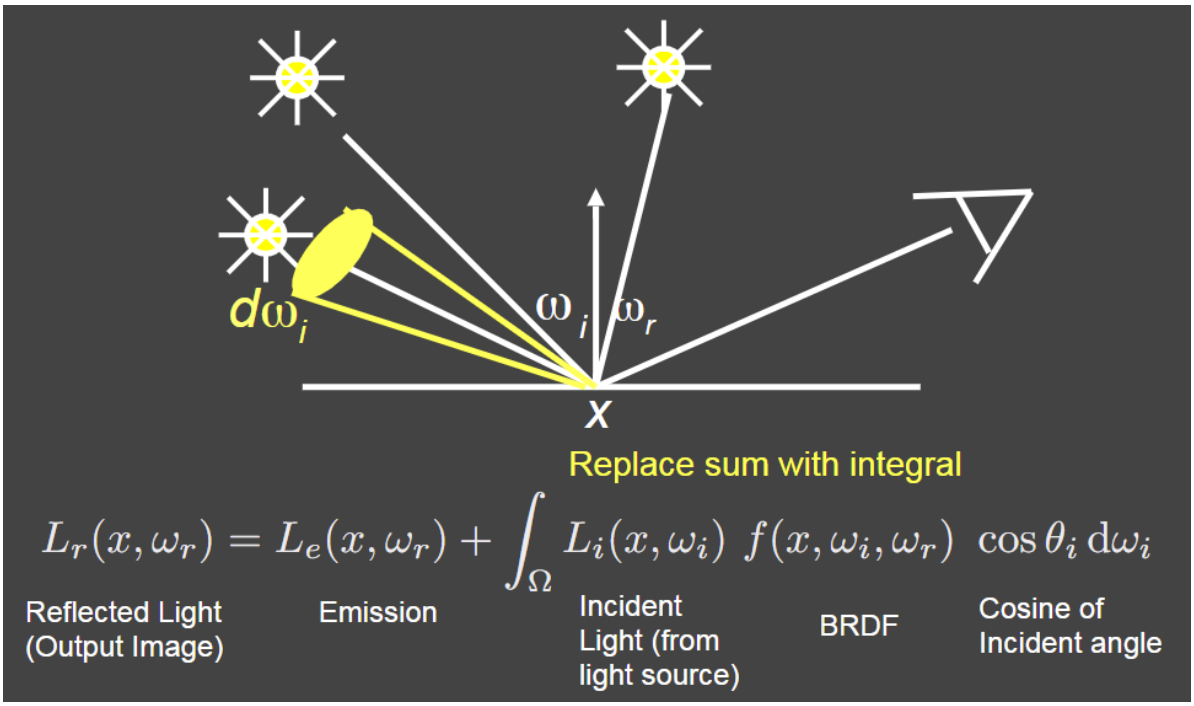
Global illumination

Understanding the rendering equation

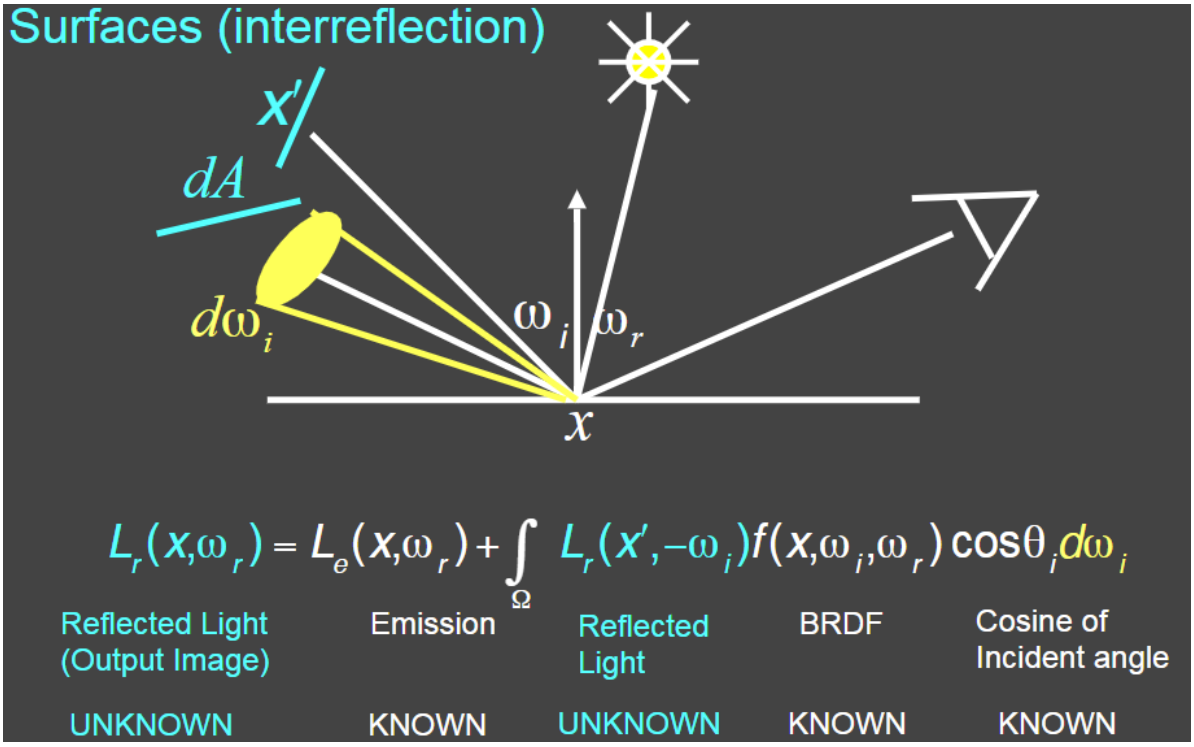
考虑所有光源



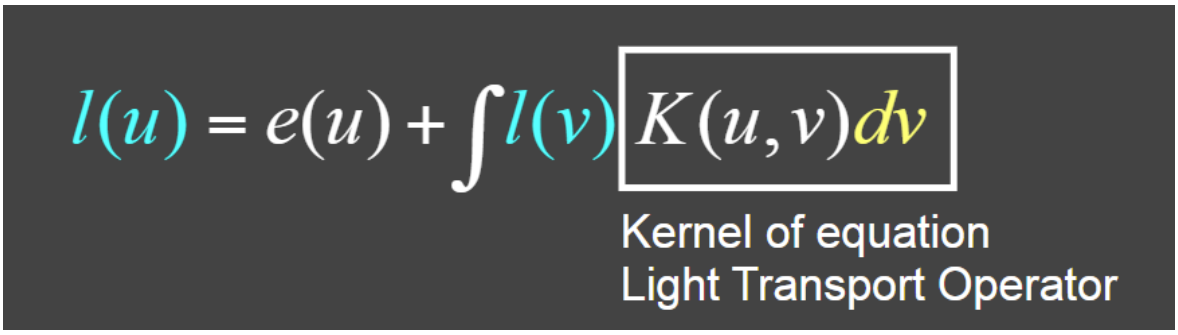
考虑面光源



考虑其它物体的反射光（当作光源）



化简方程表达方式



Is a Fredholm Integral Equation of second kind [extensively studied numerically] with canonical form

引入算子，~~省略大量过程~~

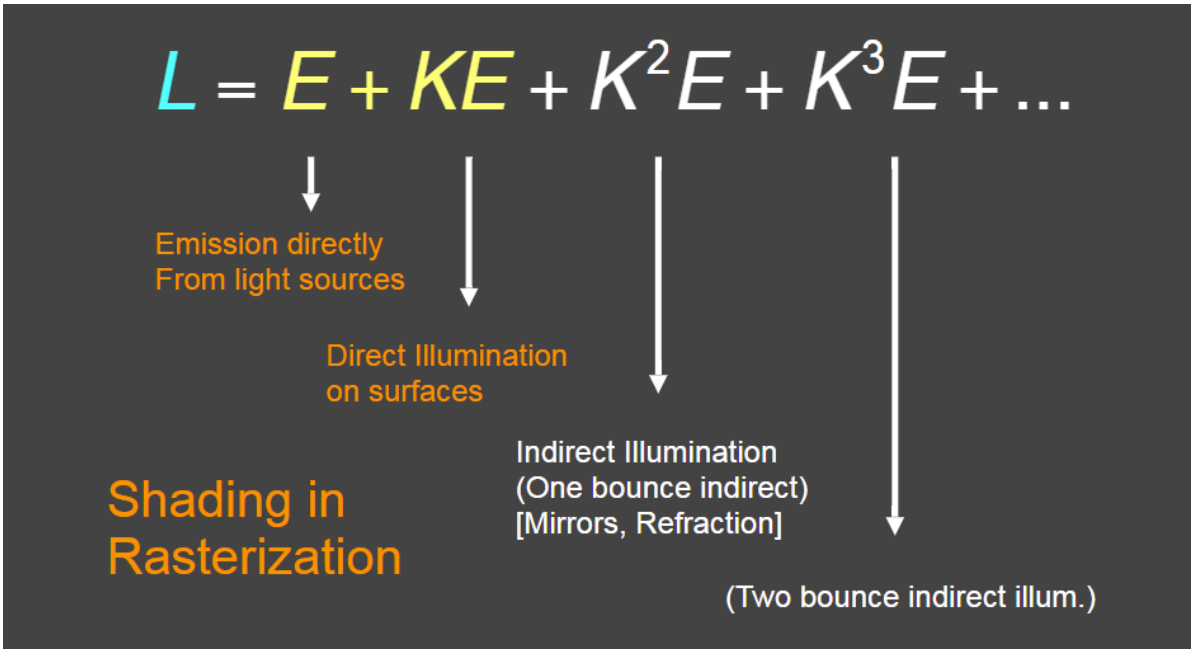
Can be discretized to a simple matrix equation[or system of simultaneous linear equations]

$$L = E + KL$$

L, E are vectors, K is the light transport matrix (反射算子)

L is set of all paths of light in scene

$$\begin{aligned} L &= (I-K)^{-1}E \\ &= (I+K+K^2+\cdots)E \\ L &= E + KE + K^2E + K^3E + \cdots \end{aligned}$$



Global illumination (全局光照) = 弹射次数超过1次的光线 = $KE + K^2E + K^3E + \cdots$

注：多次弹射后，全局光照一定会收敛

Probability review

Function of a Random Variable

$$X \sim p(x)$$
$$Y = f(X)$$

$$E[Y] = E[f(X)] = \int f(x)p(x)dx$$

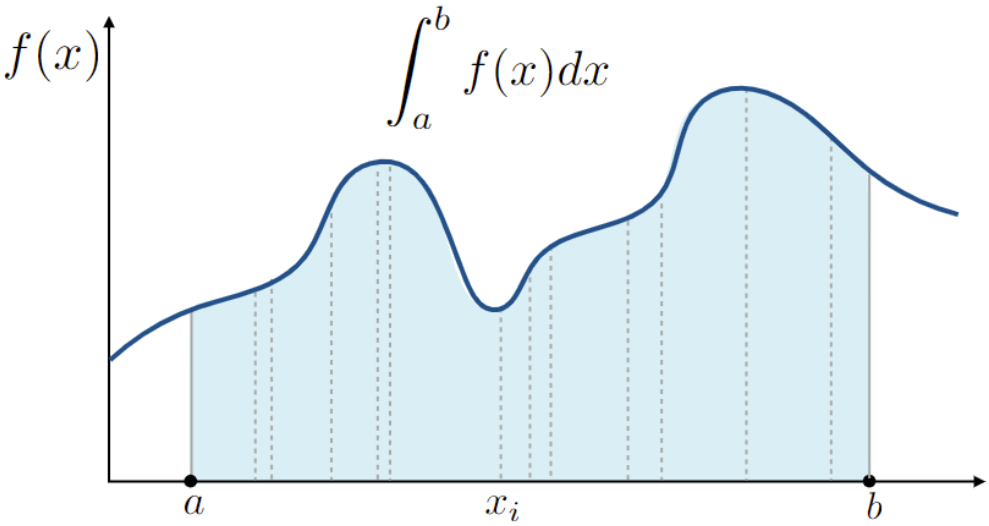
光线追踪（蒙特卡洛积分与路径追踪）

The Rendering Equation

$$L_o\left(p, \omega_o\right) = L_e\left(p, \omega_o\right) + \int_{\Omega^+} L_i\left(p, \omega_i\right) f_r\left(p, \omega_i, \omega_o\right)\left(n \cdot \omega_i\right) \mathrm{d}\omega_i$$

Monte Carlo Integration

Why: we want to solve an integral, but it can be too difficult to solve analytically.



What & How: estimate the integral of a function by averaging random samples of the function's value.

Monte Carlo estimator

$$X_i \sim p(x)$$
$$F_N = \frac{1}{N}\sum \frac{f(X_i)}{p(X_i)} \longrightarrow \int f(x)\mathrm{d}x$$

证明:

$$E[\frac{f(X)}{p(X)}] = \int \frac{f(x)}{p(x)} \cdot p(x)\mathrm{d}x = \int f(x)\mathrm{d}x$$
$$\frac{1}{N}\sum \frac{f(X_i)}{p(X_i)} \longrightarrow E[\frac{f(X)}{p(X)}]$$

Path Tracing

Whitted-style ray tracing

- Always perform specular reflections / refractions
- Stop bouncing at diffuse surfaces

Whitted-Style Ray Tracing: Problem 1

Where should the ray be reflected for glossy materials?

Mirror reflection

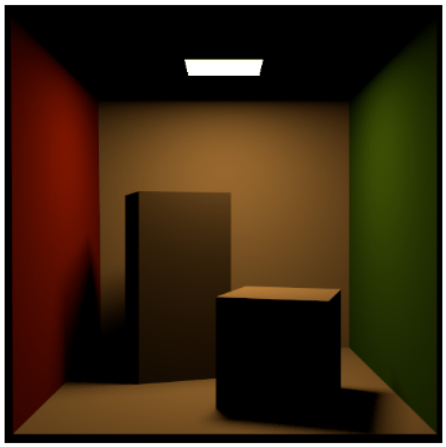
Glossy reflection

The Utah teapot

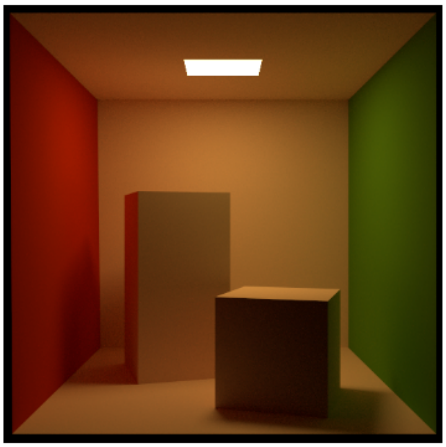
Glossy介于Mirror和Diffuse之间

Whitted-Style Ray Tracing: Problem 2

No reflections between diffuse materials?



Path traced:
direct illumination



Path traced:
global illumination

The Cornell box

color bleeding: 光照到带颜色的物体并反射后，把该物体的颜色带到其它物体上（例如，长方体左侧变红）

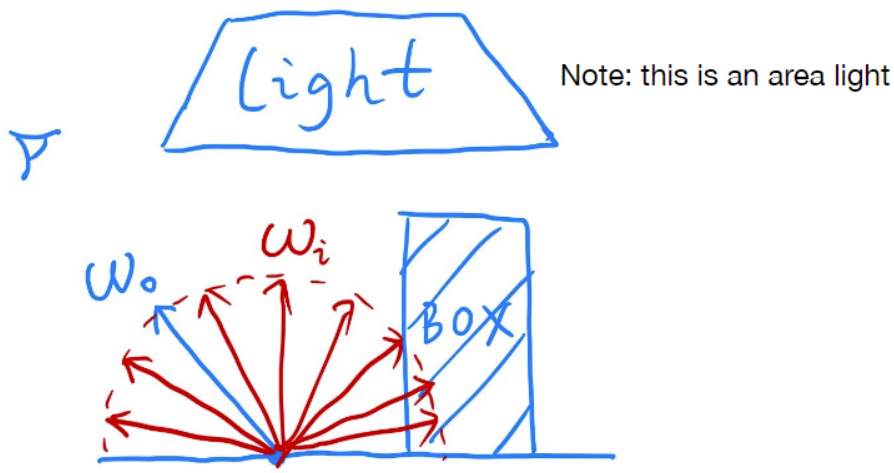
Whitted-Style Ray Tracing is Wrong. But the rendering equation is correct

- Solving an integral over the hemisphere
- Recursive execution

A Simple Monte Carlo Solution

简单场景：计算p点的直接光照

Suppose we want to render **one pixel (point)** in the following scene for **direct illumination** only



note: we assume all directions are pointing outwards

直接光照 = 只考虑来自光源的光线，其它方向$L = 0$

just an integration over directions

$$L_o(p, \omega_o) = \int_{\Omega^+} L_i(p, \omega_i) f_r(p, \omega_i, \omega_o)(n \cdot \omega_i) \mathrm{d}\omega_i$$

$$f(x) = L_i(p, \omega_i) f_r(p, \omega_i, \omega_o)(n \cdot \omega_i)$$

$$\mathrm{pdf}(\omega_i) = 1/2\pi$$

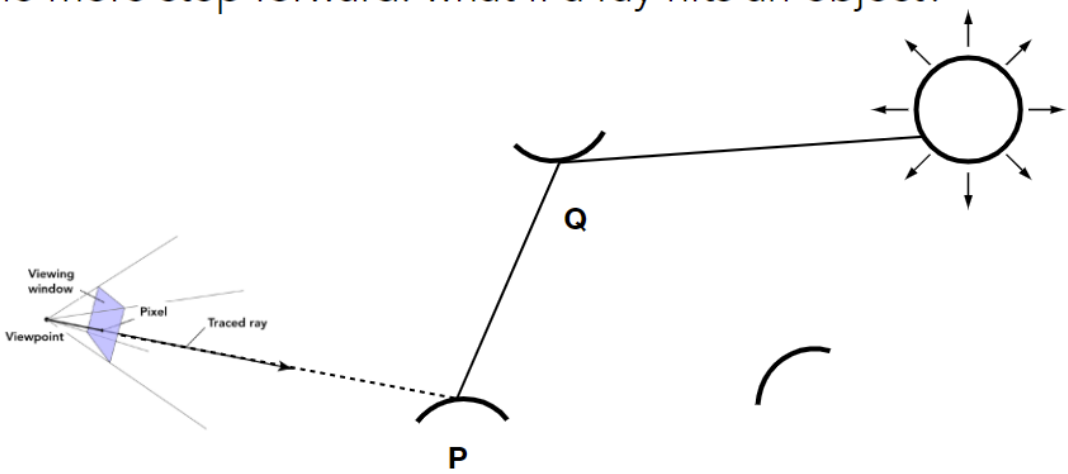
assume uniformly sampling the hemisphere

$$L_o(p, \omega_o) \approx \frac{1}{N} \sum_{i=1}^{N} \frac{L_i(p, \omega_i) f_r(p, \omega_i, \omega_o)(n \cdot \omega_i)}{\mathrm{pdf}(\omega_i)}$$

note: abuse notation a little bit for i

Introducing Global Illumination

One more step forward: what if a ray hits an object?



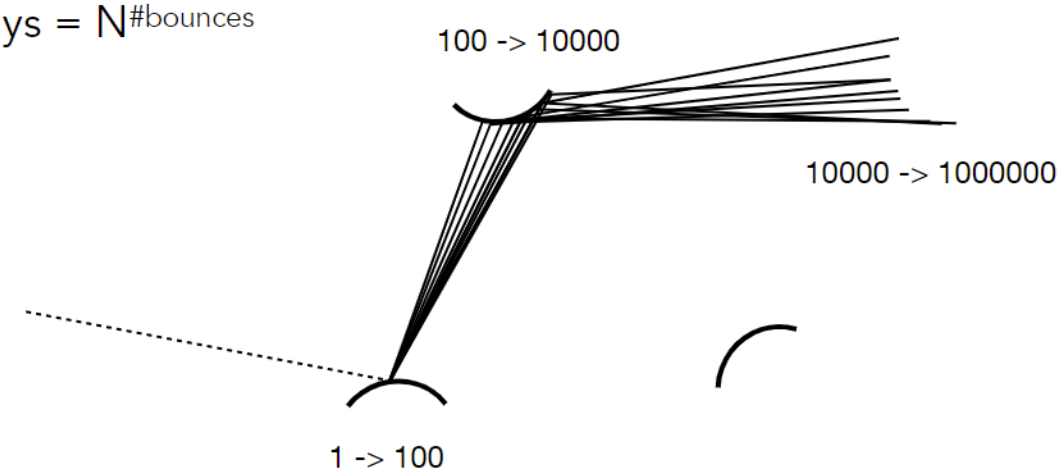
Q also reflects light to P! How much? The dir. illum. at Q!

```
// correct but useless
shade(p, wo)
    Randomly choose N directions wi~pdf
    Lo = 0.0
    For each wi
        Trace a ray r(p, wi)
        If ray r hit the light
            Lo += (1 / N) * L_i * f_r * cosine / pdf(wi)
        Else If ray r hit an object at q
            // add this, note that use -wi
            Lo += (1 / N) * shade(q, -wi) * f_r * cosine / pdf(wi)
    Return Lo
```

Problem 1: Explosion of #rays as #bounces go up

Path Tracing

$\#rays = N^{\#bounces}$



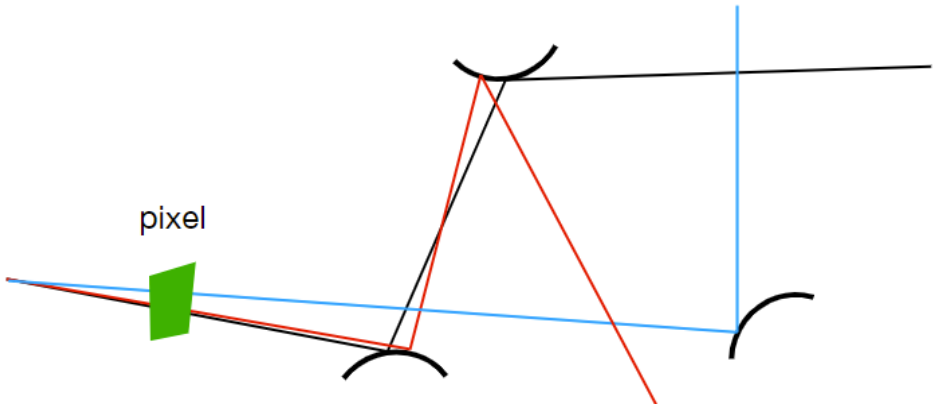
Key observation: #rays will not explode iff N = ???????

```
shade(p, wo)
    // only 1 ray is traced at each shading point
    Randomly choose ONE direction wi~pdf(w)
    Trace a ray r(p, wi)
    If ray r hit the light
        Return L_i * f_r * cosine / pdf(wi)
    Else If ray r hit an object at q
        Return shade(q, -wi) * f_r * cosine / pdf(wi)
```

This is path tracing! (FYI, Distributed Ray Tracing if N != 1)

But this will be noisy!

No problem, just trace more paths (not more rays) through each pixel and average their radiance!



形成连接视点和光源的path

Ray (Path) Generation

每个pixel发出一系列光线

```
ray_generation(camPos, pixel)
    Uniformly choose N sample positions within the pixel
    pixel_radiance = 0.0
    For each sample in the pixel
        Shoot a ray r(camPos, cam_to_sample)
        // path tracing
        If ray r hit the scene at p
            pixel_radiance += 1 / N * shade(p, sample_to_cam)
    Return pixel_radiance
```

Problem 2: The recursive algorithm will never stop!

Dilemma: the light does not stop bouncing indeed!

Cutting #bounces == cutting energy!

Russian Roulette (RR) (俄罗斯轮盘赌)

Russian Roulette is all about probability

- With probability 0 < P < 1, you are fine
- With probability 1 - P, otherwise

以一定概率停止递归

Previously, we always shoot a ray at a shading point and get the shading result **Lo**

Suppose we manually set a probability P (0 < P < 1)

- With probability P, shoot a ray and return the **shading result divided by P**: **Lo / P**
- With probability 1-P, don't shoot a ray and you'll get **0**

In this way, you can still **expect** to get Lo: E = P * (Lo / P) + (1 - P) * 0 = Lo

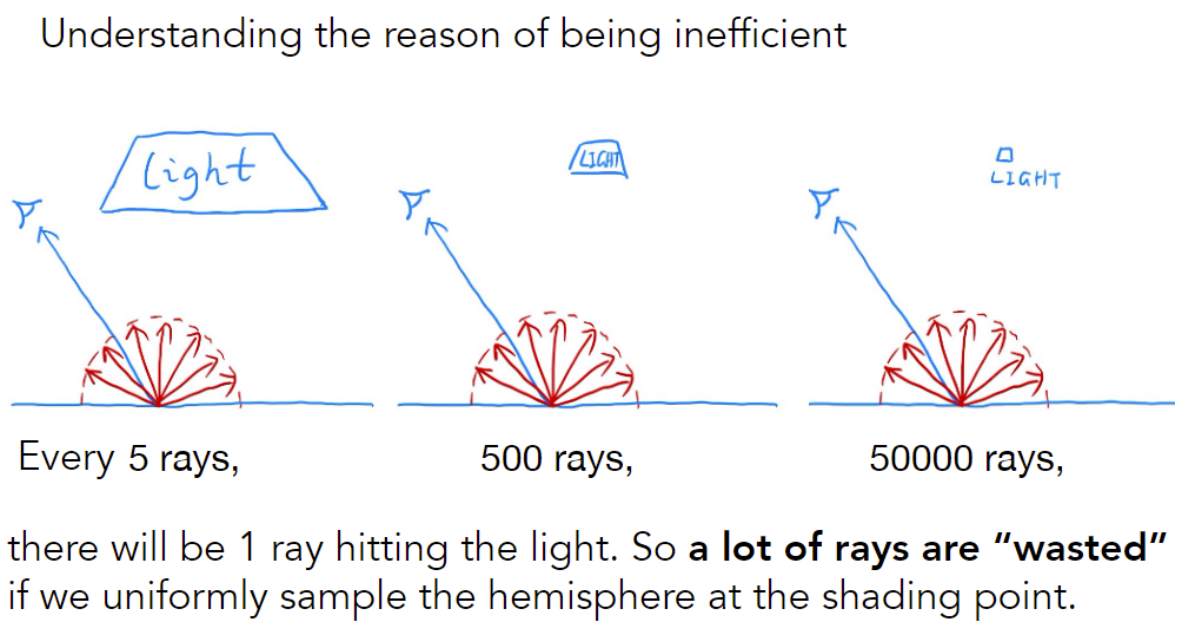
tradeoff: 引入噪声

```
shade(p, wo)
    // add RR
    Manually specify a probability P_RR
    Randomly select ksi in a uniform dist. in [0, 1]
    If (ksi > P_RR) return 0.0;

    Randomly choose ONE direction wi~pdf(w)
    Trace a ray r(p, wi)
    If ray r hit the light
        Return L_i * f_r * cosine / pdf(wi) / P_RR
    Else If ray r hit an object at q
        Return shade(q, -wi) * f_r * cosine / pdf(wi) / P_RR
```

Now we already have a correct version of path tracing!

But it's not really efficient. (noisy)

Sampling the Light (importance sampling)



Monte Carlo methods allows any sampling methods, so we can sample the light (therefore no rays are "wasted")

以下是公式推导，场景如下图所示

Assume uniformly sampling on the light: $\mathrm{pdf} = 1/A$ (**dA**)

But the rendering equation integrates on the solid angle: $L_o = \int L_i f_r \cos d\omega$ (**dw**)

两者积分域不同，不符合MC积分要求，需要变换积分变量

Since we sample on the light, can we integrate on the light?

