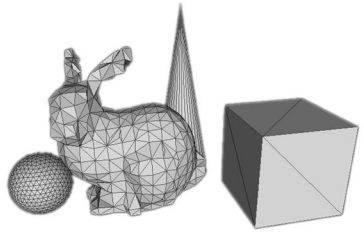
Introduction to Computer Graphics



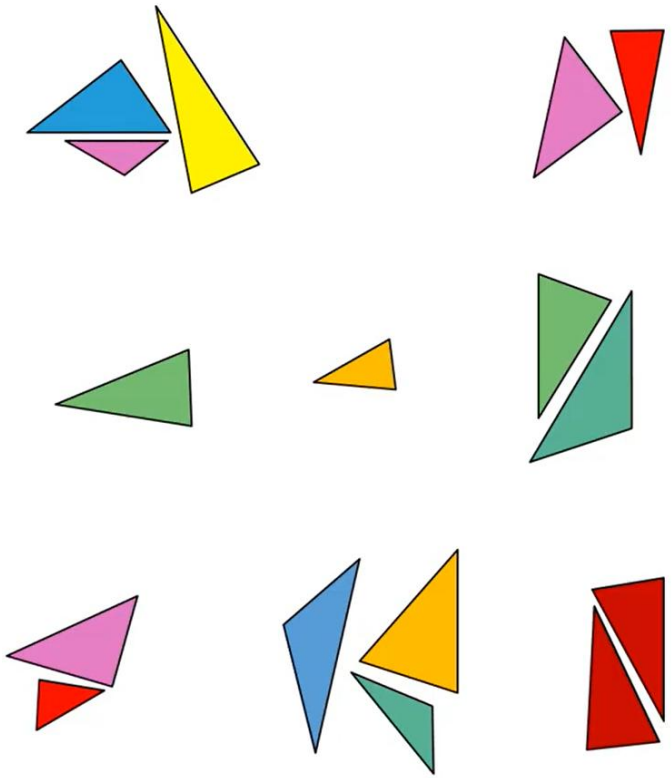
Ray Tracing 2
光线跟踪 2

光线跟踪

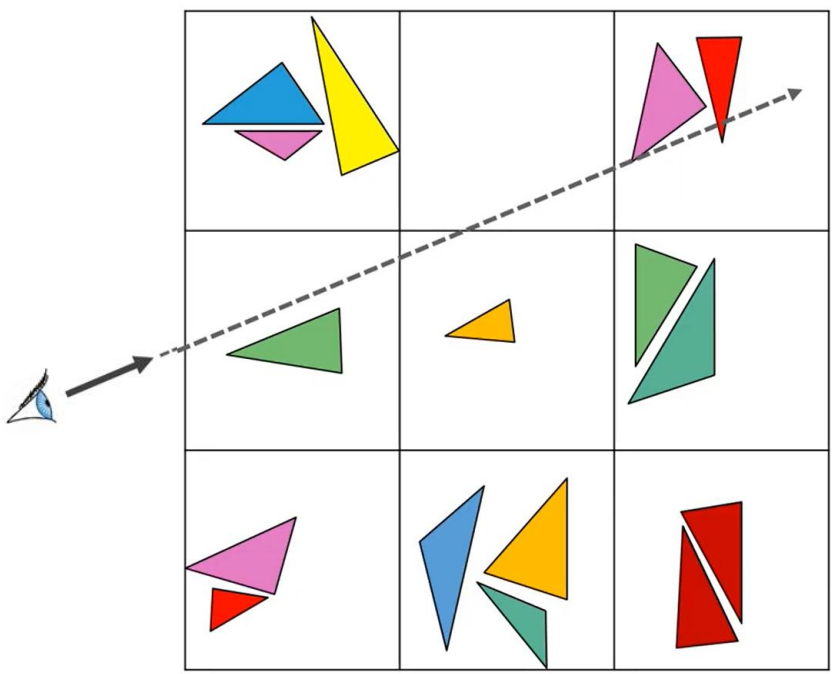
- 光线跟踪基本原理
- Whitted-Style 光线跟踪
- 光线跟踪实现细节
  - 加速结构
  - 抗锯齿 （超采样）
- 光线跟踪进阶话题
  - 辐射度概念
  - 渲染方程
  - 全局光照明、路径跟踪（Path Tracing）

