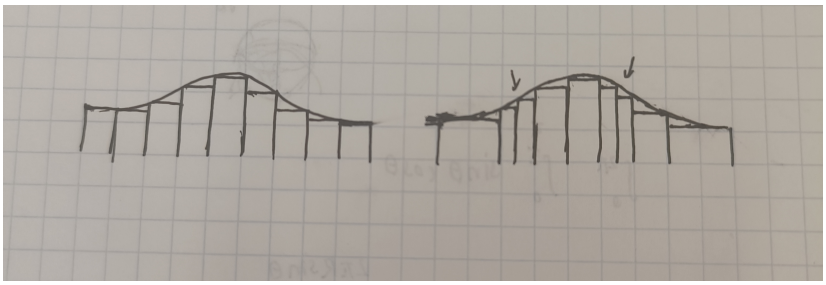
PDF 的基本思路

蒙特卡洛算法

蒙特卡洛算法就是进行多次采样并依据采样结果进行统计估计的算法。如计算求不规则图形面积时可以取一个包含了所求图形的规则图形，在这个规则图形里随机投点，并对落到所求图形里的点进行计数，则有

$$S = \lim_{N_0 \to \infty} \frac{N}{N_0} S_0$$

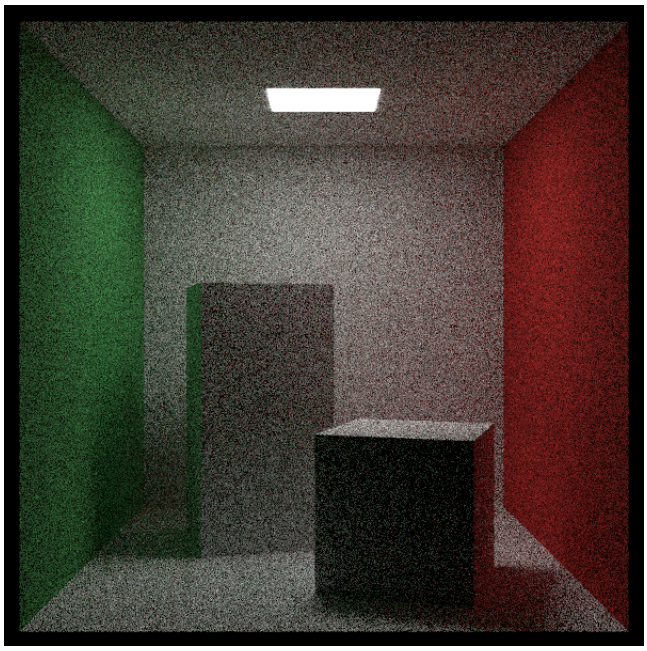
在求定积分时，我们常把所求部分在 x 轴上切成均匀的小段，再求这些小段对应的小长方体面积之和，这样在分划充分细之后显然能非常接近真实值，但如果我们在函数值变化不大的地方划分的宽一些，在变化快的地方划分的细一些，在划分段数（即采样数量）相同时显然会更接近真实值。



这就是 PDF 的基本思想，即对采样数量进行人为分配，以更快地趋近真实值。

不用 PDF 的效果

用简单的蒙特卡洛算法均匀采样生成的 cornellbox 图片有很大的噪声：



这主要是因为在定义上方的方形光源时采用的材质 DiffuseLight 的实现思路是：在对每条光线进行采样时，如果碰到了材质为 DiffuseLight 的物体，则不再继续向下递归并返回该物体的颜色。因此，只有当光射到光源时才会对其进行采样，当光源很小或者距离较远时，采样数目很小，导致噪声较大。所以说需要人为地对光源多采样，但单纯地加光源的采样数量会导致图片的亮度不正常地增加，需要对光线样本的权重进行重新分配。

通过合适的概率密度函数（PDF），我们可以把之前用的均匀的概率分布变为非均匀的，并将概率大的部分分配到比较重要的位置（如光源）。

PDF 的实现

最开始的 Lambertian 材质

Lambertian 材质中，光线全部向各方向散射，实现散射的方式如下（第二本书结束后尚未引入 PDF）：

```
    impl<T: Texture> Material for Lambertian<T> {
    fn scatter(
        &self,
        r_in: &Ray,
        rec: &HitRecord,
        attenuation: &mut Color,
        scattered: &mut Ray,
    ) -> bool {
        let mut scatter_direction = rec.normal + random_unit_vec();
        if scatter_direction.near_zero() {
            scatter_direction = rec.normal;
        }
        *scattered = Ray {
            orig: (rec.p),
            dir: (scatter_direction),
            time: r_in.time,
        };
        *attenuation = self.albedo.value(rec.u, rec.v, &rec.p);
        true
    }
}
```

