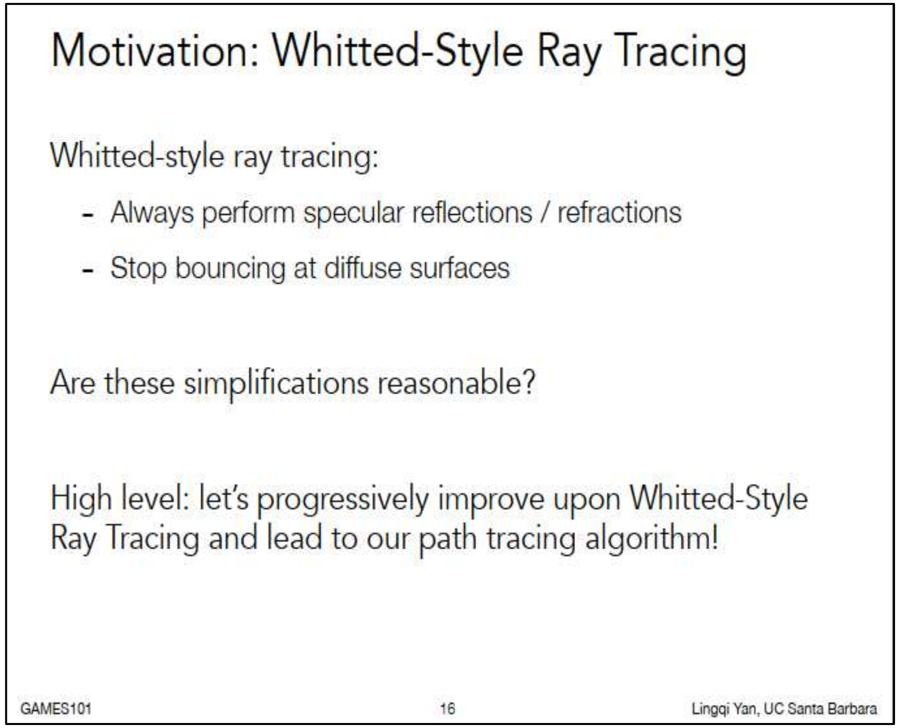
Recap

- **Whitted style RT是不正确的非物理的光线跟踪**
  - Glossy reflection: 反射光并非完全沿着镜面反射方向弹射
  - Diffuse reflection: 漫反射表面之间的光线弹射没有考虑



Recap(cont.)

- PBR(Physical Based Rendering)

- 从视点到每像素产生光线方程(射线),
- 射线和场景中物体进行求交测试，找到最近交点，
- 在最近交点处采用“渲染方程”模型，计算得到交点颜色
  - – Radiometry(辐射度量学)
    - Irradiance：单位面接接收的“所有方向”来的光强power
    - radiance:单位面积接收/发射的单位方向/单位立体角的光强power

Irradiance

Definition: The irradiance is the power per (perpendicular/projected) unit area incident on a surface point.

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

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

Incident Radiance

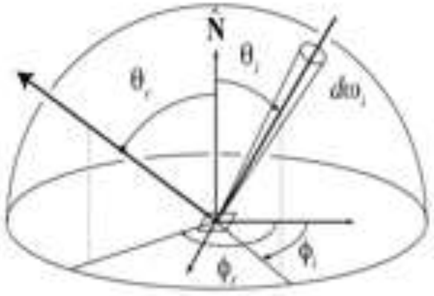
Incident radiance is the irradiance per unit solid angle arriving at the surface.

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

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

Recap(cont.)

- PBR(Physical Based Rendering)



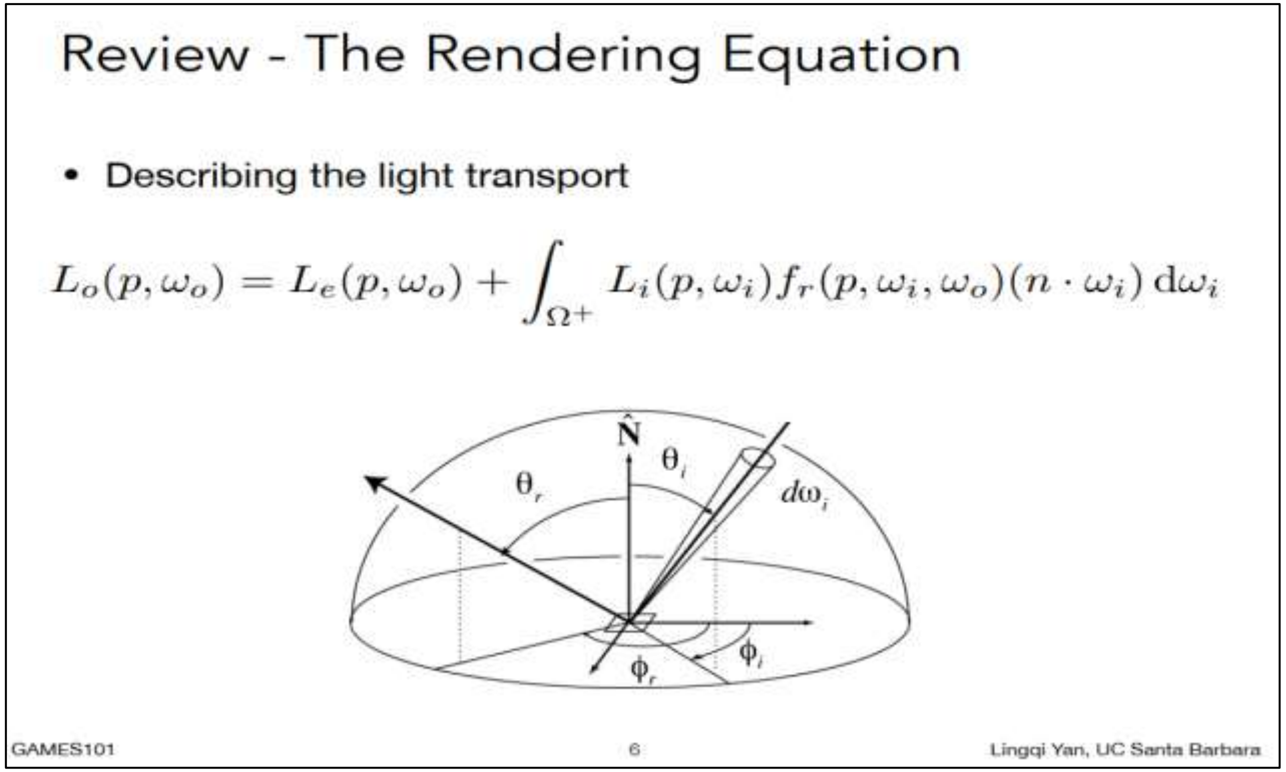
- **从视点到每像素产生光线方程（射线），**
- **射线和场景中物体进行求交测试，找到最近交点，**
- **在最近交点处采用“渲染方程”模型，计算得到交点颜色**
  - Radiometry(辐射度量学）
    - radiance：单位面积接收/发射的单位方向/立体角的光线强度power
  - BRDF(双向反射分布函数)
    - f(ωi → ωo)：物体表面从ωi方向入射的光，从ω_o方向反射出去的光强比值。描述材质的反射属性（反射率）
  - The Reflection Equation and The Rendering Equation（渲染方程）
    - **表面点p处,视线方向ω_o的出射光线强度$L_o(p, \omega_o)$ =**
      该p点处的自发射光在出射方向ω_o上的光线强度+
      该p点处正半球面内的所有入射方向ω_i 的入射光线，在出射方向ω_o的反射光线（镜面反射或漫反射）的强度之

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

注：cos(θ)=n•ωi 且 n, ω i是单位向量

Today's Theme

- **How to solve the rendering equation ?**
  - 如何求解求出L(p , w_o)？
    - 蒙特卡洛数值求解法

