

Lecture slides (CT4201/EC4215 – Computer Graphics)

Distributed Ray Tracing

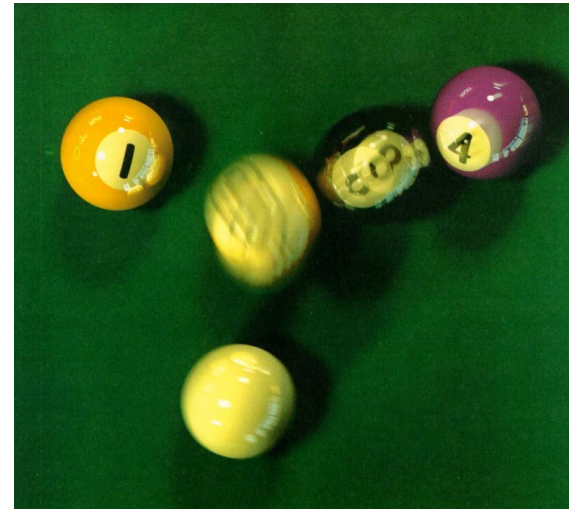
Lecturer: Bochang Moon

Distributed Ray Tracing

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



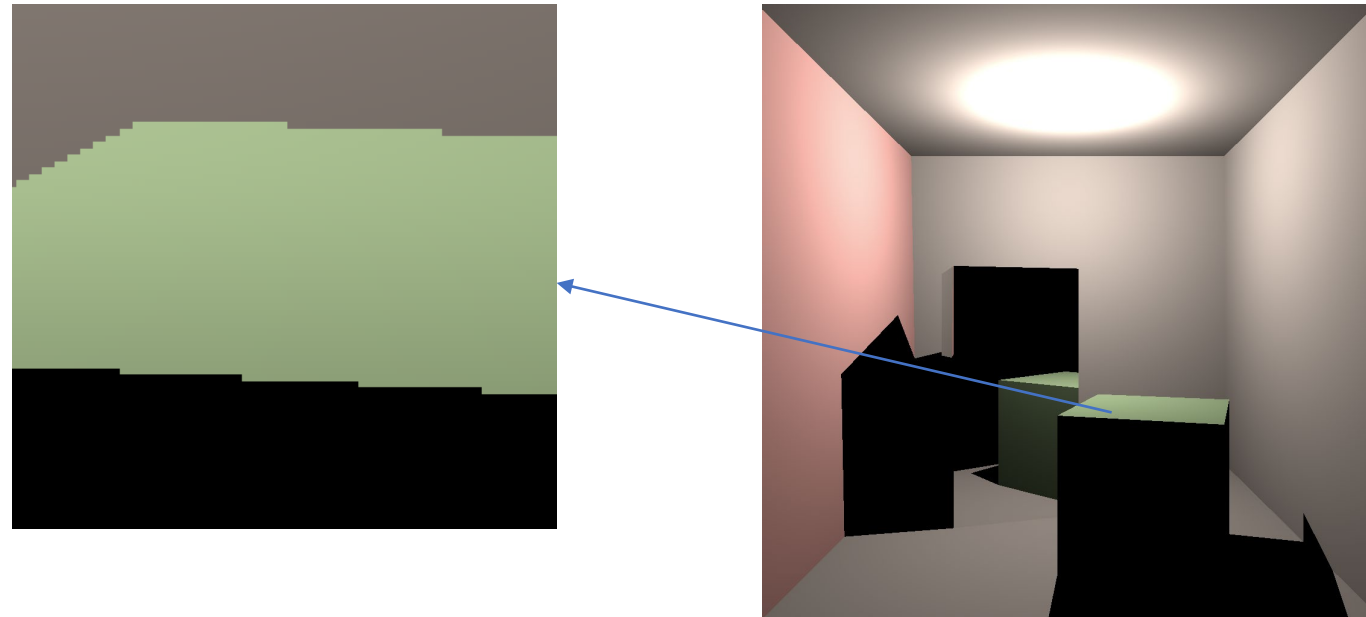
[Whitted 1980]



[Cook et al. 1984]