Need the relationship between dω and dA

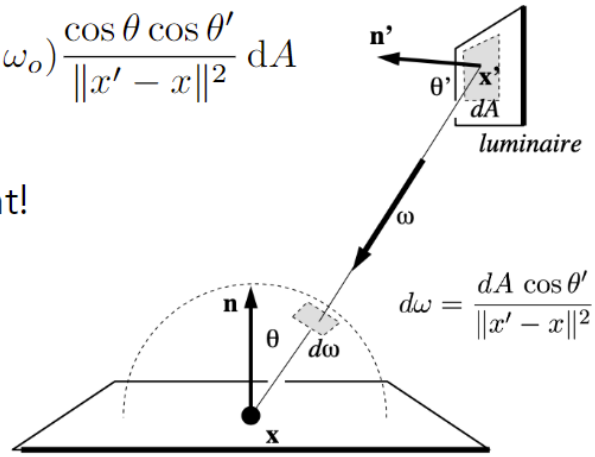
Recall the alternative def. of solid angle: Projected area on the unit sphere

$$d\omega = \frac{dA\cos\theta'}{\|x'-x\|^2}$$

注：注意这里是θ'

Then we can rewrite the rendering equation as

$$L_o(x,\omega_o) = \int_{\Omega+} L_i(x,\omega_i) f_r(x,\omega_i,\omega_o)\cos\theta \, \mathrm{d}\omega_i$$
$$= \int_A L_i(x,\omega_i) f_r(x,\omega_i,\omega_o) \frac{\cos\theta\cos\theta'}{\|x'-x\|^2} \, \mathrm{d}A$$



Now an integration on the light!

Monte Carlo integration:

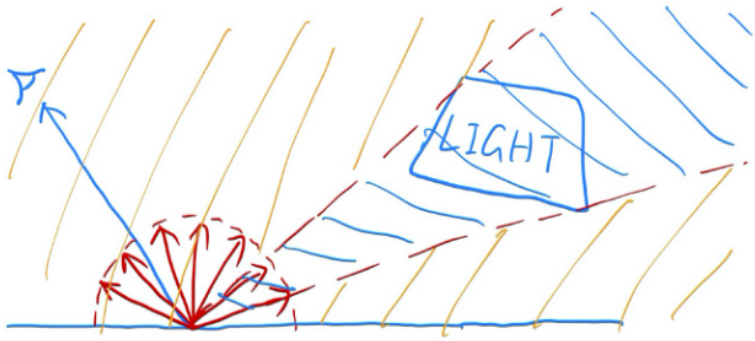
"f(x)": everything inside

Pdf: 1 / A

Previously, we assume the light is "accidentally" shot by uniform hemisphere sampling

Now we consider the radiance coming from two parts:

1. light source (direct, no need to have RR)
2. other reflectors (indirect, RR)



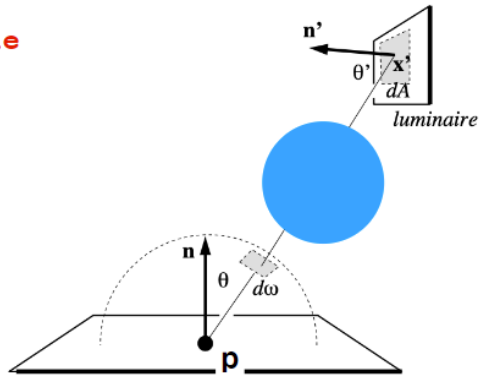

```
shade(p, wo)
    # Contribution from the light source.
    "Uniformly" sample the light at x' (pdf_light = 1 / A)
    L_dir = L_i * f_r * cos θ * cos θ' / |x' - p|^2 / pdf_light

    # Contribution from other reflectors.
    L_indir = 0.0
    Test Russian Roulette with probability P_RR
    "Uniformly" sample the hemisphere toward wi (pdf_hemi = 1 / 2pi)
    Trace a ray r(p, wi)
```

```
    # NOTE: only consider non-emitting object
    #  otherwise may sample light source again
    If ray r hit a "non-emitting" object at q
        L_indir = shade(q, -wi) * f_r * cos θ / pdf_hemi / P_RR

    Return L_dir + L_indir
# L_dir和L_indir分别计算直接和间接光照，两部分互不干扰
# 全局光照 = 直接光照 + 间接光照
# Q: L_dir和L_indir的收敛速度应该不一致?
```

One final thing: how do we know if the sample on the light is not blocked or not?

```
# Contribution from the light source.
L_dir = 0.0
Uniformly sample the light at x' (pdf_light = 1 / A)
Shoot a ray from p to x'
If the ray is not blocked in the middle
    L_dir = …
```

Now path tracing is finally done!



注：很难处理点光源（近似成小面积光源）

Some Side Notes

Path tracing (PT) is indeed difficult

- Consider it the most challenging in undergrad CS
- Why: physics, probability, calculus, coding
- Learning PT will help you understand deeper in these

PHOTO-REALISTIC (照片级真实感)

Ray tracing: Previous vs. Modern Concepts

Previous

- Ray tracing == Whitted-style ray tracing

Modern

- The general solution of light transport, including
- (Unidirectional & bidirectional) path tracing
- Photon mapping
- Metropolis light transport
- VCM / UPBP...

如何生成一张图：光栅化/光线追踪

Further Topic

Things we haven't covered / won't cover

- Uniformly sampling the hemisphere

  How? And in general, how to sample any function? (sampling)

  MCMC makes it!
- Monte Carlo integration allows arbitrary pdfs

  What's the best choice? (importance sampling)
- Do random numbers matter?

  Yes! (low discrepancy sequences)
- I can sample the hemisphere and the light

  Can I combine them? Yes! (multiple importance sampling)
- The radiance of a pixel is the average of radiance on all paths passing through it

  Why? (pixel reconstruction filter)
- Is the radiance of a pixel the color of a pixel?

  No. (gamma correction, curves, color space)
- Asking again, is path tracing still "Introductory"?

  > This time, yes. Fear the science, my friends. Prof. Yan.

材质与外观

外观是光线和材质共同作用的结果

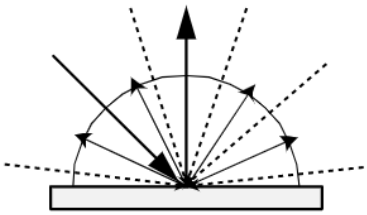
What is Material in Computer Graphics?

3D coffee mug model Rendered Rendered

Material == BRDF

Diffuse / Lambertian Material

Light is equally reflected in each output direction



$$f_r = c$$

Suppose the incident lighting is uniform:

$$\begin{aligned} L_o(\omega_o) &= \int_{H^2} f_r L_i(\omega_i) \cos\theta_i \, \mathrm{d}\omega_i \\ &= f_r L_i \int_{H^2} \cancel{(\omega_i)} \cos\theta_i \, \mathrm{d}\omega_i \\ &= \pi f_r L_i \end{aligned}$$

$$f_r = \frac{\rho}{\pi} \quad \text{— albedo (color)}$$

- diffuse material的BRDF是常数，均匀出射
- 假设入射光的radiance一样 (uniform)

  利用能量守恒：如果点不发光也不吸收，入射irradiance = 出射irradiance，那么 (最大值)

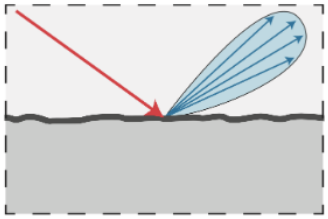
$$f_r = \frac{1}{\pi}$$

- albedo为反射率

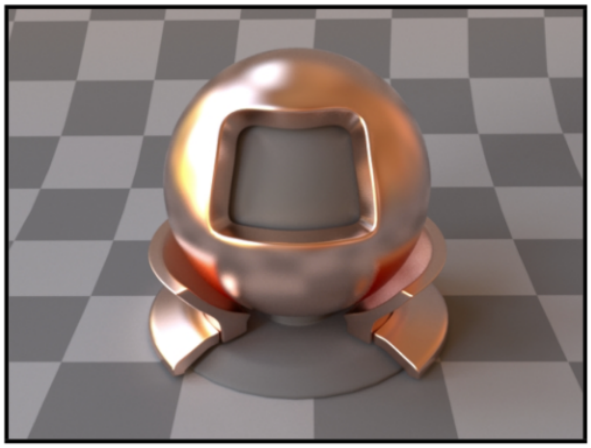
Glossy material

抛光金属，比镜子粗糙一点

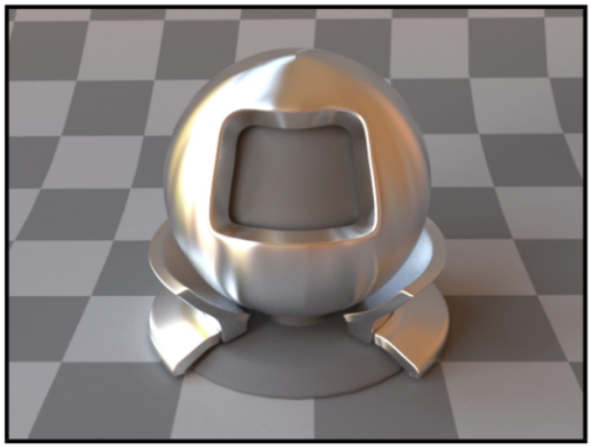
Glossy material (BRDF)



[Mitsuba renderer, Wenzel Jakob, 2010]

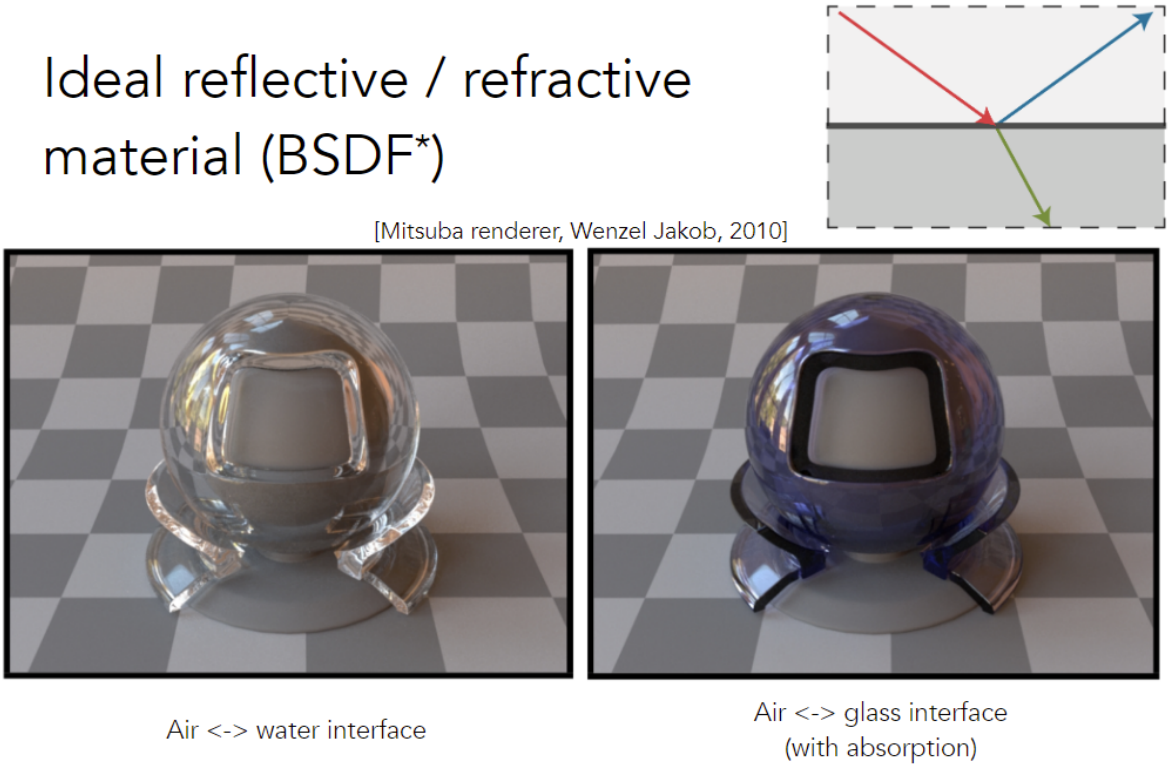


Copper



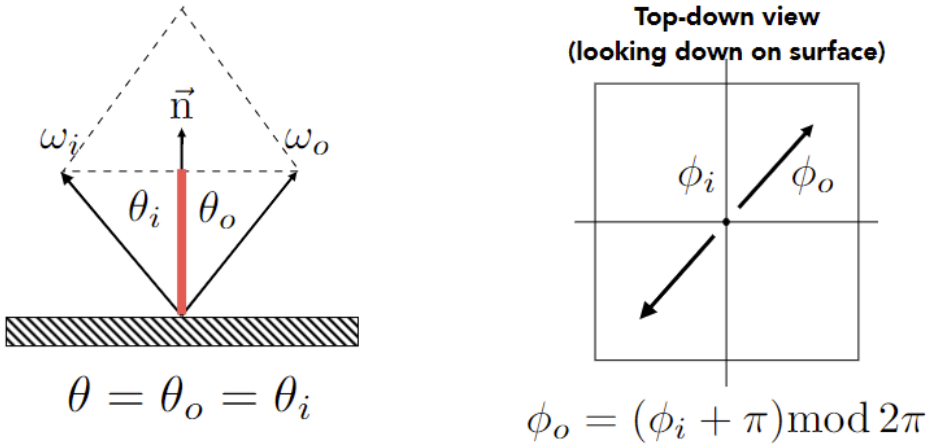
Aluminum

Ideal reflective / refractive material



glass with absorption: 折射光部分被吸收，显示出内部颜色

反射



$$\omega_o + \omega_i = 2\cos\theta\vec{n} = 2\left(\omega_i \cdot \vec{n}\right)\vec{n}$$
$$\omega_o = -\omega_i + 2\left(\omega_i \cdot \vec{n}\right)\vec{n}$$

Top-down view: 从法线上方向下看，入射光和反射光方位角相反

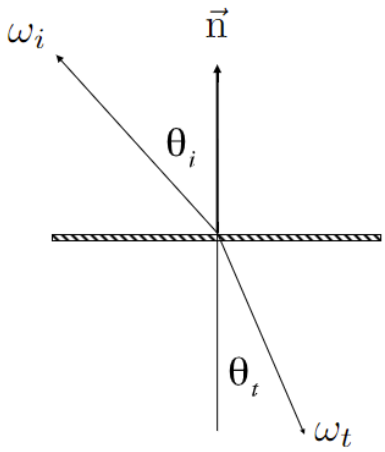
注：理想反射的BRDF很复杂（出射方向过于集中）

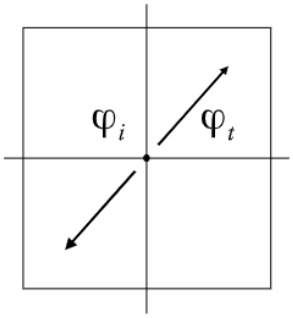
折射

仍采用几何光学描述折射

Caustics: 折射后聚集到同一点

Snell's Law





$$\varphi_t = \varphi_i \pm \pi$$

Medium	η*
Vacuum	1.0
Air (sea level)	1.00029
Water (20°C)	1.333
Glass	1.5-1.6
Diamond	2.42

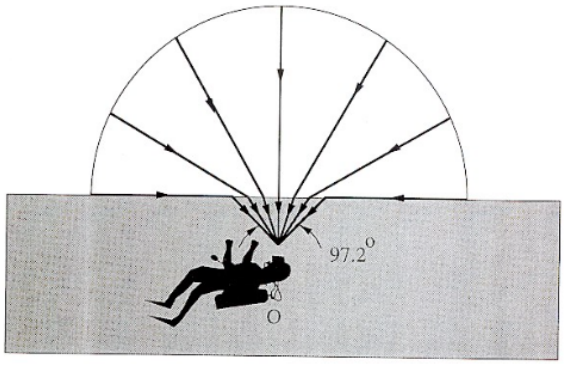
* index of refraction is wavelength dependent (these are averages)

$$\eta_i \sin\theta_i = \eta_t \sin\theta_t$$

全反射：$\eta_i > \eta_t$时可能发生（光密射到光疏）

Snell's Window / Circle

Total internal reflection



[Livingston and Lynch]

注：BT(transmit)DF + BRDF = BS(scatter)DF

Fresnel Reflection / Term（菲涅耳项）

描述反射和折射能量分配情况

例如，垂直看车窗可以看到窗外，斜着看车窗大概率看到车内反射

Reflectance depends on incident angle (and polarization of light)

This example: reflectance increases with grazing angle

导体和绝缘体菲涅耳项不同

Accurate: need to consider polarization

$$R_s = \left|\frac{n_1\cos\theta_i - n_2\cos\theta_t}{n_1\cos\theta_i + n_2\cos\theta_t}\right|^2 = \left|\frac{n_1\cos\theta_i - n_2\sqrt{1-\left(\frac{n_1}{n_2}\sin\theta_i\right)^2}}{n_1\cos\theta_i + n_2\sqrt{1-\left(\frac{n_1}{n_2}\sin\theta_i\right)^2}}\right|^2,$$

$$R_p = \left|\frac{n_1\cos\theta_t - n_2\cos\theta_i}{n_1\cos\theta_t + n_2\cos\theta_i}\right|^2 = \left|\frac{n_1\sqrt{1-\left(\frac{n_1}{n_2}\sin\theta_i\right)^2} - n_2\cos\theta_i}{n_1\sqrt{1-\left(\frac{n_1}{n_2}\sin\theta_i\right)^2} + n_2\cos\theta_i}\right|^2.$$

$$R_{\text{eff}} = \frac{1}{2}\left(R_s + R_p\right).$$

Approximate: Schlick's approximation

$$R(\theta) = R_0 + (1 - R_0)(1 - \cos\theta)^5$$

$$R_0 = \left(\frac{n_1 - n_2}{n_1 + n_2}\right)^2$$

注：$R(0) = R_0, R(\pi/2) = 1$

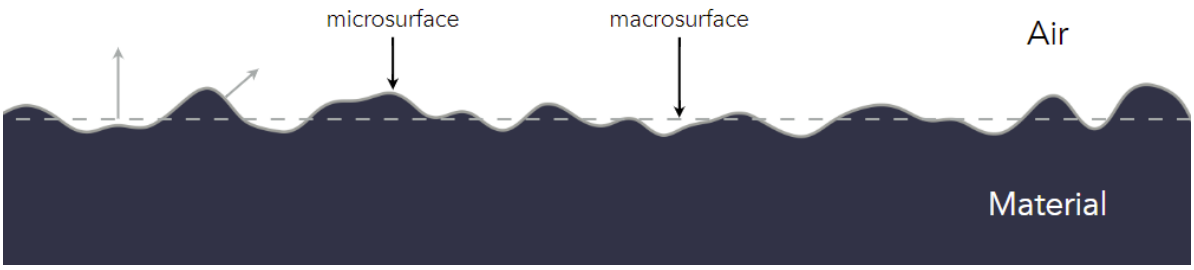
Microfacet (微表面) Material

Q：地球表面是粗糙的，如何产生图中的高光?

离得足够远时，不关心表面具体情况，只能看到总体效应

Microfacet Theory

远处看到材质，近处看到几何



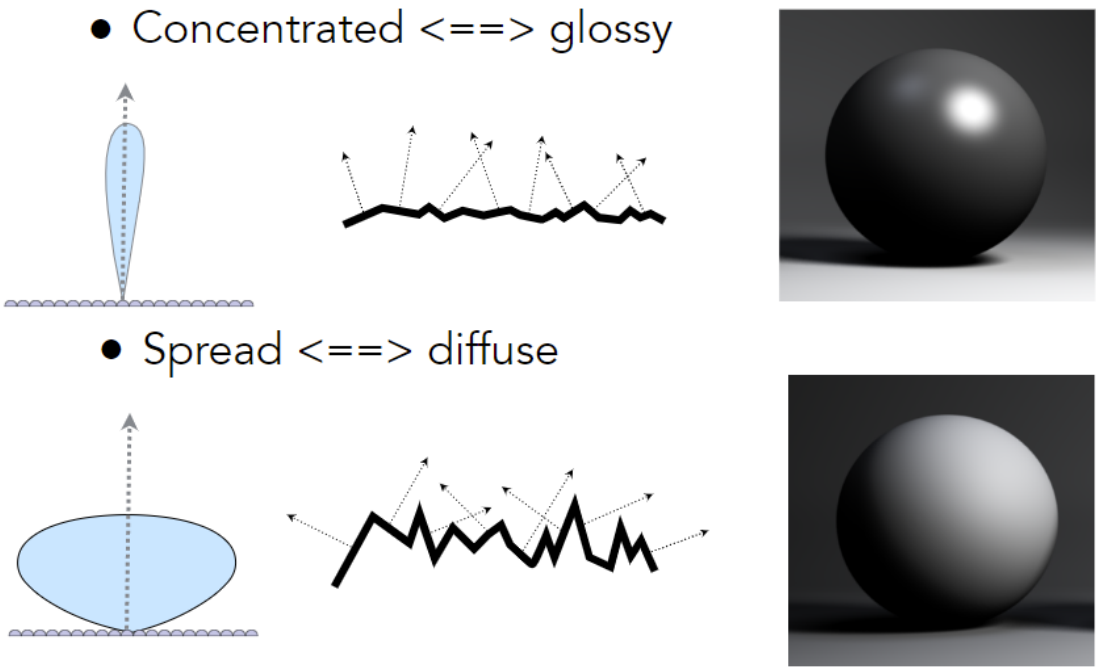
Rough surface

- Macroscale (远处): flat & rough
- Microscale (近处): bumpy & specular

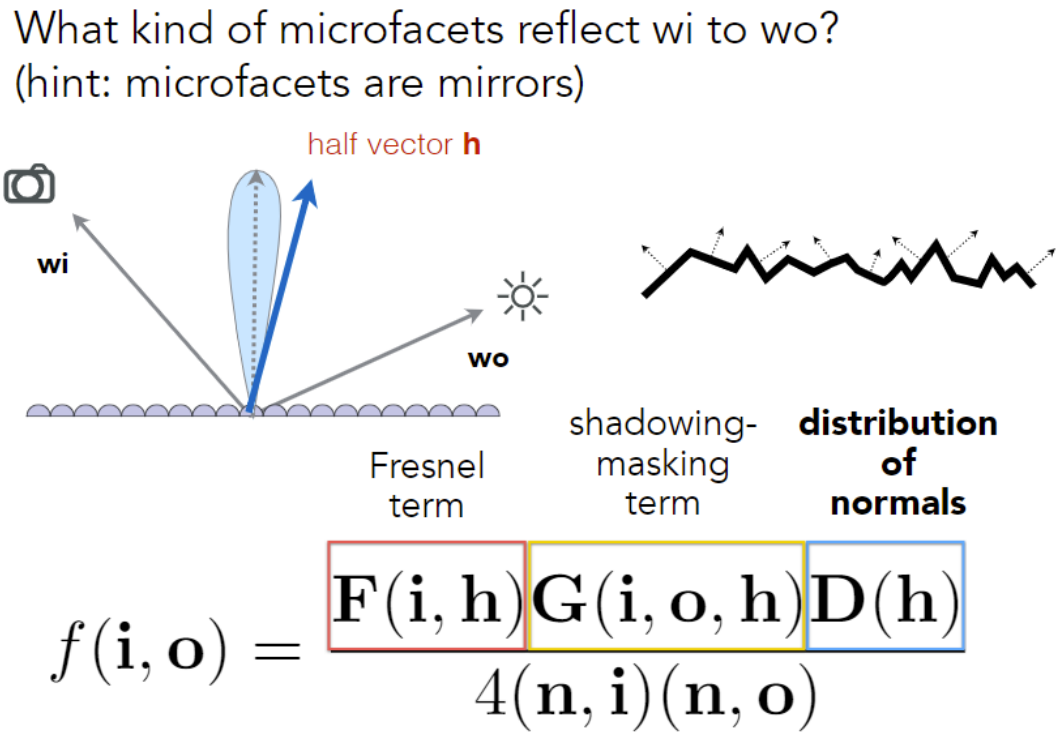
Individual elements of surface act like mirrors

- Known as Microfacets
- Each microfacet has its own normal

Key: the **distribution** of microfacets' normals



Microfacet BRDF

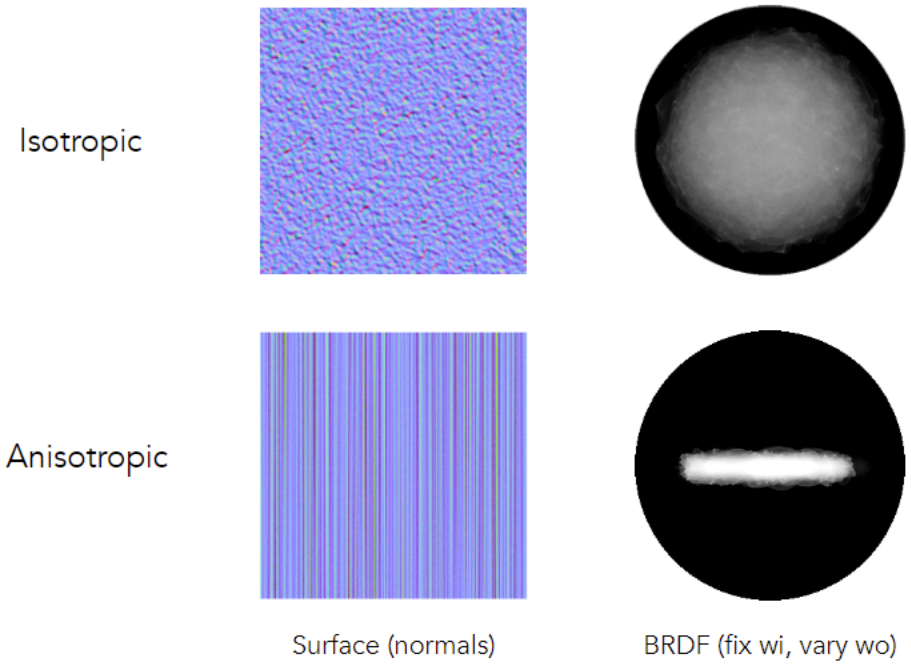


- $D(\vec{h})$：只有微表面$\vec{n}=\vec{h}$时，才能把入射光反射到出射光
- G：微表面互相遮挡

  光线平行表面时容易发生遮挡

Isotropic / Anisotropic Materials (BRDFs)

Key: directionality of underlying surface



Anisotropic (各向异性) BRDFs

Reflection depends on azimuthal angle ϕ (方位角)

$$f_r\left(\theta_i, \phi_i; \theta_r, \phi_r\right) \neq f_r\left(\theta_i, \theta_r, \phi_r-\phi_i\right)$$

Results from oriented microstructure of surface, e.g., brushed metal, Nylon, Velvet

Properties of BRDFs

- Non-negativity

$$f_r\left(\omega_i \rightarrow \omega_r\right) \geq 0$$

- Linearity (可加性)

$$L_r\left(\mathrm{p}, \omega_r\right)=\int_{H^2} f_r\left(\mathrm{p}, \omega_i \rightarrow \omega_r\right) L_i\left(\mathrm{p}, \omega_i\right) \cos \theta_i \mathrm{d} \omega_i$$

- Reciprocity principle (可逆性)

$$f_r\left(\omega_r \rightarrow \omega_i\right)=f_r\left(\omega_i \rightarrow \omega_r\right)$$

- Energy conservation (能量守恒)

$$\forall \omega_r \int_{H^2} f_r\left(\omega_i \rightarrow \omega_r\right) \cos \theta_i \mathrm{d} \omega_i \leq 1$$

- Isotropic vs. anisotropic
  - If isotropic,

$$f_r\left(\theta_i, \phi_i; \theta_r, \phi_r\right)=f_r\left(\theta_i, \theta_r, \phi_r-\phi_i\right)$$

    Then, from reciprocity,

$$f_r\left(\theta_i, \theta_r, \phi_r-\phi_i\right)=f_r\left(\theta_r, \theta_i, \phi_i-\phi_r\right)=f_r\left(\theta_i, \theta_r,\left|\phi_r-\phi_i\right|\right)$$

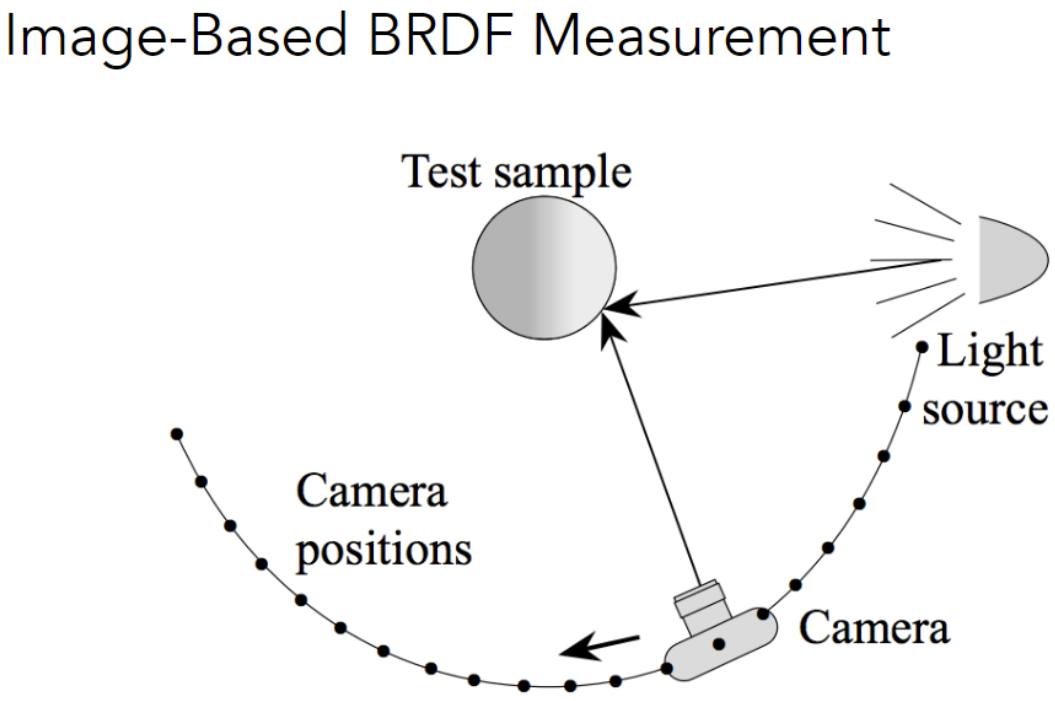
Measuring BRDFs

Avoid need to develop / derive models

- Automatically includes all of the scattering effects present

Can accurately render with real-world materials

- Useful for product design, special effects, ...



