Realistic Image Synthesis

- MIS and Path Tracing -

Philipp Slusallek

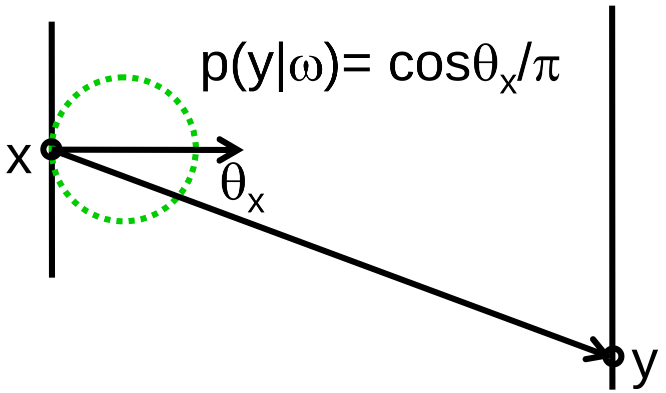
Karol Myszkowski

Gurprit Singh

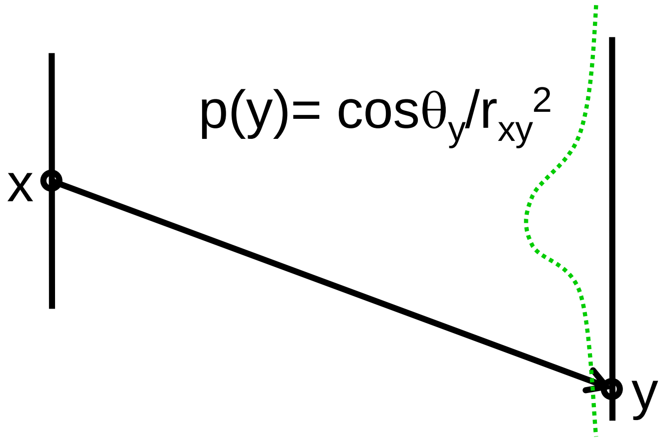
MULTIPLE IMPORTANCE SAMPLING (MIS)

Intelligent Monte Carlo Integration

- **Example: Different Probabilities**
  - Sampling directions



  - Sampling the surface



Intelligent Monte Carlo Integration

- **Multiple Importance Sampling (MIS) – Very Important**
  - Combining multiple importance distributions
    - Idea: One function $p(x)$ is too inflexible
    - Use multiple functions in parallel
- **A-priori weighted integration**
  - Weight two or more estimators
  - Weights are determined analytically or are estimated (manually)

$$I = \sum_{m=1}^{M} \frac{w_m}{N_m} \sum_{i=1}^{N_m} \frac{f(\xi_i)}{p_m(\xi_i)}$$

  - Approach with two estimators and weights ω_i $(\sum \omega_i = 1)$

$$V[w_1 S_1 + w_2 S_2] = w_1^2 V[S_1] + w_2^2 V[S_2] + 2 w_1 w_2 Cov[S_1, S_2]$$

$$Cov[S_1, S_2] = E[S_1 \cdot S_2] - E[S_1] E[S_2] \qquad \text{(zero if independent)}$$

$$\Rightarrow \frac{w_1}{w_2} = \frac{V[S_2] - Cov[S_1, S_2]}{V[S_1] - Cov[S_1, S_2]}$$

  - Weight inversely proportional to variance (similar for more estimators)

Intelligent Monte Carlo Integration

- **A-posteriori multiple importance sampling**
  - Choose samples first
  - Assign weights according to probabilities/variance of each estimator

$$I = \frac{1}{N} \sum_{m=1}^{M} \sum_{i=1}^{N} w_m \frac{f(\xi_i)}{p_m(\xi_i)} \quad \text{with} \quad \sum_{\mathrm{m=1}}^{\mathrm{M}} w_m = 1$$

- **Balance Heuristics**

$$w_i(x) = \frac{p_i(x)}{\sum p_j(x)} \qquad\qquad w_i(x) = \frac{n_i p_i(x)}{\sum n_j p_j(x)}$$

  - No other combination can be much better [Veach '97]
  - Motivation
    - Samples with low probability boost the variance with $1/p_i$
    - Assign larger weights to samples with higher probability
  - Must be able to evaluate probability of sample according to other probabilities densities

Intelligent Monte Carlo Integration

- **Other weighting heuristics**
  - Variance is additive – may have impact on already good estimators
  - Try to sharpen the weighting, avoid contribution with low probability
- **Power Heuristic and Cutoff Heuristic**

$$w_i = \frac{p_i^\beta}{\sum_k p_k^\beta}$$

$$w_i = \begin{cases} 0 & \text{if } p_i < \alpha p_{\max} \\ \dfrac{p_i}{\sum_k \{p_k \mid p_k \geq \alpha p_{\max}\}} & \text{otherwise} \end{cases}$$