其中 random_unit_vec() 实现如下:

```
pub fn random_vec(min: f64, max: f64) -> Vec3 {
    Vec3 {
        e: (
            random_double(min, max),
            random_double(min, max),
            random_double(min, max),
        ),
    }
}
pub fn random_in_unit_sphere() -> Vec3 {
    let mut p = random_vec(-1.0, 1.0);
    loop {
        if p.length_square() < 1.0 {
            break;
        }
        p = random_vec(-1.0, 1.0);
    }
```

```
    p
}
pub fn random_unit_vec() -> Vec3 {
    random_in_unit_sphere().unit_vector()
}
```

可以看到，散射的方向就是表面的法向量加上一个在以法向量为半径的单位球（入射点在球表面而非球心）中均匀随机地取到的向量的方向向量（易得这个单位向量指向任一方向的概率都是相等的）。每个散射方向对应球内的一个点，因此，光线散射到与法向量夹角为 θ 的方向的概率与 $cos\theta$ 成正比，也就是说，Lambertian 材质已经隐式地实现了对靠近法向量的多采样、远离法向量的少采样的 PDF。

附：事实上朗伯模型的完全漫反射表面是各方向反射光强和入射角的余弦值成正比而与反射无关，各个方向上的反射光具有相同的亮度，但我们的光线追踪器是“让相机发出光”，因此叙述是反过来的。

随机方向的生成

生成关于 z 轴对称的半球上的随机方向

给定一个关于与 z 轴夹角 θ 的 PDF $f(\theta)$。先取两个 $(0,1)$ 上均匀随机数 r_1、r_2，对于绕 z 轴的角度（不妨认为是与 x 轴的夹角）ϕ, 有:

$$\int_0^{\phi} \frac{1}{2\pi}\, dt = r_1$$

其意义是：因为 ϕ 在 $(0,2\pi)$ 上均匀分布，所以一个方向绕 z 轴的角度小于 ϕ 的概率为 $\frac{\phi}{2\pi}$，这个值能与 $(0,1)$ 上的一个均匀随机数（即 r_1）相对应，从而 ϕ 也能和 r_1 相对应。

对于像 ϕ 这种均匀分布的随机角度，上面的论述看起来是显然的，但是对于要服从一个并不均匀的 PDF 的 θ 来说，其满足：

$$\int_0^{\theta} 2\pi f(t) sin(t)\, dt = r_2$$

任取一个单位方向，其与 z 轴夹角小于 θ 的概率是 $\int_0^{\theta} 2\pi f(t) sin(t)\, dt$，（将半球均匀划分为小面积微元，则选到夹角为 θ 的面积微元的概率为 $2\pi sin(t) \cdot f(\theta)$）。和上面类似，均匀的随机数 r_2 与非均匀的 θ 以此建立起了对应关系。

令 $f(\theta) = \frac{cos\theta}{\pi}$，将 θ 和 ϕ 解出来，得到 $cos\theta = \sqrt{1-r_2}, \phi = 2\pi r_1$，从而得到了一个与 z 轴夹角服从 $f(\theta)$ 的随机方向

$$(cos(2\pi r_1)\sqrt{r_2}, sin(2\pi r_1)\sqrt{r_2}, \sqrt{1-r_2})$$

关于表面法向量对称的随机方向

只需要建立一个以表面法向量为 z 轴的标准正交基，再按如上方法生成关于标准正交基 z 轴对称的半球上随机方向即可。

生成服从 $f(\theta) = \frac{cos\theta}{\pi}$ 的随机方向的代码如下（CosinePdf 中的 generate() 函数）：

```
    pub fn random_cosine_direction() -> Vec3 {
    let r1 = random_double(0.0, 1.0);
    let r2 = random_double(0.0, 1.0);
    let z = (1.0 - r2).sqrt();
    let phi = 2.0 * PI * r1;
    let x = phi.cos() * r2.sqrt();
    let y = phi.sin() * r2.sqrt();
    Vec3 { e: (x, y, z) }
```

```
}
```

```
#[derive(Clone)]
pub struct CosinePdf {
    pub uvw: Onb,
}
impl Pdf for CosinePdf {
    fn value(&self, direction: &Vec3) -> f64 {
        let cosine = mul_vec_dot(direction.unit_vector(), self.uvw.axis_z);
        if cosine <= 0.0 {
            0.0
        } else {
            cosine / PI
        }
    }
    fn generate(&self) -> Vec3 {
        self.uvw.local_vec(&random_cosine_direction())
    }
}
```

