Rendering Techniques

Announcements

Assignment 3 is due 2 June

Tutorial on Friday will be dedicated for A3 questions

Any Questions?

Rendering Techniques

Ray Tracing v Rasterization

Ray Tracing Review

- Other BRDF models
- Monte Carlo Methods
- Area lights

Radiosity

Speeding up Ray Tracing

- Ray Bundling
- Bounding Volumes

https://en.wikipedia.org/wiki/Rendering_(computer_graphics)#Chronology_of_important_published_ideas

Ray Tracing (Image order)

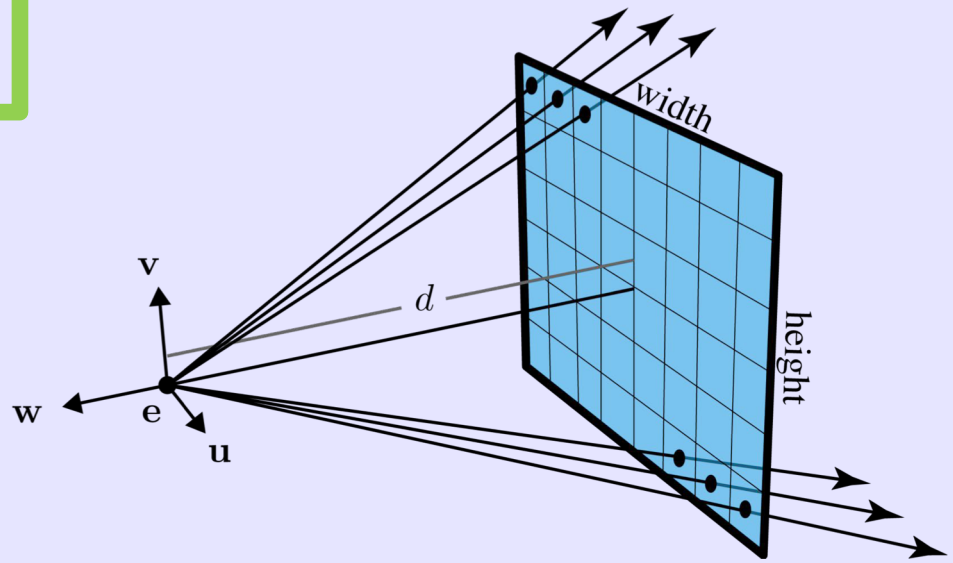
```
for each pixel in the image {
    Generate a ray
    for each object in the scene {
        if (Intersect ray with object) {
            Set pixel colour
        }
    }
}
```

Ray Tracing (Image order)

```
for each pixel in the image {
    Generate a ray
    for each object in the scene {
        if (Intersect ray with object) {
            Set pixel colour
        }
    }
}
```

Ray Tracing (Image order)

```
for each pixel in the image {
        Generate a ray
        for each object in the scene {
                if (Intersect ray with object) {
                        Set pixel colour
                }
        }
}
```



Rasterization (Object order)

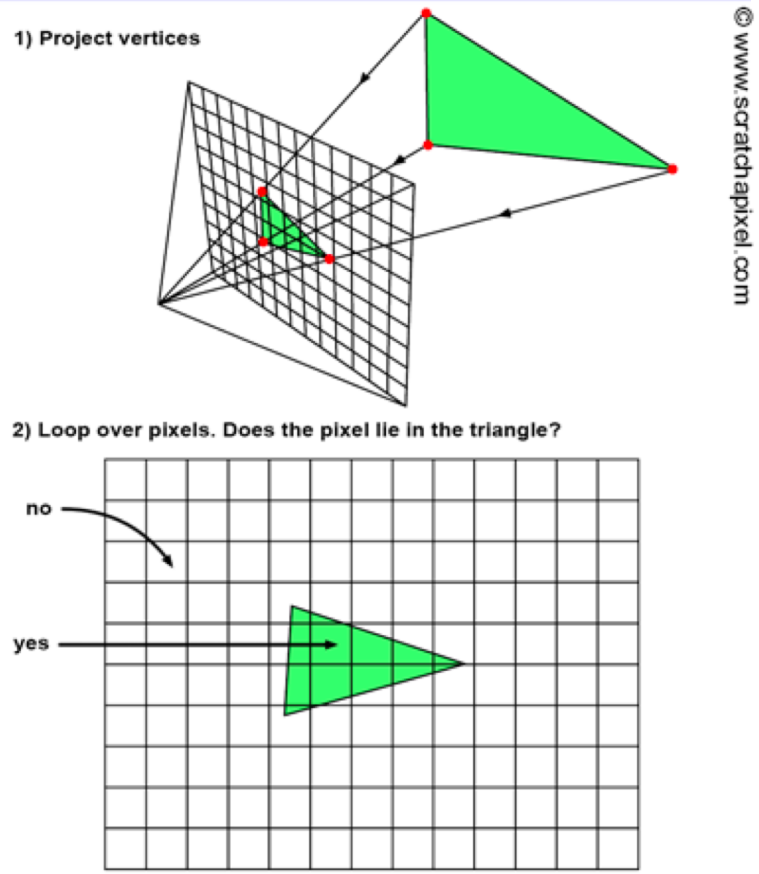
```
for each object in the scene {
    Project object onto image plane
    for each pixel in the image {
        if (object affects pixel) {
            Set pixel colour
        }
    }
}
```

Rasterization (Object order)

```
for each object in the scene {
    Project object onto image plane
    for each pixel in the image {
        if (object affects pixel) {
            Set pixel colour
        }
    }
}
```

Rasterization (Object order)

```
for each object in the scene {
    Project object onto image plane
    for each pixel in the image {
        if (object affects pixel) {
            Set pixel colour
        }
    }
}
```



Set pixel colour

Actually many options for what to do here

The Rendering Equation

$$L_o(x, \vec{w}) = L_e(x, \vec{w}) + \int_\Omega f_r(x, \vec{w}', \vec{w}) L_i(x, \vec{w}')(\vec{w}' \cdot \vec{n}) \mathrm{d}\vec{w}'$$

The Rendering Equation

$$L_o(x, \vec{w}) = L_e(x, \vec{w}) + \int_\Omega f_r(x, \vec{w}', \vec{w}) L_i(x, \vec{w}')(\vec{w}' \cdot \vec{n}) \mathrm{d}\vec{w}'$$

scarier version found here: https://en.wikipedia.org/wiki/Rendering_equation
research paper found here: https://dl.acm.org/doi/10.1145/15886.15902

The Rendering Equation

$$L_o(x, \vec{w}) = L_e(x, \vec{w}) + \int_\Omega f_r(x, \vec{w}', \vec{w}) L_i(x, \vec{w}')(\vec{w}' \cdot \vec{n}) \mathrm{d}\vec{w}'$$

outgoing light at position **x** and direction **w**

The Rendering Equation

$$L_o(x, \vec{w}) = L_e(x, \vec{w}) + \int_\Omega f_r(x, \vec{w}', \vec{w}) L_i(x, \vec{w}')(\vec{w}' \cdot \vec{n}) \mathrm{d}\vec{w}'$$

outgoing light at position **x** and direction **w**

is equal to the sum of

emitted light at position **x** and direction **w**

The Rendering Equation

$$L_o(x, \vec{w}) = L_e(x, \vec{w}) + \int_\Omega f_r(x, \vec{w}', \vec{w}) L_i(x, \vec{w}')(\vec{w}' \cdot \vec{n}) \mathrm{d}\vec{w}'$$

outgoing light at position **x** and direction **w**

is equal to the sum of

emitted light at position **x** and direction **w**

and

reflected light at position **x** and direction **w'**

The Rendering Equation

$$L_o(x, \vec{w}) = L_e(x, \vec{w}) + \int_\Omega f_r(x, \vec{w}', \vec{w}) L_i(x, \vec{w}')(\vec{w}' \cdot \vec{n}) \mathrm{d}\vec{w}'$$

the reflected light at position **x** and direction **w'**

The Rendering Equation

$$L_o(x, \vec{w}) = L_e(x, \vec{w}) + \int_\Omega f_r(x, \vec{w}', \vec{w}) L_i(x, \vec{w}')(\vec{w}' \cdot \vec{n}) \mathrm{d}\vec{w}'$$

the reflected light at position **x** and direction **w'**

is equal to the integral over all possible directions **w'**

The Rendering Equation

$$L_o(x, \vec{w}) = L_e(x, \vec{w}) + \int_\Omega f_r(x, \vec{w}', \vec{w}) L_i(x, \vec{w}')(\vec{w}' \cdot \vec{n}) \mathrm{d}\vec{w}'$$

the reflected light at position **x** and direction **w'**

the incoming light from all directions

is equal to the integral over all possible directions **w'**

The Rendering Equation

$$L_o(x, \vec{w}) = L_e(x, \vec{w}) + \int_\Omega f_r(x, \vec{w}', \vec{w}) L_i(x, \vec{w}')(\vec{w}' \cdot \vec{n}) \mathrm{d}\vec{w}'$$

the reflected light at position **x** and direction **w'**

is equal to the integral over all possible directions **w'**

the incoming light from all directions

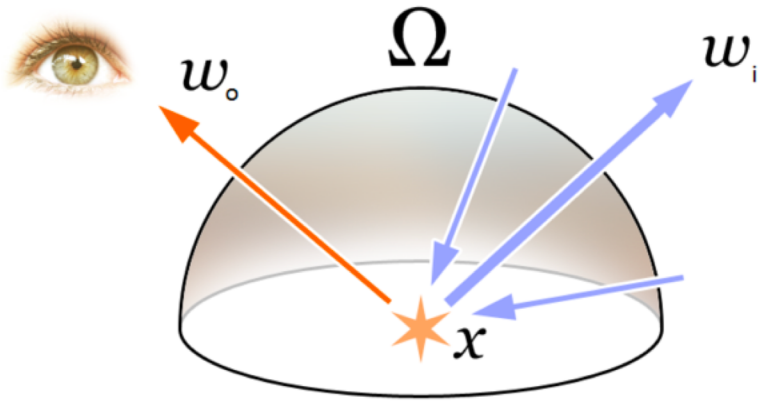
times

a function describing how light is reflected at an opaque surface

The Rendering Equation

$$L_o(x,\vec{w}) = L_e(x,\vec{w}) + \int_\Omega f_r(x,\vec{w}',\vec{w}) L_i(x,\vec{w}')(\vec{w}' \cdot \vec{n})\mathrm{d}\vec{w}'$$

the reflected light at position **x** and direction **w'**

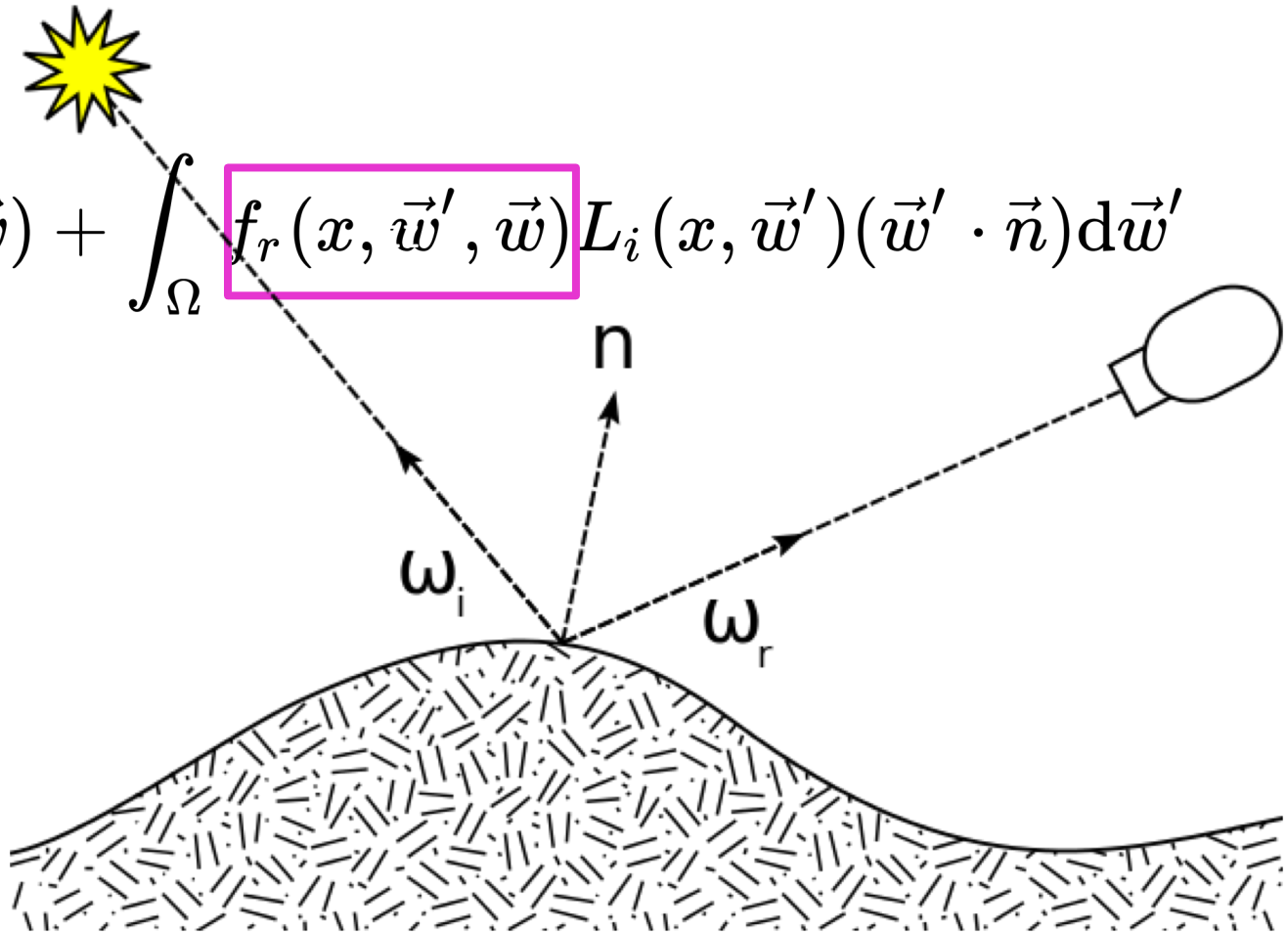


BDRFs

Bi-directional reflectance distribution function

$$L_o(x, \vec{w}) = L_e(x, \vec{w}) + \int_\Omega f_r(x, \vec{w}', \vec{w}) L_i(x, \vec{w}')(\vec{w}' \cdot \vec{n}) \mathrm{d}\vec{w}'$$

