NUSRI Summer Programme 2016

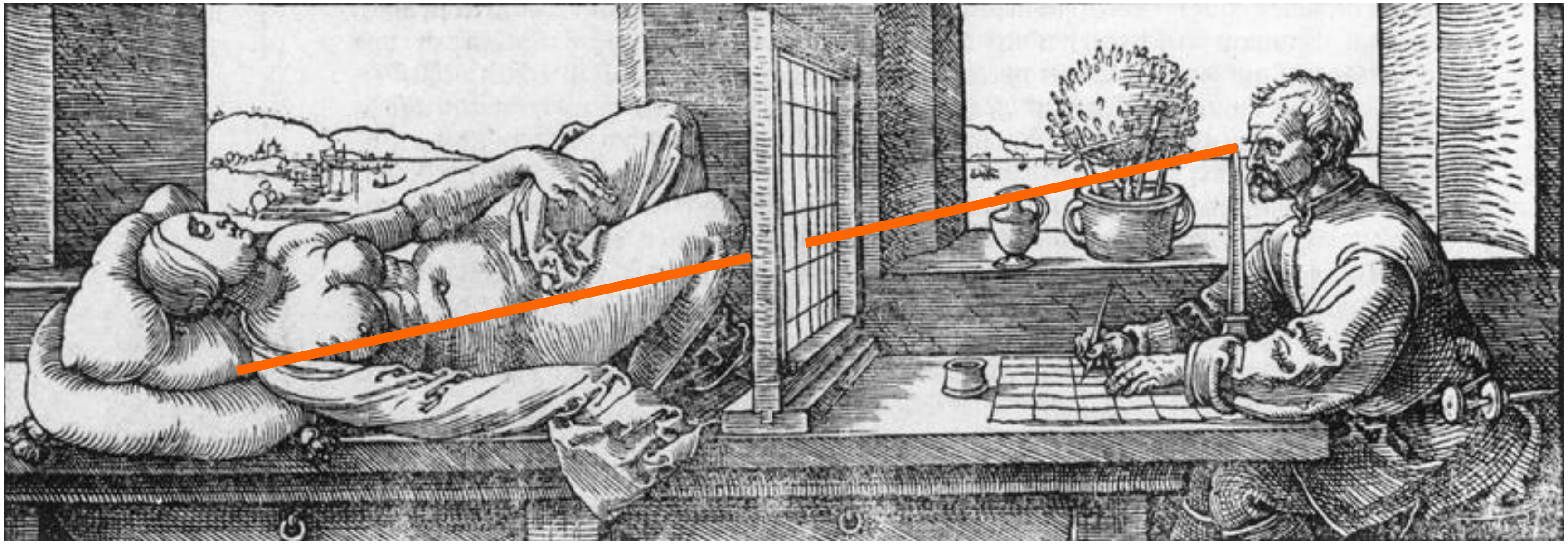
RI3004A
3D Graphics Rendering

Lecture 9
Ray Tracing

School *of* Computing
National University *of* Singapore

The Idea of "Ray Casting"

- In ancient time, it was used for the study of **perspective**



Woodcut by Albrecht Dürer, 16th century

Ray Casting

```
For every pixel
  Construct a ray from the eye
  For every object in the scene
      Find intersection with the ray
      Keep if closest
```

Used for hidden surface removal

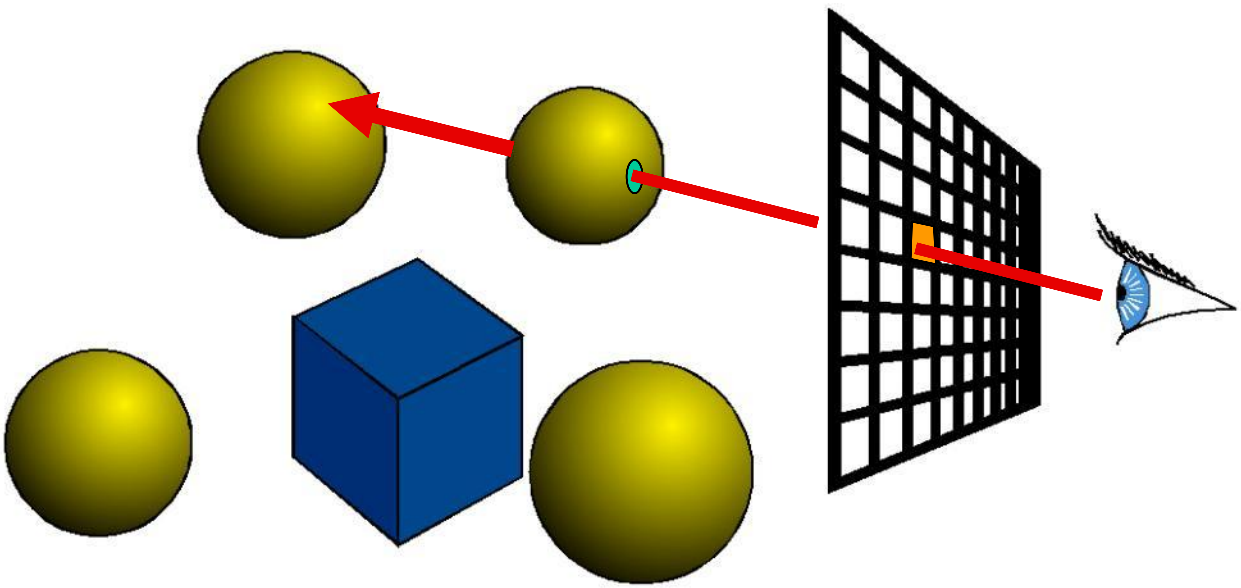


Figure by
Frédo Durand, MIT

Ray Casting and Shading

```
For every pixel
  Construct a ray from the eye
  For every object in the scene
      Find intersection with the ray
      Keep if closest
  Shade depending on light and normal vector
```

Shading can use
Phong reflection model

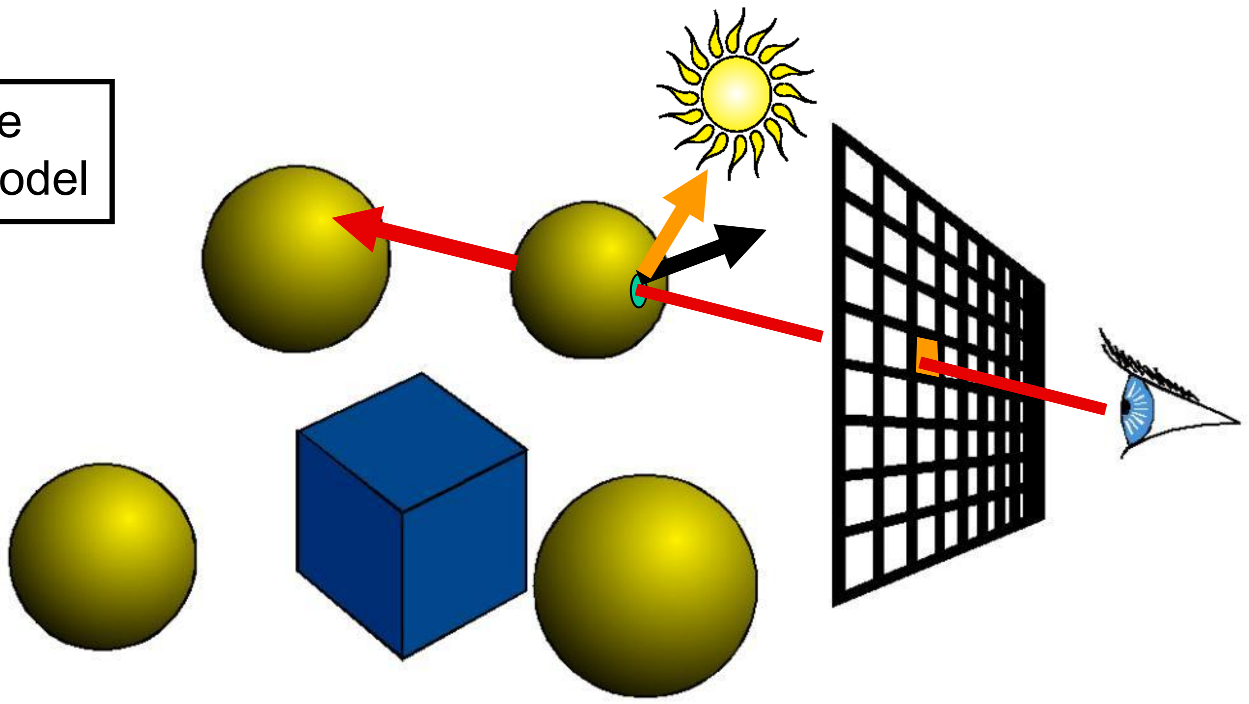


Figure by
Frédo Durand, MIT

Ray Tracing

- From the closest intersection point, **secondary rays** are shot out
  - Reflection ray
  - Refraction ray
  - Shadow rays

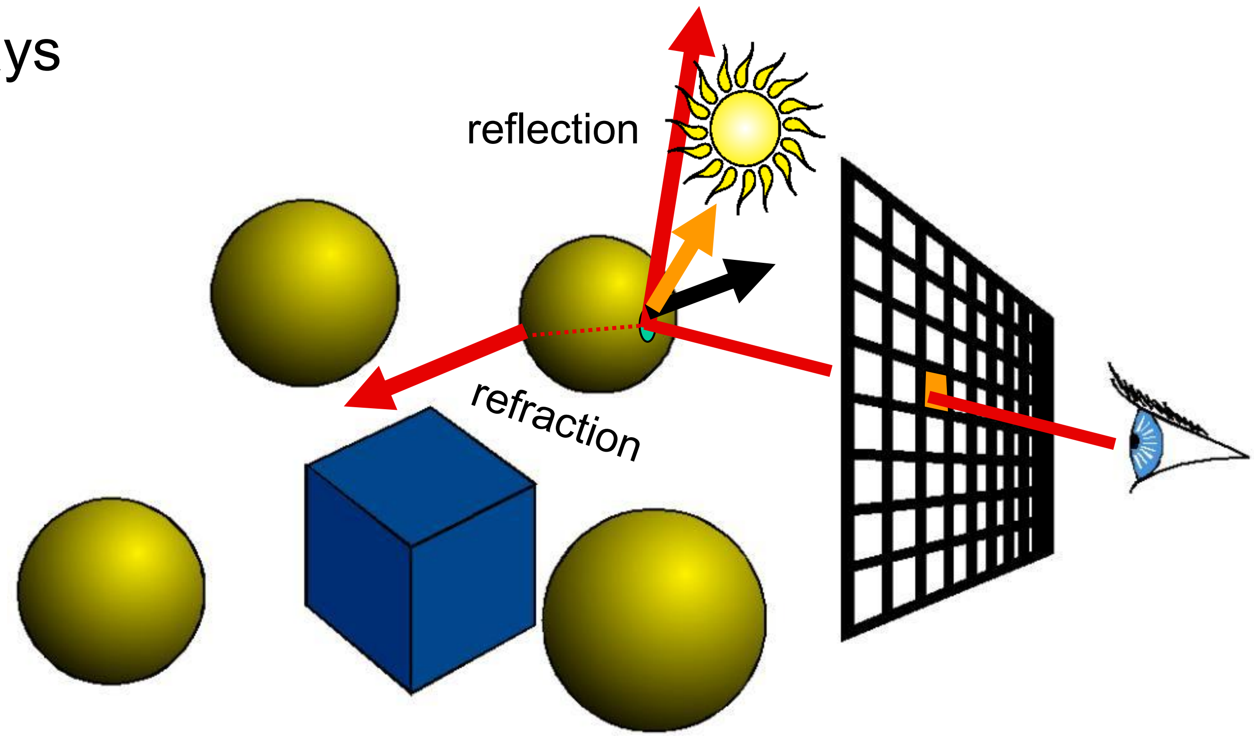
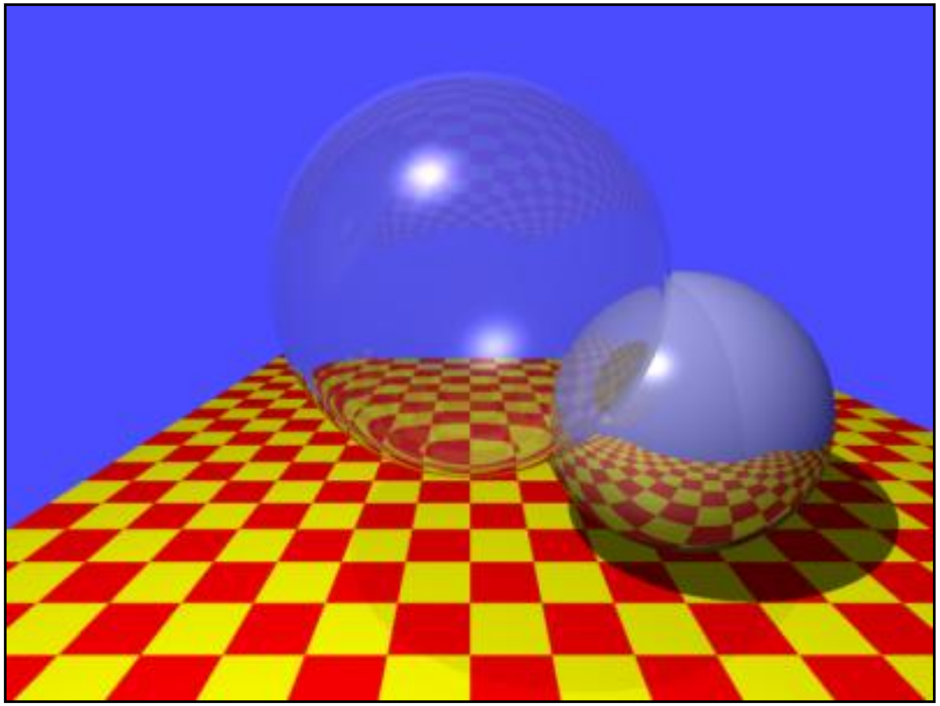


