CT5202: Photorealistic Rendering

Sampling

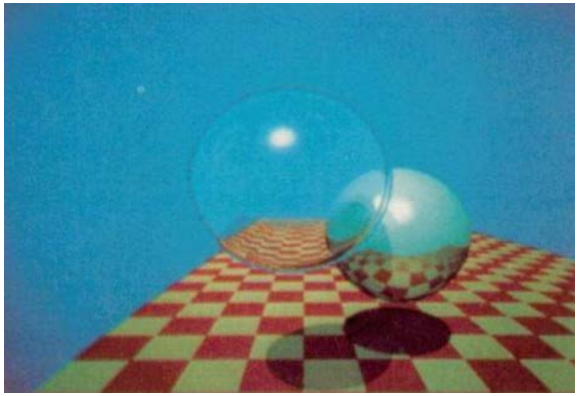
BOCHANG MOON

Sampling

- In the previous lectures,
  - We studied light transport & MC integration
- To solve the MC integration, we need draw random samples. This process is called sampling.
- Why should we do sampling in rendering?

Review: Distributed Ray Tracing

- Motivation
  - The classical ray tracing produces very clean images (look fake)
    - Perfect focus
    - Perfect reflections
    - Sharp shadows



[Whitted 1980]

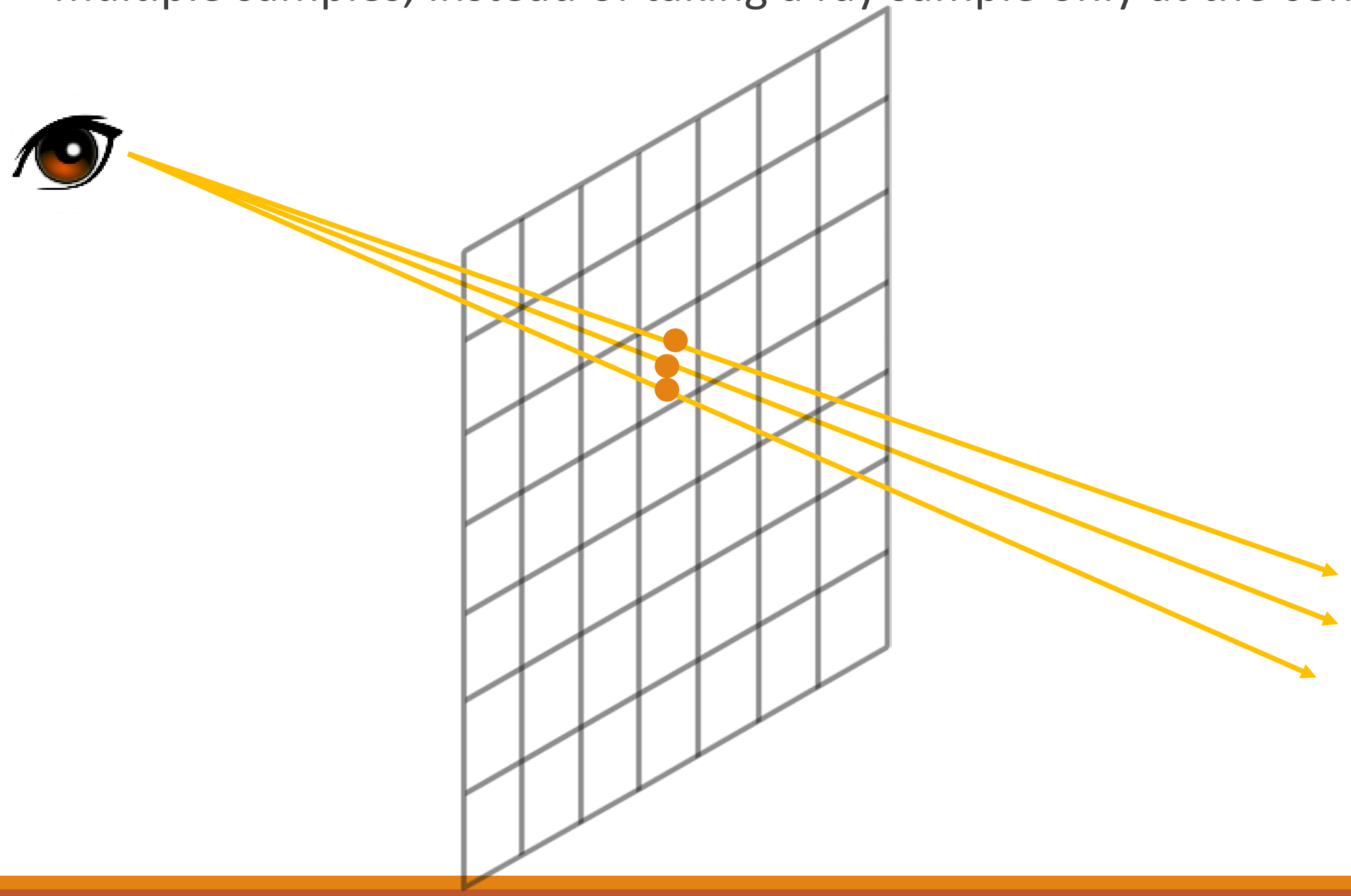
- [Cook et al. 1984]
  - The main idea is to replace the single ray with a distribution of rays

