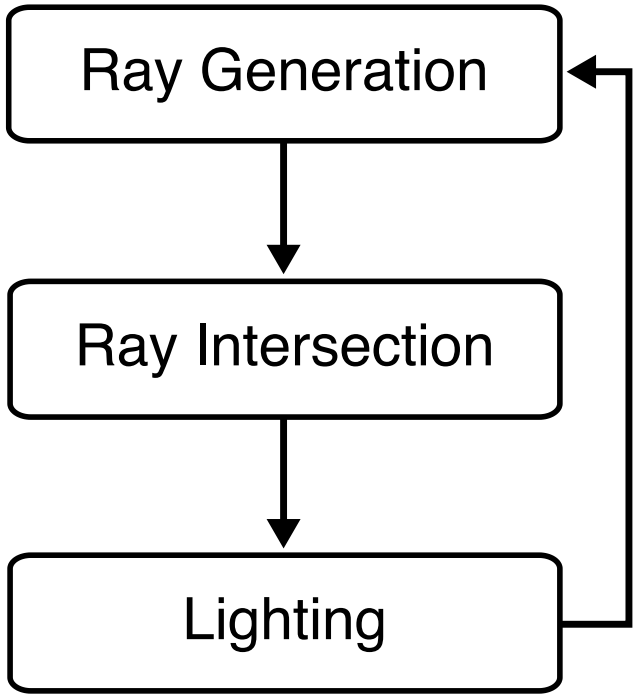
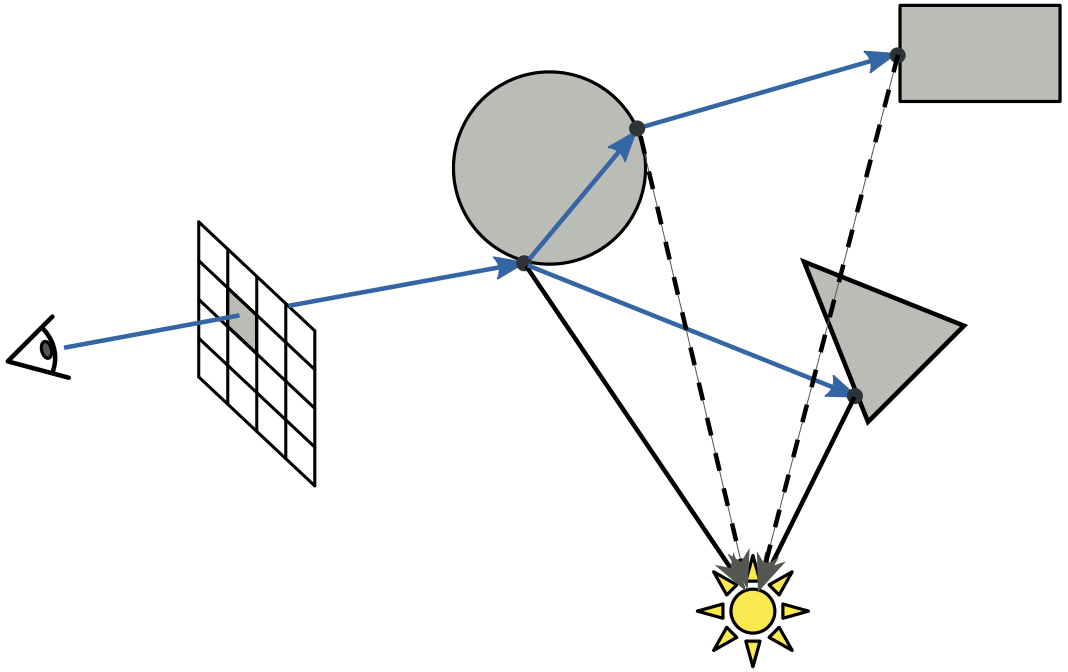
Lecture Computer Graphics

Lighting

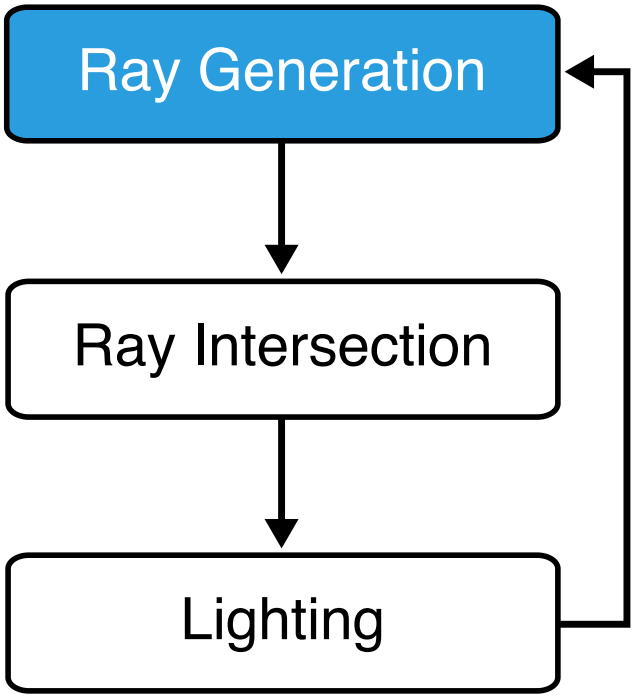
Prof. Dr. David Bommes
Computer Graphics Group

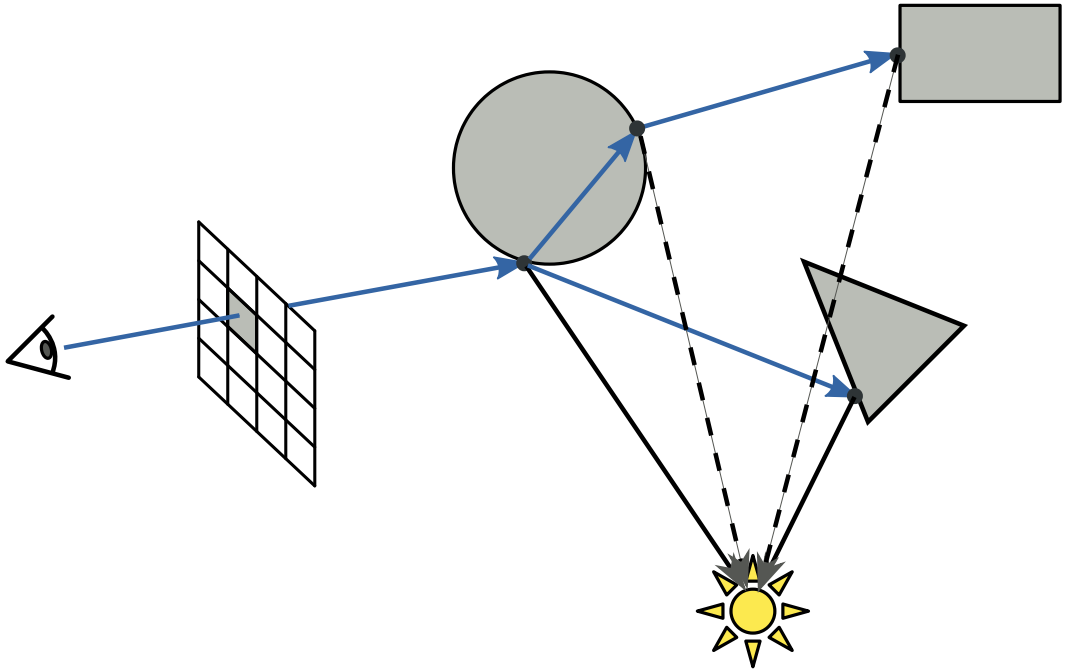
Ray Tracing Operators





Ray Tracing Operators

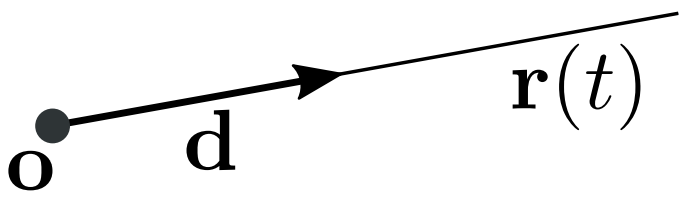




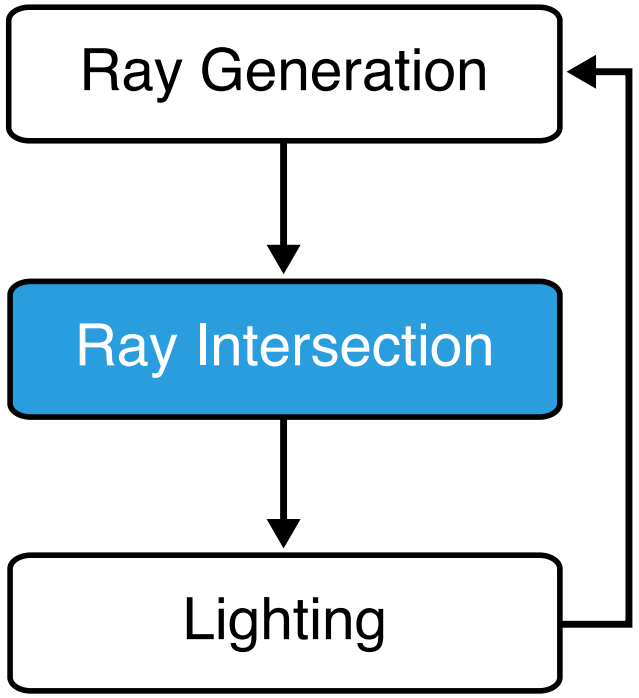
Ray Equation

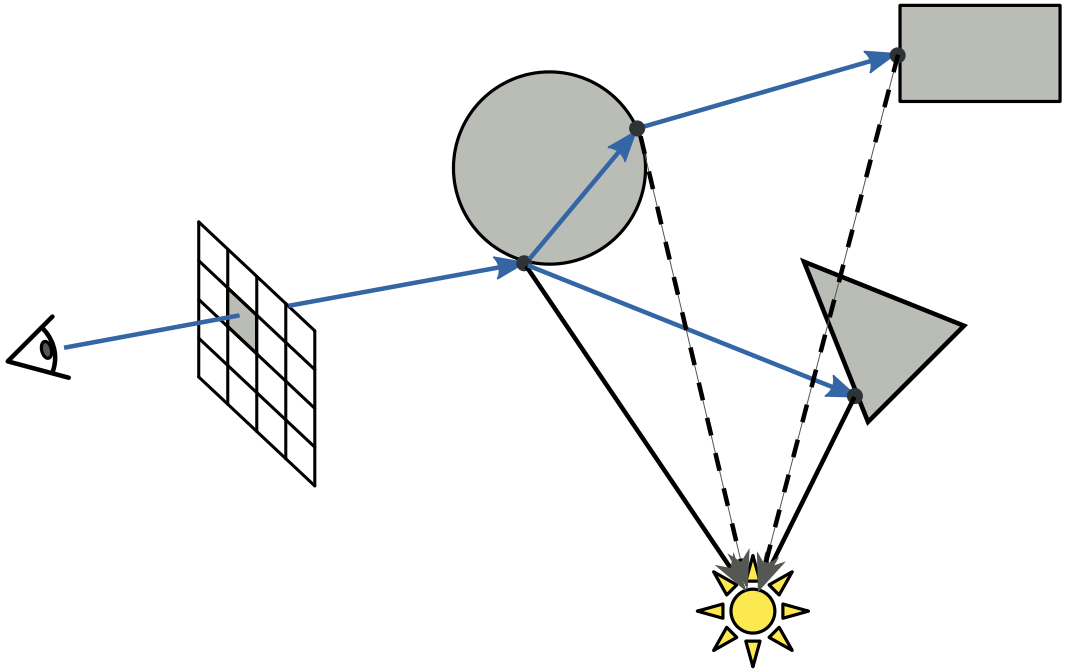
- Explicit equation for ray $\mathbf{r}(t)$, starting at origin $\mathbf{o}$ and going in (normalized) direction $\mathbf{d}$:

$$\mathbf{r}(t) = \mathbf{o} + t\,\mathbf{d}$$



Ray Tracing Operators

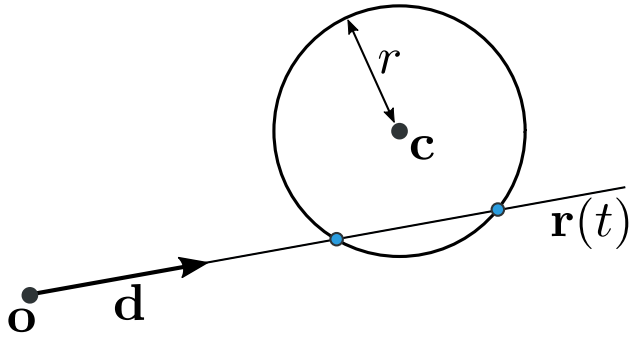




Ray-Sphere Intersection

- Implicit representation of a sphere with center $\mathbf{c}$ and radius r:

$$\|\mathbf{x} - \mathbf{c}\| - r = 0$$



- Insert *explicit* ray equation into *implicit* sphere equation:

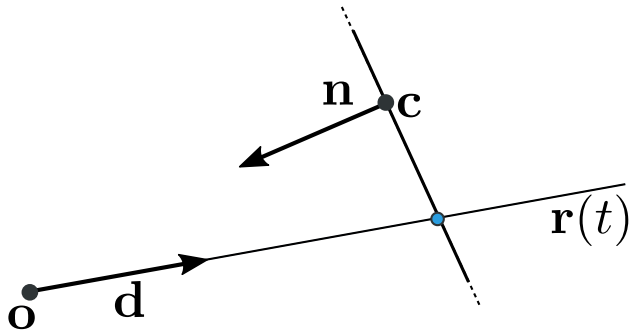
$$\|\mathbf{o} + t\mathbf{d} - \mathbf{c}\| - r = 0$$

and solve for t.

Ray-Plane Equation

- Implicit equation for a plane with normal $\mathbf{n}$ and distance d from origin:

$$\mathbf{n}^\mathsf{T}\mathbf{x} - d = 0$$



- Insert *explicit* ray equation into *implicit* plane equation

$$\mathbf{n}^\mathsf{T}\left(\mathbf{o} + t\,\mathbf{d}\right) - d = 0$$

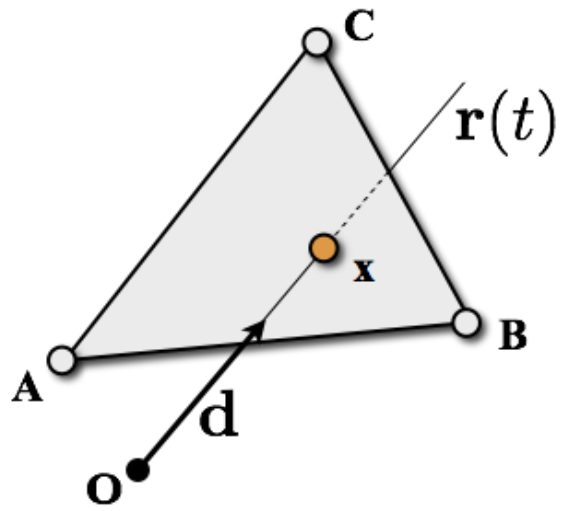
and solve for t.

Ray-Triangle Intersection (explicit)

- Ray and triangle have to coincide

$$\mathbf{o} + t\,\mathbf{d} \;=\; \alpha\mathbf{A} + \beta\mathbf{B} + \gamma\mathbf{C}$$

Four unknowns, but only three equations!

