7M836
Animation & Rendering

Global illumination, ray tracing

Arjan Kok, Kees Huizing, Huub van de Wetering
h.v.d.wetering@tue.nl

Local illumination models

- What is missing in local (Phong) illumination model
  - (Shadows)
  - Real mirrors
  - Transparency
  - Area light sources
  - Indirect diffuse reflection

Light paths

- Light path notation
  - L: light source
  - D: diffuse reflection
  - S: specular reflection
  - E: eye point

- Local reflection models: L(D|S)E
- Complete solution: L(D|S)*E

Global illumination models

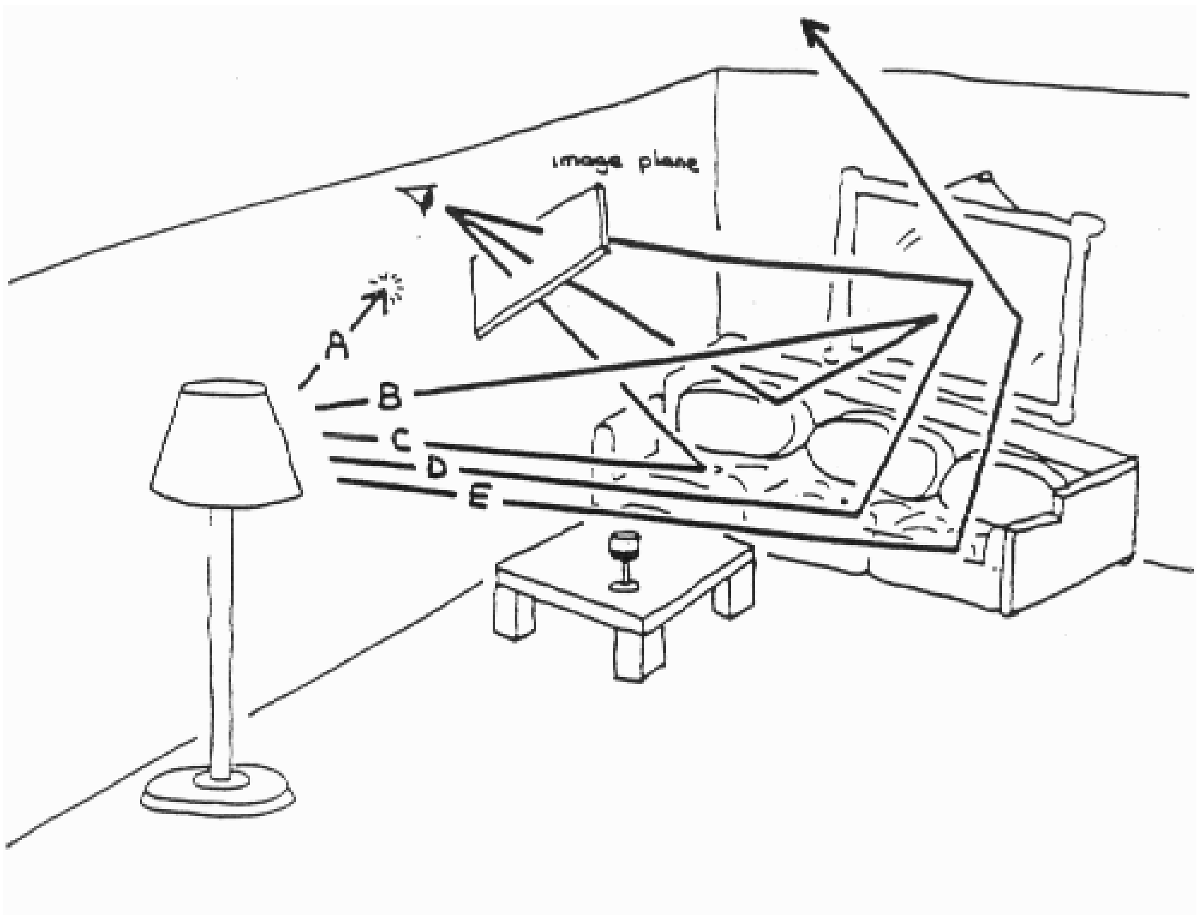
- Illumination for complete scene
- All illumination, also indirect illumination

- Several approaches:
  - Ray tracing
  - Radiosity
  - …

Brute force solutions

- Trace photons from light source into scene
- Follow paths of photons through reflections/transmissions
- At each reflection/transmission “energy” of photon is modified (part of energy absorbed by surface)
- Photons that go through image plane and reach the eye contribute to image.

Trace photons



Trace photons

- Light paths complete solution: L(D|S)*E
- Forward ray tracing

- Problem
  - Most photons will not contribute to image

Backward ray tracing

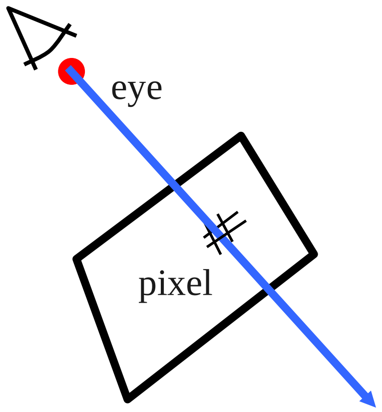
- Only trace light that arrives at viewpoint through viewing plane
- Trace light backwards from viewpoint until light source reached