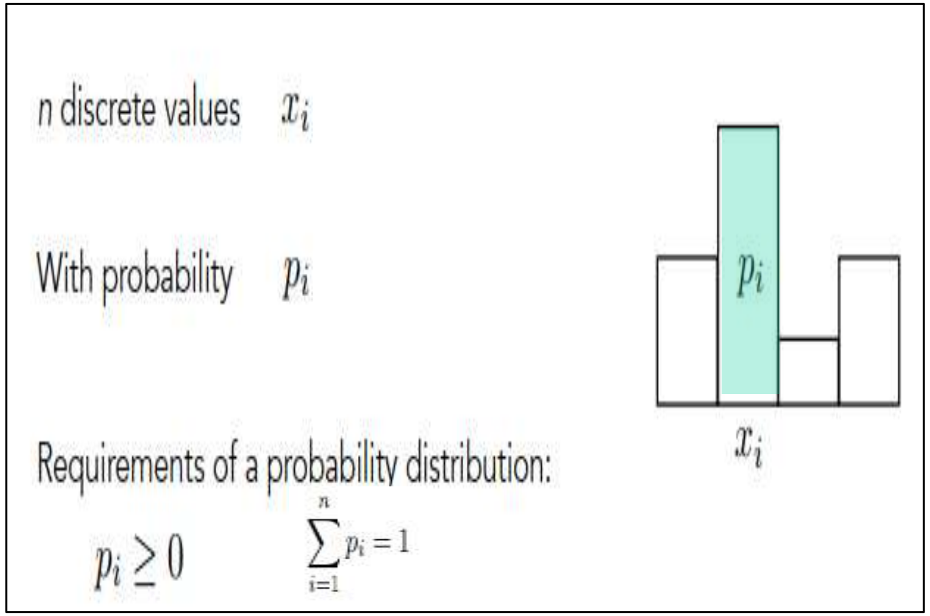


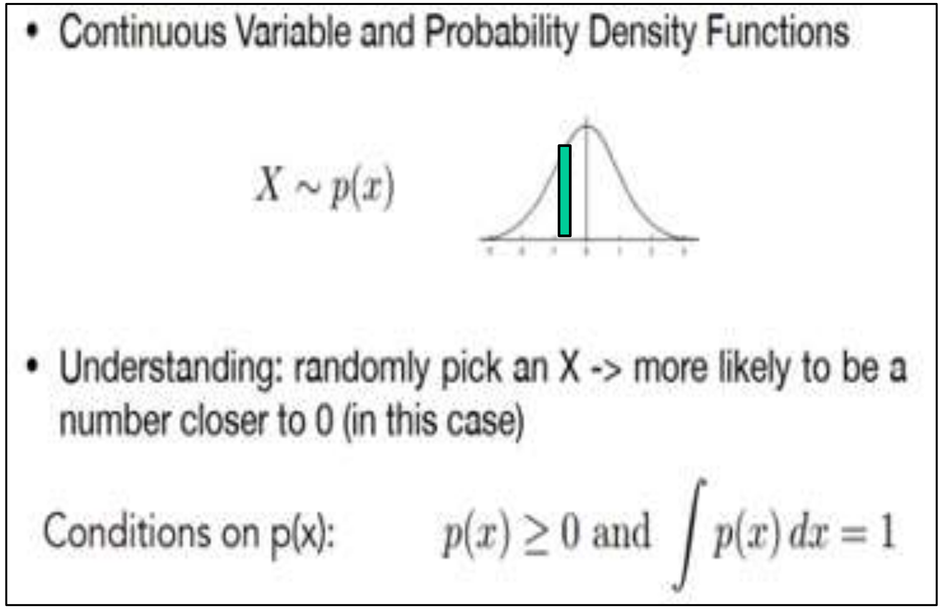
Outline

- PBR(Physical Based Rendering)
  - Solve the Rendering Equation
    - –Probability Basic Concepts（概率基础）
    - –Monte Carlo Solution（蒙特卡洛积分）
    - –Path Tracing Algorithm（路径跟踪算法）

Probability Basic Concepts

- ➢随机变量random variable : X
- ➢概率Probability: P
  - 离散时：变量值发生的频率（概率）直接给出（如图是每个小矩形的面积）Pi（0<Pi<1 且 Pi之和=1）
  - 连续时：用概率密度函数PDF(Probability Density Function) p(x)表示概率 P=p(x)dx （(0<p（x）<1 且 p(x)在X上积分=1)





Probability Basic Concepts

➢随机变量X的期望值 (expected value of X):E[X]

- 期望值反映了随机变量取值的平均水平。表示随机一次采样,最可能命中的X的值。
- 换句话说，期望值是随机试验在同样的机会下重复多次的结果计算出的**等同“期望”**的平均值。大数定律表明，随着重复次数接近无穷大，数值的算术平均值几乎肯定地收敛于期望值。

- 期望值在概率论和统计学中，是指在一个离散性随机变量试验中，“每次可能结果”乘以“每次可能结果的概率”的总和。
- 随机变量X的数学期望E[X]的计算公式如下(左：离散，右：连续)：

Expected value of X: $E[X] = \sum_{i=1}^{n} x_i p_i$

Expected value of X: $E[X] = \int x\, p(x)\, dx$

Probability Basic Concepts（cont.)

- 随机变量X的函数Y=f(X)的期望: E[Y]=?

1.离散型随机变量函数的期望

如果 X 是一个离散随机变量，其可能的取值为 x1,x2,...,xn，对应的概率为 P(X=xi)=pi，那么函数 Y=g(X) 的期望值定义为:

$$EY = \sum_{i=1}^{n} g(x_i)p_i$$

Function of a Random Variable

A function Y of a random variable X is also a random variable:

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

Expected value of a function of a random variable:

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

Outline

- PBR(Physical Based Rendering)
  - Solve the Rendering Equation
    - –Probability Basic Concepts（概率基础概念）
    - –Monte Carlo Solution（蒙特卡洛积分）
    - –Path Tracing Algorithm（路径跟踪算法）

Monte Carlo Integration

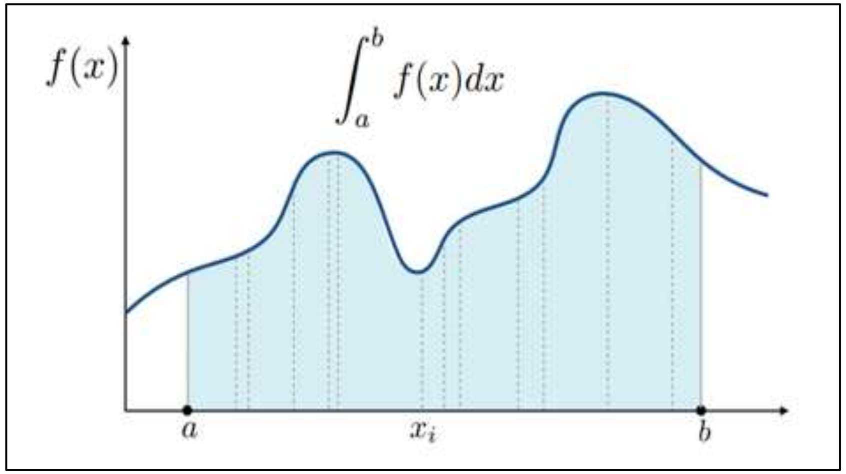
- Monte-Carlo Solution（蒙特卡洛法/蒙特卡洛估计）
  - 冯诺依曼提出的以赌城名命名的数值求解法。

- 思想：通过“随机抽样”来估计一个量的值（数值解）；而不是通过确定性的数学公式来计算（解析解）

- 用例1：求解圆周率pi
  - **使用蒙特卡罗法求圆周率的基本原理**：是通过在正方形内随机生成点，并统计落在内切圆内的点的比例，进而计算出圆周率的近似值
- 用例2：求定积分的值，又称为“蒙特卡洛积分法”
  - **蒙特卡罗法求定积分值，主要是通过在积分区域内随机采样，计算样本点的函数值并求平均来估算积分值。**

Monte Carlo Integration

- **Monte Carlo Integration蒙特卡洛积分**
  - **Why Monte Carlo Integration :**
    - solve an integral can be too difficult to solve analytically.(求积分的解析解太难)
  - define **the Monte Carlo estimator** for **the definite integral of given function**
  - 将“函数的定积分”转换为“蒙特卡洛估计/蒙特卡洛积分”



Monte Carlo Integration

$$\boxed{\int f(x)\,\mathrm{d}x = \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)} \qquad X_i \sim p(x)}$$

Some notes:

- The more samples, the less variance.
- Sample on x, integrate on x.

Monte Carlo Integration(cont.)

➢Monte Carlo Integration蒙特卡洛积分（cont.)

➢Example：

若随机变量X符合uniform均匀分布p(x)=1/(b-a), 那么Y=f(X)在[a,b]上的定积分用蒙特卡洛估计如下：

Monte Carlo Integration

$$\boxed{\int f(x)\,\mathrm{d}x = \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)} \qquad X_i \sim p(x)}$$

Some notes:

- The more samples, the less variance.
- Sample on x, integrate on x.

Example: Uniform Monte Carlo Estimator

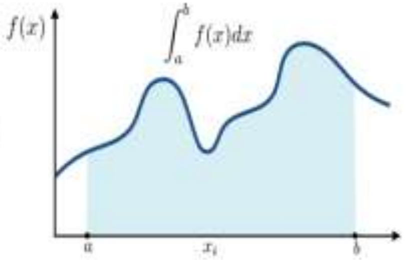
Uniform random variable

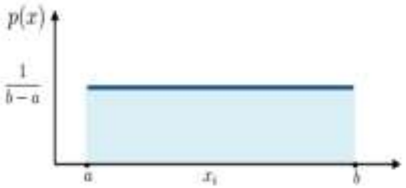
$$X_i \sim p(x) = C \text{ (constant)}$$

$$\boxed{\int_a^b p(x)\,dx = 1}$$

$$\Longrightarrow \int_a^b C\,dx = 1$$

$$\Longrightarrow C = \frac{1}{b-a}$$





Example: Uniform Monte Carlo Estimator

Let us define the Monte Carlo estimator for the definite integral of given function $f(x)$

Definite integral $\int_a^b f(x)dx$

Uniform random variable $X_i \sim p(x) = \frac{1}{b-a}$

Basic Monte Carlo estimator $F_N = \frac{b-a}{N}\sum_{i=1}^{N} f(X_i)$

Monte Carlo Integration(cont.)