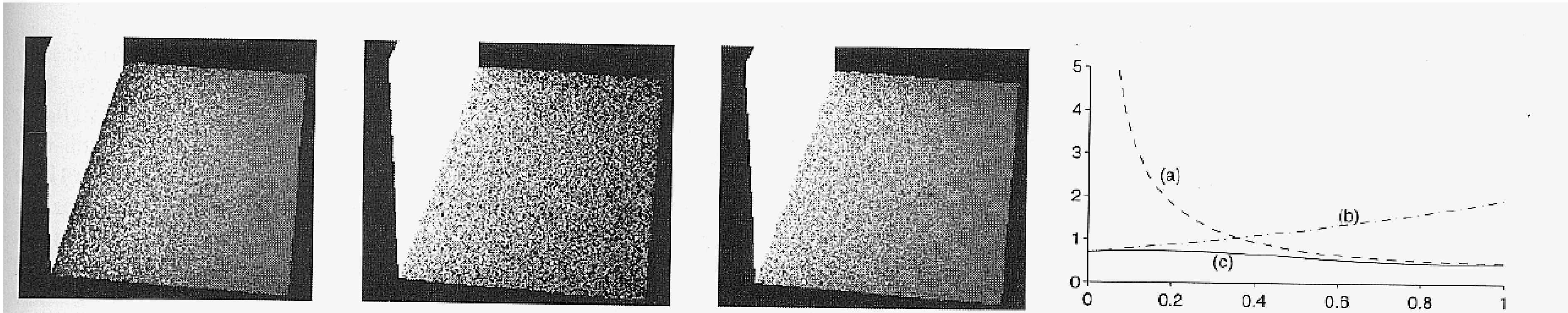
  - Reduced weight for samples with low probability
- **Maximum Heuristic**

$$w_i = \begin{cases} 1 & \text{if } p_i \text{ is maximum} \\ 0 & \text{otherwise} \end{cases}$$

  - Adaptively partitions the integration domain according to $p_i(x)$
  - But typically too many samples are thrown away to be effective

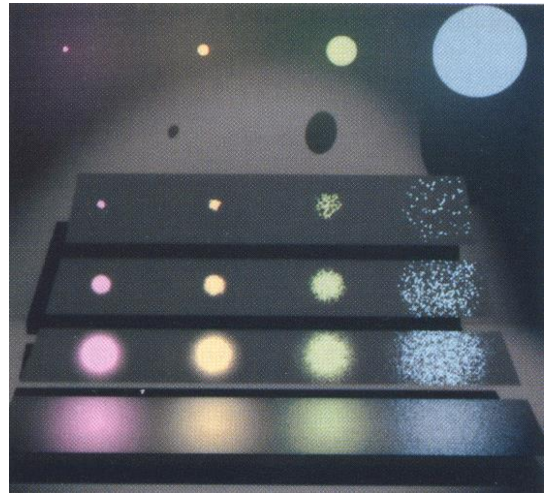
Comparison of Heuristics

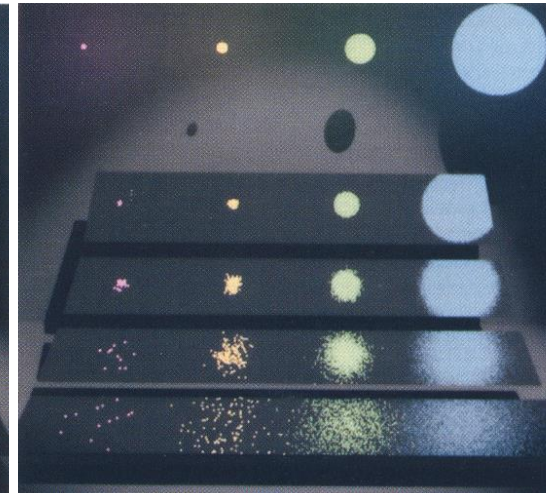
- **Example**
  - Two orthogonal surfaces, one is a light source
    - (a) Sampling of light source (3 samples per pixel)
      - Most samples near light will have very shallow angle, cos near zero
    - (b) Sampling of directions (according to projected solid angle)
      - Most samples far from light will not hit the light source
    - (c) MIS with Power Heuristics
    - (d) Standard deviation plotted over average distance to light source



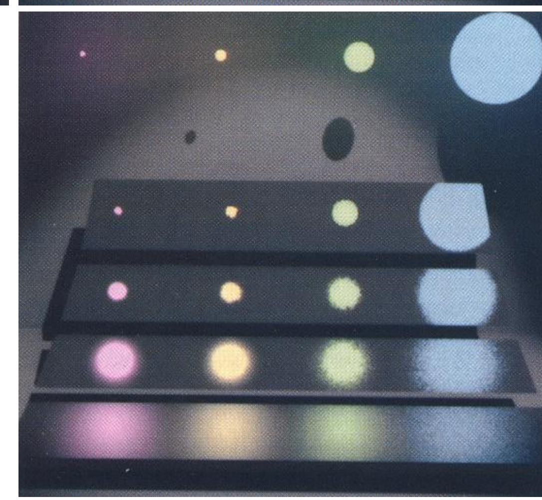
(a) (b) (c) (d)