- *(Backward) Ray tracing*

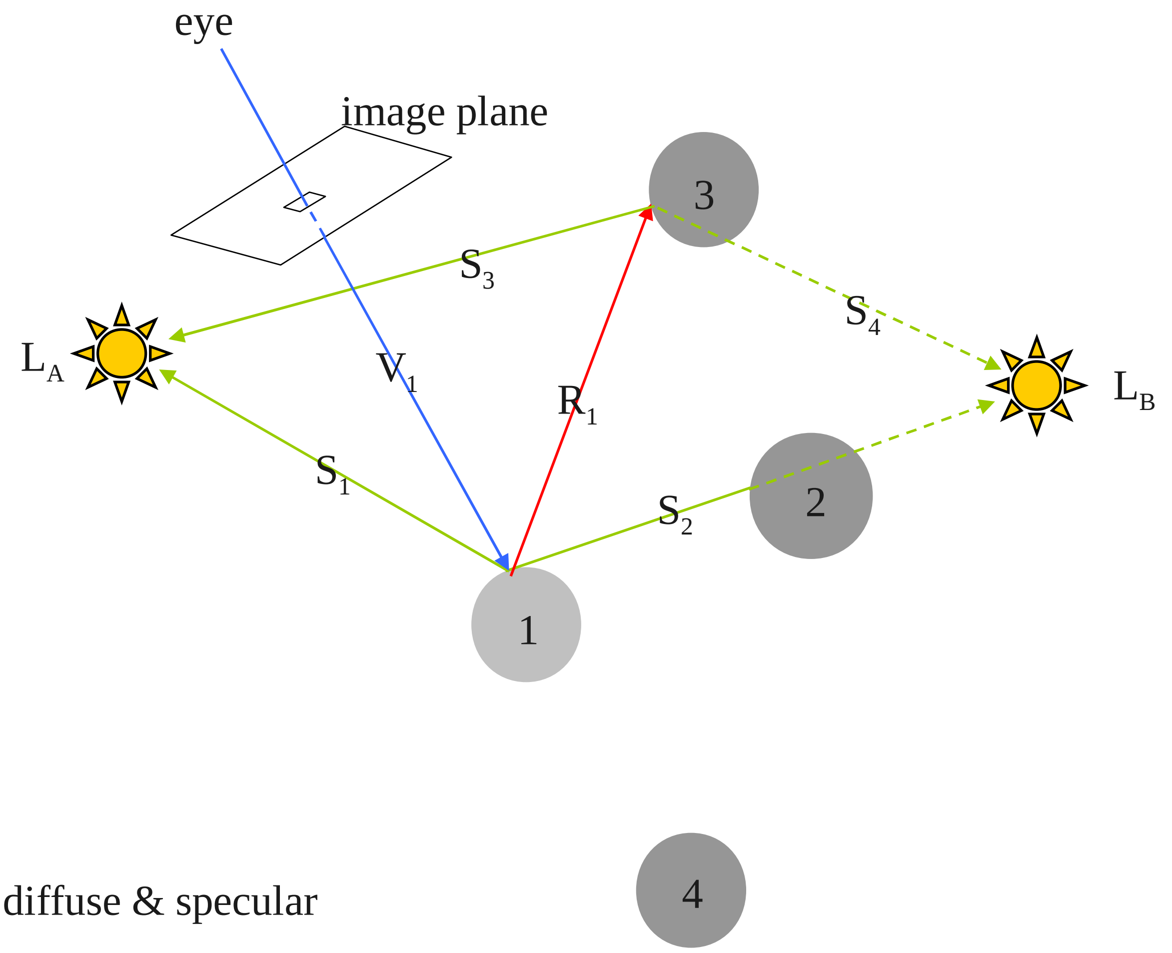
Ray tracing – basic algorithm

for all pixels in image plane

- create ray from eye point through pixel
- trace this ray on its path(s) through scene until it reaches light source(s) and collect illumination encountered during travel
- color of pixel = amount of collected light

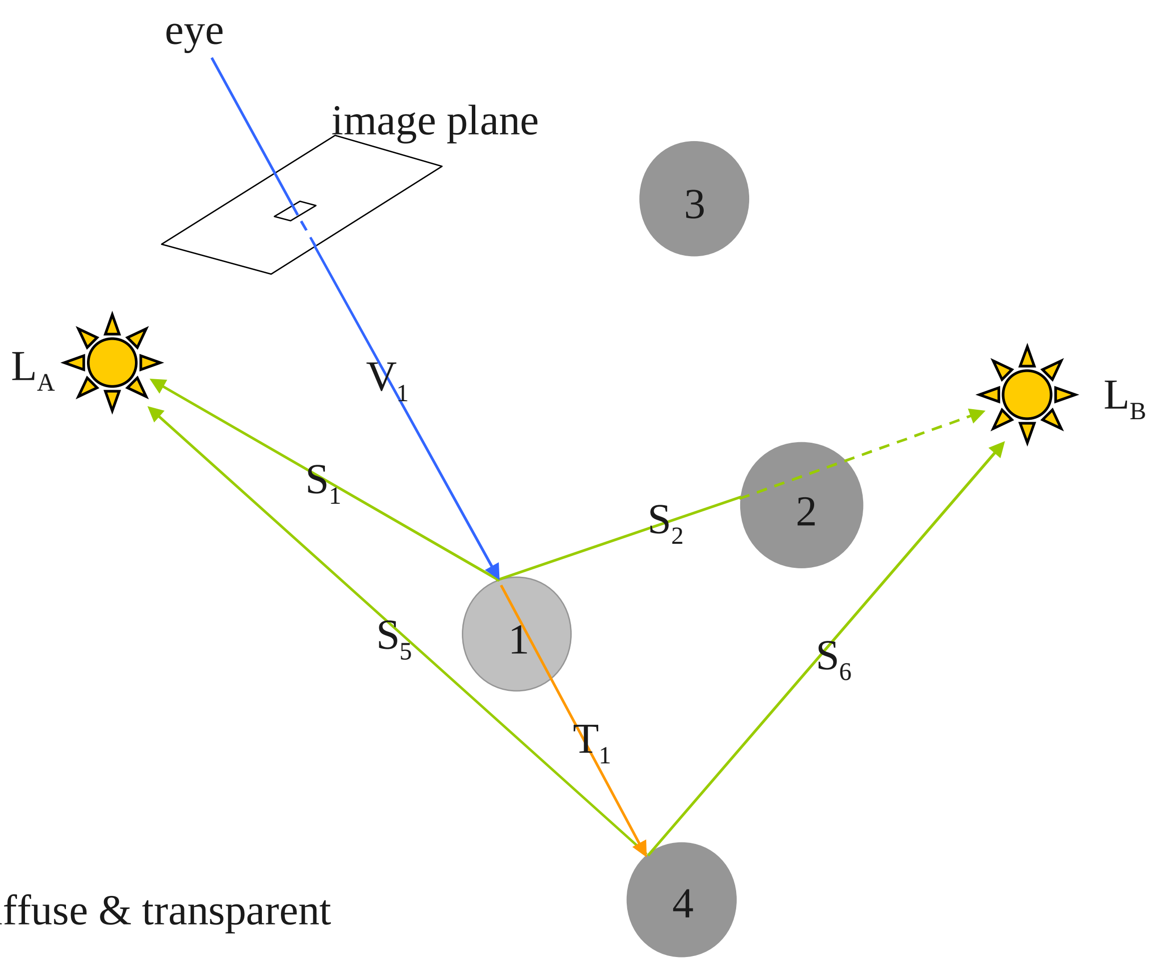


Ray tracing – example 1



Object 1 diffuse & specular

Ray tracing – example 2

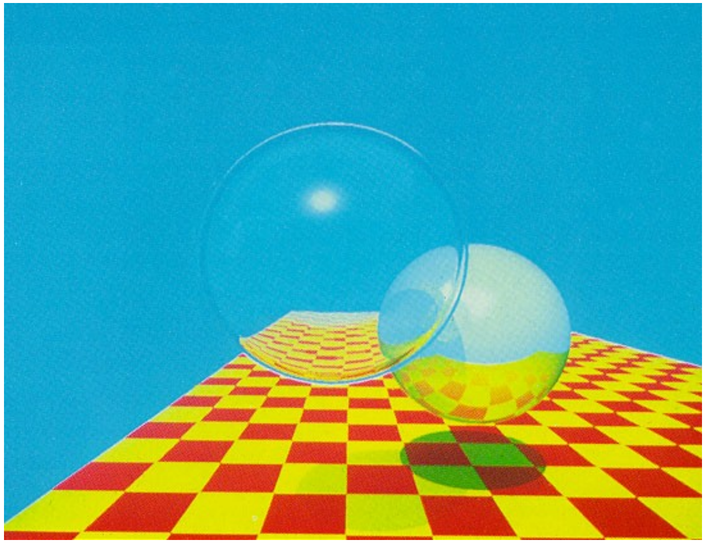


Object 1 diffuse & transparent

Ray tracing

1. Create viewing ray
2. Trace ray
3. At intersection point
   a. Compute (local) illumination. Trace shadow rays to light sources to account for shadows.
   b. If surface at intersection is specularly reflecting, trace reflection ray and compute its contribution: **continue at step 2**
   c. If surface at intersection is transparent, trace transparency ray and compute its contribution: **continue at step 2**
   d. Sum contribution of steps (a), (b) and (c)