a function describing how light is reflected at an opaque surface



BDRFs

Bi-directional reflectance distribution function

$$L_o(x, \vec{w}) = L_e(x, \vec{w}) + \int_\Omega f_r(x, \vec{w}', \vec{w}) L_i(x, \vec{w}')(\vec{w}' \cdot \vec{n}) \mathrm{d}\vec{w}'$$

a function describing how light is reflected at an opaque surface

$$f_r(x, \vec{w}', \vec{w}) = \frac{\mathrm{d}L_r(x, \vec{w})}{L_i(x, \vec{w}')(\vec{w}' \cdot \vec{n})\mathrm{d}\vec{w}'}$$

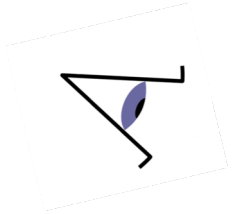
https://en.wikipedia.org/wiki/Bidirectional_reflectance_distribution_function#Models

Review: Blinn-Phong Shading Model

$$L = k_a\ I_a + \sum_{i=1}^{N} (k_d\ I_i\ \max(0, \mathbf{n} \cdot \mathbf{l_i}) + k_s\ I_i\ \max(0, \mathbf{n} \cdot \mathbf{h_i})^p)$$

Review: Blinn-Phong Shading Model

$$L = k_a\ I_a + \sum_{i=1}^{N} (k_d\ I_i\ \max(0, \mathbf{n} \cdot \mathbf{l_i}) + k_s\ I_i\ \max(0, \mathbf{n} \cdot \mathbf{h_i})^p)$$



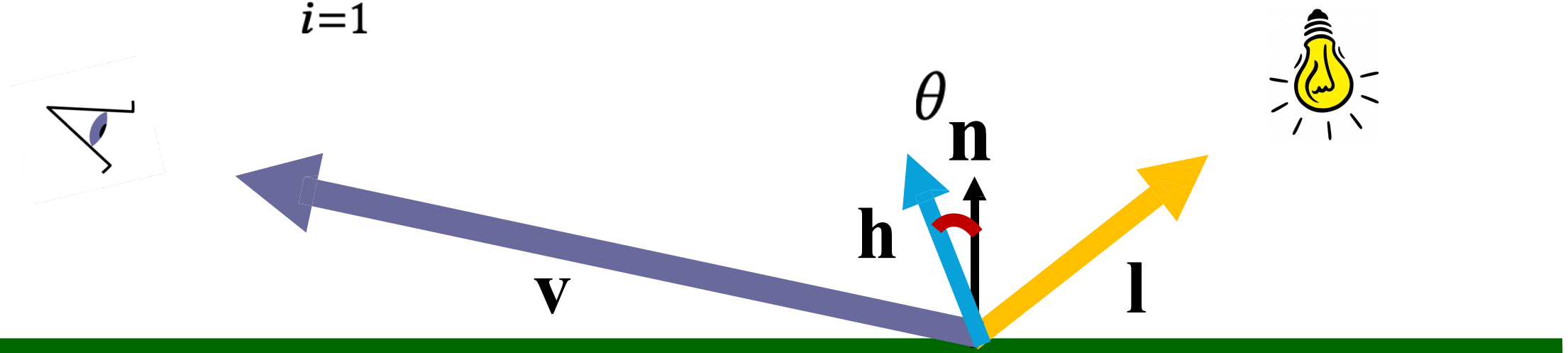
Review: Blinn-Phong Shading Model

$$L = k_a\, I_a + \sum_{i=1}^{N} (k_d\, I_i\, \max(0, \mathbf{n} \cdot \mathbf{l_i}) + k_s\, I_i\, \max(0, \mathbf{n} \cdot \mathbf{h_i})^p)$$

$$L = k_d\, I\, \max(0, \mathbf{n} \cdot \mathbf{l})$$

Review: Blinn-Phong Shading Model

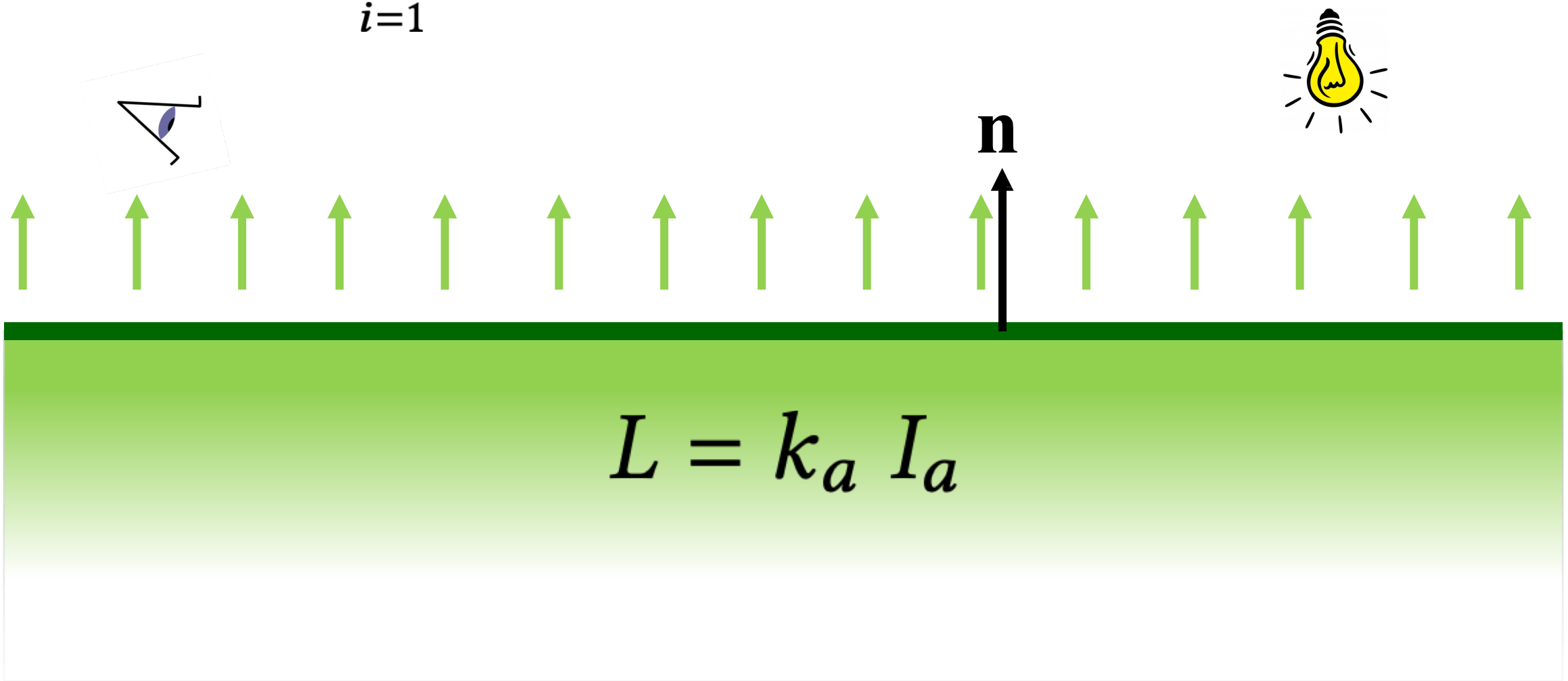
$$L = k_a\, I_a + \sum_{i=1}^{N} (k_d\, I_i\, \max(0, \mathbf{n} \cdot \mathbf{l_i}) + k_s\, I_i\, \max(0, \mathbf{n} \cdot \mathbf{h_i})^p)$$



$$L = k_s\, I\, \max(0, \mathbf{n} \cdot \mathbf{h})^p$$

Review: Blinn-Phong Shading Model

$$L = \boxed{k_a \, I_a} + \sum_{i=1}^{N} (k_d \, I_i \, \max(0, \mathbf{n} \cdot \mathbf{l_i}) + k_s \, I_i \, \max(0, \mathbf{n} \cdot \mathbf{h_i})^p)$$



Another BDRF: Cook-Torrance Shading Model

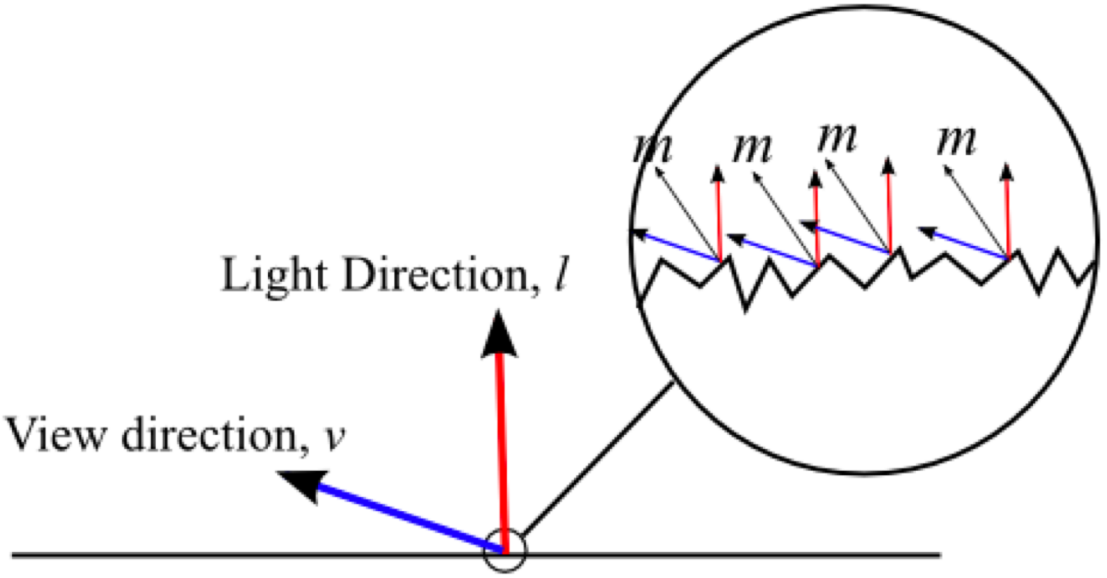
$$L = k_a\, I_a + \sum_{i=1}^{N} I_i(\mathbf{n} \cdot \mathbf{l_i})\, (k_d + k_s\, \frac{D\, G\, F}{4(\mathbf{n} \cdot \mathbf{v})(\mathbf{n} \cdot \mathbf{l_i})})$$

https://graphicscompendium.com/gamedev/15-pbr

Another BDRF: Cook-Torrance Shading Model

$$L = k_a\, I_a + \sum_{i=1}^{N} I_i(\mathbf{n}\cdot\mathbf{l_i})\,(k_d + k_s\,\frac{D\,G\,F}{4(\mathbf{n}\cdot\mathbf{v})(\mathbf{n}\cdot\mathbf{l_i})})$$

A different specular term!



https://graphicscompendium.com/gamedev/15-pbr

Another BDRF: Cook-Torrance Shading Model

$$L = k_a\, I_a + \sum_{i=1}^{N} I_i(\mathbf{n}\cdot\mathbf{l_i})\,(k_d + k_s\,\frac{D\,G\,F}{4(\mathbf{n}\cdot\mathbf{v})(\mathbf{n}\cdot\mathbf{l_i})})$$

D: Normali Distribution Function

https://graphicscompendium.com/gamedev/15-pbr

Another BDRF: Cook-Torrance Shading Model

$$L = k_a\, I_a + \sum_{i=1}^{N} I_i(\mathbf{n}\cdot\mathbf{l_i})\,(k_d + k_s\,\frac{D\,G\,F}{4(\mathbf{n}\cdot\mathbf{v})(\mathbf{n}\cdot\mathbf{l_i})})$$