Figure by
Frédo Durand, MIT

Whitted Ray Tracing

- We get
  - Hidden surface removal (from ray casting)
  - Reflection of light
  - Reflection / refraction of other objects
  - Shadows
- All the above are obtained in one single framework
  - No ad-hoc add-on
- However, it simulates only partial global illumination

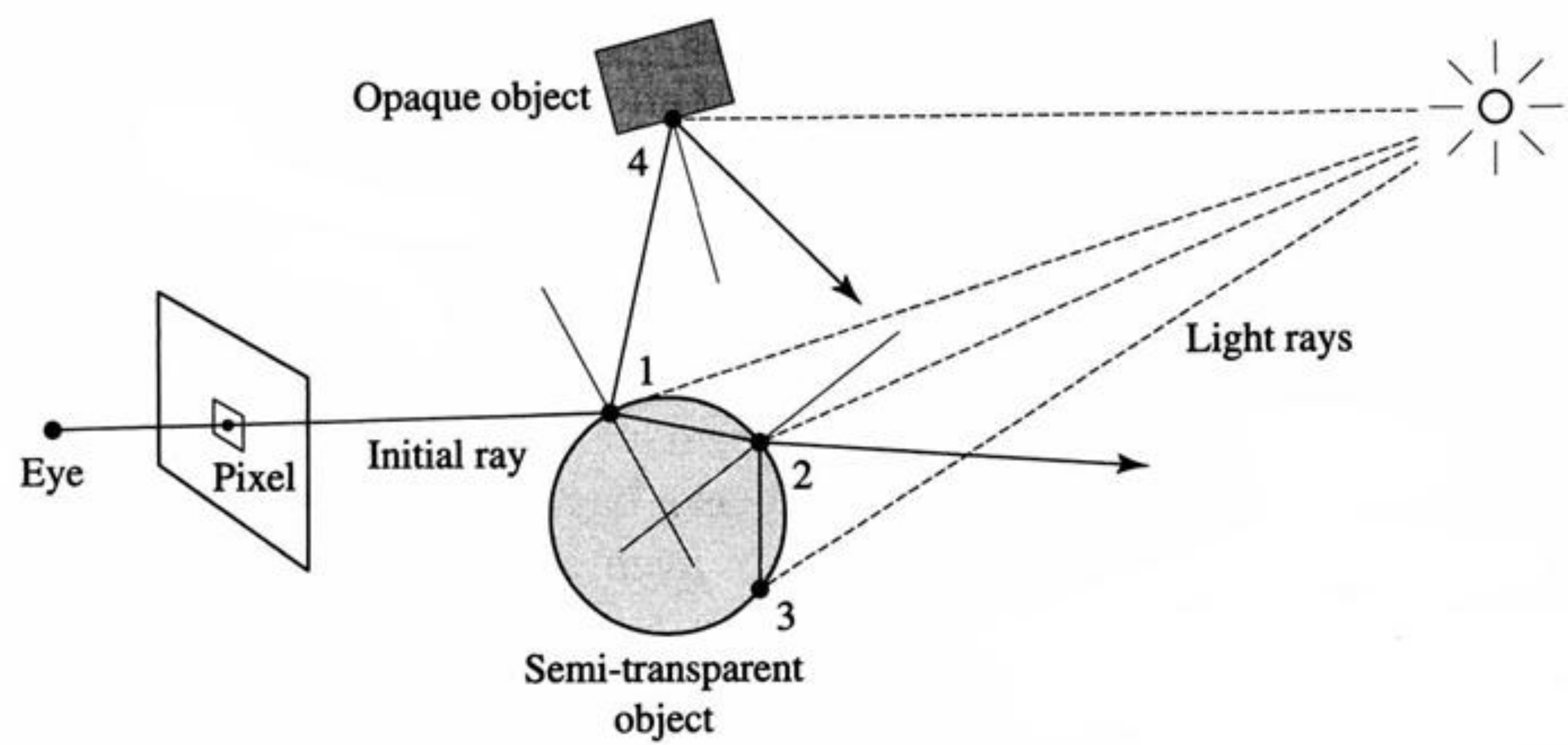
- Also called
  - <u>**Recursive Ray Tracing**</u>



Ray Tracing Details

$$\boldsymbol{I = I_{\text{local}} + k_{\text{rg}}\, I_{\text{reflected}} + k_{\text{tg}}\, I_{\text{transmitted}}}$$

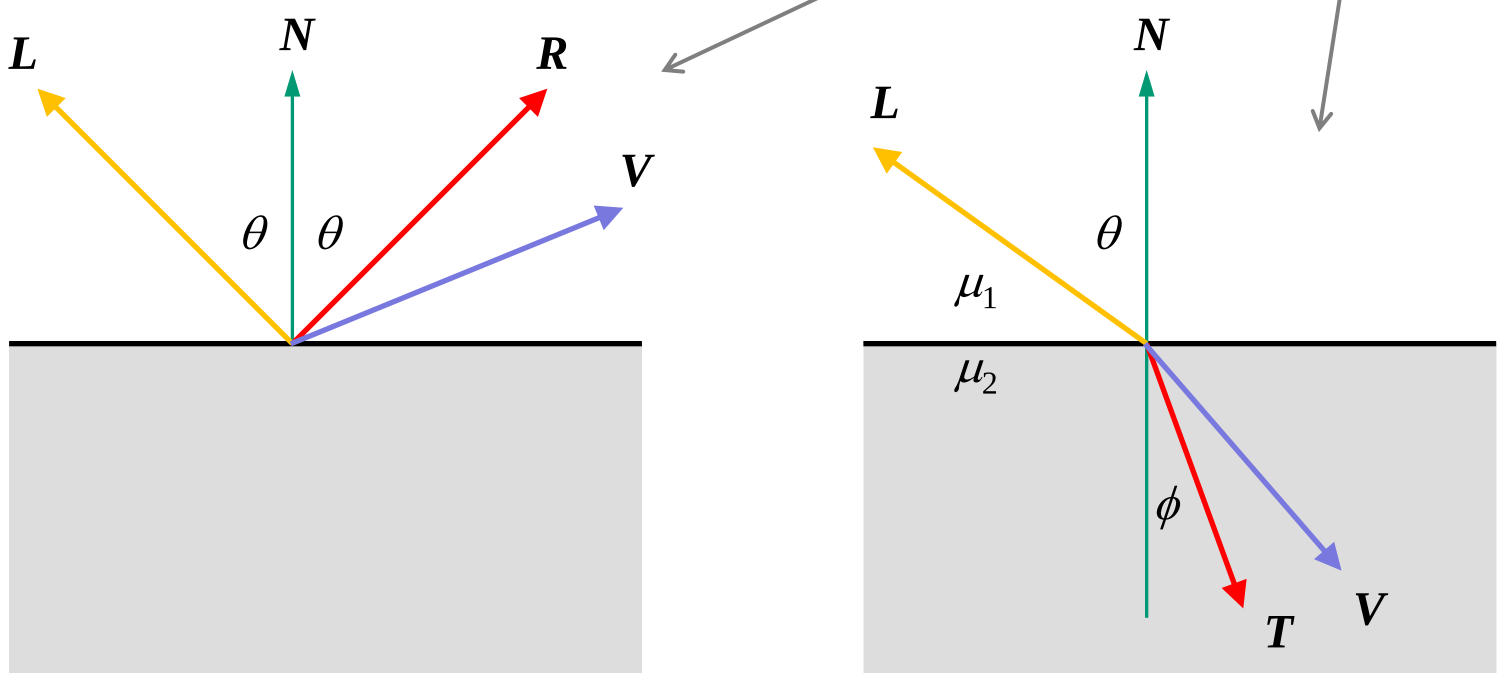
where $I_{\text{local}} = I_{\text{a}}\, k_{\text{a}} + I_{\text{source}}[\, k_{\text{d}}(\boldsymbol{N}\cdot\boldsymbol{L}) + k_{\text{r}}(\boldsymbol{R}\cdot\boldsymbol{V})^{n} + k_{\text{t}}(\boldsymbol{T}\cdot\boldsymbol{V})^{m}\,]$



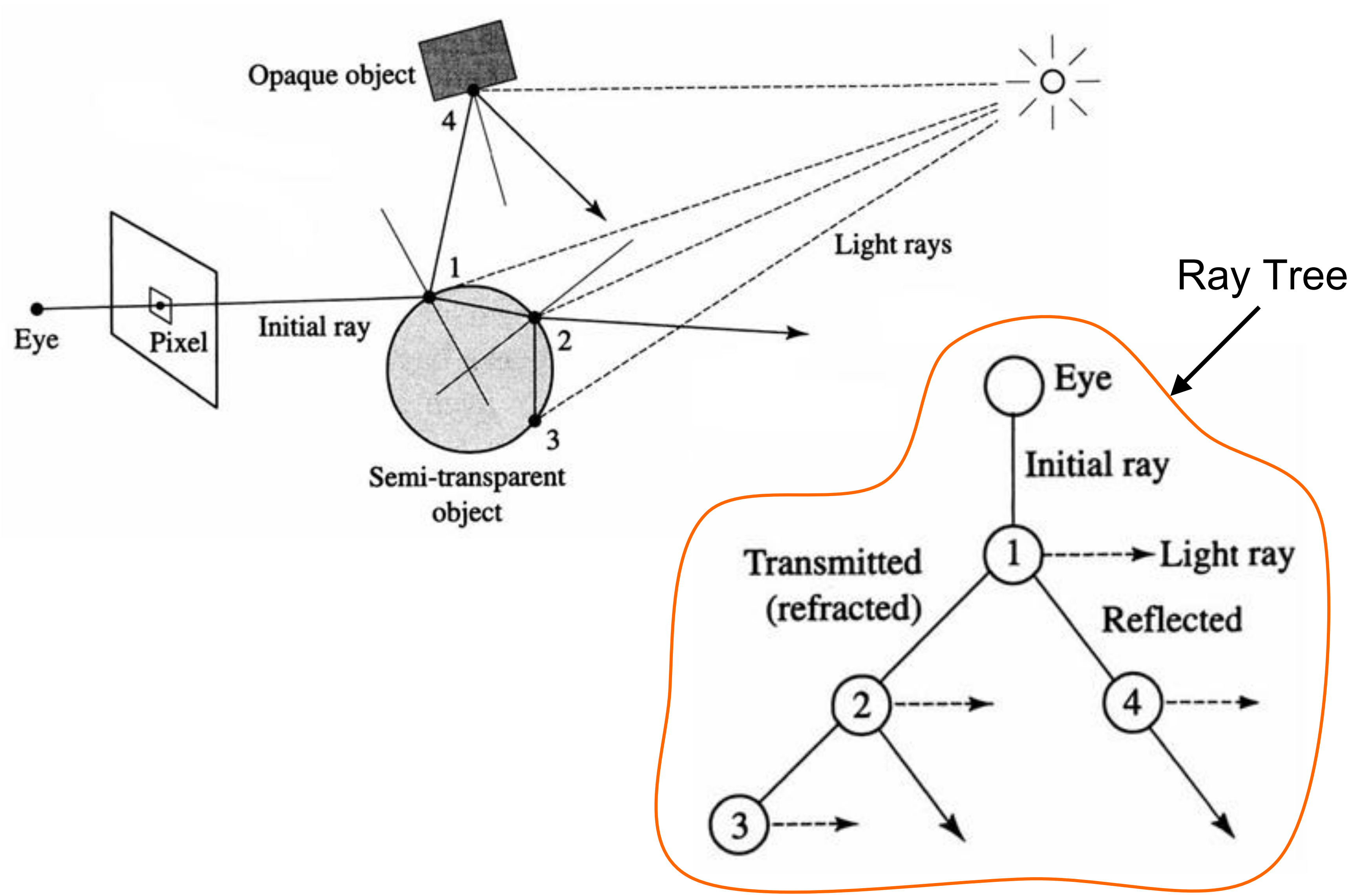
Ray Tracing Details

$$\boldsymbol{I = I_{\mathrm{local}} + k_{\mathrm{rg}}\, I_{\mathrm{reflected}} + k_{\mathrm{tg}}\, I_{\mathrm{transmitted}}}$$