[Marschner et al. 1999]

Measuring BRDFs: gonioreflectometer

General approach:

```
foreach outgoing direction wo
    move light to illuminate surface with a thin beam from wo
    for each incoming direction wi
        move sensor to be at direction wi from surface
        measure incident radiance
```

Improving efficiency:

- Isotropic surfaces reduce dimensionality from 4D to 3D
- Reciprocity reduces # of measurements by half
- Clever optical systems... (e.g. 插值)

Challenges in Measuring BRDFs

- Accurate measurements at grazing angles

  Important due to Fresnel effects
- Measuring with dense enough sampling to capture high frequency specularities
- Retro-reflection
- Spatially-varying reflectance, ...

Representing Measured BRDFs (如何存储)

Desirable qualities

- Compact representation
- Accurate representation of measured data
- Efficient evaluation for arbitrary pairs of directions
- Good distributions available for importance sampling

BRDF库：MERL BRDF Database

高级光线传播与复杂外观建模

注：课件里包含大量示意图和不同方法结果的可视化比较

Advanced Light Transport

- Unbiased light transport methods
  - Bidirectional path tracing (BDPT)
  - Metropolis light transport (MLT)
- Biased light transport methods
  - Photon mapping
  - Vertex connection and merging (VCM)
- Instant radiosity (VPL / many light methods)

Biased vs. Unbiased Monte Carlo Estimators

- unbiased

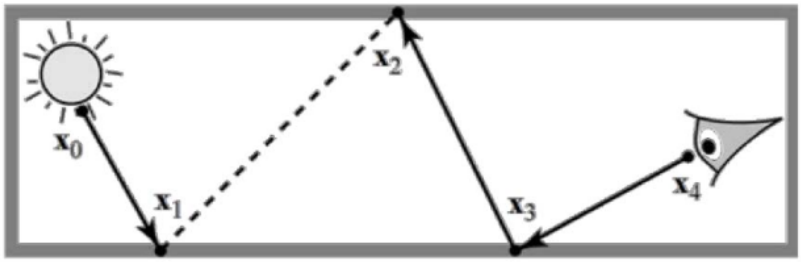
  The expected value of an unbiased estimator will always be the correct value

- biased

  One special case, the expected value converges to the correct value as infinite #samples are used — consistent

Bidirectional Path Tracing (BDPT)

- Recall: a path connects the camera and the light
- BDPT
  - Traces sub-paths from both the camera and the light
  - Connects the end points from both sub-paths



[Veach 1997]

- Suitable if the light transport is complex on the light's side (类比双向搜索)
- Difficult to implement & quite slow

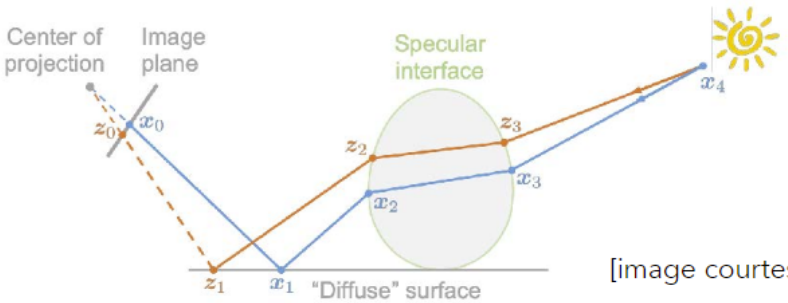
Metropolis Light Transport (MLT)

A Markov Chain Monte Carlo (MCMC) application

- Jumping from the current sample to the next with some PDF

Very good at locally (局部) exploring difficult light paths

Key idea: Locally perturb an existing path to get a new path



[image courtesy of Shuang Zhao]

- Pros
  - Works great with difficult light paths

    找到一条path作为初始sample，不断找到更多

    Caustics：specular + diffuse + specular 形成困难路径
  - Also unbiased
- Cons
  - Difficult to estimate the convergence rate
  - Does not guarantee equal convergence rate per pixel (since explore locally)
  - So, usually produces "dirty" results
  - Therefore, usually not used to render animations

Photon Mapping

- A biased approach & A two-stage method
- Very good at handling Specular-Diffuse-Specular (SDS) paths and generating caustics

Approach (variations apply)

1. photon tracing

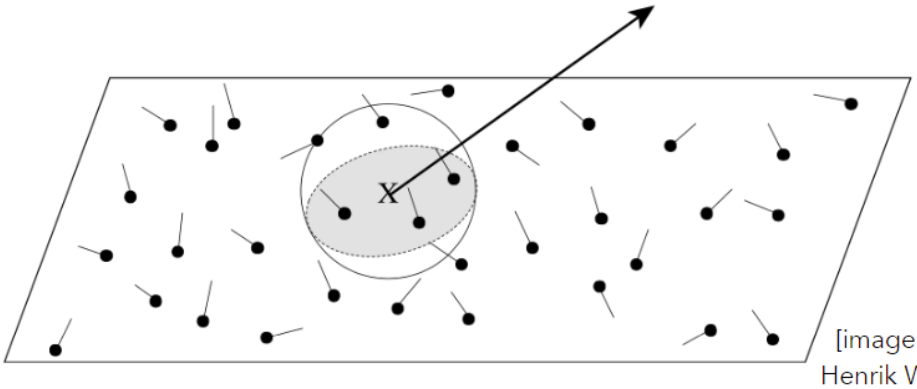
   Emitting photons from the light source, bouncing them around, then recording (stop) photons on diffuse surfaces

2. photon collection (final gathering)

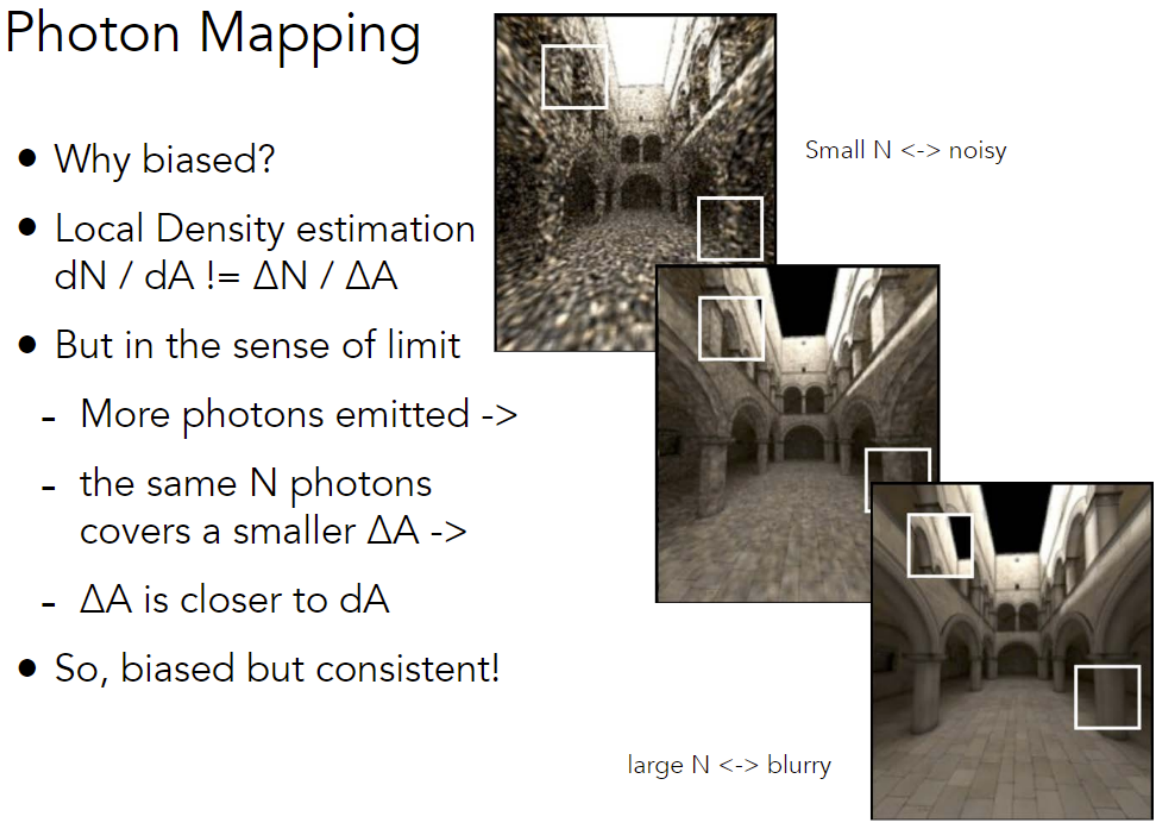
   Shoot sub-paths from the camera, bouncing them around, until they hit diffuse surfaces

Calculation — local density estimation

- Idea: areas with more photons should be brighter
- For each shading point, find the nearest N photons. Take the surface area they over



[image courtesy of Henrik Wann Jensen]



An easier understanding bias in rendering

- Biased == blurry
- Consistent == not blurry with infinite #samples

Why not do “const range” search for density estimation?

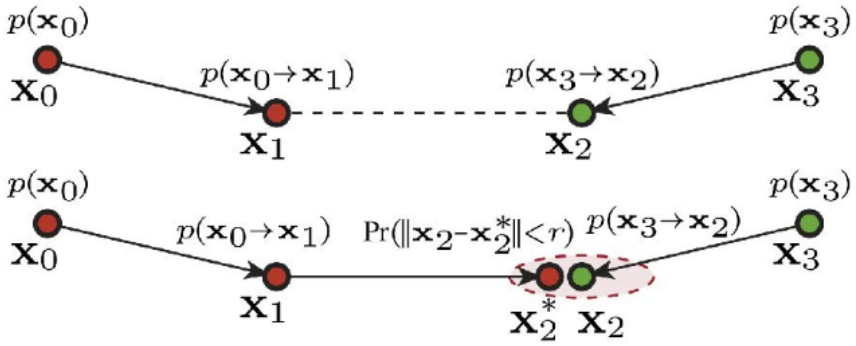
- 结果有偏，且不一致；光子越多越亮

Vertex Connection and Merging

A combination of BDPT and Photon Mapping

Key idea

- Let’s not waste the sub-paths in BDPT if their end points cannot be connected but can be merged
- Use photon mapping to handle the merging of nearby “photons”



Instant Radiosity (IR)

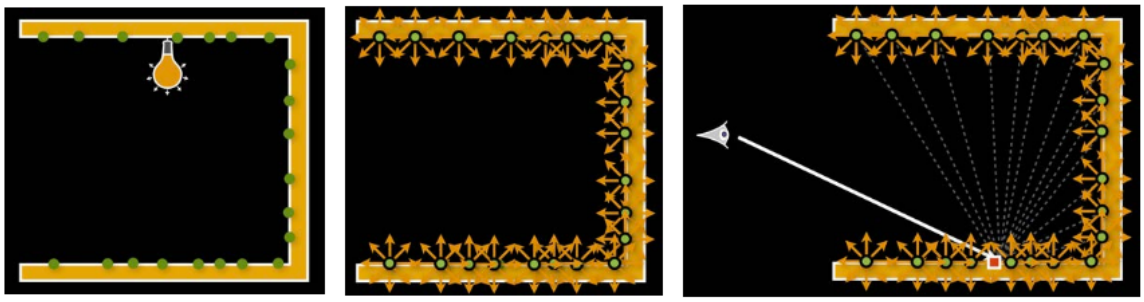
Sometimes also called many-light approaches

Key idea

- Lit surfaces can be treated as light sources

Approach

- Shoot light sub-paths and assume the end point of each sub-path is a Virtual Point Light (VPL)
- Render the scene as usual using these VPLs



Pros

- fast and usually gives good results on diffuse scenes

Cons

- Spikes will emerge when VPLs are close to shading points

  distance作为分母，出现精度问题
- Cannot handle glossy materials

Advanced Appearance Modeling

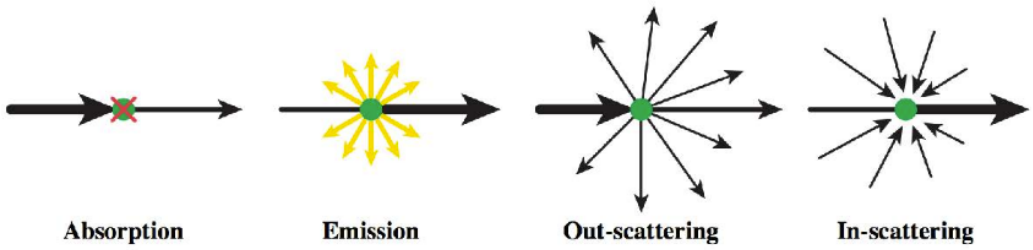
- Non-surface models
  - Participating media
  - Hair / fur / fiber (BCSDF)
  - Granular material
- Surface models
  - Translucent material (BSSRDF)
  - Cloth
  - Detailed material (non-statistical BRDF)
- Procedural appearance

Participating Media

e.g., Fog, Cloud

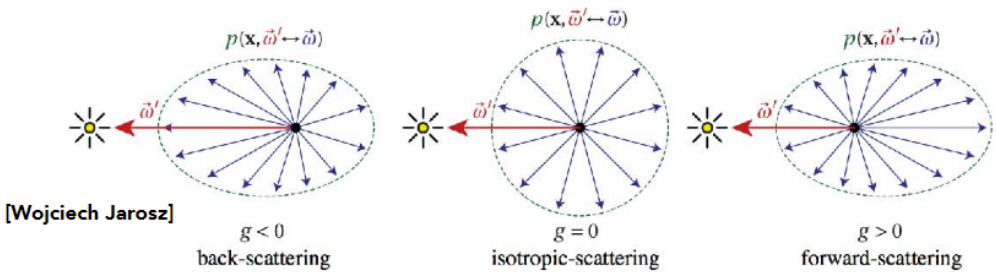
注：很多介质都可以不同程度地透光

- At any point as light travels through a participating medium, it can be (partially) absorbed and scattered.



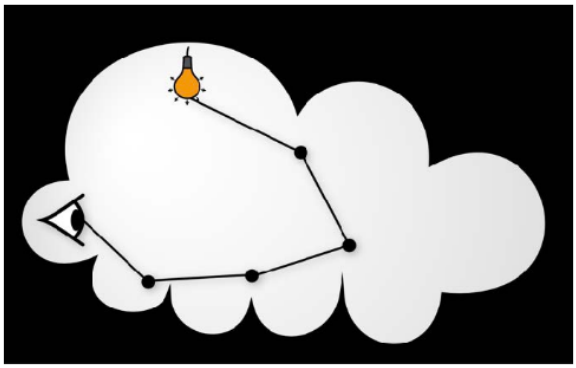
Phase Function

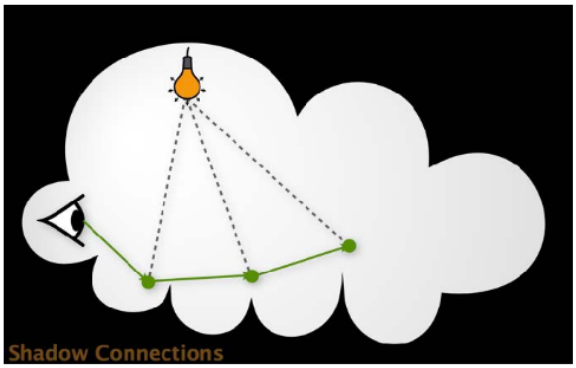
- Use Phase Function to describe the angular distribution of light scattering at any point x within participating media.



Rendering

- Randomly choose a direction to bounce
- Randomly choose a distance to go straight
- At each 'shading point', connect to the light



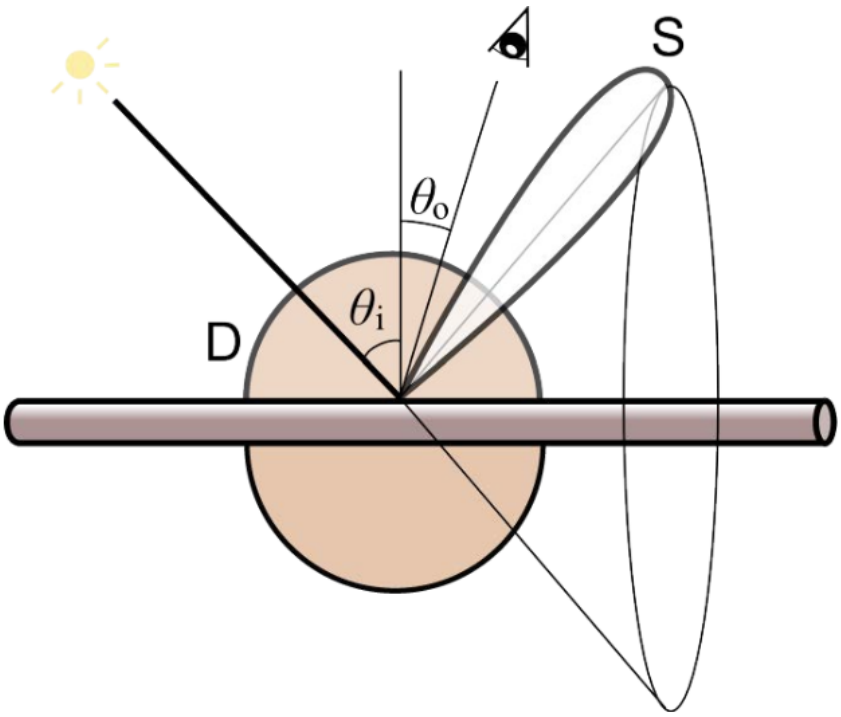


Hair Appearance

光线和曲线作用

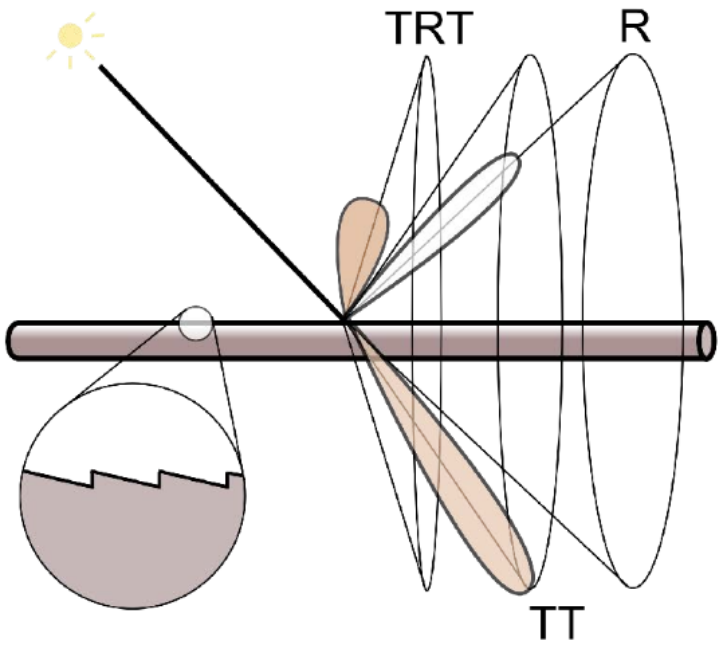
无色高光 + 有色高光 （分别为左右两边所示）

Kajiya-Kay Model



- 圆锥specular + diffuse

Marschner Model

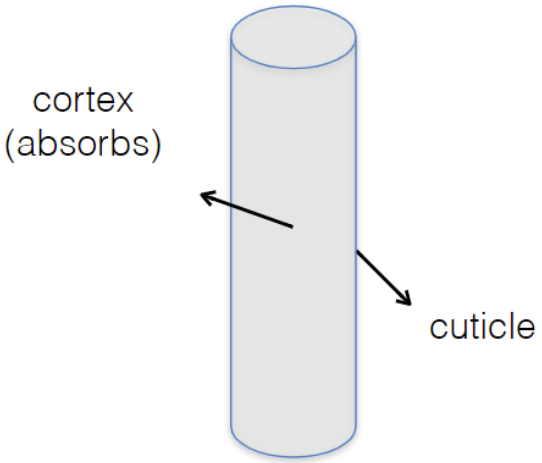


- 直接反射 (R) + 穿入头发 (T)

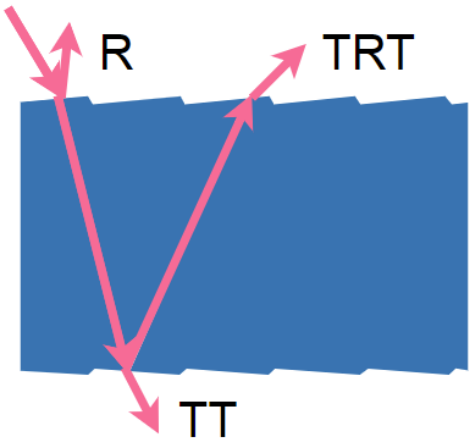
  TT: 穿透两次出去

  TRT: 穿透后在内壁反射后返回

- Glass-like cylinder



- 3 types of light interactions:
  R, TT, TRT
  (R: reflection, T: transmission)

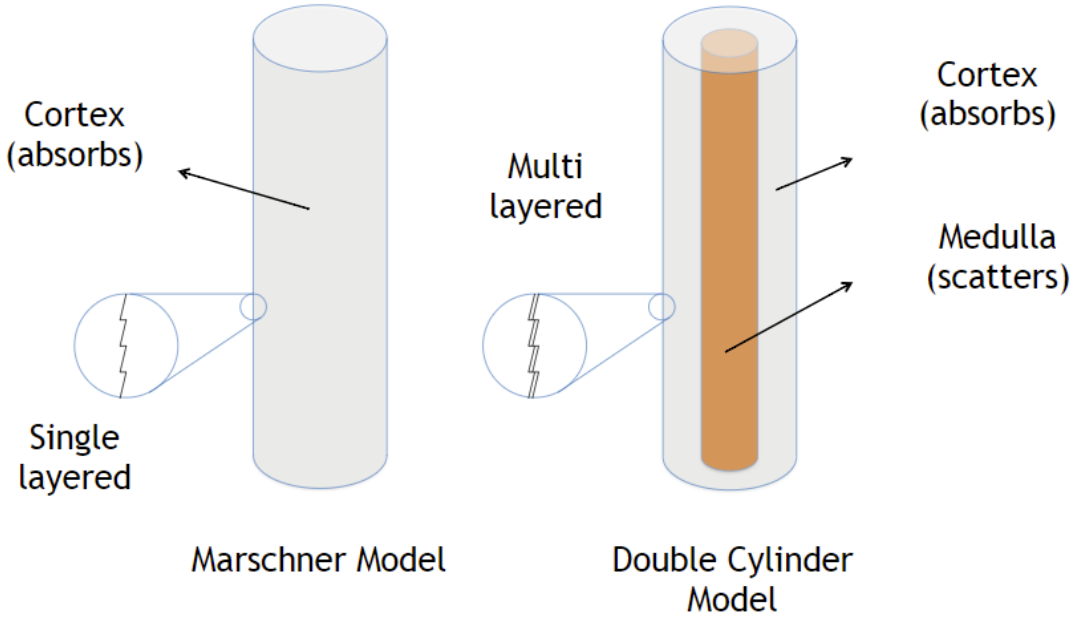


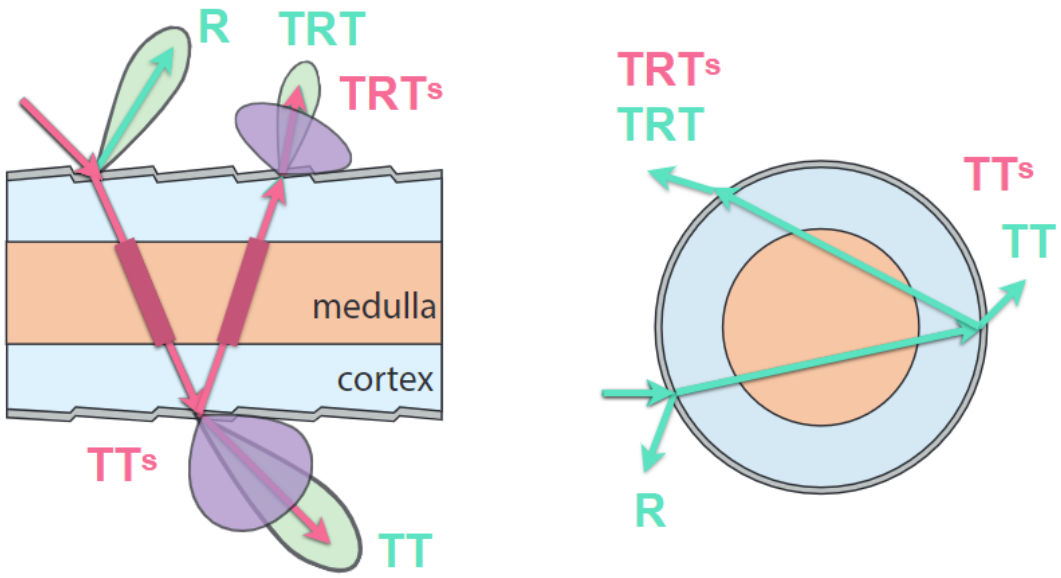
Fur Appearance

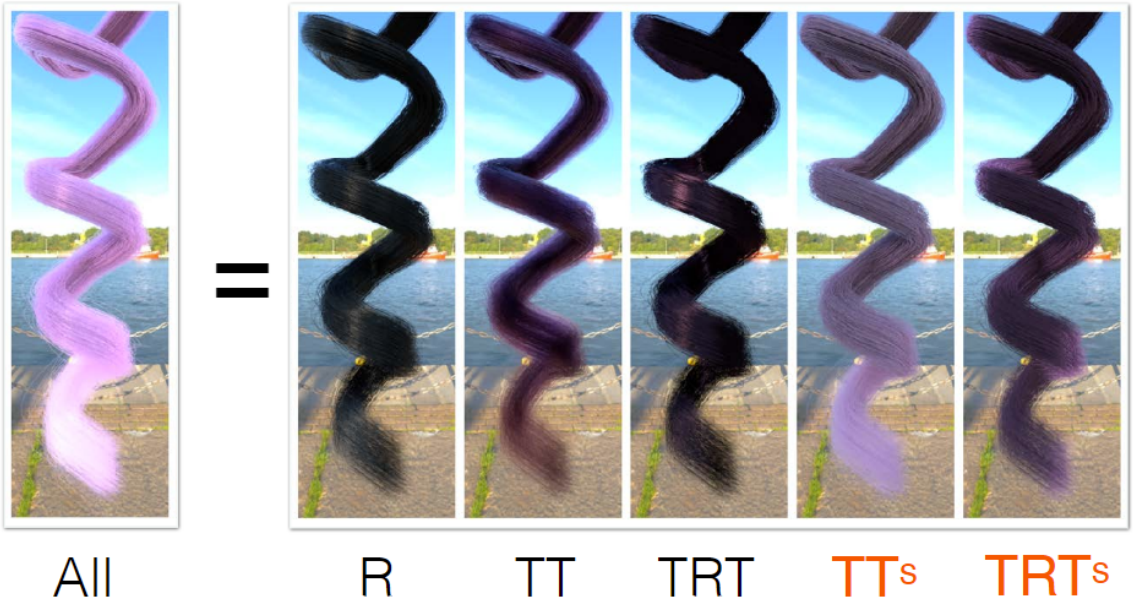
Hair/Fur contains

- Cortex: Absorbs light
- Medulla: Scatters light
- Cuticle: Covered with scales

但动物的Medulla比较大


Cortex
(absorbs)
Single
layered
Marschner Model
Multi
layered
Cortex
(absorbs)
Medulla
(scatters)
Double Cylinder
Model

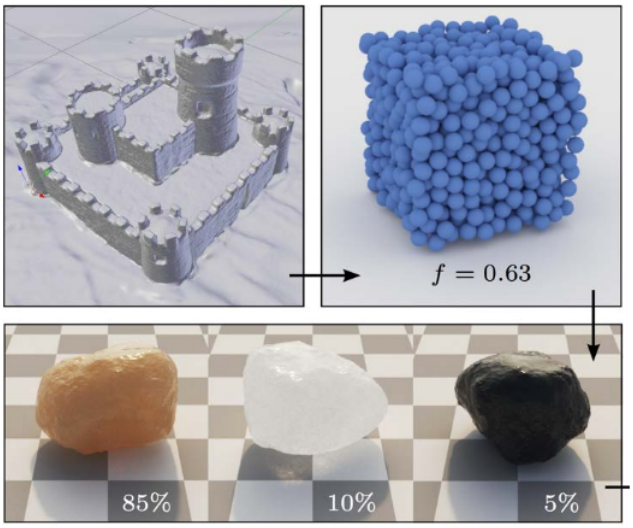

R
TRT
TRTs
medulla
cortex
TTs
TT
TRTs
TRT
TTs
TT
R

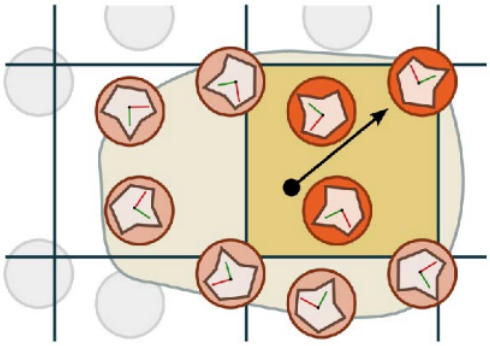

=
All
R
TT
TRT
TTs
TRTs

Granular Material

e.g., sand, rice

- Can we avoid explicit modeling of all granules?
  - Yes with procedural definition.