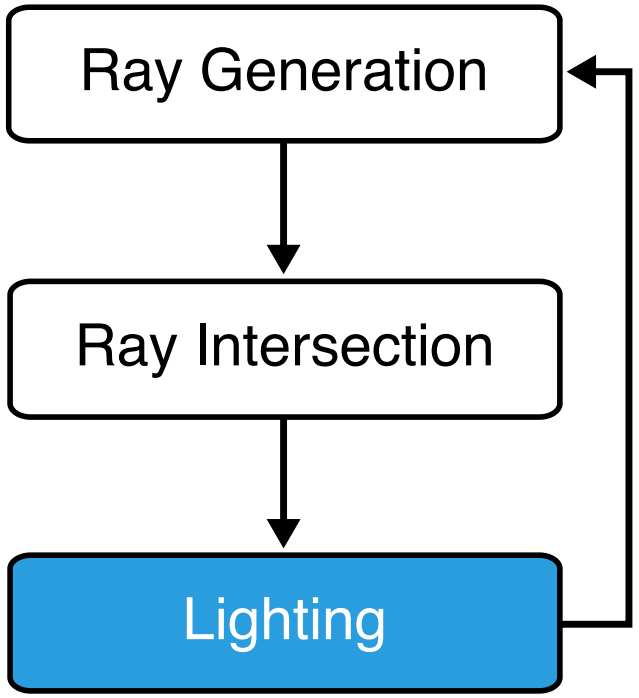


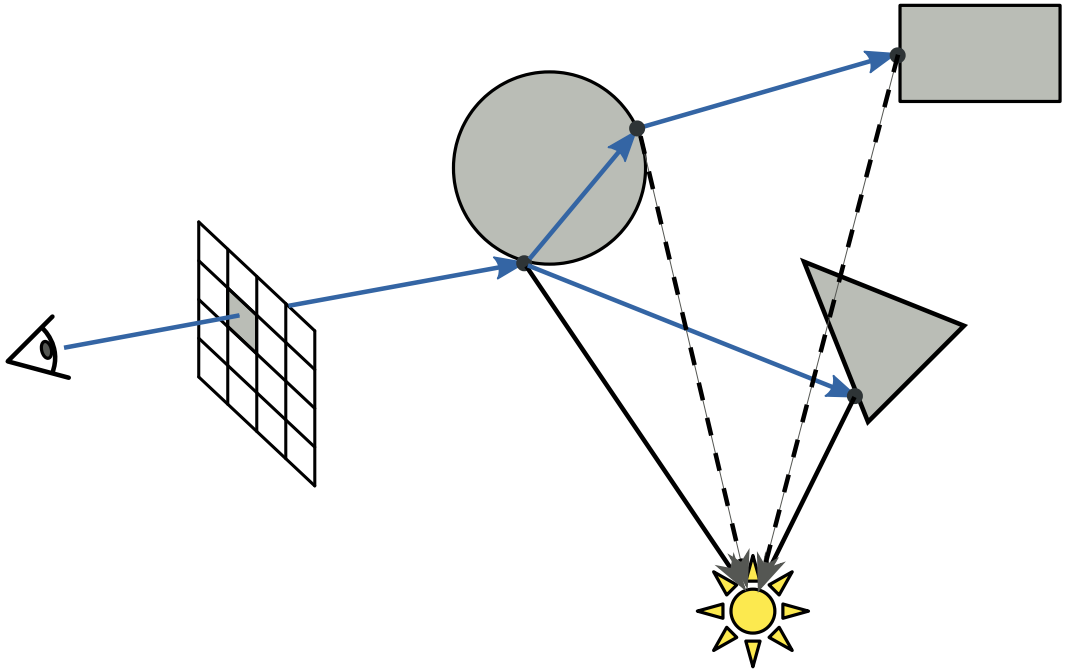
- Exploit condition $\alpha + \beta + \gamma = 1$ to eleminate α

$$\mathbf{o} + t\,\mathbf{d} \;=\; (1 - \beta - \gamma)\mathbf{A} + \beta\mathbf{B} + \gamma\mathbf{C}$$

Solve 3×3 linear system (Cramer's rule)

Ray Tracing Operators





Quiz Setup

http://cgg.unibe.ch:8080

Quiz: Pinhole Camera

What happens when we replace a pin-hole by a finite aperture?

A: The image is blurred everywhere.

A: The image is the same.

A: The image is in perfect focus at some depth, but blurred elsewhere.

A: The image is brighter.

Quiz: Pinhole Camera

Which of the following are true for a camera obscura?

A: Two objects of the same size at the same distance from the pinhole will have the same size on the image plane.

A: Moving the image plane further away from the pinhole reduces the view angle.

A: The image on the image plane is up-side down.

A: A camera obscura was first built by the ancient greeks.

Quiz: Vector Stuff

When are two vectors $\mathbf{a}$ and $\mathbf{b}$ perpendicular?

A: If $\mathbf{a} \times \mathbf{b} = 0$.

A: If $\mathbf{a} \cdot \mathbf{b} \neq 0$.

A: If $\mathbf{a} \times \mathbf{b} \neq 0$.

A: If $\mathbf{a} \cdot \mathbf{b} = 0$.

Quiz: Plane

How many degrees of freedom (DoFs) does a plane have?

A: Three

A: Four

A: Five

A: Six

Quiz: Distance to Plane

How to compute the distance of point $\mathbf{x}$ from a plane with normal vector $\mathbf{n}$ and centered at point $\mathbf{c}$?

A: $\langle \mathbf{x} - \mathbf{c}, \mathbf{n} \rangle$

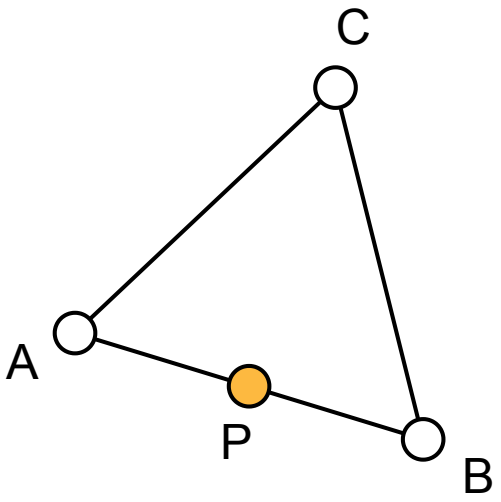
A: $\|\mathbf{x} - \mathbf{c}\| - \mathbf{n}$

A: $\mathbf{n}^\top \mathbf{x} - \mathbf{c}$

A: $\mathbf{x} \cdot \mathbf{n} - \mathbf{c} \cdot \mathbf{n}$

Quiz: Barycentric Coordinates

What are the barycentric coordinates (α, β, γ) of point $P = \alpha A + \beta B + \gamma C$?



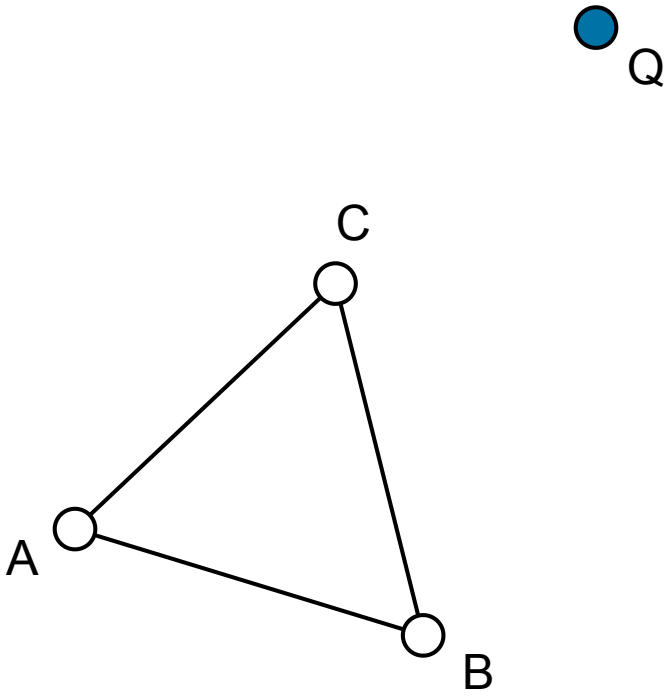
A: (0, 0.5, 0.5)

A: (1, 1, 0)

A: (0.5, 0.5, 0)

Quiz: Barycentric Coordinates

What are the barycentric coordinates (α, β, γ) of point $Q = \alpha A + \beta B + \gamma C$?



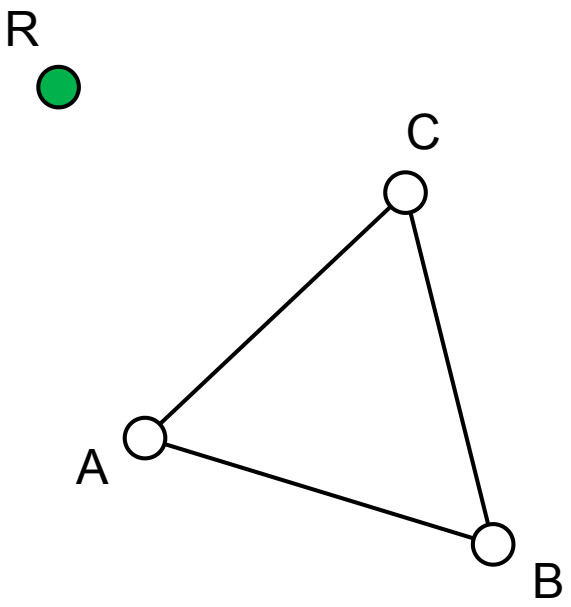
A: (-1, 1, 1)

A: (-1, 0, 2)

A: (0, 1.5, -0.5)

Quiz: Barycentric Coordinates

What are the barycentric coordinates (α, β, γ) of point $R = \alpha A + \beta B + \gamma C$?

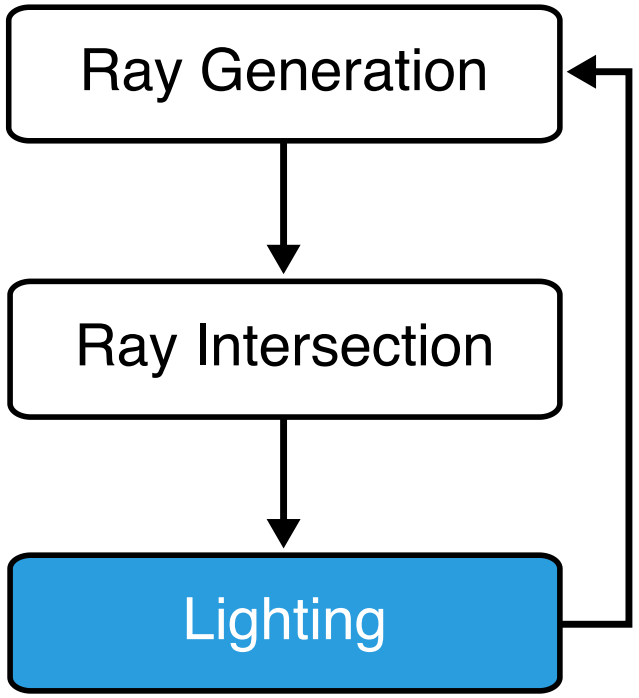


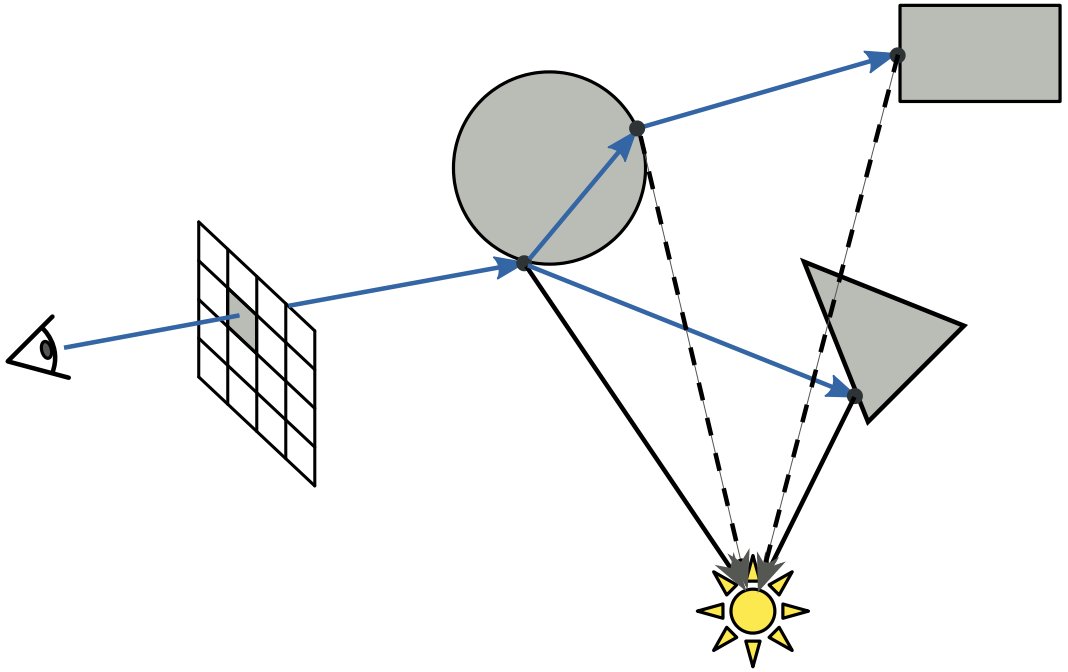
A: (-0.5, 2, -0.5)

A: (1, -1, 1)

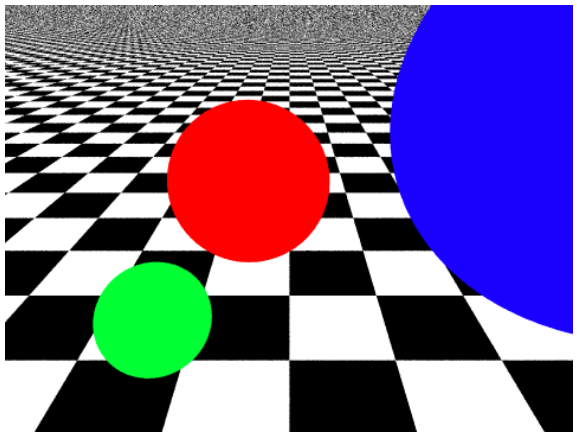
A: (0, -1, 2)

Ray Tracing Operators

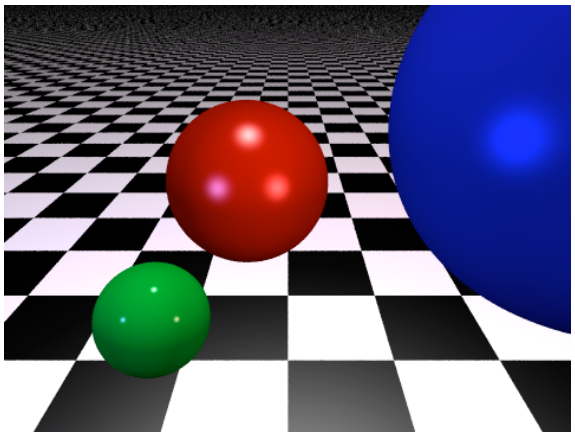




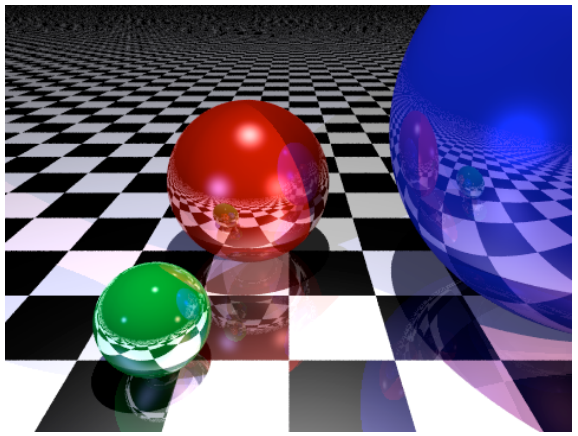
Lighting is important!