where $I_{\mathrm{local}} = I_{\mathrm{a}}\, k_{\mathrm{a}} + I_{\mathrm{source}}[\, k_{\mathrm{d}}(\boldsymbol{N}\cdot\boldsymbol{L}) + \boldsymbol{k_{\mathrm{r}}(R\cdot V)^{n}} + \boldsymbol{k_{\mathrm{t}}(T\cdot V)^{m}}\,]$

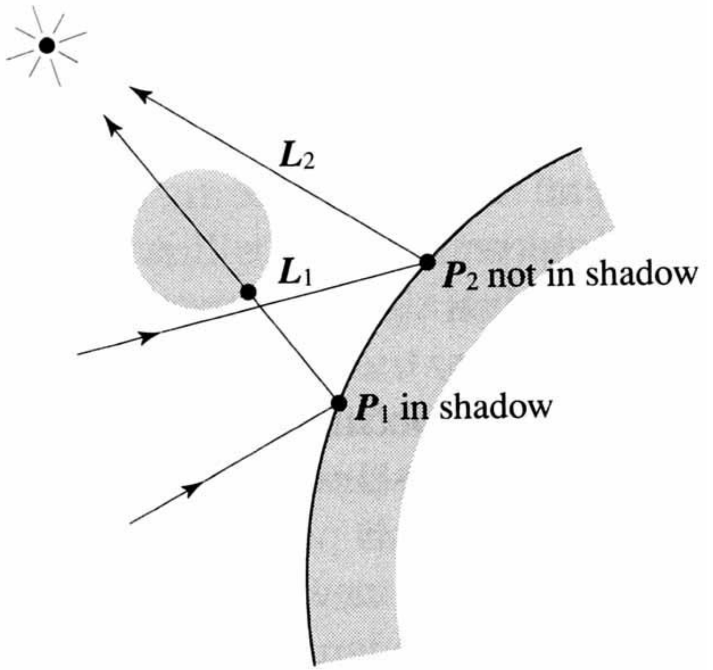


Ray Tree


Opaque object
4
Light rays
1
Eye
Pixel
Initial ray
2
3
Semi-transparent
object
Ray Tree
Eye
Initial ray
Transmitted
(refracted)
1
Light ray
Reflected
2
4
3

Shadow Rays

- Also called
  light rays or **shadow feelers**
- At each surface intersection point, a shadow ray is shot towards each light source to determine any occlusion between light source and surface point
  - Need to find only one opaque occluder to determine occlusion



$$I_{\mathrm{local}} = I_{\mathrm{a}}\, k_{\mathrm{a}} + \boldsymbol{k}_{\mathbf{shadow}}\, I_{\mathrm{source}}[\, k_{\mathrm{d}}(\boldsymbol{N}\cdot\boldsymbol{L}) + k_{\mathrm{r}}(\boldsymbol{R}\cdot\boldsymbol{V})^{n} + k_{\mathrm{t}}(\boldsymbol{T}\cdot\boldsymbol{V})^{m}\,]$$

Shadow Rays

- What if occluder is translucent?
  - Light is attenuated by the k_{tg} of the occluder
  - Refraction of light ray from light source is ignored
  - Both are physically incorrect!
  - Why is this done this way?

Scene Description

- Camera view & image resolution
  - Camera position and orientation in world coordinate frame
    - Similar to **`gluLookAt()`**
  - Field of view
    - Similar to **`gluPerspective()`**, but no need near & far plane
  - Image resolution
    - Number of pixels in each dimension
- Each point light source
  - Position
  - Brightness and color ($I_{\text{source,red}}$, $I_{\text{source,green}}$, $I_{\text{source,blue}}$)
  - A global ambient ($I_{\text{a,red}}$, $I_{\text{a,green}}$, $I_{\text{a,blue}}$)
  - Spotlight is also possible

$$\boxed{\begin{aligned} I_{\text{local}} = {} & I_{\text{a}}\, k_{\text{a}} + \\ & I_{\text{source}}[\, k_{\text{d}}(\boldsymbol{N}\cdot\boldsymbol{L}) + k_{\text{r}}(\boldsymbol{R}\cdot\boldsymbol{V})^{n} + k_{\text{t}}(\boldsymbol{T}\cdot\boldsymbol{V})^{m}\,] \end{aligned}}$$

Scene Description

$$I = I_{\text{local}} + k_{\text{rg}}\, I_{\text{reflected}} + k_{\text{tg}}\, I_{\text{transmitted}}$$

$$\text{where} \quad I_{\text{local}} = I_{\text{a}}\, k_{\text{a}} + I_{\text{source}}[\, k_{\text{d}}(\boldsymbol{N}\cdot\boldsymbol{L}) + k_{\text{r}}(\boldsymbol{R}\cdot\boldsymbol{V})^{n} + k_{\text{t}}(\boldsymbol{T}\cdot\boldsymbol{V})^{m}\,]$$

- Each object surface material
  - k_{rg}, k_{tg}, k_{a}, k_{d}, k_{r}, k_{t} (each is a RGB vector)
  - n, m
  - Refractive index μ if $k_{\text{tg}} \neq \mathbf{0}$ or $k_{\text{t}} \neq \mathbf{0}$

Can use different μ for R, G & B.

- Objects
  - Implicit representations (e.g. plane, sphere, quadrics)
  - Polygon
  - Parametric (e.g. bicubic Bezier patches)
  - Volumetric

Recursive Ray Tracing

- For each reflection/refraction ray spawned, we can trace it just like tracing the original ray
- Implemented using recursion

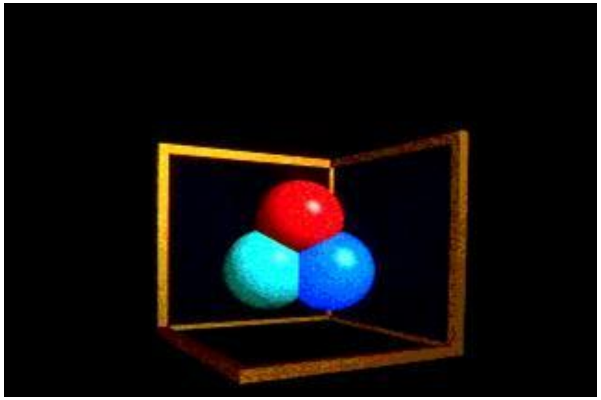
$$
\begin{aligned}
I(\boldsymbol{P}) &= I_{\mathrm{local}}(\boldsymbol{P}) + I_{\mathrm{global}}(\boldsymbol{P}) \\
&= I_{\mathrm{local}}(\boldsymbol{P}) + k_{\mathrm{rg}}\, I(\boldsymbol{P}_{\mathrm{r}}) + k_{\mathrm{tg}} I(\boldsymbol{P}_{\mathrm{t}})
\end{aligned}
$$

where:

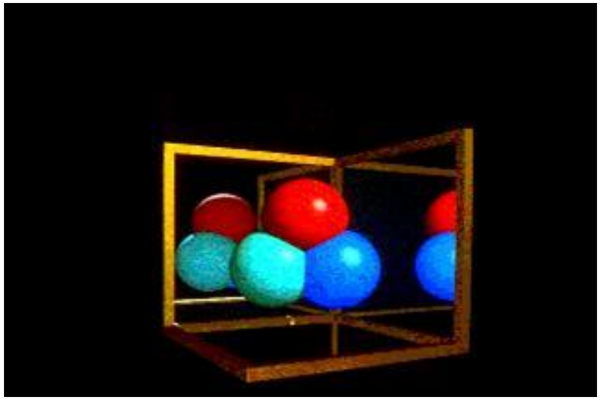
$\boldsymbol{P}$ is the hit point
$\boldsymbol{P}_{\mathrm{r}}$ is the hit point discovered by tracing the reflected ray from $\boldsymbol{P}$
$\boldsymbol{P}_{\mathrm{t}}$ is the hit point discovered by tracing the transmitted ray from $\boldsymbol{P}$
k_{rg} is the global reflection coefficient
k_{tg} is the global transmitted coefficient

