Advanced Ray Tracing

CS 4620 Lecture 24

Basic ray tracing

- Many advanced methods build on the basic ray tracing paradigm
- Basic ray tracer: one sample for everything
  - one ray per pixel
  - one shadow ray for every point light
  - one reflection ray, possibly one refraction ray, per intersection

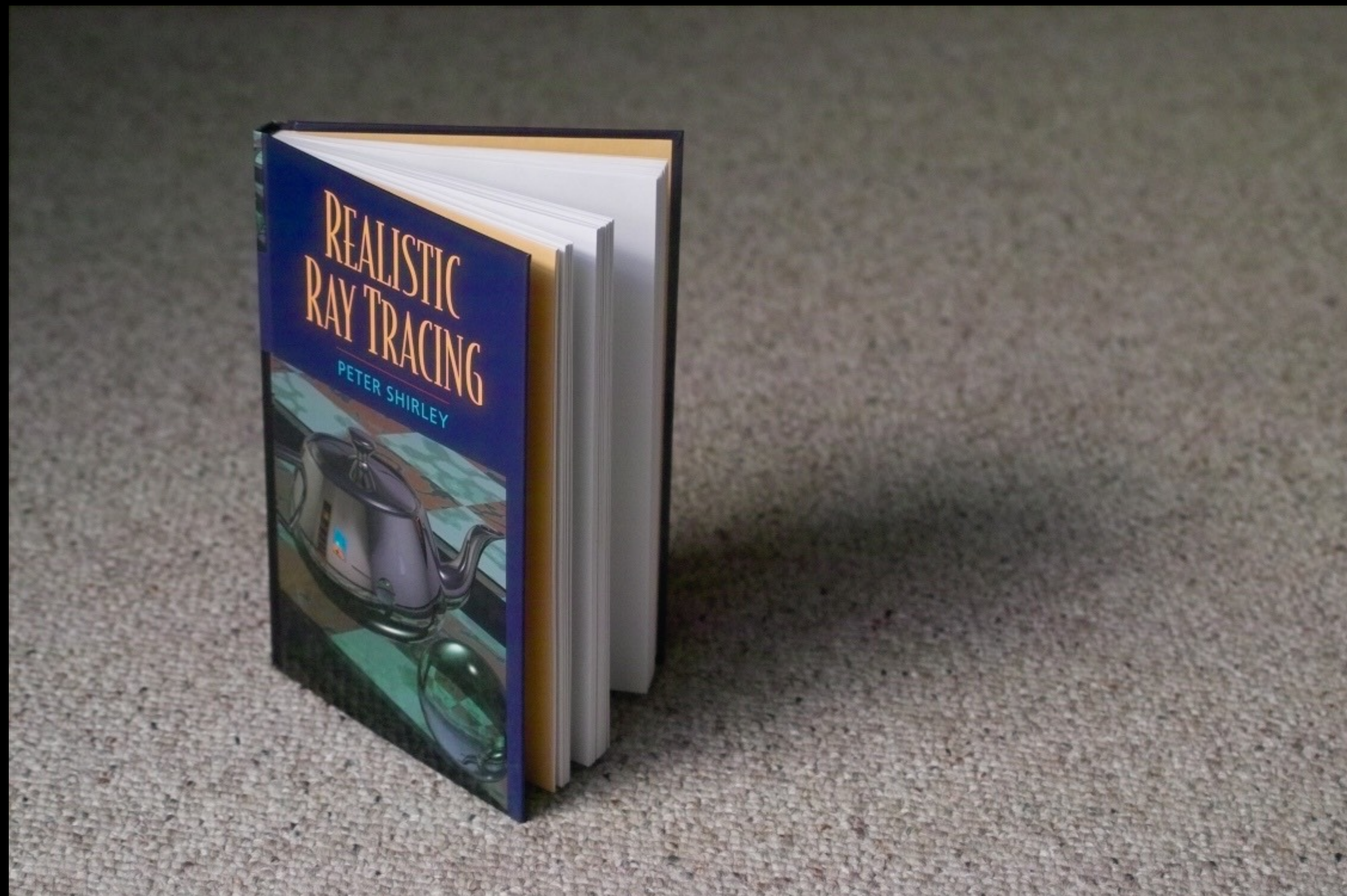
Basic ray traced image

[Glassner 89]