no illumination



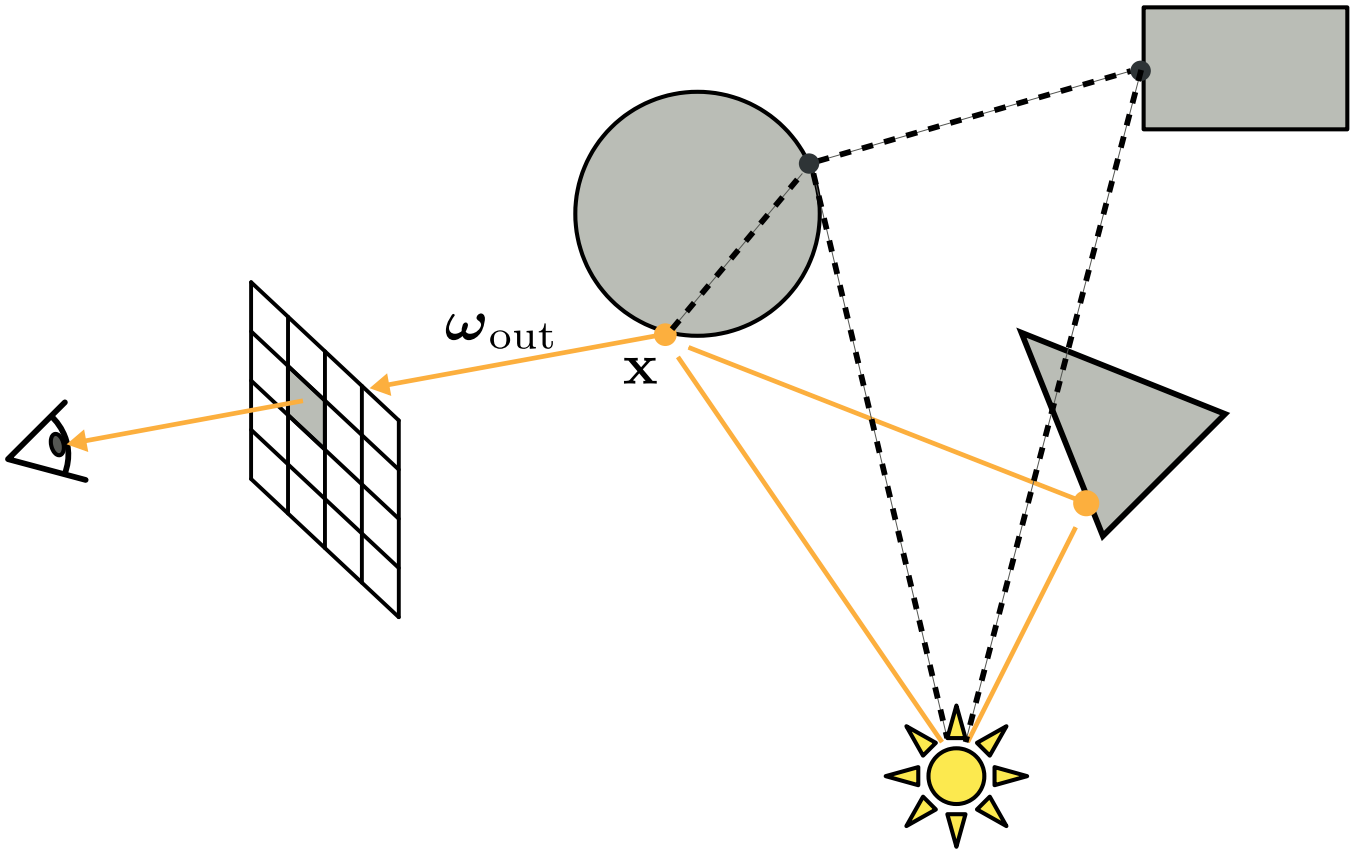
local illumination



global illumination

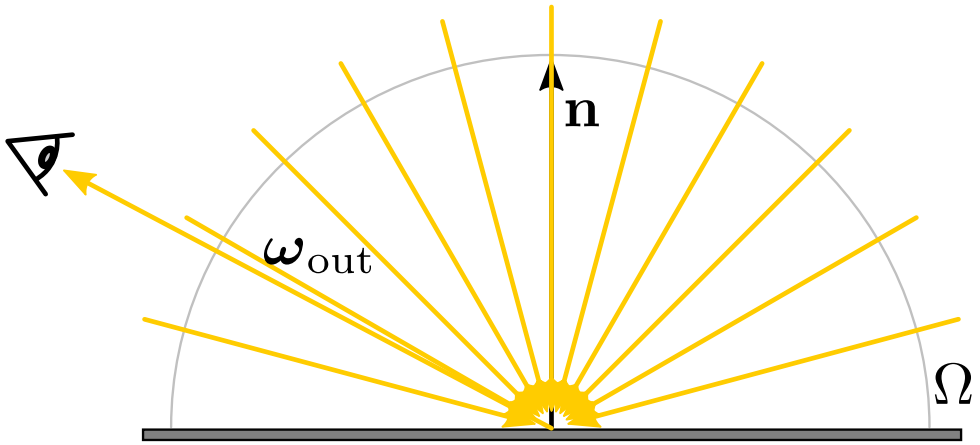
Surface Reflectance

- How much light is leaving point $\mathbf{x}$ in direction $\boldsymbol{\omega}_{\text{out}}$?



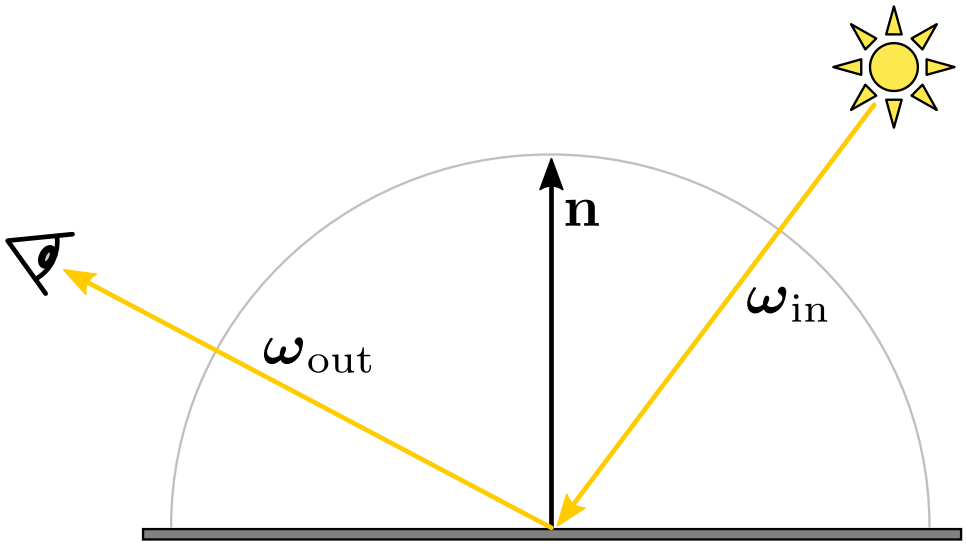
Surface Reflectance

- How much light is leaving point $\mathbf{x}$ in direction $\boldsymbol{\omega}_{\text{out}}$?



Surface Reflectance

- How much light coming in from direction ω_{in} is reflected out in direction ω_{out}?



- Determined by the object's *BRDF* $f(\omega_{\text{in}}, \omega_{\text{out}})$
  - Bidirectional Reflectance Distribution Function
  - General description of an object's material

Ray Tracing Approximations

Ray Tracing Approximations

- Most general reflection equation

$$L_{out}\left(\omega_{out}\right) = \int_{\Omega} f(\omega_{in}, \omega_{out})\, L_{in}\left(\omega_{in}\right) \cos\left(\theta_{in}\right) d\omega_{in}$$

Ray Tracing Approximations

- Most general reflection equation

$$L_{out}\left(\omega_{out}\right) = \int_{\Omega} f(\omega_{in}, \omega_{out})\, L_{in}\left(\omega_{in}\right) \cos\left(\theta_{in}\right) d\omega_{in}$$

- **Ray Tracing** approximates integral by three directions

Ray Tracing Approximations

- Most general reflection equation

$$L_{out}\left(\omega_{out}\right) = \int_{\Omega} f(\omega_{in}, \omega_{out})\, L_{in}\left(\omega_{in}\right) \cos\left(\theta_{in}\right) d\omega_{in}$$

- **Ray Tracing** approximates integral by three directions
  - directions toward light sources

Ray Tracing Approximations

- Most general reflection equation

$$L_{out}\left(\omega_{out}\right) = \int_{\Omega} f(\omega_{in}, \omega_{out})\, L_{in}\left(\omega_{in}\right) \cos\left(\theta_{in}\right) d\omega_{in}$$

- **Ray Tracing** approximates integral by three directions
  - directions toward light sources
  - directions of perfect reflection

Ray Tracing Approximations

- Most general reflection equation

$$L_{out}\left(\omega_{out}\right) = \int_{\Omega} f(\omega_{in}, \omega_{out})\, L_{in}\left(\omega_{in}\right) \cos\left(\theta_{in}\right) d\omega_{in}$$

- **Ray Tracing** approximates integral by three directions
  - directions toward light sources
  - directions of perfect reflection
  - directions of perfect refraction

Ray Tracing Approximations

- Most general reflection equation

$$L_{out}\left(\omega_{out}\right) = \int_{\Omega} f(\omega_{in}, \omega_{out})\, L_{in}\left(\omega_{in}\right) \cos\left(\theta_{in}\right) d\omega_{in}$$

- **Ray Tracing** approximates integral by three directions
  - directions toward light sources
  - directions of perfect reflection
  - directions of perfect refraction
- **Phong Lighting** approximates BRDF by three components

Ray Tracing Approximations

- Most general reflection equation

$$L_{out}\left(\omega_{out}\right) = \int_{\Omega} f(\omega_{in}, \omega_{out})\, L_{in}\left(\omega_{in}\right) \cos\left(\theta_{in}\right) d\omega_{in}$$

- **Ray Tracing** approximates integral by three directions
  - directions toward light sources
  - directions of perfect reflection
  - directions of perfect refraction
- **Phong Lighting** approximates BRDF by three components
  - ambient light

Ray Tracing Approximations

- Most general reflection equation

$$L_{out}\left(\omega_{out}\right) = \int_{\Omega} f(\omega_{in}, \omega_{out})\, L_{in}\left(\omega_{in}\right) \cos\left(\theta_{in}\right) d\omega_{in}$$

- **Ray Tracing** approximates integral by three directions
  - directions toward light sources
  - directions of perfect reflection
  - directions of perfect refraction
- **Phong Lighting** approximates BRDF by three components
  - ambient light
  - diffuse reflection

Ray Tracing Approximations

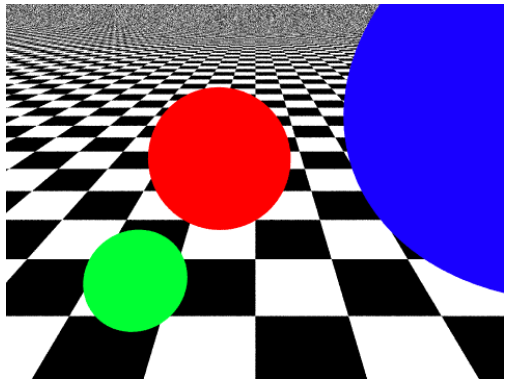
- Most general reflection equation

$$L_{out}\left(\omega_{out}\right) = \int_{\Omega} f(\omega_{in}, \omega_{out})\, L_{in}\left(\omega_{in}\right) \cos\left(\theta_{in}\right) d\omega_{in}$$

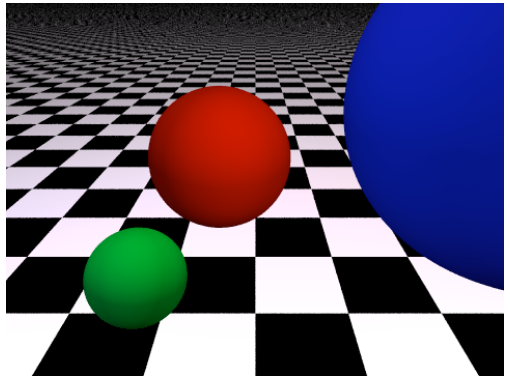
- **Ray Tracing** approximates integral by three directions
  - directions toward light sources
  - directions of perfect reflection
  - directions of perfect refraction
- **Phong Lighting** approximates BRDF by three components
  - ambient light
  - diffuse reflection
  - specular reflection

Phong Lighting Model

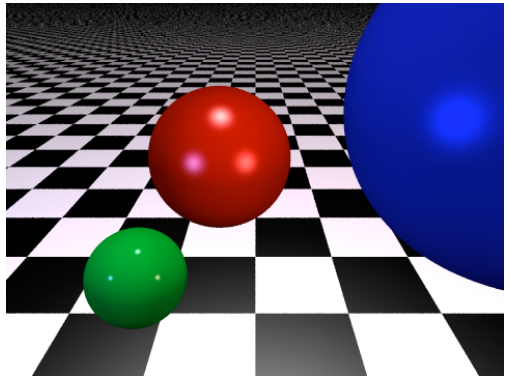
- **Ambient lighting**
  - approximate global light transport / exchange
  - uniform in, uniform out



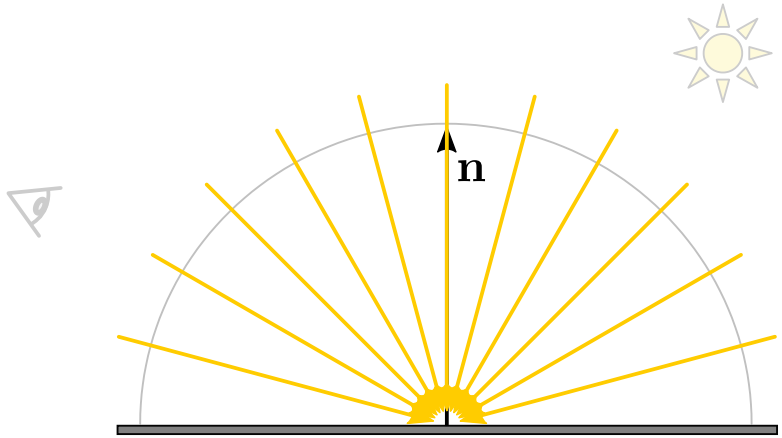
- **Diffuse lighting**
  - dull / matt surfaces
  - directed in, uniform out



- **Specular lighting**
  - shiny surfaces
  - directed in, directed out

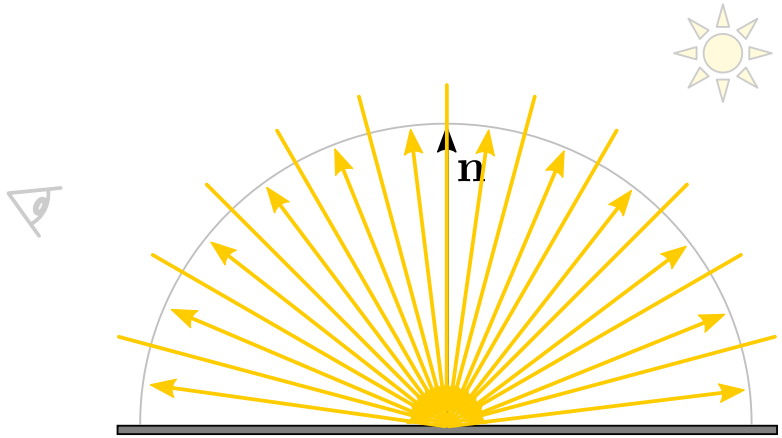


Ambient Light



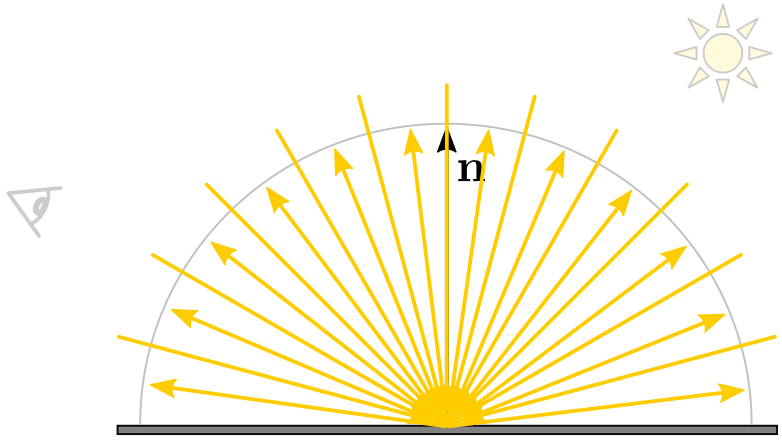
uniformly incoming light

Ambient Light



uniformly outgoing light

Ambient Light

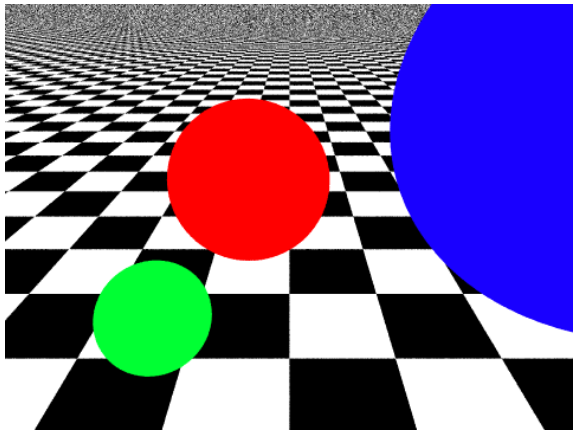


uniformly outgoing light

$$I = I_a m_a$$

- I_a: ambient light intensity in the scene
- m_a: material's ambient reflection coefficient