- Add randomness to rendering
  - Antialiasing
  - Soft shadows
  - Depth-of-field
  - Motion blur
  - Glossy reflections

[Cook et al. 1984]

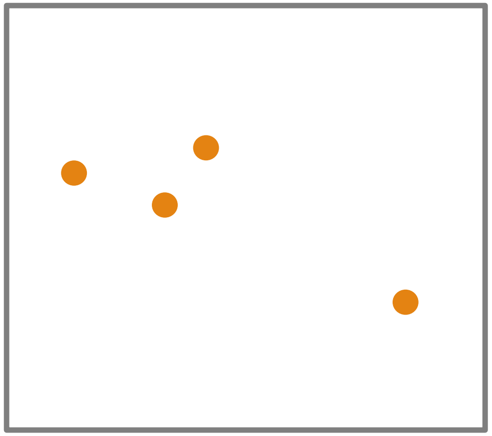
Review: Antialiasing

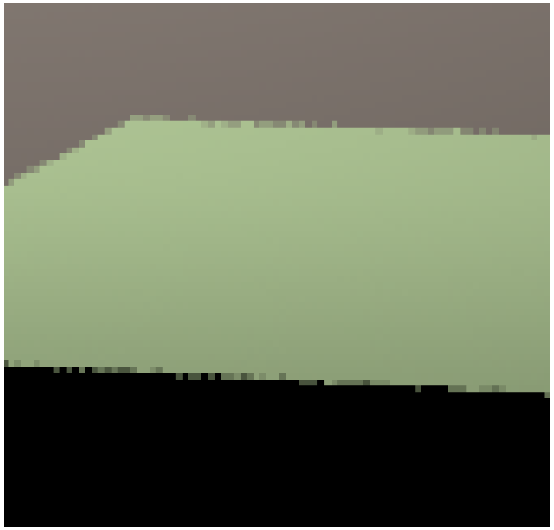
- To reduce image aliasing, we need to compute a pixel color by averaging multiple samples, instead of taking a ray sample only at the center point



Review: Antialiasing

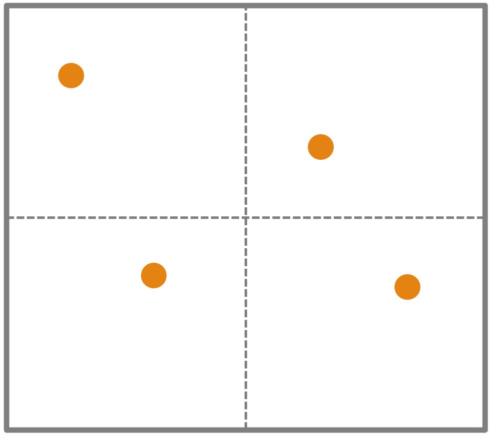
- e.g. four samples / pixel
- Random sampling: randomly generate n^2 rays
- for each pixel (x, y) do
  - $c(x, y) = 0$
  - for p = 0 to $n^2 - 1$ do
    - $c(x, y) = c(x, y) + trace(x + \epsilon_1, y + \epsilon_2)$
  - $c(x, y) = c(x, y)/n^2$
- $\epsilon \in [0,1)$ is a random number
- The regular pattern is converted into image noise