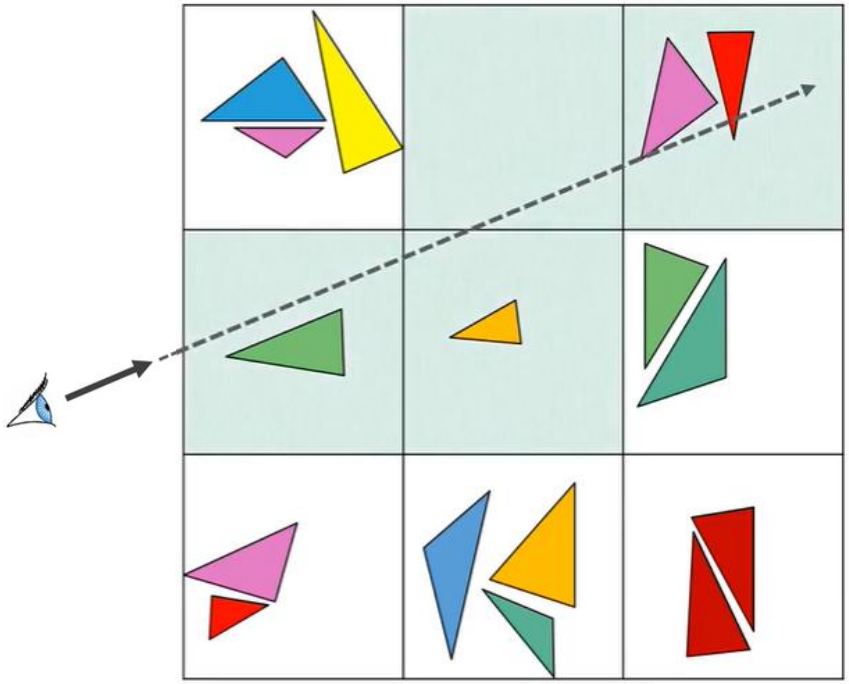
空间加速结构

- 复杂场景需要大量求交计算



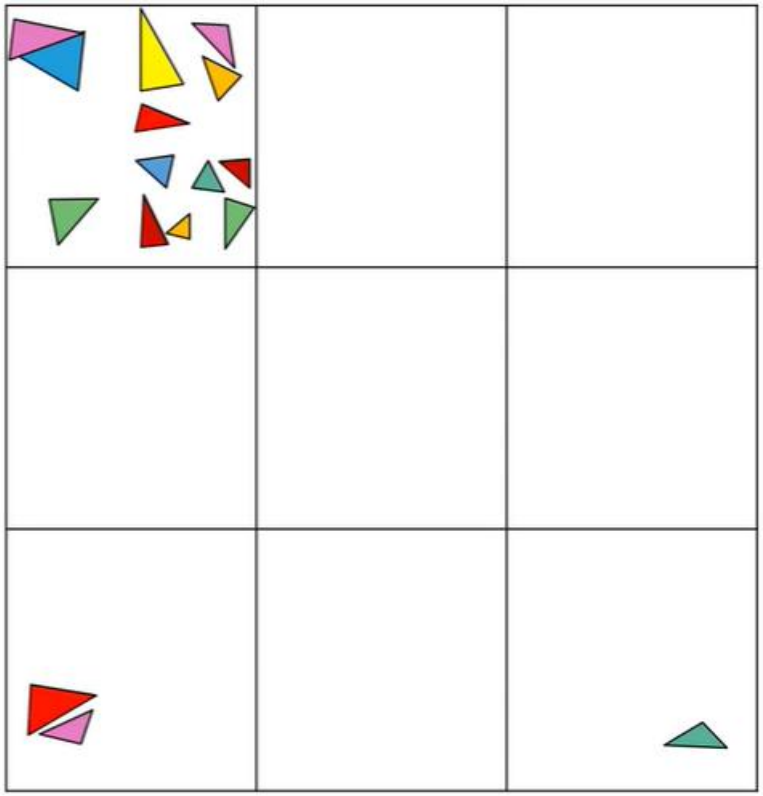
均匀分割网格

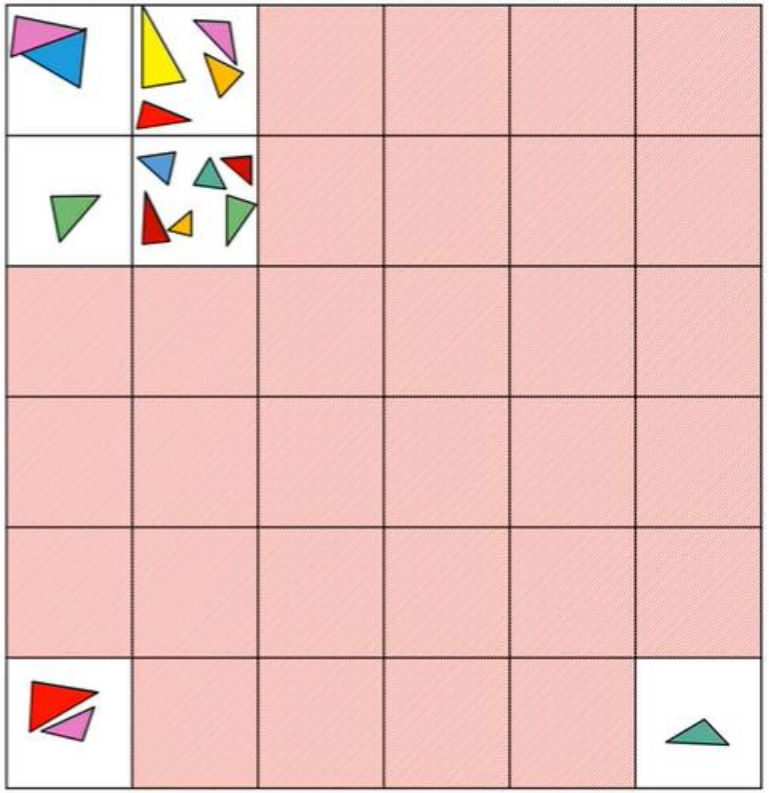




均匀分割网格