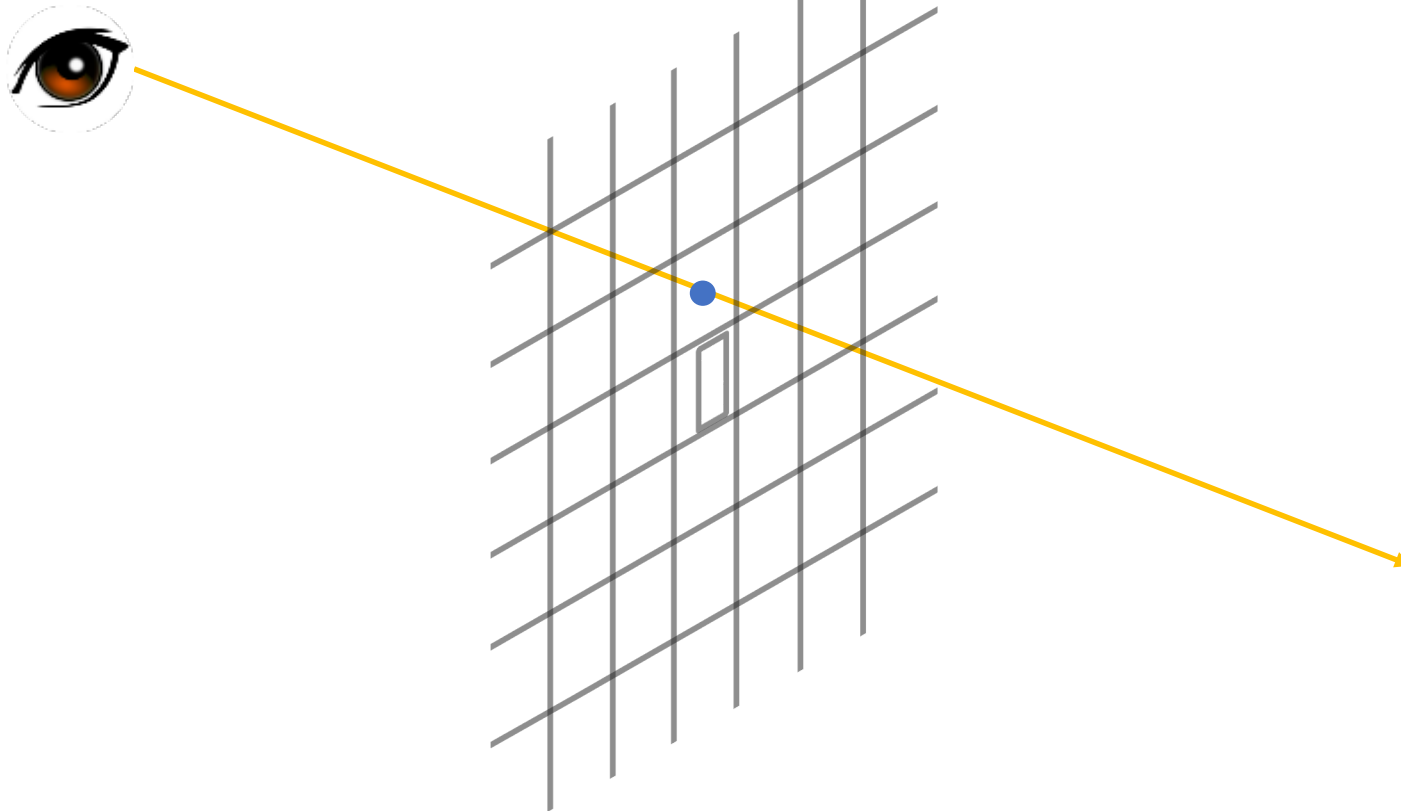
Antialiasing

- 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



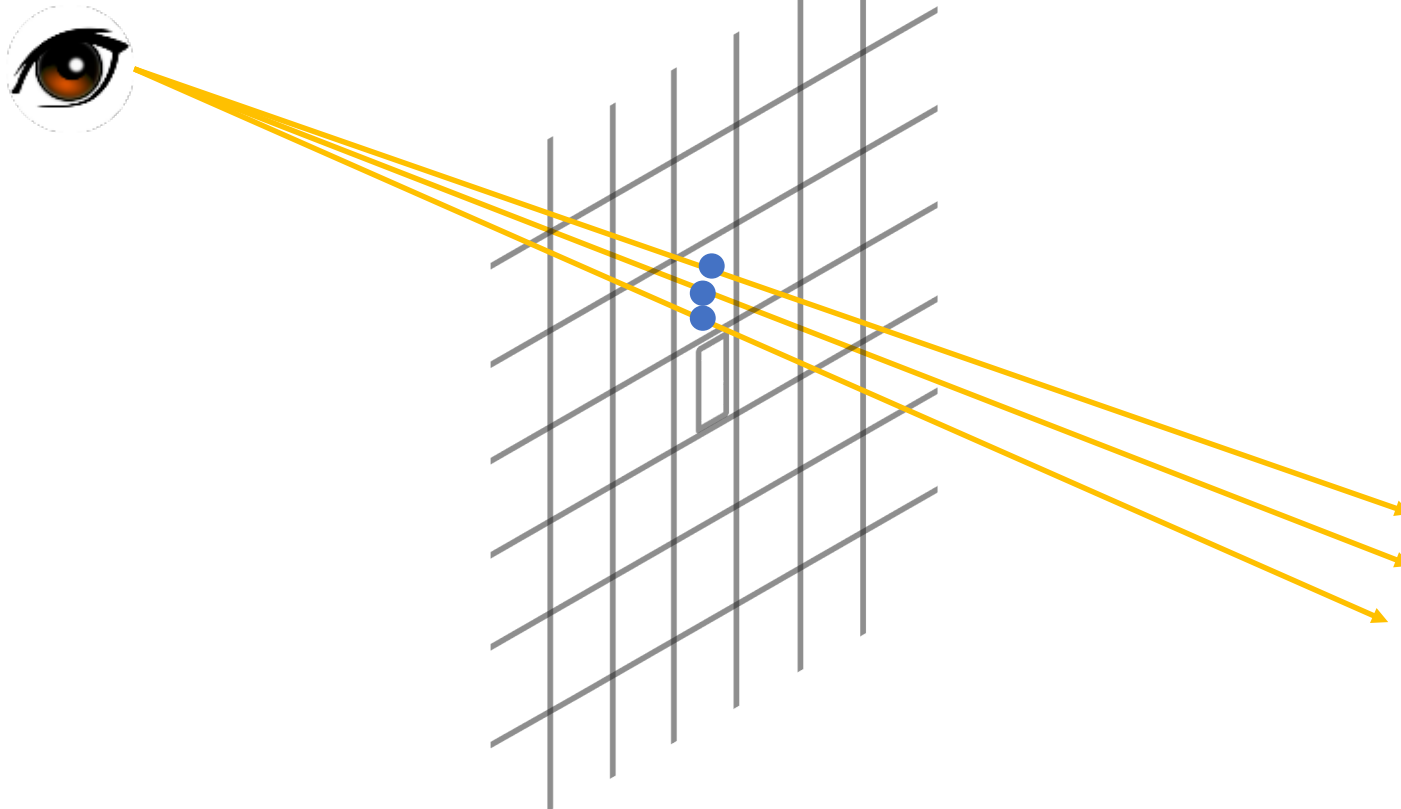
Antialiasing

- 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



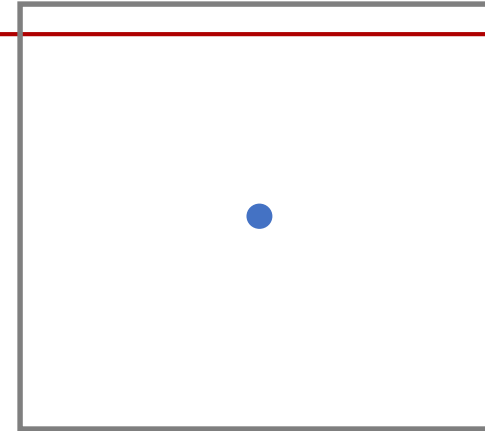
Antialiasing

- 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



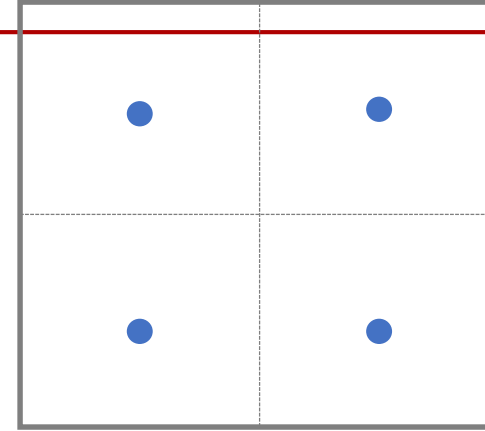
Antialiasing

- e.g. one sample / pixel
- for each pixel (x, y) do
 - $c(x, y) = \text{trace}(x + 0.5, y + 0.5)$



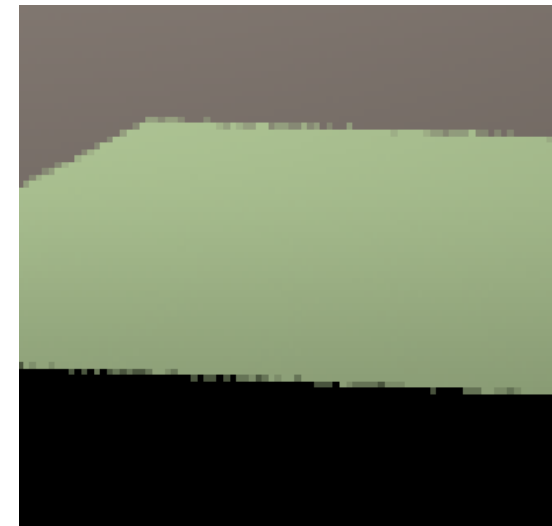
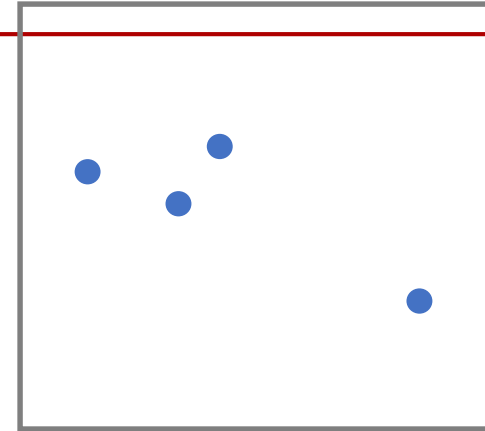
Antialiasing

- e.g. four samples / pixel
- Regular sampling: divide each pixel area to $n \times n$ grids and 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
 - $c(x, y) = c(x, y) + \text{trace}\left(x + \frac{p+0.5}{n}, y + \frac{q+0.5}{n}\right)$
 - $c(x, y) = c(x, y)/n^2$
- Equal to the traditional rendering
 - Uses a larger image resolution
 - Down-sample the image to make a target resolution
 - Still make the regular pattern!