D: Normali Distribution Function

G: Geometric Attenuation

https://graphicscompendium.com/gamedev/15-pbr

Another BDRF: Cook-Torrance Shading Model

$$L = k_a\, I_a + \sum_{i=1}^{N} I_i(\mathbf{n}\cdot\mathbf{l_i})\,(k_d + k_s\,\frac{D\,G\,F}{4(\mathbf{n}\cdot\mathbf{v})(\mathbf{n}\cdot\mathbf{l_i})})$$

D: Normali Distribution Function

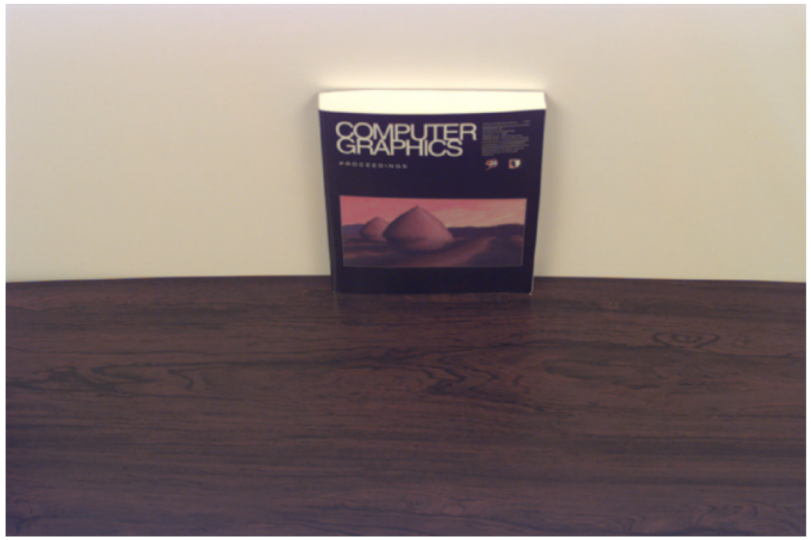
G: Geometric Attenuation

F: Fresnel Effects

https://graphicscompendium.com/gamedev/15-pbr

Another BDRF: Lafortune

To capture effects like this:



http://www.graphics.cornell.edu/pubs/1997/LFTG97.pdf

Another BDRF: Lafortune

- Generalization of Phong's reflectance model

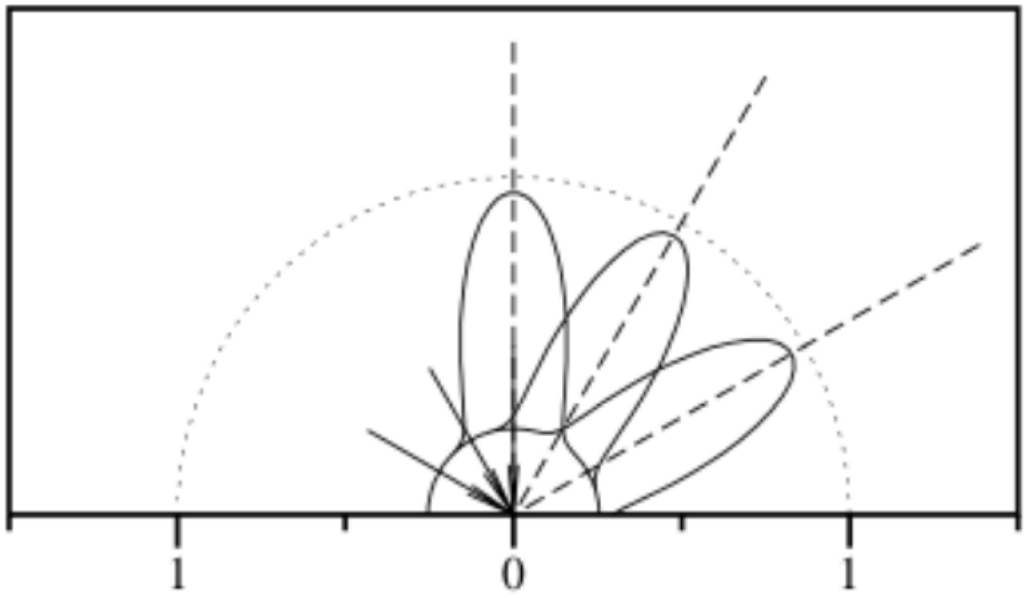
$$f_r(\omega_i, \omega_o) = C_s \left(\omega_m \cdot \omega_o\right)^n$$

- Replaces dot product with weighted dot product

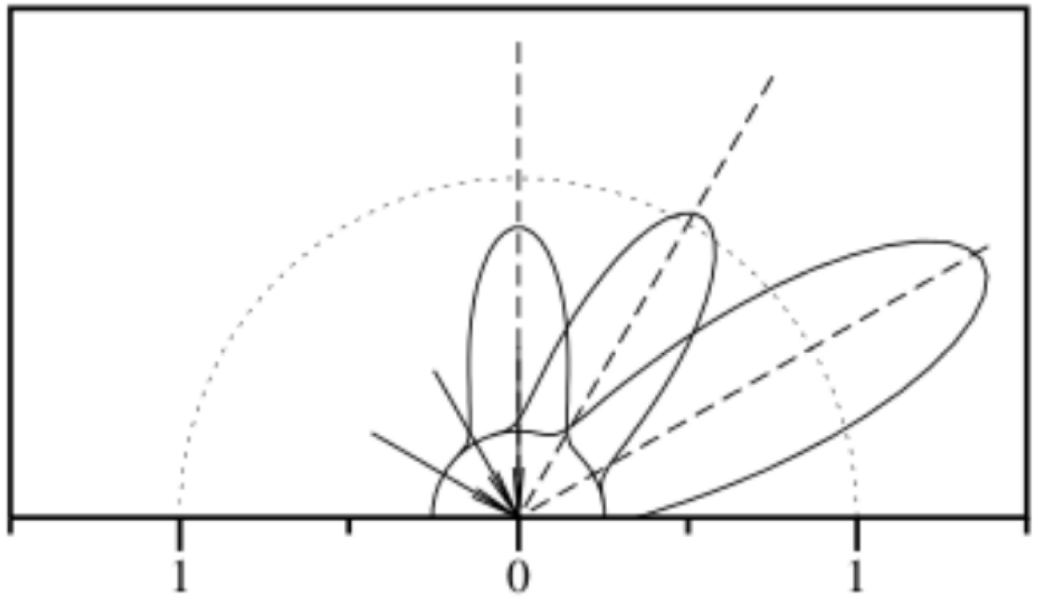
$$f_r(\omega_i, \omega_o) = \left(C_x \omega_{i,x} \omega_{o,x} + C_y \omega_{i,y} \omega_{o,y} + C_z \omega_{i,z} \omega_{o,z}\right)^n$$

http://www.graphics.cornell.edu/pubs/1997/LFTG97.pdf
https://www.cs.drexel.edu/~david/Classes/CS431/Lectures/BRDF.pdf

Another BDRF: Lafortune



Phong model

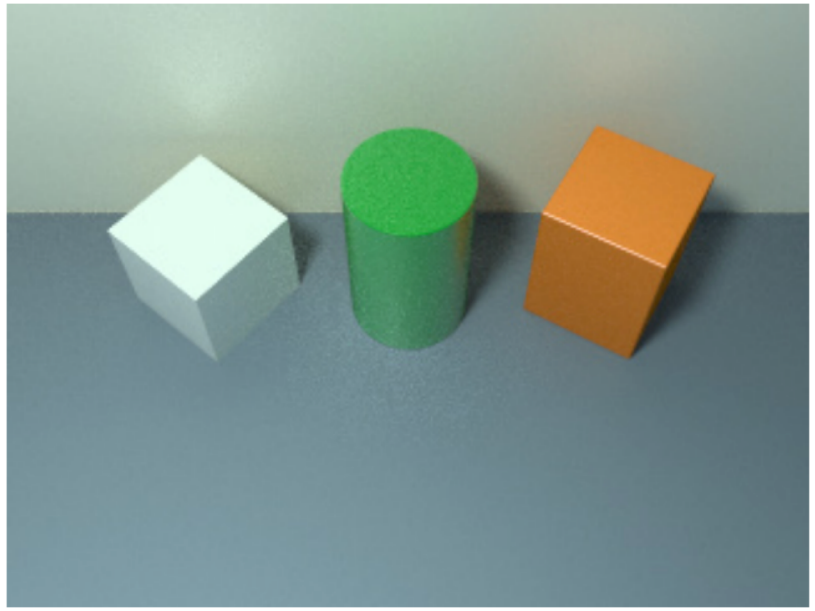
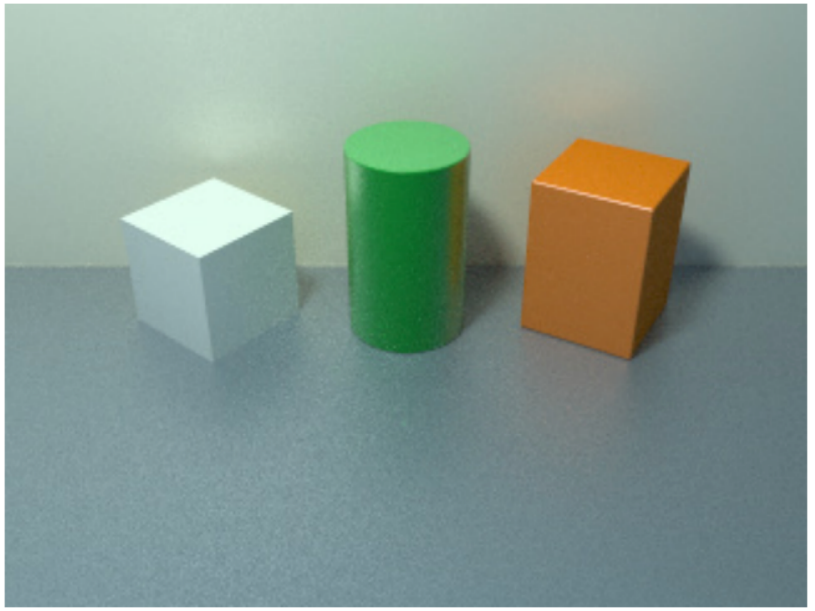
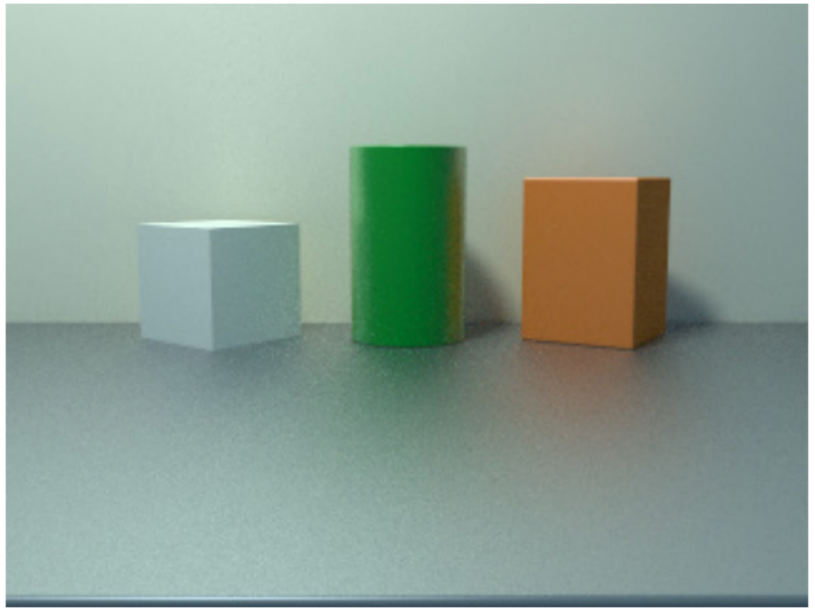


Lafortune model

http://www.graphics.cornell.edu/pubs/1997/LFTG97.pdf
https://www.cs.drexel.edu/~david/Classes/CS431/Lectures/BRDF.pdf