➢Monte Carlo Integration蒙特卡洛积分(cont.)

- **What&How Monte Carlo Integration：**
  - **estimate the integral of a function by averaging random samples of the function's value. (平均 "一个函数值的多个随机样本的值"， 估计 "这个函数的积分值")**

➢ 首先"对被积函数的变量区间进行随机均匀抽样"，然后"对抽样点的函数值求平均"

Monte Carlo Integration

$$\boxed{\int f(x)\,\mathrm{d}x = \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)} \qquad X_i \sim p(x)}$$

Some notes:

- The more samples, the less variance.
- Sample on x, integrate on x.

蒙特卡洛积分公式：

$$\int f(x)dx \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{pdf(X_i)}$$

该估计的期望：

$$E[I_N] = E\left[\frac{1}{N}\sum_{i=1}^{N}\frac{g(X_i)}{pdf(X_i)}\right]$$
$$= \frac{1}{N}\sum_{i=1}^{N}\int\frac{g(X_i)}{pdf(X_i)}pdf(X_i)dx$$
$$= \frac{1}{N}N\int g(x)dx = \int g(x)dx = I$$

1.离散型随机变量函数的期望

如果 X 是一个离散随机变量，其可能的取值为 x1,x2,...,xn，对应的概率为 P(X=xi)=pi，那么函数 Y=g(X) 的期望值定义为：

$$EY = \sum_{i=1}^{n} g(x_i)p_i$$

该估计的方差：

$$\sigma^2 = \frac{1}{N}\int\left(\frac{g(x)}{f(x)} - I\right)^2 pdf(x)dx$$

由上式可知，方差随着N的增加线性降低，由于估计的误差正比于标准差 σ，所以蒙特卡洛估计误差收敛的速度为 $O(\sqrt{N})$，即4倍的采样减少一半误差。

Monte Carlo Integration(cont.)

➢Monte Carlo Integration蒙特卡洛积分(cont.)

- ➢X的概率分布可以是任意分布(不一定是uniform均匀分布)，只要知道PDF即可
- ➢采样和积分都是在随机变量X上的(Sample on x , integrate on x)
- ➢采样数N越大，则估计的近似解的统计误差越小(The more samples , the less variance)
- ➢**所得近似解的统计误差只与采样数N有关，N越大则方差越小**
- ➢**所得近似解的统计误差与积分的维数无关。因此当积分维度较高时，蒙特卡罗方法相对于其他数值解法更优。**

Monte Carlo Integration

$$\int f(x)\,\mathrm{d}x = \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)} \qquad X_i \sim p(x)$$

Some notes:

- The more samples, the less variance.
- Sample on x, integrate on x.

蒙特卡洛积分公式：

$$\int f(x)dx \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{pdf(X_i)}$$

该估计的期望：

$$E[I_N] = E\left[\frac{1}{N}\sum_{i=1}^{N}\frac{g(X_i)}{pdf(X_i)}\right]$$
$$= \frac{1}{N}\sum_{i=1}^{N}\int\frac{g(X_i)}{pdf(X_i)}pdf(X_i)dx$$
$$= \frac{1}{N}N\int g(x)dx = \int g(x)dx = I$$

该估计的方差：

$$\sigma^2 = \frac{1}{N}\int\left(\frac{g(x)}{f(x)} - I\right)^2 pdf(x)dx$$

由上式可知，方差随着N的增加线性降低，由于估计的误差正比于标准差 σ ，所以蒙特卡洛估计误差收敛的速度为 $O(\sqrt{N})$ ，即4倍的采样减少一半误差。

Outline

- PBR(Physical Based Rendering)
  - Solve the Rendering Equation
    - –Probability Basic Concepts（概率基础概念）
    - –Monte Carlo Solution（蒙特卡洛积分）
    - –Path Tracing Algorithm（路径跟踪算法）

Path Tracing

- **Path Tracing（路径跟踪）**
  - **Dr. Eric Veach** 于1998年提出了利用蒙特卡洛法求解渲染方程中的积分方程的方法，并且引申出“路径追踪算法Path Tracing”
  - In 1997, he graduated with a PhD from Stanford University. His thesis is titled “Robust Monte Carlo Methods for Light Transport Simulation”， a highly cited paper in Computer Graphics

Path Tracing（cont.)

- **用渲染方程计算~全局光照~**
  - ➢首先，对半球面的入射光线进行积分算出反射光：采用蒙特卡洛积分法
  - ➢其次，考虑递归

Whitted-Style Ray Tracing is Wrong

But the rendering equation is correct

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

But it involves

- Solving an integral over the hemisphere, and
- Recursive execution

How do you solve an integral numerically?

GAMES101 19 Lingqi Yan, UC Santa Barbara

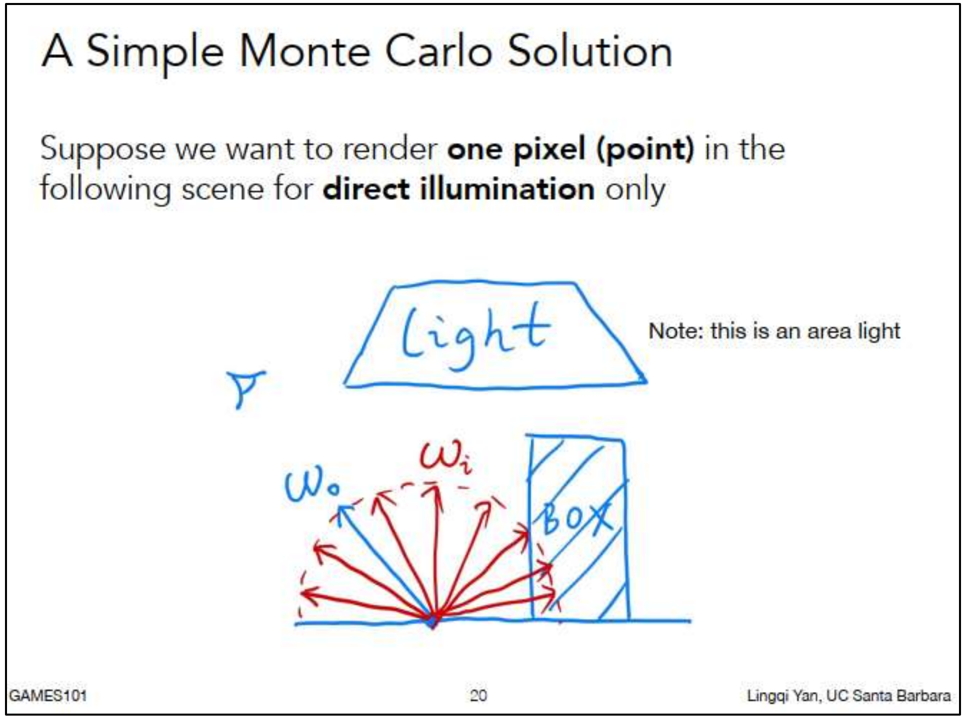
Path traced: direct illumination

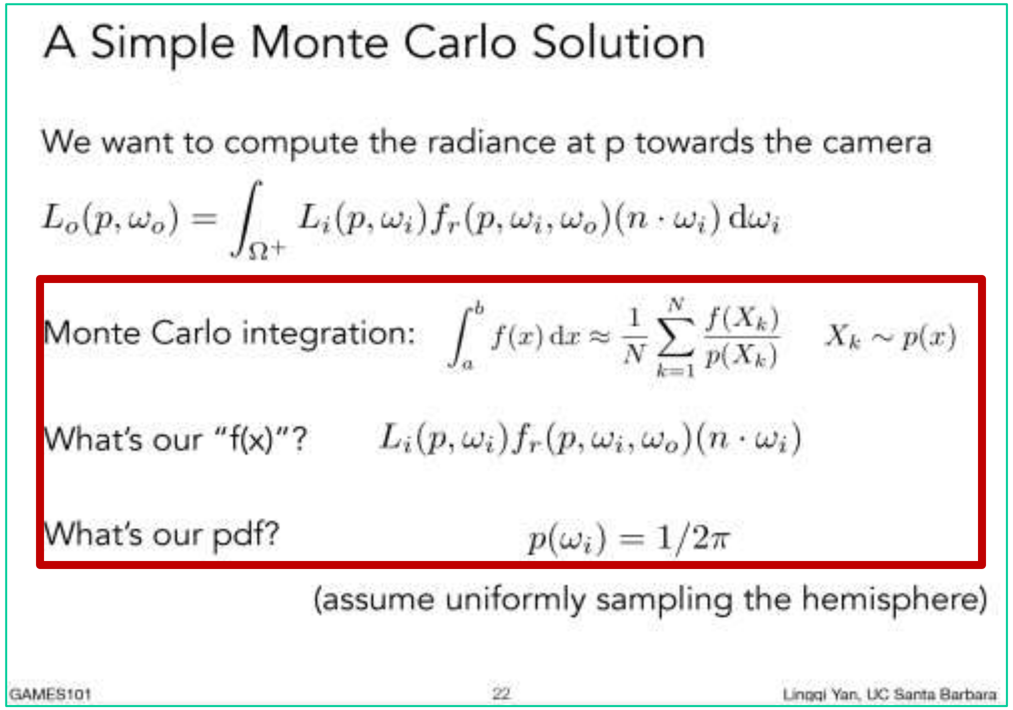
Path traced: global illumination

The Cornell box

Path Tracing（cont.)