Combination of Estimators





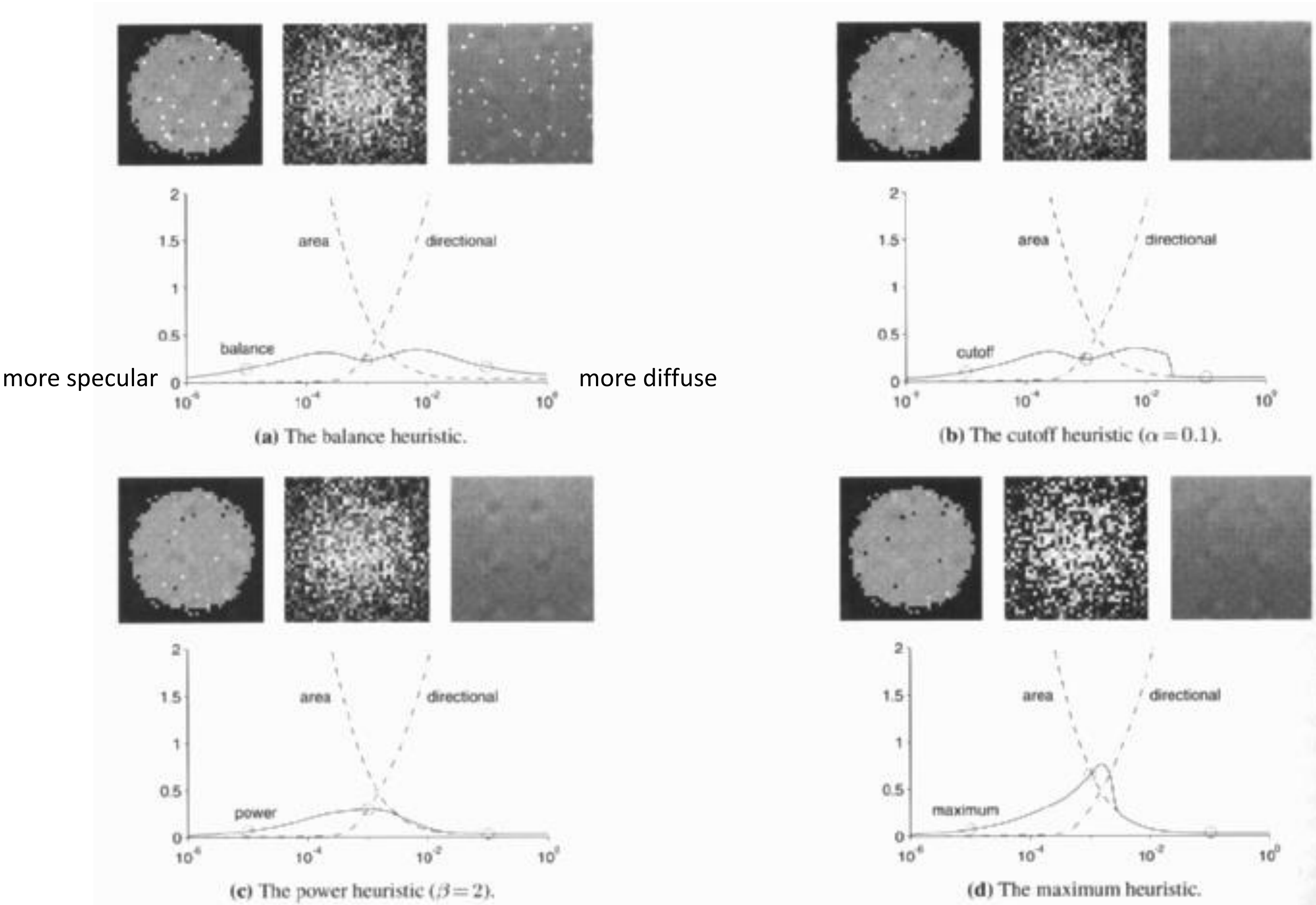
- **Sampling of Light Sources (l)**
  - Small contribution for large light sources and highly specular surfaces
- **Sampling of Directions (r)**
  - No contribution if light source is not hit (highly diffuse, small LS)



- **Ideal: Weighted combination (b)**
  - Combined advantages of both methods
  - Principle: High weight, for high probability
  - Here: Power heuristics

Comparison of Heuristics

– Image of a light source on surfaces with different roughness



(a) The balance heuristic.

(b) The cutoff heuristic ($\alpha = 0.1$).

(c) The power heuristic ($\beta = 2$).

(d) The maximum heuristic.

PATH TRACING

Rendering Equation

- **Rendering Equation in Operator Notation**
  - Short form – leaving out arguments
  - To be applied to the entire domain, all possible $(x, \omega) \in S \times \Omega_+$
    - $L = L_e + \int_{\Omega_+} f_r L(y(x, \omega_i), -\omega_i) cos\theta_i d\omega_i$ with ray tracing op. $y(x, \omega_i)$
    - $L = L_e + \boldsymbol{T}L$ with $\boldsymbol{T}X = (\boldsymbol{T}X)(x, \omega) = \int_{\Omega_+} f_r X(y(x, \omega_i), -\omega_i) cos\theta_i d\omega_i$
    - $L = (1 - \boldsymbol{T})^{-1} L_e$ (formally derived "solution")
  - Definition:
    - T is the “Transport operator”: Gathers light from all visible surfaces
- **Inversion of $(1 - \boldsymbol{T})$**
  - Cannot be done in closed form (except for trivial solutions)
    - Infinite dimensional integral
  - Can be approximated by mapping to finite dimensional space
    - Results in a linear system of equation
    - Finite Element Methods, e.g. Radiosity Methods
    - Can be nicely evaluated numerically by Monte-Carlo methods

Rendering Equation

- **Expansion of the Rendering Equation (Neumann series)**
  - $L = L_e + \int_{\Omega_+} f_r L(y(x,\omega_i), -\omega_i) cos\theta_i d\omega_i$
  - $L = L_e + \boldsymbol{T}L = L_e + \boldsymbol{T}(L_e + \boldsymbol{T}L) = L_e + \boldsymbol{T}L_e + \boldsymbol{T}\boldsymbol{T}L_e + \cdots$
  - $L = \sum_{i=0}^{\infty} \boldsymbol{T}^i L_e \quad \text{with} \quad \|\boldsymbol{T}\| < 1$ (energy conservation (at most))
- **Interpretation**
  - $i = 0$: Direct emission from light sources
  - $i = 1$: Light reflected once
  - $i = n$: Light reflected n times
- **General MC Rendering Algorithm (incl. Path Tracing)**
  - Select points and directions and shoot ray
  - At hit point:
    - Add emission term (when having reached light source!)
    - Select new direction and recursively shoot rays
    - Add contribution after attenuation by BRDF
- **But: When to stop? How many samples to take?**

