Monte Carlo Illumination

CS 5625 Lecture 4

Surface illumination integral (as sum)

- **BRDF tells you how light from a single direction is reflected**
- **Light coming from a small source behaves similarly**
- **What about light coming from everywhere?**
  - approximate incoming light with many small sources on a sphere (the little bug can't tell the difference…)
  - reflected light is sum of reflected light due to each source (each source has its size Ω_k, brightness L_k, and direction ω_k)

$$L_r(\omega_r) = \sum_k \Omega_k L_k f_r(\omega_k, \omega_r) |\omega_k \cdot \mathbf{n}|$$

reflected light in direction ω_r

"intensity" of light source k

BRDF

cosine factor

Surface illumination integral

- **Take the limit as the little area sources get smaller**
  - collection of separate brightnesses L_k becomes a function $L_i(\omega_i)$
  - size of sources turns into an integration measure $d\sigma$

$$L_r(\omega_r) = \int_{S^2_+} L_i(\omega_i) f_r(\omega_i, \omega_r) |\omega_i \cdot \mathbf{n}| d\sigma(\omega_i)$$

"The light reflected to direction ω_r is the integral, over the positive unit hemisphere, of the incoming light times the BRDF times the incoming cosine factor, with respect to surface area."

Monte Carlo Integration

- **Monte Carlo idea: design a random experiment whose average outcome is the answer we want**
- **Integration:**

$$I = \int_a^b f(x)dx$$

- **want to define an "estimator" $g(x)$ such that**

$$E\{g(x)\} = I \qquad \text{for random values of } x$$

  - **that is, the expected value of g is the answer we seek when x is chosen randomly.**

Uniform sampling

- **If *x* is chosen uniformly at random from [*a*, *b*]:**

$$E\{f(x)\} = \frac{1}{b-a}\int_a^b f(x)dx$$

- so, to get the desired answer, set

$$g(x) = (b-a)f(x)$$

- then

$$E\{g(x)\} = \int_a^b f(x)dx = I \qquad \text{for } x \text{ uniform in } [a, b]$$

Uniform sampling revisited

- **Choosing points uniformly from [a, b] is sampling from a pdf that has density 1 / (b – a).**
  - if we use an estimator g with uniformly sampled x:

$$E\{g(x)\} = \int_a^b g(x)p(x)\,dx = \frac{1}{b-a}\int_a^b g(x)dx$$

  - so if f is the desired integrand, the correct estimator is

$$g(x) = (b-a)f(x)$$

Convergence rate

- **We can get a better estimate of the expected value of *g* by generating several values and averaging them.**

$$G_n = \frac{1}{N} \sum_{i=1}^{n} g(x_i) \quad \text{where } x_i \sim p$$

- **As *n* increases, the variance of G_n decreases**

$$\sigma^2 \left\{ \sum_{i=1}^{n} g(x_i) \right\} = \sum_{i=1}^{n} \sigma^2\{g\} = N\sigma^2\{g\}$$

$$\sigma\{G_n\} = \frac{\sigma\{g\}}{\sqrt{N}}$$

Nonuniform sampling

- **Choosing points instead from some other distribution over the interval [a, b] also works just as well**
  - if we use an estimator g with $x \sim p(x)$

$$E\{g(x)\} = \int_a^b g(x)p(x)\,dx$$

  - so if f is the desired integrand, the correct estimator is

$$g(x) = \frac{f(x)}{p(x)}$$

$$E\{g(x)\} = \int_a^b \frac{f(x)}{p(x)}p(x)\,dx = \int_a^b f(x)dx$$

as long as p(x) is not zero!

Monte Carlo illumination

- **Monte Carlo integration is widely used to compute illumination integrals**
  - integrand: product of illumination and BRDF and cosine factor

$$L_r(\omega_r) = \int_{S^2_+} L_i(\omega_i) f_r(\omega_i, \omega_r) |\omega_i \cdot \mathbf{n}| d\sigma(\omega_i)$$

  - if we choose:

$$\omega_i \sim p(\omega_i) \quad \text{and set: } g(\omega_i) = \frac{L_i(\omega_i) f_r(\omega_i, \omega_r) |\omega_i \cdot \mathbf{n}|}{p(\omega_i)}$$

  - then: $E\{g(\omega_i)\} = L_r(\omega_r)$ (as long as $p > 0$ over the whole hemisphere)
  - this is an algorithm for computing L_r!