Early ray-tracing picture



640x480, 74 minutes on VAX-11/780

An Improved Illumination Model for Shaded Display,
Turner Whitted, Communications of the ACM,
June 1980, volume 23, Number 6

Example using ray tracing

Computations in ray-tracing

- Creation of viewing ray
- Intersection computation
  - Find first intersected object (ray-scene intersection)
  - Point of intersection
  - Normal (and local coordinates) of intersection point
- Creation of shadow rays
- Creation of reflection rays
- Creation of transparency rays
- Illumination

Creation of viewing ray

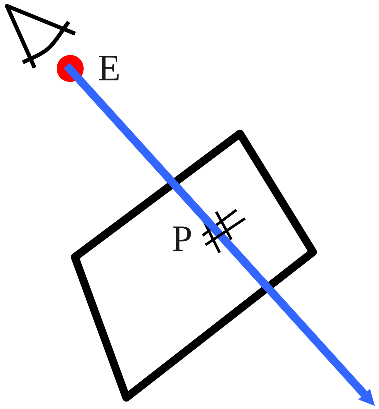
Ray is a half line that

start in the eye point (E) and

passes through a pixel (P) in the projection screen

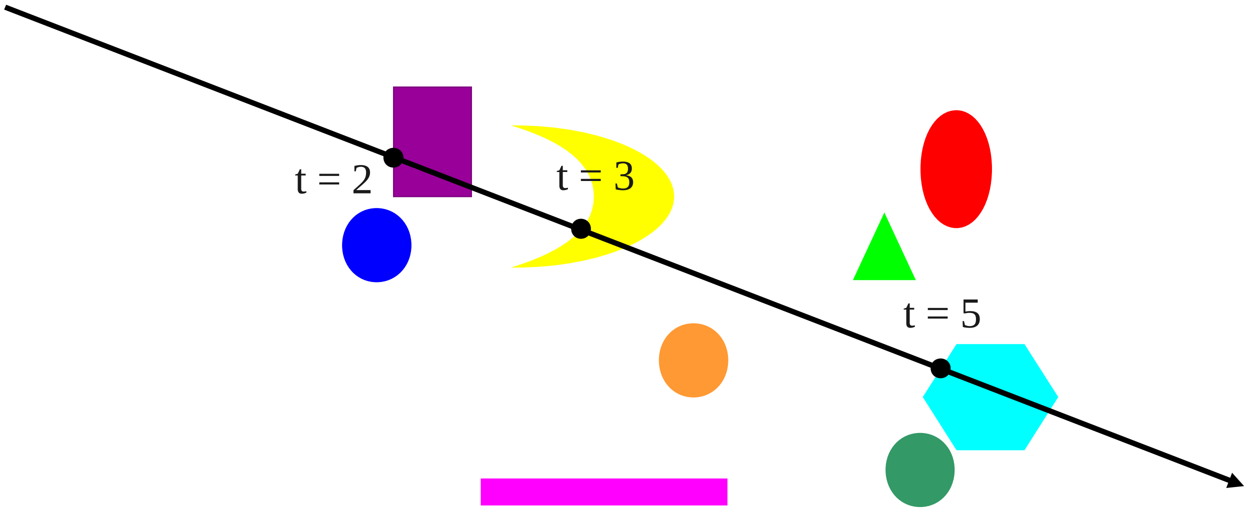
$$\overbrace{\vec{R}(t) = \vec{E} + t\vec{V}}^{\text{viewing ray}} \wedge \ t > 0$$

$$\vec{V} = \frac{\vec{P} - \vec{E}}{\left\| \vec{P} - \vec{E} \right\|}$$



Ray-scene intersection

- Compute intersection of ray R(t) with all objects in scene
- Intersection with smallest positive intersection distance t gives intersected object



Ray-object intersection

- Ray equation
  - R_o : ray origin
  - R_d : ray direction
  - $||R_d|| = 1$

$$\overbrace{\vec{R}(t)=\vec{R}_o+t\,\vec{R}_d}^{\text{general ray}} \quad \wedge \quad t>0$$

- Object description examples:
  - Implicit surface S :
  - Sphere
  - Polygons
  - …

$$S=\left\{\vec{P}\in\mathbb{R}^3 : f(\vec{P})=0\right\}$$

Ray-sphere intersection

- Sphere with radius r and center C:

  $||P - C||^2 - r^2 = 0$

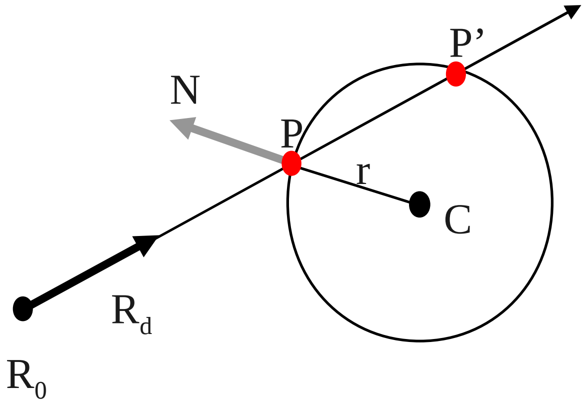
- Substitution of R(t) for P gives

  $t^2 + 2bt + c = 0$

  - $b = R_d \bullet (R_o - C)$
  - $c = ||R_o - C||^2 - r^2$

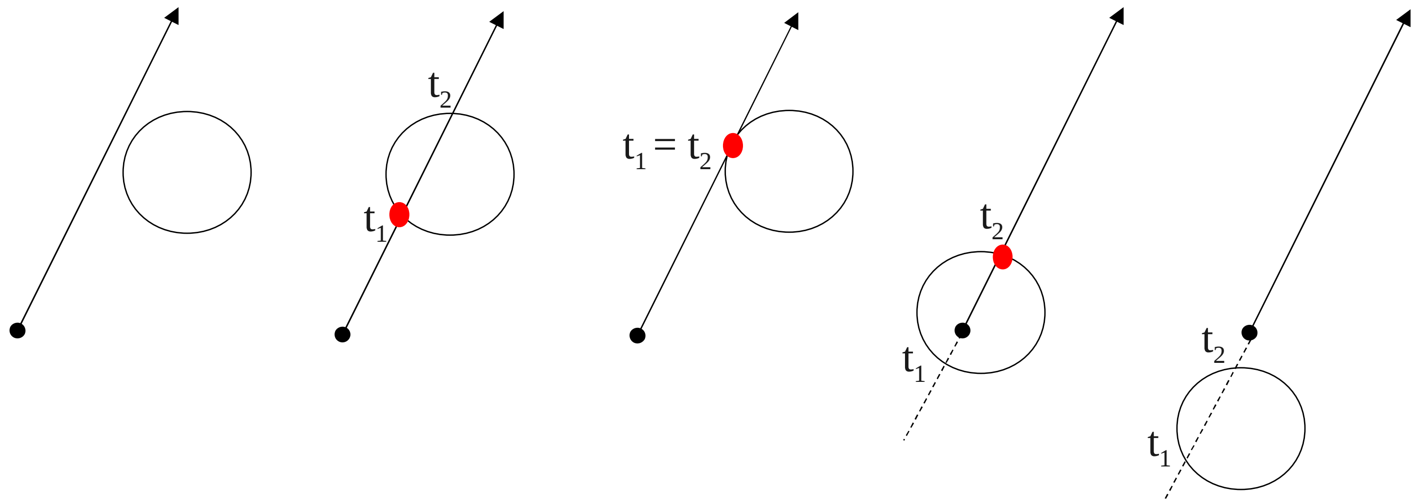
- Solution for t gives intersection points

  $t_{1,2} = -b \pm \sqrt{b^2 - c}$



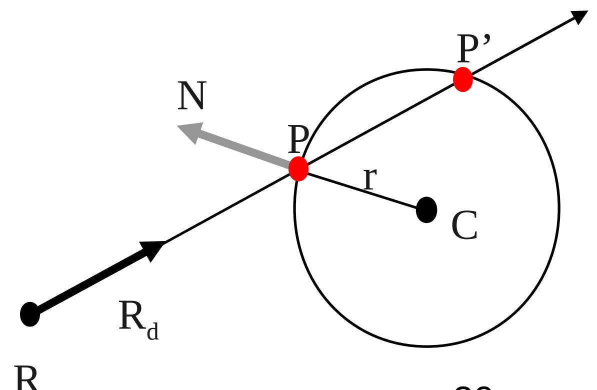
Ray-sphere intersection

- Solutions for t_1 and t_2:



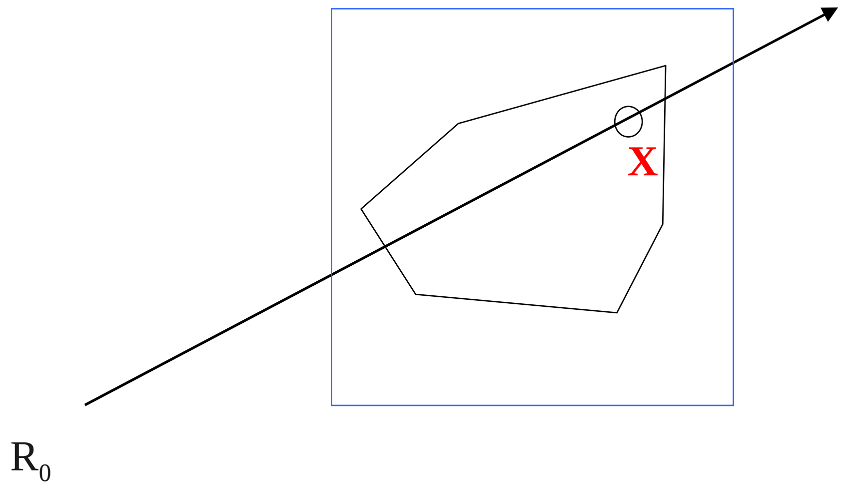
- Normal at intersection point P

  $N = (P – C) / r$



Ray-polygon intersection

- Two steps
  1. Intersect ray with plane of polygon. Compute intersection point X
  2. Determine if X is in the polygon
- Normal at intersection point is plane normal



Ray-polygon intersection

Intersection ray-plane:

Implicit equation of plane with normal N and point P using dot product:

$$\vec{N} \cdot (\vec{X} - \vec{P}) = 0$$

Substitution of X = R(t):

$$t = \frac{\vec{N} \cdot (\vec{P} - \vec{R}_o)}{\vec{N} \cdot \vec{R}_d}$$

Ray-polygon intersection

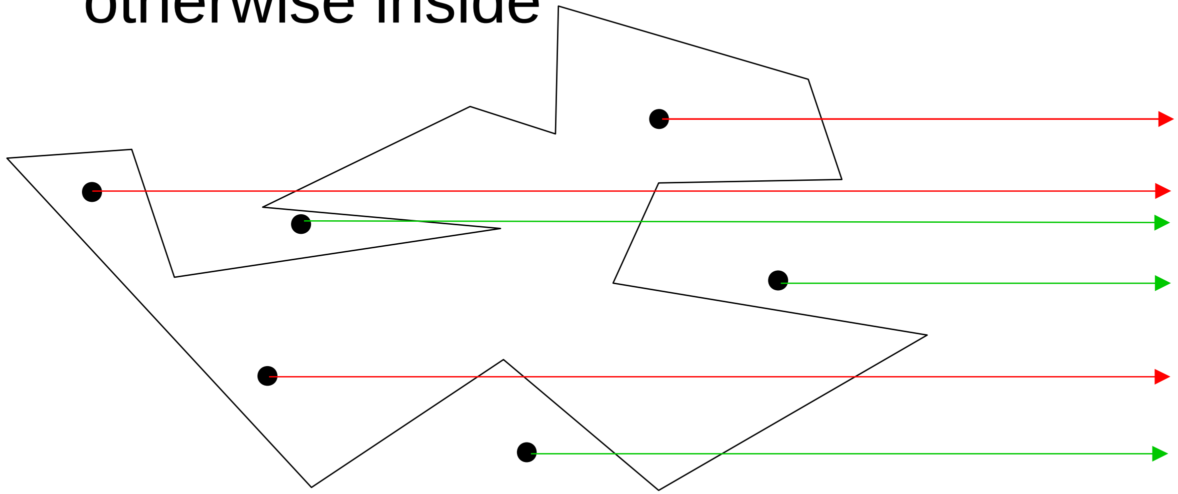
Determine if intersection point P is in a polygon with a

point in polygon test:

* Draw half line L starting at point P

* Count intersections of L with the edges of the polygon

If count even, than point outside polygon,
otherwise inside

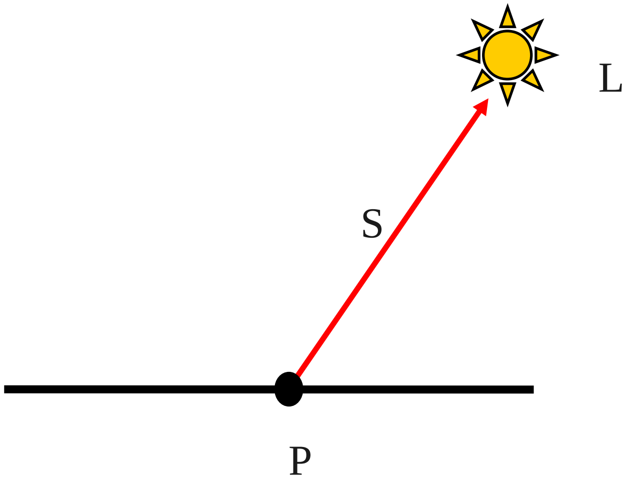


Shadow ray

- Determine if point is illuminated by light source

$$\overbrace{\vec{R}(t)=\vec{P}+t\vec{S}}^{\text{shadow ray}} \;\wedge\; t>0$$

$$\vec{S}=\frac{\vec{L}-\vec{P}}{\|\vec{L}-\vec{P}\|}$$



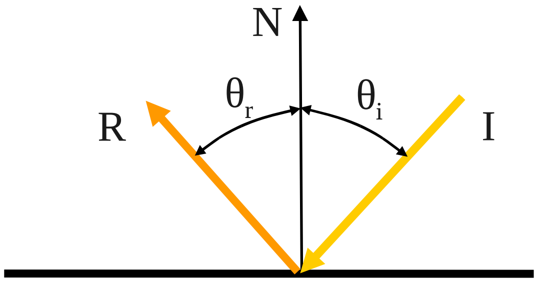
- Point P is illuminated by light source L if there is no intersection of shadow ray with any object in the scene for 0 < t < ||L – P||

Reflection ray

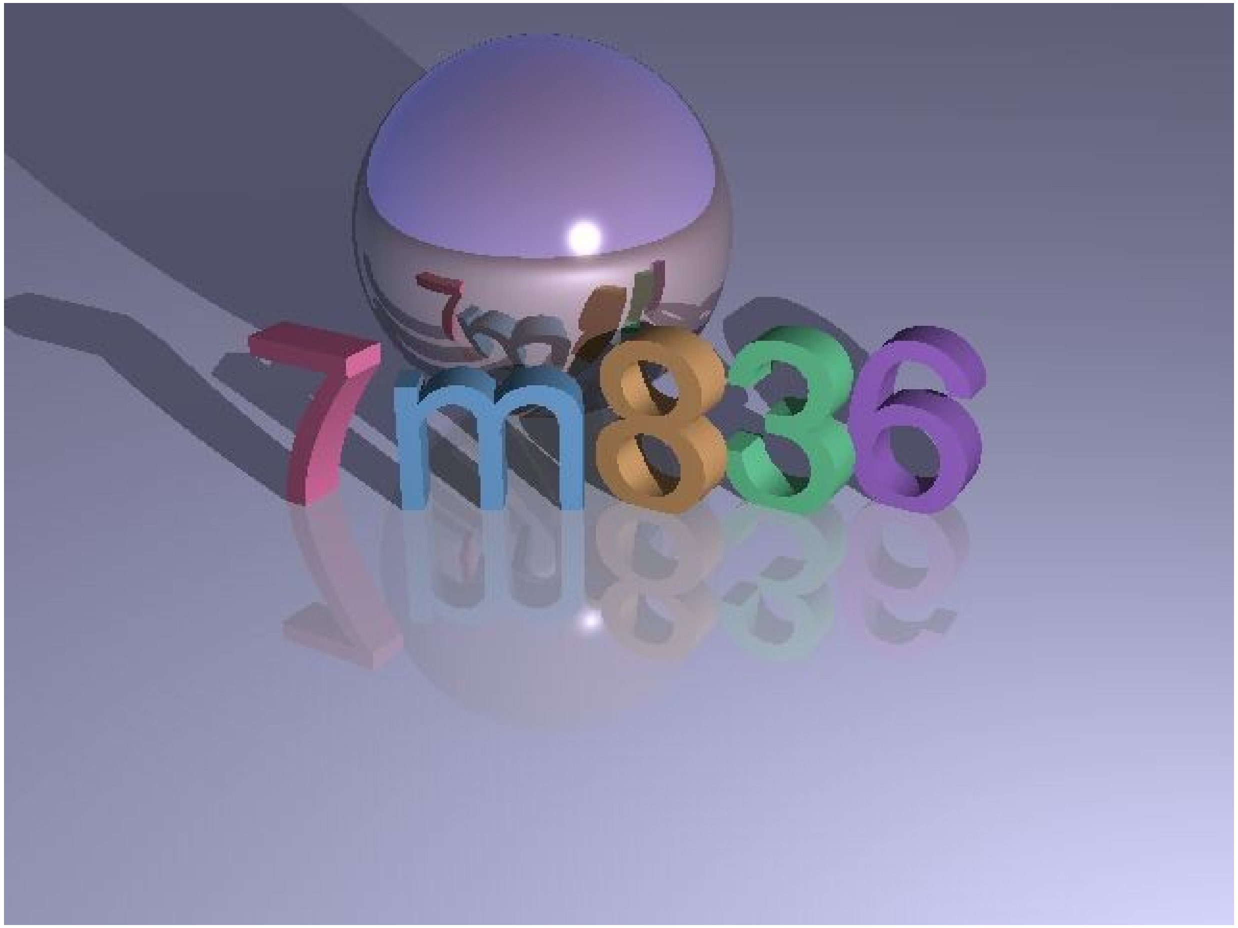
- Physical laws:
  - R, N, and I are in same plane: $\vec{R} = \alpha \vec{I} + \beta \vec{N}$
  - Angle of incidence = angle of reflection: $\theta_r = \theta_i$

- Reflection ray direction

$$\vec{R} - \vec{I} = 2\cos(\theta_i)\,\vec{N}$$
$$\vec{R} = \vec{I} + 2\,\vec{N}\cdot\vec{I}\,\vec{N}$$