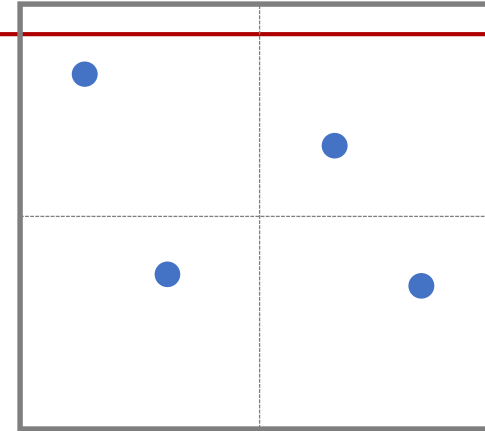
Antialiasing

- 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
 - $c(x, y) = c(x, y) + \text{trace}(x + \epsilon_1, y + \epsilon_2)$
 - $c(x, y) = c(x, y) / n^2$
- $\epsilon \in [0, 1)$ is a random number
- The regular pattern is converted into image noise



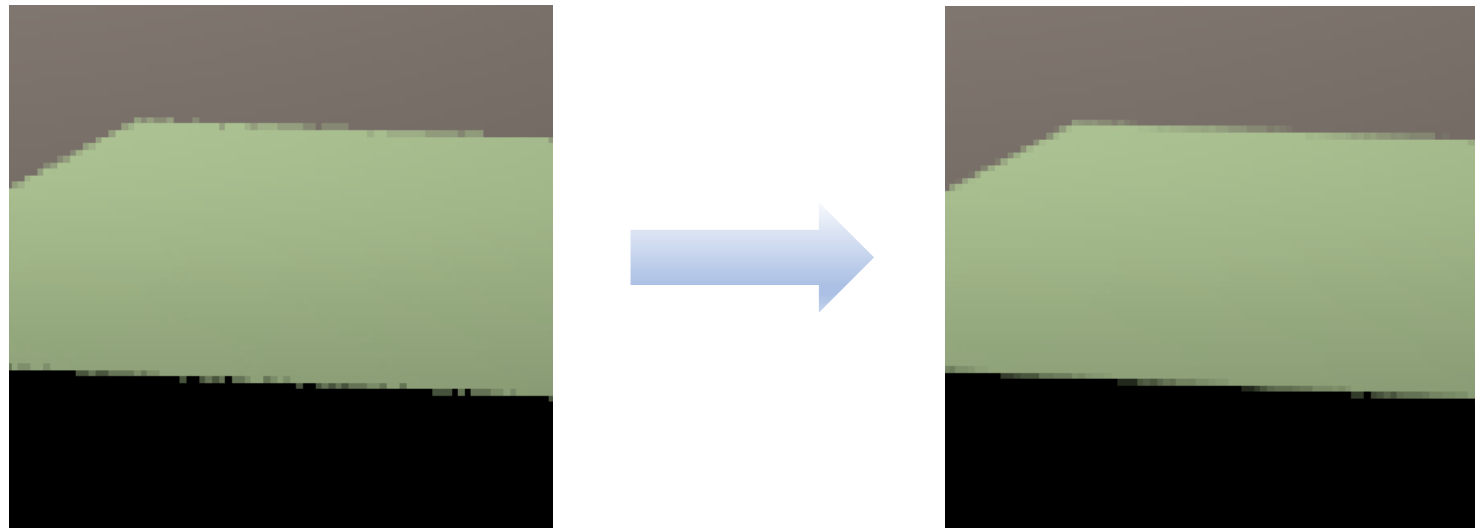
Antialiasing

- 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
 - $c(x, y) = c(x, y) + \text{trace}\left(x + \frac{p+\epsilon_1}{n}, y + \frac{q+\epsilon_2}{n}\right)$
 - $c(x, y) = c(x, y) / n^2$
- This is a hybrid approach between the regular and random sampling

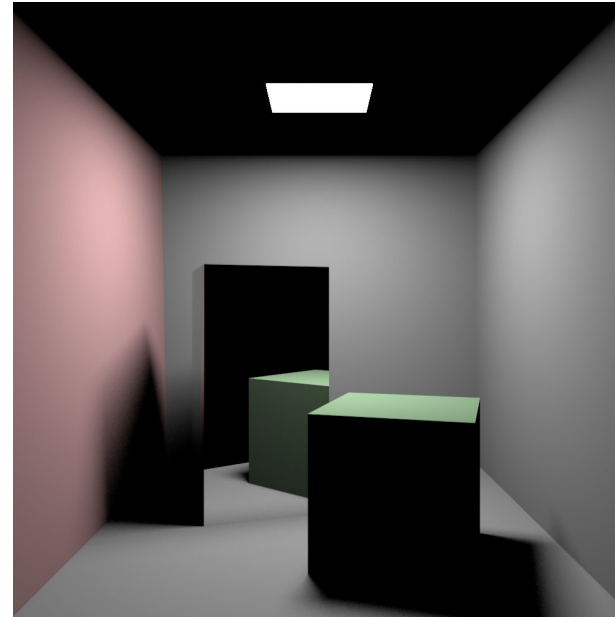
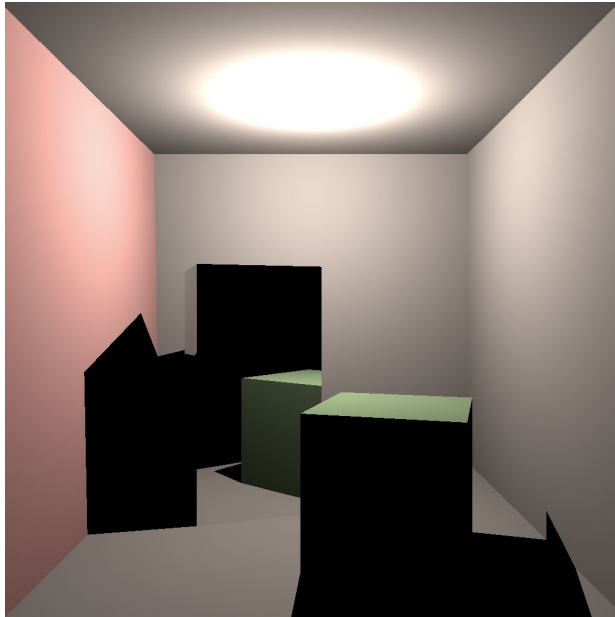


Antialiasing

- More samples?
 - e.g. 16 samples / pixel
- Ground truth solution? ∞ samples / pixel
- Note that rendering time is a function of the number of ray samples