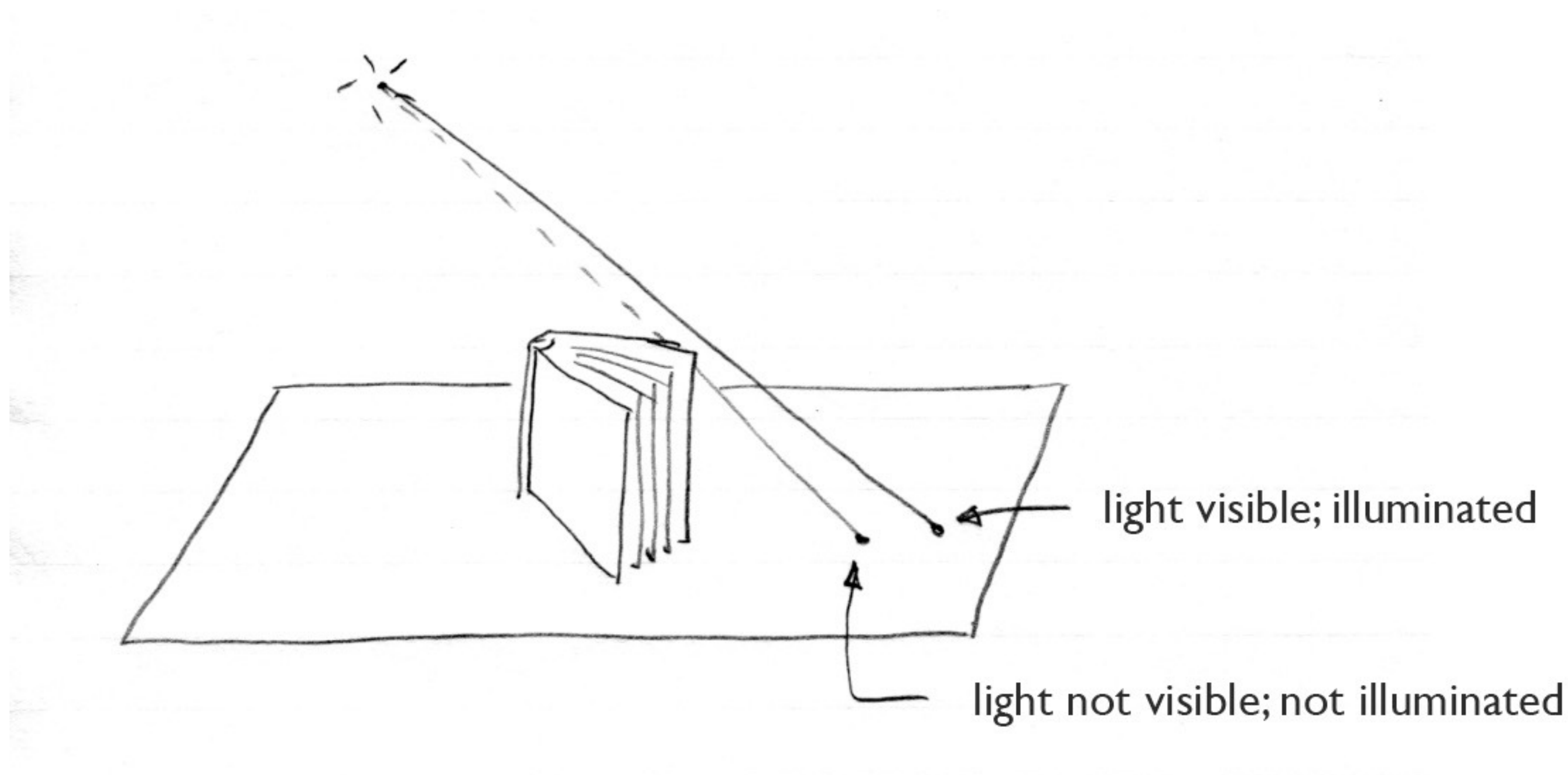
Discontinuities in basic RT

- Perfectly sharp object silhouettes in image
  - leads to aliasing problems (stair steps)
- Perfectly sharp shadow edges
  - everything looks like it’s in direct sun
- Perfectly clear mirror reflections
  - reflective surfaces are all highly polished
- Perfect focus at all distances
  - camera always has an infinitely tiny aperture
- Perfectly frozen instant in time (in animation)
  - motion is frozen as if by strobe light