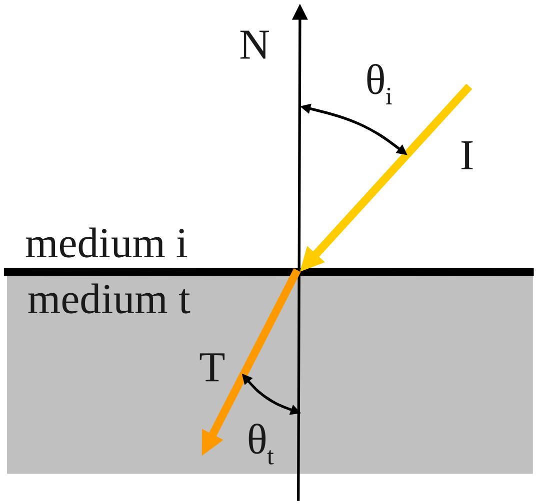


Reflection



Transparency ray

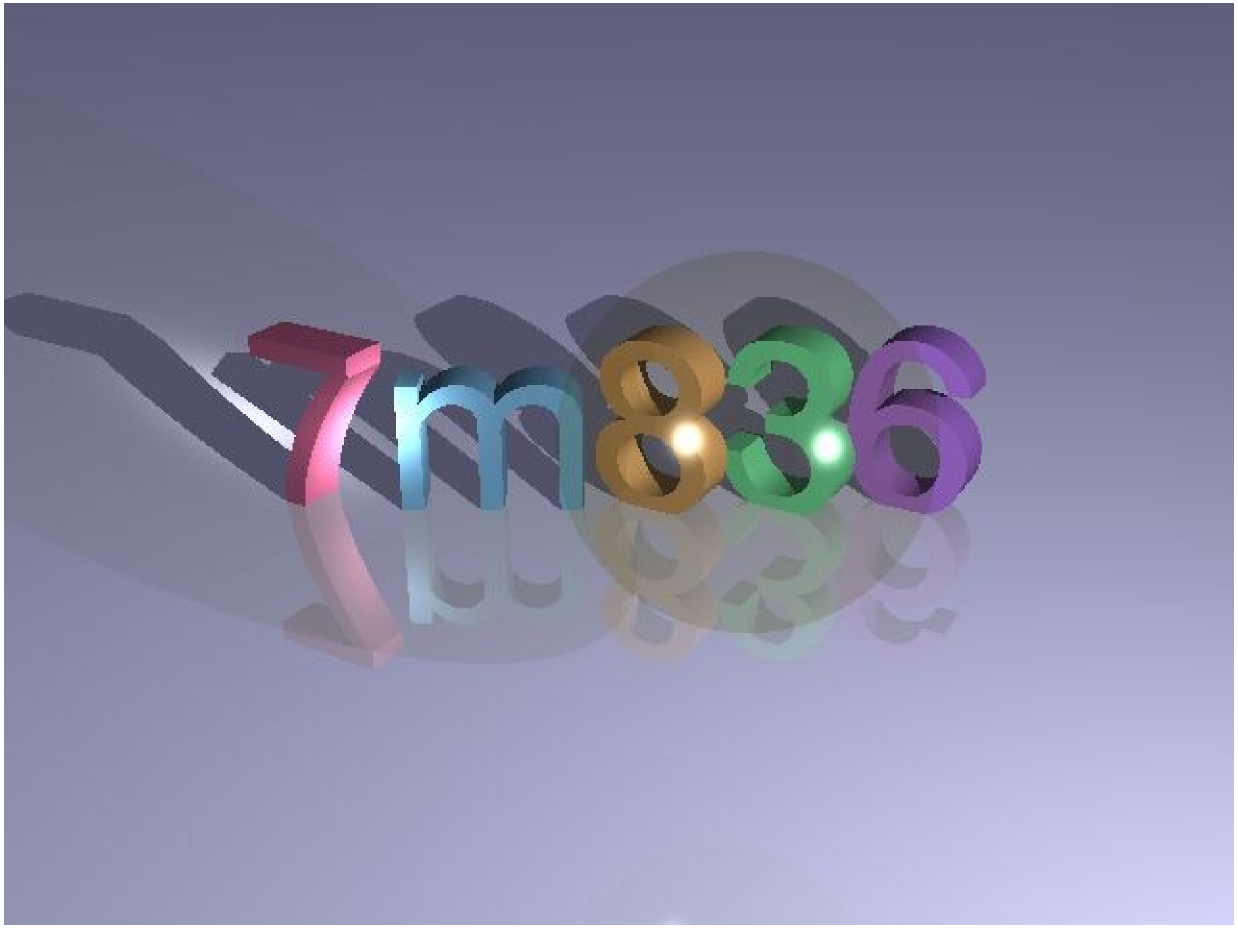
- Snell's law: $$\frac{\sin(\theta_1)}{\sin(\theta_2)} = \eta_{21} = \frac{\eta_2}{\eta_1}$$
  - η_i = index of refraction medium *i* with respect to vacuum
  - η_{it} = index of refraction medium *I* with respect to medium *t*