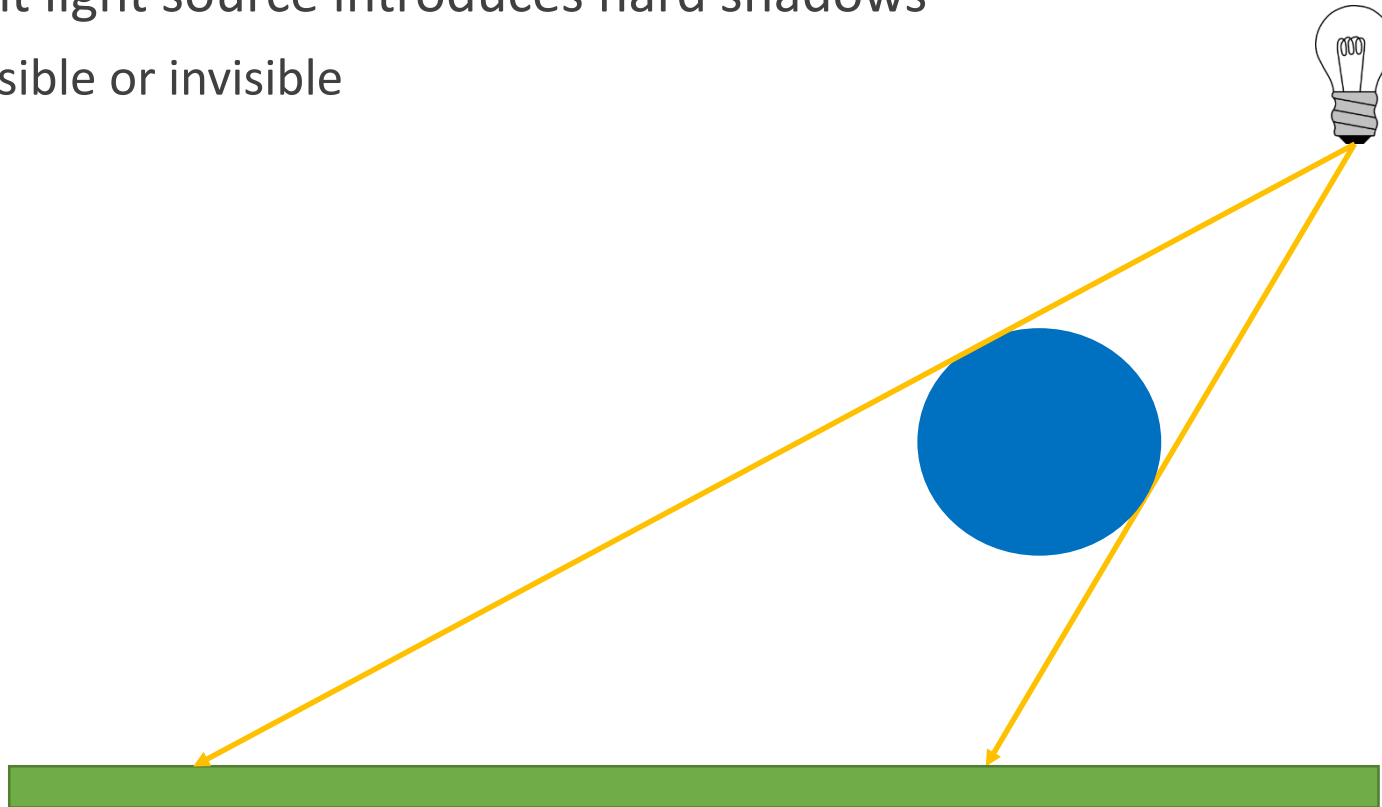


Soft Shadows



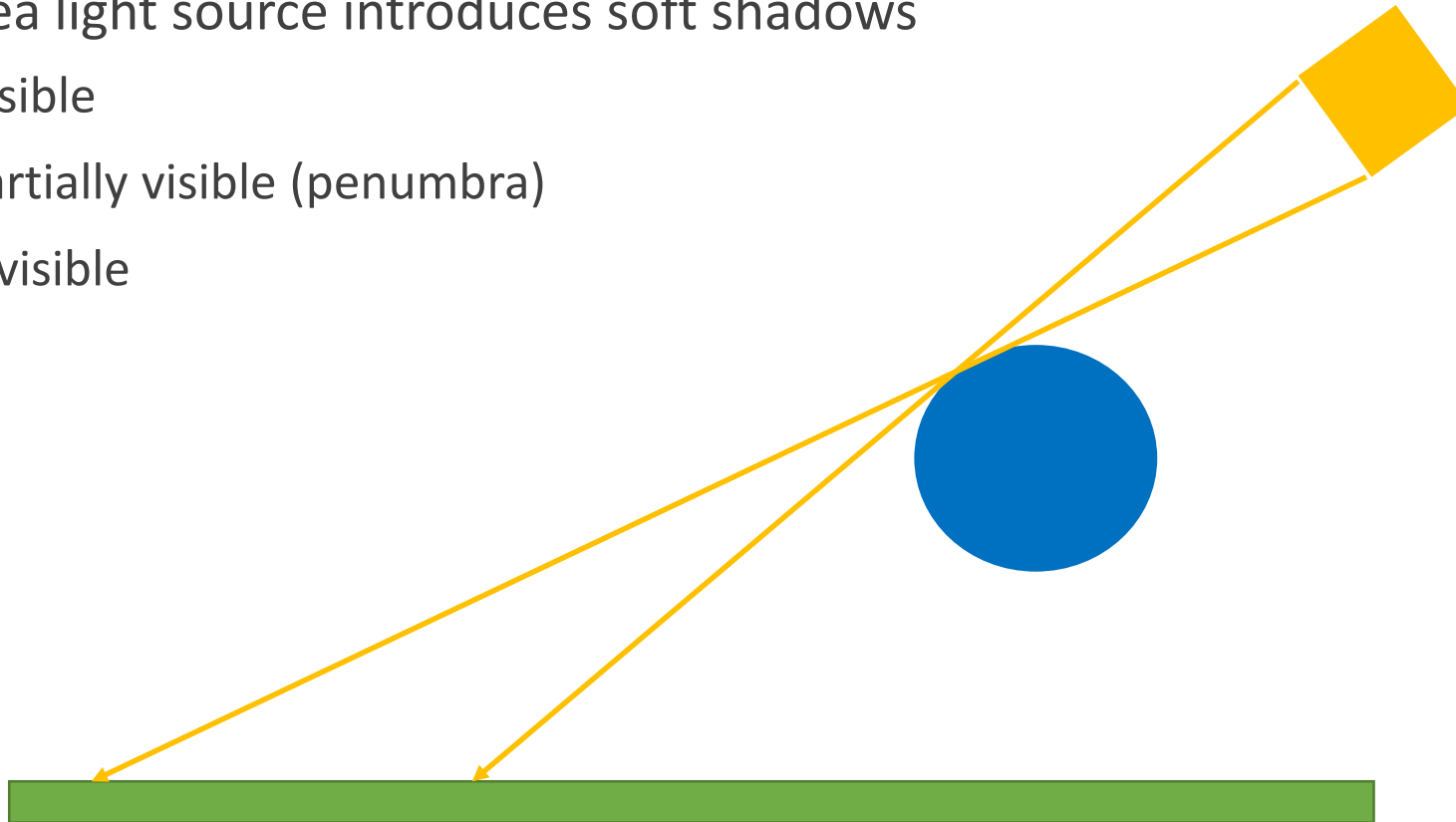
Soft Shadows

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



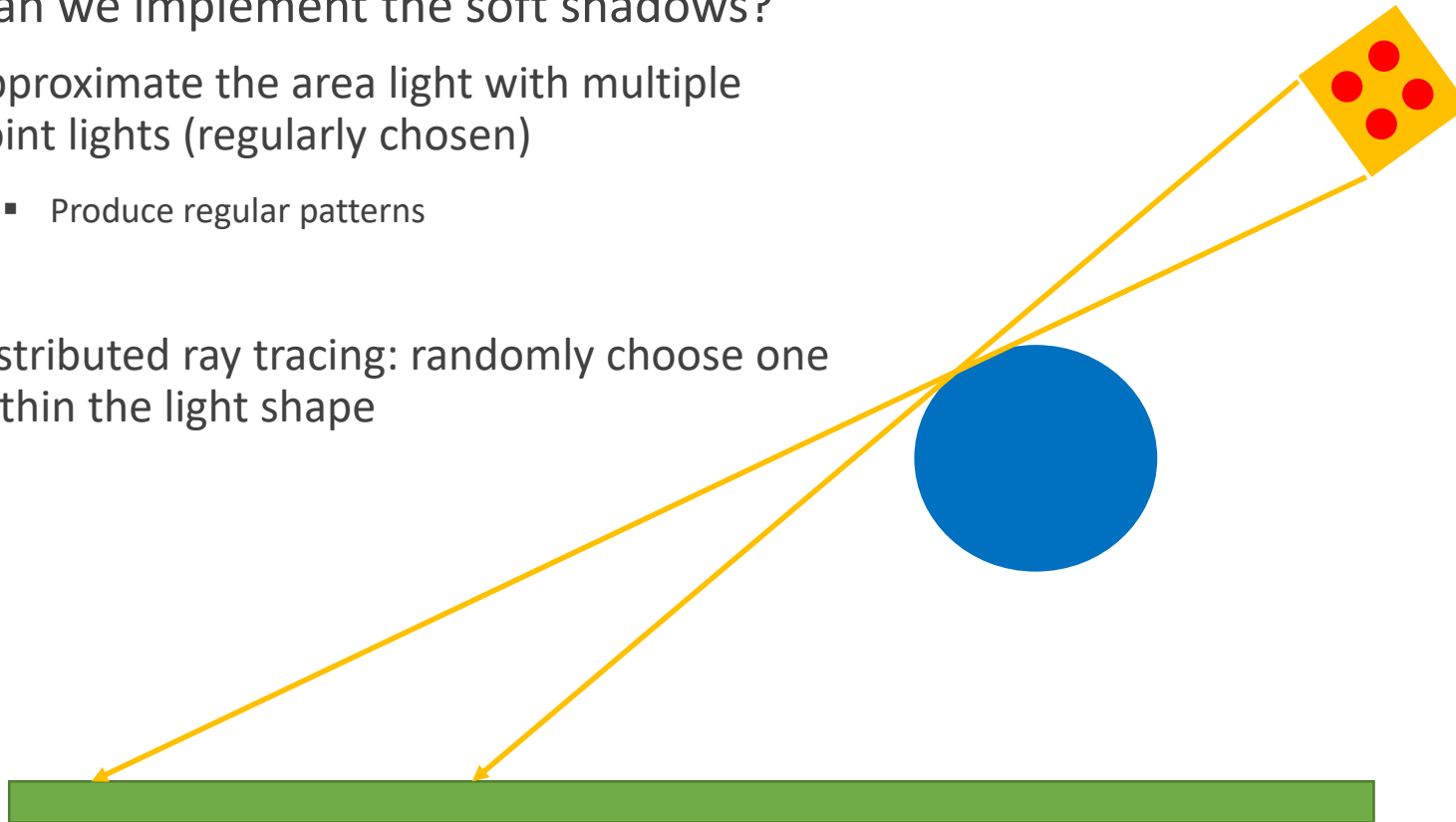
Soft Shadows

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



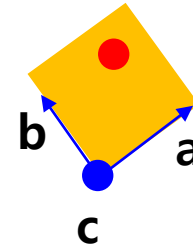
Soft Shadows