Recursive Ray Tracing

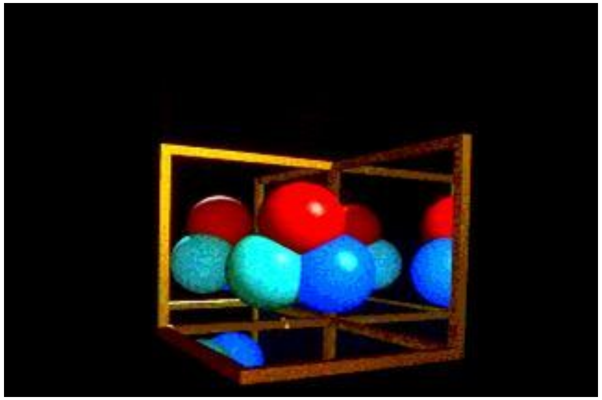
0 recursion



1 recursions



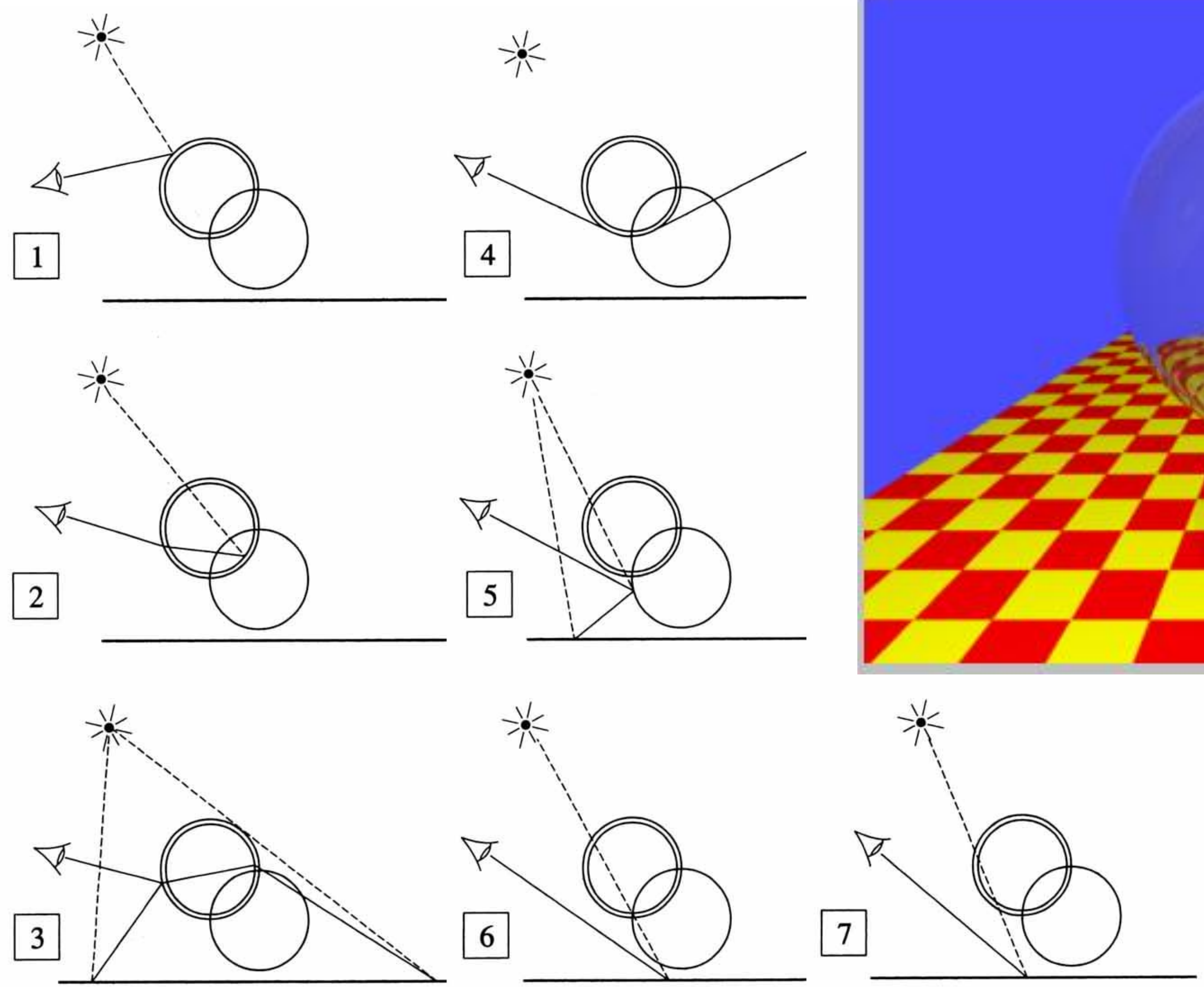
2 recursions

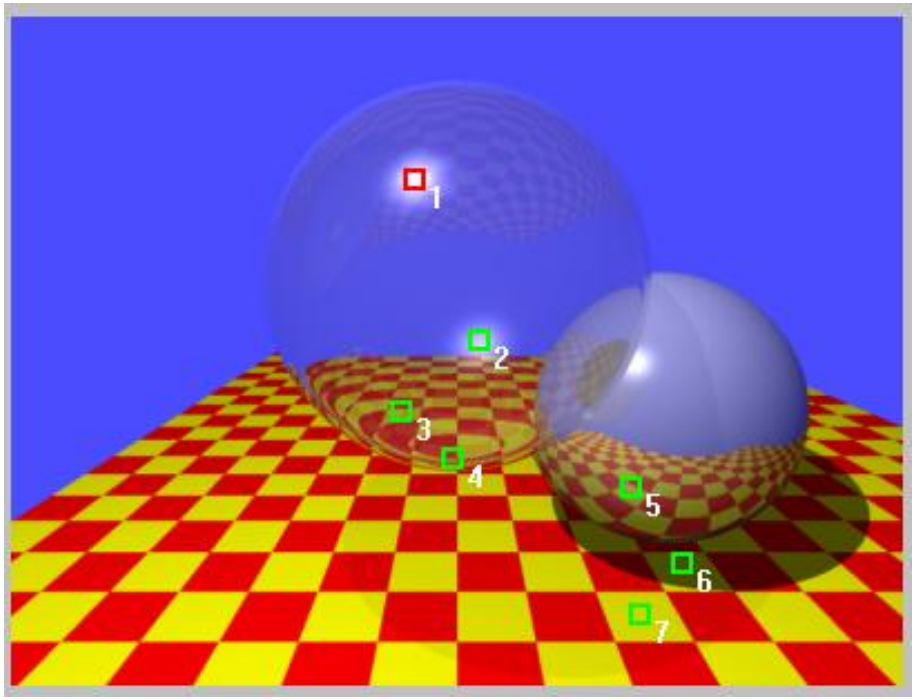


Recursive Ray Tracing

- When to stop recursion?
  - When the surface is totally diffuse (and opaque)
  - When reflected/refracted ray hits nothing
  - When maximum recursion depth is reached
  - When the contribution of the reflected/refracted ray to the color at the top level is too small
    - $(k_{rg1} \mid k_{tg1}) \times \ldots \times (k_{rg(n-1)} \mid k_{tg(n-1)}) < \mathit{threshold}$

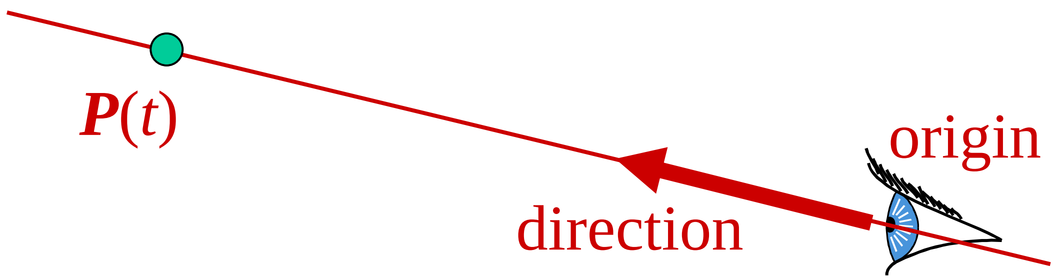
Adventures of Seven Rays



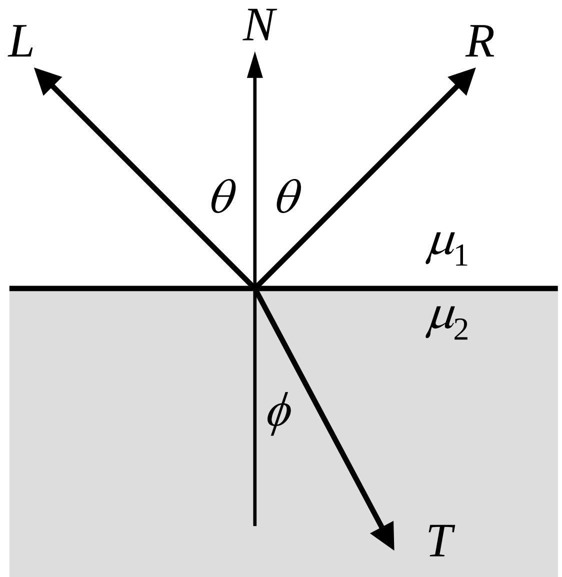


Ray Representations

- Finding ray-object **intersection** and computing **surface normal** is central to ray tracing
- Ray representations
  - Two 3D vectors
    - Ray origin position
    - Ray direction vector
  - Parametric form
    - $\boldsymbol{P}(t) = \text{origin} + t \times \text{direction}$



Computing Reflection / Refraction Rays



Reflection

$$R = 2N\cos\theta - L$$
$$= 2(N \bullet L)N - L$$

Snell's law

$$\mu_1 \sin\theta = \mu_2 \sin\phi$$

Refraction

$$\mu = \mu_1 / \mu_2$$

$$T = -\mu L + \left(\mu\cos\theta - \sqrt{1-\mu^2\left(1-\cos^2\theta\right)}\right)N$$
$$= -\mu L + \left(\mu(N \bullet L) - \sqrt{1-\mu^2\left(1-(N \bullet L)^2\right)}\right)N$$

Ray-Plane Intersection

- Plane is often represented in implicit form
  - $Ax + By + Cz + D = 0$
  - Equivalent to $\boldsymbol{N} \cdot \boldsymbol{P} + D = 0$
    - where $\boldsymbol{N} = [A\ B\ C]^T$ and $\boldsymbol{P} = [x\ y\ z]^T$
- To find ray-plane intersection, substitute ray equation $\boldsymbol{P}(t)$ into plane equation
  - We get $\boldsymbol{N} \cdot \boldsymbol{P}(t) + D = 0$
  - Solve for t to get t_0
  - If t_0 is infinity, no intersection (ray is parallel to plane)
  - Intersection point is $\boldsymbol{P}(t_0)$
  - Verify that intersection is not behind ray origin, i.e. $t_0 > 0$
- The normal at the intersection is $\boldsymbol{N}$ (or $-\boldsymbol{N}$)

Ray-Sphere Intersection

- Sphere (centered at origin) is often represented in implicit form
  - $x^2 + y^2 + z^2 - r^2 = 0$
  - Equivalent to $\boldsymbol{P}\cdot\boldsymbol{P} - r^2 = 0$
    - where $\boldsymbol{P} = [x\ y\ z]^T$
- To find ray-sphere intersection, substitute ray equation $\boldsymbol{P}(t)$ into sphere equation
  - We get $\boldsymbol{P}(t)\cdot\boldsymbol{P}(t) - r^2 = 0$

$$\boldsymbol{P}(t)\cdot\boldsymbol{P}(t) - r^2 = 0$$

$$(\boldsymbol{R}_o + t\boldsymbol{R}_d)\cdot(\boldsymbol{R}_o + t\boldsymbol{R}_d) - r^2 = 0$$

$$\boldsymbol{R}_d\cdot\boldsymbol{R}_d\, t^2 + 2\,\boldsymbol{R}_d\cdot\boldsymbol{R}_o\, t + \boldsymbol{R}_o\cdot\boldsymbol{R}_o - r^2 = 0$$

$\boldsymbol{R}_o$ is ray origin

$\boldsymbol{R}_d$ is ray direction

Ray-Sphere Intersection

- It is a quadratic equation in the form $at^2 + bt + c = 0$
  - $a = \boldsymbol{R}_\mathrm{d} \cdot \boldsymbol{R}_\mathrm{d} = 1$ (since $|\boldsymbol{R}_\mathrm{d}| = 1$)
  - $b = 2\, \boldsymbol{R}_\mathrm{d} \cdot \boldsymbol{R}_\mathrm{o}$
  - $c = \boldsymbol{R}_\mathrm{o} \cdot \boldsymbol{R}_\mathrm{o} - r^2$
- Discriminant, $d = b^2 - 4ac$
- Solutions, $t_\pm = (-b \pm \sqrt{d}\,) / (2a)$
- Three cases to consider depending on value of d
  - What are the 3 cases? What do they correspond to?
- Choose t_0 as the closest positive t value (t_+ or t_-)
- The normal at the intersection point is $\boldsymbol{P}(t_0) / |\boldsymbol{P}(t_0)|$

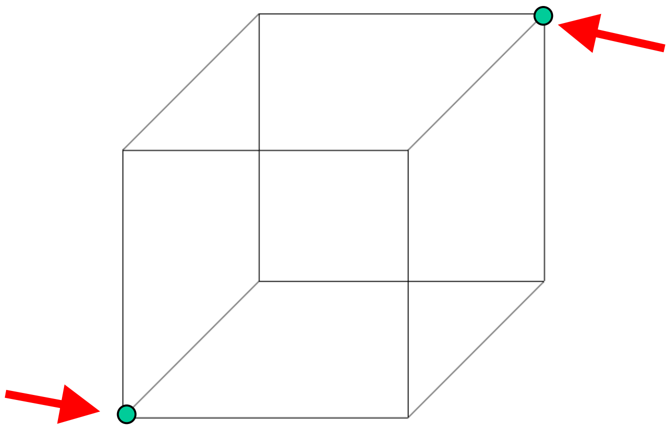
Ray-Sphere Intersection

- Very easy to compute, that is why most ray tracing images have spheres

- What if sphere is not centered at origin?
  - Transform the ray to the sphere's local coordinate frame
  - How to transform? Need to consider rotation?