Lighting Computations



ambient

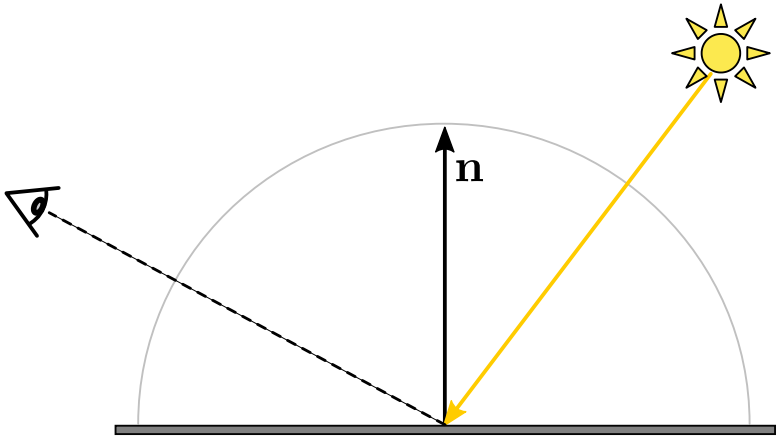
+diffuse

+specular

+shadows

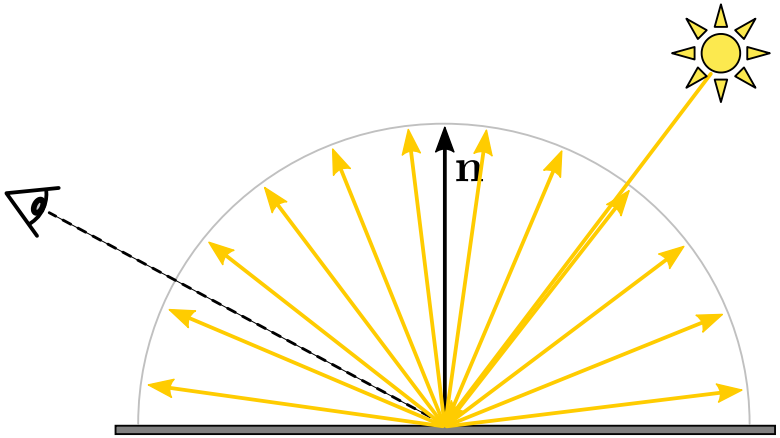
+reflections

Diffuse Reflection



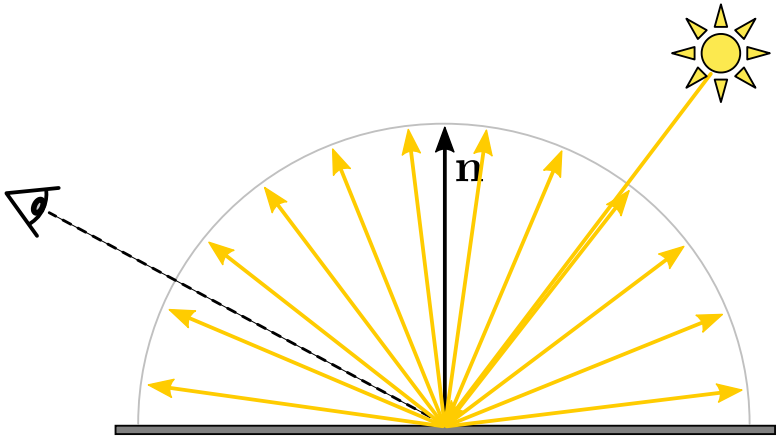
directed incoming light

Diffuse Reflection



uniformly outgoing light

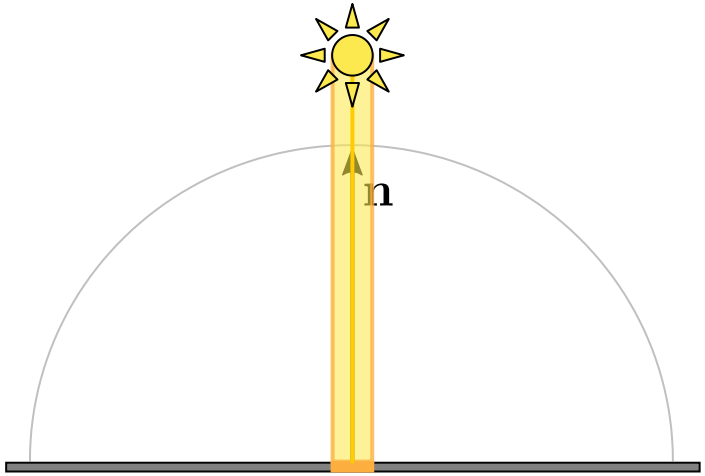
Diffuse Reflection

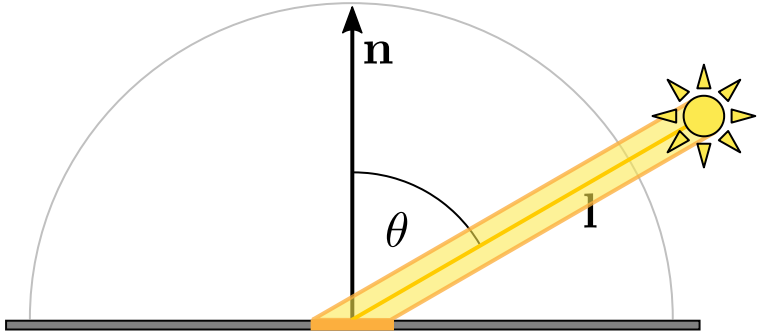


uniformly outgoing light

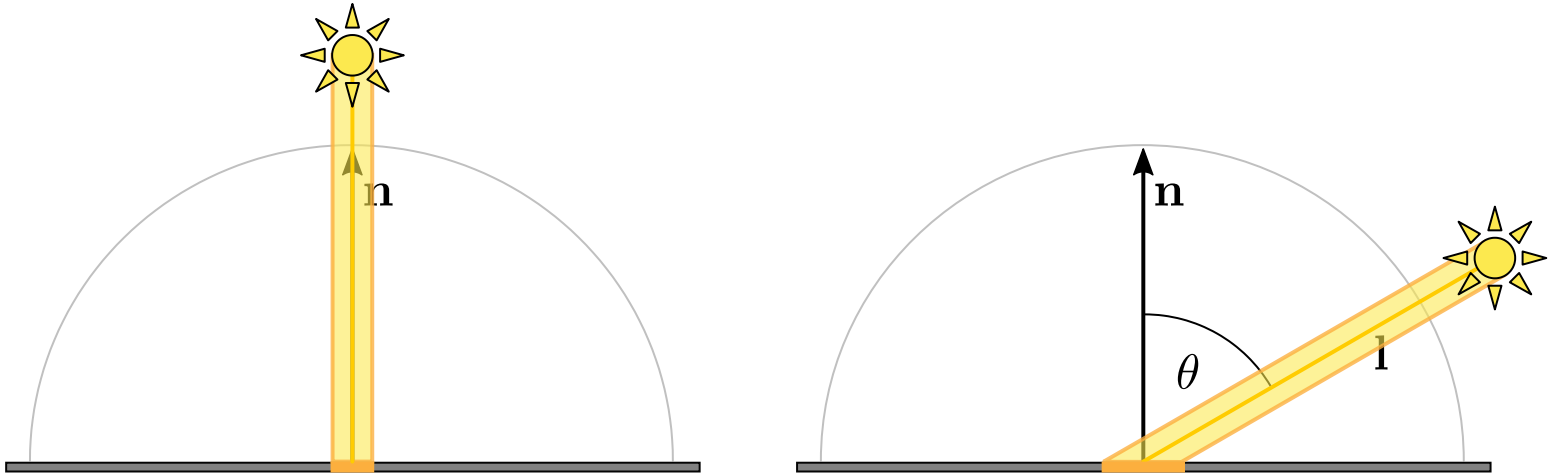
Brightness depends on how much light (density) comes in!

Diffuse Reflection



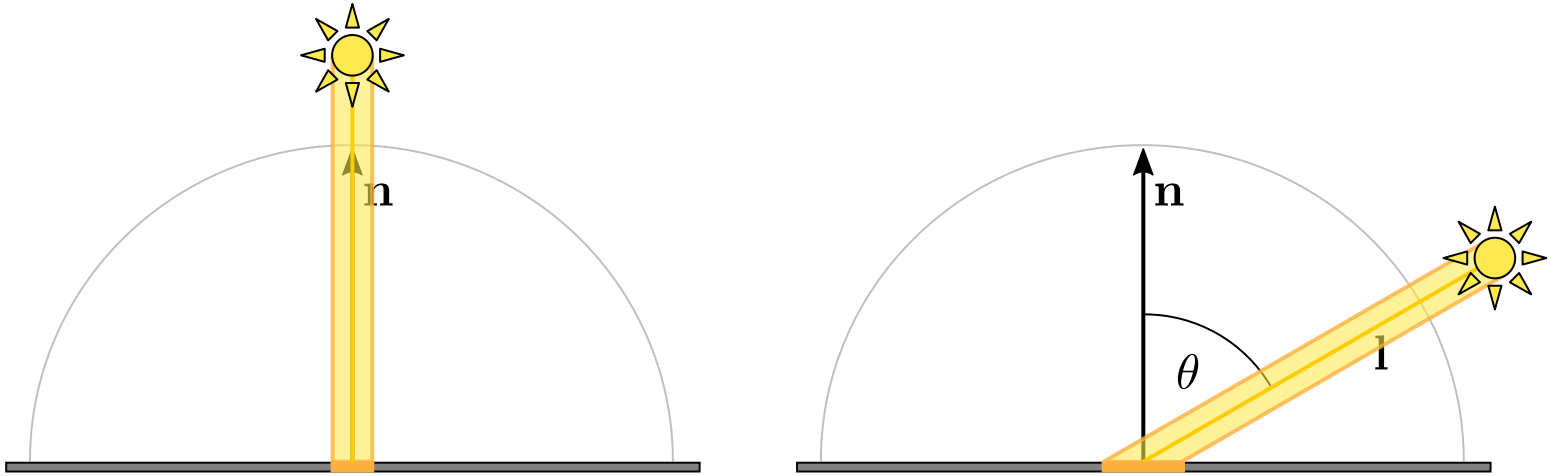


Diffuse Reflection



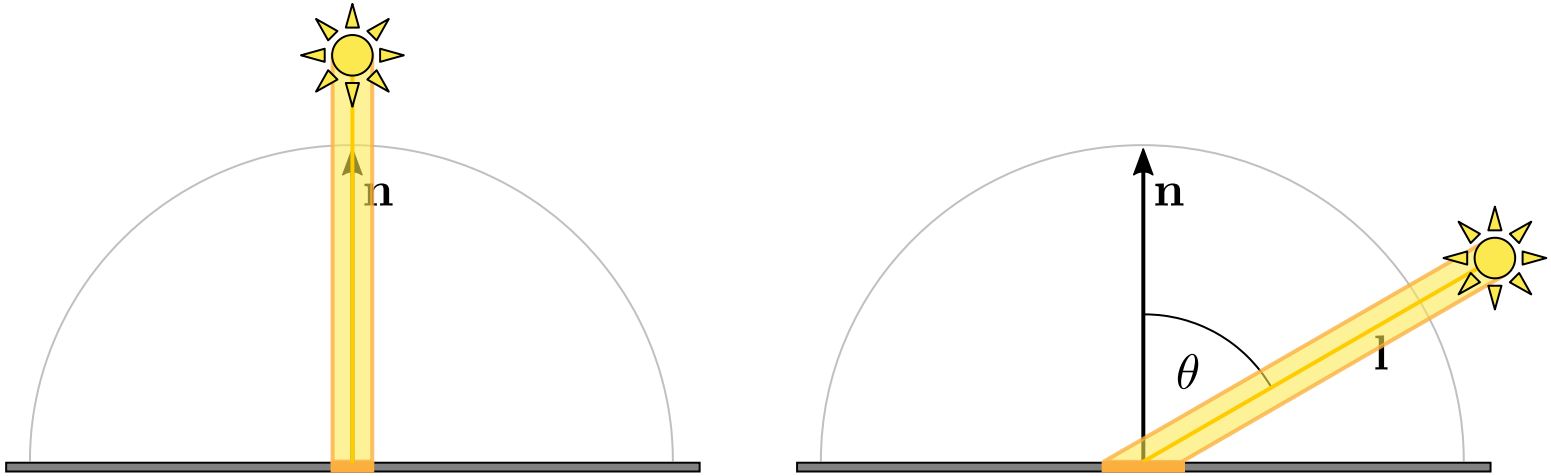
- Intensity inversely proportional to illuminated area

Diffuse Reflection



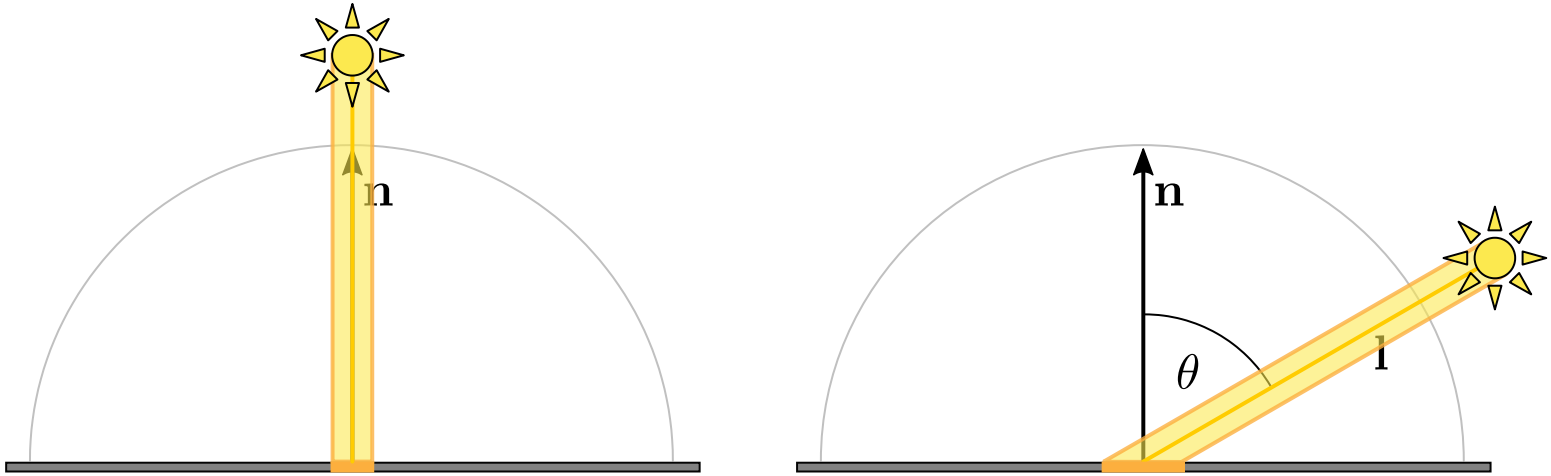
- Intensity inversely proportional to illuminated area
- Illuminated area inversely proportional to $\cos(\theta)$

Diffuse Reflection



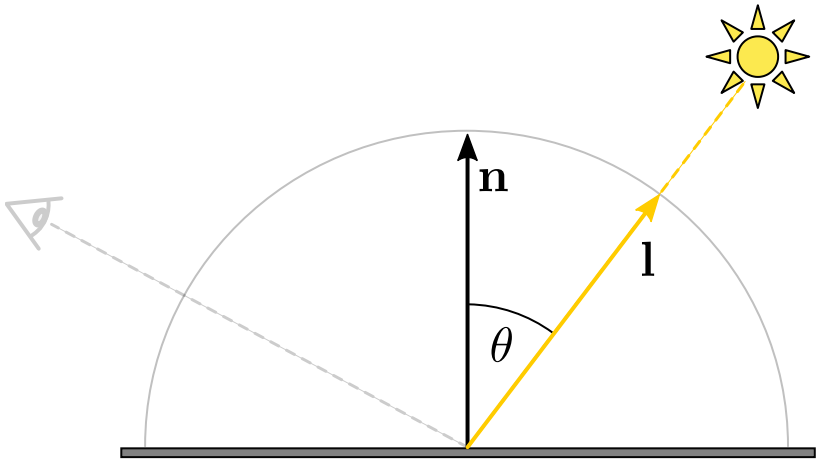
- Intensity inversely proportional to illuminated area
- Illuminated area inversely proportional to $\cos(\theta)$
- Therefore intensity proportional to $\cos(\theta)$

Diffuse Reflection

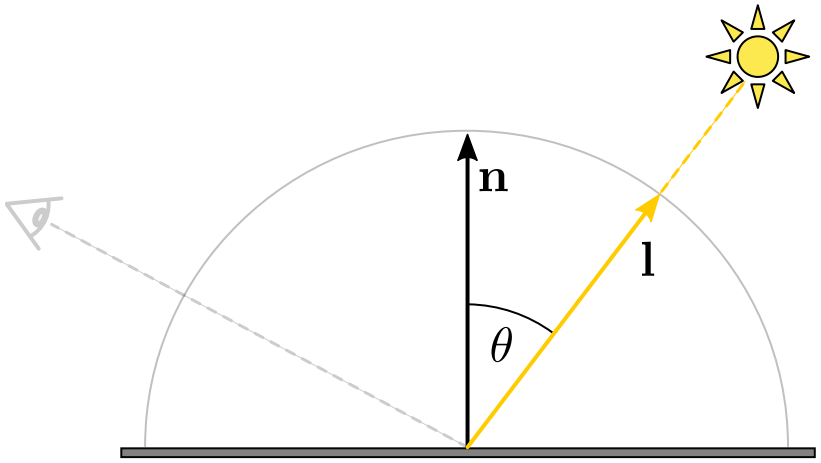


- Intensity inversely proportional to illuminated area
- Illuminated area inversely proportional to $\cos(\theta)$
- Therefore intensity proportional to $\cos(\theta)$
- So-called *Lambertian reflection*