Translucent Material

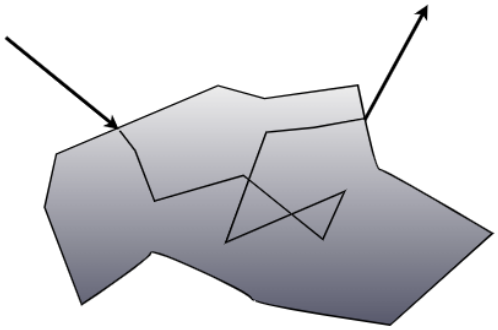
e.g. 玉石

光线可以从不同的地方离开

Subsurface Scattering

Visual characteristics of many surfaces caused by light exiting at different points than it enters

- Violates a fundamental assumption of the BRDF



[Jensen et al 2001]

[Donner et al 2008]

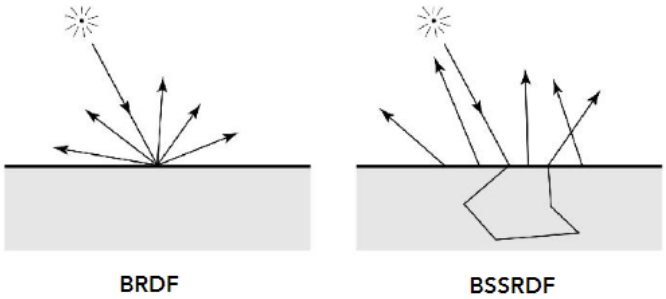
Scattering Functions

BSSRDF: generalization of BRDF; exitant radiance at one point due to incident differential irradiance at another point:

$$S\left(x_i, \omega_i, x_o, \omega_o\right)$$

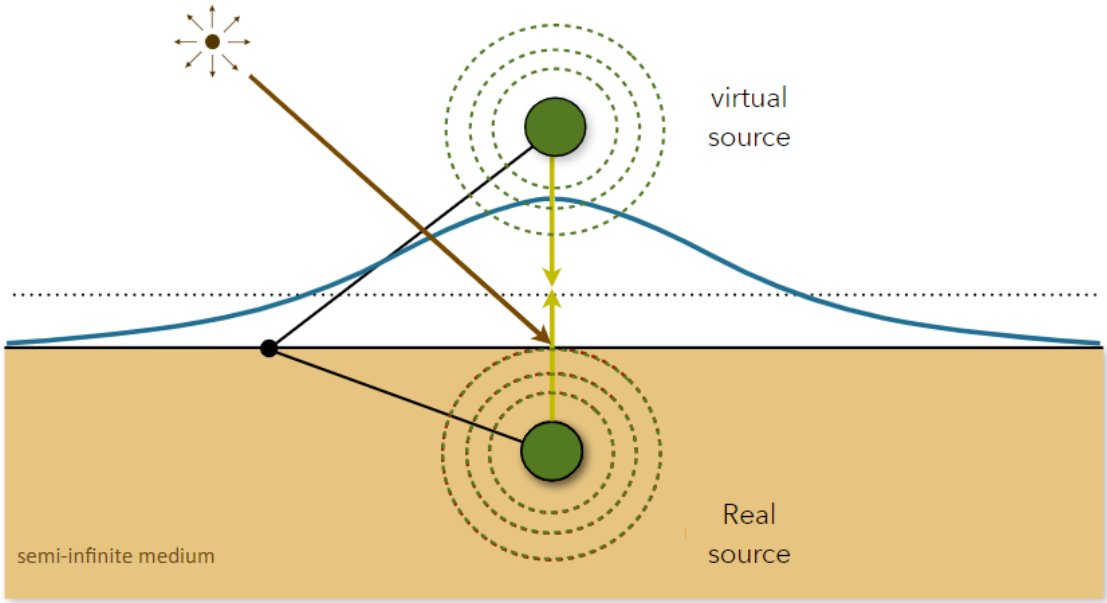
Generalization of rendering equation: integrating over all points on the surface and all directions (!)

$$L\left(x_o, \omega_o\right) = \int_A \int_{H^2} S\left(x_i, \omega_i, x_o, \omega_o\right) L_i\left(x_i, \omega_i\right) \cos\theta_i \, \mathrm{d}\omega_i \, \mathrm{d}A$$



Dipole Approximation

Approximate light diffusion by introducing two point sources



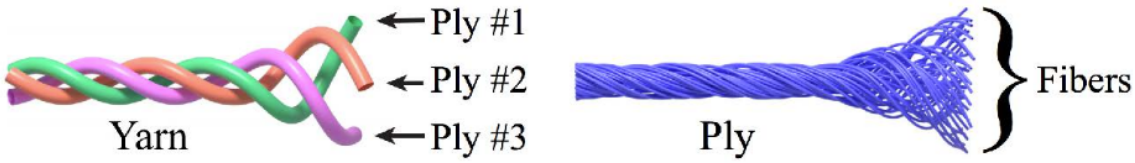
[image from Habel et al. 2013]

Application: 皮肤 link

Cloth

A collection of twisted fibers!

- Two levels of twist



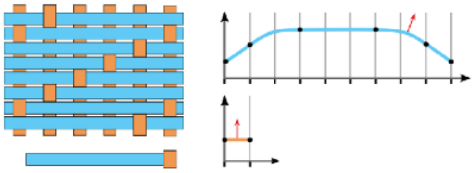
- Woven or knitted

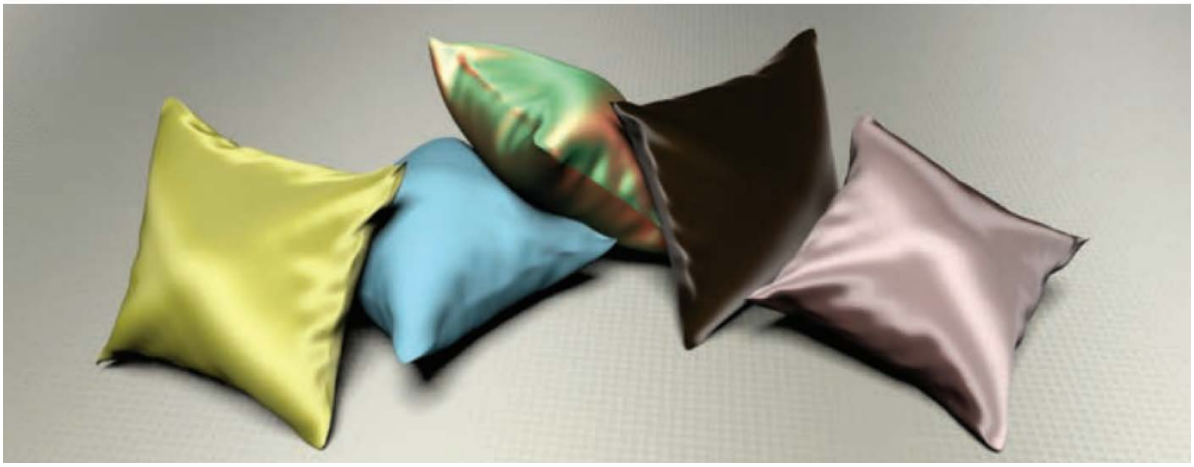
Render as Surface

表面模型非常复杂，考虑overall情况

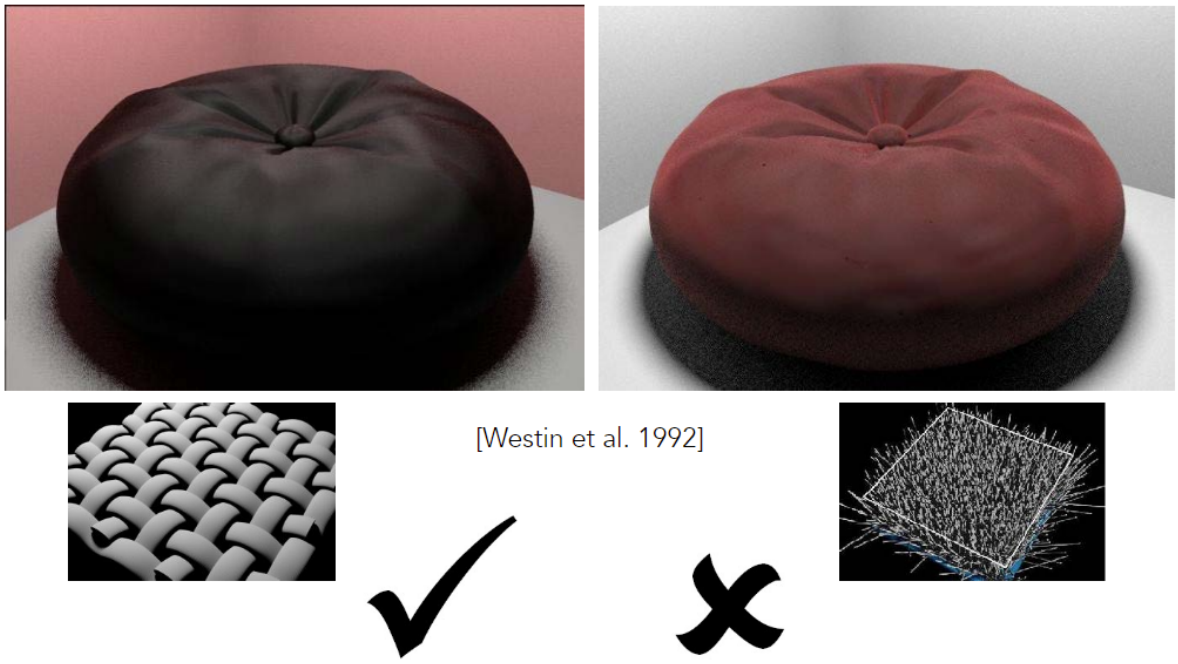
Cloth: Render as Surface

- Given the weaving pattern, calculate the overall behavior
- Render using a BRDF





Render as Surface — Limitation



Render as Participating Media

认为是空间分布的体积，划分成有不同参数的微小体积

Properties of individual fibers & their distribution -> scattering parameters

Render as Actual Fibers

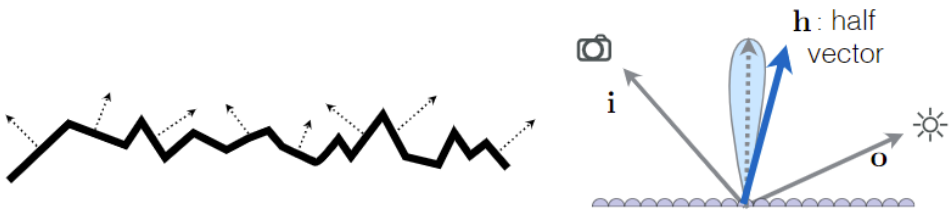
Render every fiber explicitly! (类似hair)

Detailed material (non-statistical BRDF)

render结果过于完美，没有划痕

Real world is more complicated

Recap: Microfacet BRDF



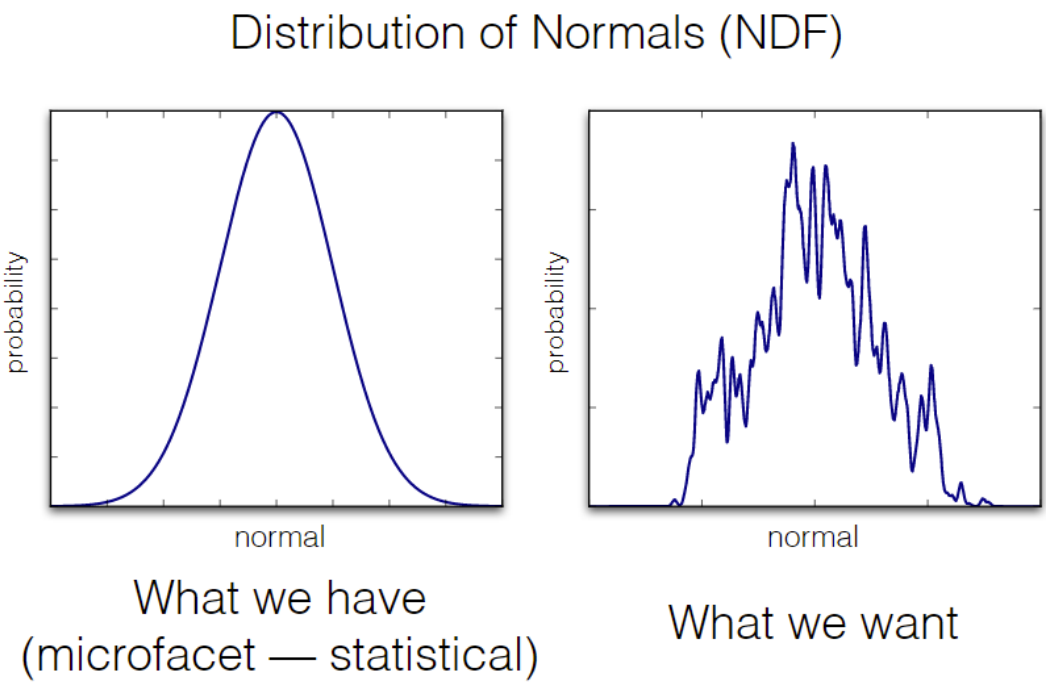
Surface = Specular microfacets + statistical normals

$$f(\mathbf{i},\mathbf{o}) = \frac{\mathbf{F}(\mathbf{i},\mathbf{h})\mathbf{G}(\mathbf{i},\mathbf{o},\mathbf{h})\boxed{\mathbf{D}(\mathbf{h})}}{4(\mathbf{n},\mathbf{i})(\mathbf{n},\mathbf{o})}$$

NDF: Normal Distribution Function

Statistical NDF vs. Actual NDF

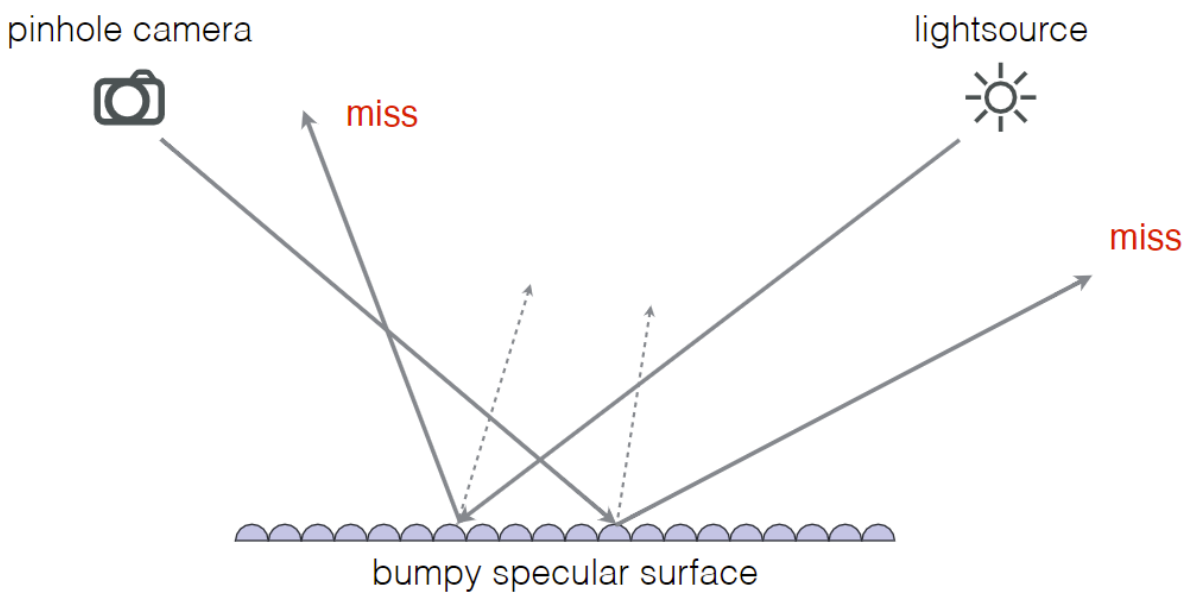
Statistical NDF 过于完美



使用法线贴图来定义Actual NDF

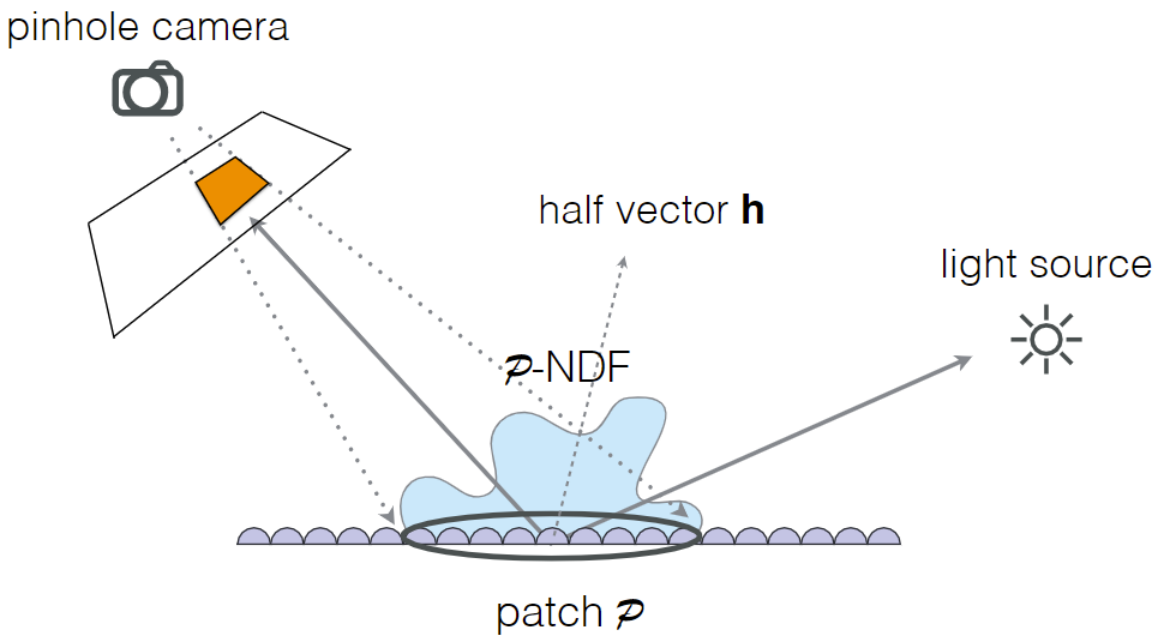
But For Rendering? Too difficult!

Difficult path sampling problem



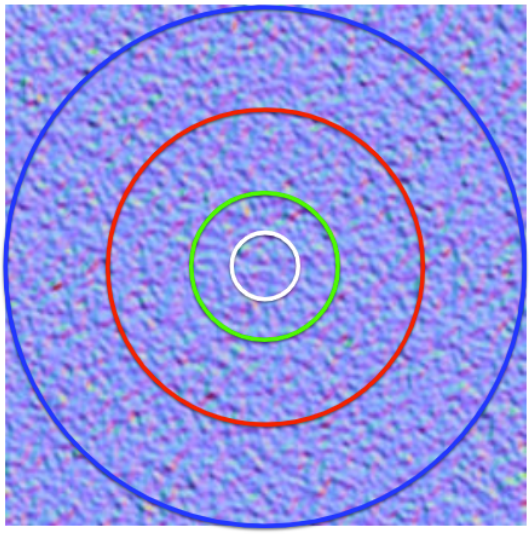
problem: 认为微表面是镜面

Solution: BRDF over a pixel

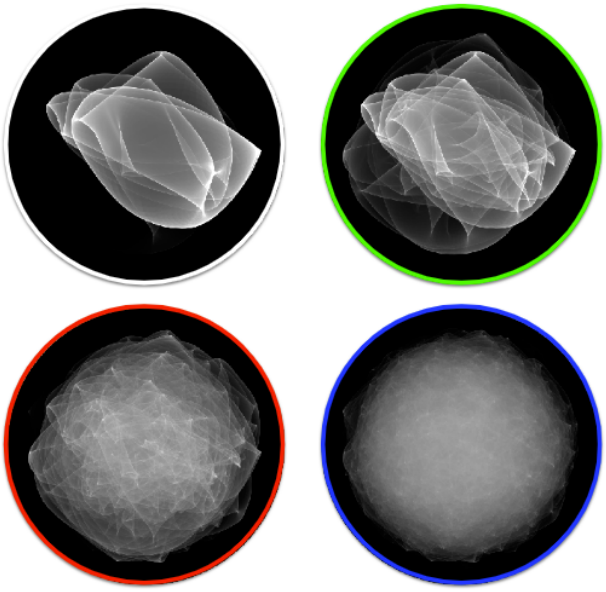


考虑被像素覆盖微表面范围的$\mathcal{P}-\mathrm{NDF}$，替代原来的光滑分布

p-NDFs have sharp features

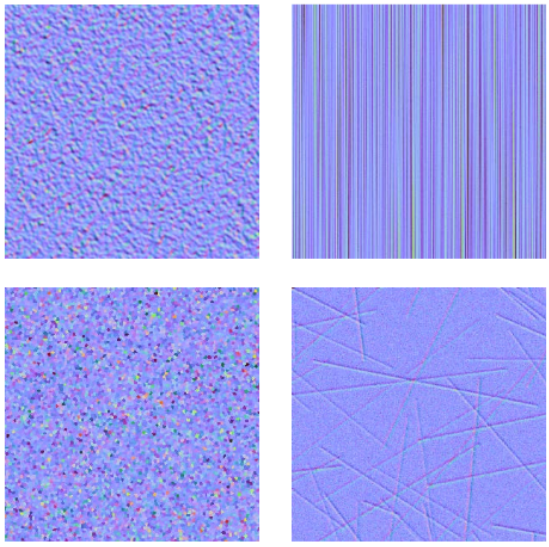


normal map

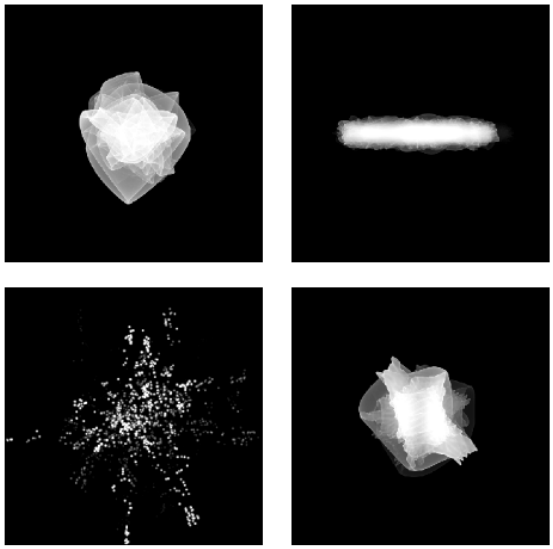


$\mathcal{P}$-NDFs

p-NDF shapes

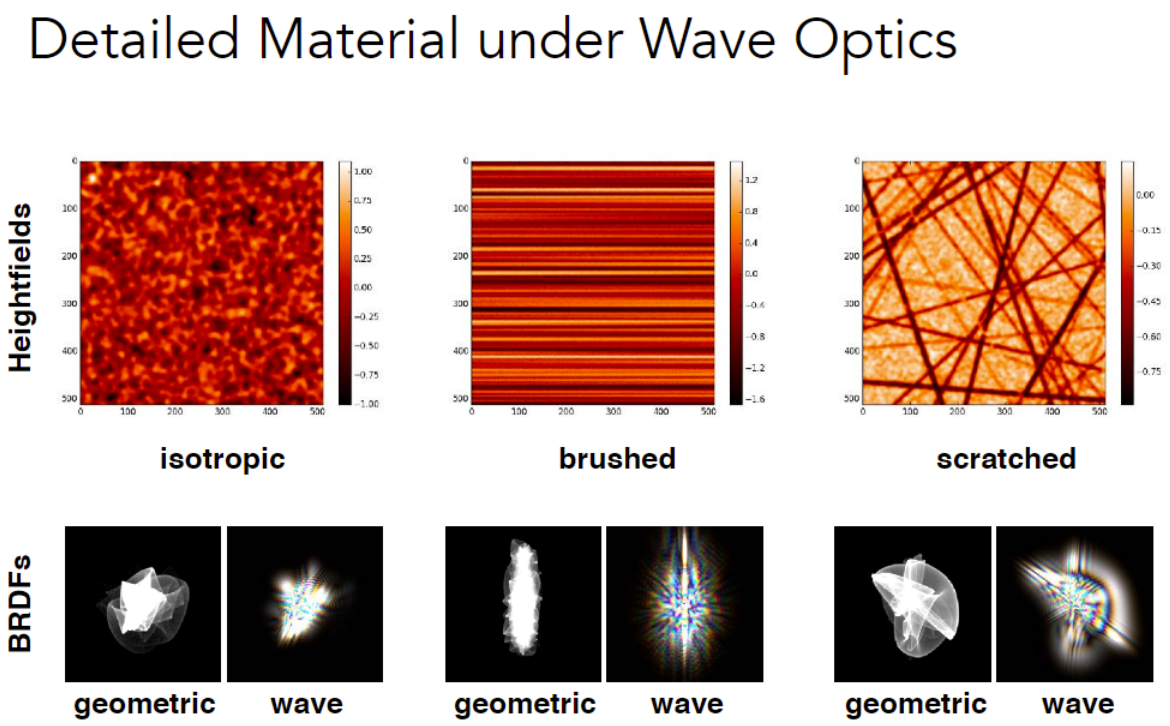


normal maps



$\mathcal{P}$-NDFs

problem: 尺度太小，需要引入波动光学



BRDF会受到干涉影响，不连续

Procedural Appearance (程序生成)

Can we define details without textures?