Soft shadows

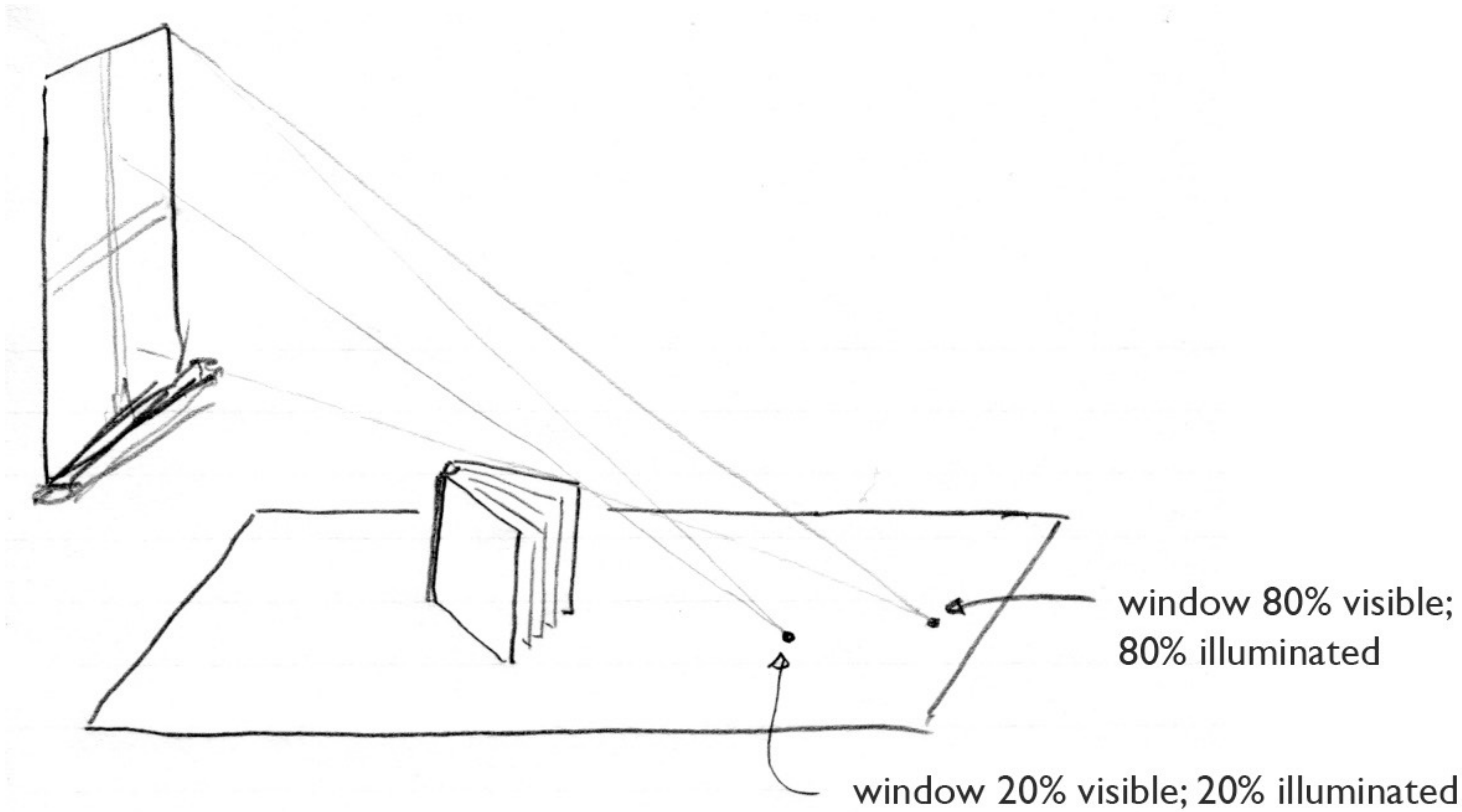


Cause of soft shadows



point lights cast hard shadows

Cause of soft shadows