Diffuse Reflection



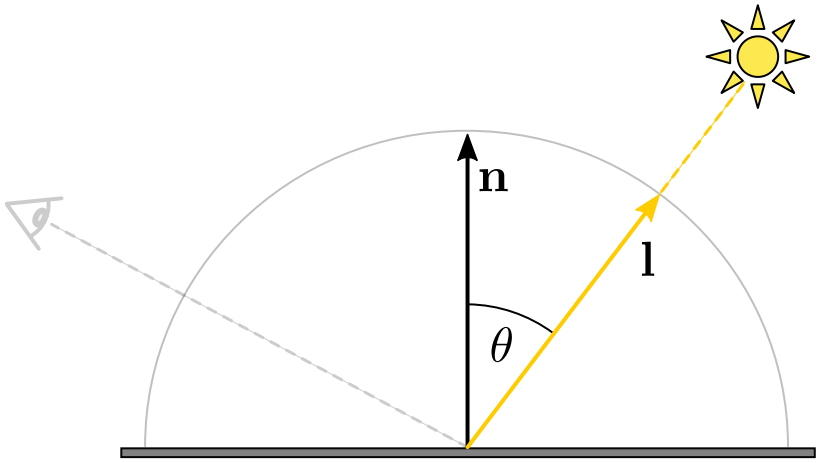
Diffuse Reflection



$$I = I_l m_d \cos\theta$$

- I_l: intensity of light source l
- m_d: material's diffuse reflection coefficient

Diffuse Reflection

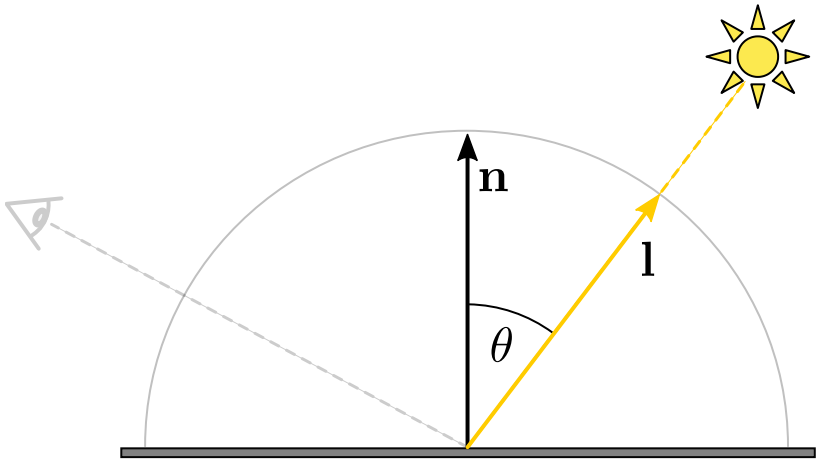


$$I = I_l m_d \cos\theta$$

How can we compute $\cos\theta$ efficiently?

- I_l: intensity of light source l
- m_d: material's diffuse reflection coefficient

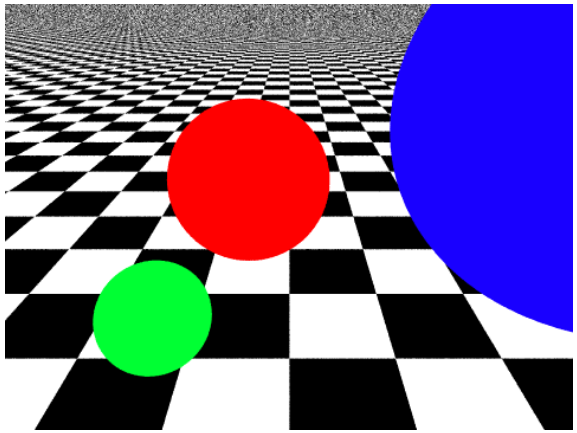
Diffuse Reflection



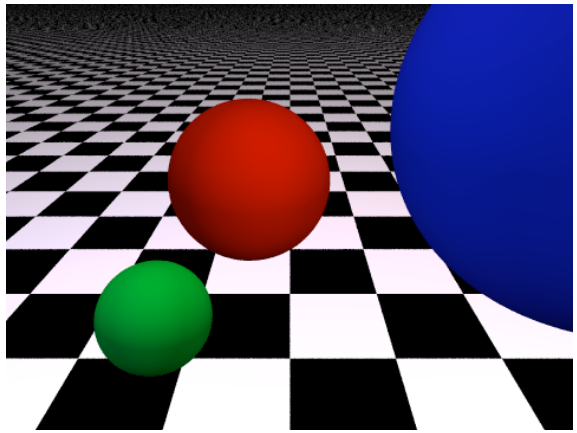
$$I = I_l \, m_d \, \cos\theta = I_l \, m_d \, (\mathbf{n} \cdot \mathbf{l})$$

- I_l: intensity of light source l
- m_d: material's diffuse reflection coefficient
- directions $\mathbf{n}$ and $\mathbf{l}$ assumed to be normalized
- no illumination if $\mathbf{n} \cdot \mathbf{l} < 0$ (why?)

Lighting Computations



ambient



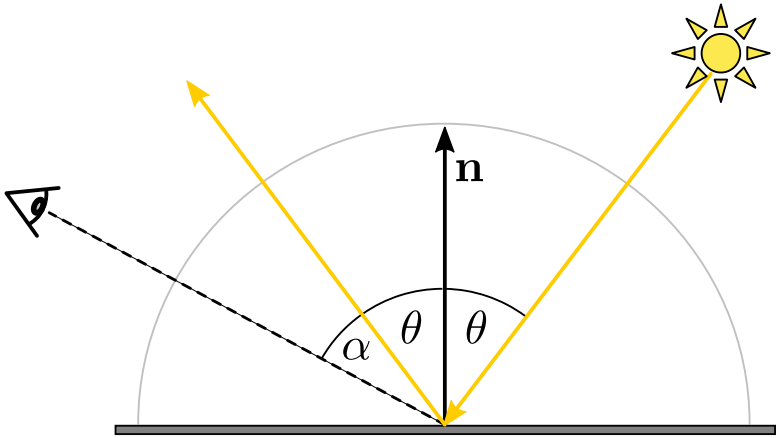
+diffuse

+specular

+shadows

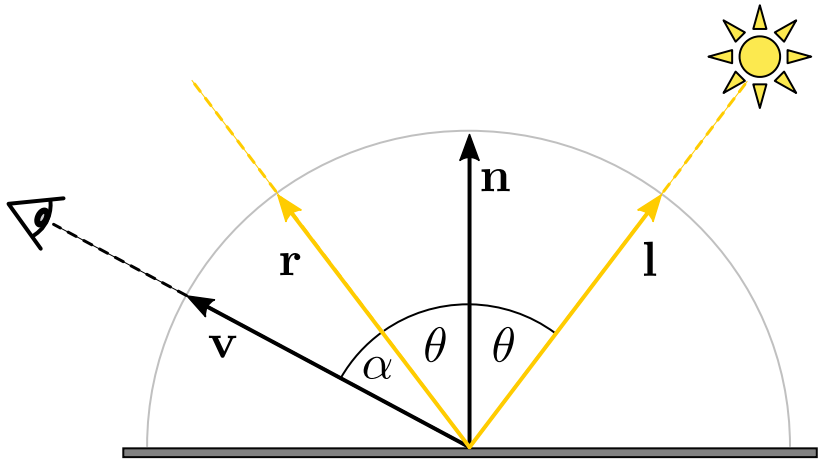
+reflections

Specular Reflection

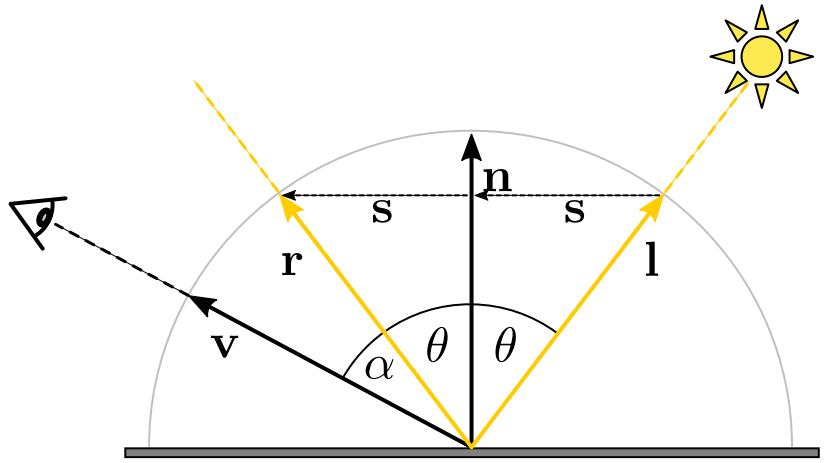


directed incoming, directed outgoing

Specular Reflection

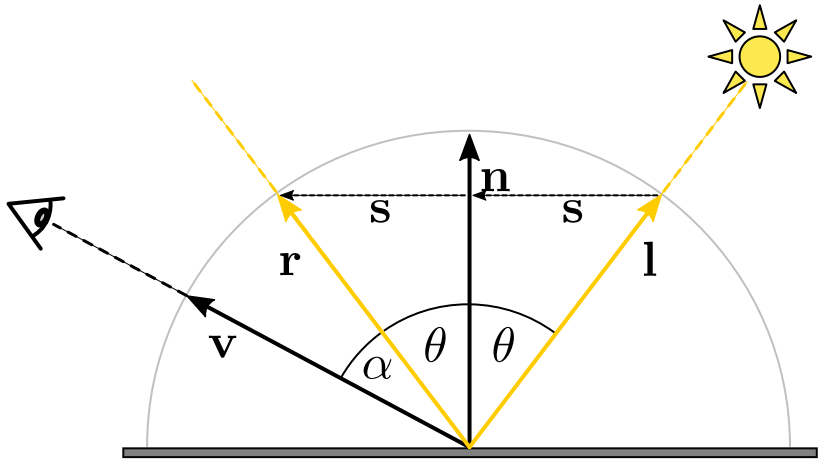


Specular Reflection



How to compute reflected ray $\mathbf{r}$?

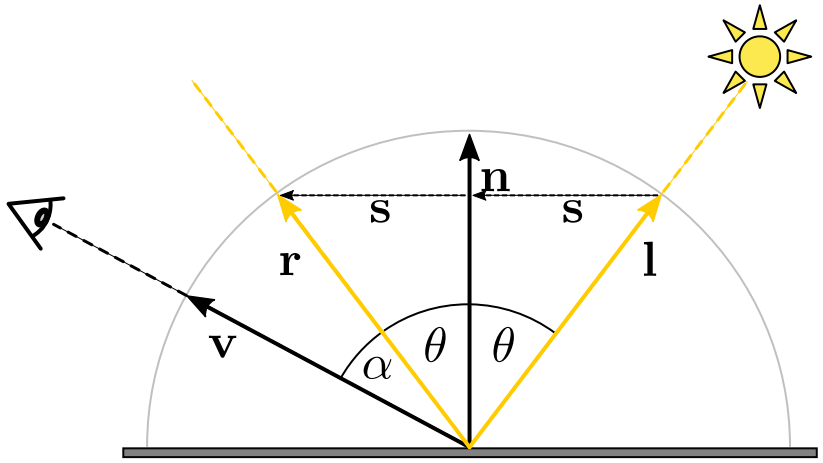
Specular Reflection



How to compute reflected ray $\mathbf{r}$?

- $\mathbf{r} = \mathbf{l} + 2\mathbf{s}$

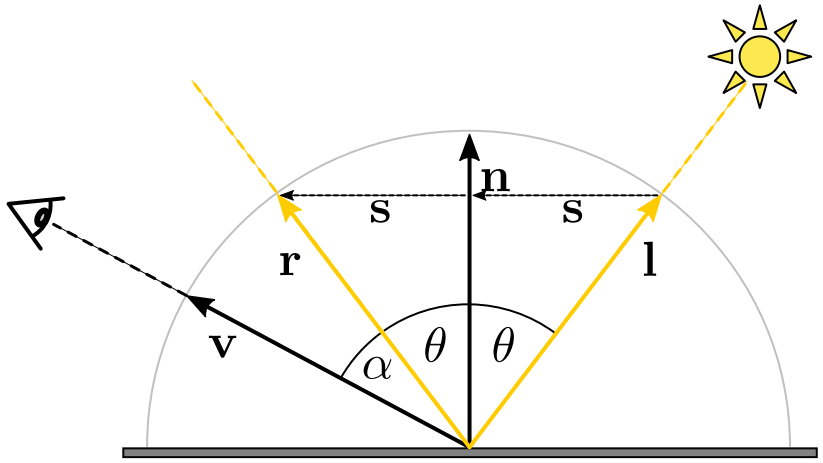
Specular Reflection



How to compute reflected ray $\mathbf{r}$?

- $\mathbf{r} = \mathbf{l} + 2\mathbf{s}$
- $\mathbf{s} = -\mathbf{l} + \mathbf{n}(\mathbf{n} \cdot \mathbf{l})$

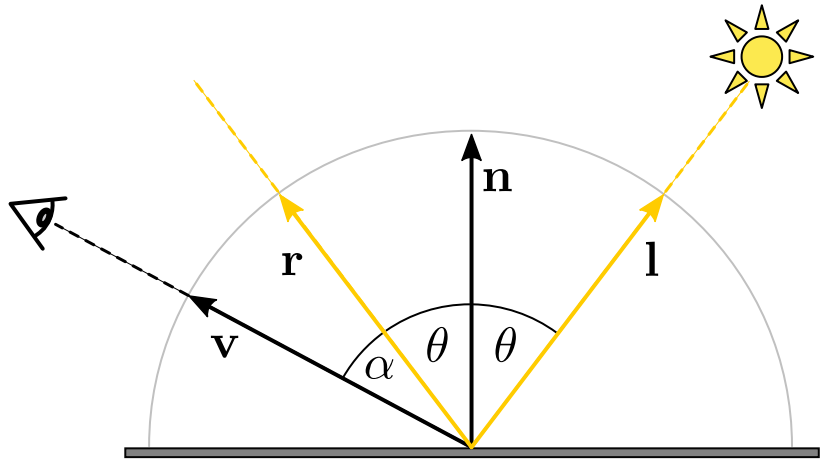
Specular Reflection



How to compute reflected ray **r**?

- $\mathbf{r} = \mathbf{l} + 2\mathbf{s}$
- $\mathbf{s} = -\mathbf{l} + \mathbf{n}(\mathbf{n} \cdot \mathbf{l})$
- $\mathbf{r} = 2\,\mathbf{n}(\mathbf{n} \cdot \mathbf{l}) - \mathbf{l}$

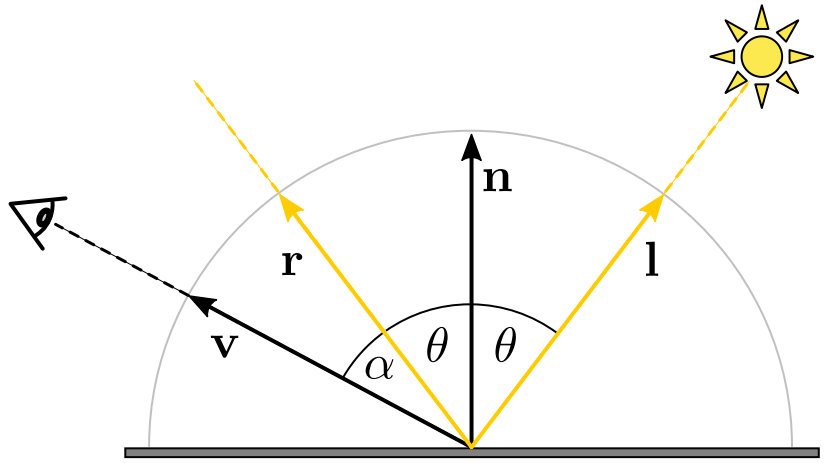
Specular Reflection



$$I = I_l \, m_s \, \cos(\alpha) = I_l \, m_s \, (\mathbf{r} \cdot \mathbf{v})$$

- I_l: intensity of light source l
- m_s: material's specular reflection coefficient
- all directions assumed to be normalized
- no illumination if $\mathbf{n} \cdot \mathbf{l} < 0$ or $\mathbf{r} \cdot \mathbf{v} < 0$

Specular Reflection

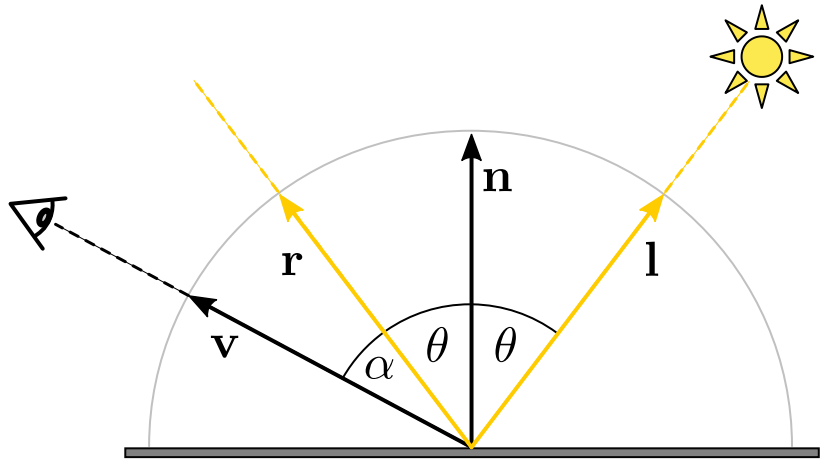


$$I = I_l m_s \cos(\alpha) = I_l m_s (\mathbf{r} \cdot \mathbf{v})$$

How can we control the surface's shininess?