Example: uniform sampling

- **If we select directions uniformly over the hemisphere…**
  - then:

    $$p(\omega_i) \sim 1/(2\pi)$$

    (see notebook for how…)

  - 2π because that is the area (solid angle) of the hemisphere; that way, probability integrates to 1
  - the correct estimator is:

$$g(\omega_i) = \frac{L_i(\omega_i) f_r(\omega_i, \omega_r) |\omega_i \cdot \mathbf{n}|}{p(\omega_i)}$$
$$= 2\pi L_i(\omega_i) f_r(\omega_i, \omega_r) |\omega_i \cdot \mathbf{n}|$$

Example: cosine-proportional sampling

- **If we select directions proportional to** $|\omega_i \cdot \mathbf{n}|$ (see notebook for how…)
  - then:

$$p(\omega_i) \sim |\omega_i \cdot \mathbf{n}|/\pi$$

  - factor of π needed so that probability integrates to 1
  - the correct estimator is:

$$g(\omega_i) = \frac{L_i(\omega_i) f_r(\omega_i, \omega_r) |\omega_i \cdot \mathbf{n}|}{p(\omega_i)}$$
$$= \pi L_i(\omega_i) f_r(\omega_i, \omega_r)$$

Rendering diffuse surface, env. lighting

- **Start with integral**

$$L_r = \int_{S^2_+} L_i(\mathbf{w}) \cdot k_d \cdot (\mathbf{w} \cdot \mathbf{n})\, d\sigma(\mathbf{w})$$

  – choose a probability density on hemisphere: uniform

$$f(\mathbf{w}) = L_i(\mathbf{w}) \cdot k_d \cdot (\mathbf{w} \cdot \mathbf{n}) \qquad p(\mathbf{w}) = \frac{1}{2\pi} \qquad g(\mathbf{w}) = \frac{f(\mathbf{w})}{p(\mathbf{w})} = 2\pi L(\mathbf{w}) k_d (\mathbf{w} \cdot \mathbf{n})$$

  – better probability density: proportional to cos theta

$$f(\mathbf{w}) = L_i(\mathbf{w}) \cdot k_d \cdot (\mathbf{w} \cdot \mathbf{n}) \qquad p(\mathbf{w}) = \frac{\cos\theta}{\pi} \qquad g(\mathbf{w}) = \frac{f(\mathbf{w})}{p(\mathbf{w})} = \pi L(\mathbf{w}) k_d$$

Code: diffuse surface / environment light

```
Color shade(x, k_d, N) {
    result = black;
    w = sample_hemisphere(N);
    if !shadow(x, w) {
        L_env = environment.eval(w)
        result += L_env * k_d
            * dot(w, N) * 2π;
    }
    return result;
}
```

uniform hemisphere sampling

```
Color shade(x, k_d, N) {
    result = black;
    w = sample_cos_hemisphere(N);
    if !shadow(x, w) {
        L_env = environment.eval(w)
        result += L_env * k_d * π;
    }
    return result;
}
```

cosine-proportional sampling

Rendering any surface, env. lighting

- **Start with integral**

$$L_r = \int_{S^2_+} L_i(\mathbf{w}) \cdot f_r(\mathbf{v}, \mathbf{w}) \cdot (\mathbf{w} \cdot \mathbf{n}) \, d\sigma(\mathbf{w})$$

  - one choice of pdf: proportional to BRDF

$$f(\mathbf{w}) = L_i(\mathbf{w}) \cdot f_r(\mathbf{v}, \mathbf{w}) \cdot (\mathbf{w} \cdot \mathbf{n}) \qquad p(\mathbf{w}) = \frac{f_r(\mathbf{v}, \mathbf{w})}{M(\mathbf{v})} \qquad g(\mathbf{w}) = \frac{f(\mathbf{w})}{p(\mathbf{w})} = M(\mathbf{v}) L(\mathbf{w}) \cdot (\mathbf{w} \cdot \mathbf{n})$$

  - another choice: proportional to environment brightness (this generalizes uniform)

$$f(\mathbf{w}) = L_i(\mathbf{w}) \cdot f_r(\mathbf{v}, \mathbf{w}) \cdot (\mathbf{w} \cdot \mathbf{n}) \qquad p(w) = \frac{1}{M} L_i(\mathbf{w}) \qquad g(\mathbf{w}) = \frac{f(\mathbf{w})}{p(\mathbf{w})} = M f(\mathbf{v}, \mathbf{w}) \cdot (\mathbf{w} \cdot \mathbf{n})$$

Code: any surface / environment light

```
Color shade(x, V, brdf, N) {
    result = black;
    w, p_w = environment.sample(N);
    if !shadow(x, w) {
        L_env = environment.eval(w)
        f_r = brdf.eval(V, w);
        result += L_env * f_r
            * dot(w, N) / p_w;
    }
    return result;
}
```