Russian Roulette

- **Unbiased Termination of Infinite Sequence**
  - Abort sequence with a certain probability α
  - Need to correct for the missed contribution

$$F_n' = \begin{cases} 0 & \xi \leq \alpha \\ F_n(x)/(1-\alpha) & \text{else} \end{cases}$$

$$E[F_n'] = E\left[0 \cdot \alpha + \frac{F_n}{(1-\alpha)}(1-\alpha)\right] = E[F_n]$$

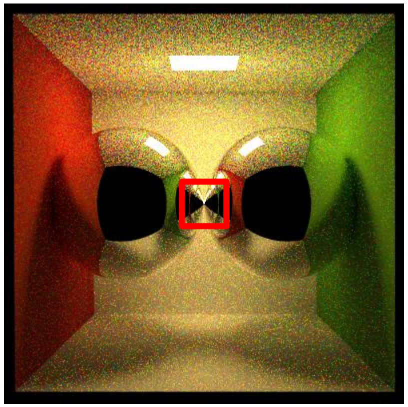
  - In rendering, often choose (1 – alpha) to be:
    - Constant
    - The albedo (avg. reflectivity): probability that a photon is reflected at all
    - Path throughput: Contribution to final pixel (possibly relative contribution)
    - Efficiency-optimized: Threshold based on avg. variance & avg. ray count over neighboring pixels
  - Conclusion
    - Adds variance/noise but is unavoidable for an unbiased solution

Russian Roulette

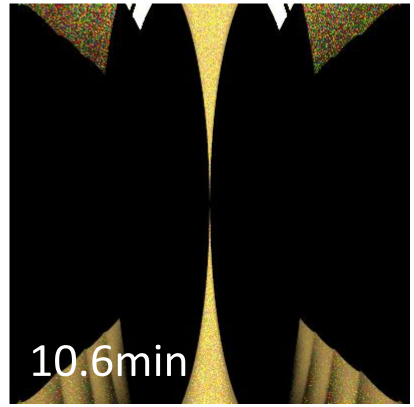
- **Experiments by Thiago Ize, University of Utah**
  - Effects of Russian Roulette
  - 5000 rays per pixel; perfect reflection, with highly occluded areas
- **Four comparisons**
  1. Fixed max. depth for rays (bias depends on max. depth and scene)
     - Strong bias in significantly occluded areas as rays are terminated before hitting a light source. Need very high max. depth, which is costly
  2. RR with fixed kill probability
     - Introduction of speckle noise due to occasional strong boosting of rays
     - 10x bounce with 50% chance: 2^10=1024x
  3. RR with kill probability proportional to *importance* (throughput) of ray
     - Pure importance sampling, should give good results
  4. "Efficiency-optimized RR" [Veach PhD thesis, Chapter10]
     - Estimating kill probability based on statistics of surrounding pixels
- **Results**
  - Strategy (3) slightly less efficient than (4), but easier to implement

Russian Roulette Experiments

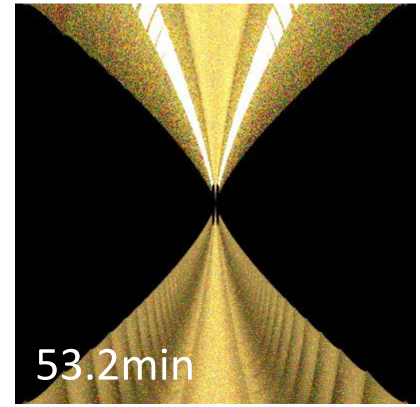
input scene



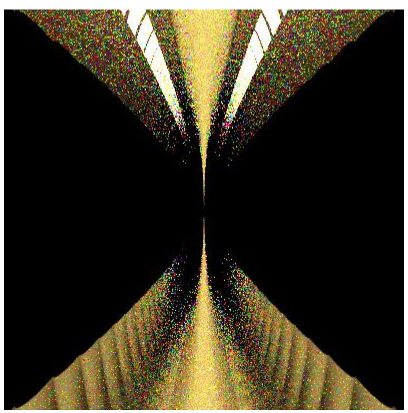
path depth < 11



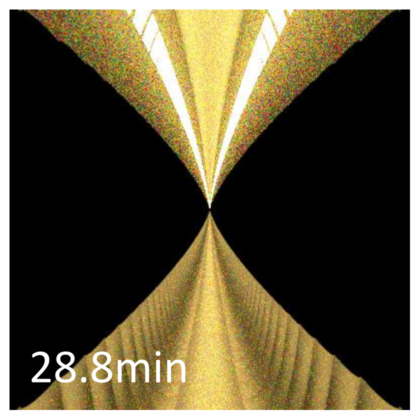
path depth < 101



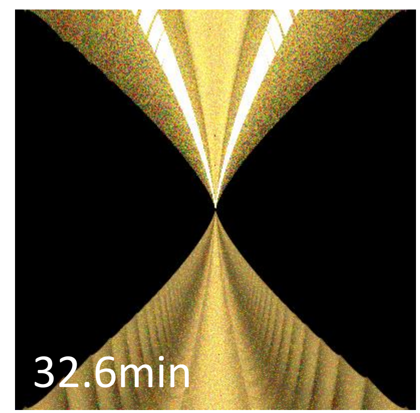
RR with p=0.3



efficiency optimized



p = throughput / 0.01



- **Thiago Ize (University of Utah, currently Solid Angle)**
  - http://www.cs.utah.edu/~thiago/cs7650/hw12/

Measuring Equation

- **Rendering equation is a continuous density function**
  - Provides radiance [Watt per area and solid angle]
- **Sensors measure finite values (energy or power)**
  - Energy falling on a pixel, patch irradiance, …

► **Measure the continuous function over a finite domain**
  - Choose initial samples according to Measurement Equation