Yes! Define a noise function on the fly. (e.g. Perlin Noise)

- 3D noise -> internal structure
- Thresholding (noise -> binary noise)
- Complex noise functions can be very powerful.

相机与透镜

相机

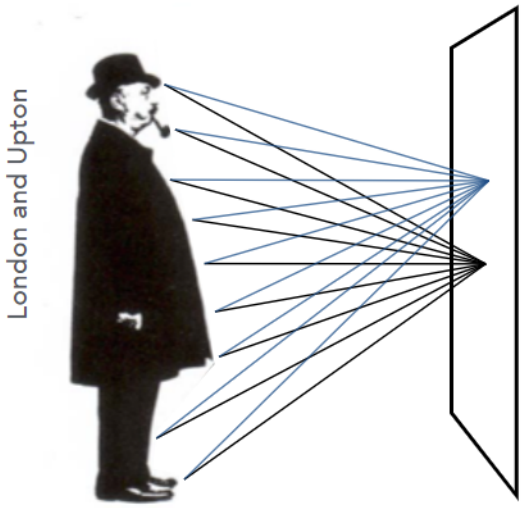
Imaging = Synthesis + Capture

What's Happening Inside the Camera?

1. Pinholes (小孔) & Lenses (透镜) Form Image on Sensor
2. Shutter (快门) Exposes Sensor For Precise Duration (控制进光时间)
3. Sensor Accumulates Irradiance During Exposure

Why Not Sensors Without Lenses?

Each sensor point would integrate light from all points on the object, so all pixel values would be similar
i.e. the sensor records **irradiance**



but there is computational imaging research…

Pinhole Image Formation

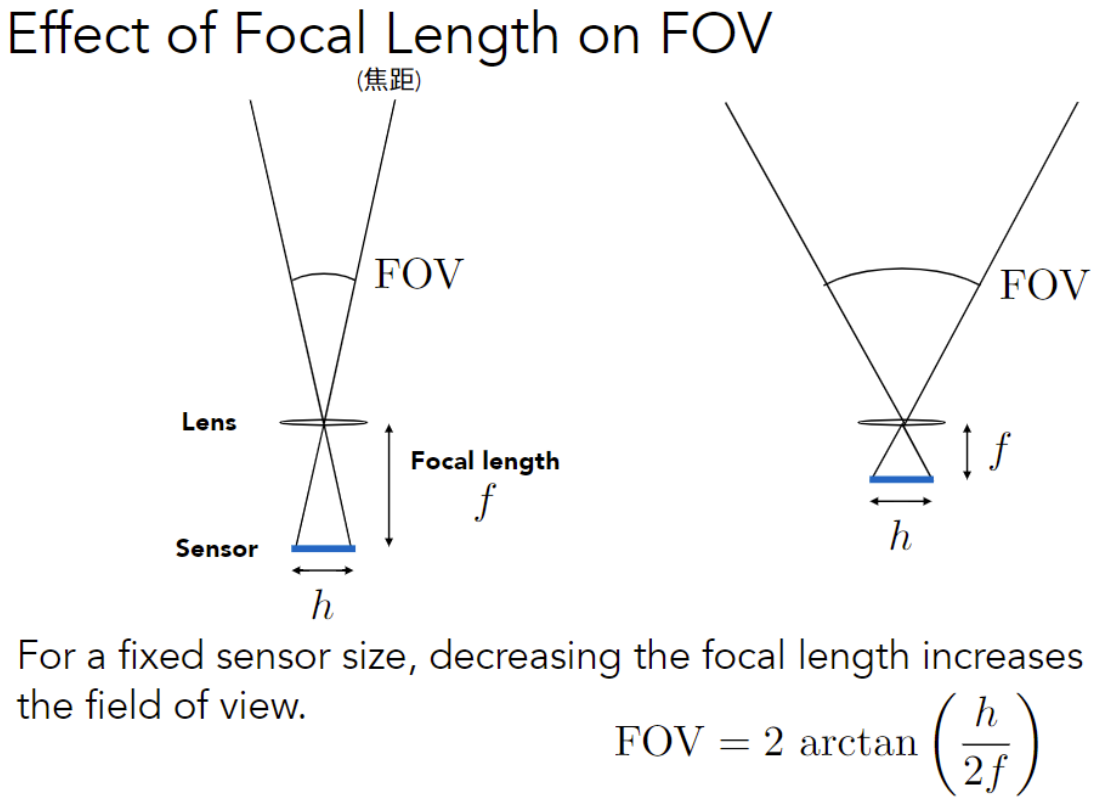
小孔成像

注：Pinhole的结果没有深度，不会有景深

Field of View (FOV) (视场)

观测范围

Focal Length (焦距)



For historical reasons, it is common to refer to angular field of view by focal length of a lens used on a 35mm-format film (36 x 24mm)

Maintain FOV on Smaller Sensor: decrease focal length of lens

Exposure (曝光度)

H = T x E (总能量)

Exposure = time x irradiance

- Exposure time (T)

  Controlled by shutter

- Irradiance (E)

  Power of light falling on a unit area of sensor

  Controlled by lens aperture (光圈) and focal length

Exposure Controls in Photography

Aperture size (类似瞳孔)

- Change the f-stop by opening / closing the aperture (if camera has iris control)

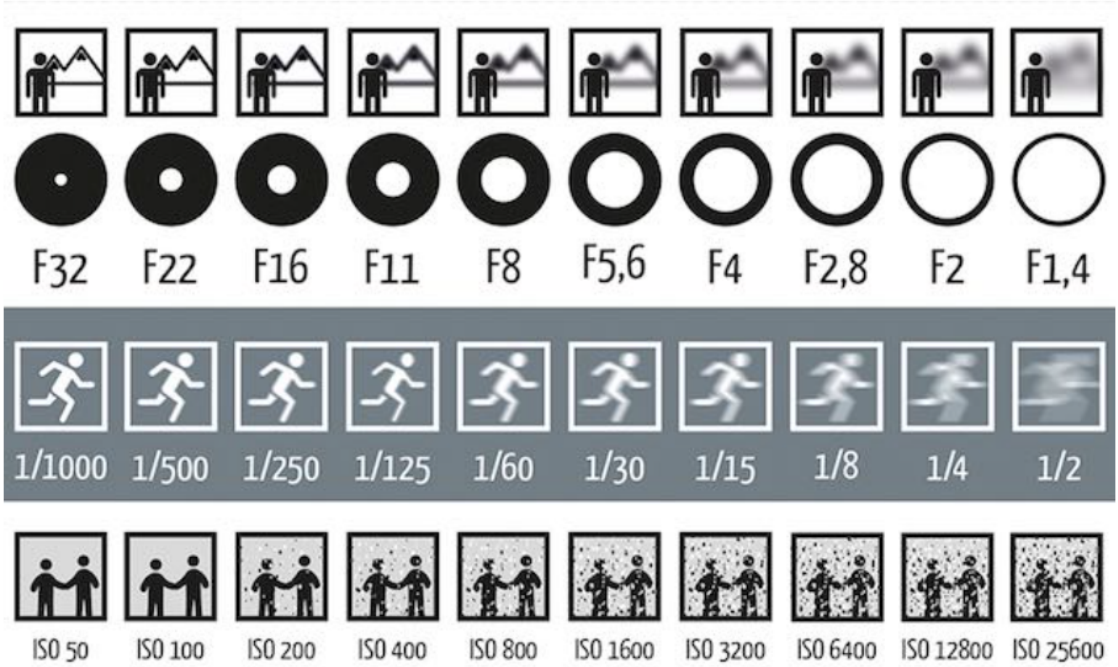
Shutter speed

- Change the duration the sensor pixels integrate light

ISO gain (感光度)

- Change the amplification (analog and/or digital) between sensor values and digital image values

Exposure: Aperture, Shutter, Gain (ISO)



ISO (Gain)

也会放大噪声

Film: trade sensitivity for grain

Digital: trade sensitivity for noise

- Multiply signal before analog-to-digital conversion
- Linear effect (ISO 200 needs half the light as ISO 100)

F-Number (F-Stop): Exposure Levels

Written as FN or F/N. N is the f-number.

Informal understanding: the inverse-diameter of a round aperture

Shutter Speed

控制进光时间

Side Effect of Shutter Speed

- Motion blur

  Note: motion blur is not always bad! (think about anti-aliasing, 时间采样)
- Rolling shutter

  相对于超高速物体，快门打开过程有一定时间

  different parts of photo taken at different times

Constant Exposure: F-Stop vs Shutter Speed

Photographers must trade off depth of field (景深, 大光圈引起) and motion blur for moving subjects

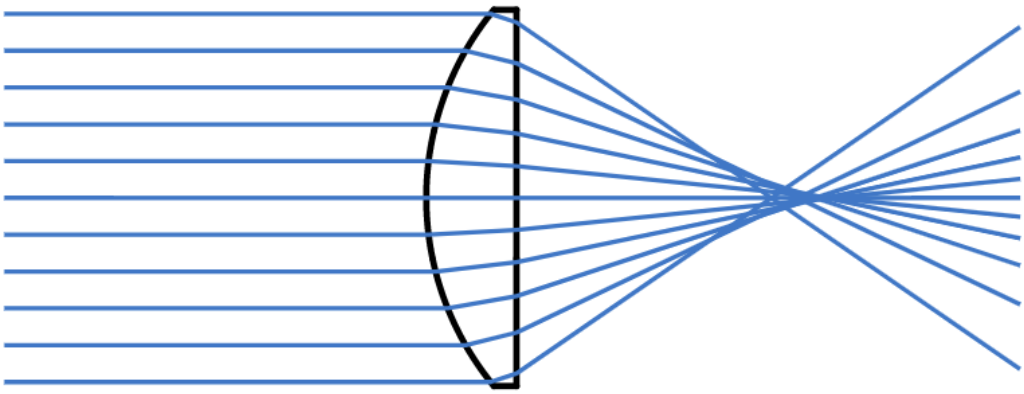
Fast and Slow Photography

Normal exposure = extremely fast shutter speed x (large aperture and/or high ISO)

- High-Speed Photography
- Long-Exposure Photography (延时摄影)

透镜

- Real Lens Designs Are Highly Complex
- Real Lens Elements Are Not Ideal – Aberrations



Real plano-convex lens (spherical surface shape).

Lens does not converge rays to a point anywhere

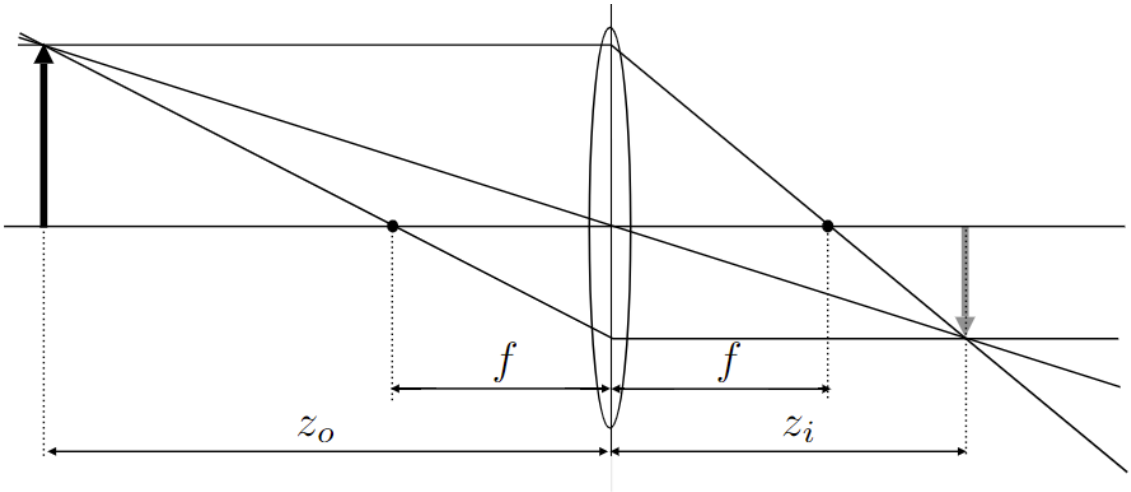
Thin Lens Approximation

Ideal Thin Lens (不考虑厚度)

1. All parallel rays entering a lens pass through its focal point. (Parallel Ray)
2. All rays through a focal point will be in parallel after passing the lens. (Focal Ray)
3. Focal length can be arbitrarily changed (in reality, yes! 棱镜组).

The Thin Lens Equation

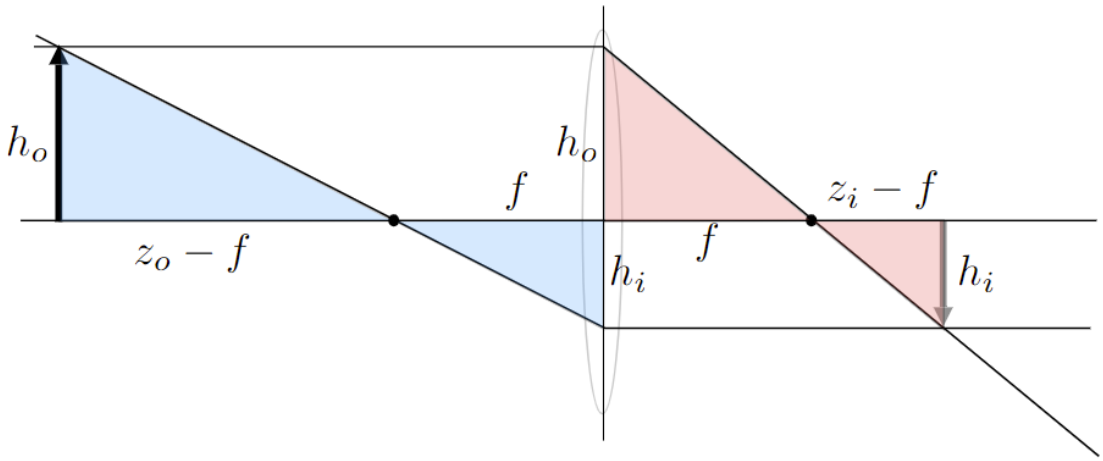
平行光过焦点，穿过中心的光 (Chief Ray) 不改变方向



$$\boxed{\frac{1}{f} = \frac{1}{z_i} + \frac{1}{z_o}}$$

证明：

Gauss' Ray Tracing Construction



$$\frac{h_o}{z_o - f} = \frac{h_i}{f} \qquad \frac{h_o}{f} = \frac{h_i}{z_i - f}$$

$$\frac{h_o}{h_i} = \frac{z_o - f}{f} \qquad \frac{h_o}{h_i} = \frac{f}{z_i - f}$$

$$\frac{z_o - f}{f} = \frac{f}{z_i - f}$$

Object / image heights factor out - applies to all rays

$$(z_o - f)(z_i - f) = f^2$$

Newtonian Thin Lens Equation

$$z_o z_i - (z_o + z_i)f + f^2 = f^2$$

$$z_o z_i = (z_o + z_i)f$$

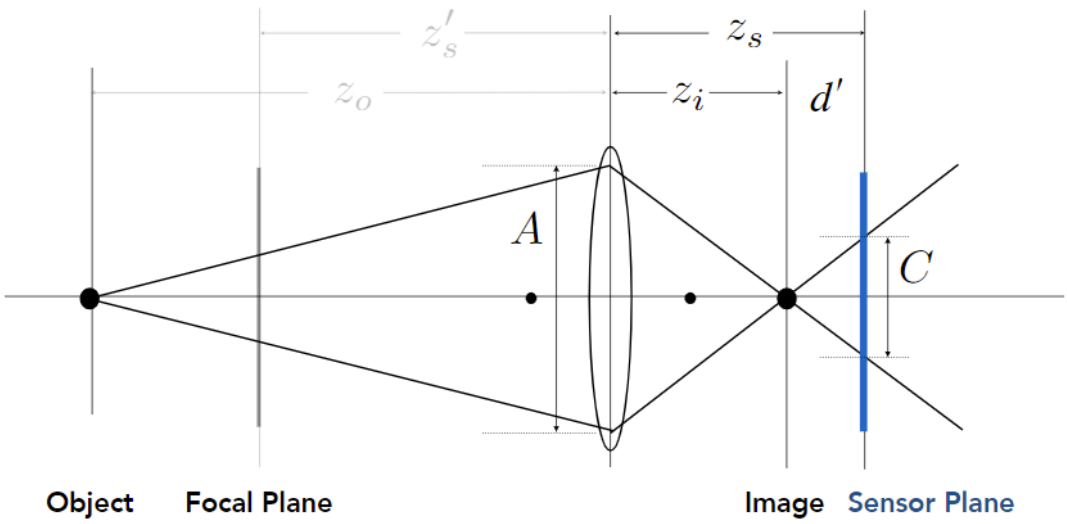
$$\frac{1}{f} = \frac{1}{z_i} + \frac{1}{z_o}$$

Gaussian Thin Lens Equation

Defocus Blur

造成景深

Computing Circle of Confusion (**CoC**) Size



Circle of confusion is proportional to the size of the aperture

$$\frac{C}{A} = \frac{d'}{z_i} = \frac{|z_s - z_i|}{z_i}$$

- 假设focal plane经过透镜被聚焦到sensor plane
- 若物体在focal plane之外，聚焦的image会在sensor plane之前，

  光线继续传播到sensor plane变成圆 (CoC), 造成confusion
- A越大，CoC越大，景深越严重

Revisiting F-Number (a.k.a. F-Stop)

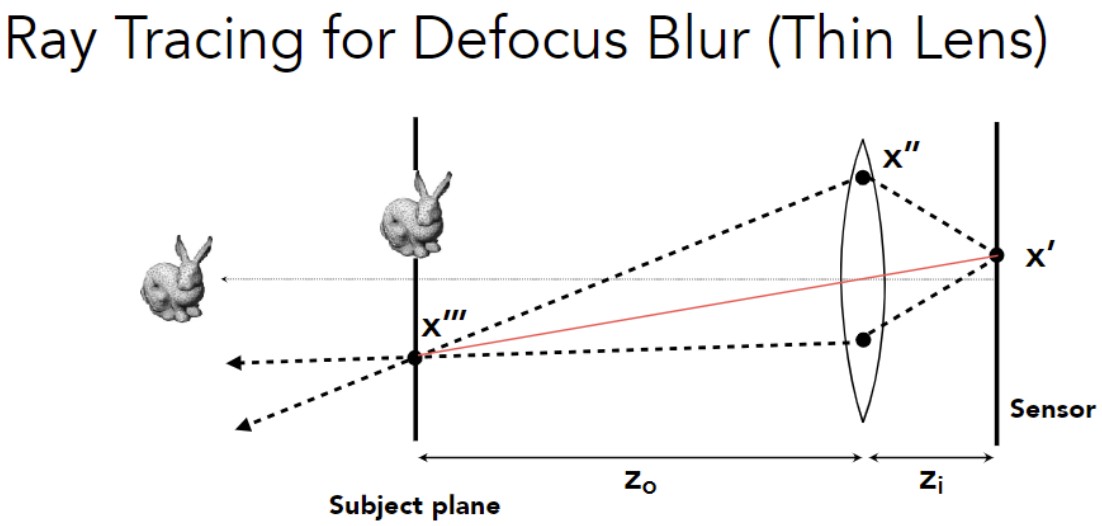
Formal definition: The f-number of a lens is defined as the focal length divided by the diameter of the aperture

$$N = f/D$$

$$C = A\frac{|z_s - z_i|}{z_i} = \frac{f}{N}\frac{|z_s - z_i|}{z_i}$$

Ray Tracing Ideal Thin Lenses

之前提到的RT假设相机是小孔，无法形成景深



(One possible) Setup:

- Choose sensor size (成像平面), lens focal length and aperture size
- Choose depth of subject of interest z_o (拍摄平面)

  Calculate corresponding depth of sensor z_i from thin lens equation

Rendering:

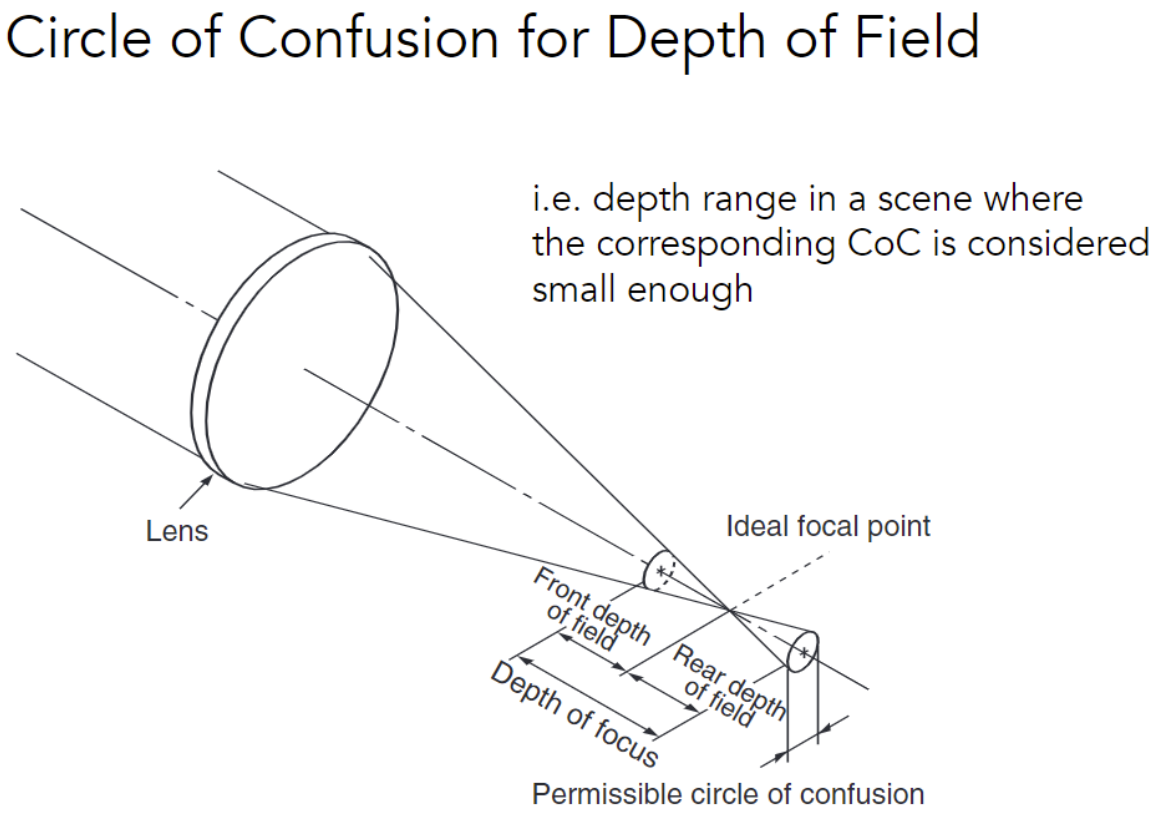
- For each pixel x’ on the sensor (actually, film (胶片))
- Sample random points x” on lens plane
- You know the ray passing through the lens will hit x’”

  because x’” is in focus, consider virtual ray (x’, center of the lens)) (考虑红色直线与其下方的黑色虚线)
- Estimate radiance on ray x” -> x’”

Depth of Field

景深：画面中景象清晰的深度范围

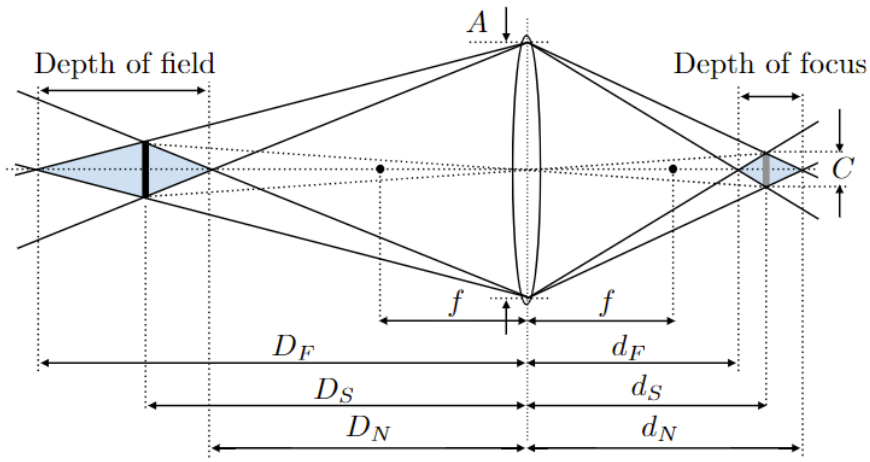
focal plane的物体不模糊，光圈影响模糊的范围



- 景象清晰 = CoC在focal plane附件足够小 (pixel有大小)

  求解这些CoC对应的区域

Depth of Field (FYI)



$$\frac{d_N - d_S}{d_N} = \frac{C}{A}$$
$$\frac{d_S - d_F}{d_F} = \frac{C}{A}$$
$$N = \frac{f}{A}$$
$$\frac{1}{D_F} + \frac{1}{d_F} = \frac{1}{f}$$
$$\frac{1}{D_S} + \frac{1}{d_S} = \frac{1}{f}$$
$$\frac{1}{D_N} + \frac{1}{d_N} = \frac{1}{f}$$

$$\mathrm{DOF} = D_F - D_N$$

$$D_F = \frac{D_S f^2}{f^2 - NC(D_S - f)} \qquad D_N = \frac{D_S f^2}{f^2 + NC(D_S - f)}$$

光场、颜色与感知

Light Field / Lumigraph (光场)

What do we see? 看到各个方向的光线

如果有Painted backdrop可以完美记录这些信息，看Painted backdrop和真实看到的景象一样 (虚拟现实)

The Plenoptic Function (全光函数)

Q: What is the set of all things that we can ever see?

A: The Plenoptic Function (Adelson & Bergen)

Grayscale snapshot: $P(\theta, \phi)$

intensity of light

- Seen from a single view point
- At a single time
- average on wavelength

Color snapshot: $P(\theta, \phi, \lambda)$

- As a function of wavelength

Movie: $P(\theta, \phi, \lambda, t)$

- 在时间t，看向θ, ϕ方向，看到的波长(颜色)为λ的光的强度

Holographic movie (全息电影): $P\left(\theta, \phi, \lambda, t, V_X, V_Y, V_Z\right)$

- Seen from ANY viewpoint

The Plenoptic Function

$$P\left(\theta, \phi, \lambda, t, V_X, V_Y, V_Z\right)$$

- Can reconstruct every possible view, at every moment, from every position, at every wavelength
- Contains every photograph, every movie, everything that anyone has ever seen! it completely captures our visual reality!
- This is what we see!