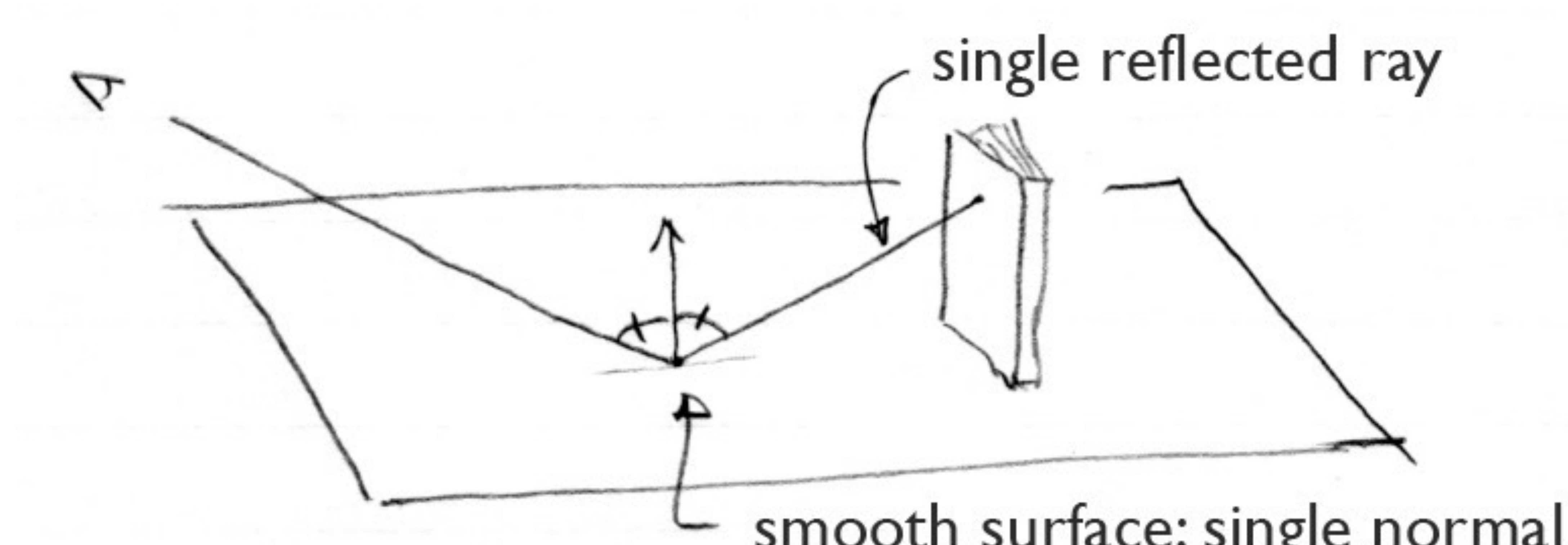


area lights cast soft shadows

Glossy reflection

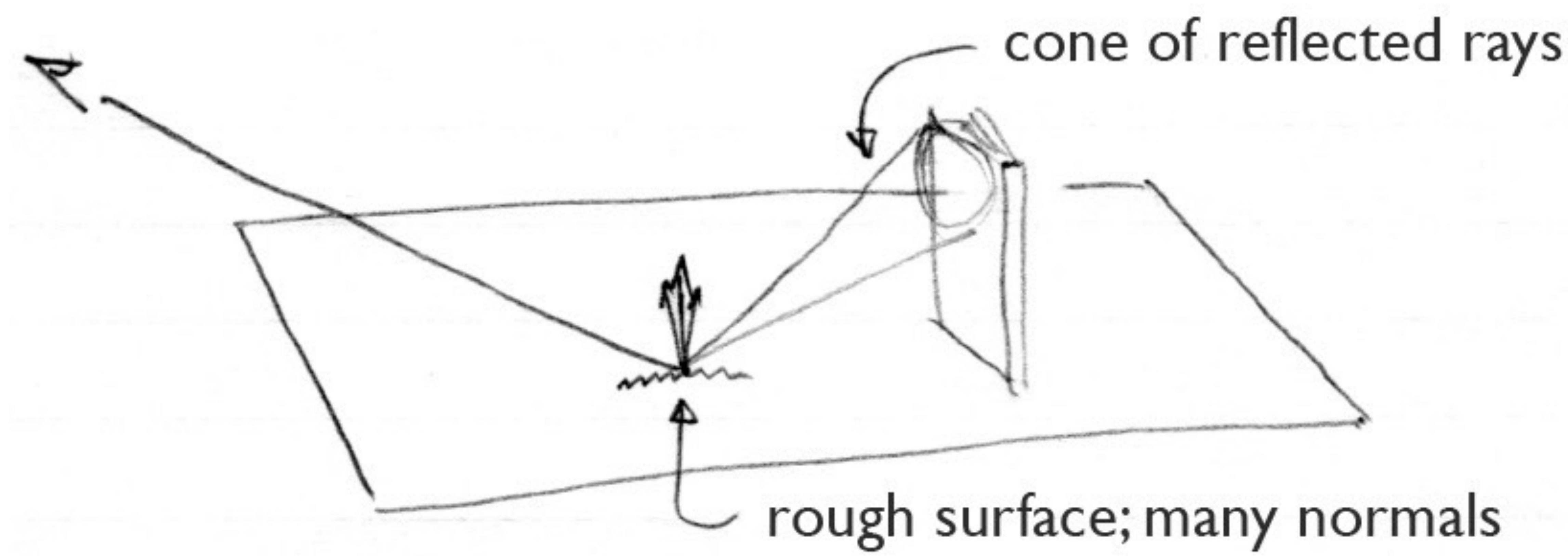
[Lafortune et al. 97]