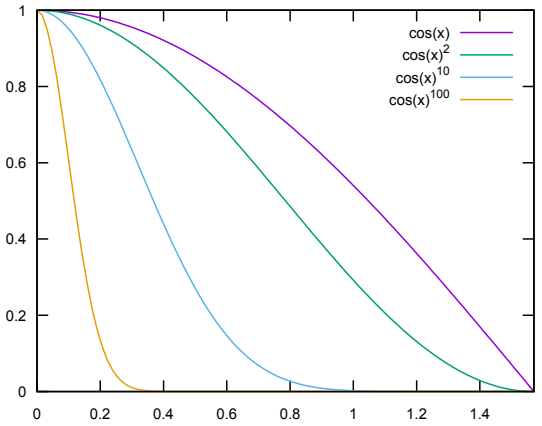
- I_l: intensity of light source l
- m_s: material's specular reflection coefficient
- all directions assumed to be normalized
- no illumination if $\mathbf{n} \cdot \mathbf{l} < 0$ or $\mathbf{r} \cdot \mathbf{v} < 0$

Specular Reflection



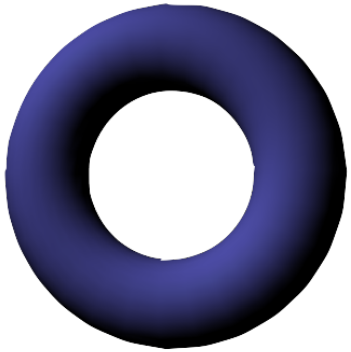
$$I = I_l \, m_s \, \cos^s(\alpha) = I_l \, m_s \, (\mathbf{r} \cdot \mathbf{v})^s$$

- I_l: intensity of light source l
- m_s: material's specular reflection coefficient
- all directions assumed to be normalized
- no illumination if $\mathbf{n} \cdot \mathbf{l} < 0$ or $\mathbf{r} \cdot \mathbf{v} < 0$
- s: cosine exponent controls shininess



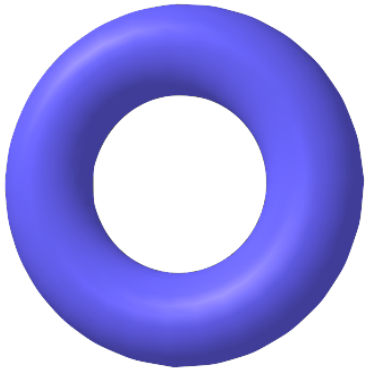
Phong Lighting Model

ambient: $I_a m_a$



diffuse: $I_l m_d \left(\mathbf{n} \cdot \mathbf{l}\right)$

specular: $I_l m_s (\mathbf{r} \cdot \mathbf{v})^s$



$$I = I_a m_a + I_l \left(m_d \left(\mathbf{n} \cdot \mathbf{l}\right) + m_s (\mathbf{r} \cdot \mathbf{v})^s \right)$$

Colors & Lighting

Colors & Lighting

- Lighting depends on wavelength λ

$$I_\lambda = I_{a,\lambda} m_{a,\lambda} + I_{l,\lambda}\left(m_{d,\lambda}(\mathbf{n}\cdot\mathbf{l}) + m_{s,\lambda}(\mathbf{r}\cdot\mathbf{v})^s\right)$$

Colors & Lighting

- Lighting depends on wavelength λ

$$I_\lambda = I_{a,\lambda} m_{a,\lambda} + I_{l,\lambda}\left(m_{d,\lambda}(\mathbf{n}\cdot\mathbf{l}) + m_{s,\lambda}(\mathbf{r}\cdot\mathbf{v})^s\right)$$

- We approximate it by RBG components

$$I_R = I_{a,R} m_{a,R} + I_{l,R}\left(m_{d,R}(\mathbf{n}\cdot\mathbf{l}) + m_{s,R}(\mathbf{r}\cdot\mathbf{v})^s\right)$$

$$I_G = I_{a,G} m_{a,G} + I_{l,G}\left(m_{d,G}(\mathbf{n}\cdot\mathbf{l}) + m_{s,G}(\mathbf{r}\cdot\mathbf{v})^s\right)$$

$$I_B = I_{a,B} m_{a,B} + I_{l,B}\left(m_{d,B}(\mathbf{n}\cdot\mathbf{l}) + m_{s,B}(\mathbf{r}\cdot\mathbf{v})^s\right)$$

Colors & Lighting

- Lighting depends on wavelength λ

$$I_\lambda = I_{a,\lambda} m_{a,\lambda} + I_{l,\lambda}\left(m_{d,\lambda}(\mathbf{n}\cdot\mathbf{l}) + m_{s,\lambda}(\mathbf{r}\cdot\mathbf{v})^s\right)$$

- We approximate it by RBG components

$$I_R = I_{a,R} m_{a,R} + I_{l,R}\left(m_{d,R}(\mathbf{n}\cdot\mathbf{l}) + m_{s,R}(\mathbf{r}\cdot\mathbf{v})^s\right)$$

$$I_G = I_{a,G} m_{a,G} + I_{l,G}\left(m_{d,G}(\mathbf{n}\cdot\mathbf{l}) + m_{s,G}(\mathbf{r}\cdot\mathbf{v})^s\right)$$

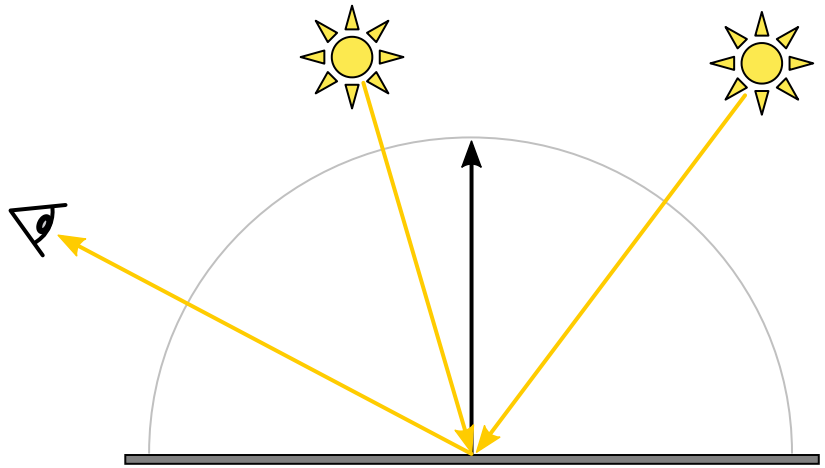
$$I_B = I_{a,B} m_{a,B} + I_{l,B}\left(m_{d,B}(\mathbf{n}\cdot\mathbf{l}) + m_{s,B}(\mathbf{r}\cdot\mathbf{v})^s\right)$$

- For RGB light colors/intensities $\mathbf{I}$ and RGB material

Try it yourself!

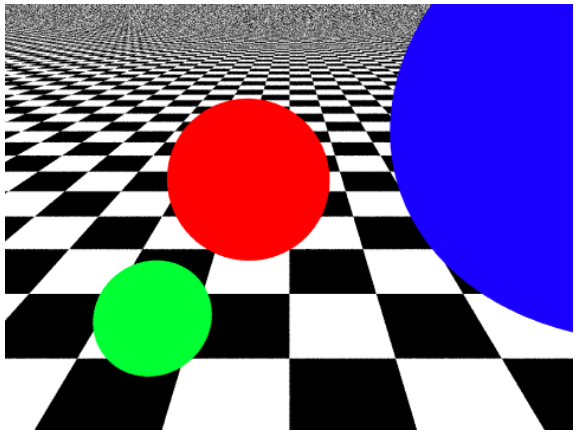
Multiple Light Sources

We assumed linear superposition of light contributions and therefore can simply sum over all light sources

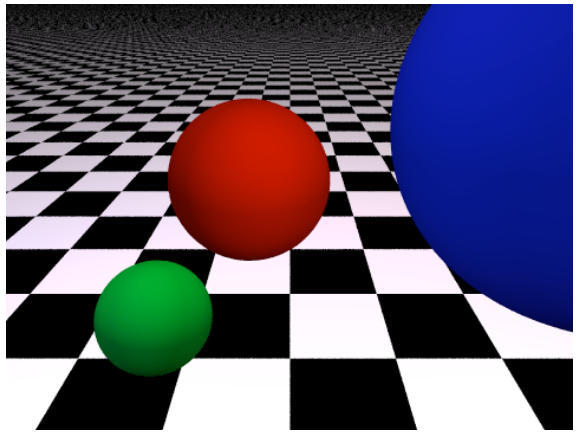


$$I = I_a m_a + \sum_l I_l \left(m_d \left(\mathbf{n} \cdot \mathbf{l}_l \right) + m_s (\mathbf{r}_l \cdot \mathbf{v})^s \right)$$

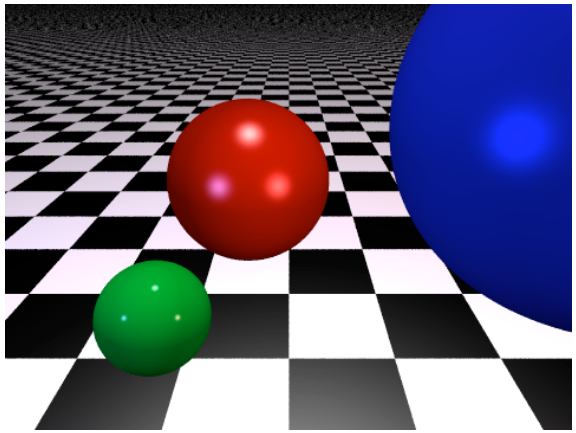
Lighting Computations



ambient



+diffuse



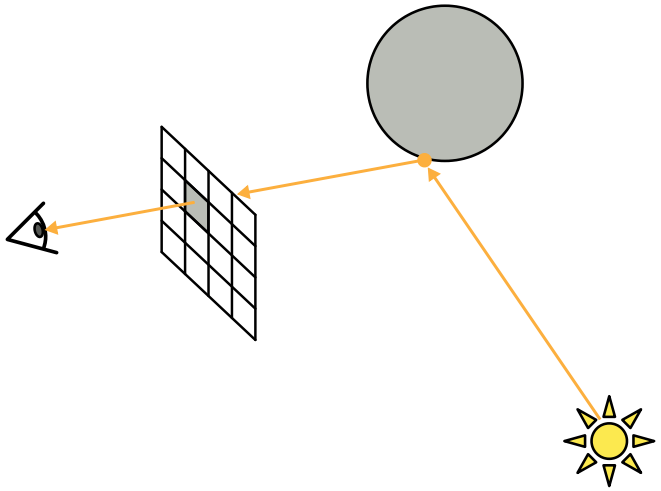
+specular

+shadows

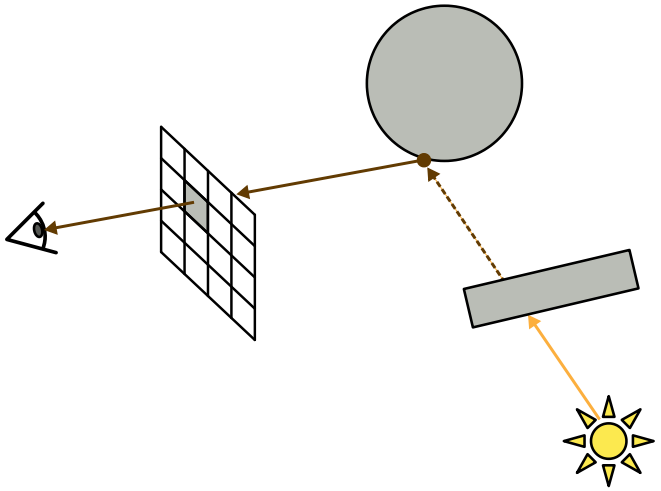
+reflections

Shadows

- Send *shadow ray* from intersection point to light source.
- Discard diffuse and specular contribution if light source is blocked by another object.



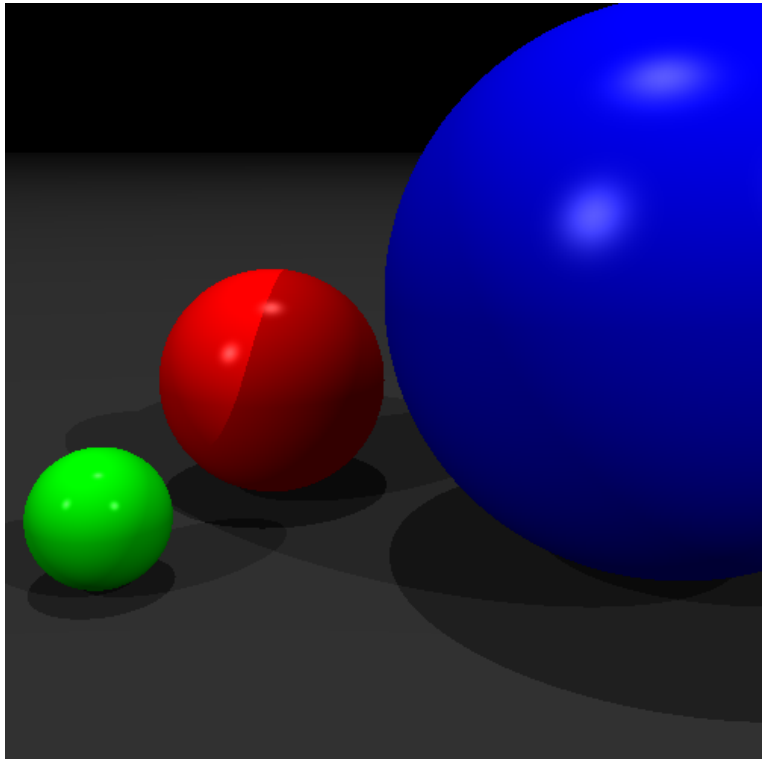
Point in light: ambient + diffuse + specular



Point In shadow: ambient lighting only

Shadows

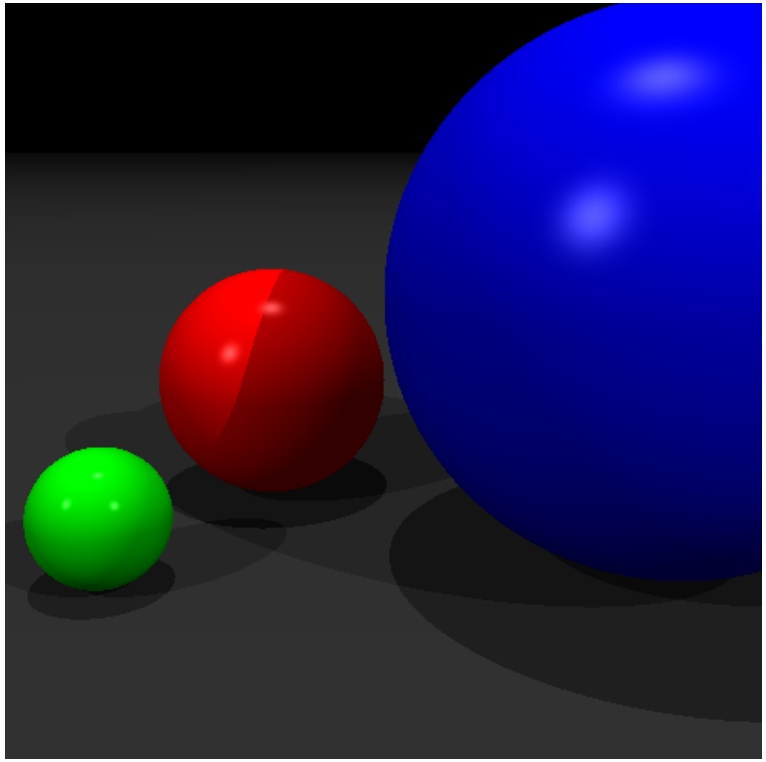
- Send *shadow ray* from intersection point to light source.
- Discard diffuse and specular contribution if light source is blocked by another object.