- **采用蒙特卡洛积分解渲染方程**
  - **Simple Monte Carlo Solution简单蒙特卡洛积分：直接光照（无递归）**
    - **只考虑光源来的直接光，没有考虑其它物体来的反射光**





Path Tracing（cont.)

- 采用蒙特卡洛积分解渲染方程(cont.)
  - Simple Monte Carlo Solution简单蒙特卡洛求解:直接光照(cont.)
    - 只考虑光源来的入射光，没有考虑其它物体来的入射光（无递归）
    - shade(p,wo)计算交点颜色：从P点出发，在半球方向上随机采样N次，若方向wi交到光源，就代入蒙特卡洛公式计算。 最后Lo（p,wo) 就是这N次随机采样的函数值的平均值。

A Simple Monte Carlo Solution

So, in general

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

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

(note: abuse notation a little bit for i)

What does it mean?

A correct shading algorithm for direct illumination!

A Simple Monte Carlo Solution

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

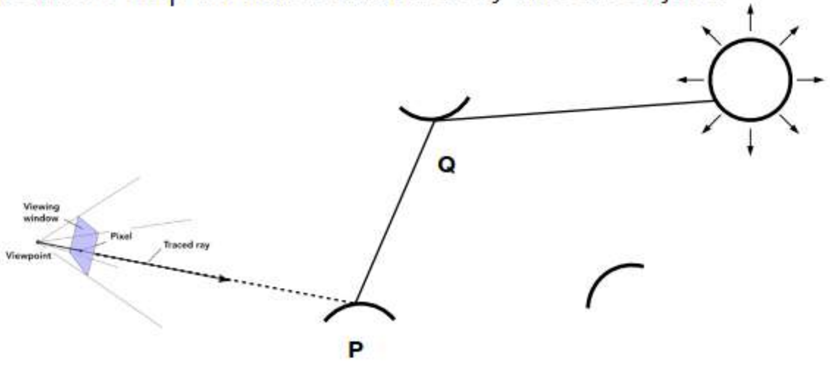
```
shade(p, wo)
    Randomly choose N directions wi~pdf
    Lo = 0.0
    For each wi
        Trace a ray r(p, wi)
        If ray r hit the light
            Lo += (1 / N) * L_i * f_r * cosine / pdf(wi)
    Return Lo
```

Path Tracing（cont.)

- 采用蒙特卡洛积分解渲染方程(cont.)
  - Recursive Monte Carlo Solution ：
    - 全局光照（直接光照+间接光照）: 需要递归计算~光线二次弹射/继续跟踪光线~
      - 需要计算Q弹射到P点的间接光，即需要从P点出发继续跟踪新光线

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!

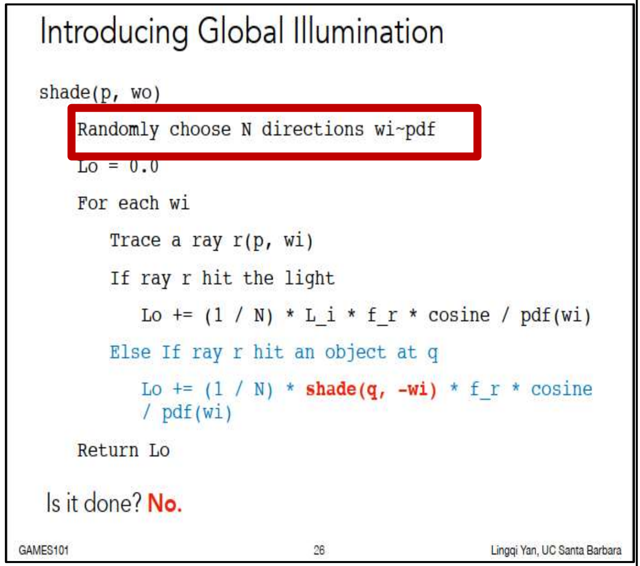
Introducing Global Illumination

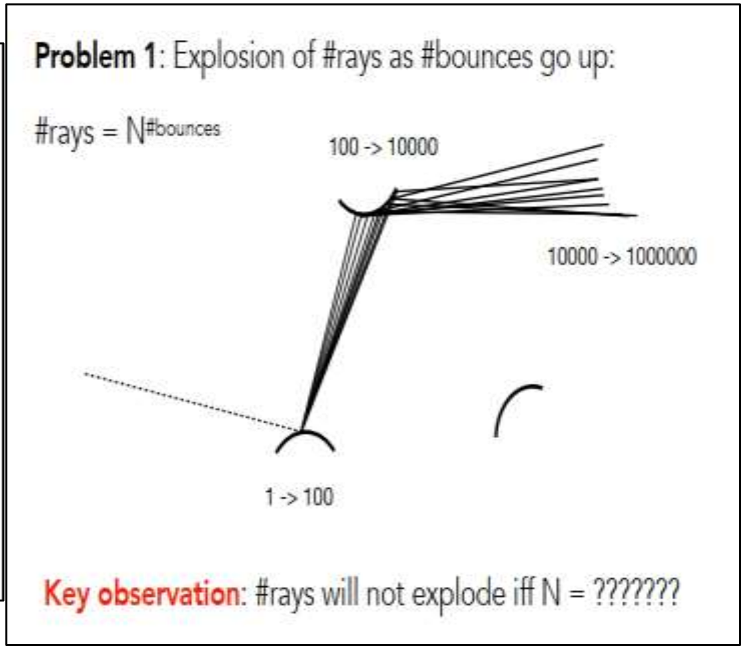
```
shade(p, wo)
    Randomly choose N directions wi~pdf
    Lo = 0.0
    For each wi
        Trace a ray r(p, wi)
        If ray r hit the light
            Lo += (1 / N) * L_i * f_r * cosine / pdf(wi)
        Else If ray r hit an object at q
            Lo += (1 / N) * shade(q, -wi) * f_r * cosine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

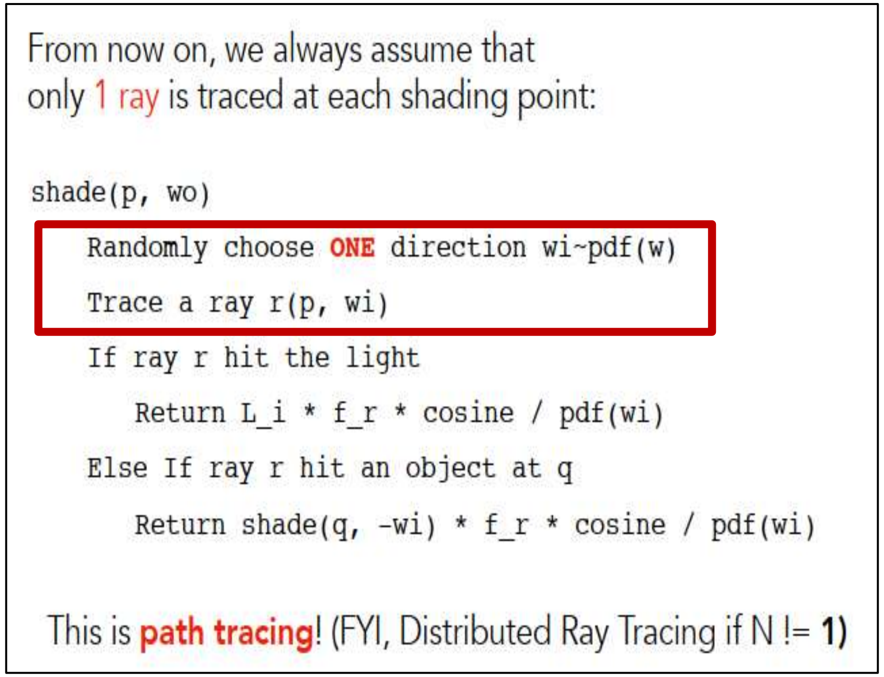
Is it done? **No.**

Path Tracing（cont.)

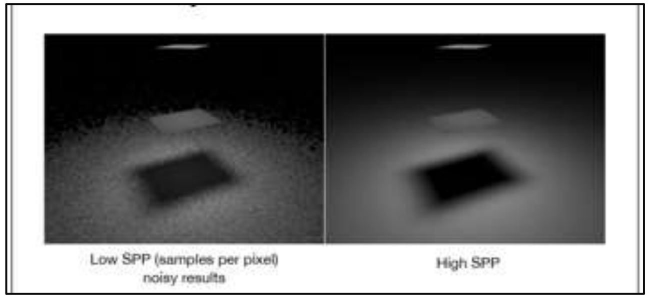
- ➢采用蒙特卡洛积分解渲染方程(cont.)
  - ➢Recursive Monte Carlo Solution (cont.)
    - ✓全局光照(直接光照+间接光照)：计算中引入递归~二次弹射，继续跟踪光线
    - ➢Problem1：光线数量会指数增长(“爆炸explosion”)
      - ➢解决方法：路径跟踪（光线采样数N=1）（若N！=1则是分布式光线跟踪Distributed RT）







Path Tracing(cont.)

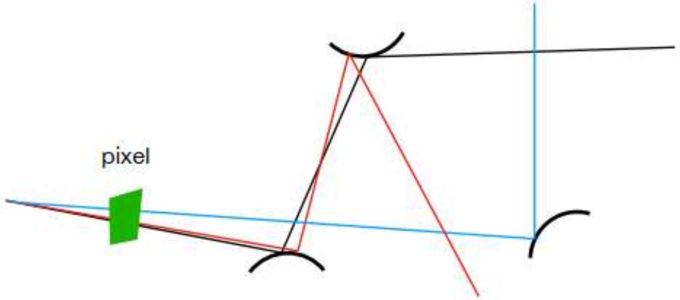


- 采用蒙特卡洛积分解渲染方程(cont.)
  - Recursive Monte Carlo Solution (cont.)
    - ✓ 全局光照(直接光照+间接光照)：计算中引入递归~二次弹射，继续跟踪光线~
    - Problem：光线数量会指数增长(“爆炸explosion”)
      - 解决方法：路径跟踪（N=1，即只跟踪一条从视点到像素,到场景交点，再到光源的路径path）
      - **路径跟踪问题1~”noisy 噪音”，解决办法：每像素均匀跟踪多条“路径”后着色，再平均**

Ray Generation

But this will be noisy!

No problem, just trace more **paths** through each pixel and average their radiance!



GAMES101 29 Lingqi Yan, UC Santa Barbara

Ray Generation

Very similar to ray casting in ray tracing