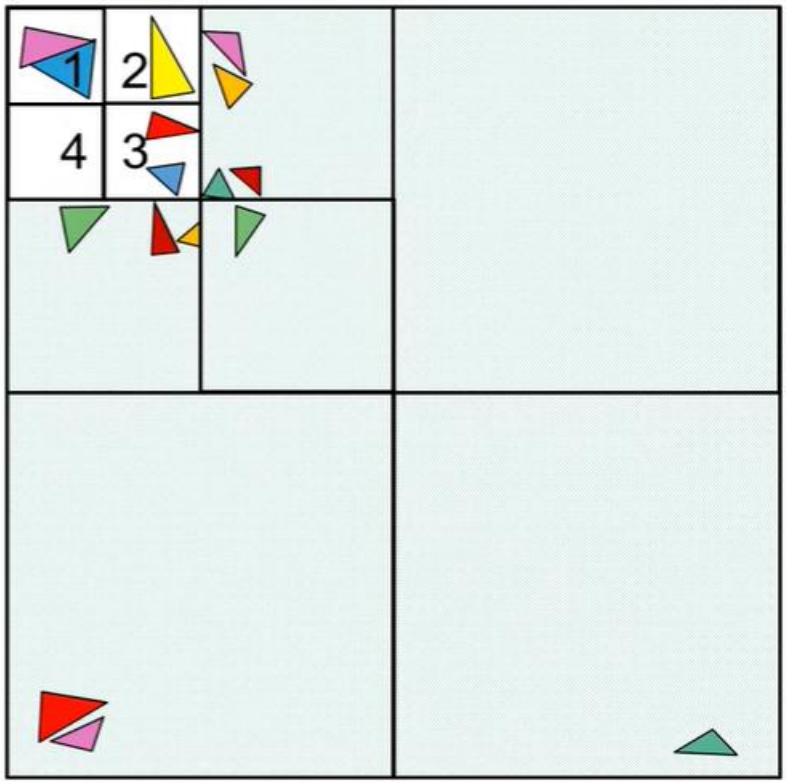
- 物体分布不均匀？
- 更小的网格-> 大量空间和计算浪费





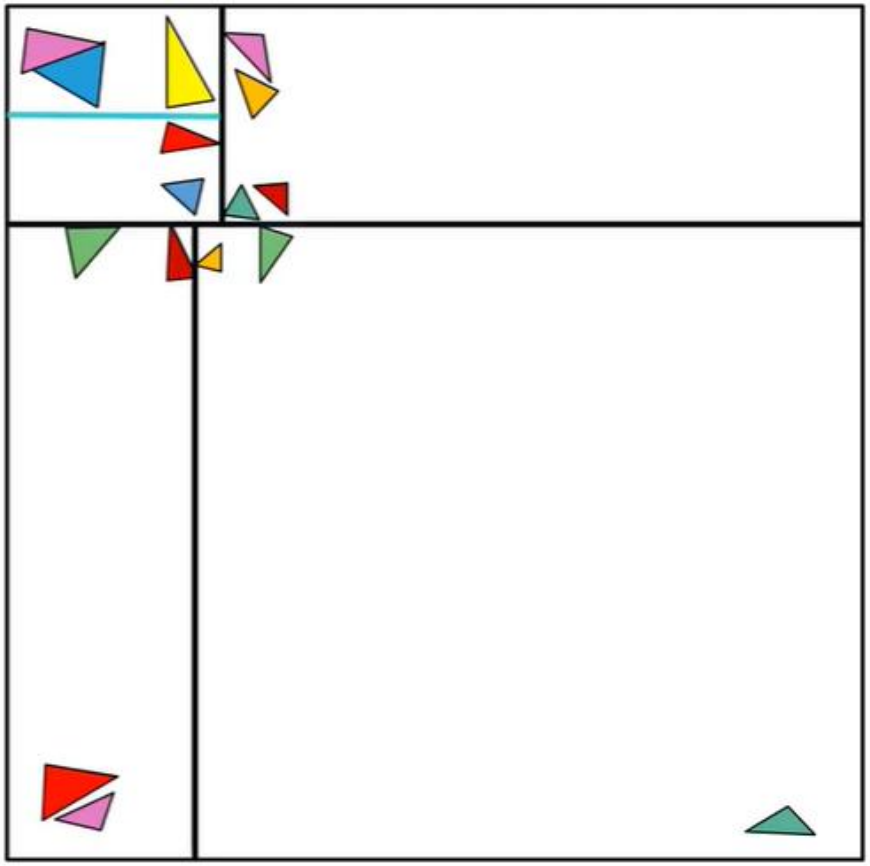
Octree（八叉树）

- 当子空间为空或者三角形个数较少时，不再细分
- 每一次细分，原网格一分为八 -> 深度加深，指数爆炸