Another BDRF: Lafortune

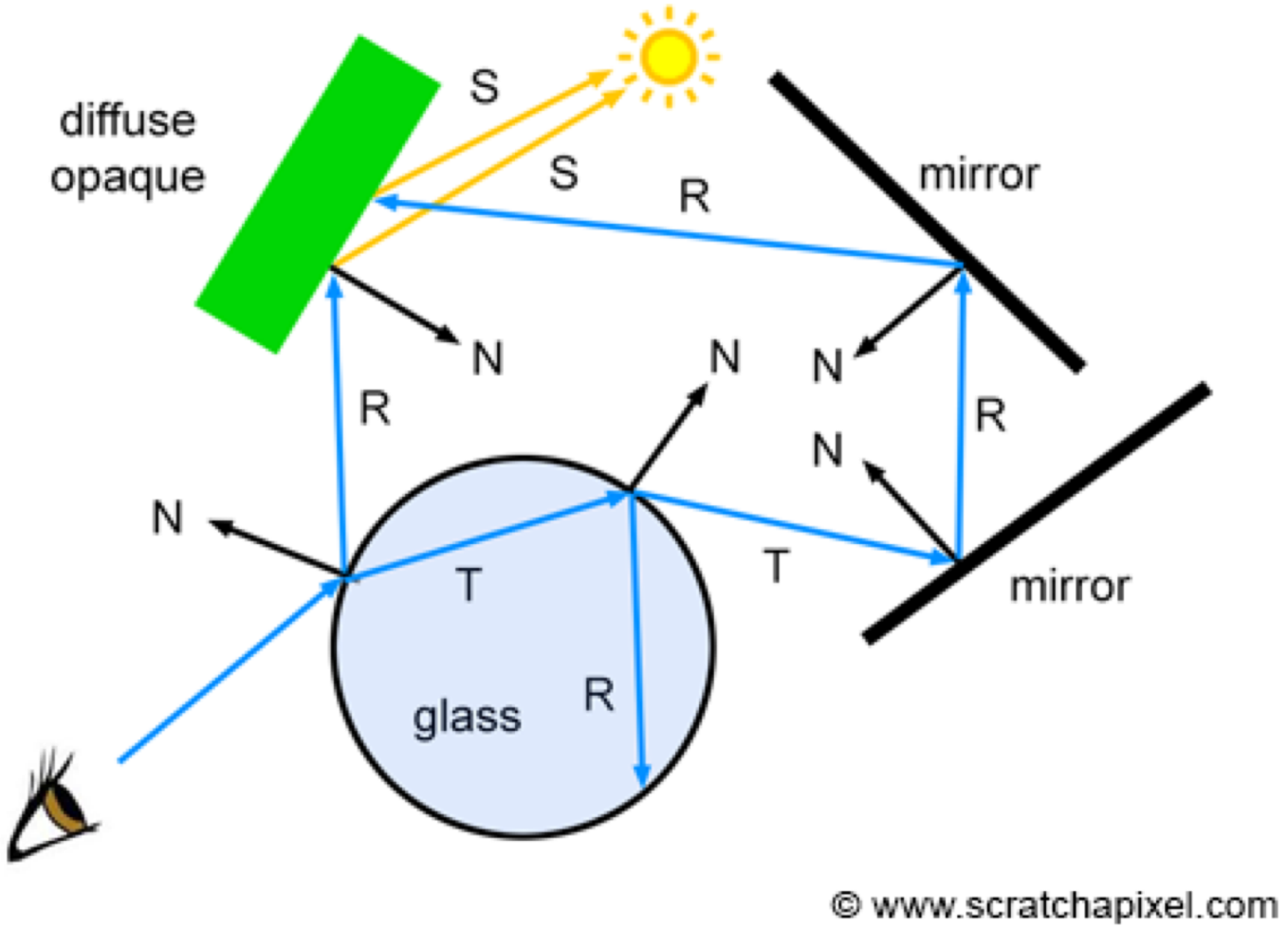
Rendered results!

http://www.graphics.cornell.edu/pubs/1997/LFTG97.pdf
https://www.cs.drexel.edu/~david/Classes/CS431/Lectures/BRDF.pdf

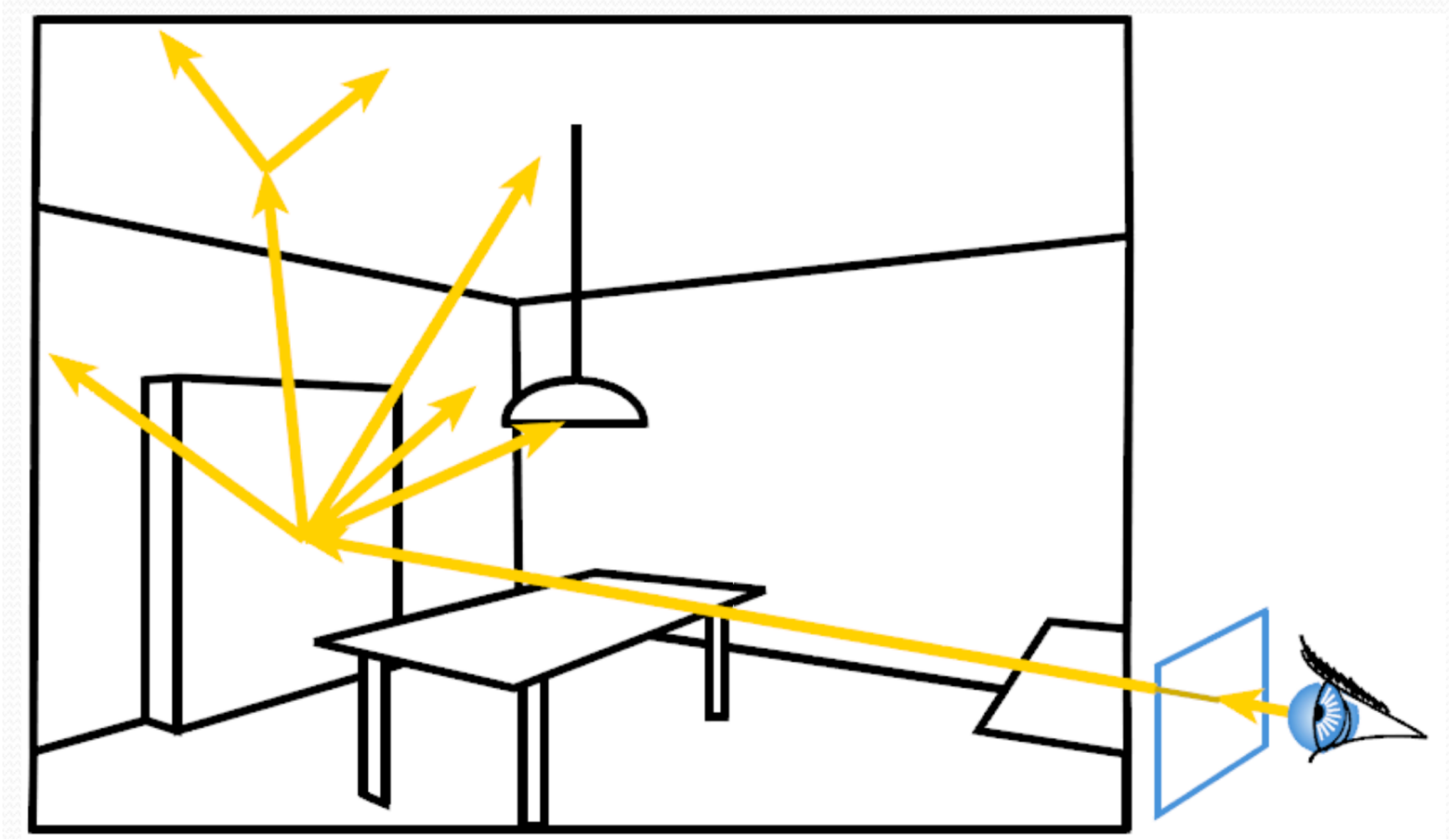
Monte Carlo Methods

Regular ray tracing is not a Monte Carlo method



Monte Carlo Methods

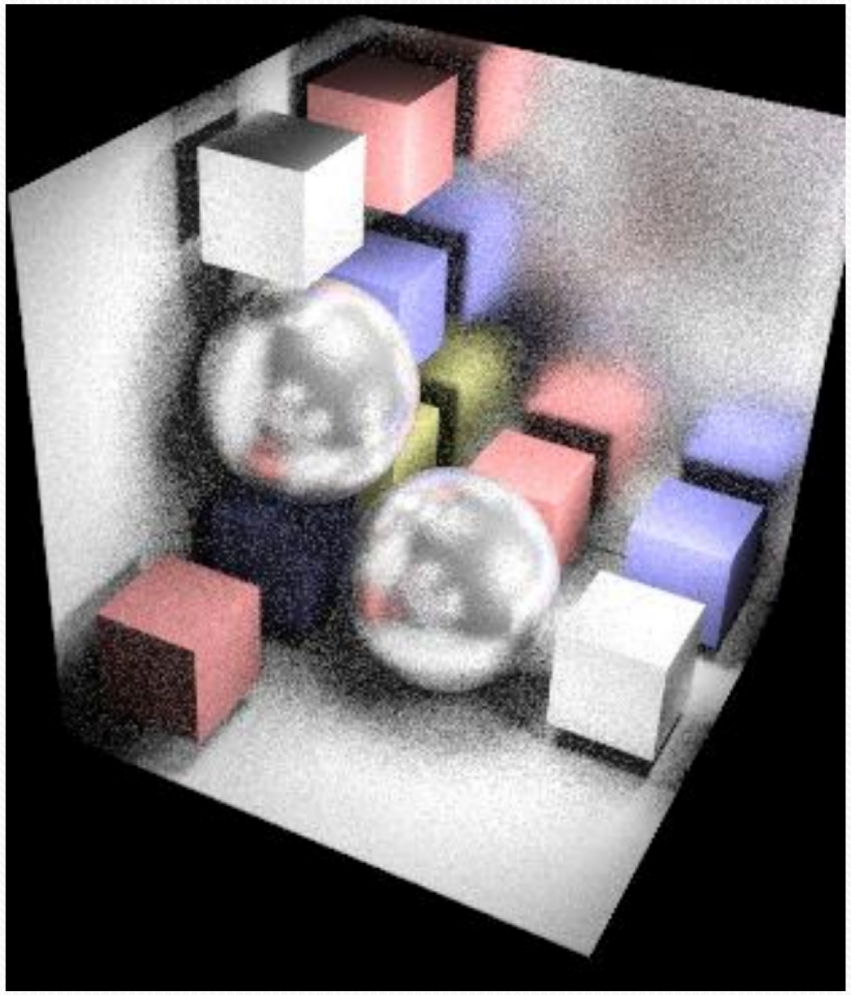
Rely on random sampling to “solve” rendering equation



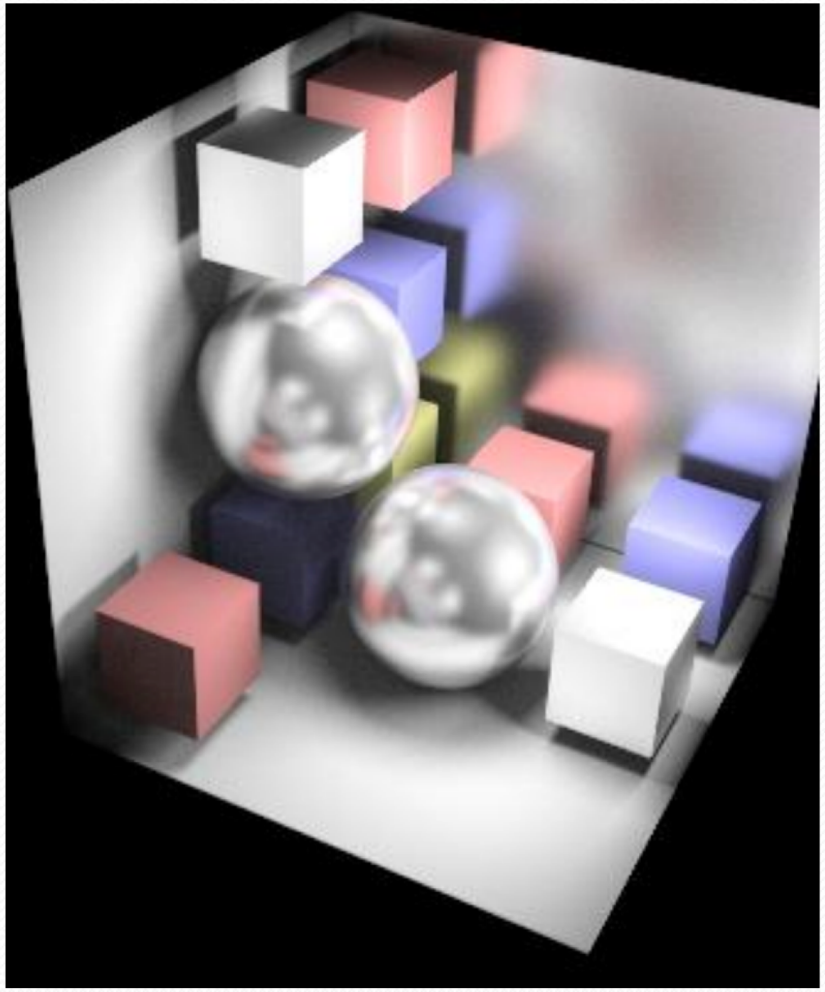
http://courses.cs.tau.ac.il/Computer-Graphics/10a/lectures/05a-RayTracingAdv.pdf

Monte Carlo Methods

16 random rays per pixel
no recursion



16 random rays per pixel
100 level recursion



Lights

Two types of lights:

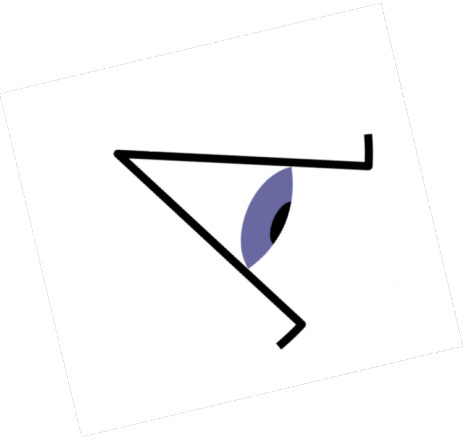
Directional Light:

Direction of light does not depend on the position of the object. **Light is very far away**

Point Light

Direction of light depends on position of object relative to light.