其中 Onb 为标准正交基类。

将 PDF 应用到光线追踪器中

应用 PDF 后，计算 Lambertian 材质散射光线的方式变为：

$$\frac{1}{N}\sum\frac{color(r_{pdf})\cdot P_{scatter}(r_{pdf})}{P_{pdf}(r_{pdf})} \tag{1}$$

与应用 PDF 前的计算方式相比：

$$\frac{1}{N}\sum color(r_{scatter}) \tag{2}$$

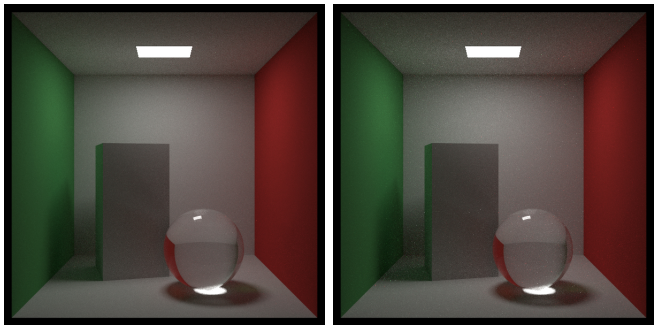
其中 r_{pdf} 和 $r_{scatter}$ 分别表示依据任一 PDF 选取采样的光线和根据材料散射特性选取采样光线（已经论述其实质上是特殊的 PDF），当 $N\to+\infty$ 时，(1) 式可认为是

$$\frac{1}{N}\sum\frac{color(r)\cdot P_{scatter}(r)}{P_{pdf}(r)}\cdot P_{pdf}(r)$$

(2) 式可认为是

$$\frac{1}{N}\sum color(r)\cdot P_{scatter}(r)$$

这两个式子是相等的。也就是说，虽然在程序里的变化只是把 Lambertian 材质的散射 PDF 显式地写了出来，在计算中相除为 1 可以直接消掉。但即使换成别的 PDF，只要采样数目够大，它仍然能很接近真实值。下方的左右图分别是采用了 $f(\theta)=\frac{cos\theta}{\pi}$ 和 $f(\theta)=\frac{1}{2\pi}$ 两种不同 PDF 的最终画面（采样率相同）：