Ray-Box Intersection

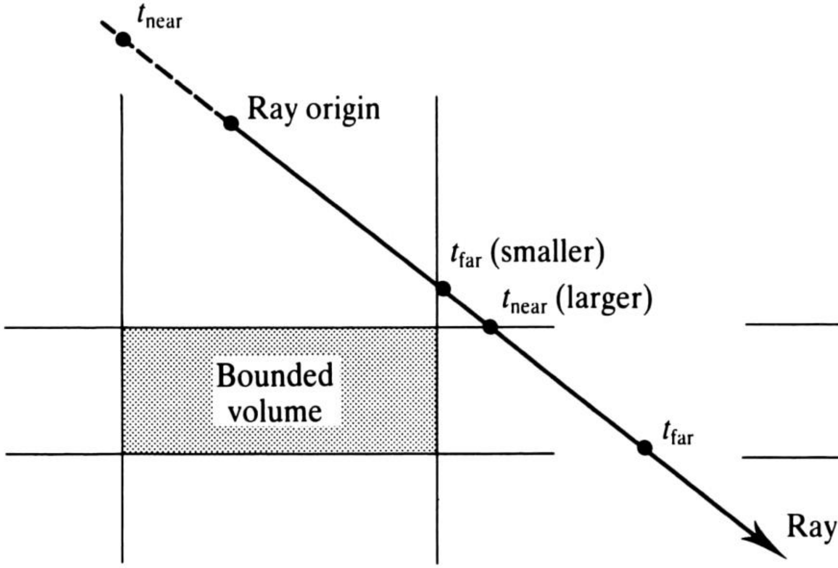
- A 3D box is defined by 3 pairs of parallel planes, where each pair is orthogonal to the other two pairs
- If 3D box is **axis-aligned**, only need to specify the coordinates of the two diagonally opposite corners
  - The 3 pairs of planes can be deduced easily

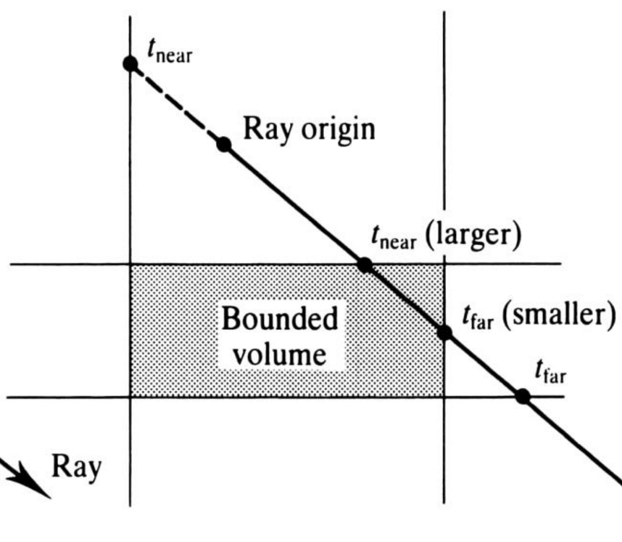


Ray-Box Intersection

- To find ray-box intersection
  - For each pair of parallel plane, find the distance to the first plane (t_{near}) and to the second plane (t_{far})
  - Keep the largest t_{near} so far, and smallest t_{far} so far
  - If largest t_{near} > smallest t_{far}, no intersection
  - Otherwise, the intersection is at $\boldsymbol{P}$(largest t_{near})

How to find normal vector?





Ray-Triangle Intersection

- Finding intersection between a ray and a general polygon is difficult
  - 1) Compute ray-plane intersection
  - 2) Determine whether intersection is within polygon
    - Tedious for non-convex polygon
  - Interpolation of attributes at the vertices are not well-defined
- Much easier to find ray-triangle intersection
  - Can use the **barycentric coordinates** method
  - Interpolation of attributes at the vertices are well-defined using the barycentric coordinates

Barycentric Coordinates

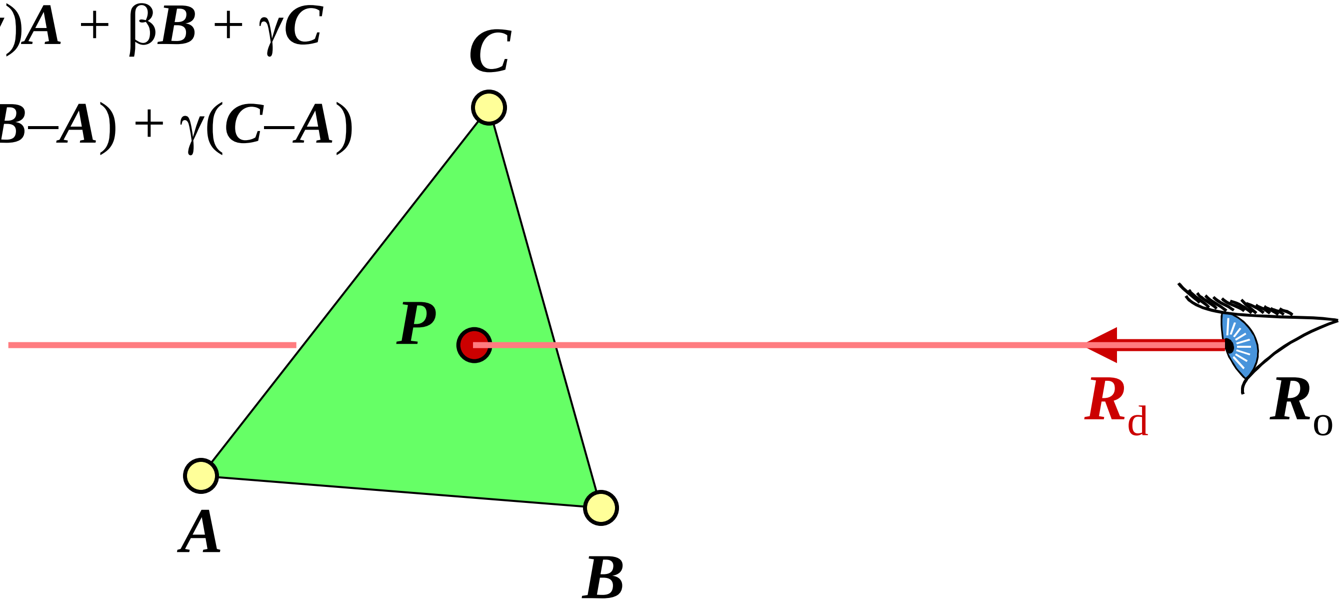
- The barycentric coordinates of a point $\boldsymbol{P}$ on a triangle $\boldsymbol{ABC}$ is (α, β, γ) such that

$$\boldsymbol{P} = \alpha\boldsymbol{A} + \beta\boldsymbol{B} + \gamma\boldsymbol{C} \quad \text{where} \quad \alpha + \beta + \gamma = 1 \quad \text{and} \quad 0 \le \alpha, \beta, \gamma \le 1$$

- We can rewrite it as

$$\boldsymbol{P} = (1-\beta-\gamma)\boldsymbol{A} + \beta\boldsymbol{B} + \gamma\boldsymbol{C}$$

$$\boldsymbol{P} = \boldsymbol{A} + \beta(\boldsymbol{B}-\boldsymbol{A}) + \gamma(\boldsymbol{C}-\boldsymbol{A})$$



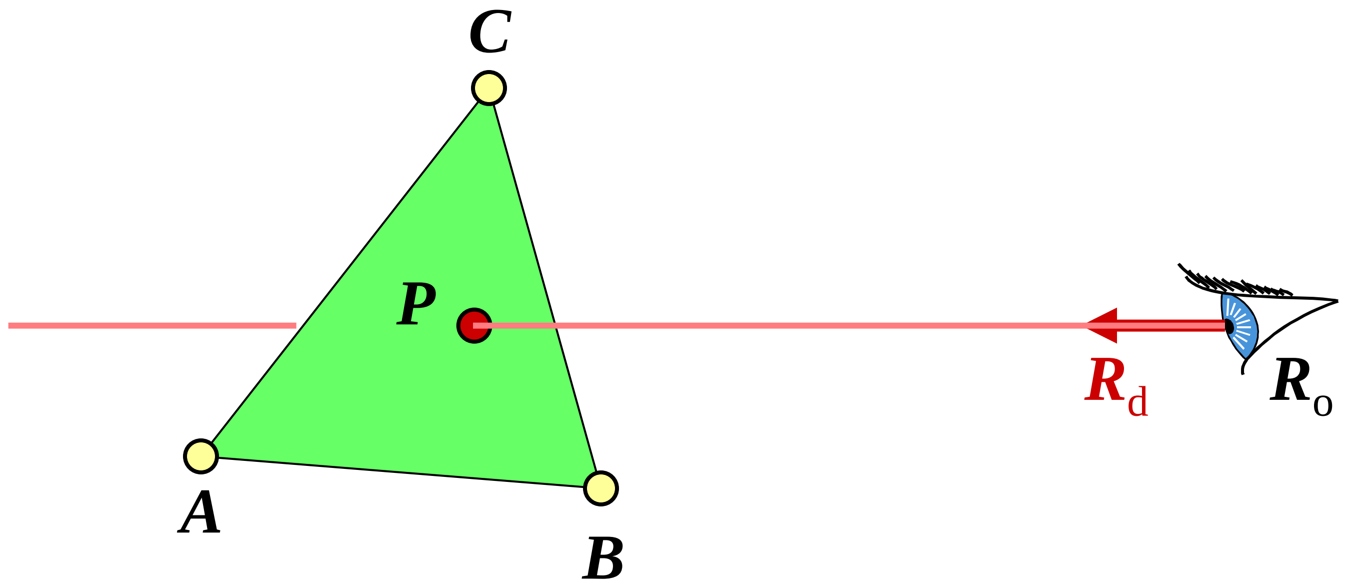
Barycentric Coordinates

- To find ray-triangle intersection, we let

$$\boldsymbol{P}(t) = \boldsymbol{A} + \beta(\boldsymbol{B}-\boldsymbol{A}) + \gamma(\boldsymbol{C}-\boldsymbol{A})$$

$$\boldsymbol{R}_{\mathrm{o}} + t\boldsymbol{R}_{\mathrm{d}} = \boldsymbol{A} + \beta(\boldsymbol{B}-\boldsymbol{A}) + \gamma(\boldsymbol{C}-\boldsymbol{A})$$

- Solve for t, β and γ
- Intersection if $\beta + \gamma < 1$ & $\beta, \gamma > 0$ & $t > 0$



Barycentric Coordinates

- Expand $\boldsymbol{R}_\mathrm{o} + t\boldsymbol{R}_\mathrm{d} = \boldsymbol{A} + \beta(\boldsymbol{B}-\boldsymbol{A}) + \gamma(\boldsymbol{C}-\boldsymbol{A})$

$$\left.\begin{aligned} R_{ox} + tR_{dx} &= A_x + \beta(B_x - A_x) + \gamma(C_x - A_x) \\ R_{oy} + tR_{dy} &= A_y + \beta(B_y - A_y) + \gamma(C_y - A_y) \\ R_{oz} + tR_{dz} &= A_z + \beta(B_z - A_z) + \gamma(C_z - A_z) \end{aligned}\right\} \text{3 equations, 3 unknowns}$$

- Regroup and write in matrix form

$$\begin{bmatrix} A_x - B_x & A_x - C_x & R_{dx} \\ A_y - B_y & A_y - C_y & R_{dy} \\ A_z - B_z & A_z - C_z & R_{dz} \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \\ t \end{bmatrix} = \begin{bmatrix} A_x - R_{ox} \\ A_y - R_{oy} \\ A_z - R_{oz} \end{bmatrix}$$