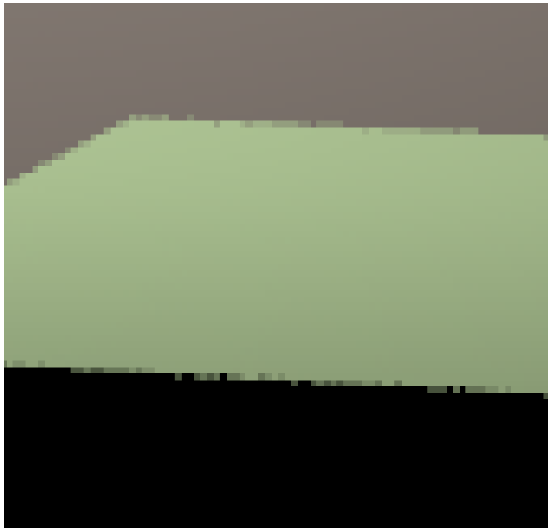




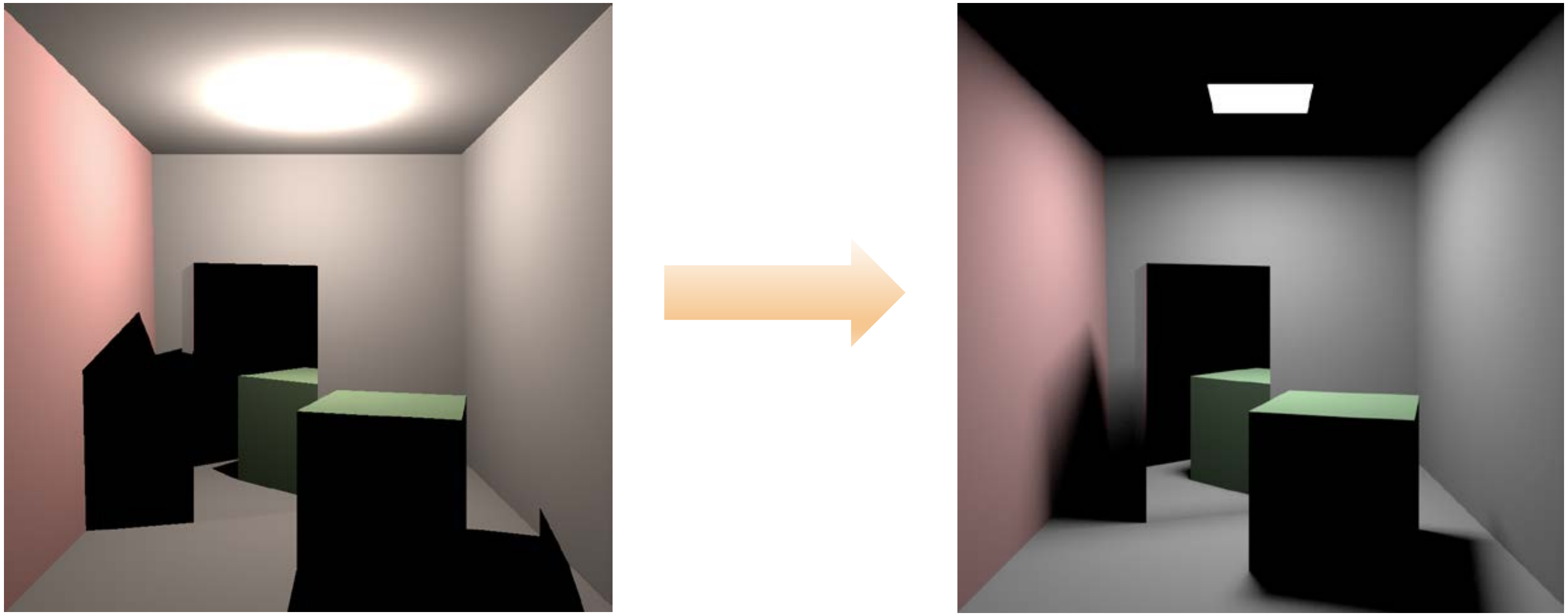
Review: Antialiasing

- e.g. four samples / pixel
- Jittering (stratified sampling): randomly generate a ray within each grid
- for each pixel (x, y) do
  - $c(x,y) = 0$
  - for p = 0 to n – 1 do
    - for q = 0 to n – 1 do
      - $c(x,y) = c(x,y) + trace\left(x + \frac{p+\epsilon_1}{n}, y + \frac{q+\epsilon_2}{n}\right)$
    - $c(x,y) = c(x,y)/n^2$
- This is a hybrid approach between the regular and random sampling