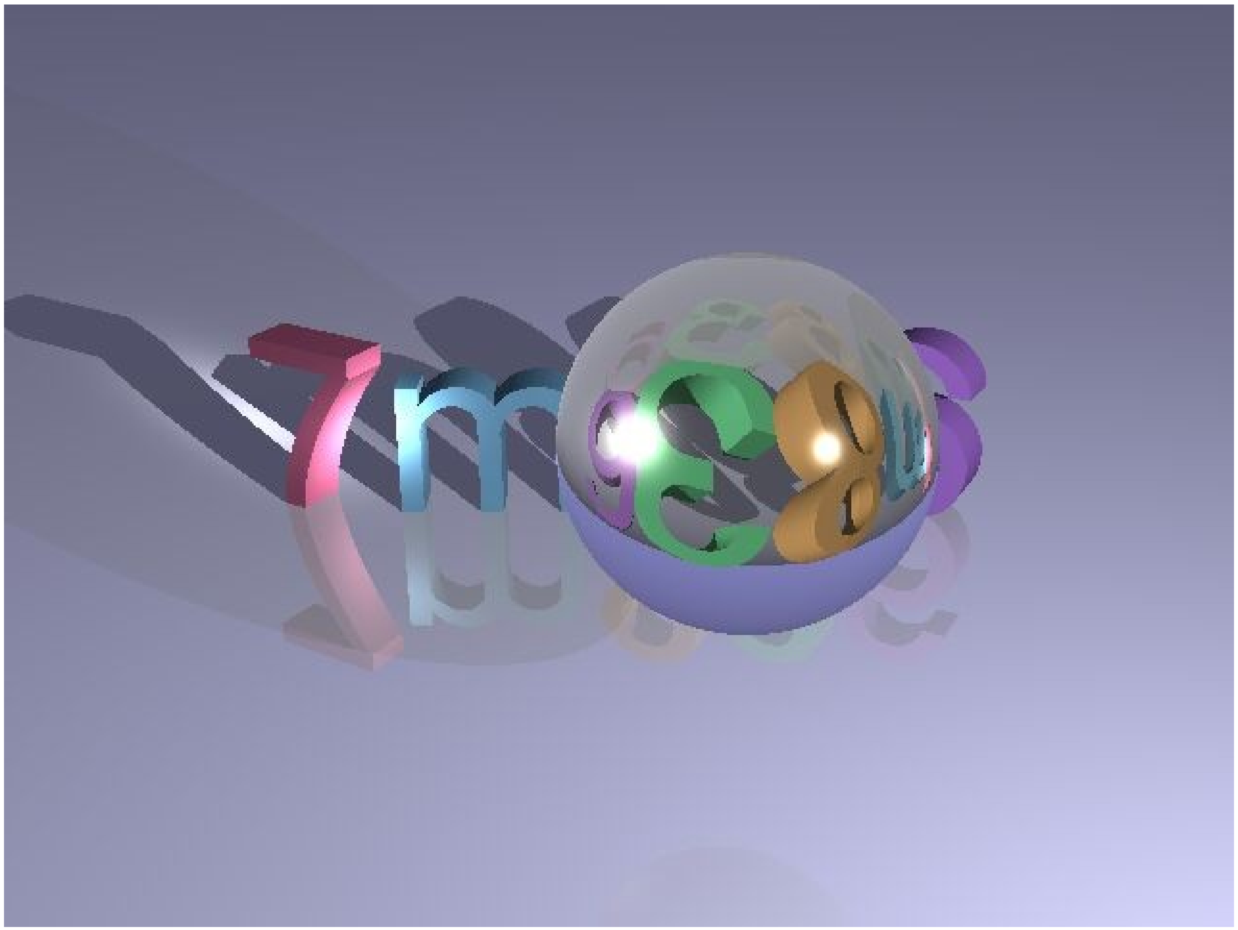
- Transparency ray direction

$$T = \eta_{it} I + (\eta_{it} \cos(\theta_i) - \sqrt{1 + \eta_{it}^2 (\cos^2(\theta_i) - 1)}) N$$

Transparency



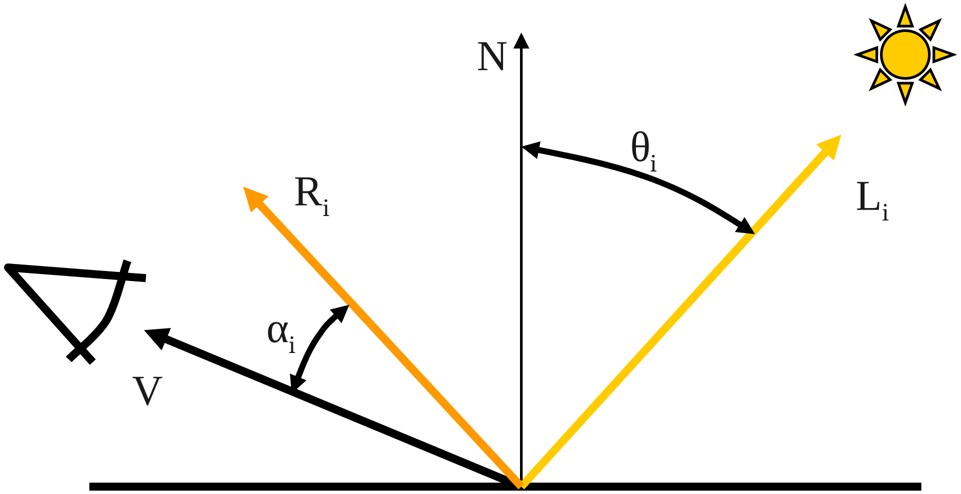
Transparency



Illumination

- Phong illumination model for *local illumination*

$$I = I_e + k_a C_a I_a + \sum_i I_i \left(k_d C_d \cos(\theta_i) + k_s C_s \cos^n(\alpha_i) \right)$$



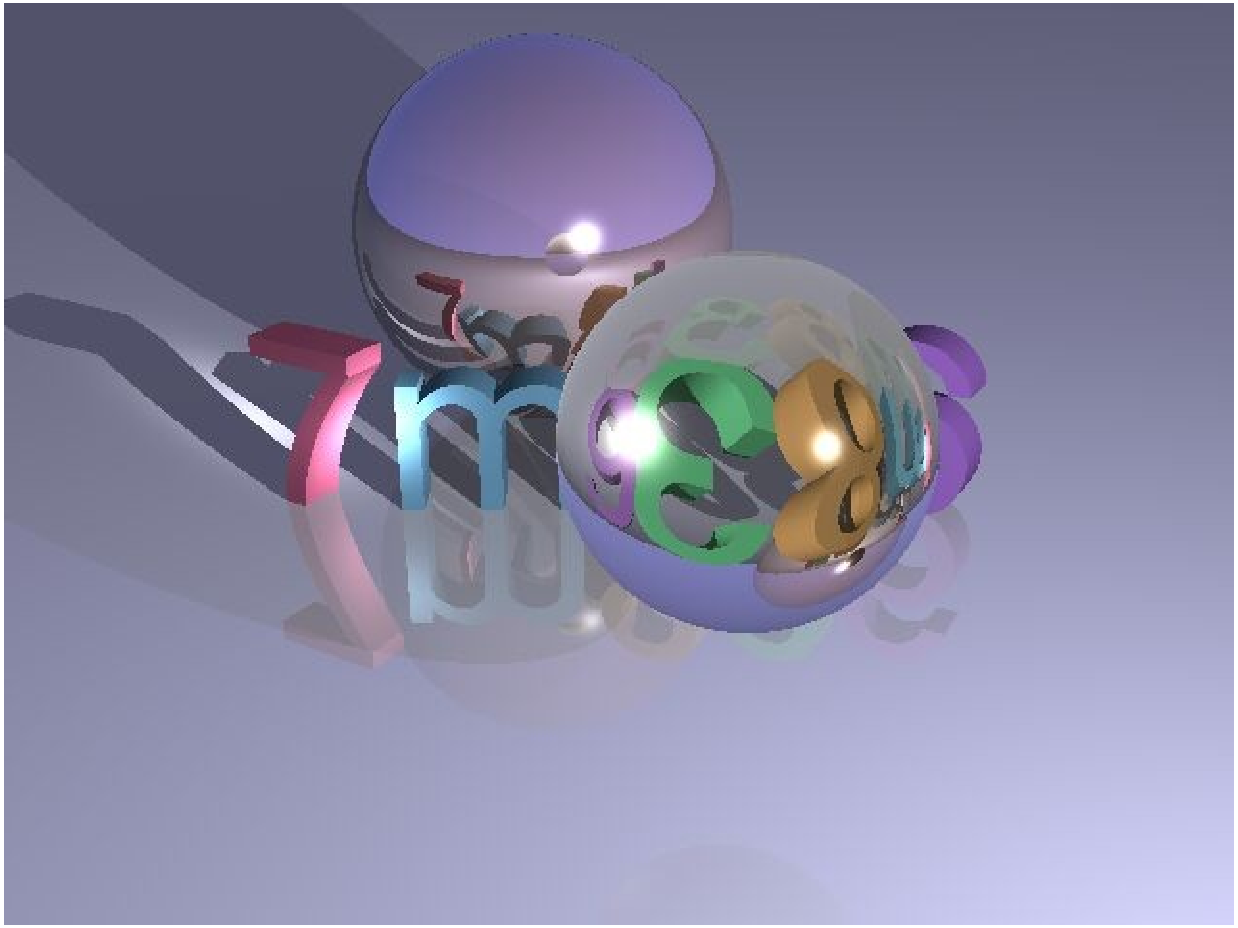
Illumination for ray tracing

- Extension of local model with shadow information, mirroring and transparency

$$I = I_e + k_a C_a I_a + \sum_i S_i I_i \left(k_d C_d \cos(\theta_i) + k_s C_s \cos^n(\alpha_i) \right) + k_s C_s I_R + k_t C_t I_T$$