Barycentric Coordinates

- Use Cramer's Rule to solve for t, β and γ

$$\beta = \frac{\begin{vmatrix} A_x - R_{ox} & A_x - C_x & R_{dx} \\ A_y - R_{oy} & A_y - C_y & R_{dy} \\ A_z - R_{oz} & A_z - C_z & R_{dz} \end{vmatrix}}{|A|} \qquad \gamma = \frac{\begin{vmatrix} A_x - B_x & A_x - R_{ox} & R_{dx} \\ A_y - B_y & A_y - R_{oy} & R_{dy} \\ A_z - B_z & A_z - R_{oz} & R_{dz} \end{vmatrix}}{|A|}$$

$$t = \frac{\begin{vmatrix} A_x - B_x & A_x - C_x & A_x - R_{ox} \\ A_y - B_y & A_y - C_y & A_y - R_{oy} \\ A_z - B_z & A_z - C_z & A_z - R_{oz} \end{vmatrix}}{|A|}$$

| | denotes the determinant

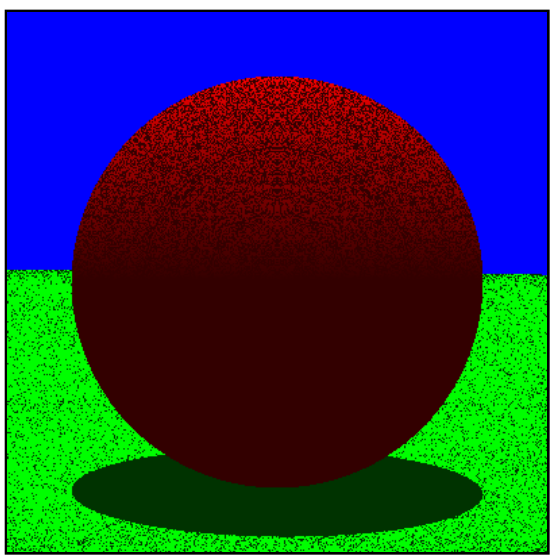
Advantages of Barycentric Intersection

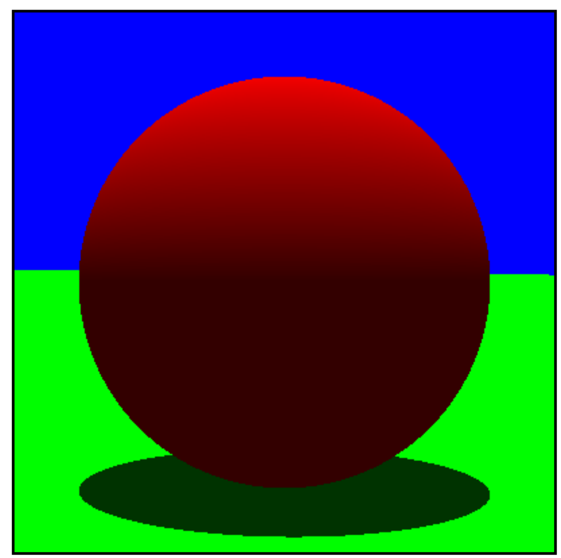
- Efficient
- No need to store plane equation
- Barycentric coordinates are useful for linear interpolation of normal vectors, texture coordinates, and other attributes at the vertices
  - For example, the interpolated normal at $\boldsymbol{P}$ is $\boldsymbol{N_P} = (1-\beta-\gamma)\boldsymbol{N_A} + \beta\boldsymbol{N_B} + \gamma\boldsymbol{N_C}$ (should do a normalization)

The "Epsilon" Problem

- Should not accept intersection for very small positive t
  - May falsely intersect the surface at the ray origin
  - **Method 1:** Use an epsilon value $\varepsilon > 0$, and accept an intersection only if its $t > \varepsilon$
  - **Method 2:** When a new ray is spawned, advanced the ray origin by an epsilon distance ε in the ray direction

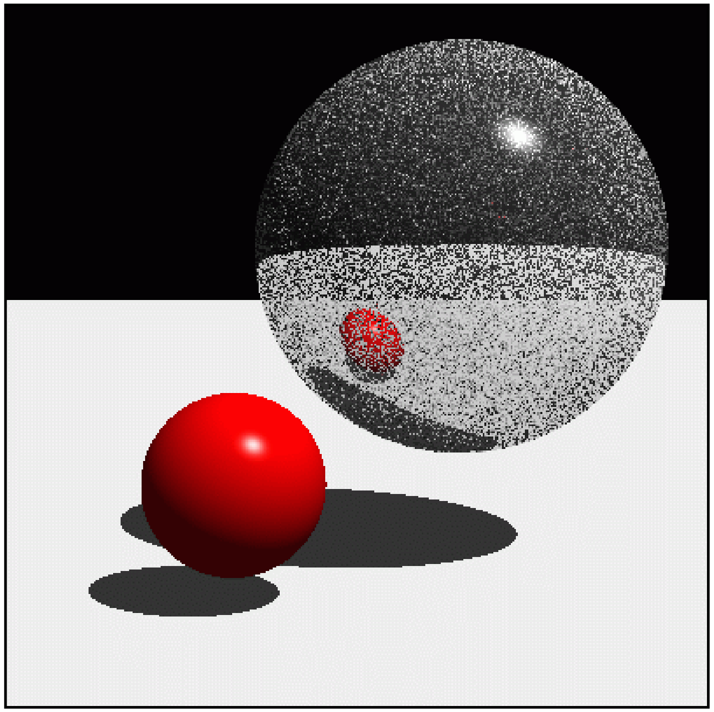
without ε



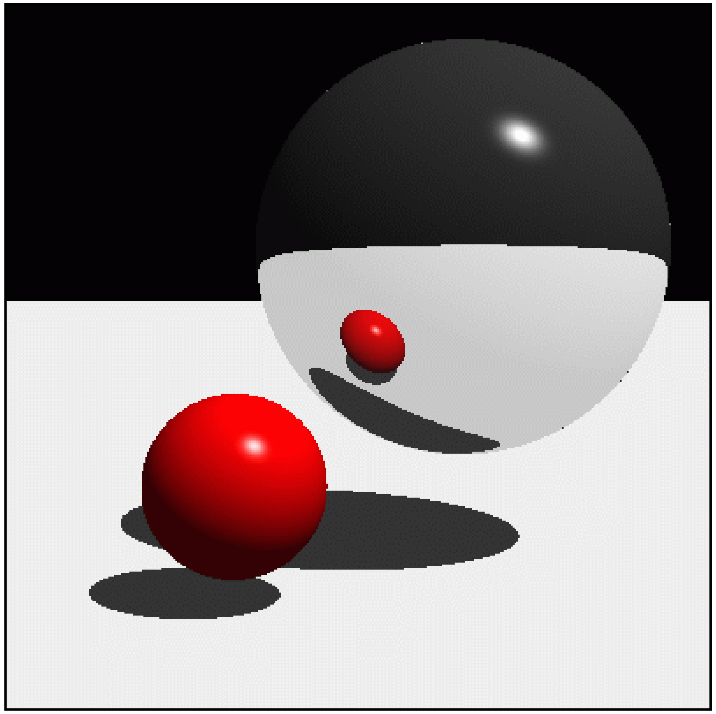


with ε

The "Epsilon" Problem



without ε



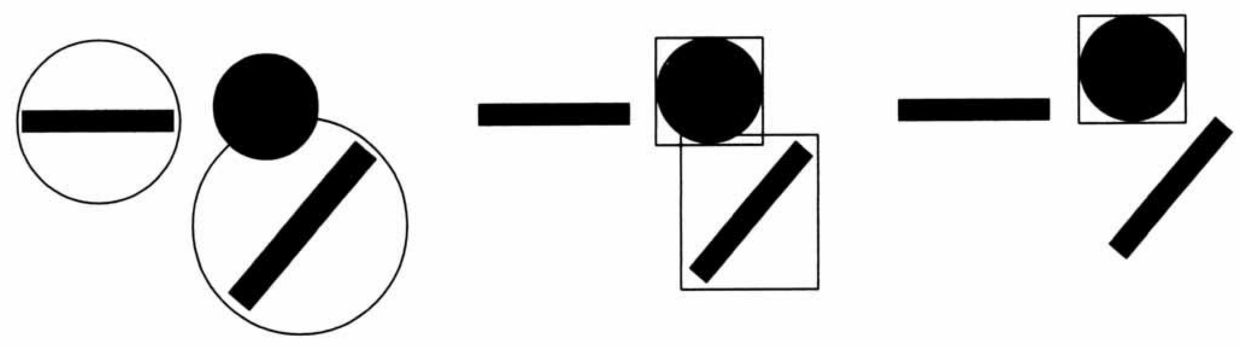
with ε

Ray Tracing Acceleration

- Most ray tracing research have been in
  - Acceleration techniques for ray-scene intersection
  - Extension to simulate more complete global illumination (in a later lecture)
  - Real-time ray tracing!
- Some common acceleration techniques
  - Adaptive recursion depth control
  - First-hit speed-up using z-buffer method
    - Can use item buffer to identify first-hit object at each pixel
  - Bounding volumes
  - Bounding volume hierarchies
  - Spatial subdivision

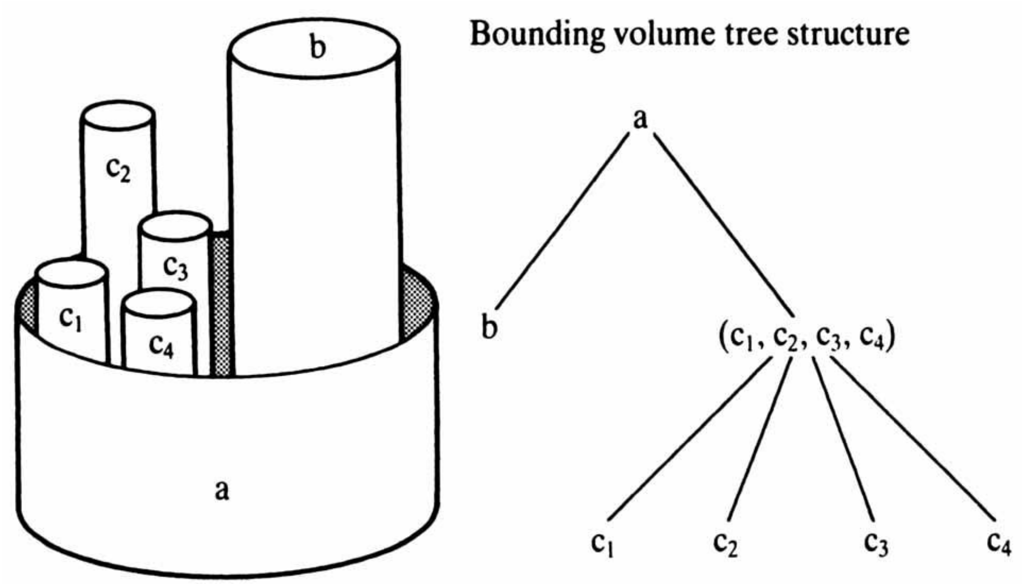
Bounding Volumes

- Use a simple shape to enclose each more complex object
  - If ray does not intersect bounding volume, no need to test complex object (quick reject)
  - Simple shapes are efficient for testing ray intersection
    - Common bounding volumes are spheres, AABBs (axis-aligned bounding boxes), and OBBs (oriented bounding boxes)
  - However, there is trade-off between intersection efficiency and tightness



Bounding Volume Hierarchy

- Can organized bounding volumes into hierarchy



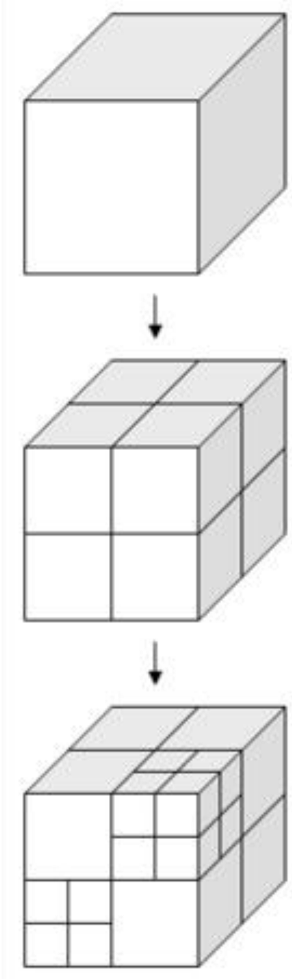
- However, good hierarchies are usually constructed manually

Spatial Subdivision

- Subdivide 3D space into regions, and associate each region with a list of objects that occupy (fully or partially) the region
- When a ray is traced into a region, query the object list and perform intersection tests with the objects
- Since we are looking for the nearest intersection, the ray should be traced in a front-to-back order through the regions
- Common spatial subdivisions for ray tracing
  - Uniform grid
  - Octree
  - BSP