Directional Light



Lights

Two types of lights:

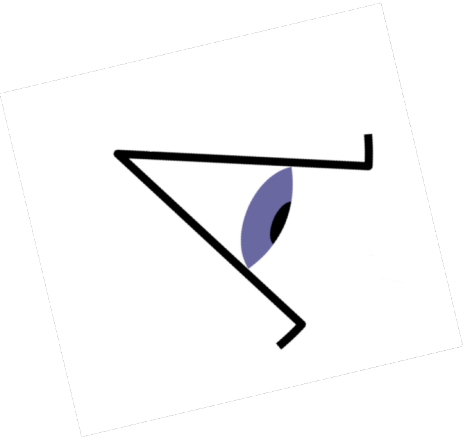
Directional Light:

Direction of light does not depend on the position of the object. **Light is very far away**

Point Light

Direction of light depends on position of object relative to light.

Point Light



Lights

Three types of lights:

Directional Light:

Direction of light does not depend on the position of the object. **Light is very far away**

Point Light

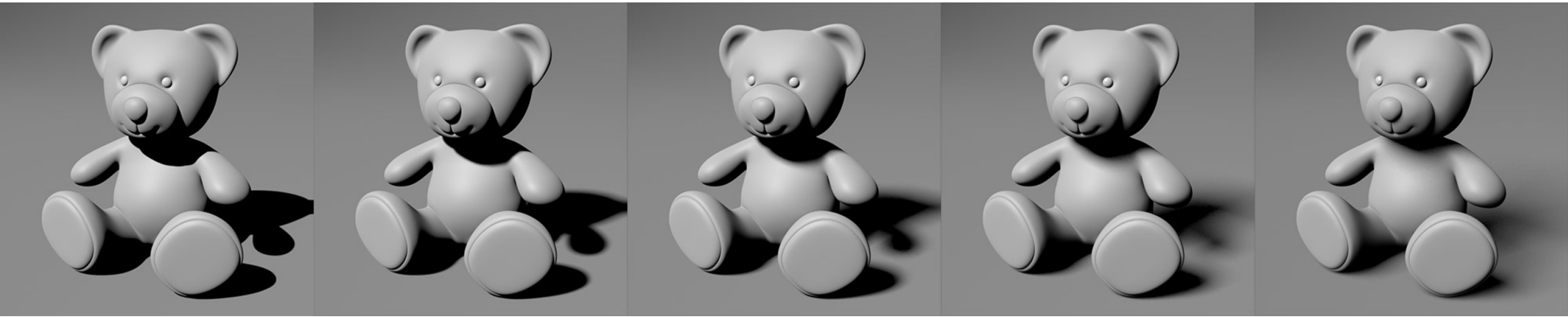
Direction of light depends on position of object relative to light.

Area Light

Light emitted from larger areas (e.g. squares, circles).

Area Light

Hard v soft shadows

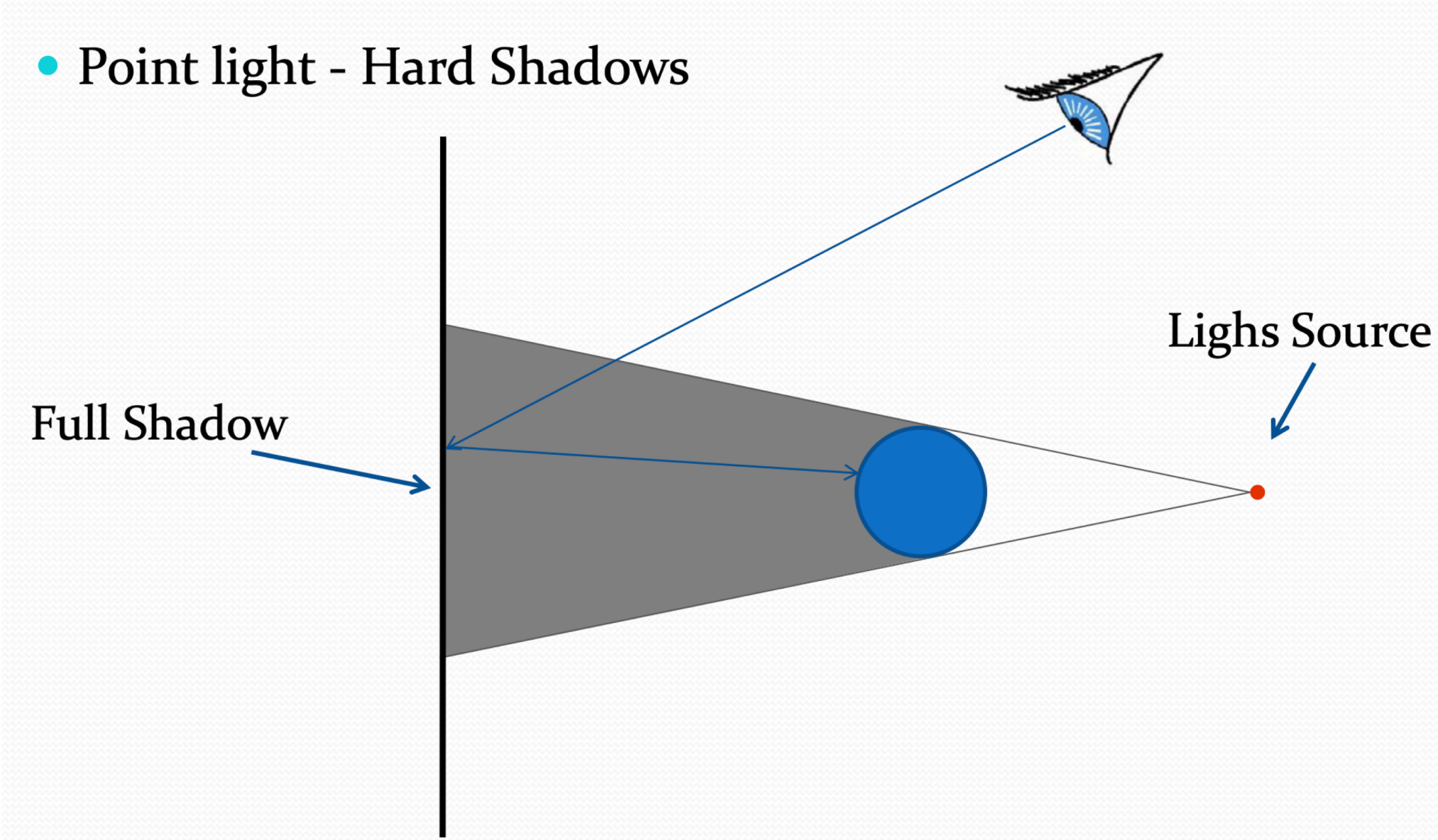


Hard shadow

More realistic soft shadows

Area Light

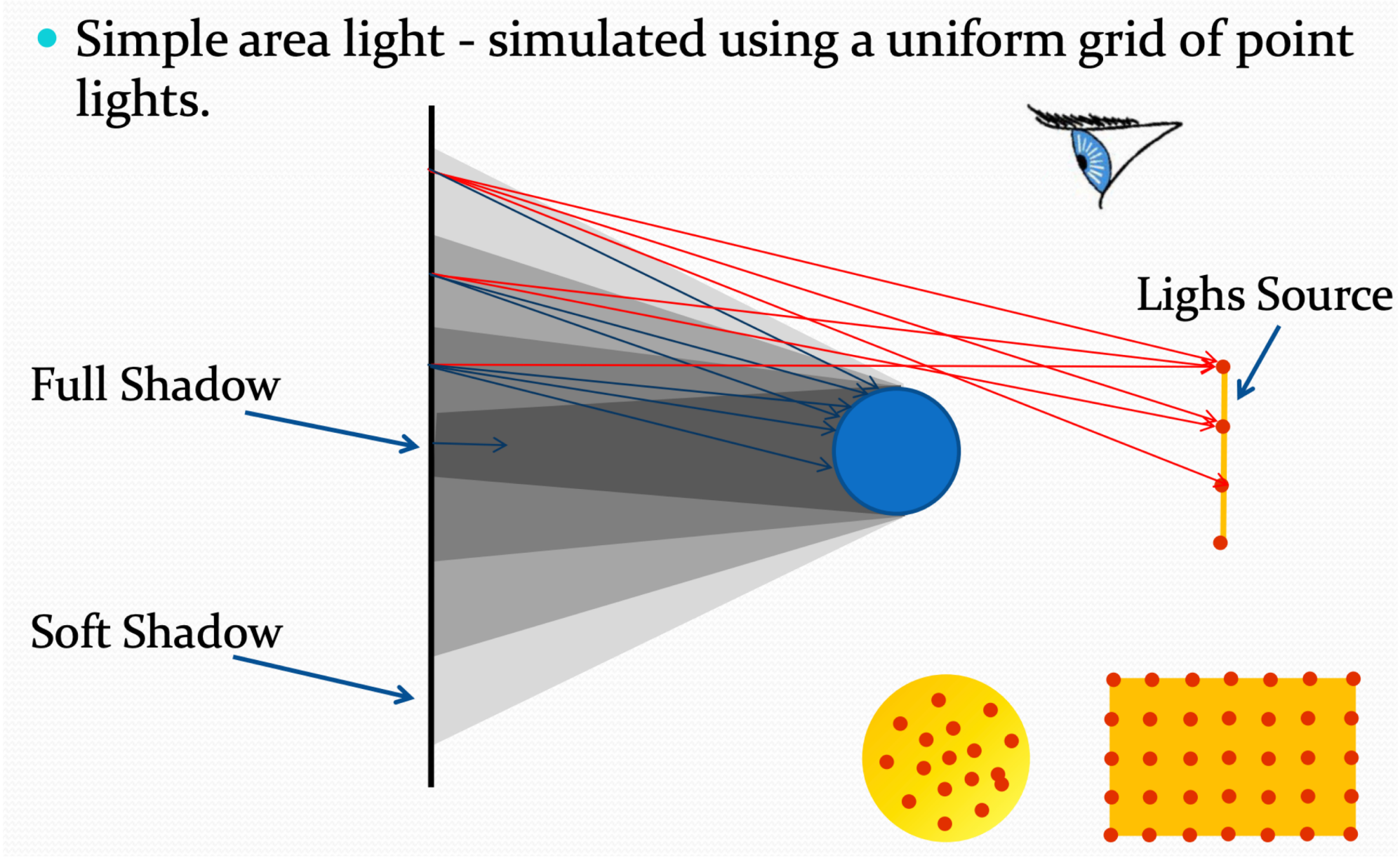
- Point light - Hard Shadows



http://courses.cs.tau.ac.il/Computer-Graphics/10a/lectures/05a-RayTracingAdv.pdf

Area Light

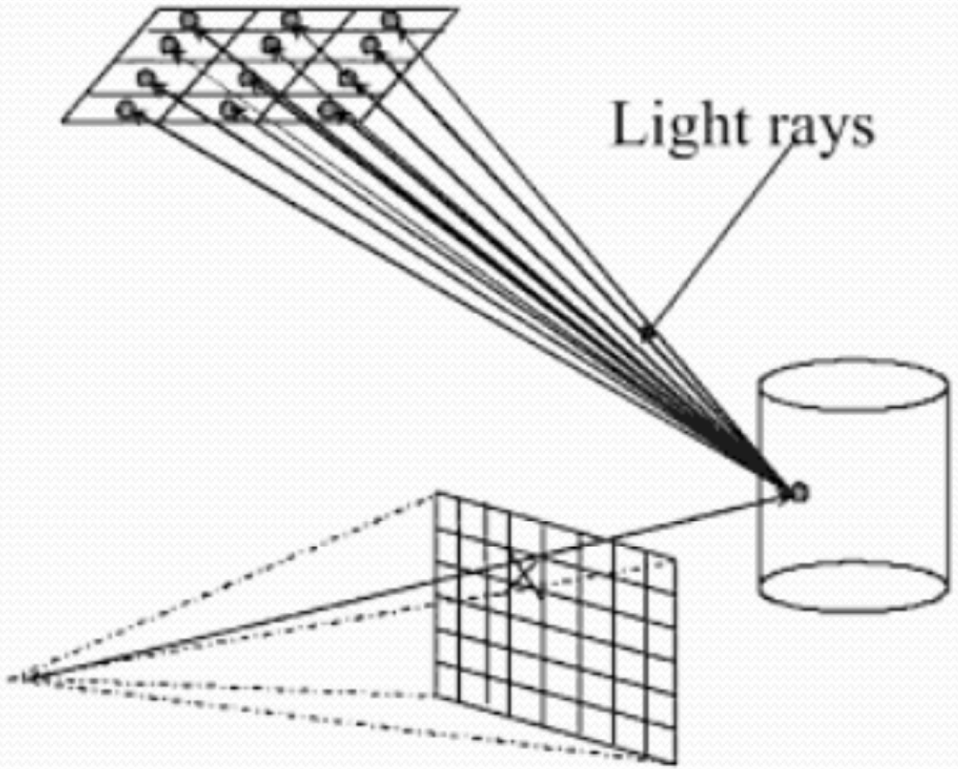
- Simple area light - simulated using a uniform grid of point lights.



http://courses.cs.tau.ac.il/Computer-Graphics/10a/lectures/05a-RayTracingAdv.pdf

Area Light

- Disadvantages of the simple uniform method:
  - Very time consuming
  - If the grid resolution is low, artifacts appear in the shadows.