- 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

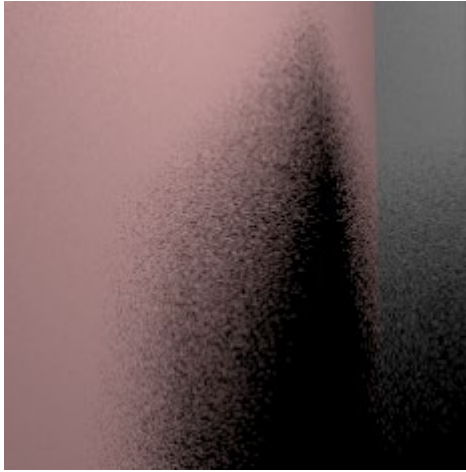


Soft Shadows

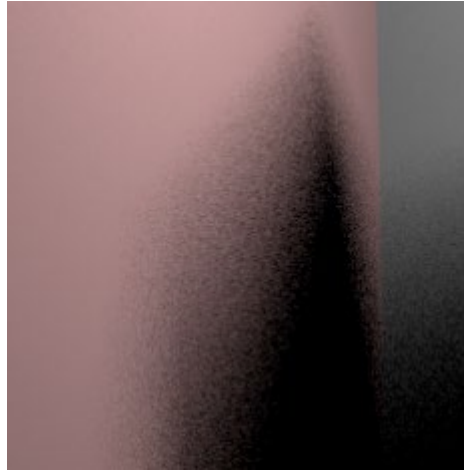
- e.g. area light defined as a parallelogram
 - Select a random point
 - $l_{pos} = c + \varepsilon_1 a + \varepsilon_2 b$
 - $\varepsilon_1, \varepsilon_2 \in [0,1)$
 - Generate a shadow ray from this point