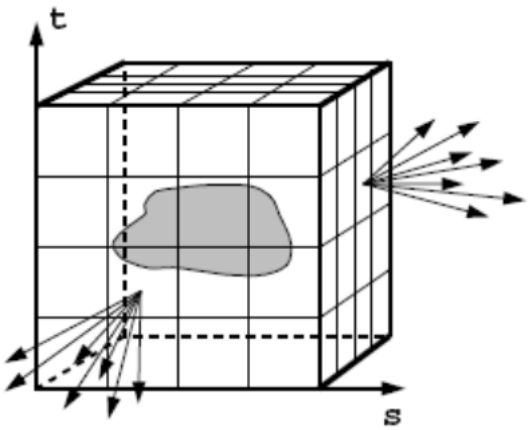
Light Field / Lumigraph (光场)

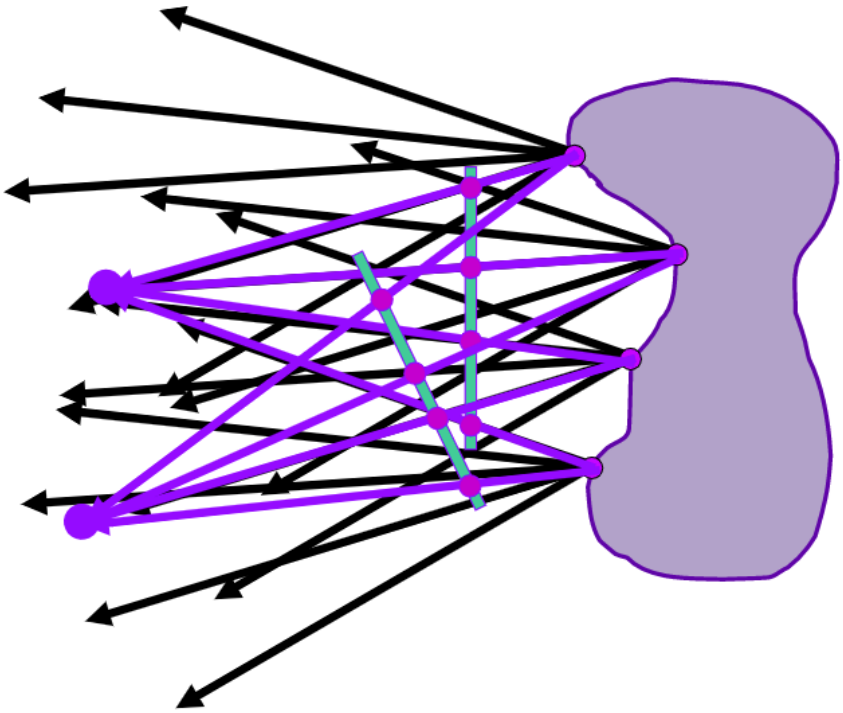
Ray = 2D direction + 2D position

Q：如何记录物体?

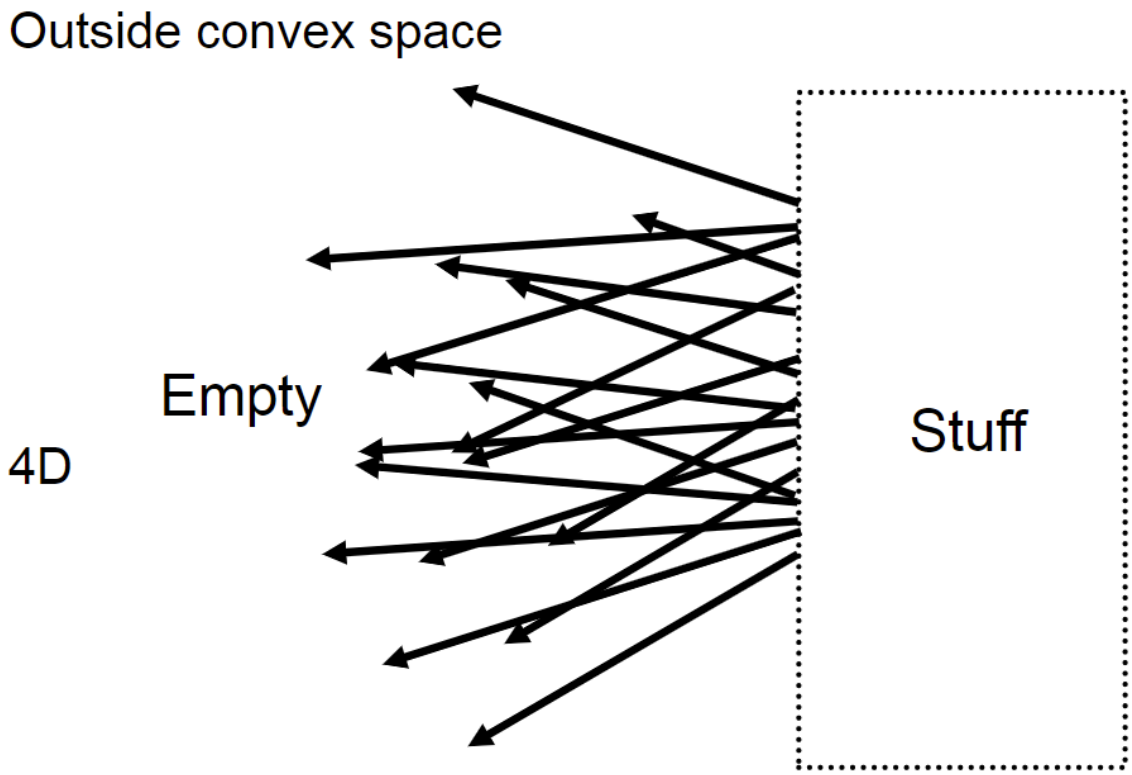
Only need plenoptic surface

记录任何位置往外部不同方向投射的光线强度 (即部分全光函数) (4D)

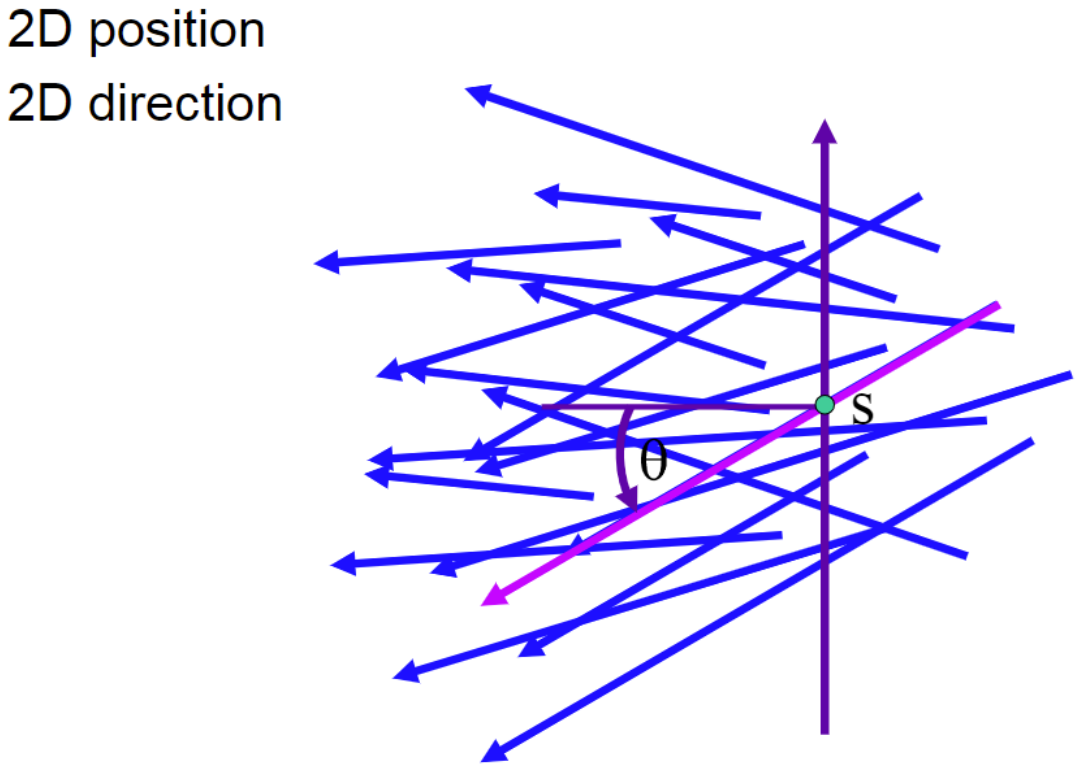




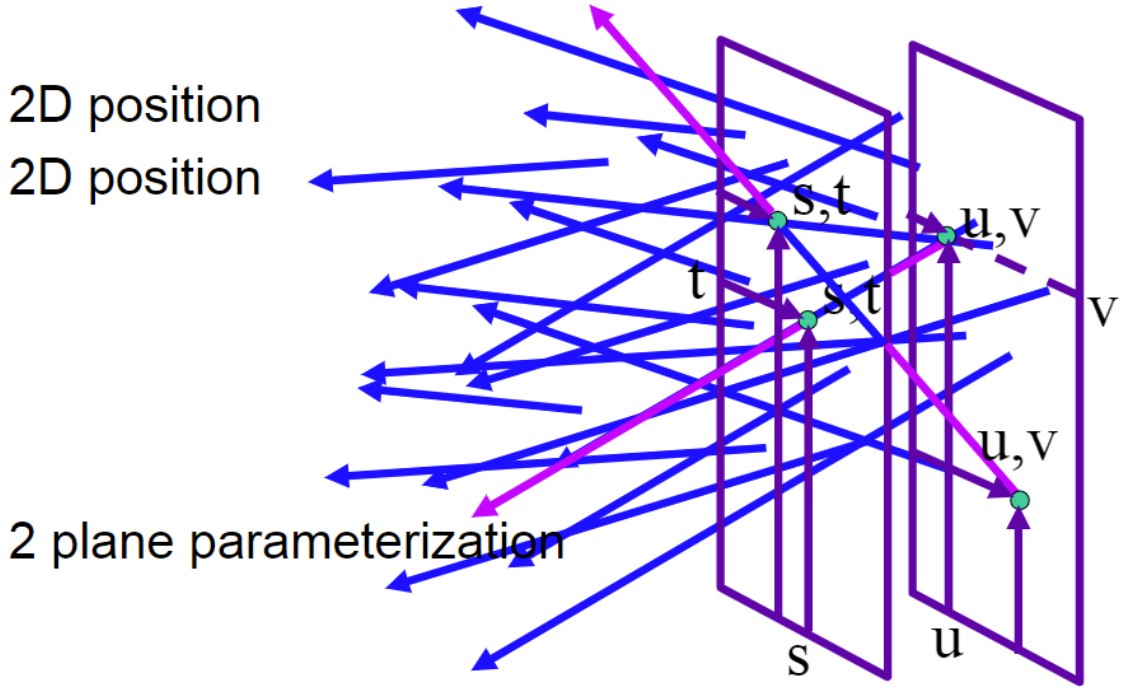
可以得到任意位置看到物体的结果



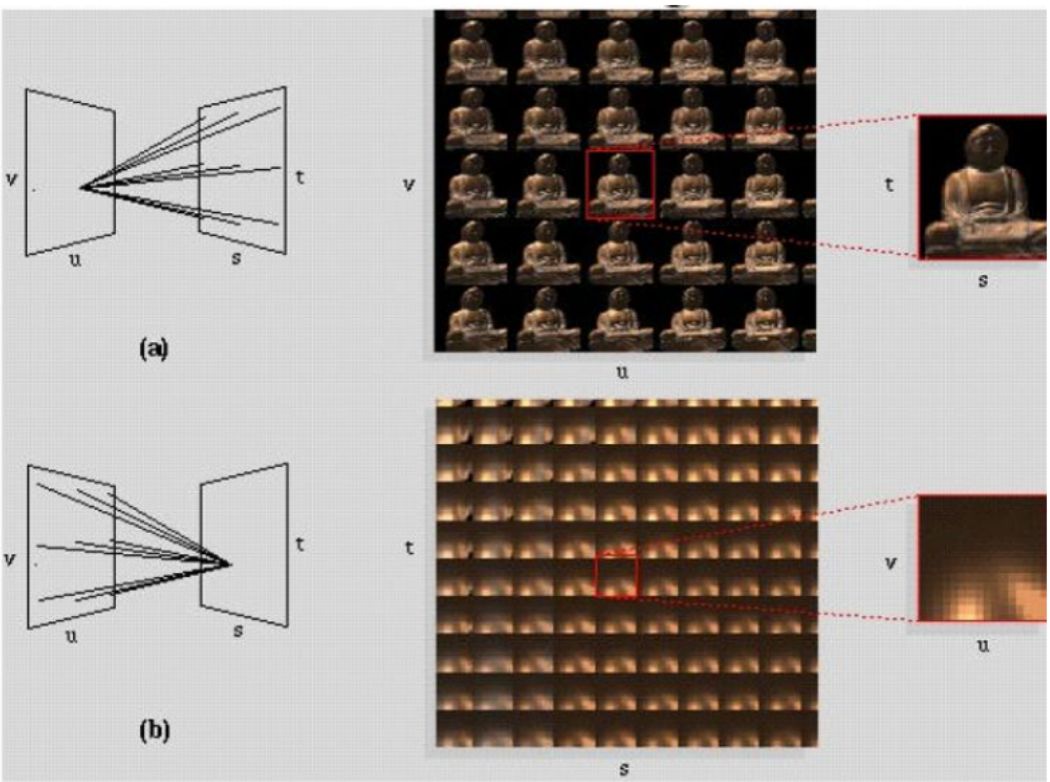
Q：如何记录光场？



取一个平面，对于平面任意点记录经过的光线方向



光场 = 2个平面 + $\{(u, v, s, t)\}$



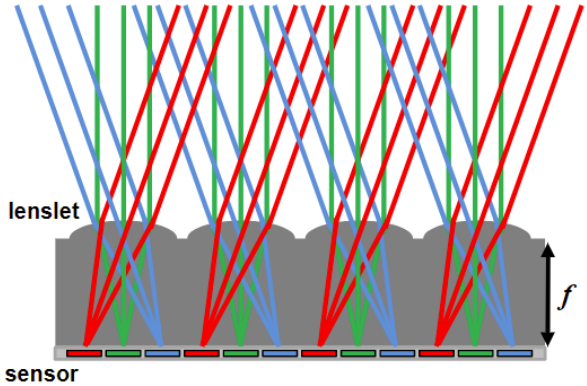
- hold u,v

  不同角度的图像

- hold s,t

  irradiance展开成radiance，打到某个点上不同方向的光

  Light Field Camera原理



Credit: JJ Harrison

普通相机在图中Lenslets的位置直接用sensor记录光线，得到平均结果（irradiance）

而 Lenslets (小透镜) 可以把光线分开，在底下的sensor分别记录（radiance）

注：Light Field Camera没有普通相机的Lens

Light Field Camera

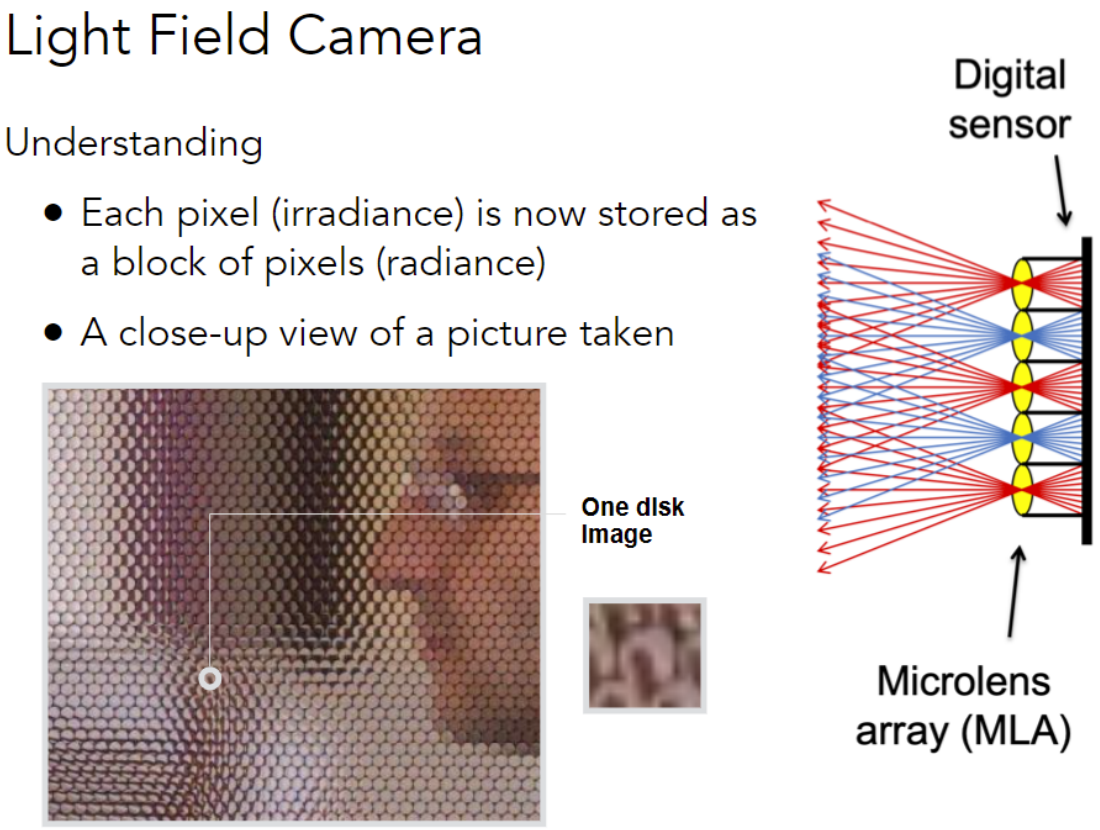
注：Light Field Camera没有普通相机的Lens

Microlens design

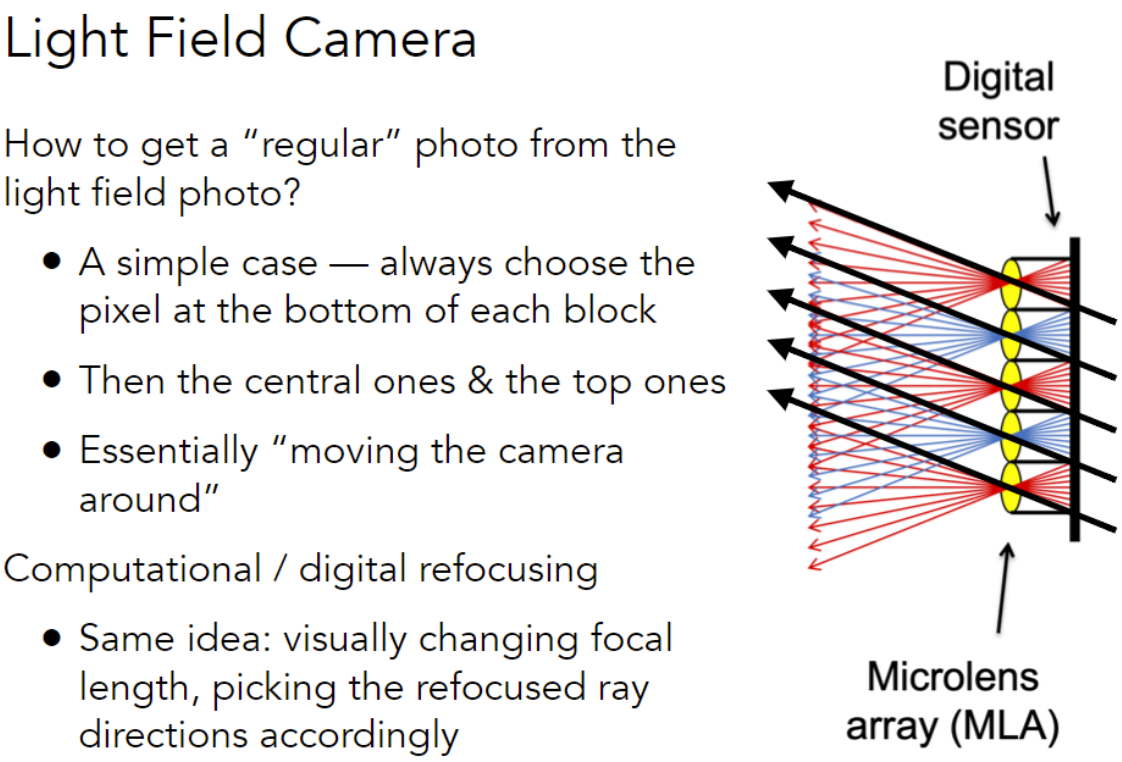
Most significant function

- Computational Refocusing

  (virtually changing focal length & aperture size, etc. after taking the photo)



Q: How to get a “regular” photo from the light field photo?



得到的结果就是相机看向黑色箭头方向（即在Lec19的Why Not Sensors Without Lenses?情况下可以直接区分radiance）

In all, all these functionalities are available because

- The light field contains everything

Any problems to light field cameras?

- Insufficient spatial resolution (需要记录四维信息)

  (same film used for both spatial and directional information)
- High cost (intricate designation of microlenses)

Color

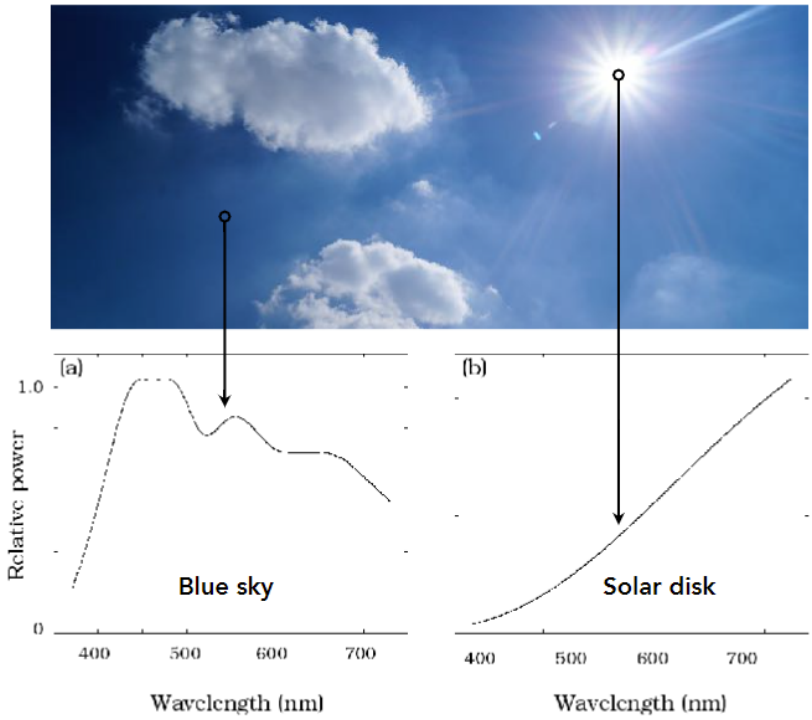
Physical Basis of Color

Spectrum: 光包含的成分信息

Spectral Power Distribution (SPD)

光线在不同波长谱功率密度的分布，具有可加性 (光线叠加)

Daylight Spectral Power Distributions Vary



What is color

- Color is a phenomenon of human perception; it is not a universal property of light
- Different wavelengths of light are not “colors”

Biological Basis of Color

Retinal Photoreceptor Cells (感光细胞)

- Rod Cells

  Rods are primary receptors in very low light (“scotopic” conditions), e.g. dim moonlight

  - ~120 million rods in eye
  - 感知强度，但不感知颜色

    Perceive only shades of gray, no color

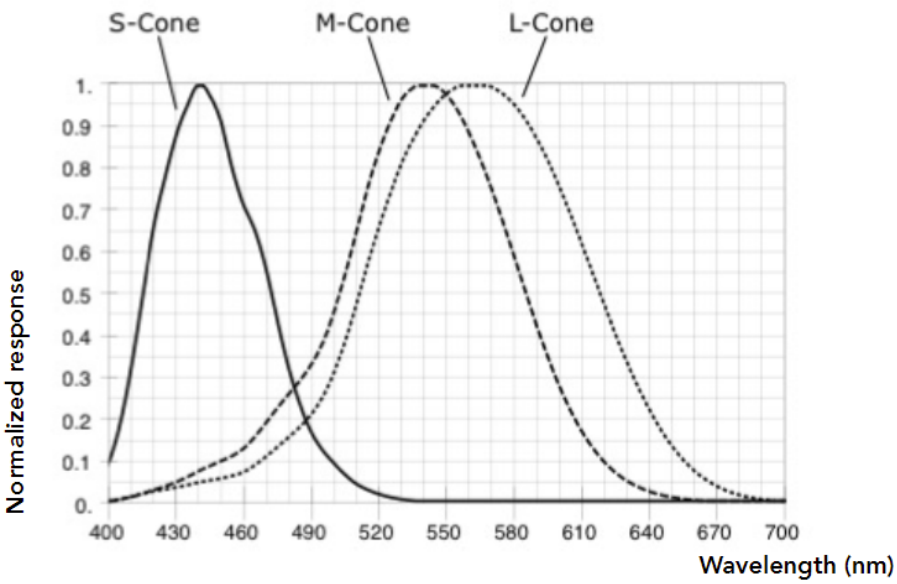
- Cone Cell

  Cones are primary receptors in typical light levels (“photopic”)

  - ~6-7 million cones in eye
  - Three types of cones, each with different spectral sensitivity
  - Provide sensation of color (感知颜色)

Spectral Response of Human Cone Cells

Three types of **cone** cells: S, M, and L (corresponding to peak response at short, medium, and long wavelengths)



Tristimulus Theory of Color

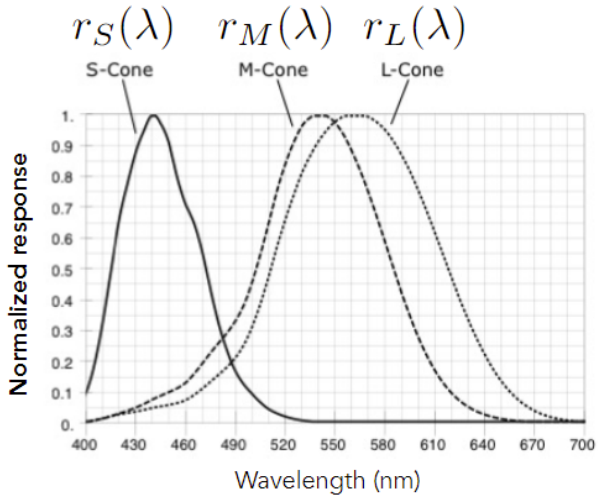
Spectral Response of Human Cone Cells

Now we have three detectors (S, M, L cone cells), each with a different spectral response curve

$$S = \int r_S(\lambda) s(\lambda)\, d\lambda$$

$$M = \int r_M(\lambda) s(\lambda)\, d\lambda$$

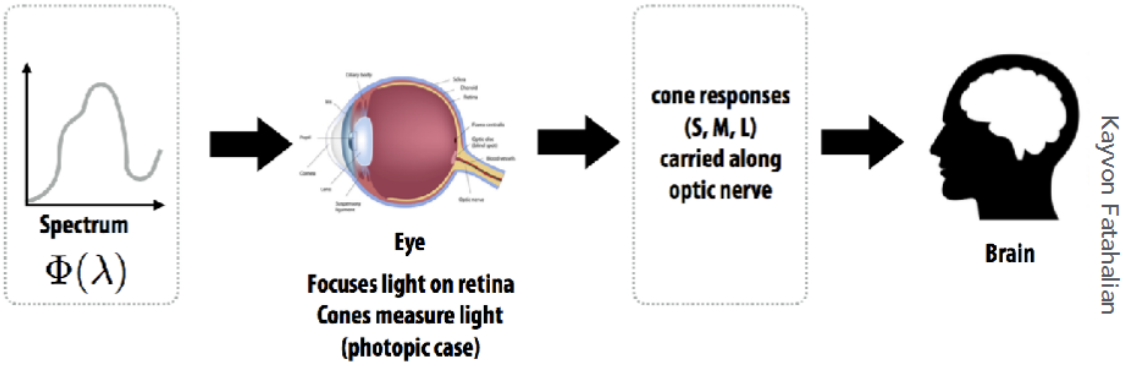
$$L = \int r_L(\lambda) s(\lambda)\, d\lambda$$



The Human Visual System

- Human eye does not measure and brain does not receive information about each wavelength of light

- Rather, the eye “sees” only three response values (S, M, L), and this is only info available to brain



Metamerism (同色异谱)

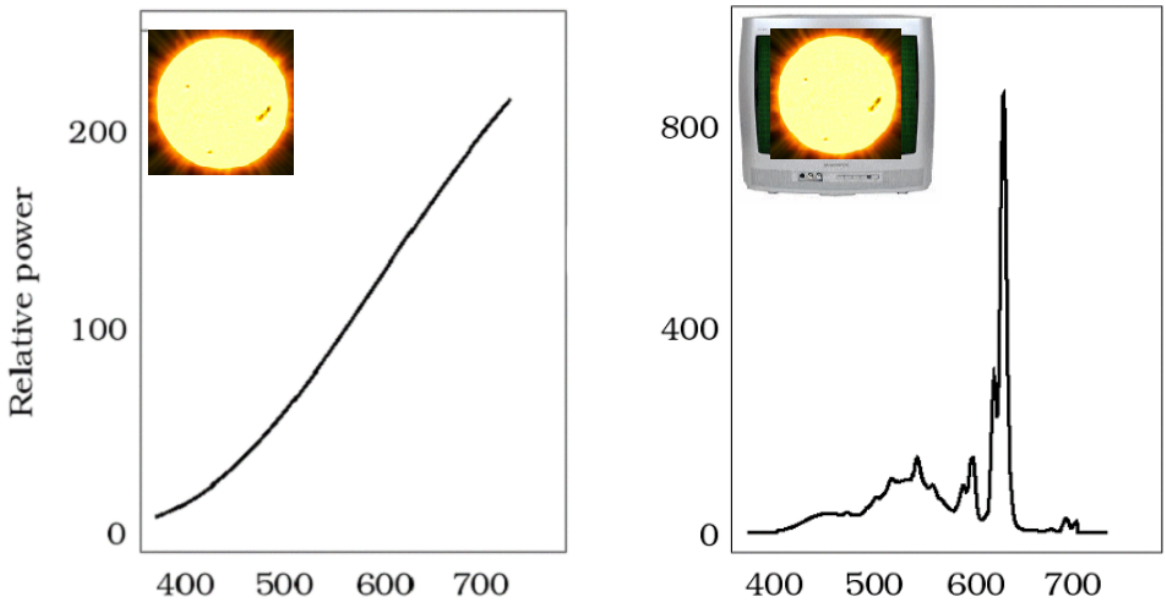
Metamers are two different spectra (∞-dim) that project to the same (S,M,L) (3-dim) response.