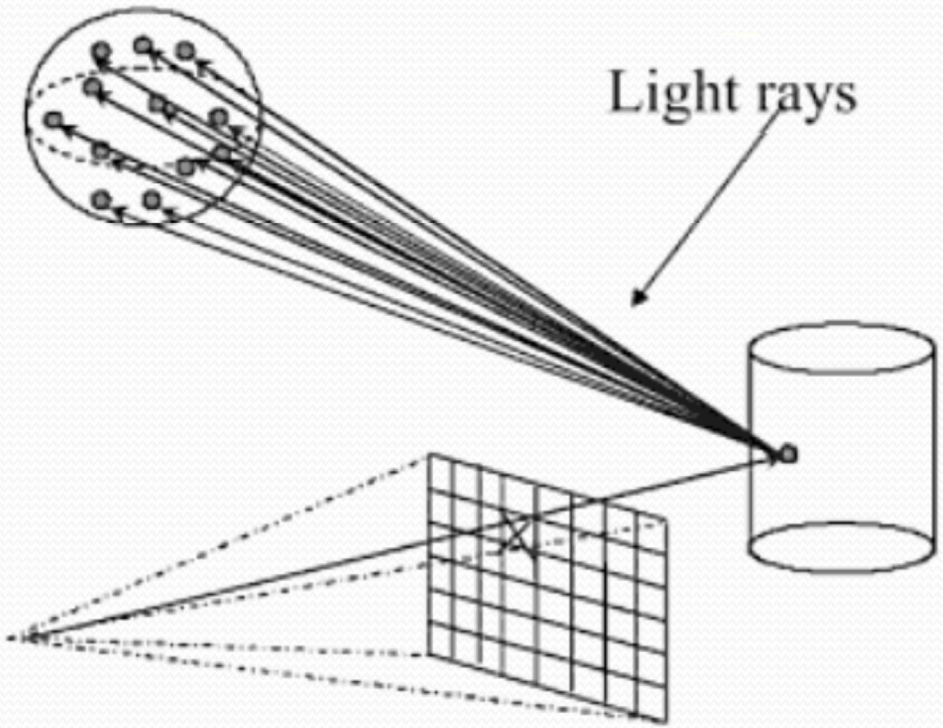


http://courses.cs.tau.ac.il/Computer-Graphics/10a/lectures/05a-RayTracingAdv.pdf

Area Light

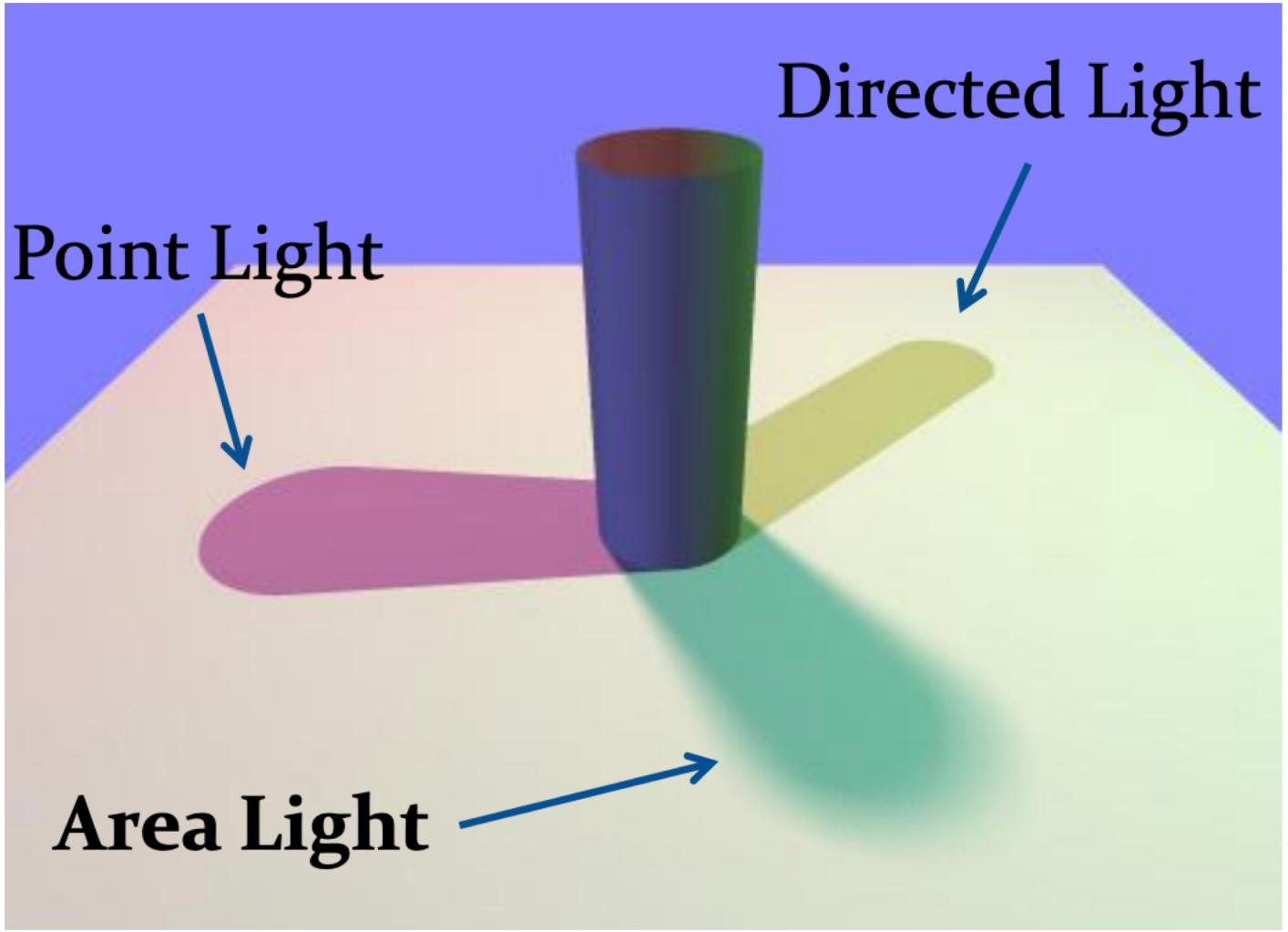
Monte-Carlo Area light

- Light is modeled as a sphere
- Highest intensity in the middle. Gradually fade out.
- Shoot n rays to random points in the sphere
- Average their value.



http://courses.cs.tau.ac.il/Computer-Graphics/10a/lectures/05a-RayTracingAdv.pdf

Area Light

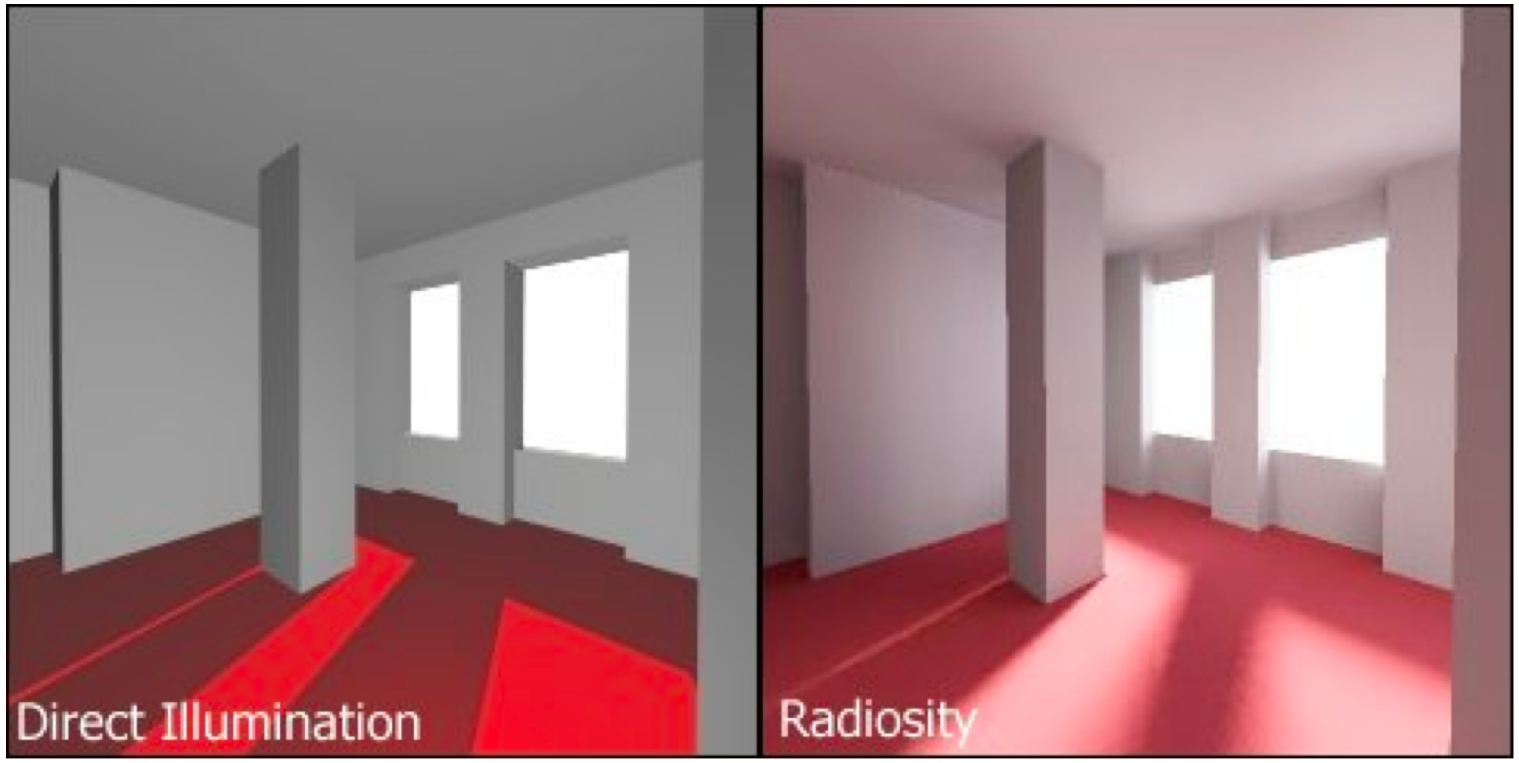


http://courses.cs.tau.ac.il/Computer-Graphics/10a/lectures/05a-RayTracingAdv.pdf

Radiosity

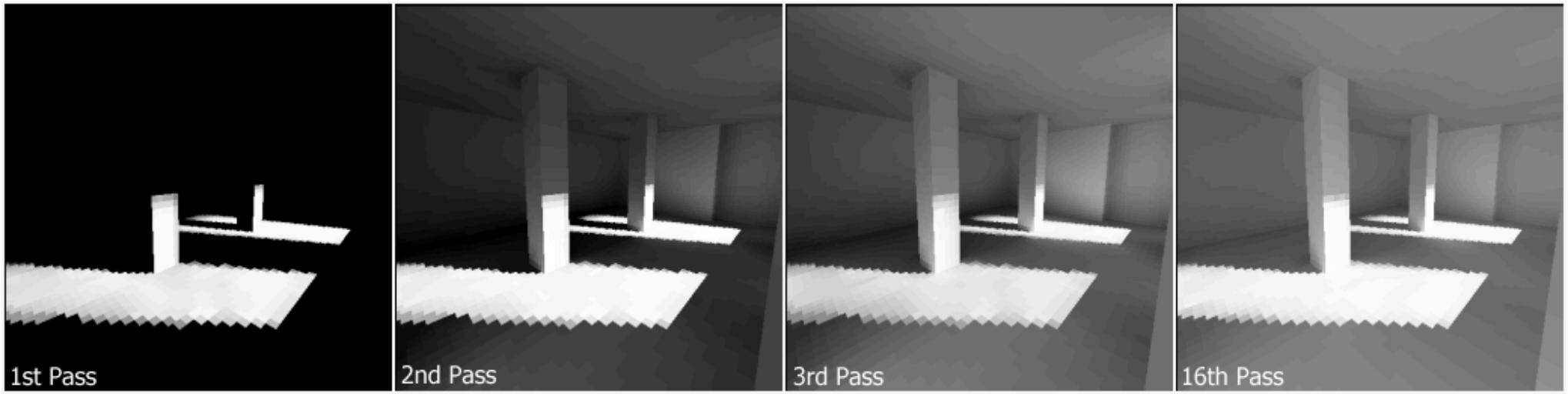
Non-Monte Carlo global illumination algorithm