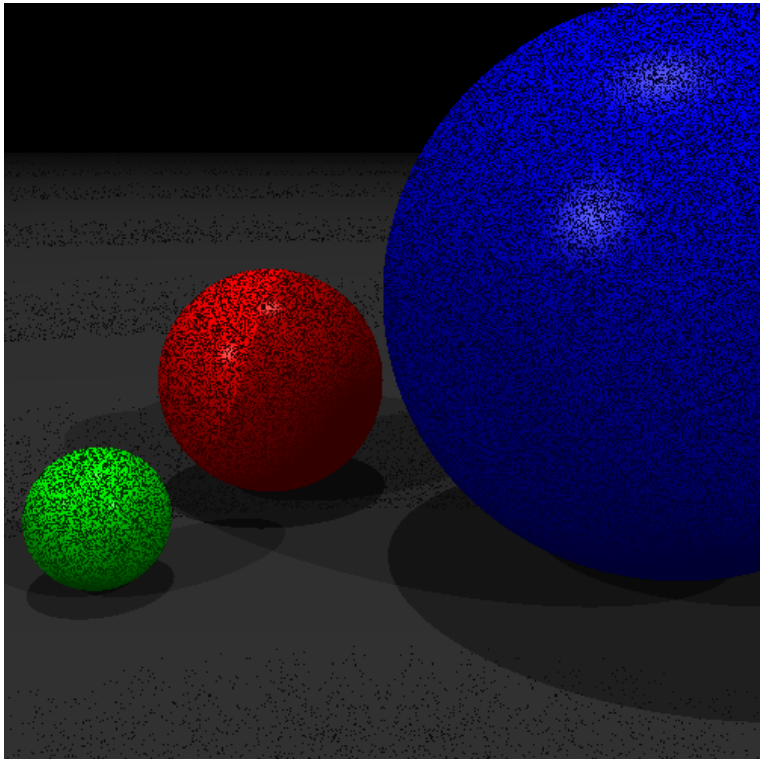
we want this

Shadows

- Send *shadow ray* from intersection point to light source.
- Discard diffuse and specular contribution if light source is blocked by another object.



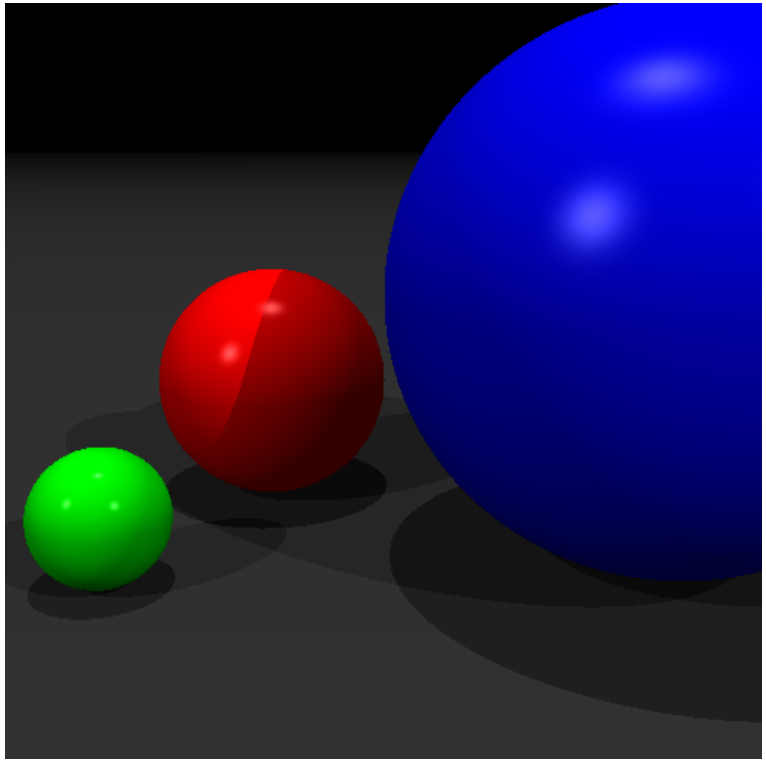
we want this



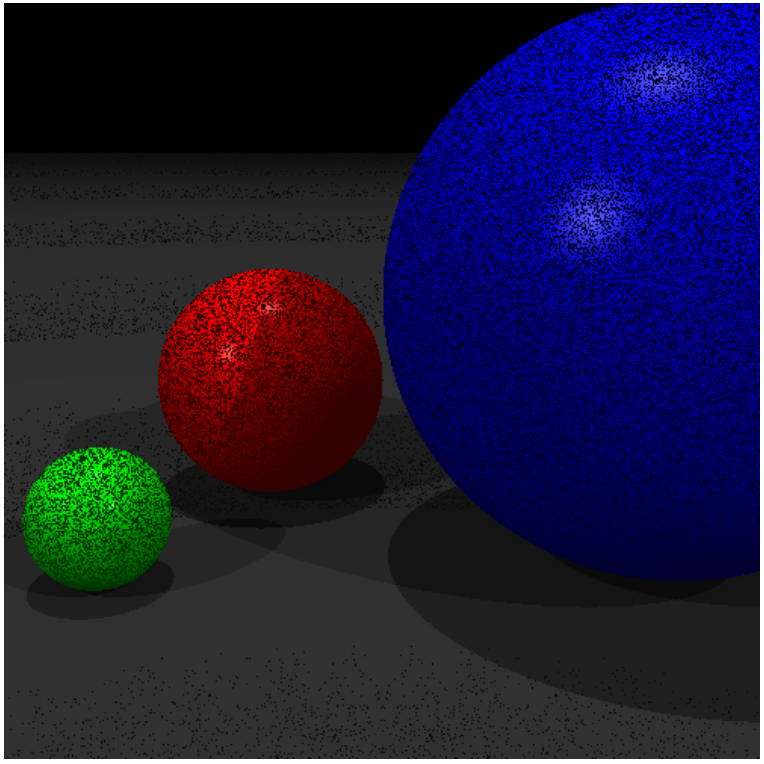
but we get this

Shadows

- Send *shadow ray* from intersection point to light source.
- Discard diffuse and specular contribution if light source is blocked by another object.



we want this

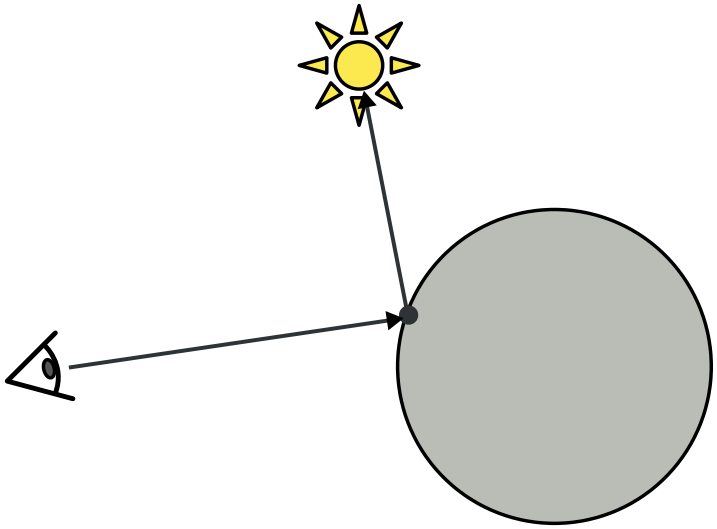


but we get this

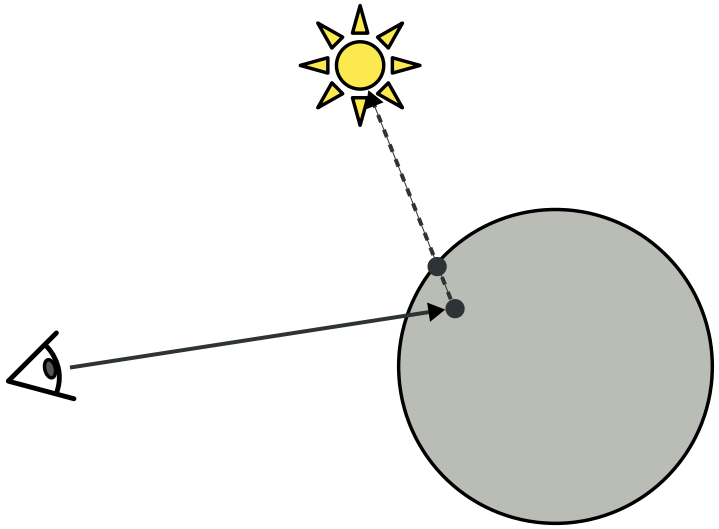
Why??

Shadows

- Floating point errors might lead to erroneous self-shadowing (*shadow acne*).
  - Solution 1: Discard secondary intersection points that are too close.
    - in our implementation: slightly displace ray origin along new ray direction.
  - Solution 2: Offset primary intersection point along surface normal.