Cause of glossy reflection



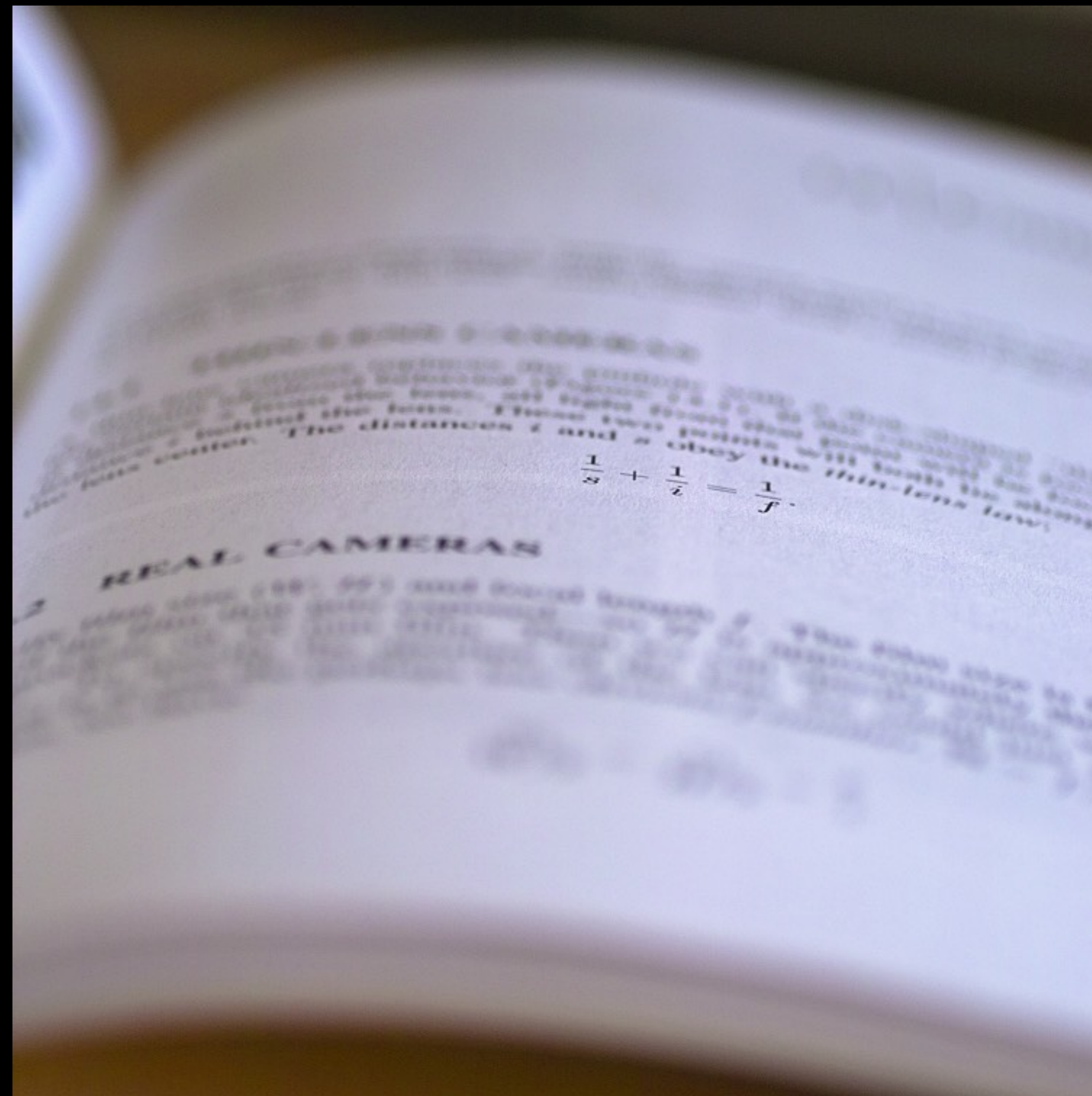
smooth surfaces produce sharp reflections