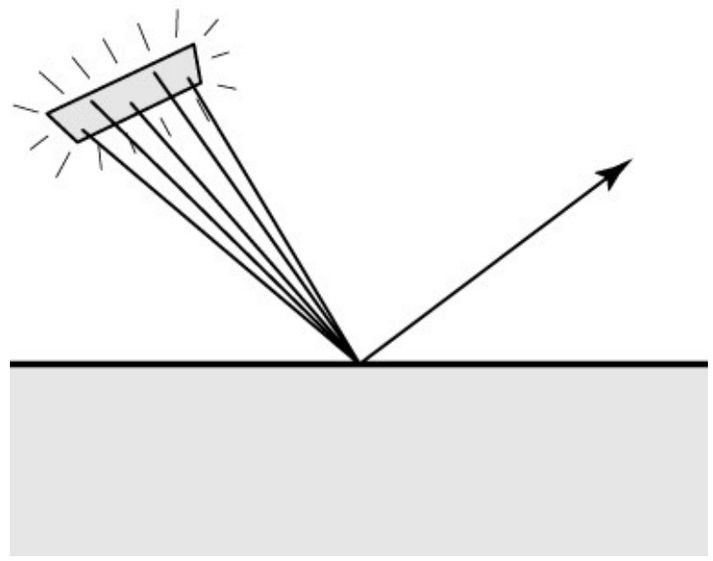
sampling by lighting environment

```
Color shade(x, V, brdf, N) {
    result = black;
    w, p_w = brdf.sample(N);
    if !shadow(x, w) {
        L_env = environment.eval(w)
        f_r = brdf.eval(V, w);
        result += L_env * f_r
            * dot(w, N) / p_w;
    }
    return result;
}
```

sampling by BRDF

Illumination from area lights

- **Area sources defined by a surface and a radiance L_s**
- **For illumination from an area source, the same math applies**
  - light defines the incident radiance distribution (directions that hit the light have radiance L_s, others zero)
  - sampling by BRDF is a poor choice unless light is very large
- **Choosing samples according to L_i**
  - there are ways to do this exactly for particular shapes
  - simple, general way is to choose points randomly on the light using a pdf over surface area
  - to make this work out easily, rewrite illumination integral as an area integral



Lighting by integrating over a source

1. **illumination from a point source**

$$L_r = I \cdot f_r(\mathbf{v}, \mathbf{w}) \cdot \frac{\cos\theta_r}{r^2} = I \cdot f_r(\mathbf{v}, \mathbf{w}) \cdot \frac{|\mathbf{n} \cdot \mathbf{w}|}{\|\mathbf{x} - \mathbf{y}\|^2}$$

2. **illumination from a small area source**

$$L_r = L_s A_s \cdot f_r(\mathbf{v}, \mathbf{w}) \cdot \frac{|\mathbf{n_x} \cdot \mathbf{w}|\,|\mathbf{n_y} \cdot \mathbf{w}|}{\|\mathbf{x} - \mathbf{y}\|^2}$$

3. **illumination from a bunch of small area sources**

$$L_r = \sum_{i=1}^{n} L_s \cdot f_r(\mathbf{v}, \mathbf{w}_i) \cdot \frac{|\mathbf{n_x} \cdot \mathbf{w}_i|\,|\mathbf{n_y} \cdot \mathbf{w}_i|}{\|\mathbf{x} - \mathbf{y}_i\|^2} \Delta A_i$$

4. **illumination from a large area source**

$$L_r = \int_S L_s \cdot f_r(\mathbf{v}, \mathbf{w}(\mathbf{y})) \cdot \frac{|\mathbf{n_x} \cdot \mathbf{w}(\mathbf{y})|\,|\mathbf{n_y} \cdot \mathbf{w}(\mathbf{y})|}{\|\mathbf{x} - \mathbf{y}\|^2} dA(\mathbf{y}) \qquad \mathbf{w}(\mathbf{y}) = \frac{\mathbf{y} - \mathbf{x}}{\|\mathbf{y} - \mathbf{x}\|}$$

Rendering any surface, area source lighting

- **Start with integral**

$$L_r = \int_S L_s \cdot f_r(\mathbf{v}, \mathbf{w}(\mathbf{y})) \cdot \frac{|\mathbf{n_x} \cdot \mathbf{w}(\mathbf{y})| \, |\mathbf{n_y} \cdot \mathbf{w}(\mathbf{y})|}{\|\mathbf{x} - \mathbf{y}\|^2} dA(\mathbf{y})$$

  - choose a probability density on S: uniform

$$f(\mathbf{y}) = L_s \cdot f_r(\mathbf{v}, \mathbf{w}(\mathbf{y})) \cdot \frac{|\mathbf{n_x} \cdot \mathbf{w}(\mathbf{y})| \, |\mathbf{n_y} \cdot \mathbf{w}(\mathbf{y})|}{\|\mathbf{x} - \mathbf{y}\|^2}$$

$$p(\mathbf{y}) = \frac{1}{A}$$

$$g(\mathbf{y}) = \frac{f(\mathbf{y})}{p(\mathbf{y})} = L_s A \cdot f_r(\mathbf{v}, \mathbf{w}(\mathbf{y})) \cdot \frac{|\mathbf{n_x} \cdot \mathbf{w}(\mathbf{y})| \, |\mathbf{n_y} \cdot \mathbf{w}(\mathbf{y})|}{\|\mathbf{x} - \mathbf{y}\|^2}$$

Code: any surface / area light

```
Color shade(x, V, brdf, N) {
    result = black;
    for light in lights {
        y, p_y = light.sample(x);
        if !shadow(x, y) {
            L = normalize(y - x);
            f_r = brdf.eval(V, L);
            result += light.radiance * f_r
                * dot(L, N) * dot(-L, light.normal)
                / distSqr(x, y)
                / p_y;
        }
    }
    return result;
}
```