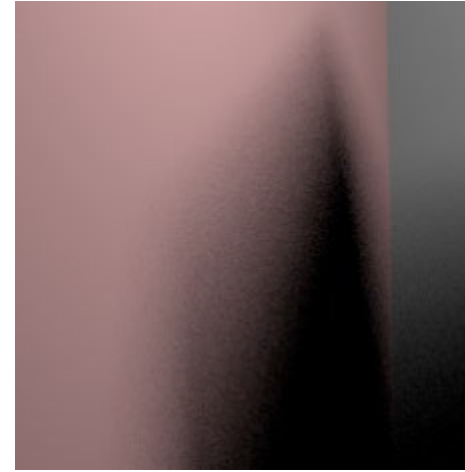
Soft Shadows



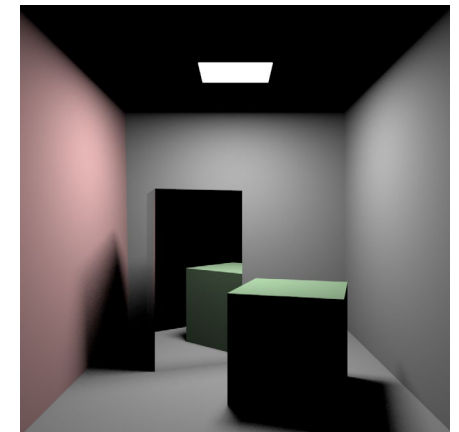
4 shadow rays /
pixel



16 shadow rays /
pixel

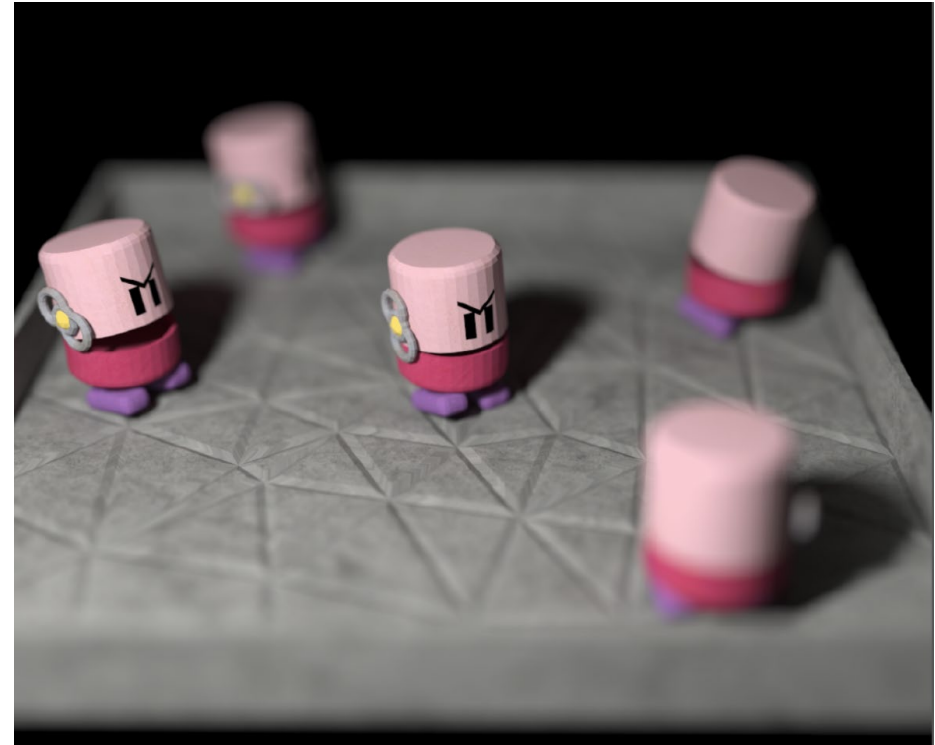


64 shadow rays /
pixel



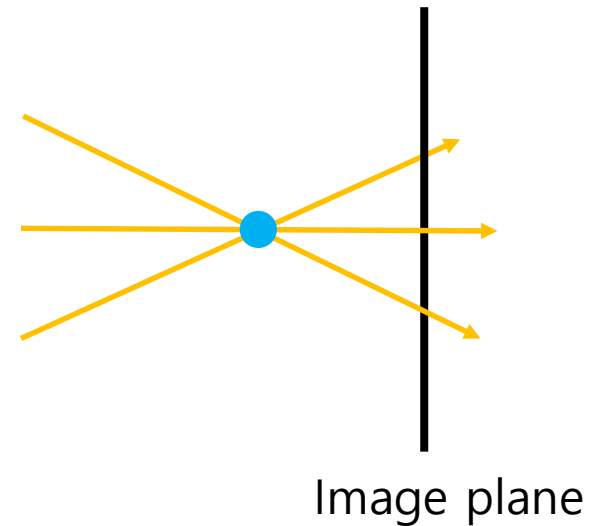
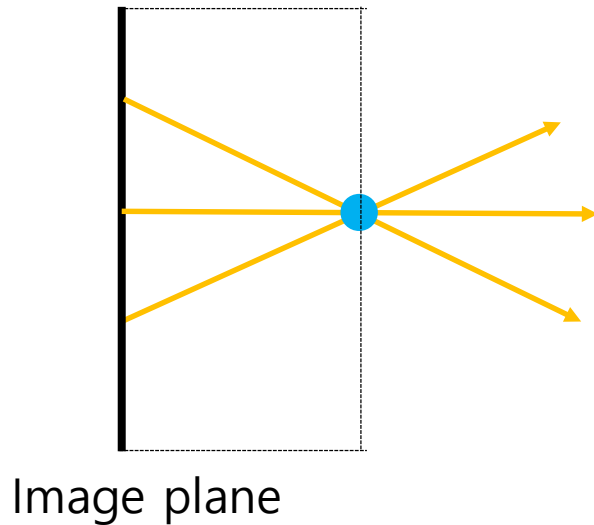
Depth of field

- This effect occurs when using a nonzero size “lens”
 - Approximate real cameras that have lenses with focal lengths



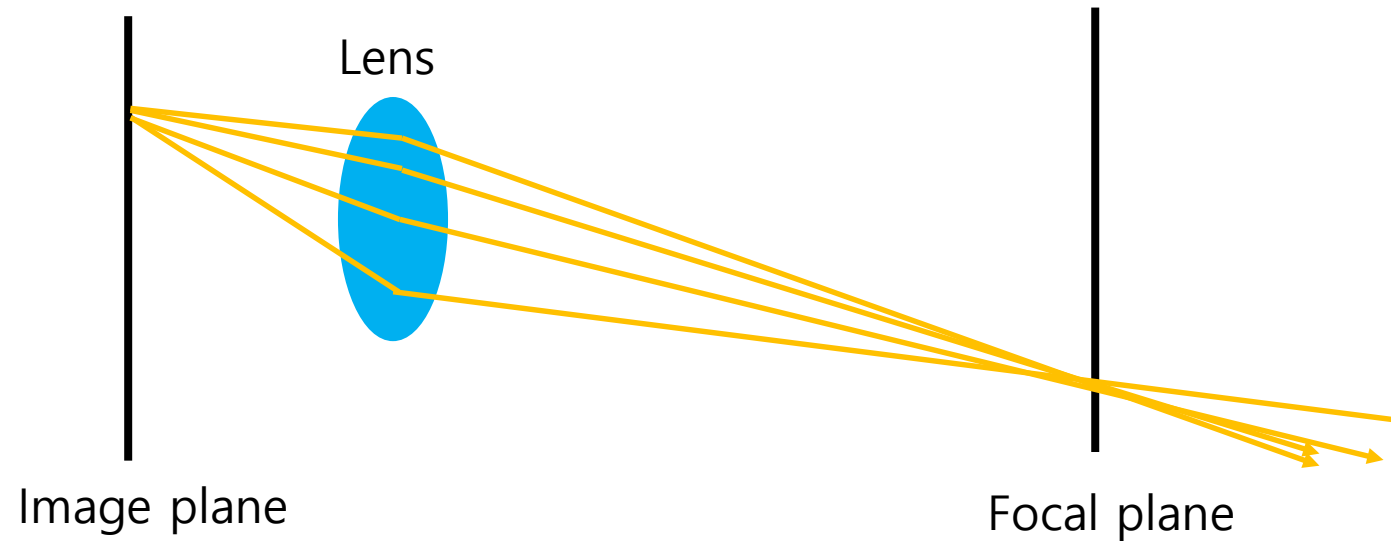
Depth of field

- This effect occurs when using a nonzero size “lens”
 - Approximate real cameras that have lenses with focal lengths
 - Can model the lens with a disk
- Previously, every object is focused due to a pinhole camera model



Depth of field

- This effect occurs when using a nonzero size “lens”
 - Approximate real cameras that have lenses with focal lengths
 - Can model the lens with a disk