View independent



https://en.wikipedia.org/wiki/Radiosity_(computer_graphics)

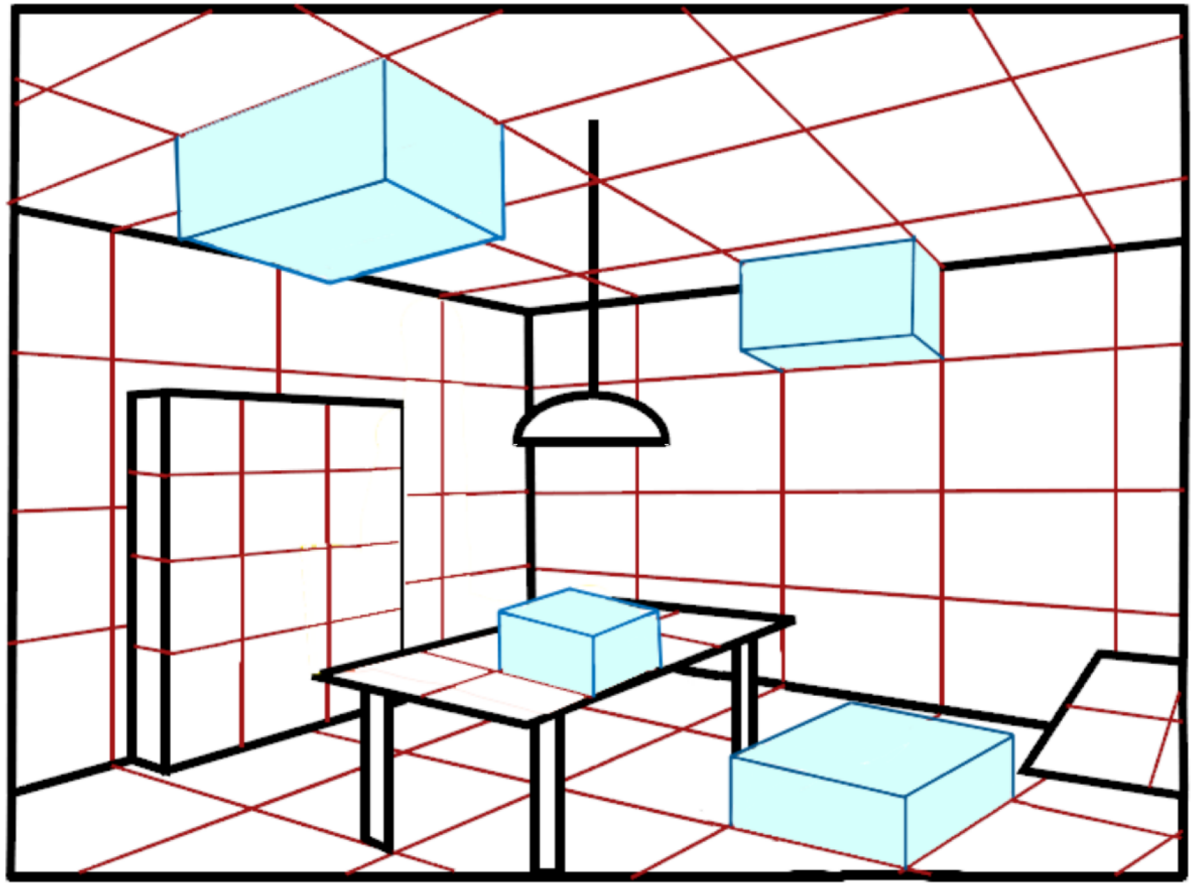
Radiosity



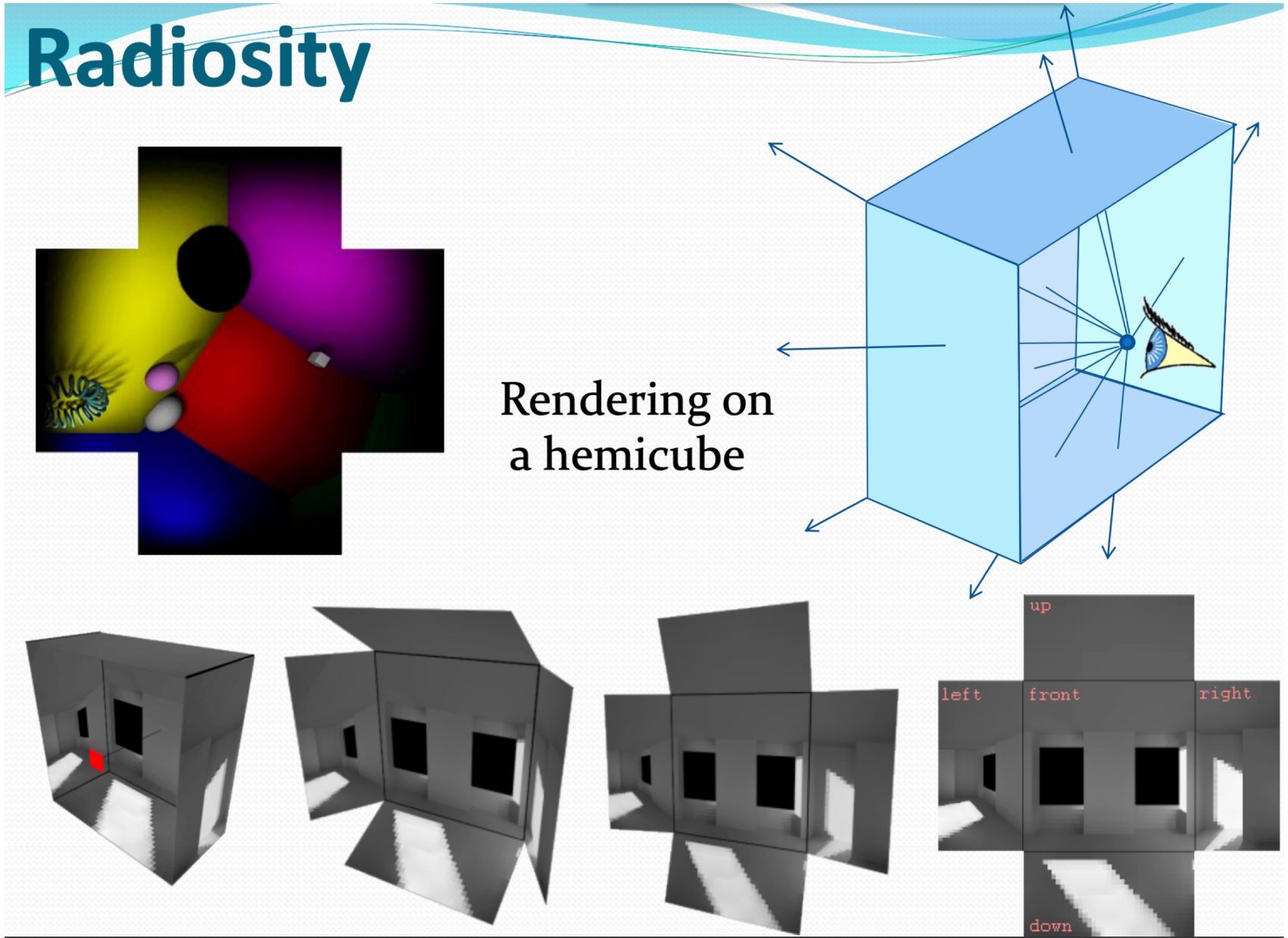
Radiosity

Calculating the amount of light a patch receives

- Construct a “Hemicube” on the patch
- Render the scene on the hemicube
- Average the color.



Radiosity



Speedy Ray Tracing

Ray sorting

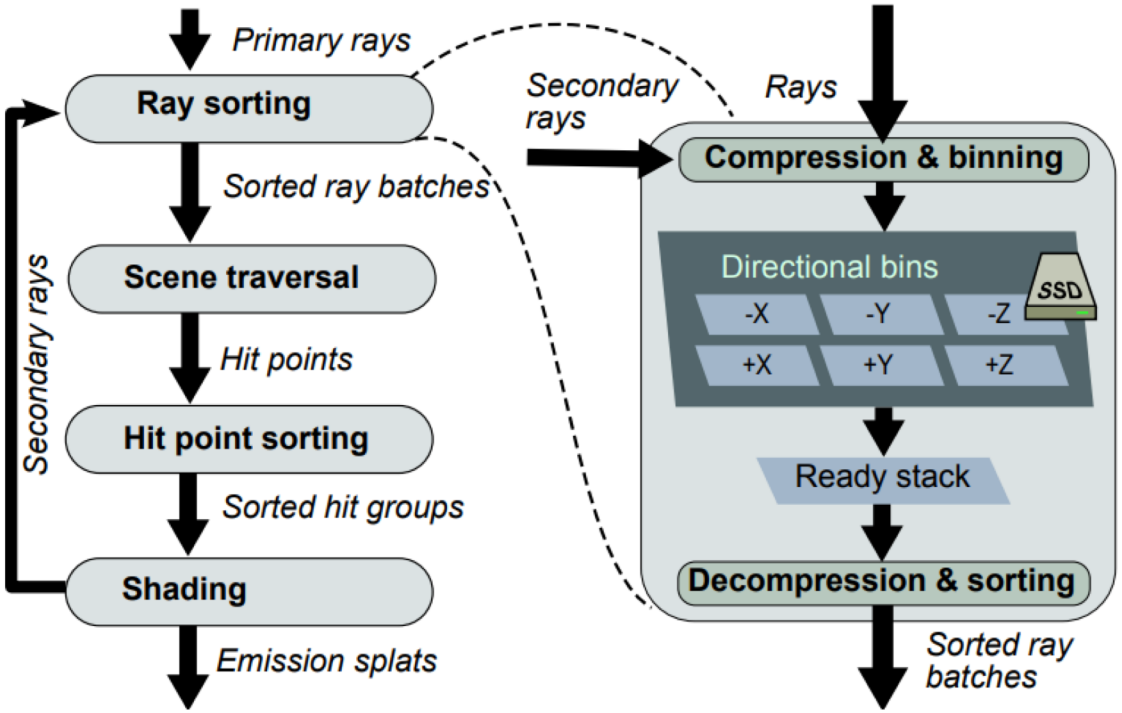


Figure 2: *Left: We ensure coherent path tracing using two sorting stages. Right: An exploded view of ray sorting.*

https://www.disneyanimation.com/technology/innovations/hyperion
https://www.youtube.com/watch?v=QTA2Qt5Wk-A&feature=emb_title
https://disney-animation.s3.amazonaws.com/uploads/production/publication_asset/70/asset/Sorted_Deferred_Shading_For_Production_Path_Tracing.pdf