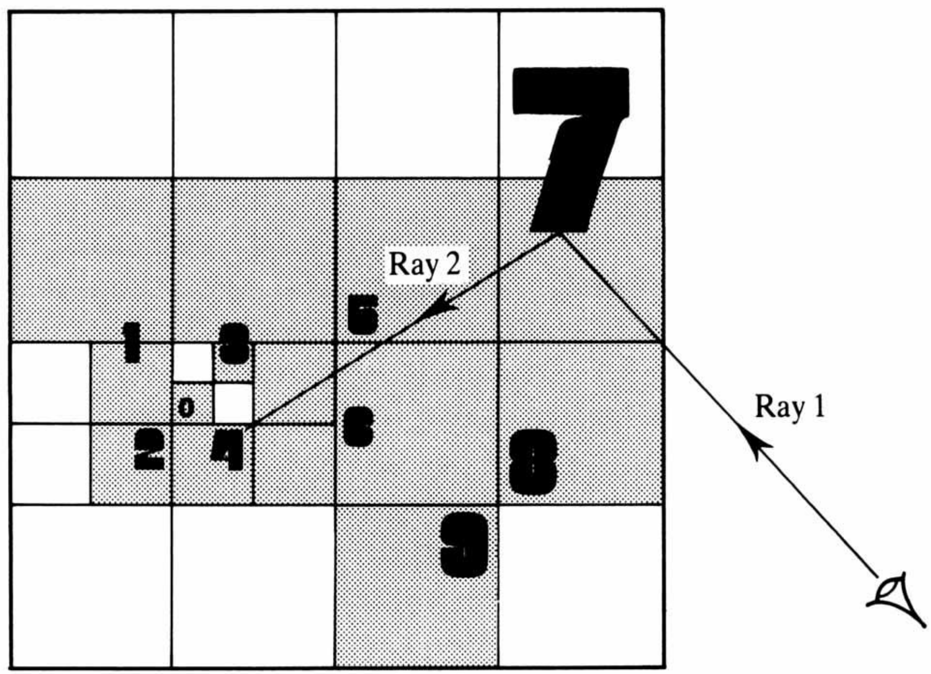
Octree

- Each cubic region is conditionally and recursively subdivided into 8 equal sub-regions
  - Different possible conditions for subdivision
    - Scheme 1: Subdivide a cell if it is occupied by more than one object
    - Scheme 2: Subdivide a cell if it is occupied by any object until the maximum allowable depth
- Ray-cell intersection can be easily tested in front-to-back order

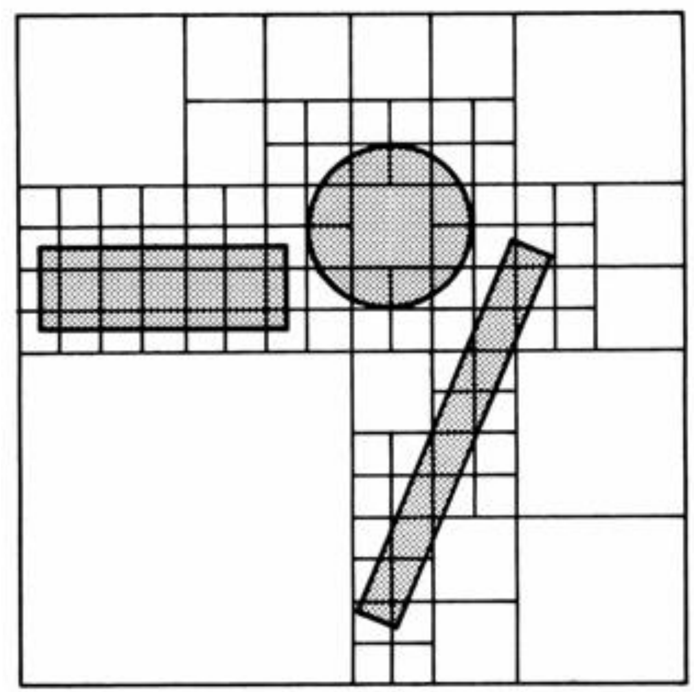


Octree Cell Subdivision Schemes

<u>Scheme 1</u>: Subdivide a cell if it is occupied by more than one object

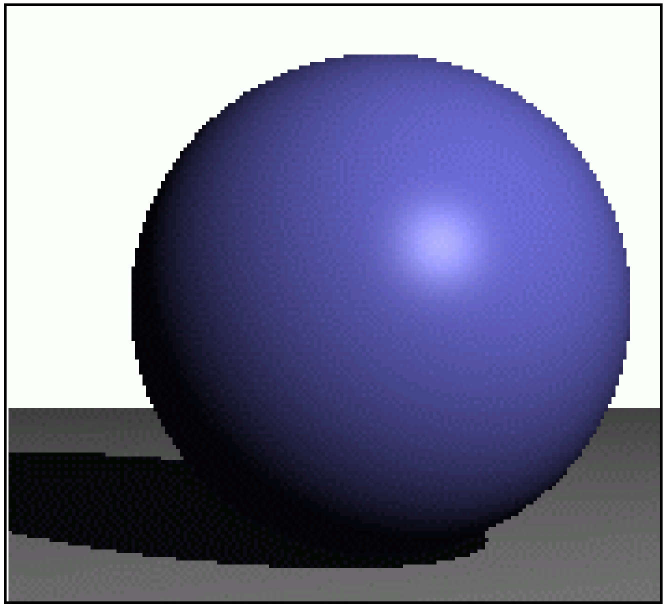


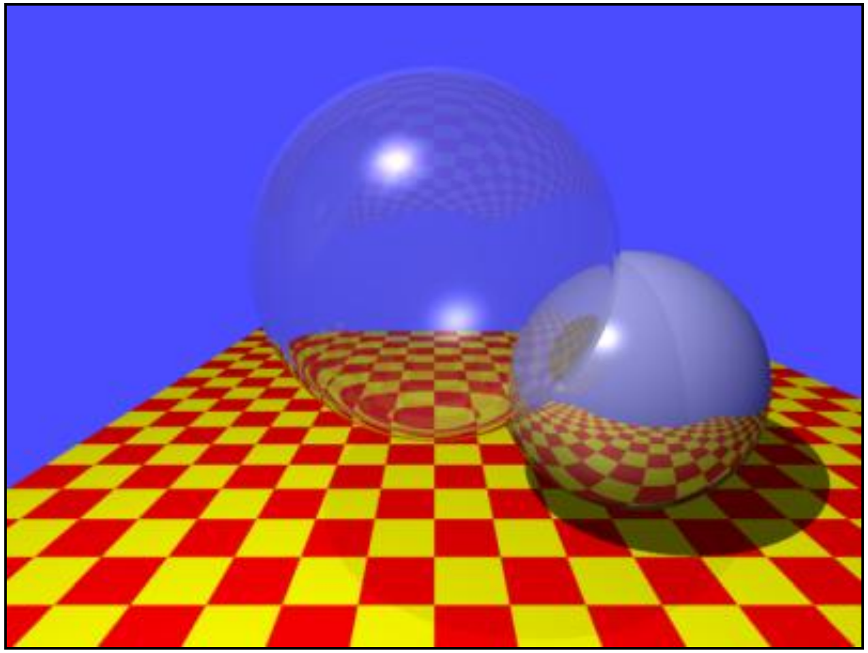
<u>Scheme 2</u>: Subdivide a cell if it is occupied by any object until the maximum allowable depth



Limitations Of Whitted Ray Tracing

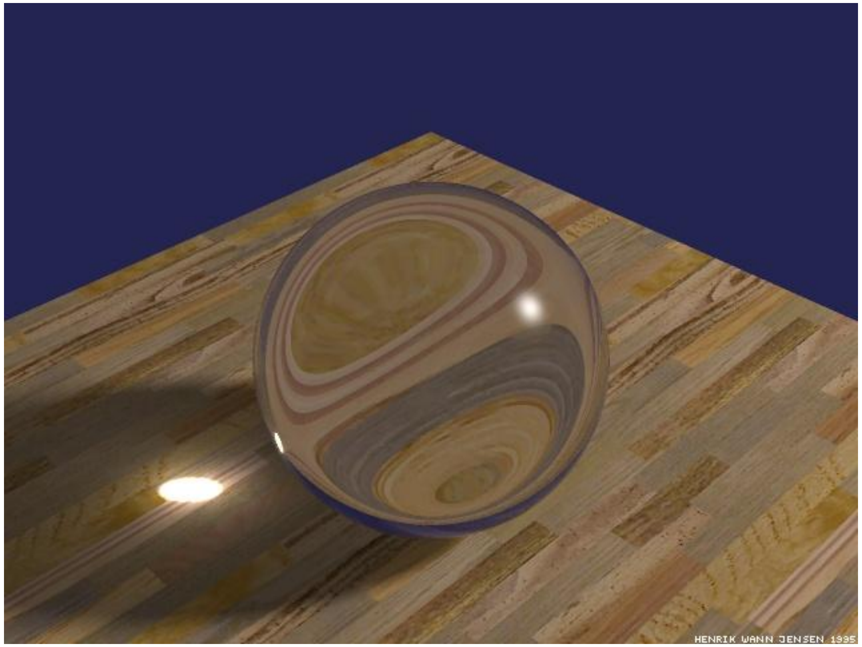
- Hard shadows
- Inconsistency between highlights and reflections
  - Sharp reflections but blurred highlights
- Aliasing (jaggies)



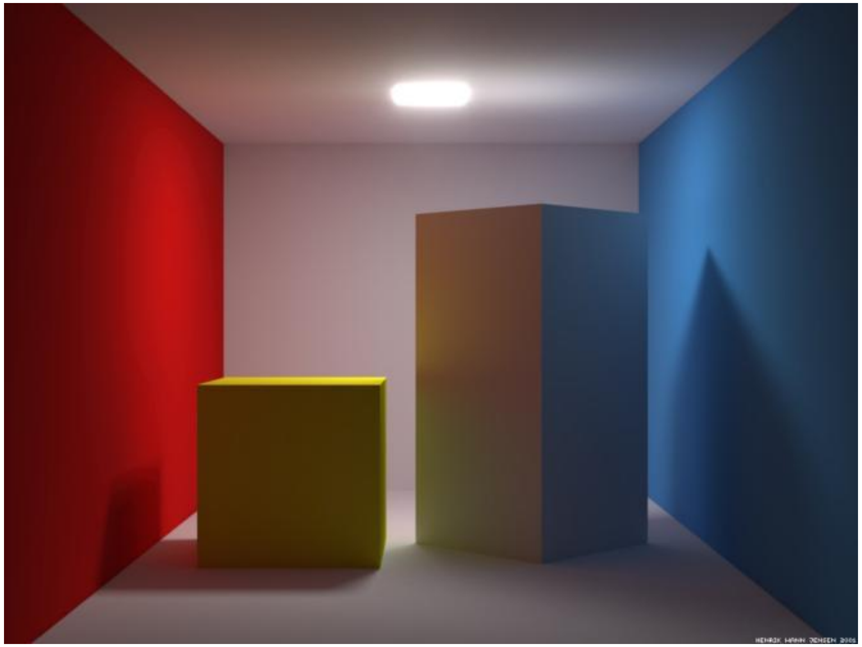


Limitations Of Whitted Ray Tracing

- Compute only a subset of light transports
  - For example, cannot simulate caustics, and color bleeding