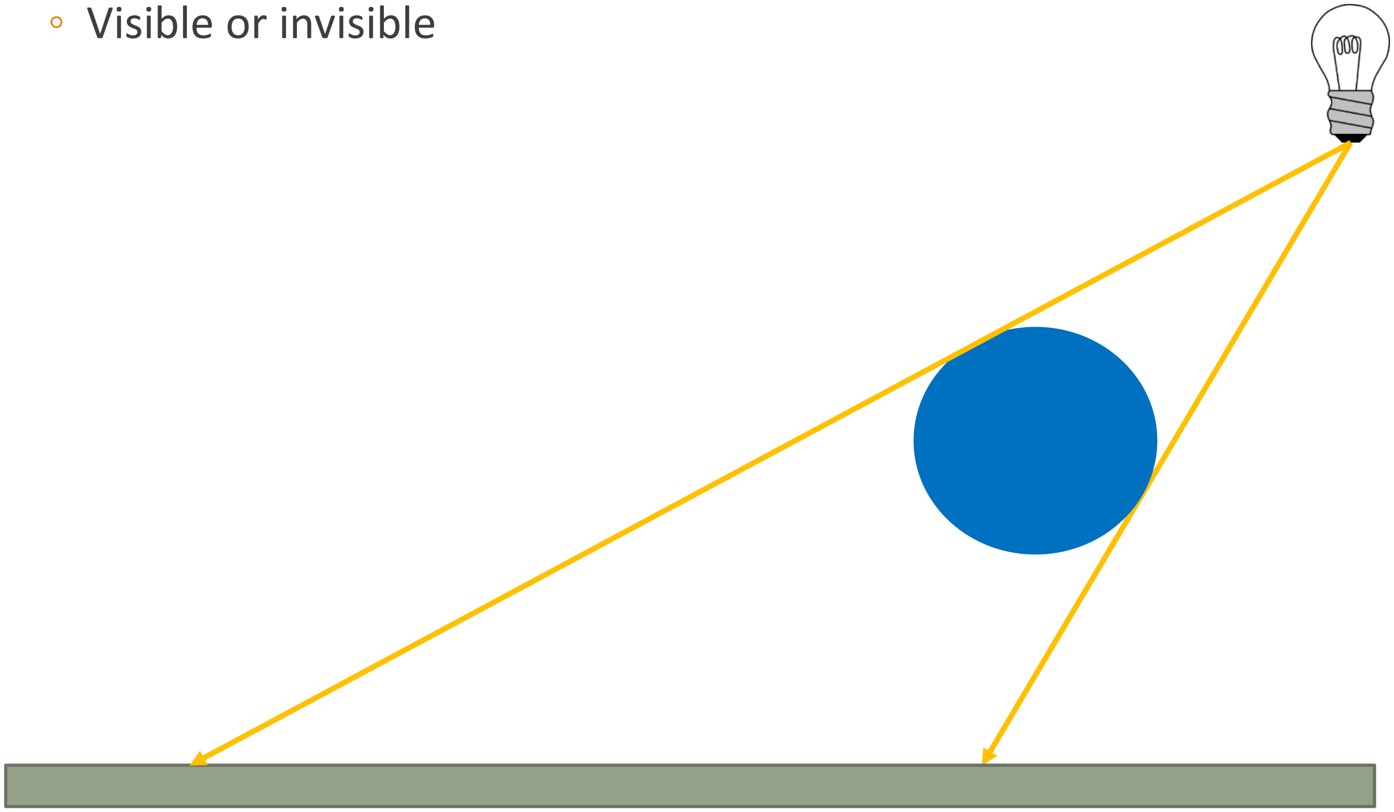


Review: Soft Shadows



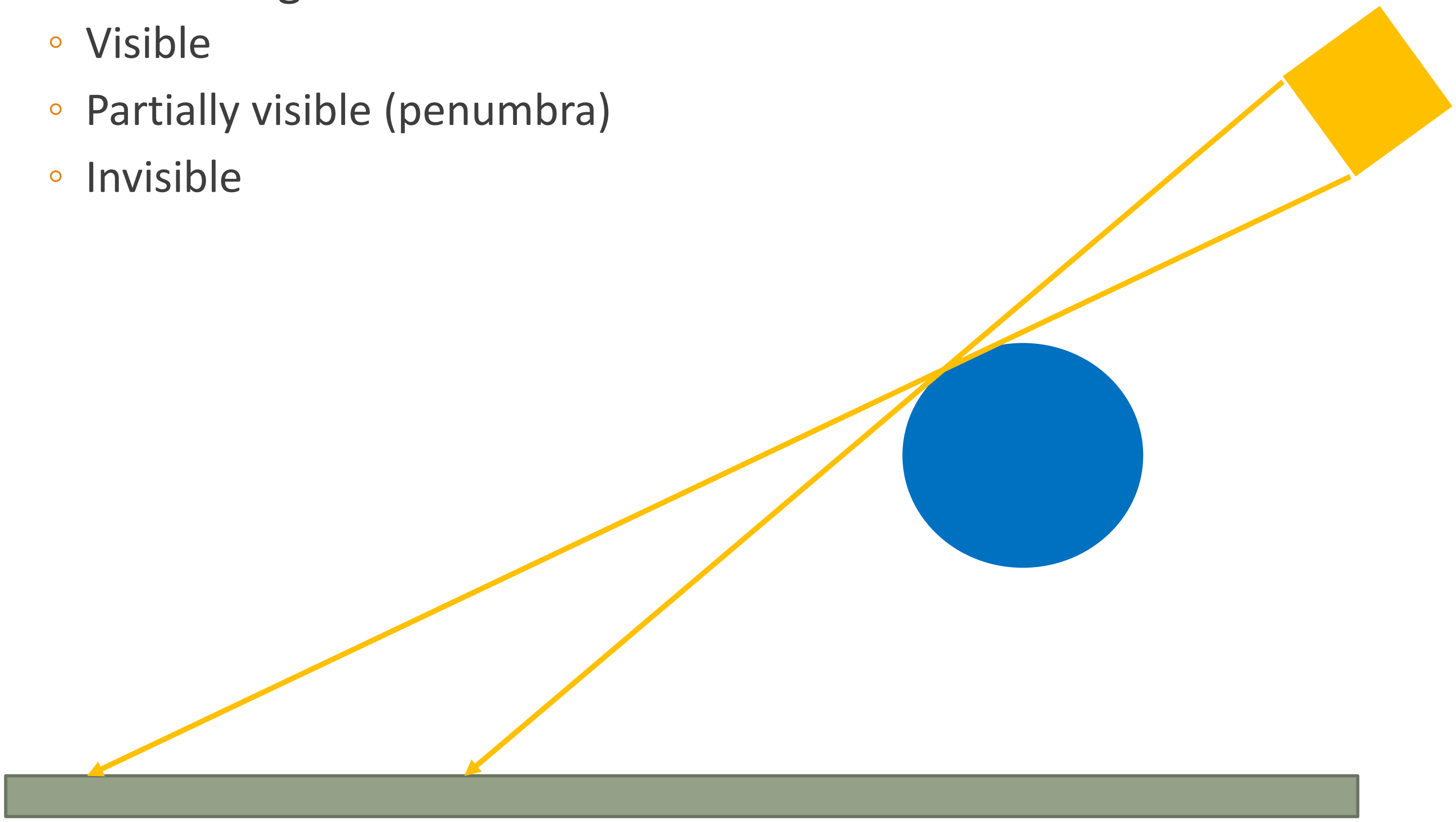
Review: Soft Shadows

- A point light source introduces hard shadows
  - Visible or invisible