Cause of glossy reflection

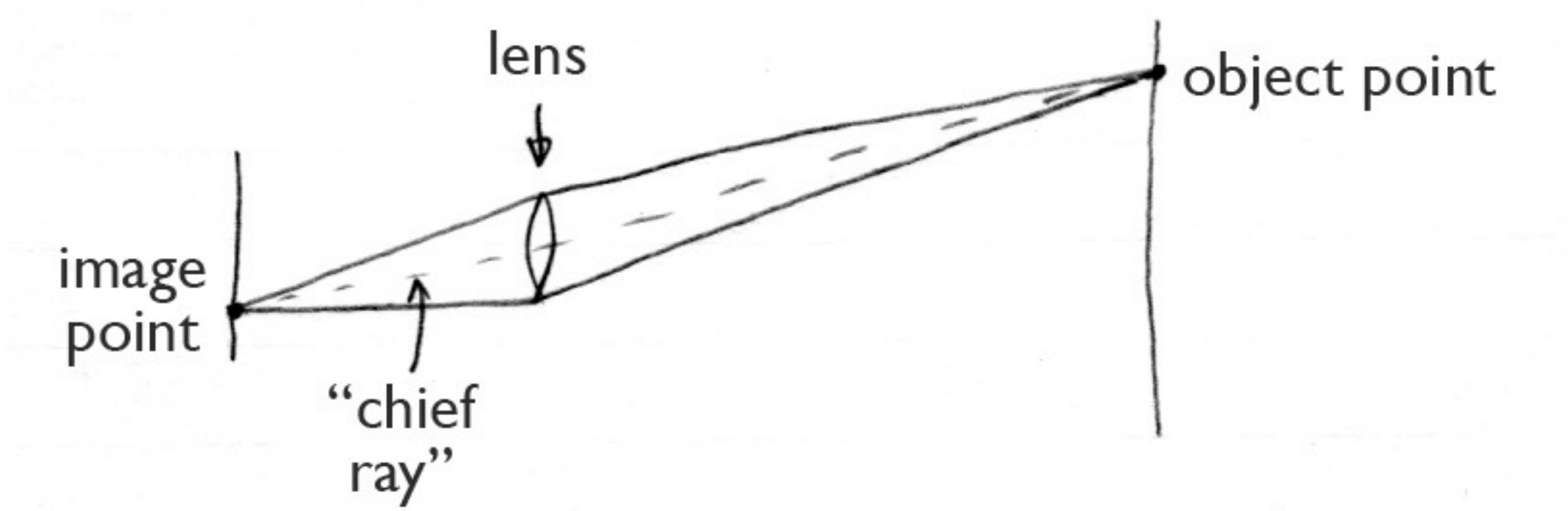


rough surfaces produce soft (glossy) reflections

Depth of field

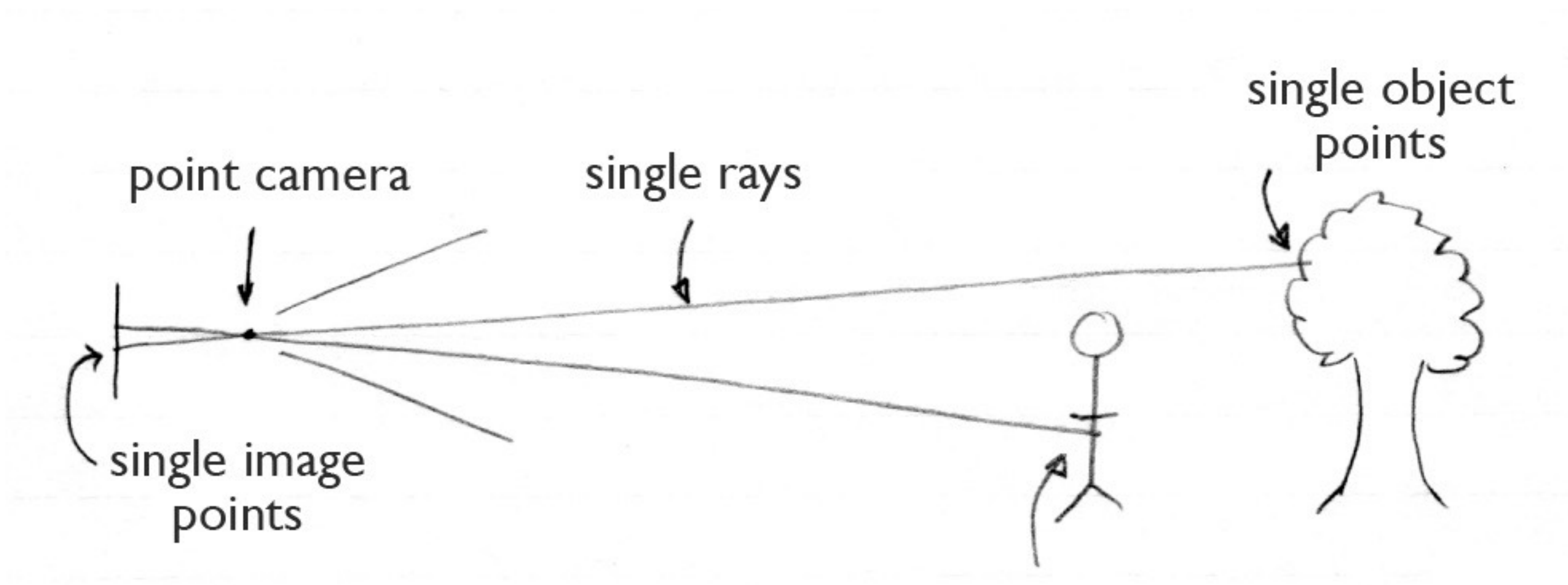


Cause of focusing effects



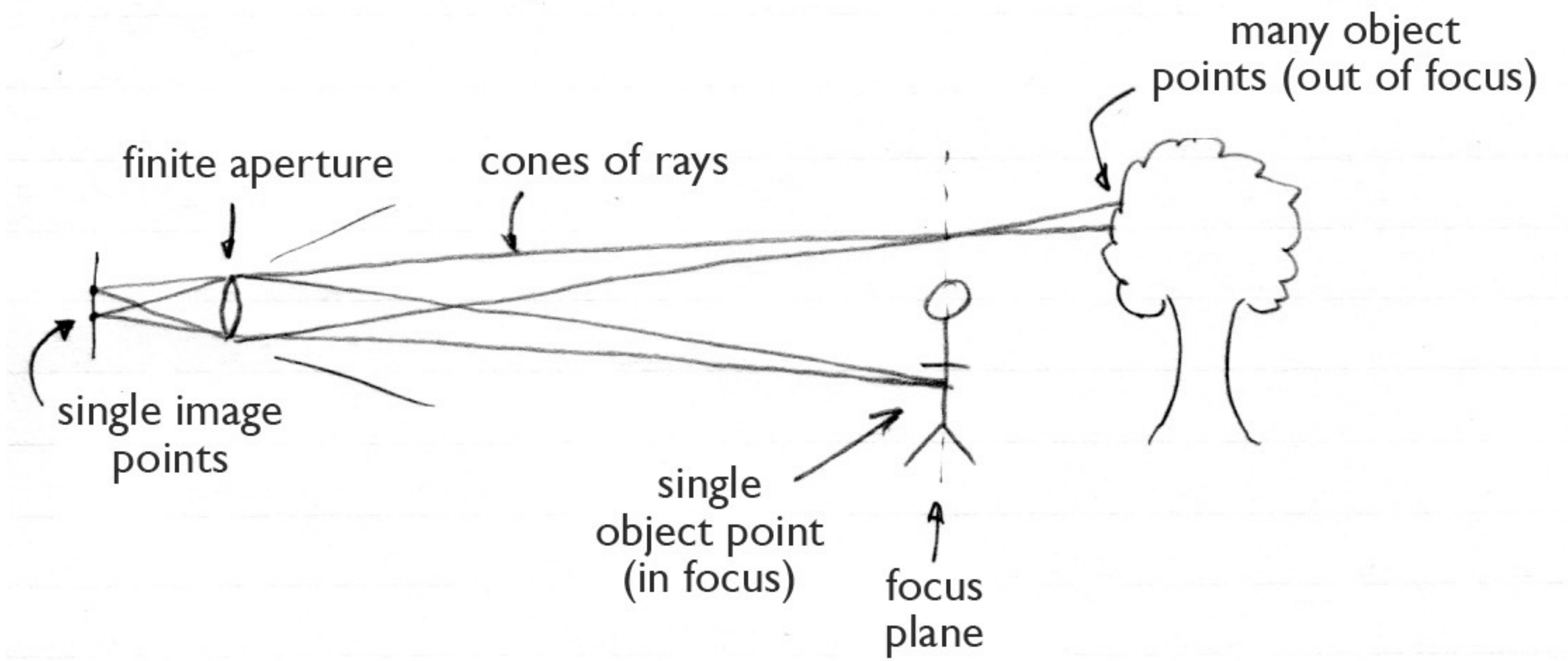