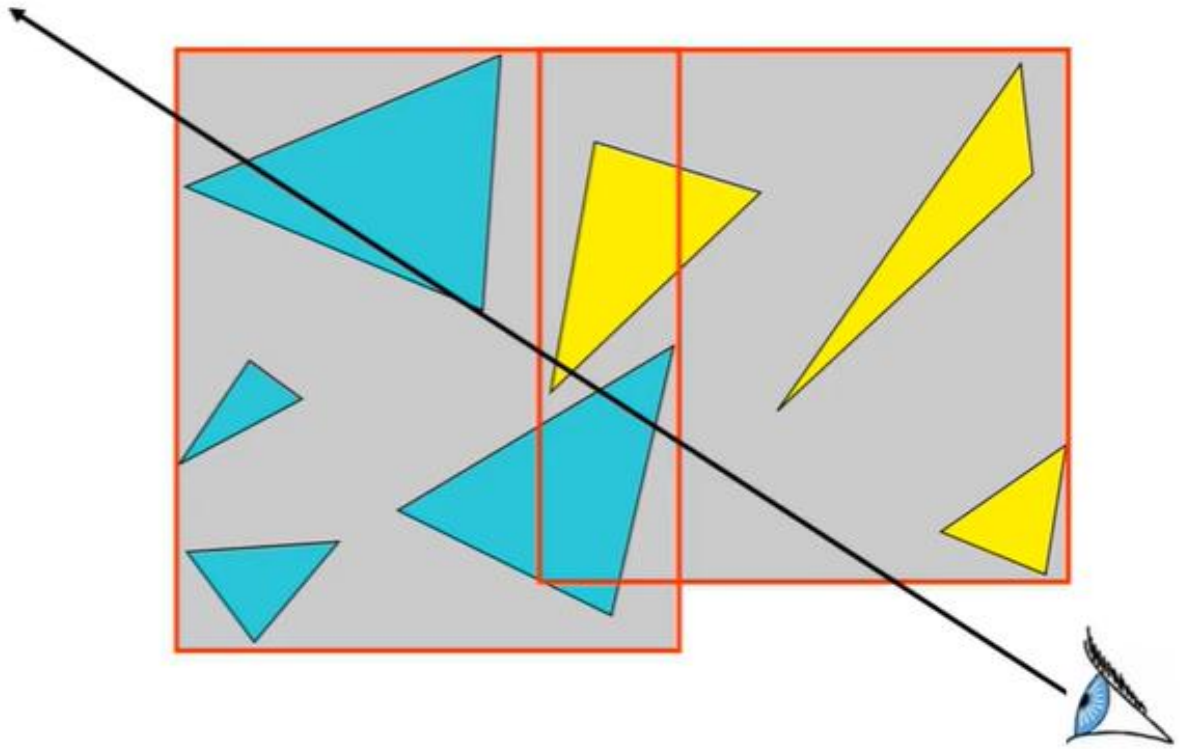
KD Tree

- 每次只分一个维度（每次一分为二）
- 可能某个三角形会隶属于左右两个空间



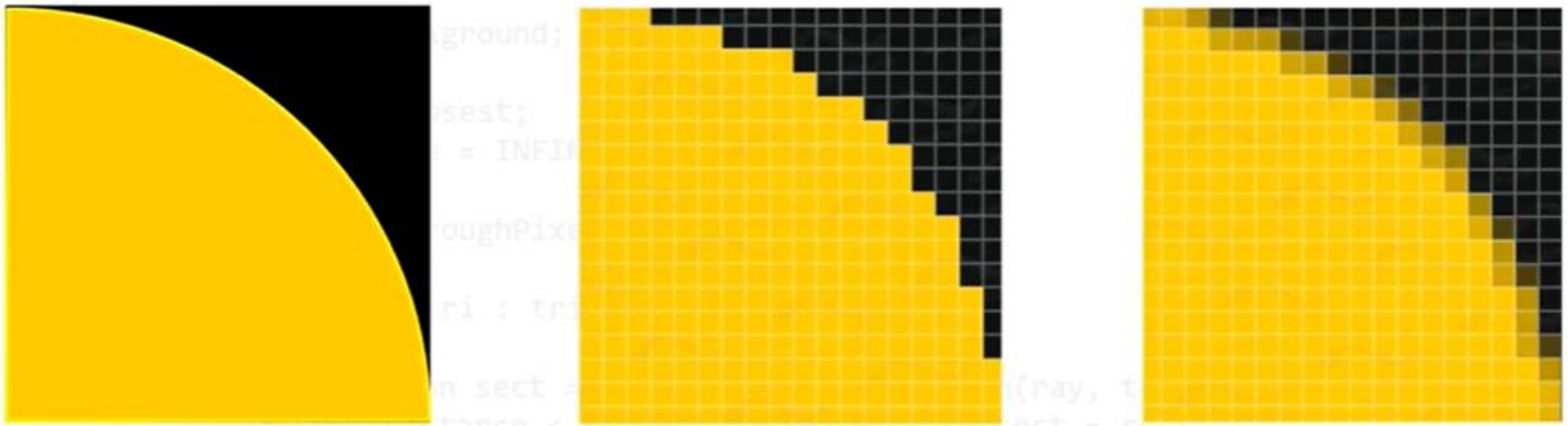
BVH (Bounding Volume Hierarchy)