```
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)
        If ray r hit the scene at p
            pixel_radiance += 1 / N * shade(p, sample_to_cam)
    Return pixel_radiance
```

GAMES101 30 Lingqi Yan, UC Santa Barbara

Path Tracing（cont.)

➢采用蒙特卡洛积分解渲染方程(cont.)

➢Recursive Monte Carlo Solution (cont.)

✓全局光照（直接光照+间接光照）: 计算中引入递归~二次弹射，继续跟踪光线~

➢ Problem：光线数量会“爆炸explosion”，

➢ 解决方法：路径跟踪（N=1，只跟踪一条从视点到像素再到光源的路径path）

➢ **路径跟踪问题1~”noisy 噪音”，解决办法：每像素跟踪多条“路径”后着色，再平均**

➢ **路径跟踪问题2~“递归不会停，真实情况是弹射无数次”，解决办法？ 限制递归次数会损失能量，不好！**

Path Tracing

Now are we good? Any other problems in shade()?

```
shade(p, wo)
    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)
```

Problem 2: The recursive algorithm will never stop!

GAMES101 31 Lingqi Yan, UC Santa Barbara

Path Tracing

Dilemma: the light does not stop bouncing indeed!

Cutting #bounces == cutting energy!

3 bounces

GAMES101 32 Lingqi Yan, UC Santa Barbara

Path Tracing

Dilemma: the light does not stop bouncing indeed!

Cutting #bounces == cutting energy!

17 bounces

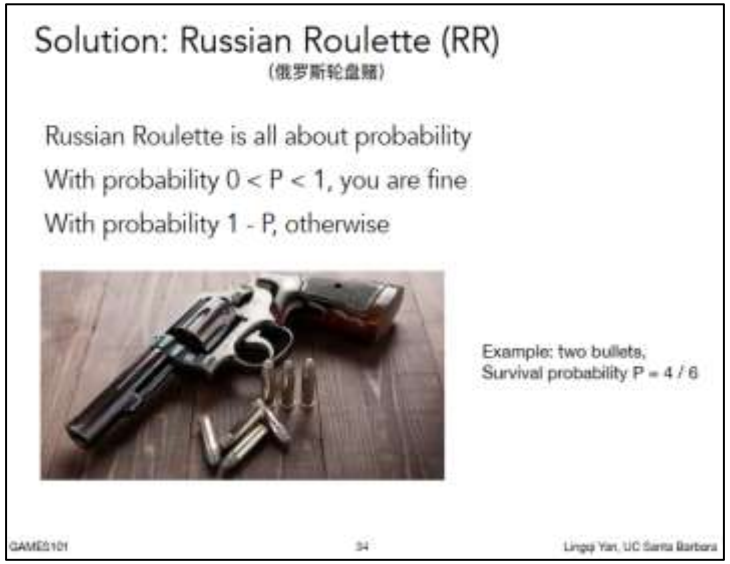
GAMES101 33 Lingqi Yan, UC Santa Barbara

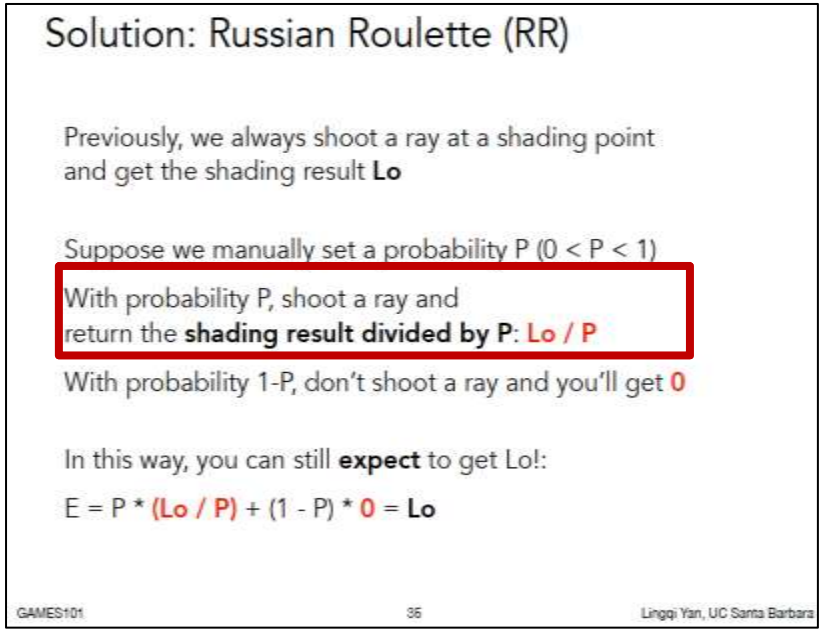
Path Tracing（cont.)

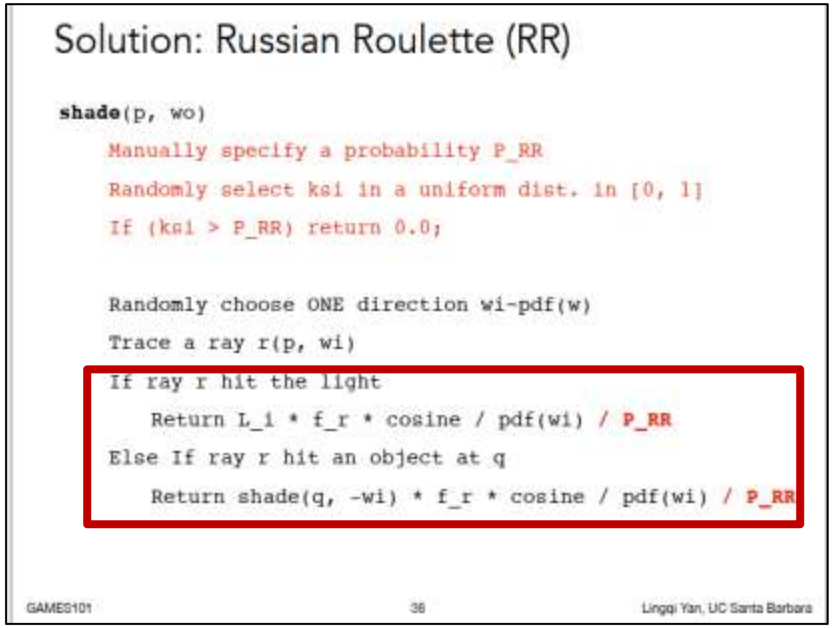
➢采用蒙特卡洛积分解渲染方程(cont.)

➢Recursive Monte Carlo Solution (cont.)

- ✓全局光照（直接光照+间接光照）: 计算中引入递归~二次弹射，继续跟踪光线~
- ➢ Problem : 光线数量会“爆炸explosion”，
  - ➢ 解决方法 : **路径跟踪**（N=1，只跟踪一条从视点到像素再到光源的路径path）