what lenses do (roughly)

Cause of focusing effects



point aperture produces always-sharp focus

Cause of focusing effects

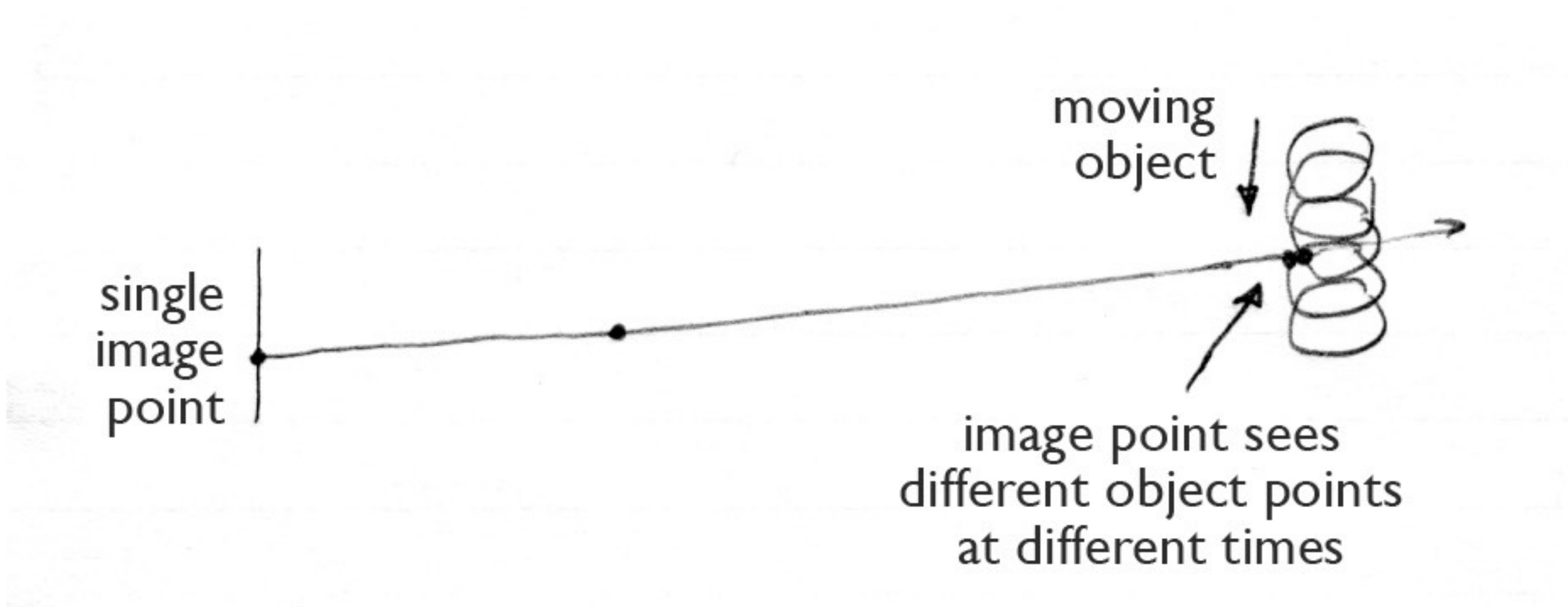


finite aperture produces limited depth of field

Motion blur

[Cook, Porter, Carpenter 1984]