Speedy Ray Tracing

Speedy Ray Tracing

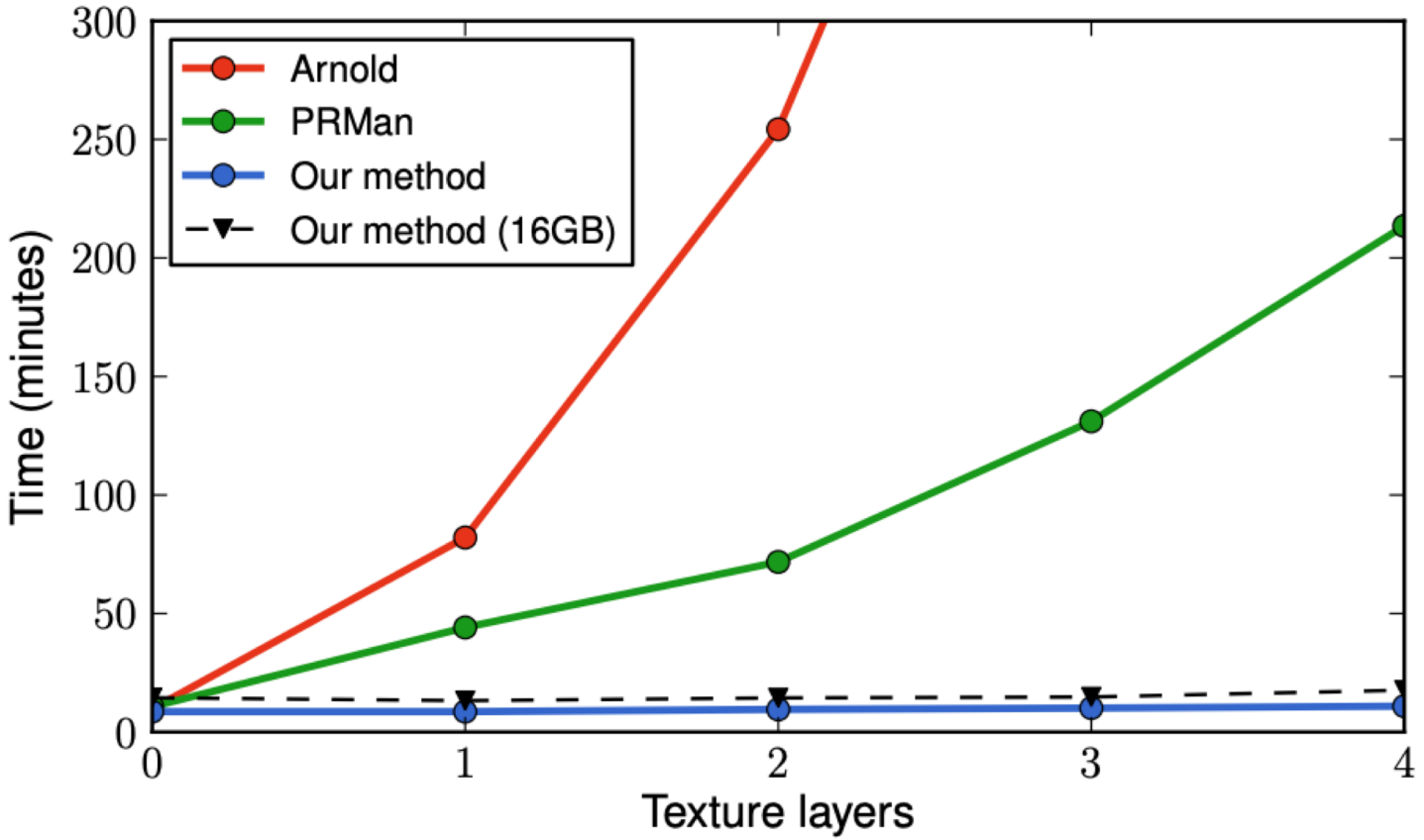
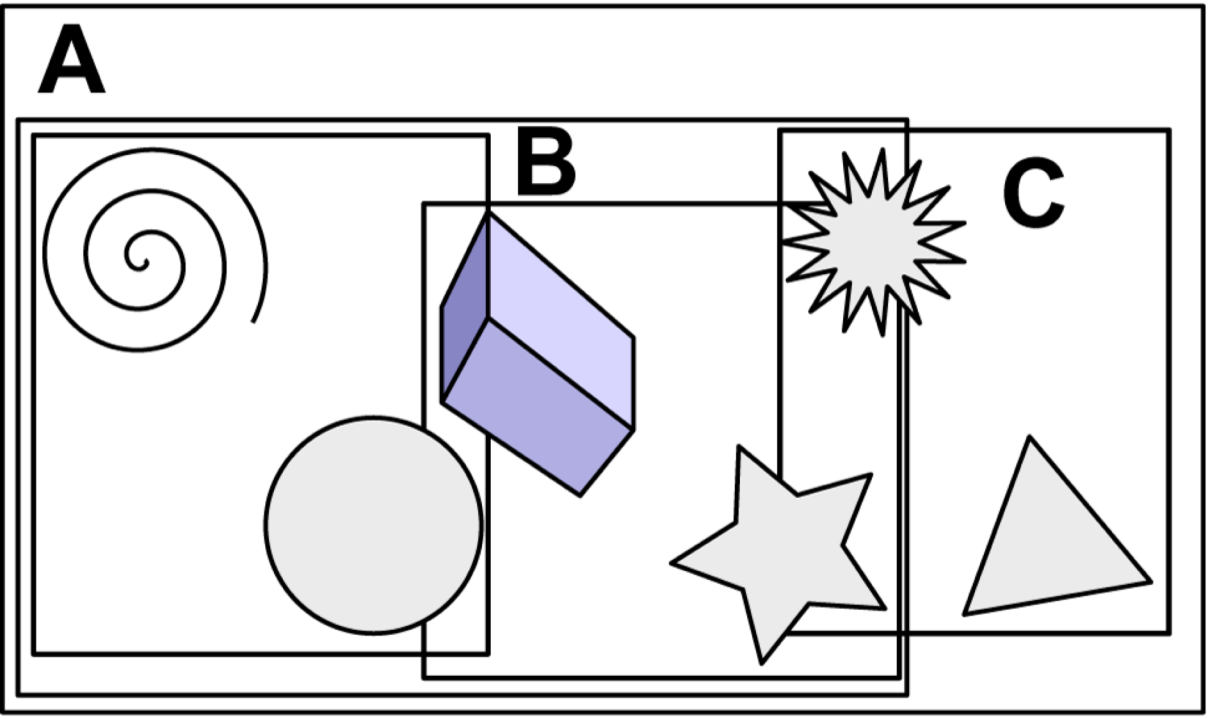


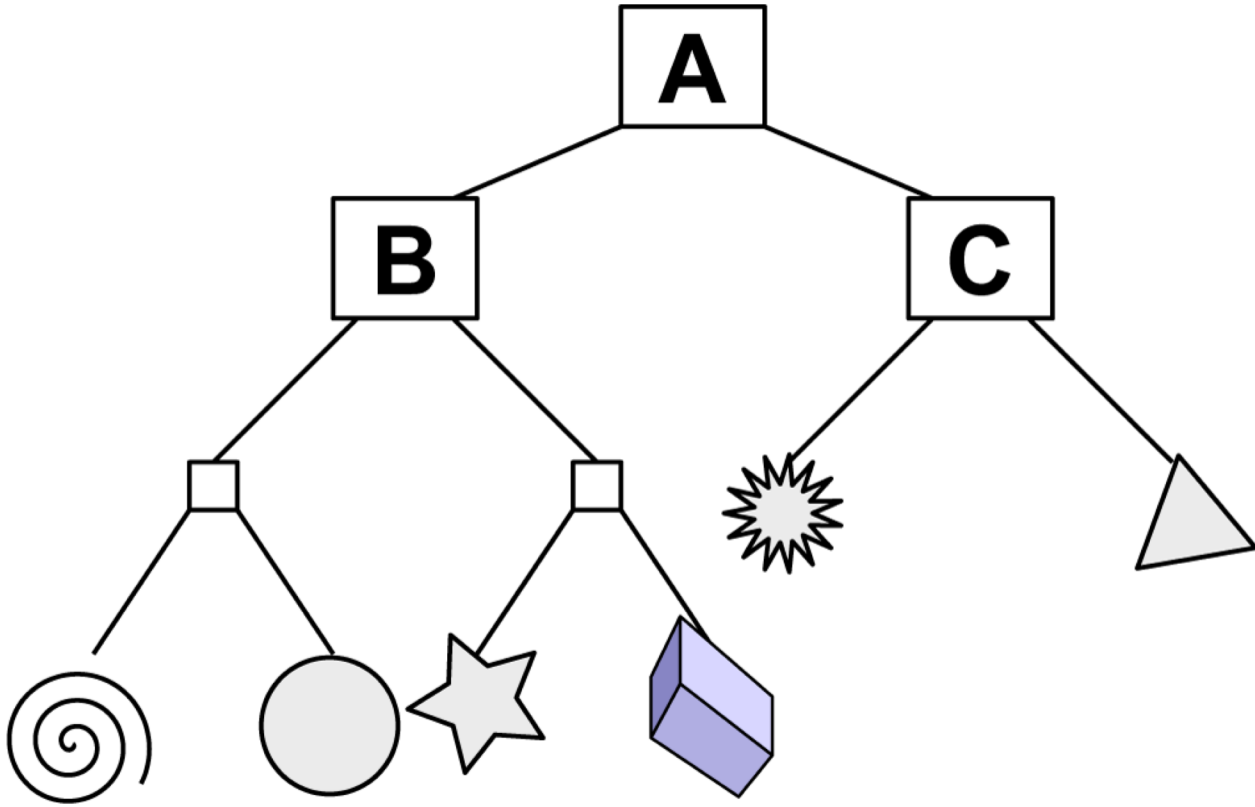
Figure 9: *Comparing texture performance of our method against production renderers on the interior scene.*

Speedy Ray Tracing

Speedy Ray Tracing

Bounding volumes (more on this next week!)

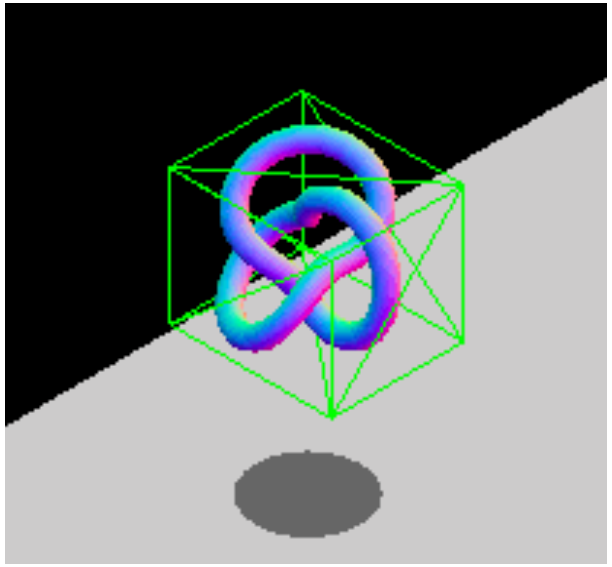




https://en.wikipedia.org/wiki/Bounding_volume

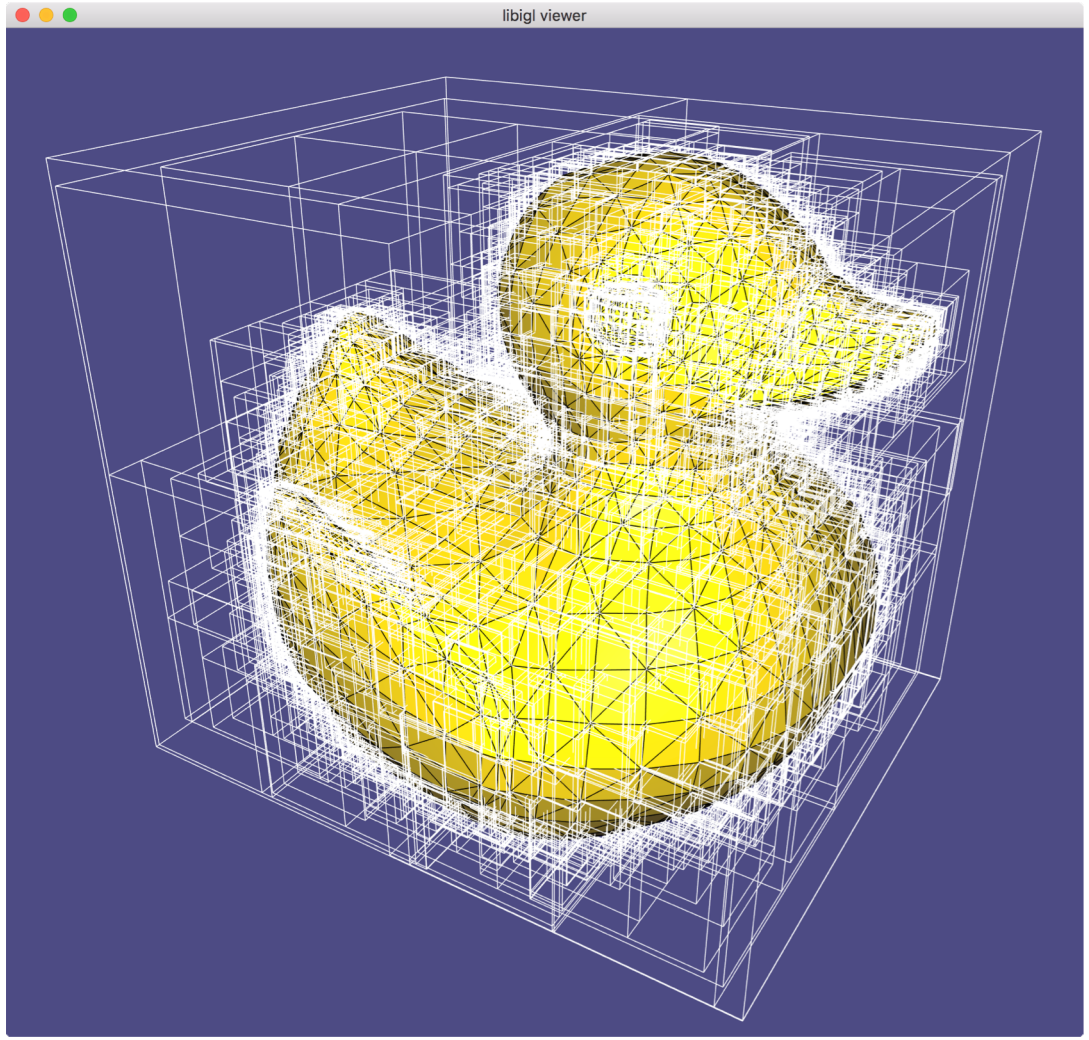
Speedy Ray Tracing

Also used for collision detection in games (hit boxes)



Speedy Ray Tracing

No need to loop over every triangle for intersection testing



Done for today