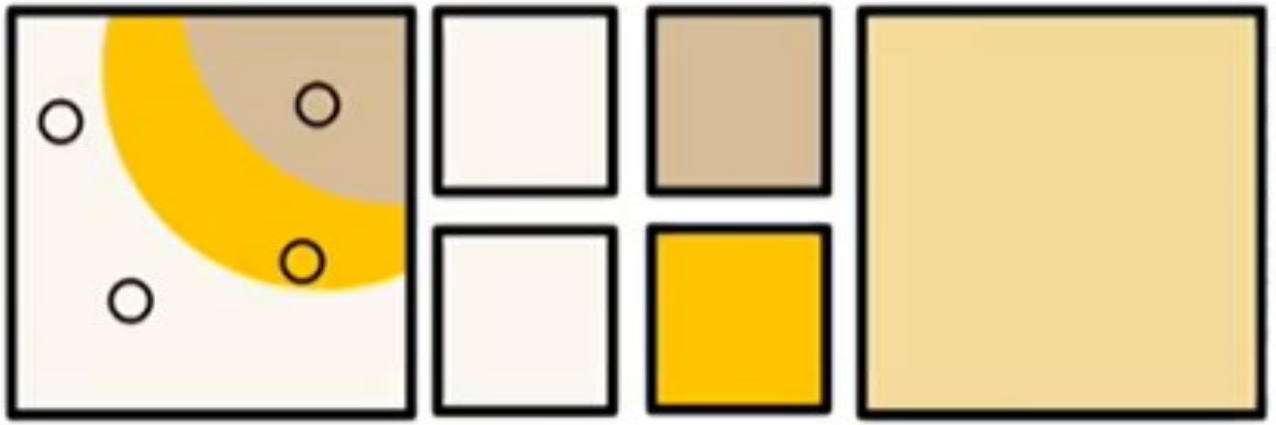
- 类似KD Tree剖分方式
- 允许区间重叠，不会出现一个三角形隶属两边的情况



抗锯齿

- 随机采样





Whitted-Style 方法的不足

- Whitted-Style 光线跟踪
  - 对每个点，计算局部光照明
  - 如果是镜面或者折射面，则继续递归计算
  - 如果不是镜面或者折射面，则仅包含局部光照明
- 定性、不准确
- 没有反映出完整的全局光照明效果，尤其是当物体表面不是镜面或者折射面时

辐射度学

- 辐射度学（Radiometry）是度量电磁辐射能量传输的学科，也是基于物理着色模型的基础。
- 准确度量光照中的物理量

- 能量(Energy): 符号Q，单位J。

- 功率(Power):单位瓦特（Watts），或者焦耳 / 秒（J/s），辐射度学中，辐射功率也被称为辐射通量（Radiant Flux）或者通量（Flux），指单位时间内通过表面或者空间区域的能量的总量，符号Φ