右图除了噪声比左图稍微多了一些之外，与左图基本一致。

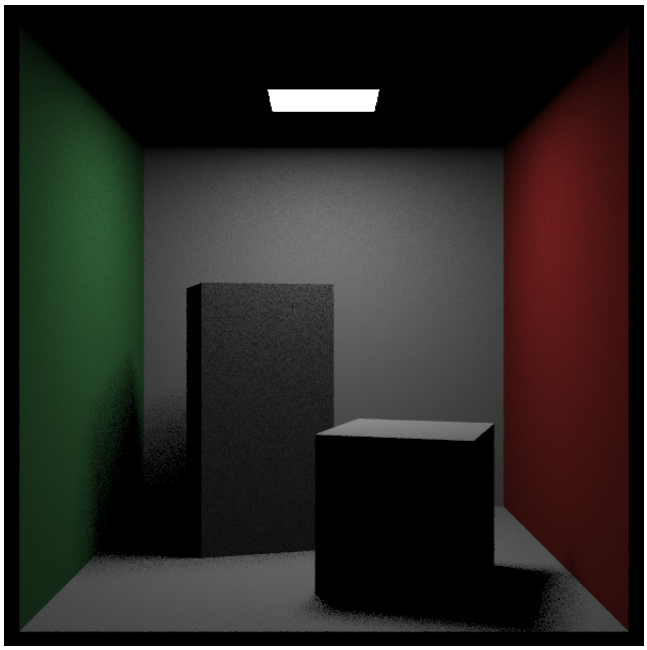
对光源的采样

光源对图片噪声的影响是很大的，为了人为地对光源进行多采样，可以定义一个和光源相关的 PDF。

先考虑只对光源发出的光线进行采样, 由于光源的形状不同,其发光情况也不同,所以对于每个作光源的 Hittable 物体都要定义 pdf_value() 和 random() 两个函数，分别用来分配正确的 PDF 权重和产生随机采样光线。这个 PDF 被定义为 HittablePdf：

```
#[derive(Clone)]
pub struct HittablePdf<'a, H: Hittable> {
    pub o: Point3,
    pub ptr: &'a H,
}
impl<'a, H: Hittable> Pdf for HittablePdf<'a, H> {
    fn value(&self, direction: &Vec3) -> f64 {
        self.ptr.pdf_value(&self.o, direction)
    }
    fn generate(&self) -> Vec3 {
        self.ptr.random(&self.o)
    }
}
```

对初始图片用上 HittablePdf 后，其效果如下图所示：



这个图片显然是“不正确”的。原因也很显然，它只对光源方向的光进行了采样，其它方向，尤其是天花板上的点被采样的概率为 0。在 PDF 中，即时一个方向的采样概率极小，只要采样数目足够最终也能得出正确的图片，但是当采样概率为 0 时根本不对那个方向进行采样，也无从得出正确的图片。但是，对光源方向的光的采样（至少在数学公式上）是正确的。

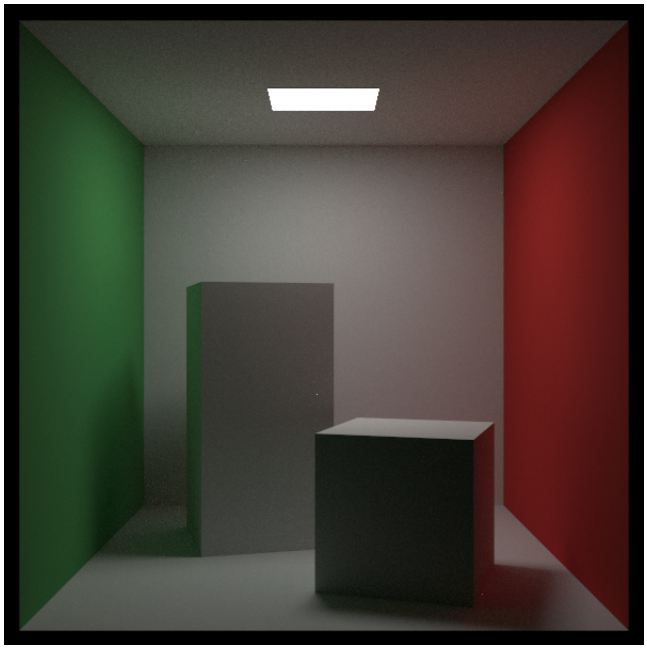
混合采样

如果把正常的 PDF 和只对光源采样的 PDF 正确地结合在一起，对其它方向的采样概率就不是 0，图片也就能是正常的。

对于任两个 PDF，它们取算术或加权平均，积分值仍然是 1，它仍然是一个合理的 PDF。对于采样方向只需要依据权值进行概率上的调整即可，由此定义的 MixturePdf 如下（没加权）：

```
    pub struct MixturePdf<'a, P1: Pdf + ?Sized, P2: Pdf + ?Sized> {
    pub p1: &'a P1,
    pub p2: &'a P2,
}
impl<'a, P1: Pdf + ?Sized, P2: Pdf + ?Sized> Pdf for MixturePdf<'a, P1, P2> {
    fn value(&self, direction: &Vec3) -> f64 {
        0.5 * self.p1.value(direction) + 0.5 * self.p2.value(direction)
    }
    fn generate(&self) -> Vec3 {
        if random_double(0.0, 1.0) < 0.5 {
            self.p1.generate()
        } else {
            self.p2.generate()
        }
    }
}
```

用 MixturePdf 把 HittablePdf 和正常的 CosinePdf 结合到一起，就可以得到对光源着重采样的正确的图片：



总结

PDF 可以通过巧妙地分配采样权重，在相同采样数目下取得更好的效果。除了对 DiffuseLight 进行着重采样之外，也可以对 Dielectric 等和场景光线有关的物体进行着重采样，以提高图片效果。