- These will appear to have the same color to a human

The existence of metamers is critical to color reproduction

- Don’t have to reproduce the full spectrum of a real world scene (不需要光谱一致)
- Example: A metamer can reproduce the perceived color of a real-world scene on a display with pixels of only three colors

The theory behind color matching



Color Reproduction / Matching

Additive model: 越混合越白

Additive Color

- Given a set of primary lights, each with its own spectral distribution (e.g. R,G,B display pixels):

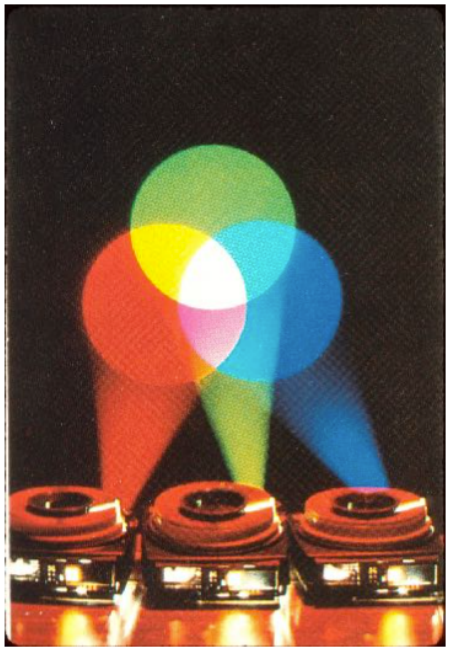
$$s_R(\lambda),\ \ s_G(\lambda),\ \ s_B(\lambda)$$

- Adjust the brightness of these lights and add them together:

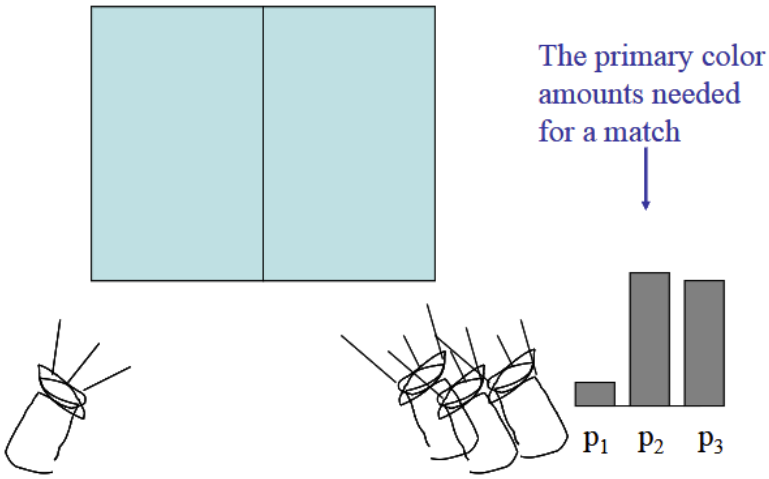
$$R\, s_R(\lambda) + G\, s_G(\lambda) + B\, s_B(\lambda)$$

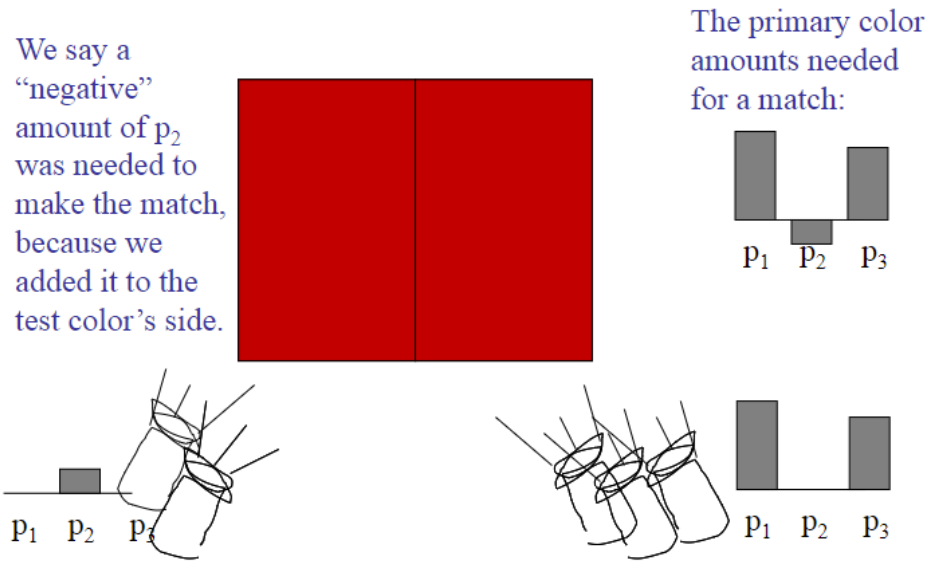
- The color is now described by the scalar values:

$$R,\ G,\ B$$



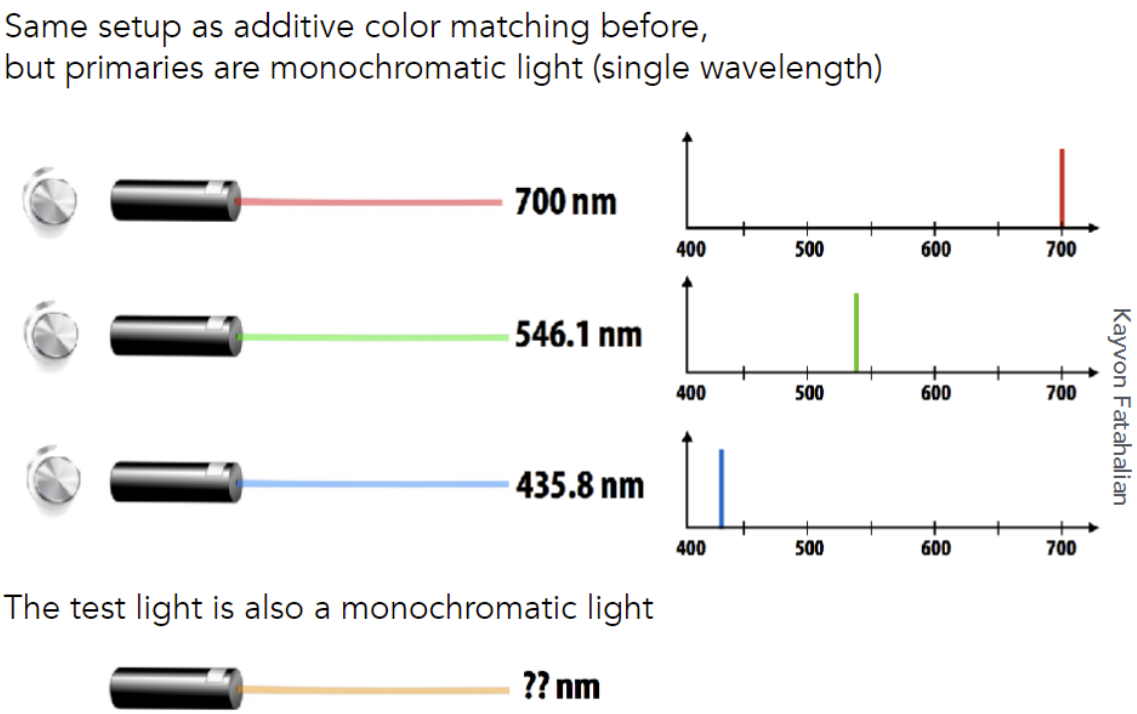
Additive Color Matching Experiment





如果basis选不好系数可能是负数

CIE RGB Color Matching Experiment



Graph plots how much of each CIE RGB primary light must be combined to match a monochromatic light of wavelength given on x-axis

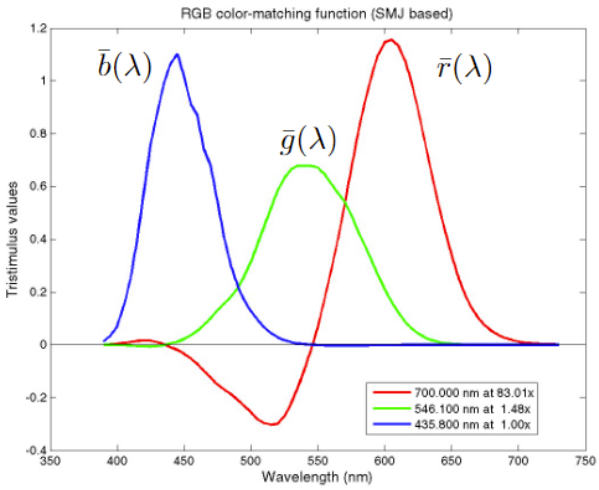
For any spectrum s, the perceived color is matched by the following formulas for scaling the CIE RGB primaries

any spectrum s = combination of monochromatic light

$$R_{\text{CIE RGB}} = \int_\lambda s(\lambda)\bar{r}(\lambda)\,d\lambda$$

$$G_{\text{CIE RGB}} = \int_\lambda s(\lambda)\bar{g}(\lambda)\,d\lambda$$

$$B_{\text{CIE RGB}} = \int_\lambda s(\lambda)\bar{b}(\lambda)\,d\lambda$$



Careful: these are not response curves or primary spectra!

Color space

Standardized RGB (sRGB)

- makes a particular monitor RGB standard
- other color devices simulate that monitor by calibration
- widely adopted today
- gamut (色域) is limited

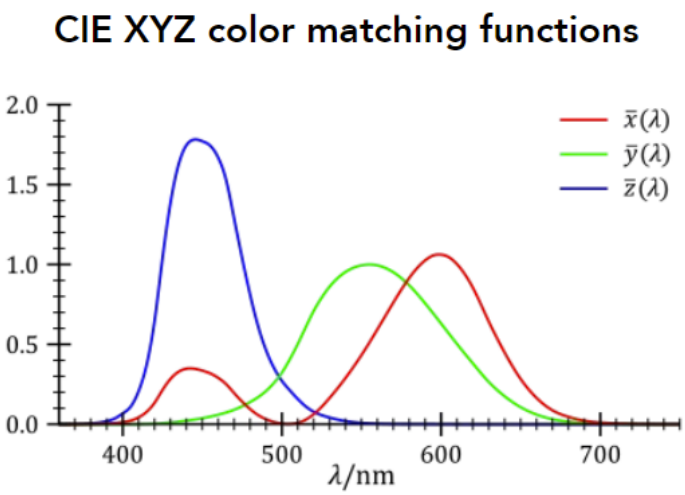
A Universal Color Space: CIE XYZ

Imaginary set of standard color primaries X, Y, Z

- Primary colors with these matching functions do not exist (人为定义)
- Y is luminance (brightness regardless of color) (可以反映亮度)

Designed such that

- Matching functions are strictly positive
- Span all observable colors

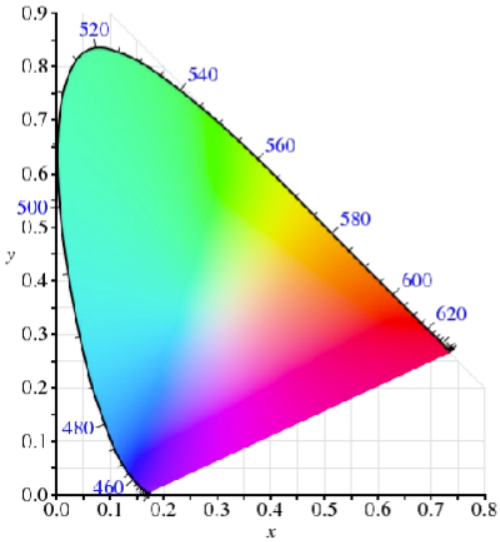


Separating Luminance, Chromaticity

Luminance: Y

Chromaticity: x, y, z, defined as (色度)

$$x = \frac{X}{X+Y+Z}$$
$$y = \frac{Y}{X+Y+Z}$$
$$z = \frac{Z}{X+Y+Z}$$



- since $x + y + z = 1$, we only need to record two of the three
  - usually choose x and y, leading to (x, y) coords at a specific brightness Y

- Y影响亮度，固定Y改变X,Z
- 图像以x,y为坐标，leading to (x, y) coords at a specific brightness Y

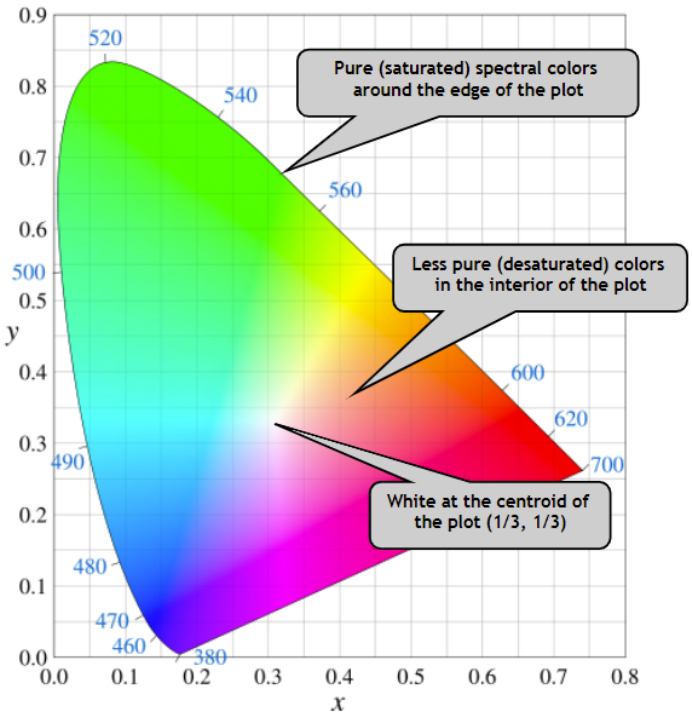
CIE Chromaticity Diagram

The curved boundary

- named spectral locus
- corresponds to monochromatic light (each point representing a pure color of a single wavelength)

Any color inside is less pure

- i.e. mixed

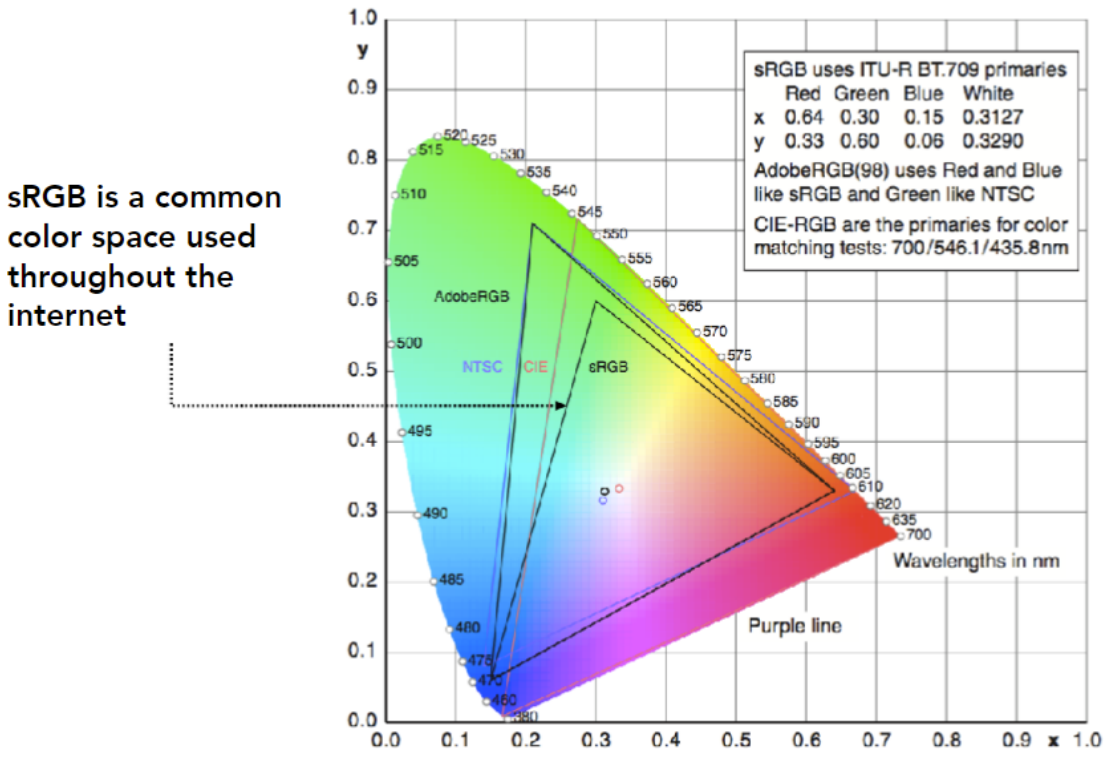


Gamut (色域)

Gamut is the set of chromaticities generated by a set of color primaries

- Different color spaces represent different ranges of colors

  So they have different gamuts, i.e. they cover different regions on the chromaticity diagram

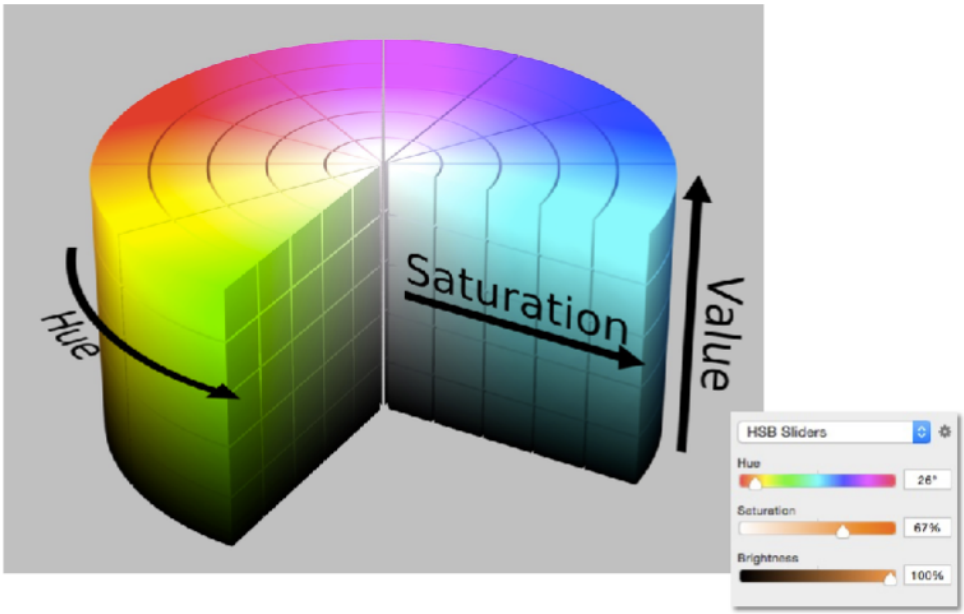


Perceptually Organized Color Spaces

HSV Color Space (Hue-Saturation-Value)

Axes correspond to artistic characteristics of color

Widely used in a "color picker"

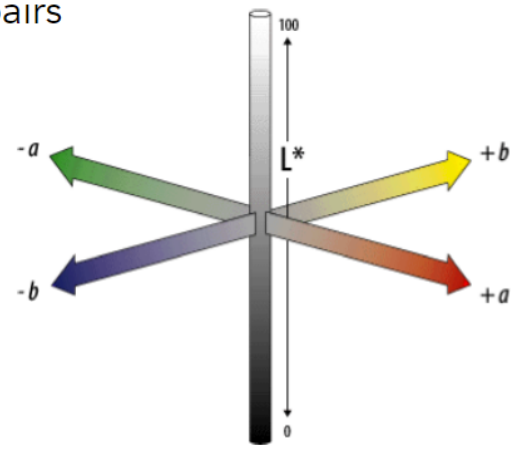


- Hue (色调)
  - the "kind" of color, regardless of attributes (不同颜色)
- Saturation (饱和度)
  - the "colorfulness" (偏白还是偏当前颜色)
- Lightness (or value)
  - the overall amount of light

CIELAB Space (AKA L*a*b*)

A commonly used color space that strives for perceptual uniformity

- L* is lightness (brightness)
- a* and b* are color-opponent pairs
  - a* is red-green
  - b* is blue-yellow



Opponent Color (互补色) Theory (类比反义词)

There's a good neurological basis

- the brain seems to encode color early on using three axes:
  - white — black, red — green, yellow —blue
- the white — black axis is lightness; the others determine hue and saturation

CMYK: A Subtractive Color Space

Subtractive color model: 越混合越黑

Subtractive color model

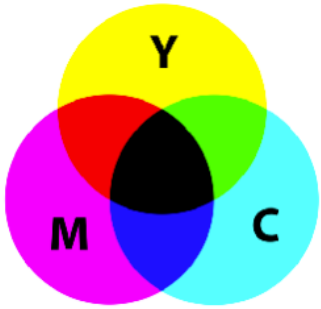
- The more you mix, the darker it will be

Cyan, Magenta, Yellow, and Key

Widely used in printing

Question:

- If mixing C, M and Y gives K, why do you need K?



A：K比较常用，靠混合成本太高

Other Topics

- HDR
- 伽马校正

动画与模拟（基本概念、质点弹簧系统、运动学）

Introduction to Computer Animation

Animation

"Bring things to life"

- Communication tool
- Aesthetic issues often dominate technical issues

An extension of modeling

- Represent scene models as a function of time

Output: sequence of images that when viewed sequentially provide a sense of motion

- Film: 24 frames per second
- Video (in general): 30 fps
- Virtual reality: 90 fps

Keyframe (关键帧) animation

e.g. flash

Animator (e.g. lead animator) creates keyframes

Assistant (person or computer) creates in-between frames ("tweening")

Keyframe Interpolation

Think of each frame as a vector of parameter values

- Linear interpolation usually not good enough
- Recall splines (e.g. 贝塞尔曲线C^1) for smooth / controllable interpolation

Physical simulation

$$F = ma$$

Generate motion of objects using numerical simulation

Mass Spring System (质点弹簧系统)

Example of Modeling a Dynamic System

A Simple Spring

一根弹簧连接a,b质点

Idealized spring

Strength proportional to displacement (Hooke's Law)

$$\boldsymbol{f}_{a\to b} = k_S \frac{\boldsymbol{b}-\boldsymbol{a}}{\|\boldsymbol{b}-\boldsymbol{a}\|}(\|\boldsymbol{b}-\boldsymbol{a}\| - l)$$
$$\boldsymbol{f}_{b\to a} = -\boldsymbol{f}_{a\to b}$$

Problem: oscillates (振动) forever

Dot Notation for Derivatives

$$\dot{\boldsymbol{x}} = \boldsymbol{v}$$
$$\ddot{\boldsymbol{x}} = \boldsymbol{a}$$

Introducing Energy Loss

阻尼振动

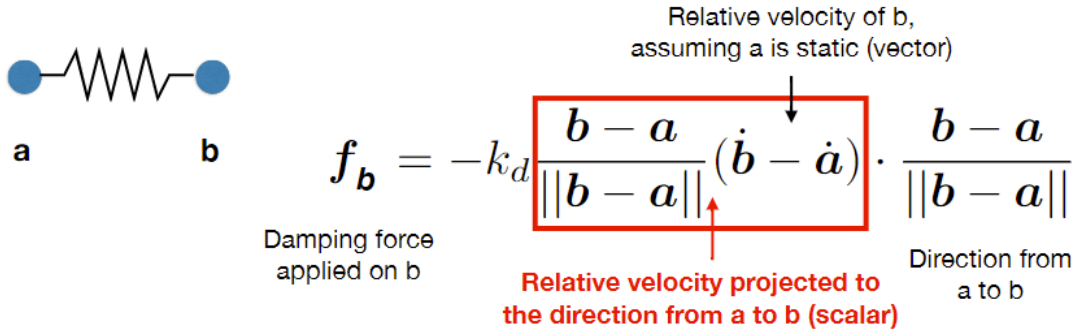
Simple motion damping $\boldsymbol{f} = -k_d\dot{\boldsymbol{b}}$

- Behaves like viscous drag on motion
- Slows down motion in the direction of velocity

Problem: slows down all motion

- should spring also fall to the ground more slowly? (a,b没有相对运动)

<u>Internal</u> Damping for Spring



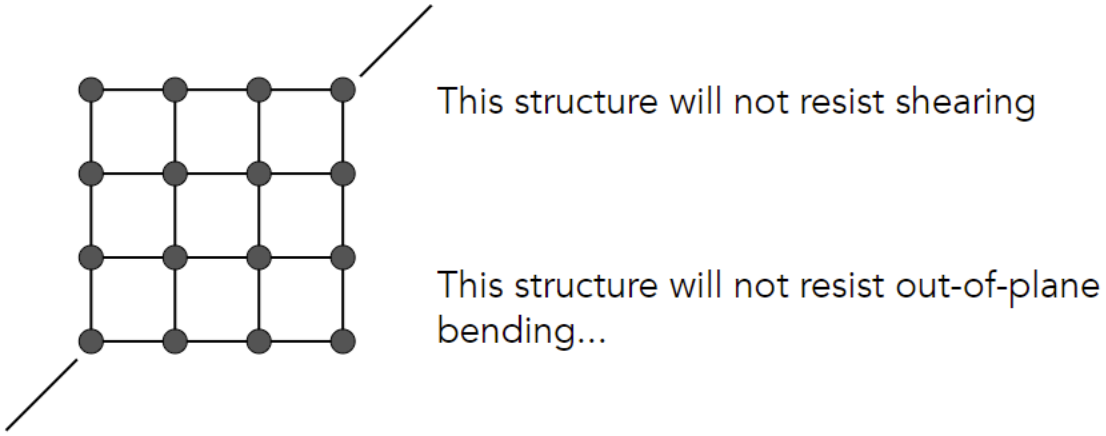
- 不妨以a点为参考系，计算$(\dot{\boldsymbol{b}}-\dot{\boldsymbol{a}})$在$\mathbf{b}-\mathbf{a}$的投影大小（e.g. 圆周运动）
- Damp only the internal, spring-driven motion
- Viscous drag only on change in spring length

Won't slow group motion for the spring system

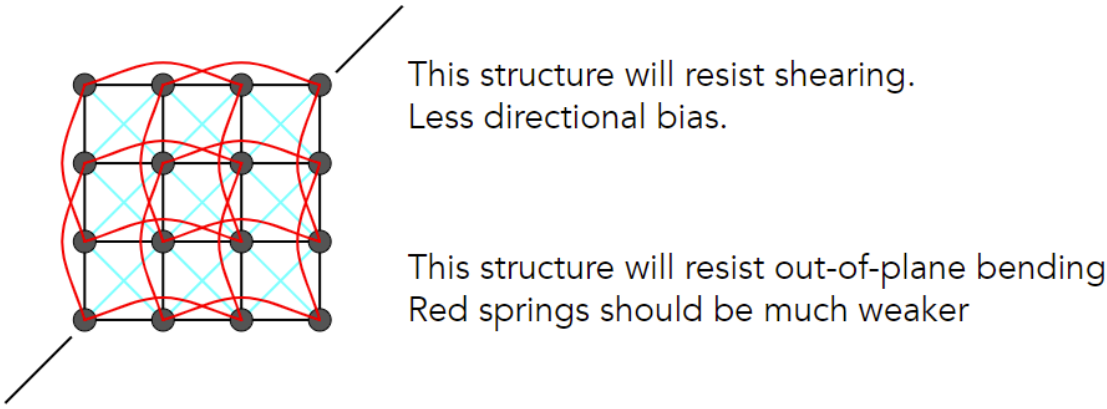
- Note: This is only one specific type of damping

Structures from Springs

Behavior is determined by structure linkages



- 无法抵抗shear（e.g. 拉开左下角和右上角）
- 无法抵抗out-of-plane bending（e.g. 沿着竖直方向弯折）



Aside: FEM (Finite Element Method) Instead of Springs

Particle Systems (粒子系统)

Model dynamical systems as collections of large numbers of particles

Each particle's motion is defined by a set of physical (or non-physical) forces

Popular technique in graphics and games

- Easy to understand, implement
- Scalable: fewer particles for speed, more for higher complexity

Challenges

- May need many particles (e.g. fluids)
- May need acceleration structures (e.g. to find nearest particles for interactions)

Animations

For each frame in animation

- [If needed] Create new particles

- Calculate forces on each particle (hard)
- Update each particle's position and velocity
- [If needed] Remove dead particles
- Render particles

Particle System Forces

Attraction and repulsion forces

- Gravity, electromagnetism, . . .
- Springs, propulsion, ...