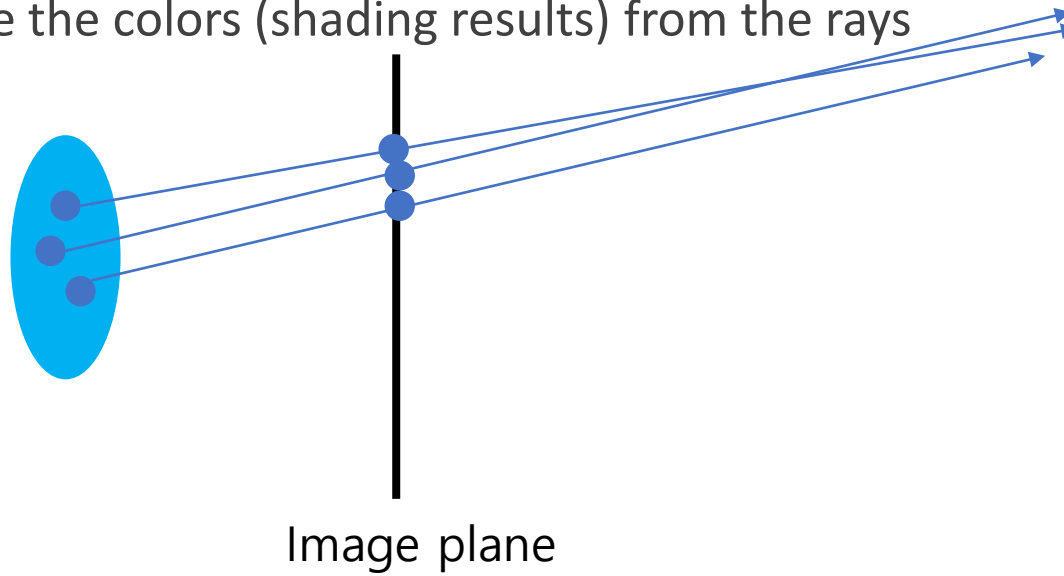
Depth of field

- Implementations

- Generate a primary ray using:

- Randomly choose a point on lens (e.g., disk)
 - Randomly choose a point within a pixel area

- Average the colors (shading results) from the rays



Glossy Reflection

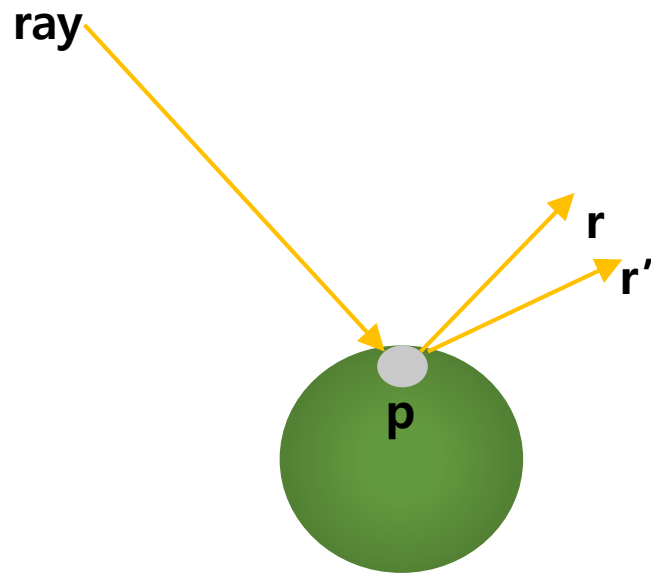
- Some surfaces (e.g., brushed metal) are between the ideal diffuse (e.g., Lambertian) and specular surfaces (e.g., mirror)



Cook et al. 1984

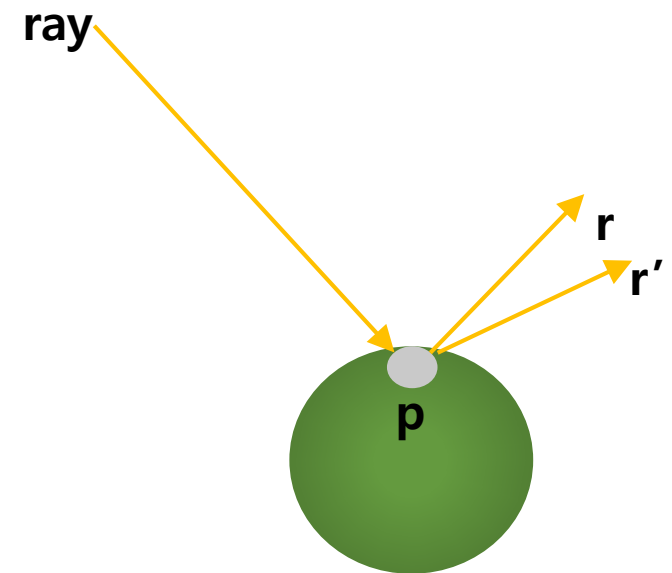
Glossy Reflection

- Can simulate the glossy reflection by generating secondary rays (e.g., reflected ray) with random direction, instead of using the direction of mirror reflections



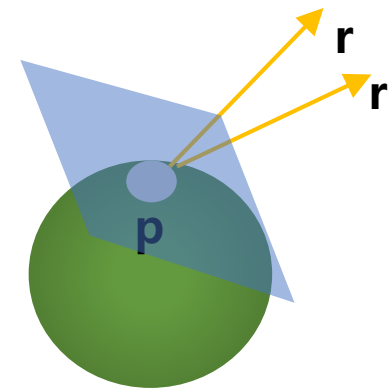
Glossy Reflection

- Generate a basis (\mathbf{u} , \mathbf{v} , \mathbf{w}) from a single vector, \mathbf{r}
 - Previously, we generate a basis from two vectors
 - $\mathbf{w} = \frac{\mathbf{r}}{\|\mathbf{r}\|}$
 - $\mathbf{u} = \frac{\mathbf{t} \times \mathbf{w}}{\|\mathbf{t} \times \mathbf{w}\|}$ (\mathbf{t} can be any vector if \mathbf{t} is not collinear with \mathbf{w})
 - $\mathbf{v} = \mathbf{w} \times \mathbf{u}$