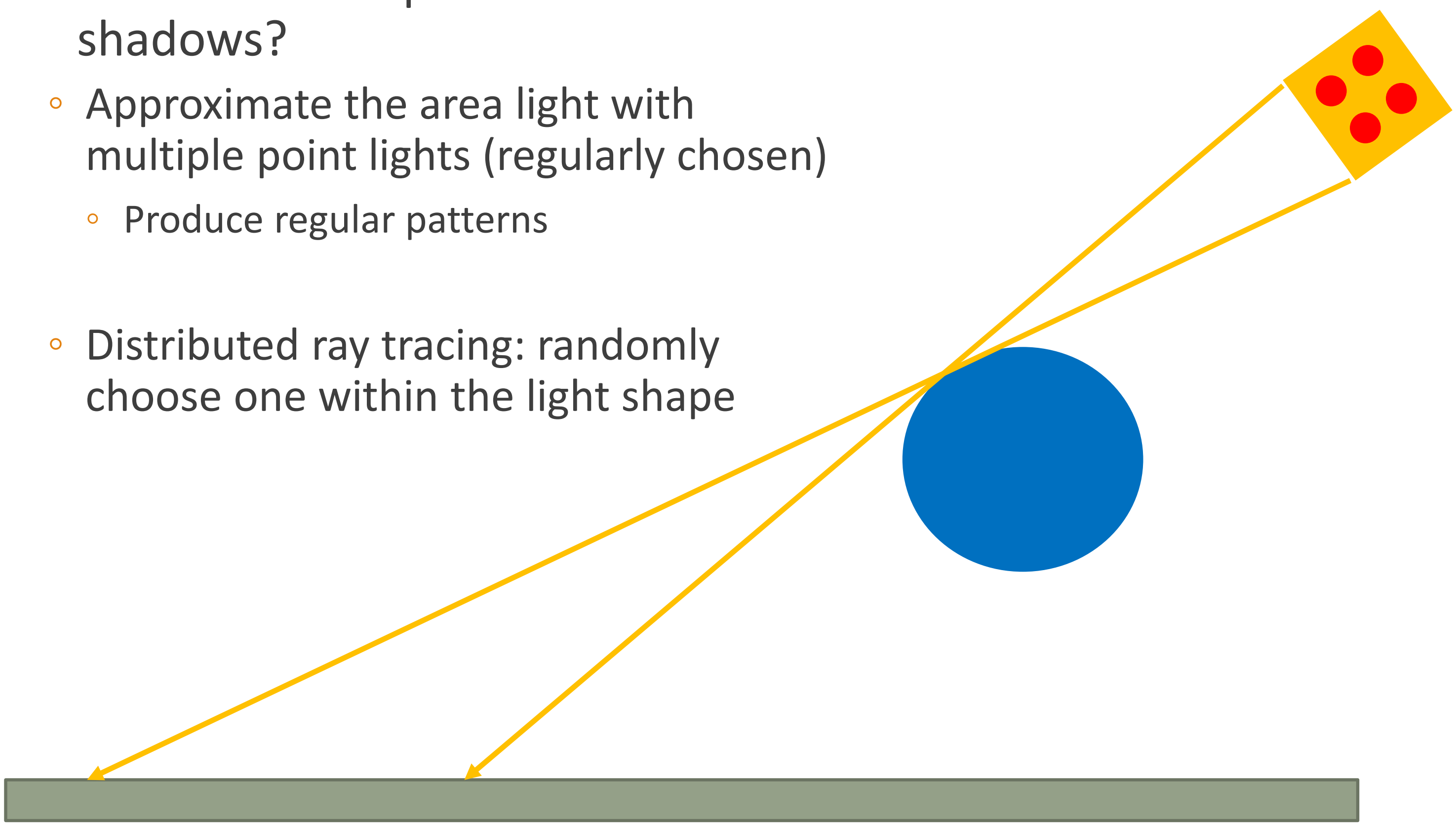
Review: Soft Shadows

- An area light source introduces soft shadows
  - Visible
  - Partially visible (penumbra)
  - Invisible



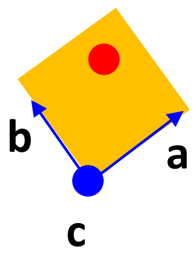
Review: Soft Shadows

- How can we implement the soft shadows?
  - Approximate the area light with multiple point lights (regularly chosen)
    - Produce regular patterns
  - Distributed ray tracing: randomly choose one within the light shape