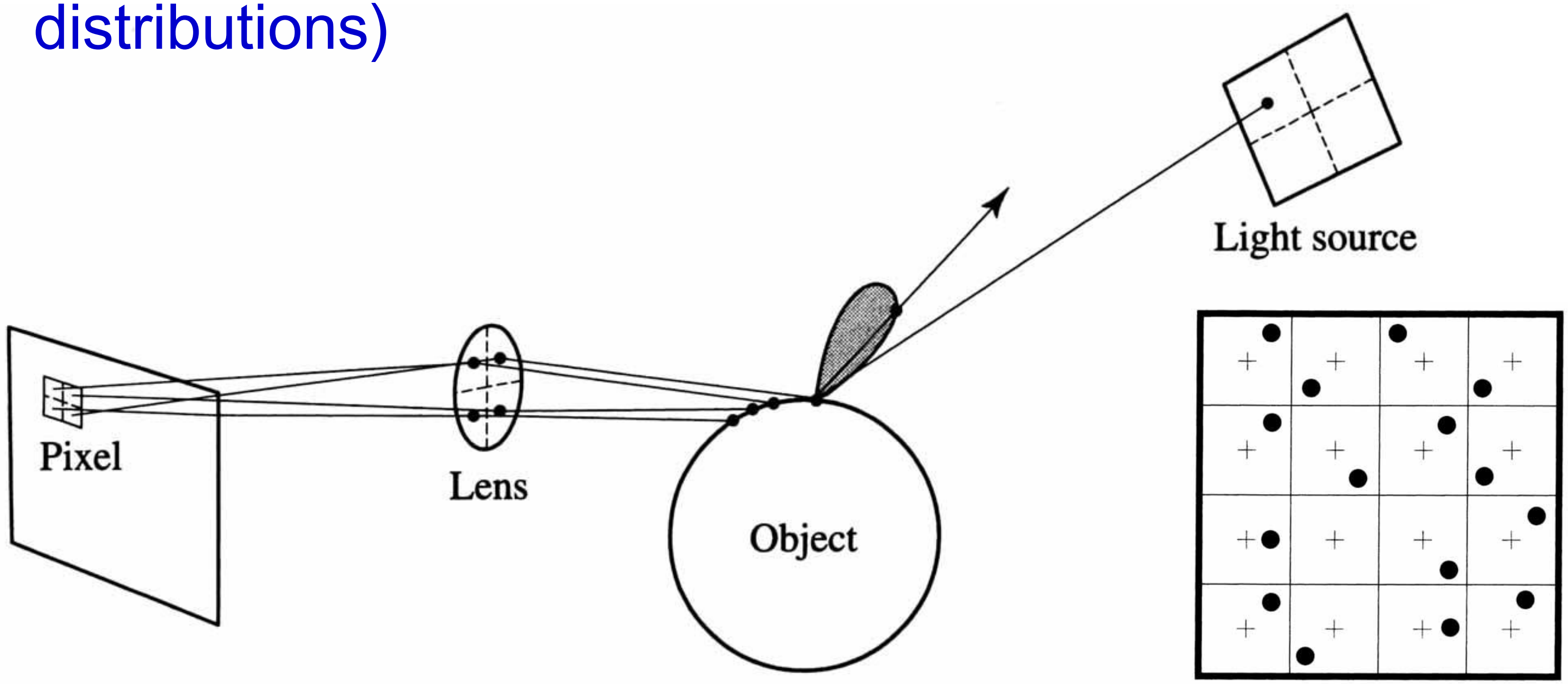
Caustics caused by focusing of light



Color bleeding caused by diffuse-to-diffuse interactions

Distributed Ray Tracing

- For each pixel, shoot multiple random rays
- At each intersection, the reflection, refraction & shadow rays are randomly perturbed (according to some distributions)



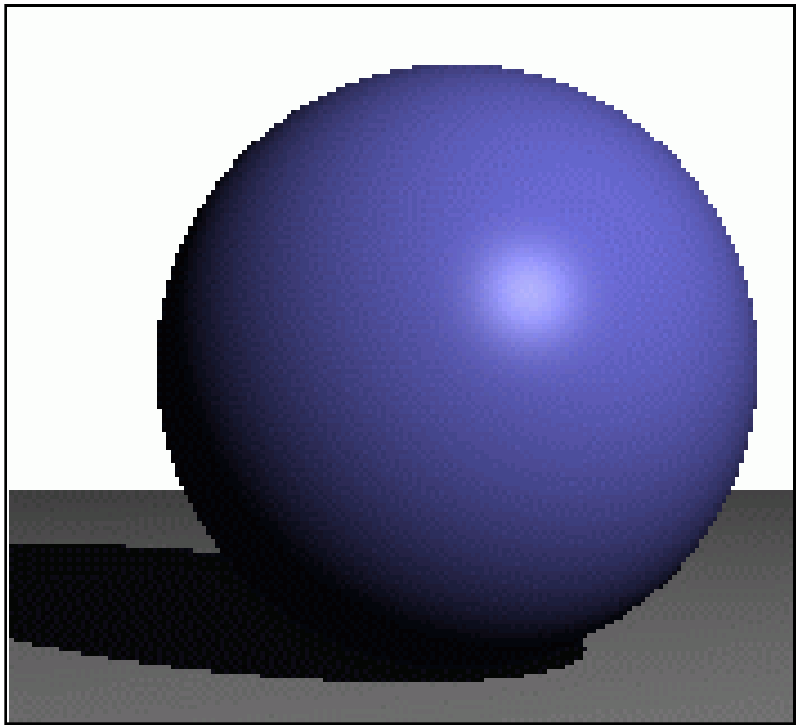
Stratified or jittered sampling

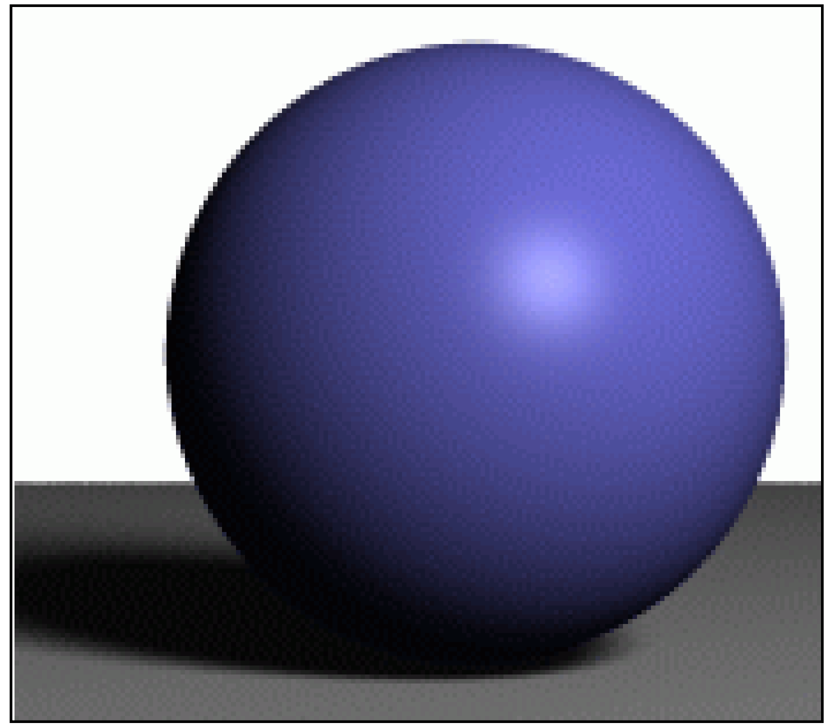
Distributed Ray Tracing

- Able to simulate the followings
  - Area lights and soft shadows
  - Blurred reflections and refractions
  - Anti-aliasing
  - Depth of field
  - Motion blur

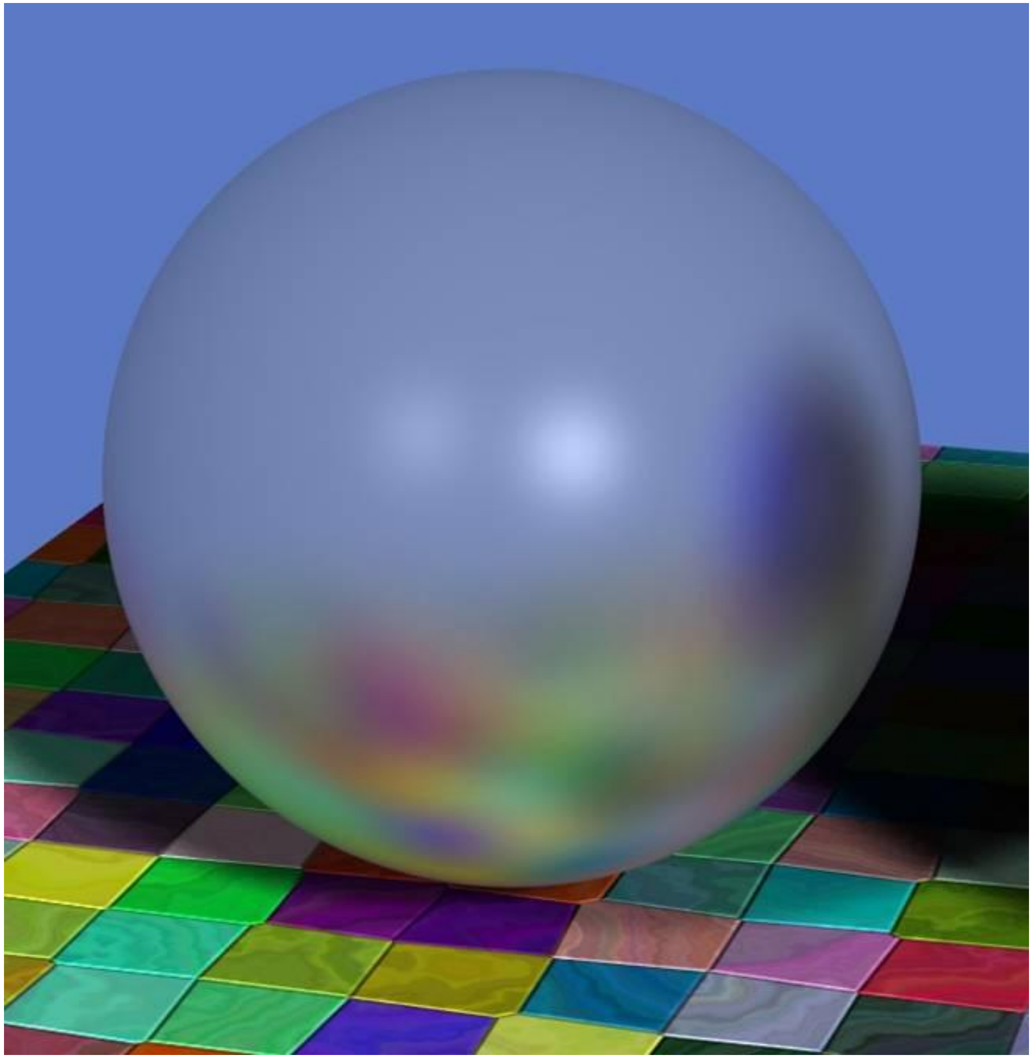
- However, it does not increase the subset of light transports simulated by Whitted ray tracing

Area Lights & Soft Shadows

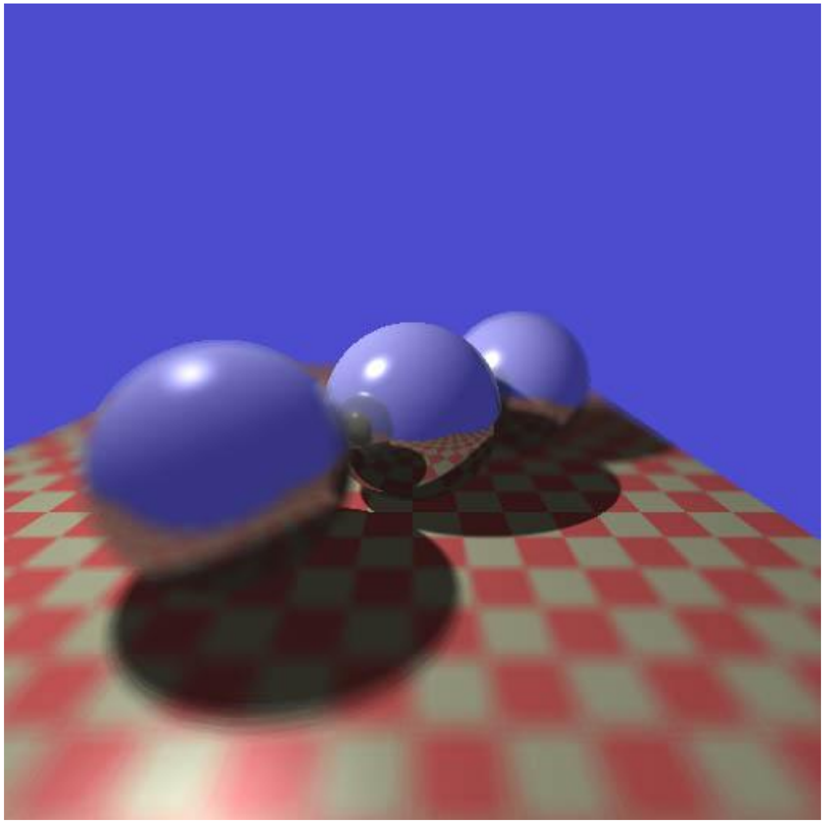




Glossy Reflections



Depth Of Field Effect & Motion Blur



End of Lecture 9