Damping forces

- Friction, air drag, viscosity, . . .

Collisions

- Walls, containers, fixed objects, . . .
- Dynamic objects, character body parts, . . .

Example: boid model

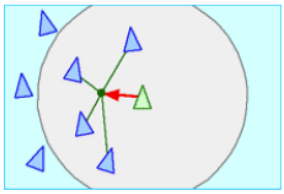
Model each bird as a particle

Subject to very simple forces:

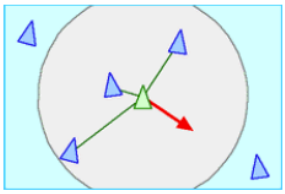
- attraction to center of neighbors
- repulsion from individual neighbors
- alignment toward average trajectory of neighbors

Simulate evolution of large particle system numerically

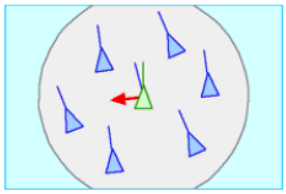
Emergent complex behavior (also seen in fish, bees, ...)



attraction



repulsion



alignment

Kinematics (运动学)

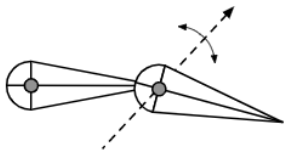
Forward Kinematics

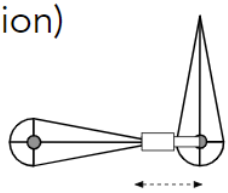
Articulated skeleton

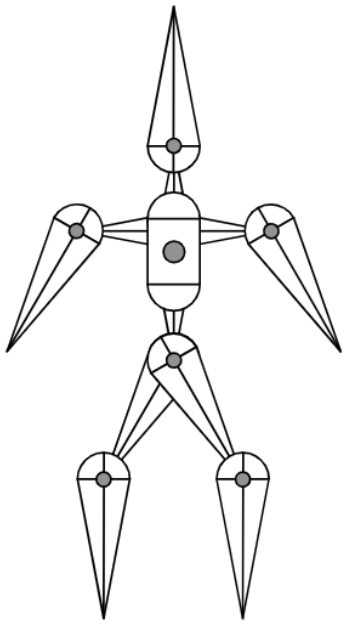
- Topology (what's connected to what)
- Geometric relations from joints
- Tree structure (in absence of loops)

Joint types

- Pin (1D rotation)
- Ball (2D rotation)
- Prismatic joint (translation)

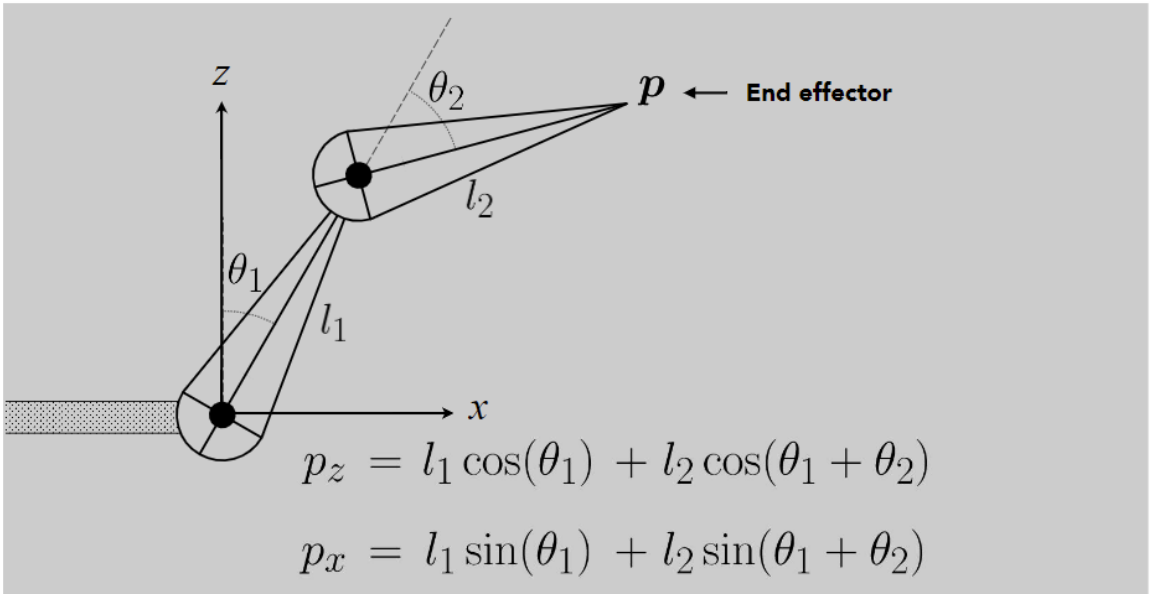






Example: simple two segment arm in 2D

Animator provides angles, and computer determines position *p* of end-effector



Animation is described as angle parameter values as a function of time

Strengths

- Direct control is convenient
- Implementation is straightforward

Weaknesses

- Animation may be inconsistent with physics
- Time consuming for artists

Inverse Kinematics

Animator provides position of end-effector, and computer must determine joint angles that satisfy constraints

Why is the problem hard?

- Multiple solutions in configuration space
- Solutions may not always exist

Numerical solution to general N-link IK problem (优化问题)

- Choose an initial configuration
- Define an error metric (e.g. square of distance between goal and current position)
- Compute gradient of error as function of configuration
- Apply gradient descent (or Newton's method, or other optimization procedure)

Rigging

Rigging is a set of higher level controls (e.g. 控制点) on a character that allow more rapid & intuitive modification of pose, deformations, expression, etc. (逆运动学的应用)

- Like strings on a puppet (提线木偶)
- Captures all meaningful character changes
- Varies from character to character

Blend Shapes

插值控制点，而不是插值整个model

Simplest scheme: take linear combination of vertex positions

Spline used to control choice of weights over time

Motion Capture

给人加控制点，并记录控制点动作

- Record real-world performances (e.g. person executing an activity)
- Extract pose as a function of time from the data collected

Strengths

- Can capture large amounts of real data quickly
- Realism can be high

Weaknesses

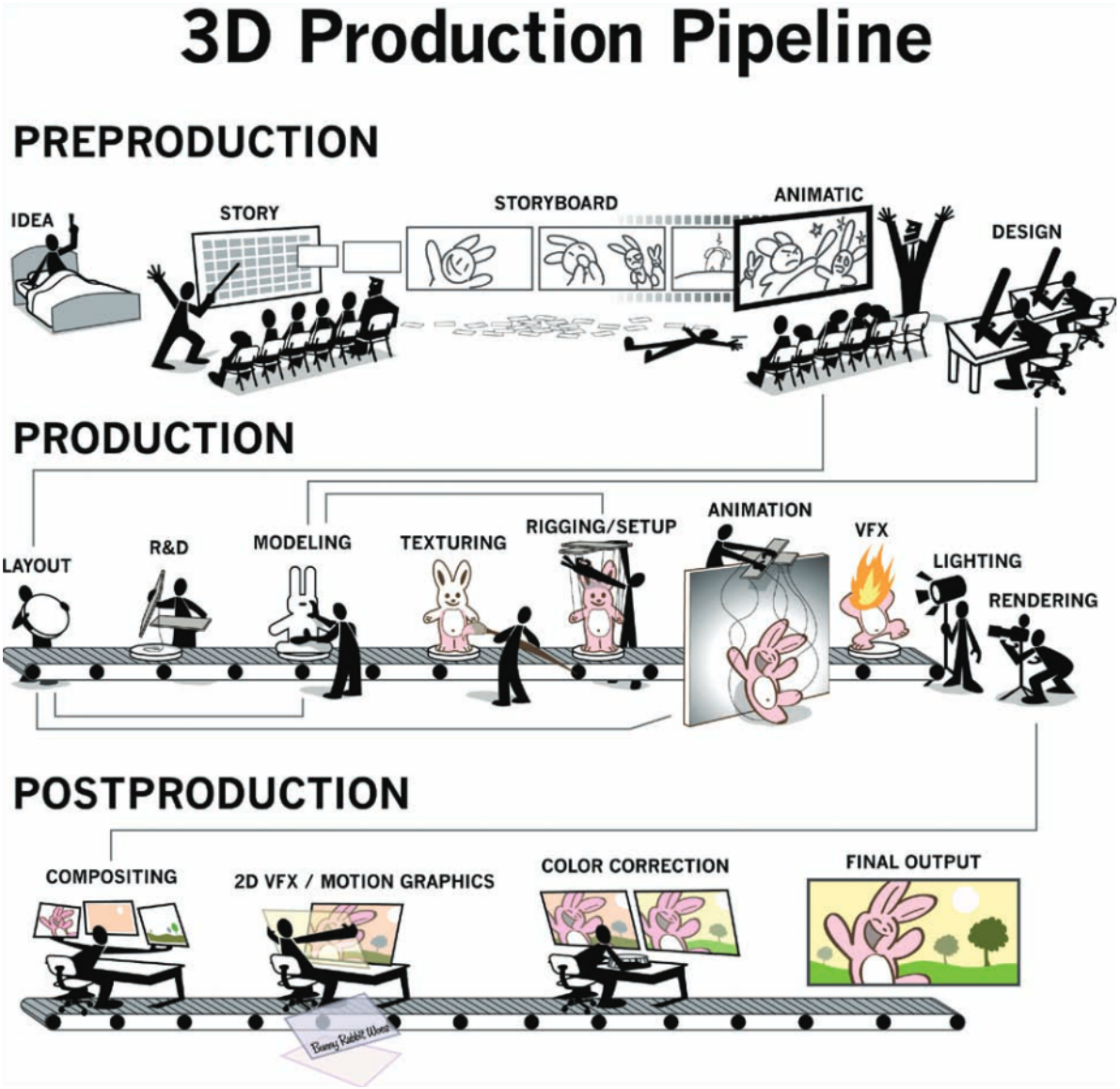
- Complex and costly set-ups
- Captured animation may not meet artistic needs, requiring alterations

Optical Motion Capture

- Markers on subject
- Positions by triangulation from multiple cameras
- 8+ cameras, 240 Hz, occlusions are difficult

Uncanny valley (恐怖谷效应)

The Production Pipeline

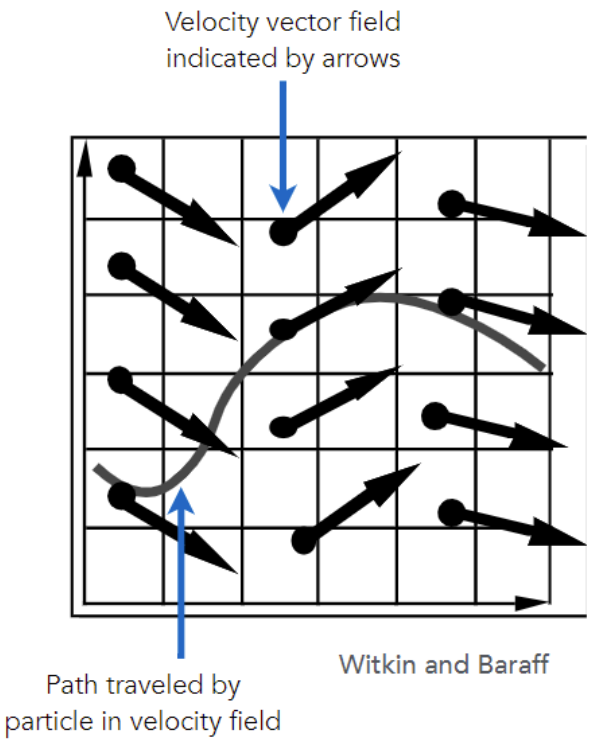


动画与模拟（求解常微分方程，刚体与流体）

Single particle simulation

To start, assume motion of particle determined by a velocity vector field that is a function of position and time:

$$\mathbf{v}(x,t)$$



Ordinary Differential Equation (ODE)

- “First-order” refers to the first derivative being taken.
- “Ordinary” means no “partial” derivatives, i.e. x is just a function of t

$$\frac{dx}{dt} = \dot{x} = v(x, t)$$

We can solve the ODE, subject to a given initial particle position x_0, by using forward numerical integration

Explicit Euler method

Euler’s Method (a.k.a. Forward Euler, Explicit Euler)

用上一个时间的位置、速度、加速度

- Simple iterative method
- Commonly used
- Very inaccurate
- Most often goes <u>unstable</u>

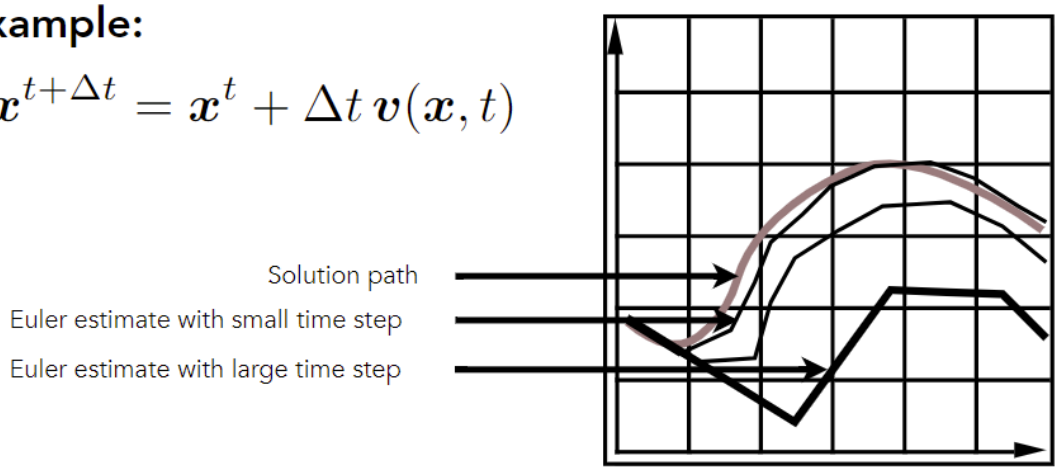
$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^t + \Delta t \dot{\boldsymbol{x}}^t$$
$$\dot{\boldsymbol{x}}^{t+\Delta t} = \dot{\boldsymbol{x}}^t + \Delta t \ddot{\boldsymbol{x}}^t$$

Euler’s Method - Errors (误差)

With numerical integration, errors accumulate. Euler integration is particularly bad

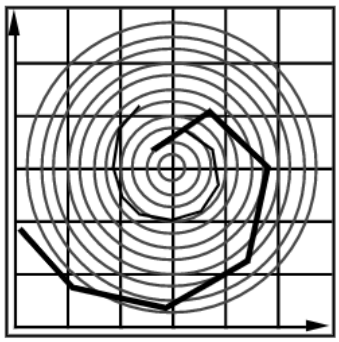
Example:

$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^{t} + \Delta t\, \boldsymbol{v}(\boldsymbol{x}, t)$$



Witkin and Baraff

Instability of the Euler Method (稳定性)



- 无法模拟圆周运动，肯定会偏离

Two key problems:

- Inaccuracies increase as time step Δt increases
- Instability is a common, serious problem that can cause simulation to diverge

Errors and Instability

Solving by numerical integration with finite differences leads to two problems:

Errors

- Errors at each time step accumulate.

  Accuracy <u>decreases</u> as simulation proceeds
- Accuracy may not be critical in graphics applications

Instability

- Errors can compound, causing the simulation to **diverge** even when the underlying system does not
- Lack of stability is a fundamental problem in simulation, and cannot be ignored

Instability and improvements

Some Methods to Combat Instability

Midpoint method / Modified Euler

- Average velocities at start and endpoint

Adaptive step size

- Compare one step and two half-steps, recursively, until error is acceptable

Implicit methods

- Use the velocity at the next time step (hard)

Position-based / Verlet integration

- Constrain positions and velocities of particles after time step

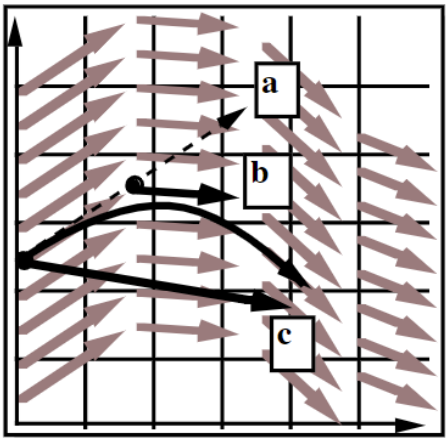
Midpoint Method

midpoint的速度更有代表性

Midpoint method

- Compute Euler step (a)
- Compute derivative at midpoint of Euler step (b)
- Update position using midpoint derivative (c)

$$x_{\text{mid}} = x(t) + \Delta t/2 \cdot v(x(t), t)$$
$$x(t + \Delta t) = x(t) + \Delta t \cdot v(x_{\text{mid}}, t)$$



Witkin and Baraff

Modified Euler

Modified Euler

- Average velocity at start and end of step
- Better results

$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^t + \frac{\Delta t}{2}\left(\dot{\boldsymbol{x}}^t + \dot{\boldsymbol{x}}^{t+\Delta t}\right)$$
$$\dot{\boldsymbol{x}}^{t+\Delta t} = \dot{\boldsymbol{x}}^t + \Delta t\ \ddot{\boldsymbol{x}}^t$$

$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^t + \Delta t\ \dot{\boldsymbol{x}}^t + \frac{(\Delta t)^2}{2}\ddot{\boldsymbol{x}}^t$$

- 引入二次误差项

Adaptive Step Size

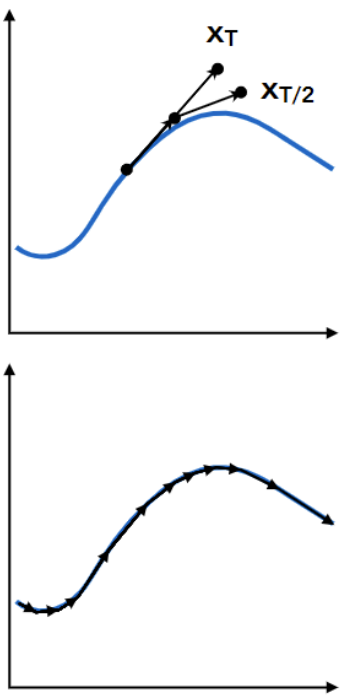
根据Δt, $\Delta t/2$分别得到x的误差来动态修改Δt

Adaptive step size

- Technique for choosing step size based on error estimate
- Very practical technique
- But may need very small steps!

Repeat until error is below threshold:

- Compute x_T an Euler step, size T
- Compute $x_{T/2}$ two Euler steps, size T/2
- Compute error $\| x_T - x_{T/2} \|$
- If (error > threshold) reduce step size and try again



Implicit Euler Method

Use derivatives in the future, for the current step

$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^t + \Delta t \dot{\boldsymbol{x}}^{t+\Delta t}$$
$$\dot{\boldsymbol{x}}^{t+\Delta t} = \dot{\boldsymbol{x}}^t + \Delta t \ddot{\boldsymbol{x}}^{t+\Delta t}$$

- Solve nonlinear problem for $\boldsymbol{x}^{t+\Delta t}$ and $\dot{\boldsymbol{x}}^{t+\Delta t}$
- Use root-finding algorithm, e.g. Newton's method
- Offers much better stability

Q: How to determine / quantize “stability”?

- We use the local truncation error (every step) and total accumulated error (overall)
- Absolute values do not matter, but the orders w.r.t. step
- Implicit Euler has order 1, which means that
  - Local truncation error: $O(h^2)$ and
  - Global truncation error: $O(h)$ (h is the step, i.e. Δt)
- Understanding of O(h)
  - If we halve h, we can expect the error to halve as well
  - 阶数越高越好

Runge-Kutta Families

A family of advanced methods for solving ODEs

- Especially good at dealing with non-linearity
- It’s order-four version is the most widely used, a.k.a. RK4

Initial condition:

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0.$$

RK4 solution:

$$\begin{aligned} y_{n+1} &= y_n + \frac{1}{6}h\left(k_1 + 2k_2 + 2k_3 + k_4\right), \\ t_{n+1} &= t_n + h \end{aligned}$$

where

$$\begin{aligned} k_1 &= f(t_n, y_n), \\ k_2 &= f\left(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}\right), \end{aligned}$$

$$\begin{aligned} k_3 &= f\left(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}\right), \\ k_4 &= f\left(t_n + h, y_n + hk_3\right). \end{aligned}$$

Position-Based / Verlet Integration

非物理的简化

Idea:

- After modified Euler forward-step, <u>constrain positions</u> of particles to prevent divergent, unstable behavior
- Use constrained positions to calculate velocity
- Both of these ideas will dissipate energy, stabilize

Pros / cons

- Fast and simple
- Not physically based, dissipates energy (error)

Rigid body simulation

Simple case

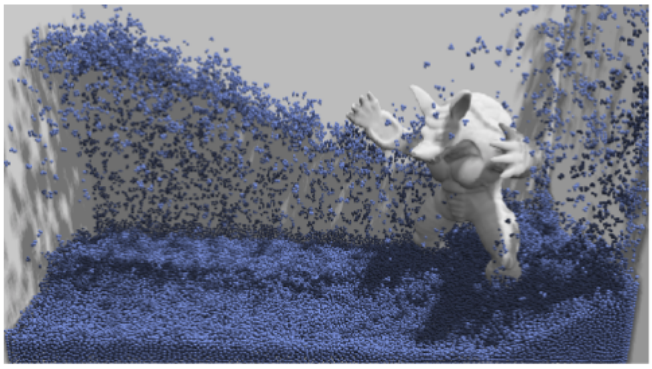
- Similar to simulating a particle
- Just consider a bit more properties

$$\frac{d}{dt}\begin{pmatrix} \mathrm{X} \\ \theta \\ \dot{\mathrm{X}} \\ \omega \end{pmatrix} = \begin{pmatrix} \dot{\mathrm{X}} \\ \omega \\ \mathrm{F}/M \\ \Gamma/I \end{pmatrix}$$

X : positions
θ : rotation angle
ω : angular velocity
F : forces
Γ : torque
I : momentum of inertia

Fluid simulation

A Simple Position-Based Method



Key idea

- Assuming water is composed of small rigid-body spheres
- Assuming the water cannot be compressed (i.e. const. density)

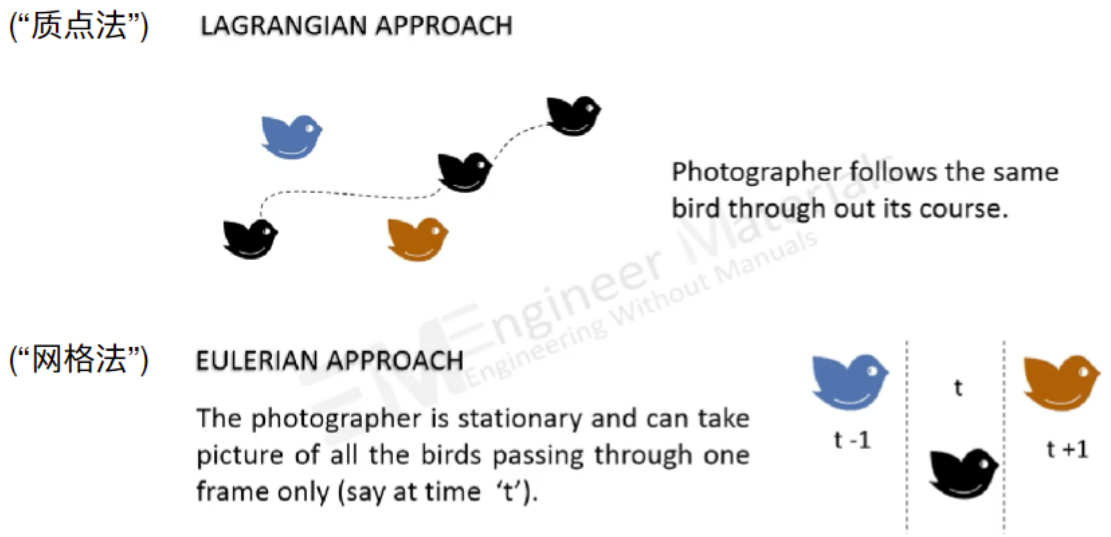
考虑整体密度，而不是单个点的位置

So, as long as the density changes somewhere, it should be "corrected" via changing the positions of particles

- You need to know the gradient of the density w.r.t. each particle's position
- Update? Just gradient descent! (回到正确的密度)

Eulerian vs. Lagrangian

Two different views to simulating large collections of matters



- 质点法考虑每个质点的运动
- 网格法把空间划分成网格，考虑网格的变化

Material Point Method (MPM)

Hybrid, combining Eulerian and Lagrangian views

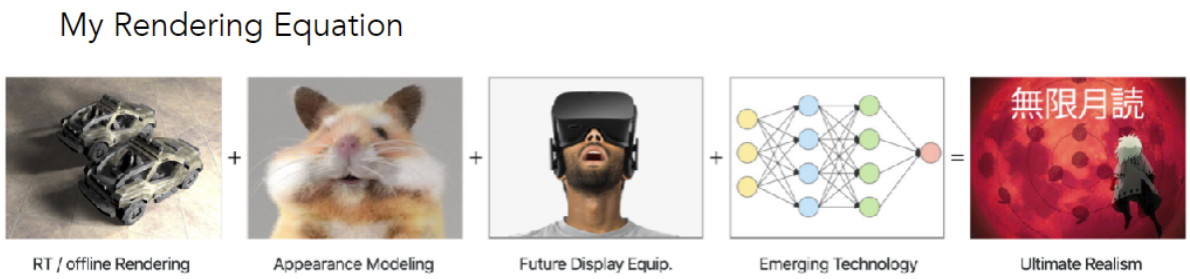
- Lagrangian: consider particles carrying material properties
- Eulerian: use a grid to do numerical integration
- Interaction
  - particles transfer properties to the grid
  - grid performs update, then interpolate back to particles

Advanced Topics

GAMES 201: 高级物理引擎实战

GAMES 202: 高质量实时渲染

Advanced Image Synthesis



> Join my group and become legendary!