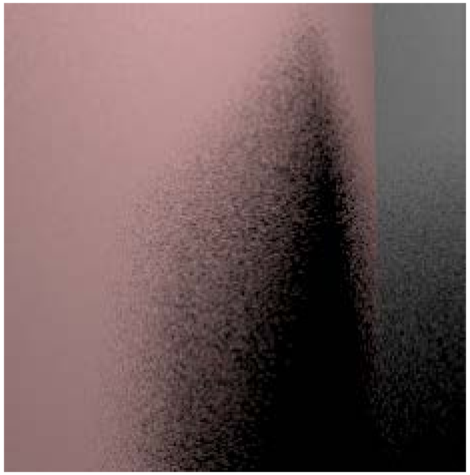


Review: Soft Shadows

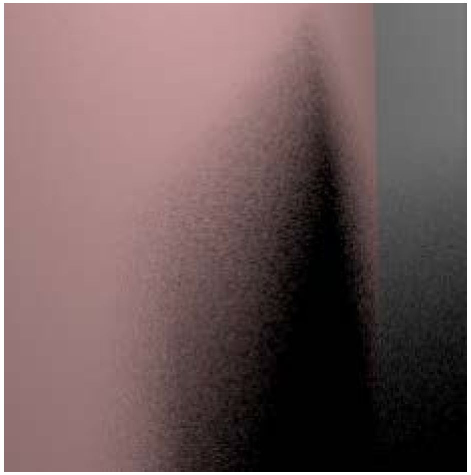
- e.g. area light defined as a parallelogram
  - Select a random point
    - $\boldsymbol{l_{pos}} = \boldsymbol{c} + \varepsilon_1 \boldsymbol{a} + \varepsilon_2 \boldsymbol{b}$
    - $\varepsilon_1, \varepsilon_2 \in [0,1)$
  - Generate a shadow ray from this point



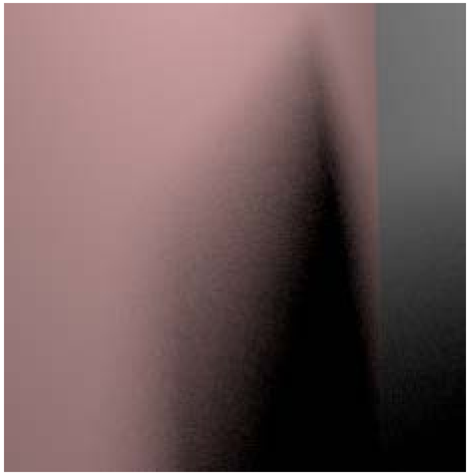
Review: Soft Shadows



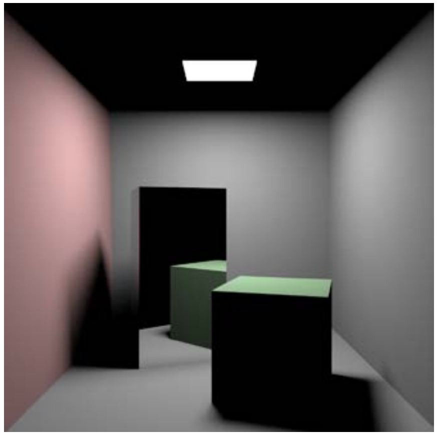
4 shadow rays / pixel



16 shadow rays / pixel

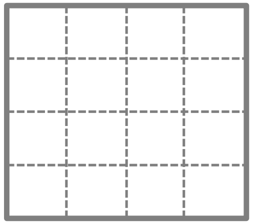


64 shadow rays / pixel

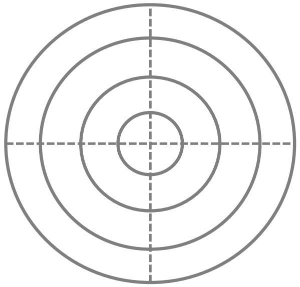


Random Point Selection

- Problem: draw N samples from $[0,1]^2$
  - i.e., provide $(x_1, y_1), \dots, (x_N, y_N)$
  - Techniques
    - Random sampling
    - Jittering



- Problem: draw N samples from a disk shape (a camera lens)
  - Randomly select u_i, v_i from [0,1]
  - $\phi_i = 2\pi u_i$
  - $r_i = v_i R$



  - Note
    - This will cover all the area of a circle with a radius R
    - The downside is that the selected points will be distributed non-uniformly

Inverse Transform Sampling

- 1D density function $f(x)$ with $x \in [x_{min}, x_{max}]$
- Q. Can we generate random numbers α_i so that they can have density $f(x)$ using a set of uniform random numbers $\xi_i \in [0,1]$?