- **Measuring Equation**
  - With sensor's sensitivity function M(...)
  - Measuring pixel values (energy on the film of a camera)

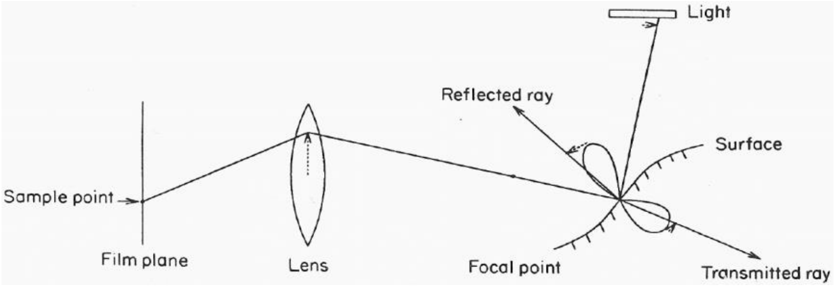
$$M_i = \int_{\text{shutter opening}} \int_{\text{pixel}} \int_{\text{lens aperture}} M(x,\omega,t) L(x,\omega,t) d\omega_i dx dt = \mathbf{M} L$$

  - Measuring flux/power on a surface patch

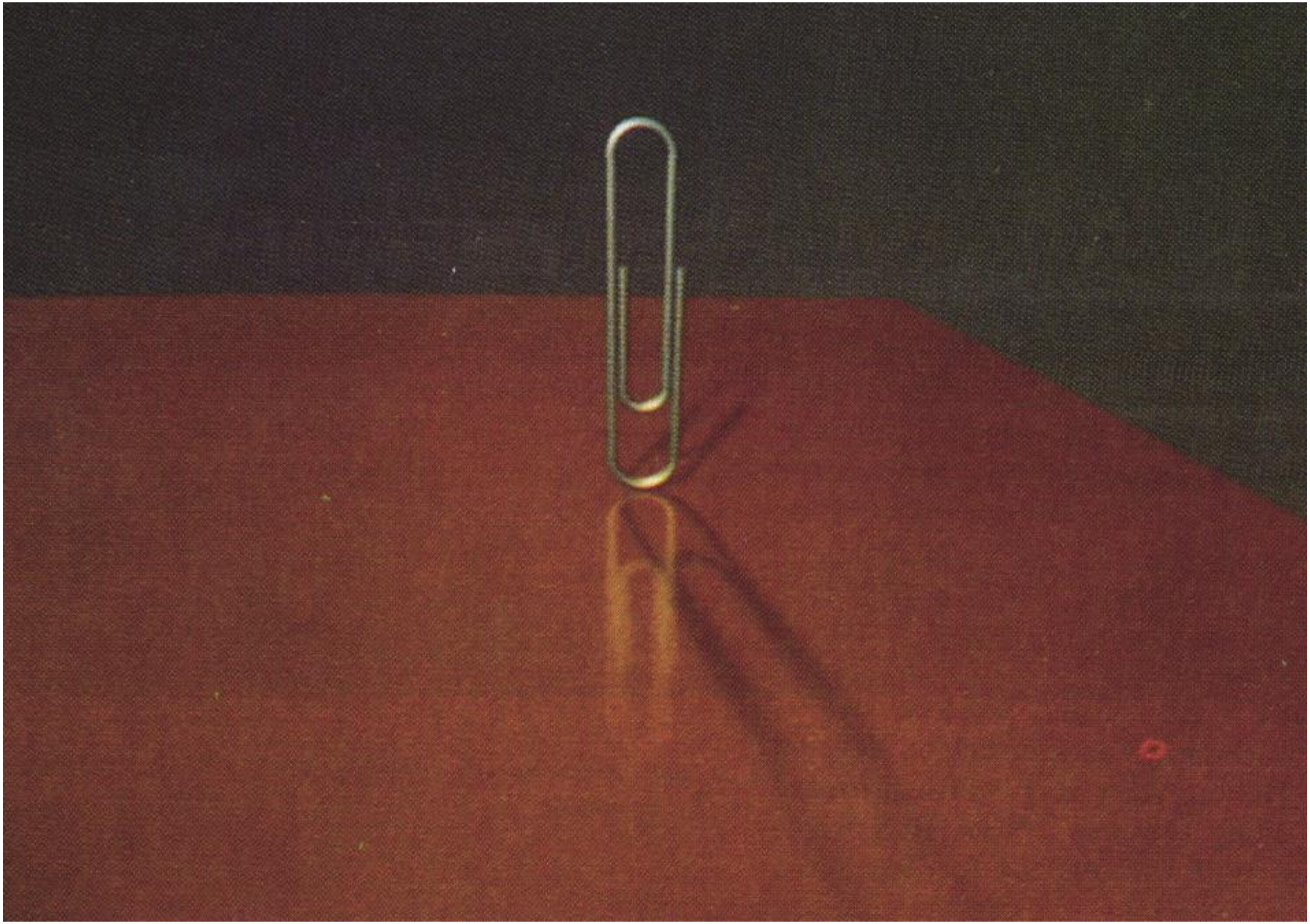
$$M_i = \int_{\text{patch area}} \int_{\Omega_+} M(x,\omega,t) L(x,\omega,t) d\omega_i dx dt = \mathbf{M} L$$

Distribution Ray Tracing

- **Question: How many sample to take?**
- **Was called „*Distributed* Ray-Tracing“ [Cook´84]**
  - Gathering approach
- **Integration over pixel (anti-aliasing)**
  - Measuring device collects photons
  - Here: Sampling with many ray paths
- **Real camera with aperture (depth of field)**
  - Sample over lens aperture and according to optical properties
- **Finite shutter opening (motion blur)**
  - Sample over opening time, consider moving camera and objects
- **Glossy reflections (highlights)**
  - Sample glossy parts of the BRDF
- **Real light sources (area lights)**
  - Sample light sources