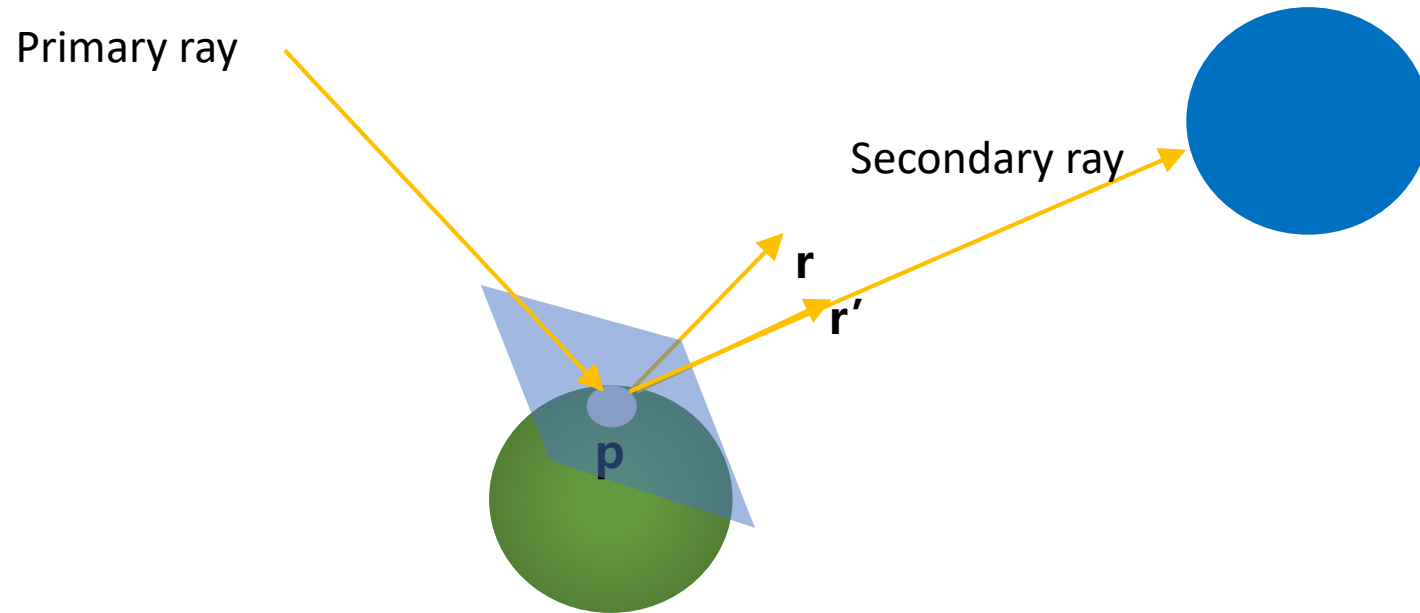
Glossy Reflection

- Generate a basis (\mathbf{u} , \mathbf{v} , \mathbf{w}) from a single vector, \mathbf{r}
 - $\mathbf{w} = \frac{\mathbf{r}}{\|\mathbf{r}\|}$, $\mathbf{u} = \frac{\mathbf{t} \times \mathbf{w}}{\|\mathbf{t} \times \mathbf{w}\|}$, $\mathbf{v} = \mathbf{w} \times \mathbf{u}$
 - \mathbf{u} and \mathbf{v} constructs a 2D square which is perpendicular to \mathbf{r}
- Let a is the width of the square (similar to the size of highlights)
- Sample a 2D random point from a unit square $[0,1]^2$
 - $[\varepsilon_1, \varepsilon_2] = [0,1]^2$
- Transform the point into one on the square
 - $\mathbf{u} = -\frac{a}{2} + \varepsilon_1 a$
 - $\mathbf{v} = -\frac{a}{2} + \varepsilon_2 a$
- Construct the reflected direction
 - $\mathbf{r}' = \mathbf{r} + \mathbf{u}\mathbf{u} + \mathbf{v}\mathbf{v}$



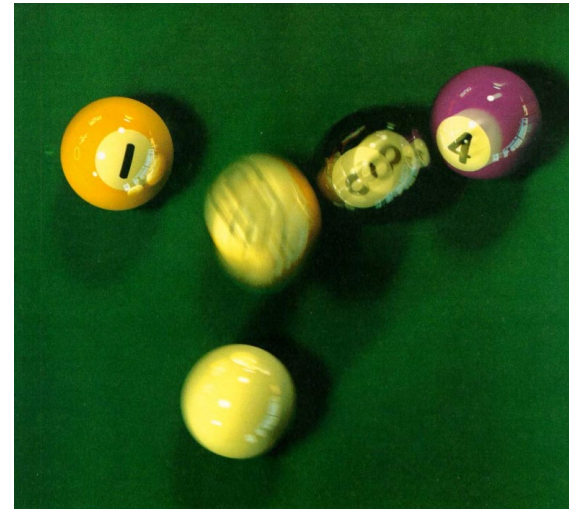
Glossy Reflection

- The reflected ray's
 - Origin: p
 - Direction: r'



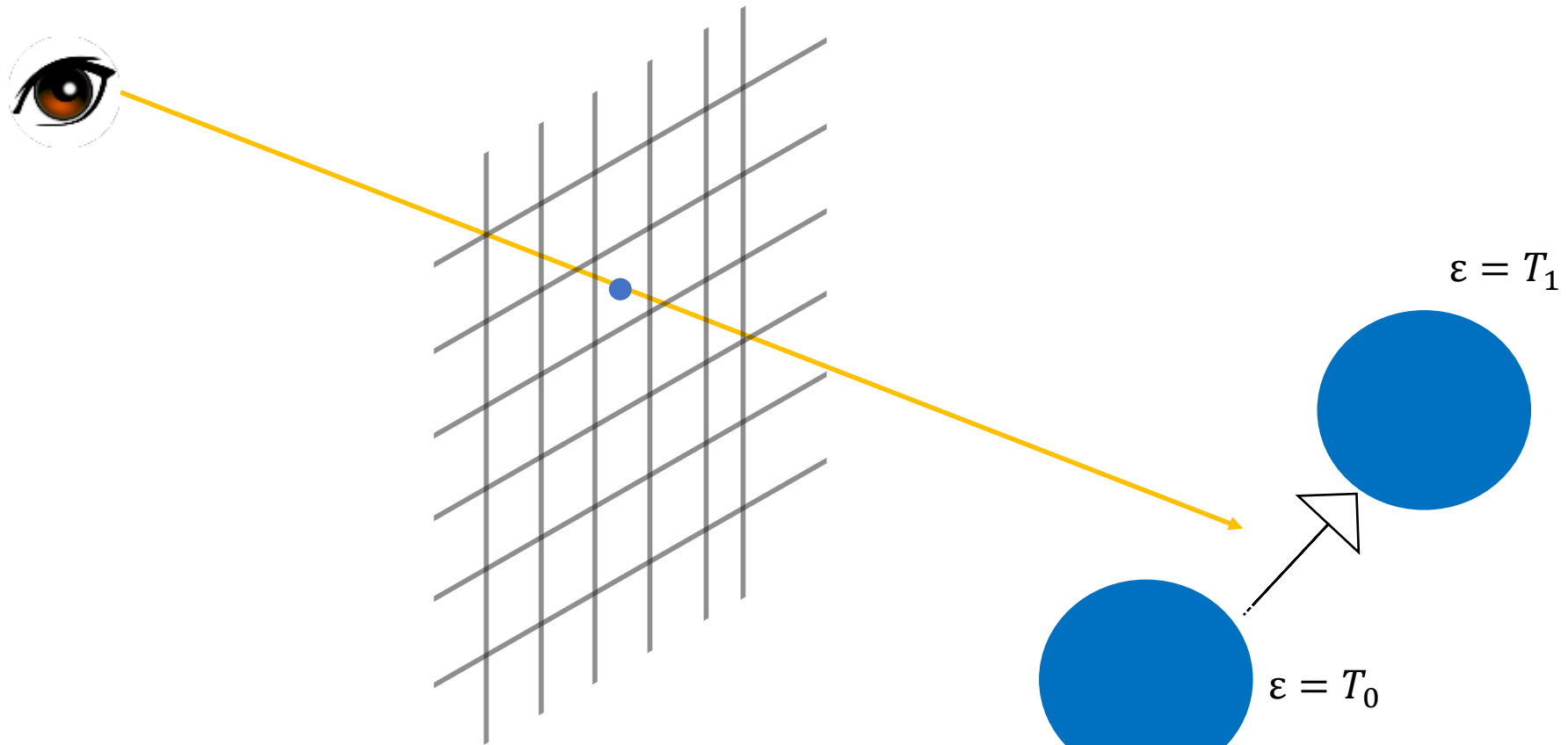
Motion Blur

- Objects can move when the aperture of a real camera is open
- It introduces a blurred appearance of a moving object
- General procedure:
 - Set up a time interval $[T_0, T_1]$
 - Randomly choose a time $\varepsilon \in [T_0, T_1]$
 - A moving object travels using a transform matrix
 - e.g., given two rotation matrices (start and end), we can find an intermediate transform related to ε_1



[Cook et al. 1984]

Motion Blur



Discussion

- Distributed ray tracing introduces a way to simulate distributed effects, by using a distribution of rays
- Whitted vs. distributed ray tracing
 - In the classical ray tracing, rays are determined deterministically
 - Distributed ray tracer uses a random number to simulate the distributed effects
- Next class?
 - Global illumination methods

Further Readings

- Chapter 13
- Distributed Ray Tracing [Cook et al. 1984]