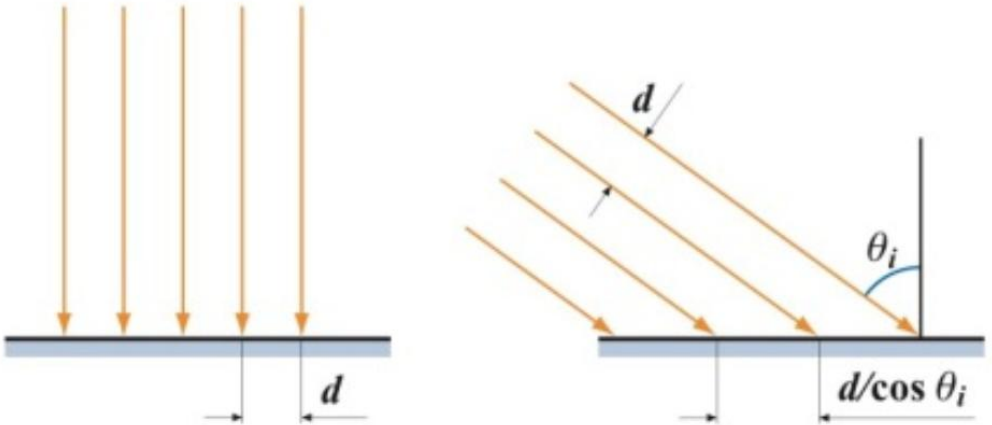
$$\Phi = \frac{dQ}{dt}$$

辐照度和幅出度

- 辐照度（Irradiance），指单位时间内到达单位面积的辐射能量，或到达单位面积的辐射通量，也就是通量对于面积的密度。用符号E表示，单位W/m^2。

$$E = \frac{d\Phi}{dA}$$

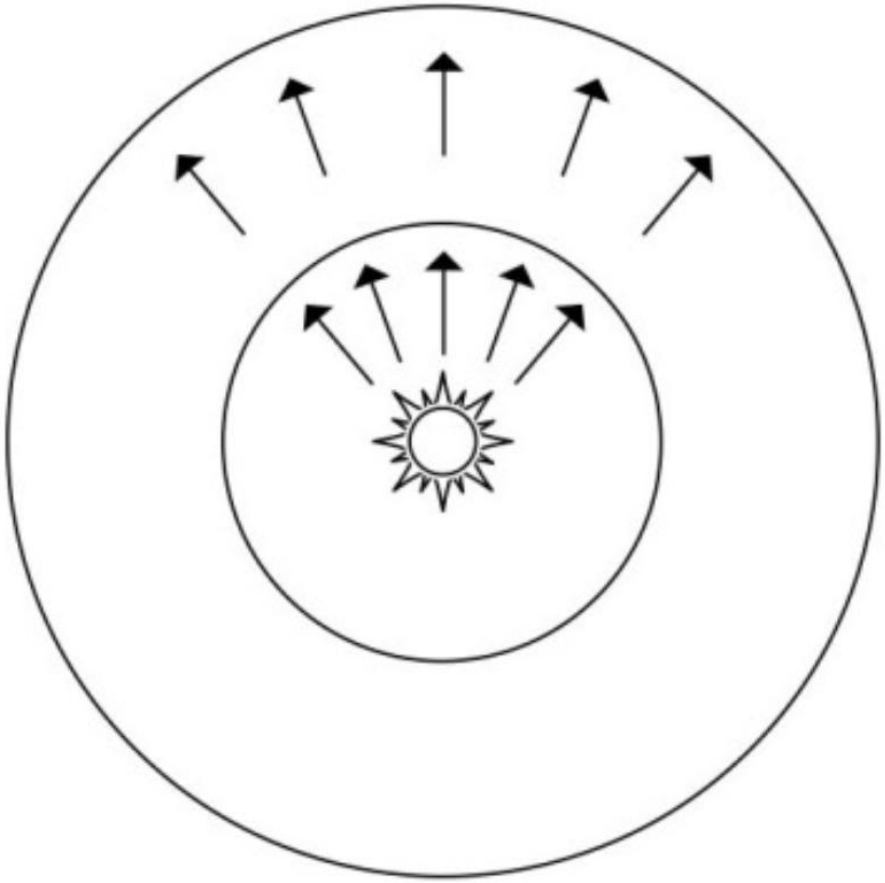
- 考虑入射角度和表面法向的关系



辐照度和幅出度

- 辐出度（Radiant Existance），也称为辐射出射度、辐射度（Radiosity），用符号M表示。辐出度与辐照度类似，唯一的区别在辐出度衡量的是离开表面的通量密度，辐照度衡量的是到达表面的通量密度。辐照度和辐出度都可以称为辐射通量密度（Radiant Flux Density）。

用辐照度概念理解光衰减和距离的关系



辐射强度

- 立体角（Solid Angle）：立体角可以看成是弧度的三维扩展。立体角则是度量三维角度的量，用符号ω表示，单位为立体弧度（也叫球面度，Steradian，简写为sr），等于立体角在单位球上对应的区域的面积:

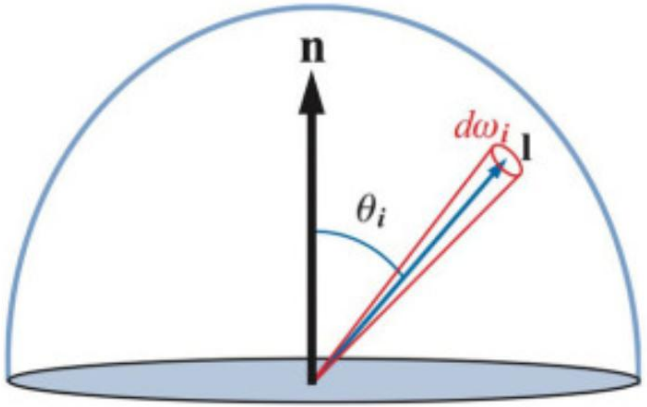
$$\omega = \frac{S}{r^2}$$

- 整个球面的立体角是4π，半球面的立体角是2π。

辐射强度

- 辐射强度：指通过单位立体角的辐射通量。用符号I表示，单位W/sr。

$$I = \frac{d\Phi}{d\omega}$$

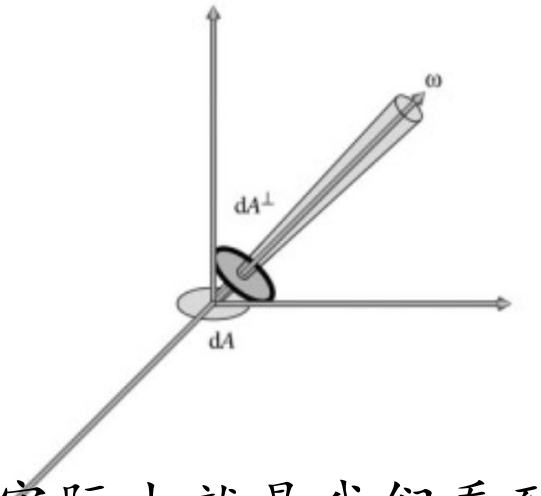


- 之所以引入辐射强度，是因为有时候要度量通过一个点(上图中的球心）的通量的密度，但因为点的面积是0，无法使用辐照度，所以引入辐射强度。辐射强度不会随距离变化而变化，不像点光源的辐照度会随距离增大而衰减，这是因为立体角不会随距离变化而变化。

辐射率

- 我们常需要度量从一个微小面积表面出发，射向某个微小方向的通量（或者来自某个微小方向，照射到微小面积表面的通量），辐射率就是度量这种情况的量。
- 辐射率（Radiance），指每单位面积每单位立体角的辐射通量密度。用符号L表示，单位W/m^2sr，定义为：

$$L = \frac{d\Phi}{d\omega dA^{\perp}}$$



- 辐射率不随距离增加而衰减，实际上就是我们看到的颜色。

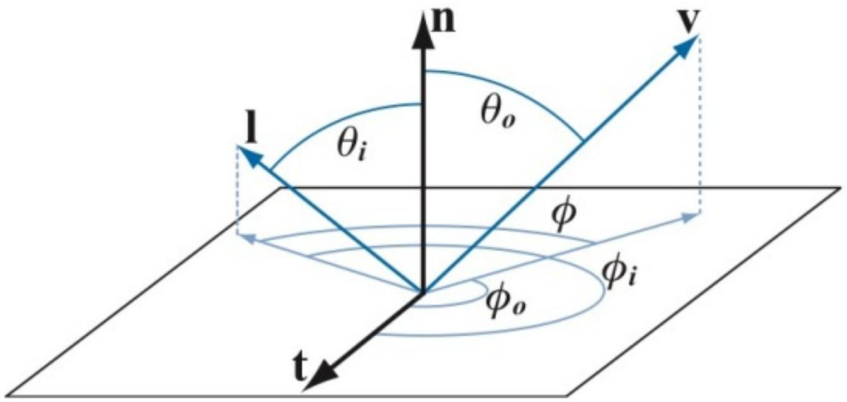
BRDF

- 我们看到一个表面，实际上是周围环境的光照射到表面上，然后表面将一部分光反射到我们眼睛里。
- 双向反射分布函数BRDF（Bidirectional Reflectance Distribution Function）就是描述表面入射光和反射光关系的。

BRDF

- 定义：

$$f(l,v) = \frac{dL_o(v)}{dE(l)}$$

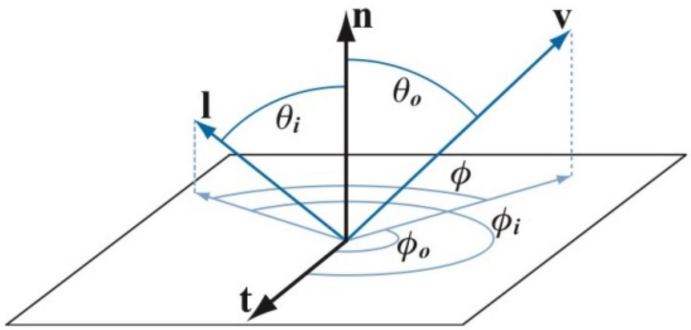


- 如何理解：入射光l打到点p的一个微小面元，形成辐照度E, E按照一定比例朝v方向反射，把这个比值定义为入射方向l，出射方向v的BRDF值。
- 经过推导，朗伯表面的BRDF=c/π, c是[0,1]间的常数。