    - ➢ **路径跟踪问题1~”noisy 噪音”，解决办法 : 每像素跟踪多条“路径”后着色，再平均**
    - ➢ **路径跟踪问题2~“递归不会停，真实情况就是弹射无数次”.解决办法 : 俄罗斯轮盘赌， 即以一定的生存概率P_RR（自定义）跟踪光线. （如果随机数ksi> P_RR则停止）**







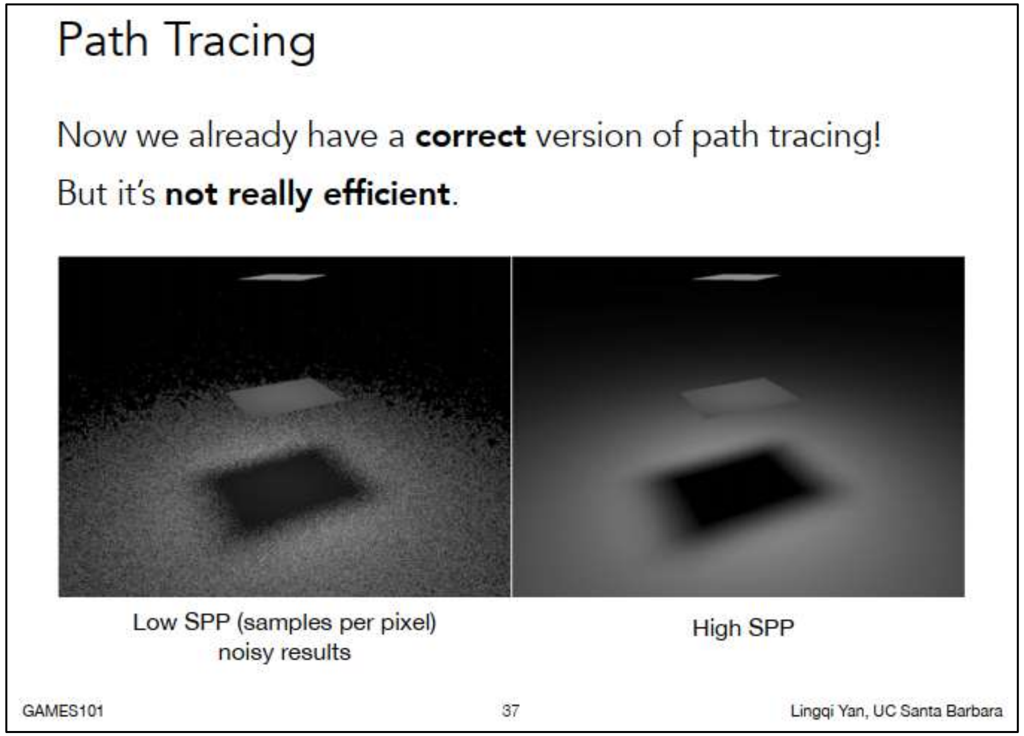
Path Tracing（cont.)

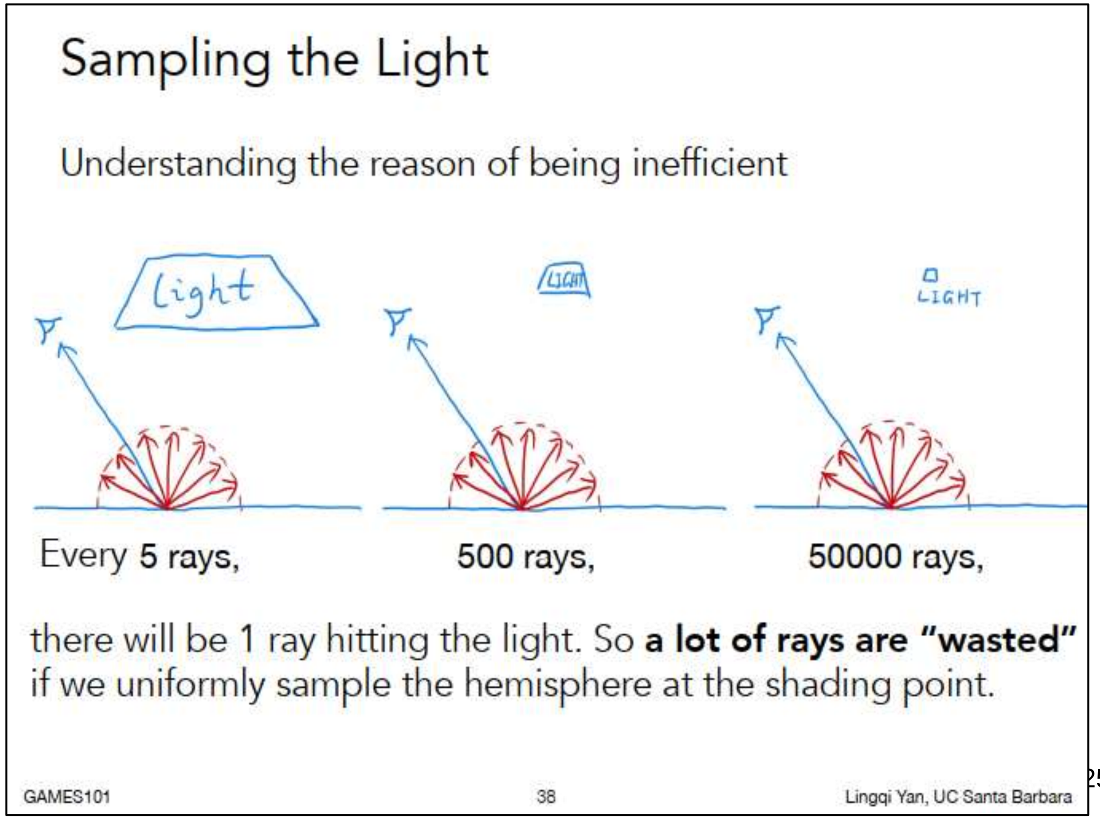
➢Path Tracing is “not really efficient”

- ➢Low SPP（samples per pixel) -> quick but noisy
- ➢High SPP(samples per pixel) -> clean but slow

希望low SPP下的效果也好，分析原因：when light area is small ,**A lot of rays are “wasted”**

- ➢ **前面方法中，从着色点向各个方向默认是均匀采样pdf(wi)=1/2pi（打出光线），能否打中光源和光源大小有关，**

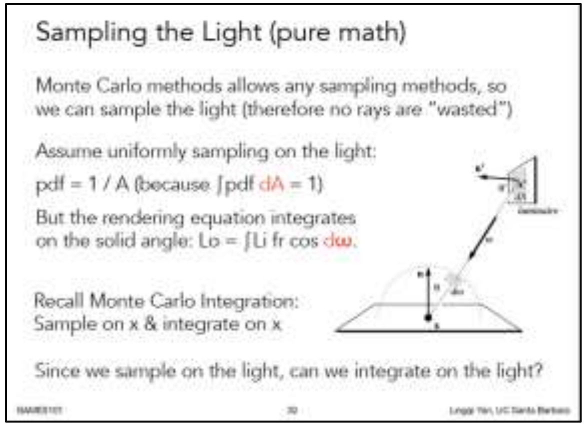


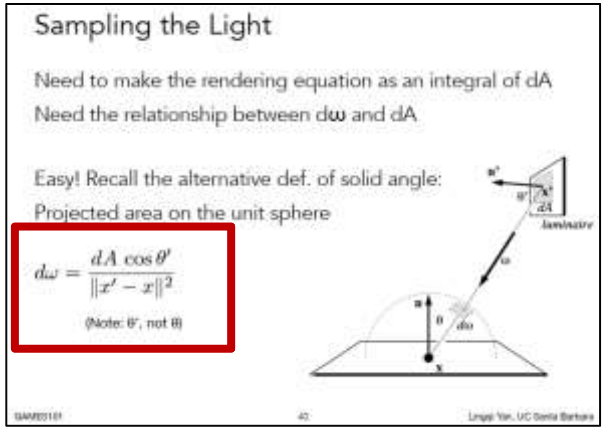


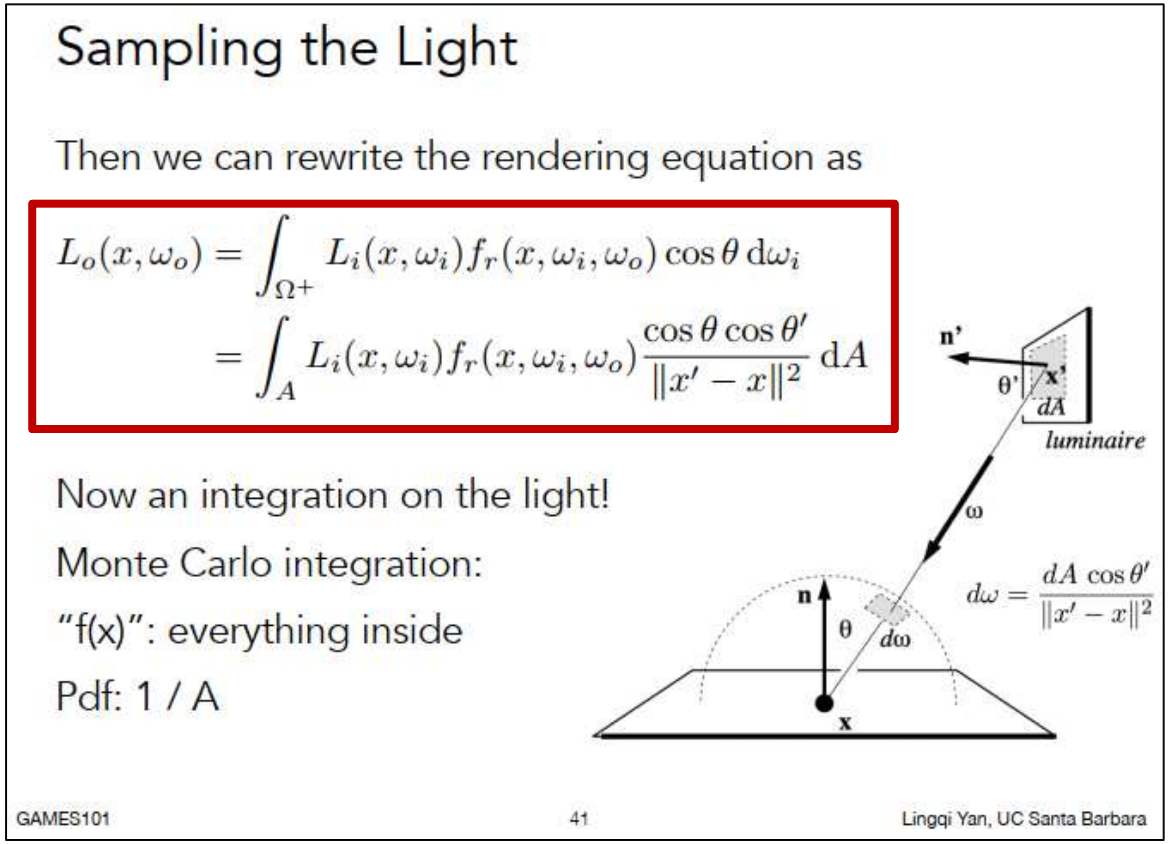
Path Tracing（cont.)

➢Path Tracing is “not really efficient”（cont.)

- ➢希望low SPP下的效果也好，分析原因：when light area is small ,A lot of rays are “wasted”
  - ➢ 解决办法：sampling on the light（改写渲染方程，对立体角采样dw转换为对光源采样dA）



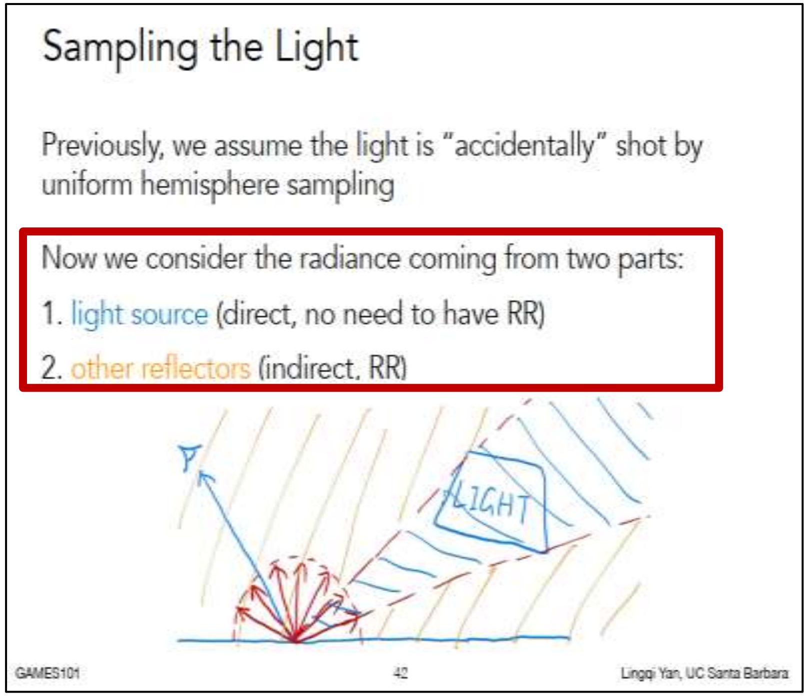


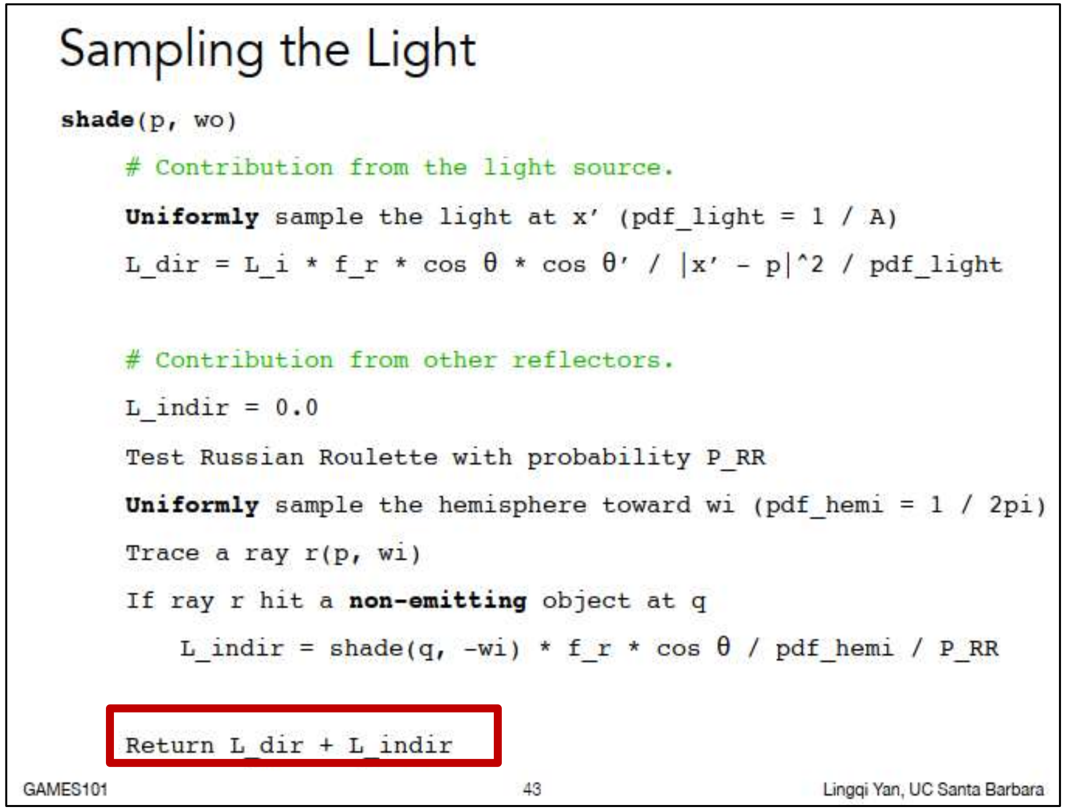


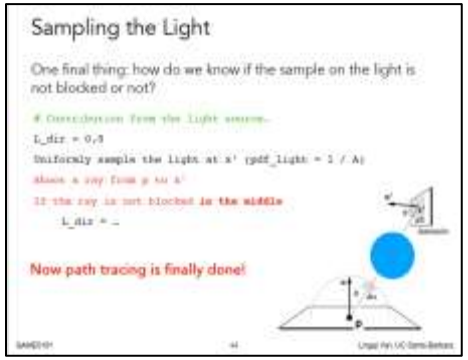
Path Tracing(cont.)

➢Path Tracing is "not really efficient"(cont.)

- ➢希望low SPP下的效果也好，分析原因：when light area is small ,A lot of rays are "wasted"
  - ➢ 解决办法：sampling on the light: 改写渲染方程，对立体角采样dw转换为对光源采样dA
  - ➢ **算法修改：对光源的采样分为两部分：光源方向的采样（直接光照）+其它方向的采样（间接光照）**







Path Tracing（cont.)

- Path Tracing Some Side Notes:
  - 点光源不好作，一般都采用面积光源
  - Path Tracing is difficult , but correct, worth learning

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