- Cumulative probability distribution function $F(x)$:
  - $P(\alpha < x) = F(x) = \int_{x_{min}}^{x} f(x')d\mu$
- $\alpha_i = F^{-1}(\xi_i)$
- Example:
  - $y = x^2 \quad or\ f(x) = x^2 \quad (x > 0)$
  - $x = \sqrt{y} \quad or\ f^{-1}(x) = \sqrt{x}$

Inverse Transform Sampling

- 1D density function $f(x)$ with $x \in [x_{min}, x_{max}]$
- Q. Can we generate random numbers α_i so that they can have density $f(x)$ using a set of uniform random numbers $\xi_i \in [0,1]$?
- Problem: generate random points x_i that have the following density:
  - $f(x) = \frac{3x^2}{2} \;\; on \; [-1,1]$
  - 1. $F(x) = \frac{x^3 + 1}{2}$
  - 2. $F^{-1}(x) = \sqrt[3]{2x - 1}$
  - 3. $(x_1, \dots, x_N) = \left(\sqrt[3]{2\xi_1 - 1}, \dots, \sqrt[3]{2\xi_N - 1}\right)$

  - Note
    - ξ can be jittered samples

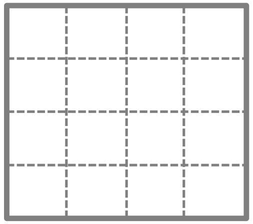
Inverse Transform Sampling

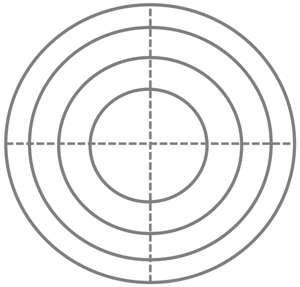
- 2D density function $f(x, y)$ with $(x, y) \in [x_{min}, x_{max}] \times [y_{min}, y_{max}]$

- $P(\alpha_x < x \text{ and } \alpha_y < y) = F(x, y) = \int_{y_{min}}^{y} \int_{x_{min}}^{x} f(x', y') d\mu(x', y')$
- Steps.
  - Choose x_i using a marginal distribution $F(x, y_{max})$
  - Choose y_i according to $\frac{F(x_i, y)}{F(x_i, y_{max})}$

Inverse Transform Sampling

- Example:
  - $P(r < r_0 \; and \; \phi < \phi_0) = F(r_0, \phi_0) = \int_0^{\phi_0} \int_0^{r_0} \frac{r dr d\phi}{\pi R^2} = \frac{\phi r^2}{2\pi R^2}$
- Procedure.
  - Generate two random numbers $\xi_1, \xi_2 \in [0,1]$
  - Choose ϕ_i using a marginal distribution $F(r_{max}, \phi)$
    - $F(r_{max}, \phi) = \frac{\phi R^2}{2\pi R^2}$
    - $\phi = 2\pi\xi_1$
  - Choose r_i according to $\mathrm{F(r|}\phi_i) = \frac{F(r,\phi_i)}{F(r_{max},\phi_i)}$
    - Similarly, $r = R\sqrt{\xi_2}$





Rejection

- Choose some random points according to a distribution and reject some of them
- Example: draw uniform random points within the unit circle
  - Choose uniform random samples $(x, y) \in [-1,1]^2$
  - Reject the samples outside the circle
- Procedure
  - $i = 1$
  - $while\ (i < N)$
    - $x_i = -1 + 2 \times rand()$ // $rand()$ will return uniform random sample in [0, 1]
    - $y_i = -1 + 2 \times rand()$
    - $if\ (x_i^2 + y_i^2 < r)$
      - $i = i + 1$

Rejection

- Choose some random points according to a distribution and reject some of them

- Pros
  - Very simple to code
- Cons
  - Can be very inefficient given complex scenarios