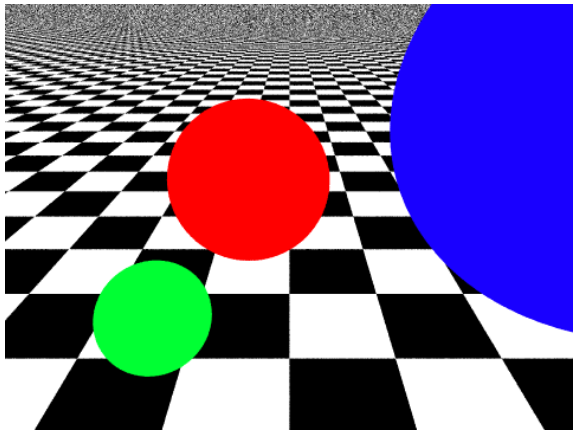


exact computation

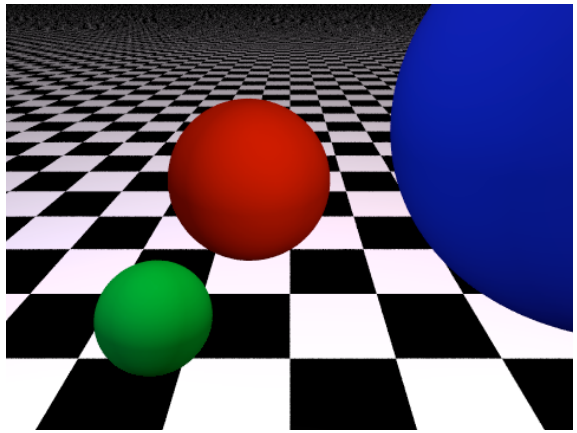


precision problems

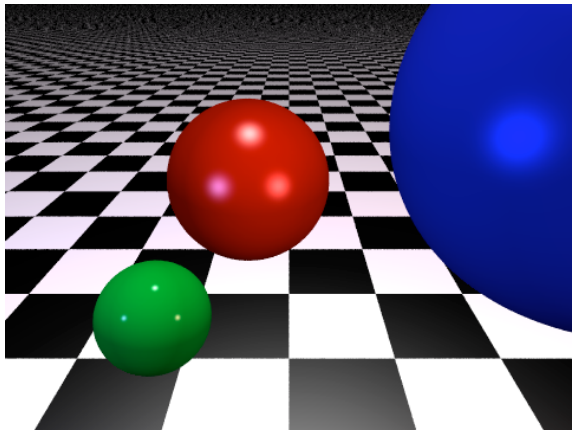
Lighting Computations



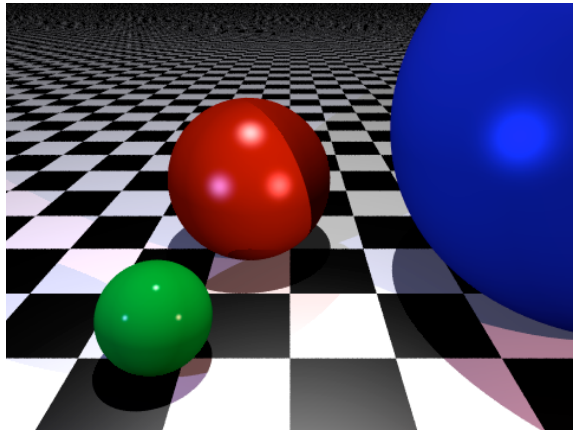
ambient



+diffuse



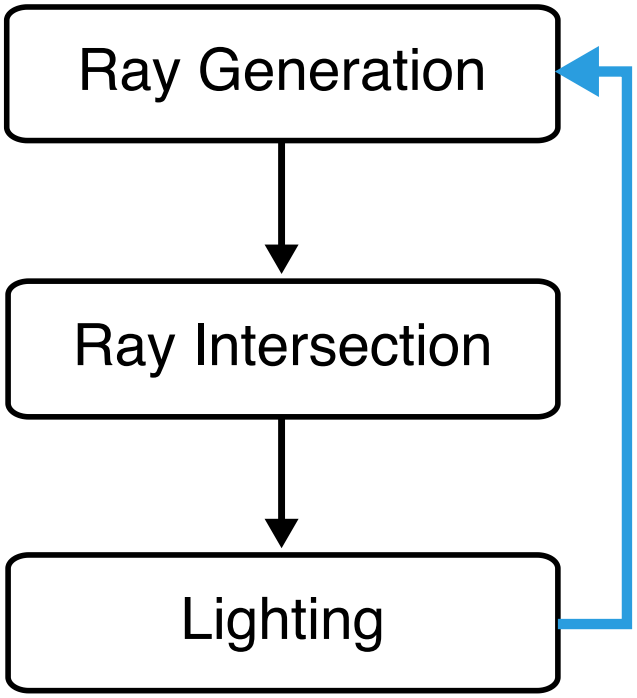
+specular

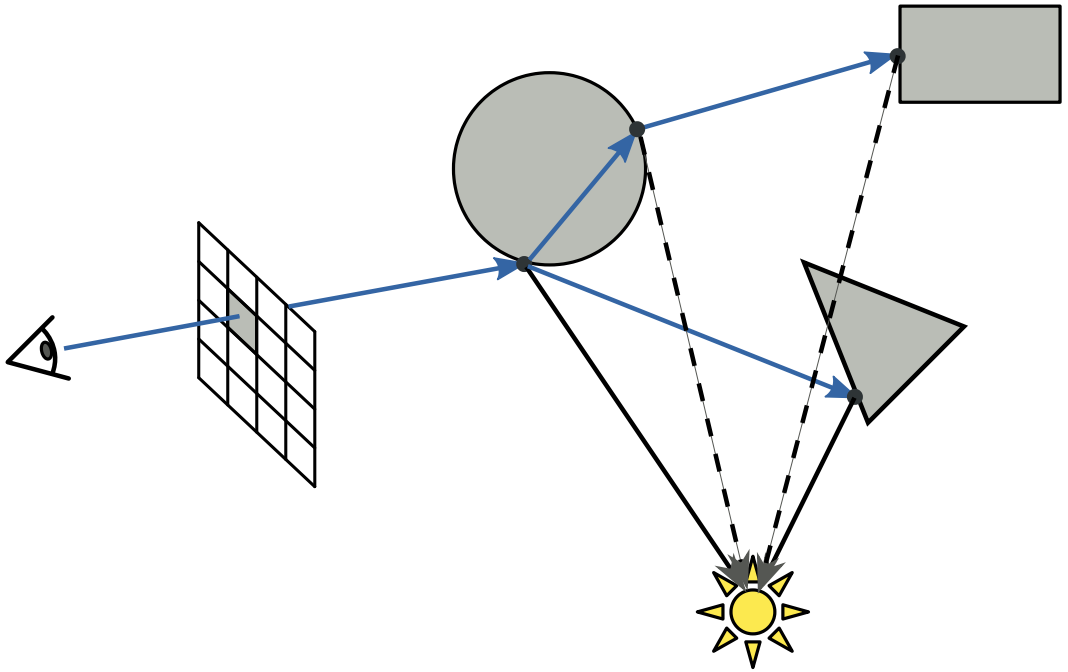


+shadows

+reflections

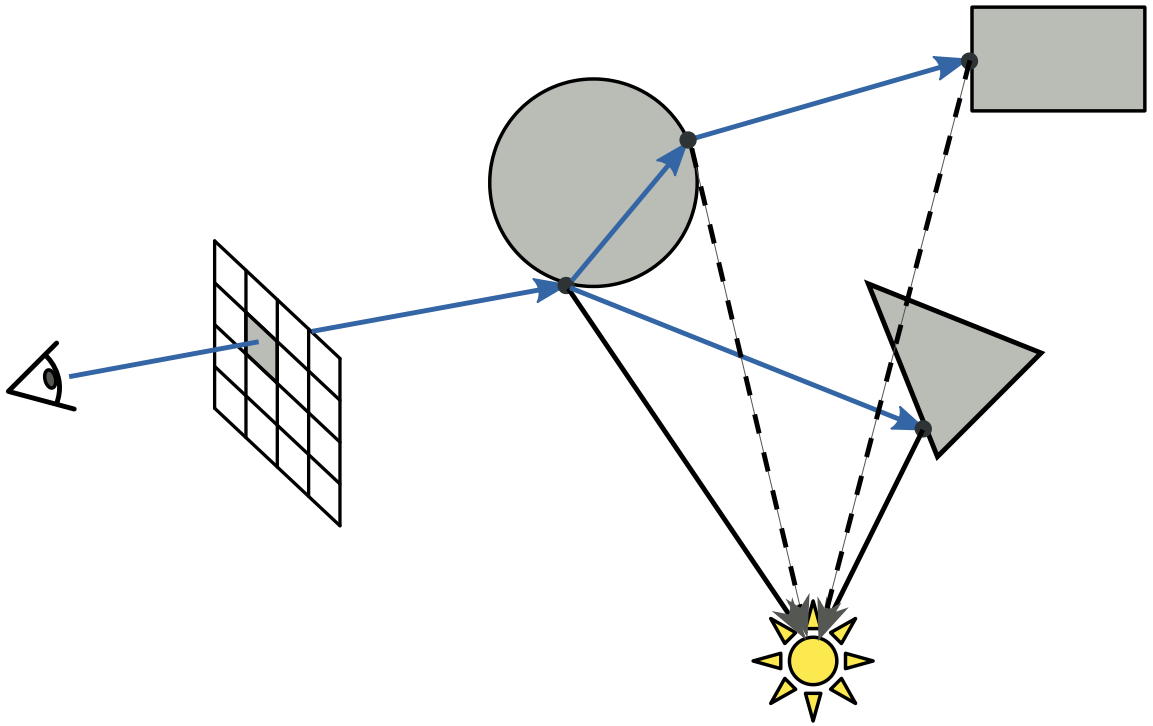
Ray Tracing Operators





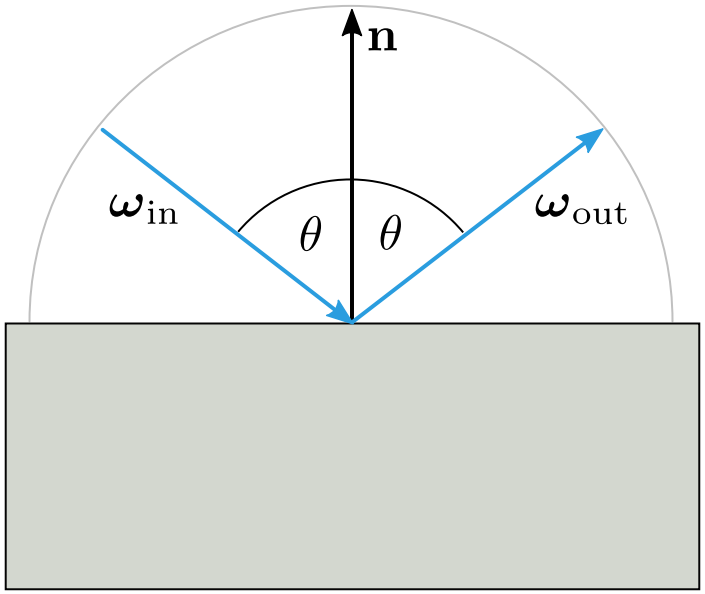
Recursive Ray Tracing

- At each intersection point, reflect and/or refract incoming viewing ray at surface normal, and trace child rays *recursively*.

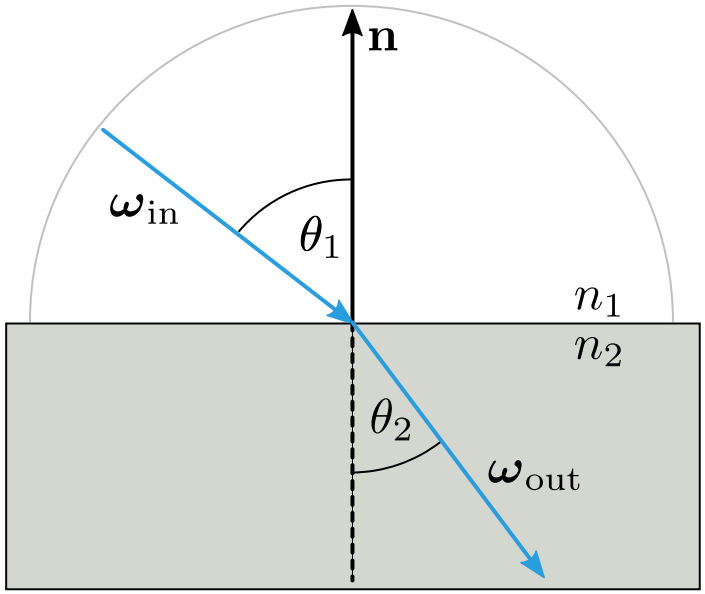


Recursive Ray Tracing

- At each intersection point, reflect and/or refract incoming viewing ray at surface normal, and trace child rays *recursively*.



$$\boldsymbol{\omega}_{\text{out}} = \left(\mathbf{I} - 2\mathbf{n}\mathbf{n}^{\mathsf{T}}\right)\boldsymbol{\omega}_{\text{in}}$$

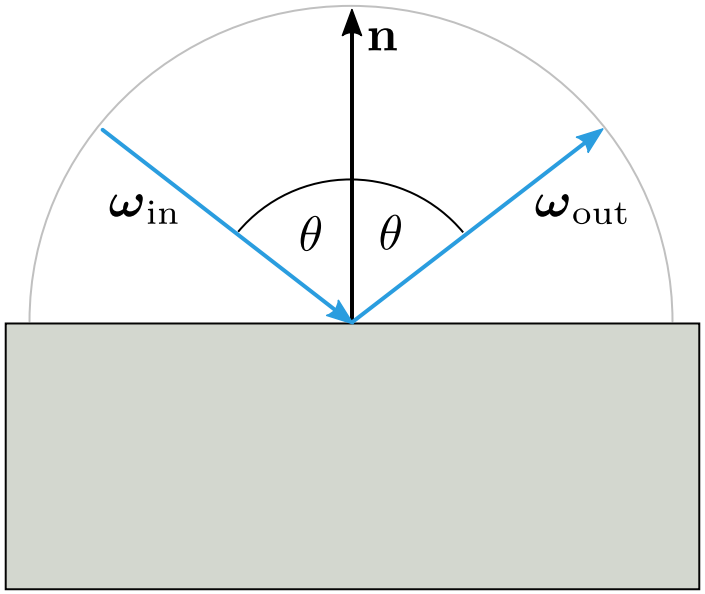


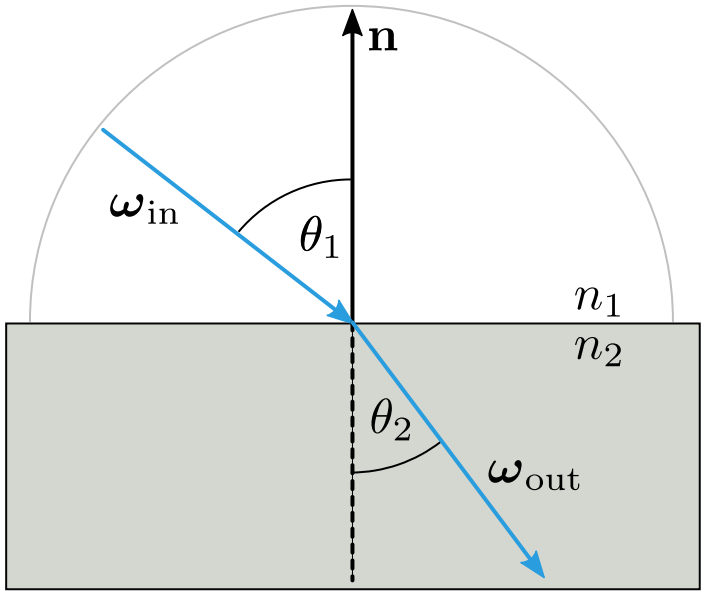
$$n_1 \sin\theta_1 = n_2 \sin\theta_2$$

Snell's law with refraction indices n_1, n_2.

Recursive Ray Tracing

- At each intersection point, reflect and/or refract incoming viewing ray at surface normal, and trace child rays *recursively*.





$$\omega_{out} = \left(\mathbf{I} - 2\mathbf{n}\mathbf{n}^T\right)\omega_{in}$$

Remember precision issue: You need to offset ray origin!

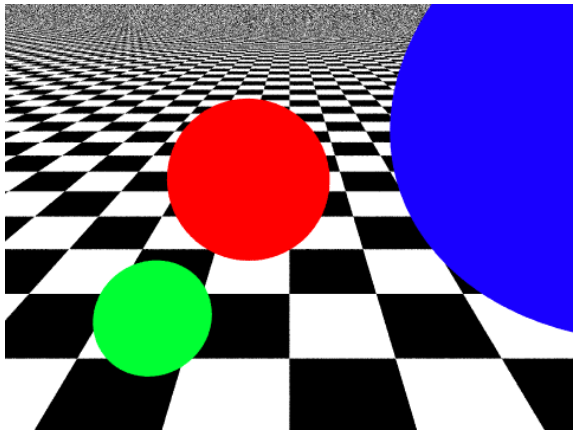
$$n_1 \sin\theta_1 = n_2 \sin\theta_2$$

Snell's law with refraction indices n_1, n_2.

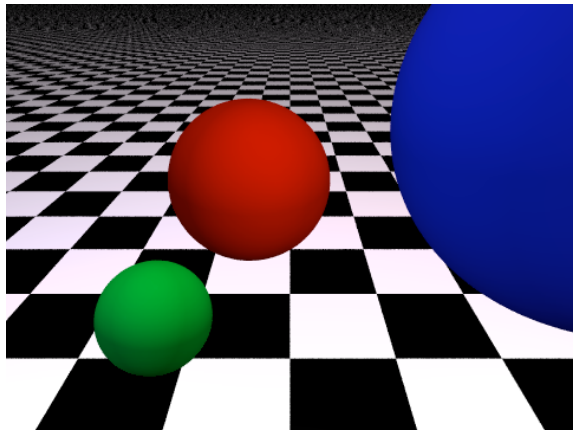
Recursive Ray Tracing

- At each intersection point, reflect and/or refract incoming viewing ray at surface normal, and trace child rays *recursively*.
- The final color is interpolated between local illumination, reflection, and refraction based on material properties.

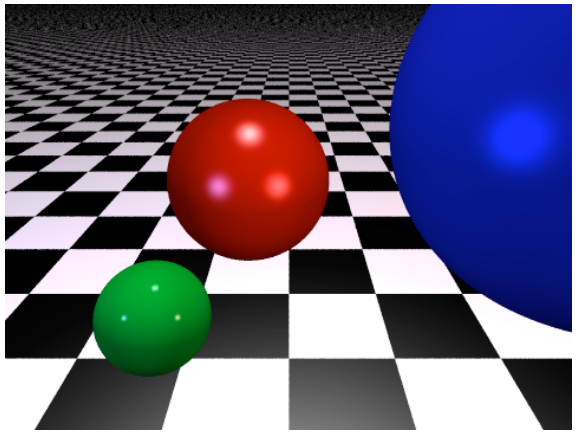
Lighting Computations



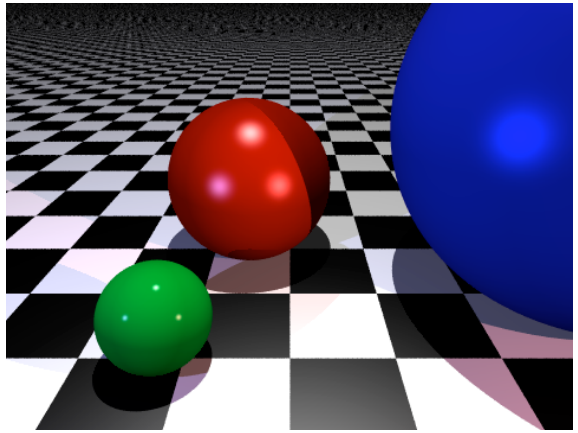
ambient



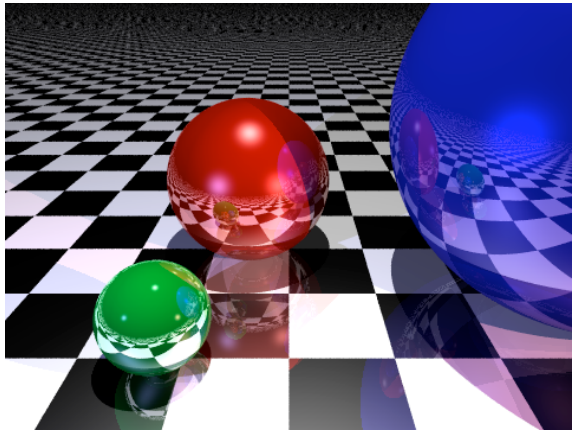
+diffuse



+specular



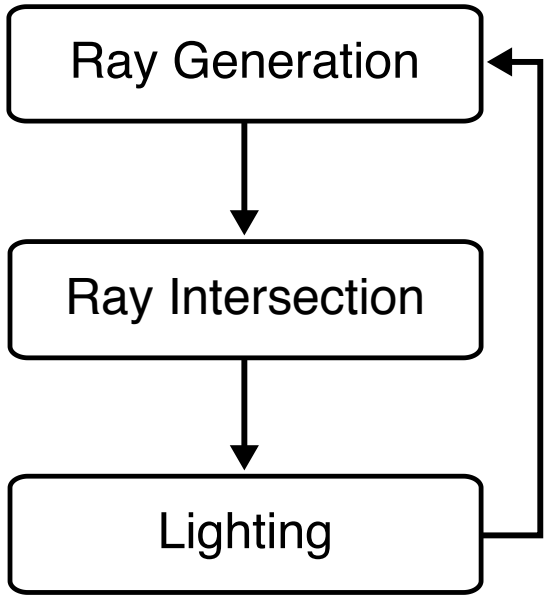
+shadows

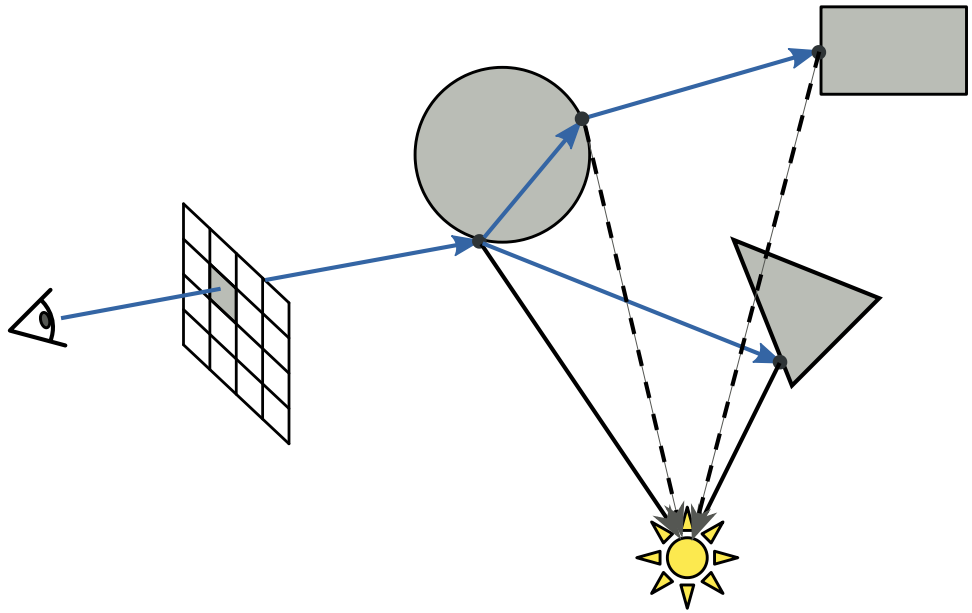


+reflections

Ray Tracing Pipeline

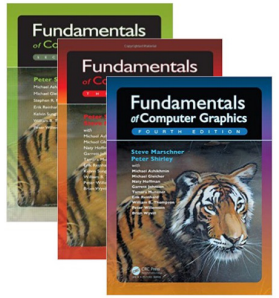
- Now you know about ray generation, ray intersection, lighting computations, and recursive ray tracing.
- That's all you need to implement a complete (basic) ray tracer!
- Next week: Raytracing **triangle meshes** and **acceleration data structures**



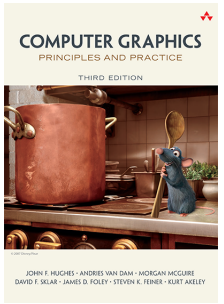


Literature

- Shirley et al.: *Fundamentals of Computer Graphics*, 3rd Edition, AK Peters, 2009.
  - Chapter 10, 20



- Hughes et al.: *Computer Graphics: Principles and Practice*, 3rd Edition, Addison-Wesley, 2014.
  - Chapter 27.5



- Akenine-Möller, Haines, Hoffman: *Real-Time Rendering*, Taylor & Francis, 2008.
  - Chapters 5 and 7