Cause of motion blur



Pixar—*Monsters University* (2013)

Illumination as integration

- Surface reflection with Lambertian surfaces is the place to start
  - first way to do it: integrate over light source
  - 1. illumination from a point source

$$L_r = I \cdot k_d \cdot \frac{\cos \theta_r}{r^2}$$

  - 2. illumination from a small area source

$$L_r = L_s A \cdot k_d \cdot \frac{\cos \theta_r \cos \theta_s}{r^2}$$

  - 3. illumination from a bunch of small area sources

$$L_r = \sum_{i=1}^{n} L_s \cdot k_d \cdot \frac{\cos \theta_{r,i} \cos \theta_{s,i}}{r_i^2} \Delta A_i$$

  - 4. illumination from a large area source

$$L_r = \int_S L_s \cdot k_d \cdot \frac{\cos \theta_r(\mathbf{y}) \cos \theta_s(\mathbf{y})}{|\mathbf{x} - \mathbf{y}|^2} d\mathbf{y}$$

Area source by Monte Carlo

- Start with integral

$$L_r = \int_S L_s \cdot k_d \cdot \frac{\cos\theta_r(\mathbf{y})\cos\theta_s(\mathbf{y})}{|\mathbf{x}-\mathbf{y}|^2} d\mathbf{y}$$

  – choose a probability density on S: uniform

$$f(\mathbf{y}) = L_s \cdot k_d \cdot \frac{\cos\theta_r(\mathbf{y})\cos\theta_s(\mathbf{y})}{|\mathbf{x}-\mathbf{y}|^2}$$

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

$$g(\mathbf{y}) = \frac{f(\mathbf{y})}{p(\mathbf{y})} = L_s A \cdot k_d \cdot \frac{\cos\theta_r(\mathbf{y})\cos\theta_s(\mathbf{y})}{|\mathbf{x}-\mathbf{y}|^2}$$

Area source by Monte Carlo

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


Illumination as integration

- Surface reflection with arbitrary BRDF
  - first way to do it: integrate over light 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

$$\mathbf{w}(\mathbf{y}) = \frac{\mathbf{y} - \mathbf{x}}{\|\mathbf{y} - \mathbf{x}\|}$$

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

Area source by Monte Carlo

- 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

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

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

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

Area source by Monte Carlo

```
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;
}
```


Illumination as integration

- Surface reflection with Lambertian surfaces is the place to start
  - second way to do it: integrate over incoming directions
  - 1. illumination from a directional source

$$L_r = I \cdot k_d \cdot \cos\theta$$

  - 2. illumination from a small area at infinity (a small solid angle)

$$L_r = L_i \sigma \cdot k_d \cdot \cos\theta$$

  - 3. illumination from a bunch of small solid angles

$$L_r = \sum_i L_i \cdot k_d \cdot \cos\theta_i \, \Delta\sigma_i$$

  - 4. illumination from an entire environment

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

Diffuse surface in env. by Monte Carlo

- 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

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

  - why π? Cosine-distributed points on the hemisphere cast uniformly distributed shadows on the unit circle.

Diffuse surface in env. by Monte Carlo

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


Illumination as integration

- Surface reflection with arbitrary BRDF
  - second way to do it: integrate over incoming directions
  - 1. illumination from a directional source

$$L_r = I \cdot f_r(\mathbf{v}, \mathbf{w}) \cdot \cos\theta$$

  - 2. illumination from a small area at infinity (a small solid angle)

$$L_r = L_i \sigma \cdot f_r(\mathbf{v}, \mathbf{w}) \cdot \cos\theta$$

  - 3. illumination from a bunch of small solid angles

$$L_r = \sum_i L_i \cdot f_r(\mathbf{v}, \mathbf{w}) \cdot \cos\theta_i \, \Delta\sigma_i$$

  - 4. illumination from an entire environment

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

Arbitrary surface in env. by Monte Carlo

- 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 cos theta

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

  - another choice: proportional to environment brightness

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

  - why π? Cosine-distributed points on the hemisphere cast uniformly distributed shadows on the unit circle.

Diffuse surface in env. by Monte Carlo

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


The Blue Umbrella

- Recent Pixar short
- Made partly to showcase new more photorealistic rendering
  - much of it based on the ideas in this lecture

worth a look:
http://rainycitytales332.tumblr.com