- Is it still “Introductory”?
  - Not really, but it’s “modern" :)
  - And so learning it will be rewarding also because …

Is Path Tracing Correct?

Yes, almost 100% correct, a.k.a. PHOTO-REALISTIC

Photo Path traced: global illumination

The Cornell box — http://www.graphics.cornell.edu/online/box/compare.html

Path Tracing (cont.)

- Ray tracing: Previous vs. Modern Ray Tracing

Ray tracing: Previous vs. Modern Concepts

- Previous
  - Ray tracing == Whitted-style ray tracing
- Modern (my own definition)
  - **The general solution of light transport**, including
  - (Unidirectional & bidirectional) path tracing
  - Photon mapping
  - Metropolis light transport
  - VCM / UPBP…

Path Tracing(cont.)

- Things haven’t covered/won’t cover
  - 如何在半球上进行均匀采样？(如何进行采样)
  - 选择什么样的pdf(wi)是最好的？（重要性采样Importance sampling ）
  - ……

Things we haven’t covered / won’t cover

- Uniformly sampling the hemisphere
  - How? And in general, how to sample any function? (sampling)
- Monte Carlo integration allows arbitrary pdfs
  - What's the best choice? (importance sampling)
- Do random numbers matter?
  - Yes! (low discrepancy sequences)

GAMES101 48 Lingqi Yan, UC Santa Barbara

Things we haven’t covered / won’t cover

- I can sample the hemisphere and the light
  - Can I combine them? Yes! (multiple imp. 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.

GAMES101 49 Lingqi Yan, UC Santa Barbara

Path Tracing（cont.)