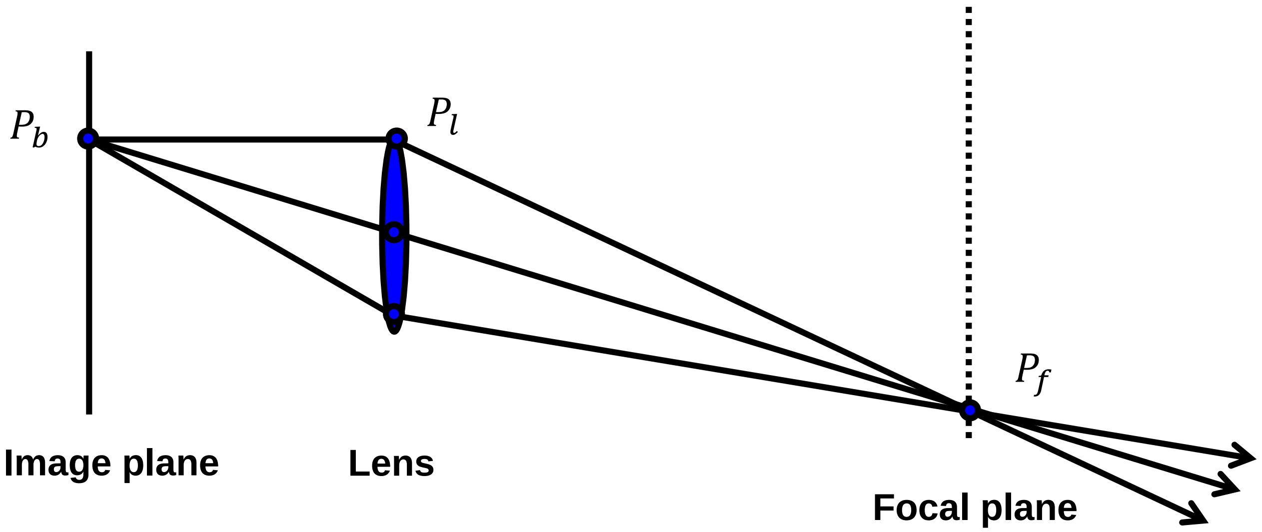


Glossy Reflection

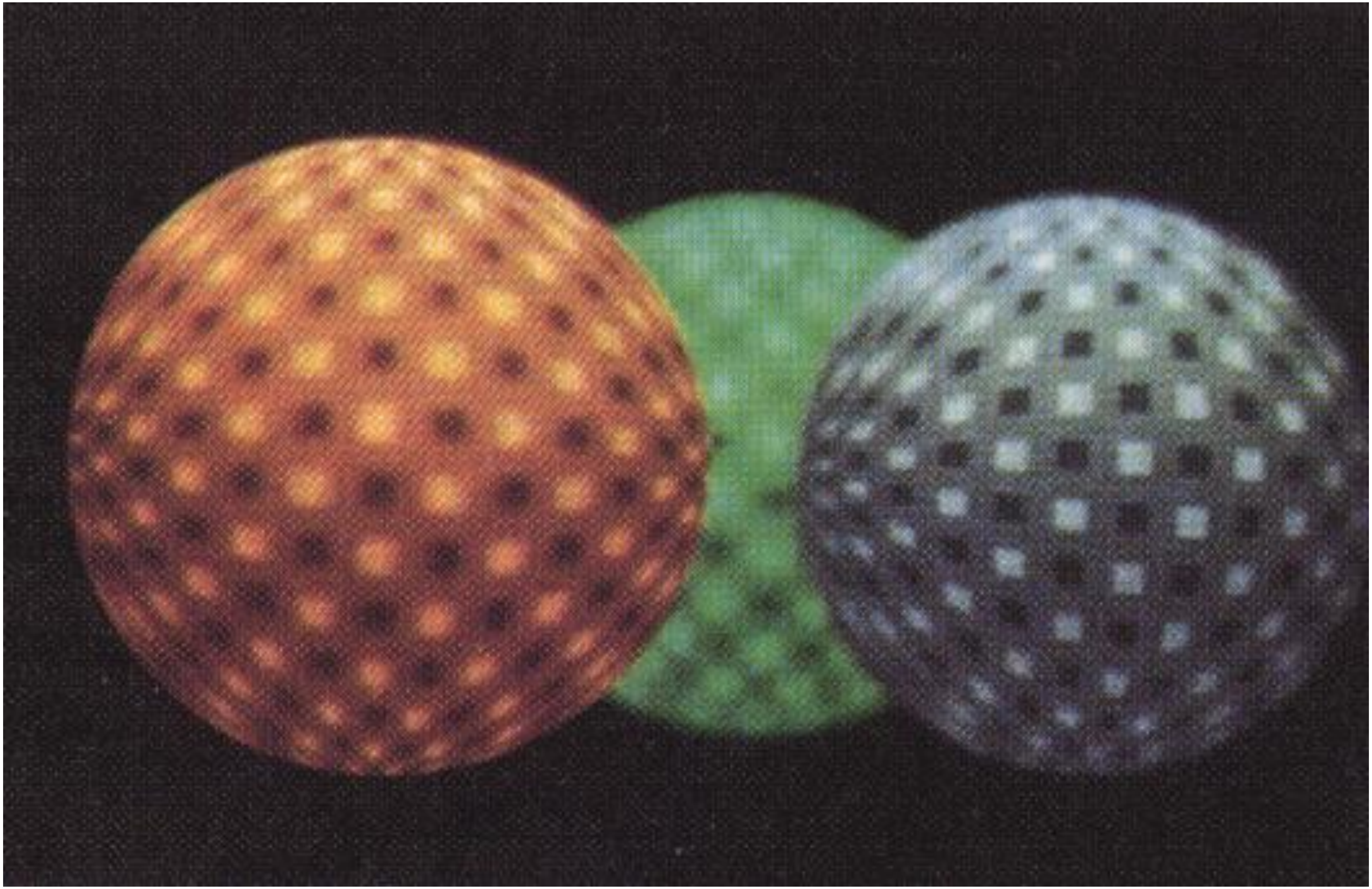


Depth of Field

- **Thin lens model**
  - Unique mapping of point on image plane to points on focal plane
  - Determined with straight ray through center of the lens
- **Sampling the lens aperture**
  - Choose point P_b on image plane and P_l on lens
  - Compute point P_f by shooting ray through the lens center
  - Sample scene with ray from P_l through P_f



Depth of Field



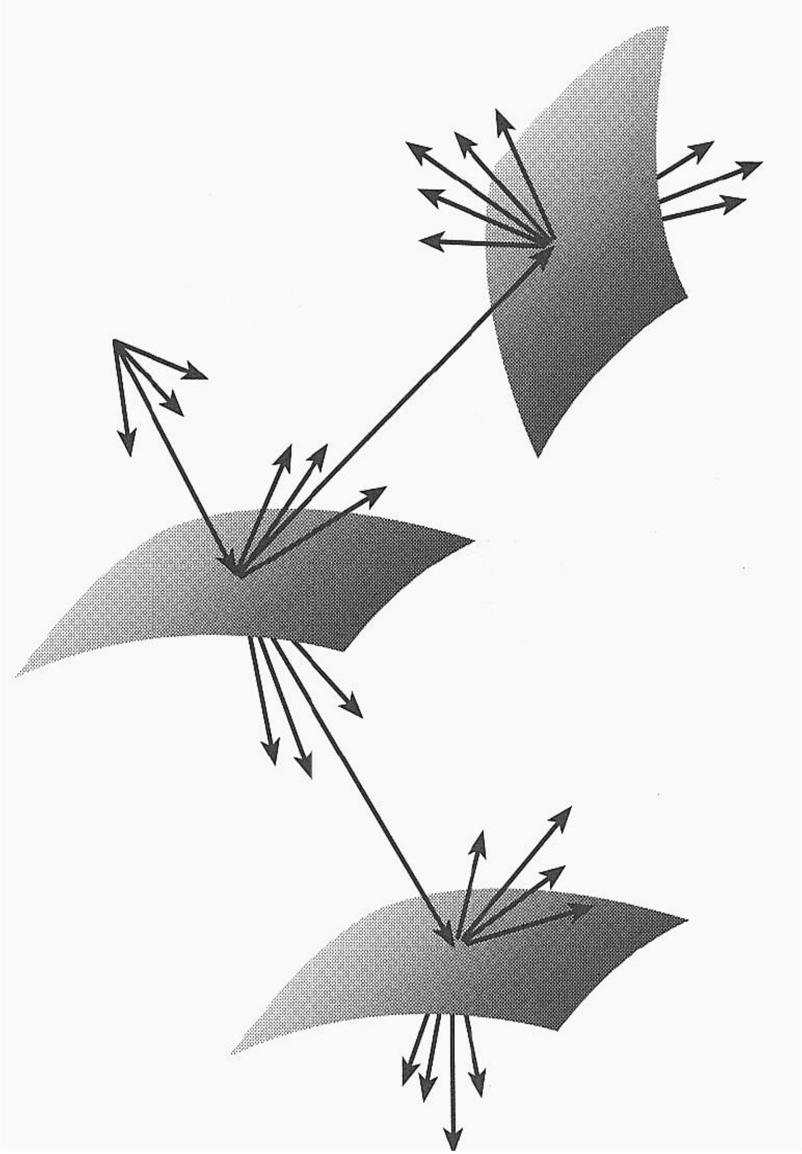
Motion Blur

- **Models Finite Exposure Time**
  - Shutter opening time ($t_0 \leq t \leq t_1$)
  - Assumes instantaneous opening and closing
    - Can easily be generalized by modeling the shape of the aperture at each time instance
- **Algorithm**
  - Assign ray a time t between t_0 and t_1
  - Transform objects in the scene to the positions at t
    - Alternately: Inversely transform ray
    - Camera might move as well
  - Compute intersection with object

Motion Blur

Distribution Ray Tracing

- **Fundamental Principle**
  - Monte Carlo Integration
    - But not formulated as such (yet)
  - Only point-wise evaluation of all integrals
    - BRDF, emission, and reflected light
  - No use of importance sampling or filtering (yet)
- **Problems**
  - Combinatoric explosion of additional rays with depth
  - Deeper rays contribute less
  - Maximum damage:
    - Do ***more work*** for ***less value***
    - We clearly need a better solution!