反射方程与渲染方程

- 反射方程：

$$L_o(v) = \int_{\Omega} f(l, v) L_i(l) cos\theta_i d\omega_i$$



- 渲染方程：

$$L_o(v) = L_e + \int_{\Omega} f(l, v) L_i(l) cos\theta_i d\omega_i$$

路径跟踪

$$L_o(v) = L_e + \int_{\Omega} f(l, v) L_i(l) cos\theta_i d\omega_i$$

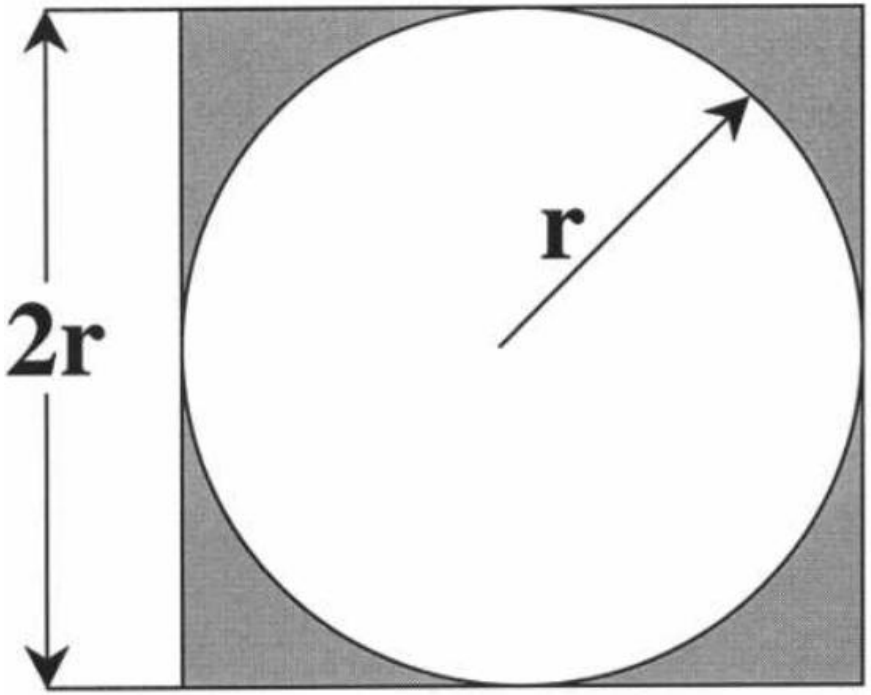
自发光

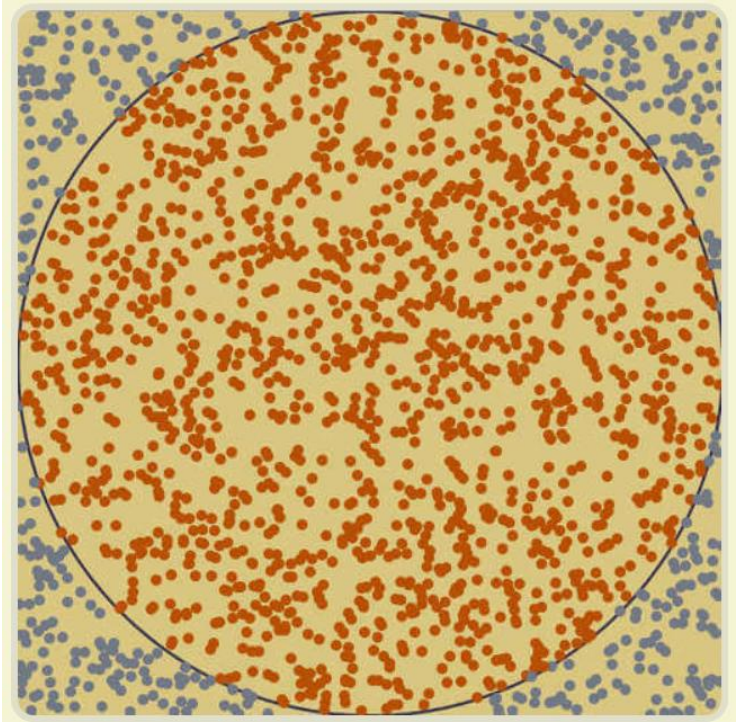
BRDF

入射辐射率
考虑递归

- 需要进行半球面积分，多次折射之后，引起指数灾难。

蒙特卡洛模拟





路径跟踪

```
void Render(Image finalImage, count numSamples) {
  foreach (pixel in finalImage) {
    foreach (i in numSamples) {
      Ray r = camera.generateRay(pixel);
      pixel.color += TracePath(r, 0);
    }
    pixel.color /= numSamples;  // Average samples.
  }
}
```

路径跟踪

```
Color TracePath(Ray ray, count depth) {
  if (depth >= MaxDepth) {
    return Black;  // Bounced enough times.
  }

  ray.FindNearestObject();
  if (ray.hitSomething == false) {
    return Black;  // Nothing was hit.
  }

  Material material = ray.thingHit->material;
  Color emittance = material.emittance;

  // Pick a random direction from here and keep going.
  Ray newRay;
  newRay.origin = ray.pointWhereObjWasHit;

  // This is NOT a cosine-weighted distribution!
  newRay.direction = RandomUnitVectorInHemisphereOf(ray.normalWhereObjWasHit);

  // Probability of the newRay
  const float p = 1/(2*M_PI);

  // Compute the BRDF for this ray (assuming Lambertian reflection)
  float cos_theta = DotProduct(newRay.direction, ray.normalWhereObjWasHit);
  Color BRDF = material.reflectance / M_PI ;

  // Recursively trace reflected light sources.
  Color incoming = TracePath(newRay, depth + 1);

  // Apply the Rendering Equation here.
  return emittance + (BRDF * incoming * cos_theta / p);
}
```

蒙特卡洛采样解决指数爆炸问题

Smallpt:99行全局光照明代码

- http://www.kevinbeason.com/smallpt/#moreinfo