- 知乎【WebGL与光线追踪】https://zhuanlan.zhihu.com/p/430999130
- Three JS（基于webGL) 实现 Path Tracing
  - 例程：**https://erichlof.github.io/THREE.js-PathTracing-Renderer/**

- Cornell Box Demo This demo renders the famous old Cornell Box, but at 30-60 FPS - even on mobile!

For comparison, here is a real photograph of the original Cornell Box vs. a rendering with the three.js PathTracer:

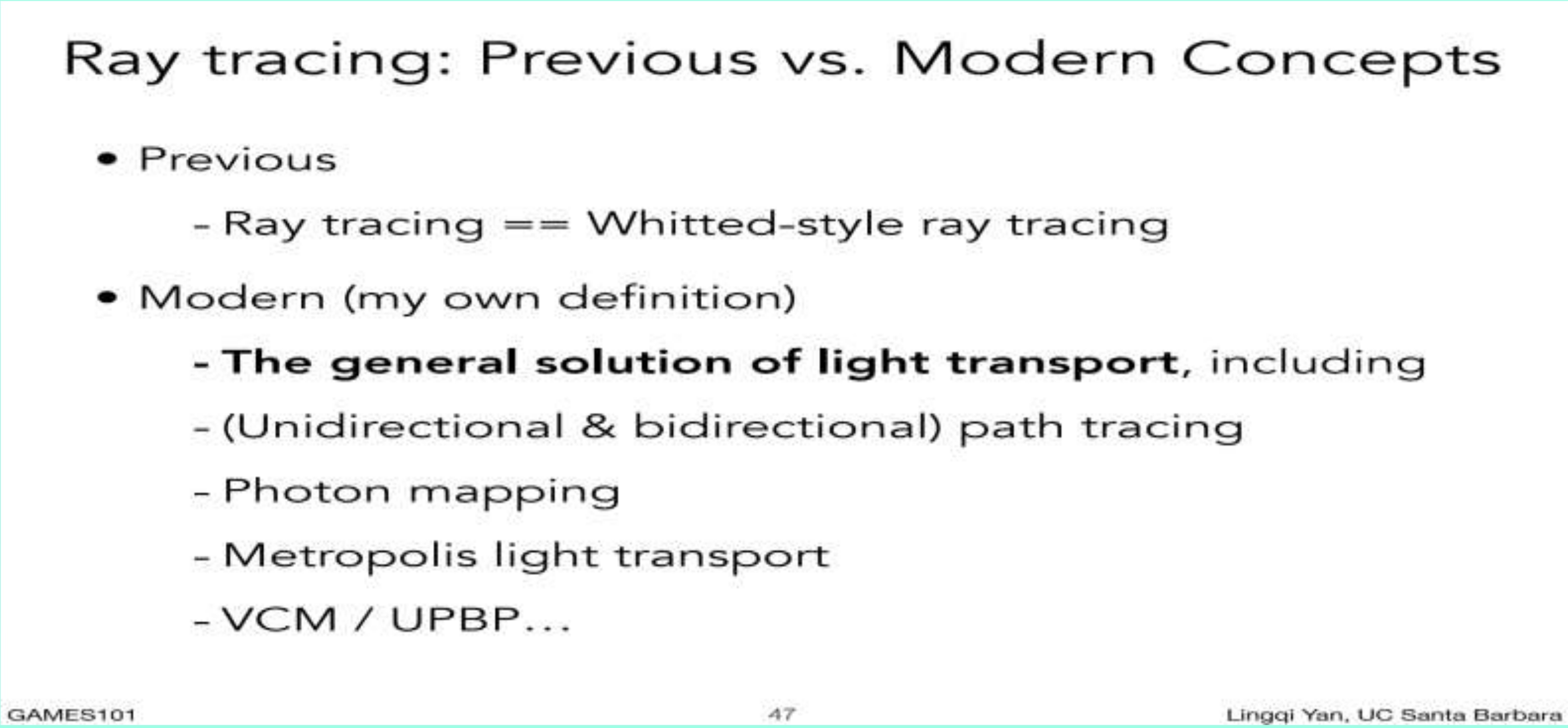
Path Tracing(cont.)

- webGPU实现算法path tracing
  - http://maierfelix.github.io/2020-01-13-webgpu-ray-tracing/
  - 在WebGPU Node开源项目中,电脑需要使用nvidia的RTX显卡

Summary

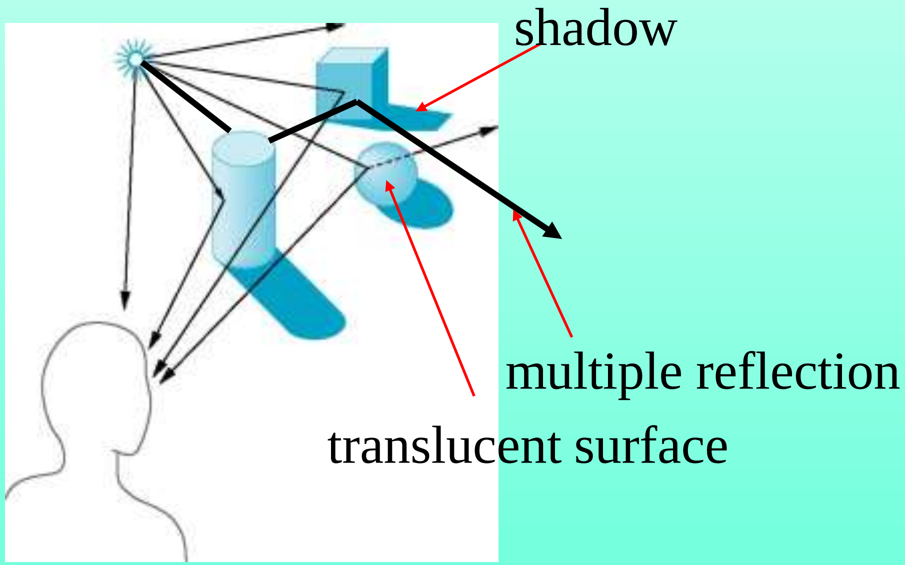
Ray Tracing（光线跟踪）：

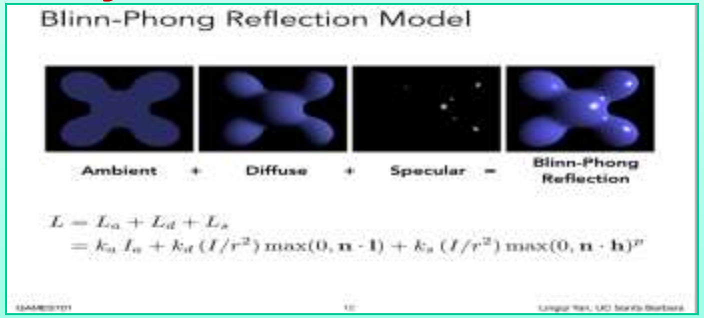
- ➢Ray Casting~逆向跟踪
- ➢Whitted-Style~逆向跟踪，递归
- ➢Path Tracing~逆向跟踪，递归，渲染方程及蒙特卡洛积分

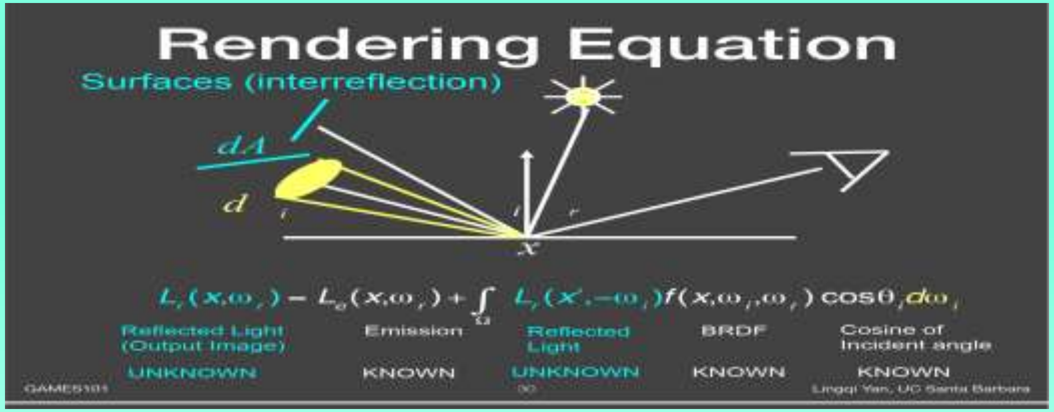


Summary(cont.)

计算模型：Blinn-Phong, Whitted-Style Model, Rendering Equation







Summary（cont.)

- **Local Illumination 局部光照**
  - consider scattering of light between light and the object（只考虑直接光照,）
  - Not consider the multiple scattering of light between the objects（不考虑物体之间的光线弹射）
  - can not produce shadow ,but can use other technique to generate hard shadows
- **Global Illumination 全局光照**
  - Direct illumination+ Indirect illumination（直接光照+间接光照）
  - Consider the multiple scattering of light between the objects，Multiple scattering from object to object（考虑光线在物体之间的多次弹射）
  - Global effect includes soft shadows（可产生软阴影））