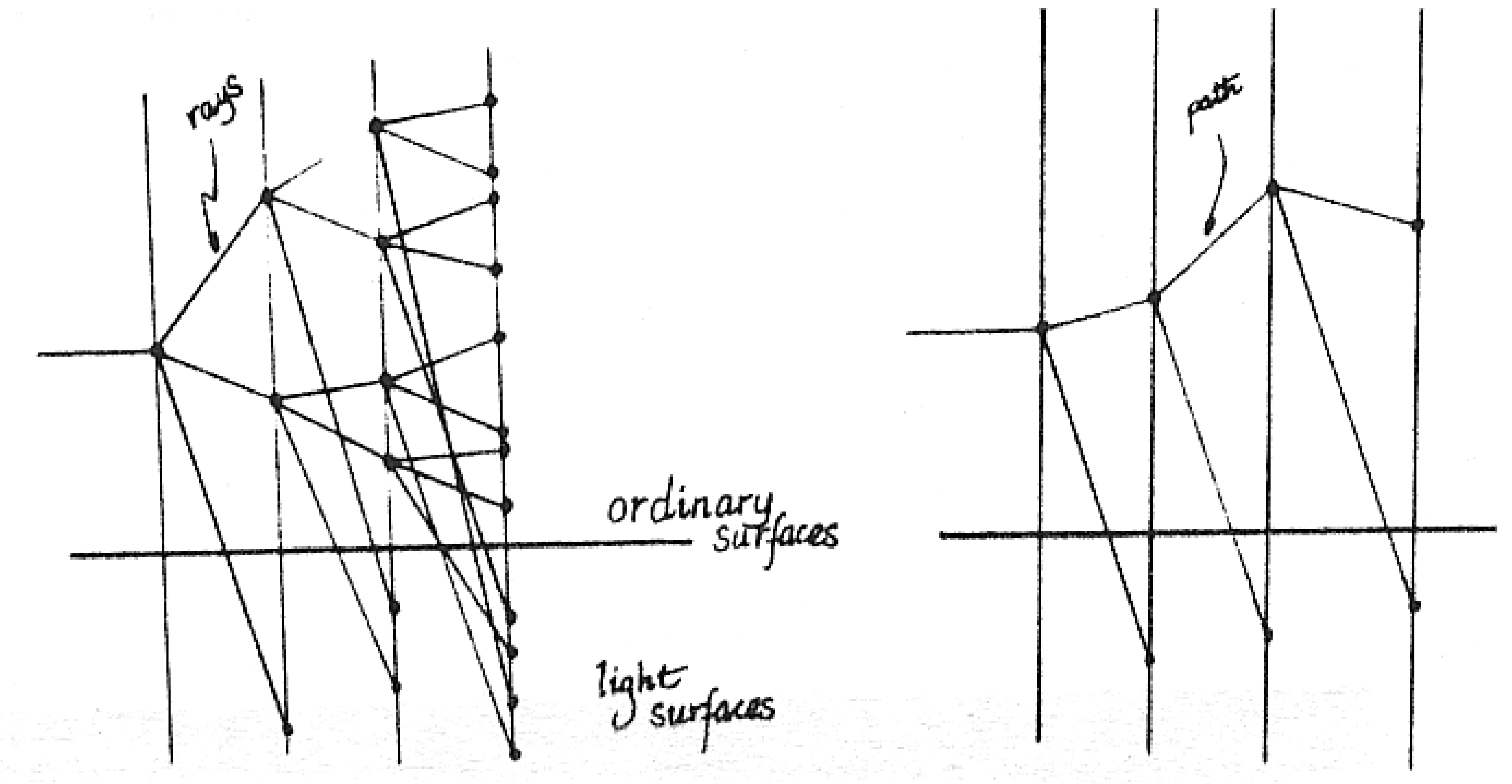


Path Tracing [Kajiya´86]

- **Path Tracing: Trace *only a single ray* per hit point**
  - Randomly decide to absorb (Russian Roulette)
  - Randomly decide which reflection term to sample (e.g. diffuse, glossy)
  - Randomly sample this term recursively
- **Would still be very slow**
  - Very low probability to hit the light source

- **Definition: Next Event Estimator (Light Source Sampling)**
  - At every hit point: Try to gather some energy from light sources
    - Randomly choose a light source
    - Randomly choose a position on the light source
    - Trace a “shadow ray” to this position and add any contribution
    - Essentially this is a form of bidirectional path tracing (see later)

Comparison



(figure by Kajiya)

Path Tracing

- **„Next Event Estimation“ is Stratification**
  - Split scene into separate strata
    - Light sources
    - Non light sources
- **Use different sampling strategies**
  - Light sources: Directed sampling on light's surfaces
    - Select a light source (e.g. importance sampling based on its total power)
    - Select a sample point on its surface (e.g. uniformly distributed)
  - Non light sources: Directional sampling
    - Chose (e.g. cosine or BRDF weighted) direction
- **Beware:**
  - What happens if a directional sample hits a light source ????

Path Tracing

- **„Next Event Estimation“ is Stratification**
  - Split scene into separate strata
    - Light sources
    - Non light sources
- **Use different sampling strategies**
  - Light sources: Directed sampling on light's surfaces
    - Select a light source (e.g. importance sampling based on its total power)
    - Select a sample point on its surface (e.g. uniformly distributed)
  - Non light sources: Directional sampling
    - Chose (e.g. cosine or BRDF weighted) direction
- **Beware:**
  - What happens if a directional sample hits a light source ????
  - **IT MUST NOT BE COUNTED !!!!**
    - Otherwise, we would count light sources twice (with some probability)

Summary: Random Walk Methods

- **Gathering Solution (Ray/Path Tracing Methods)**
  - Start at the measuring device
  - Propagate path according to measurement function and BRDFs
  - Measure
    - Only at light sources
    - By connecting from hit points to light sources
      - (Only at end of path)
      - At every hit point
- **Shooting Solution (Photon/Light Tracing Methods)**
  - Start at the lights, choose power per sample
  - Propagate light according to emission functions and BRDFs
  - Measure
    - Only when “photons” hit the measurement device
    - By connecting from hit point to measurement device
      - (Only at end of path when photon would be “absorbed“)
      - At every hit point