- S_i = shadow factor (0, 1)
- I_R = intensity of reflection ray
- I_T = intensity of transmission ray

Ray tracing



Ray tracing

- Light paths ray tracing: LS*E and LDS*E
- Allows for transparency with refraction
- Easy shadow computation
- Easy to program

- Inefficient

Ray tracing

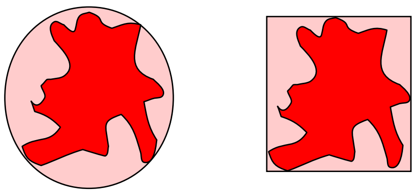
Ray tracing - efficiency

- Ray tracing is computationally expensive
- Need for efficient computation methods

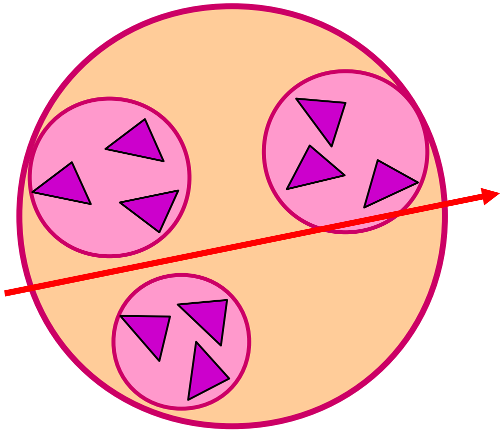
- Efficiency ray tracing determined by
  - Number of rays
    - Number of pixels
    - Number of light sources
    - Number of specular and transparent objects
    - Recursion depth
  - Efficiency ray-object intersections
  - Number of intersections to be computed

Ray tracing – bounding volume

- Reduce number of complex ray-object intersection computations by providing complex objects with a simple geometry around complex geometry
  - Sphere
  - Cube

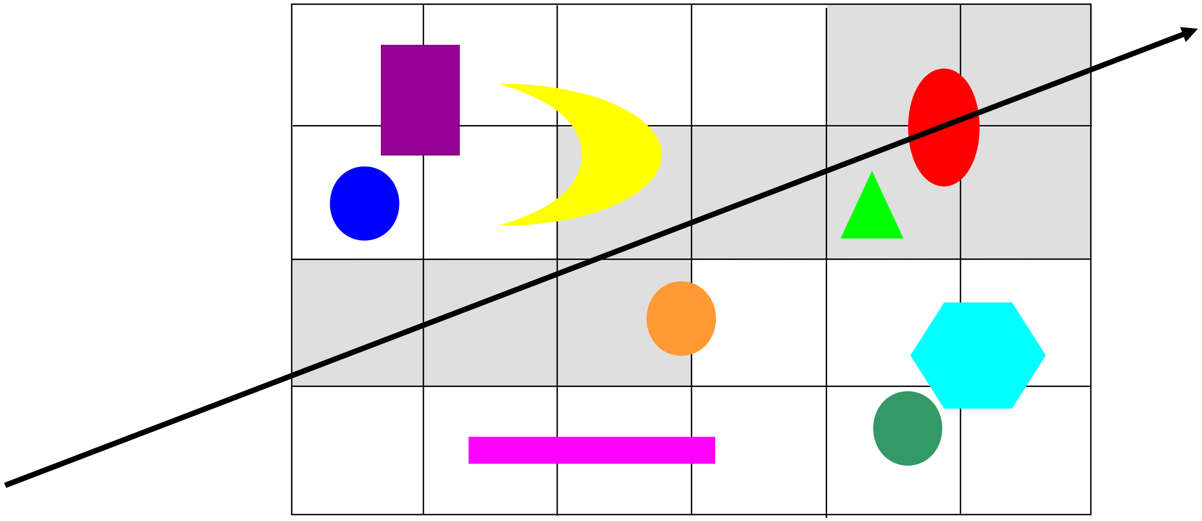


- Do intersection with simple geometry
  - Only if intersection found, do intersection with complex geometry
- Hierarchy of bounding volumes



Ray tracing – space subdivision

- Partition scene into small cells
- Store in each cell pointers to objects it contains
- Trace ray through cells and only compute intersections with objects in visited cell
- When intersection found, stop tracing rays through cells

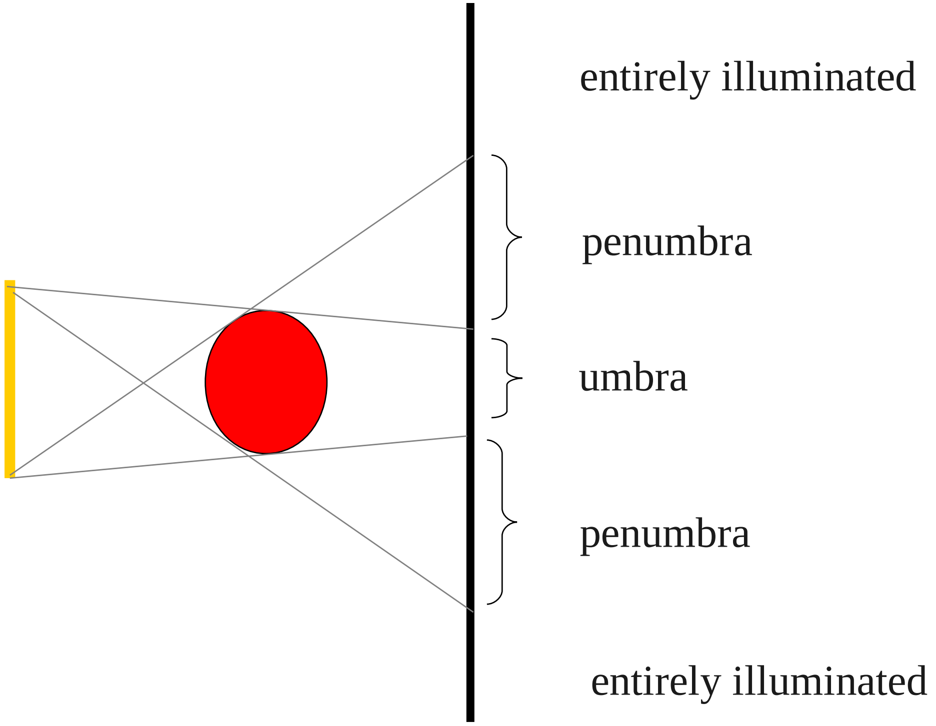


Ray tracing extensions

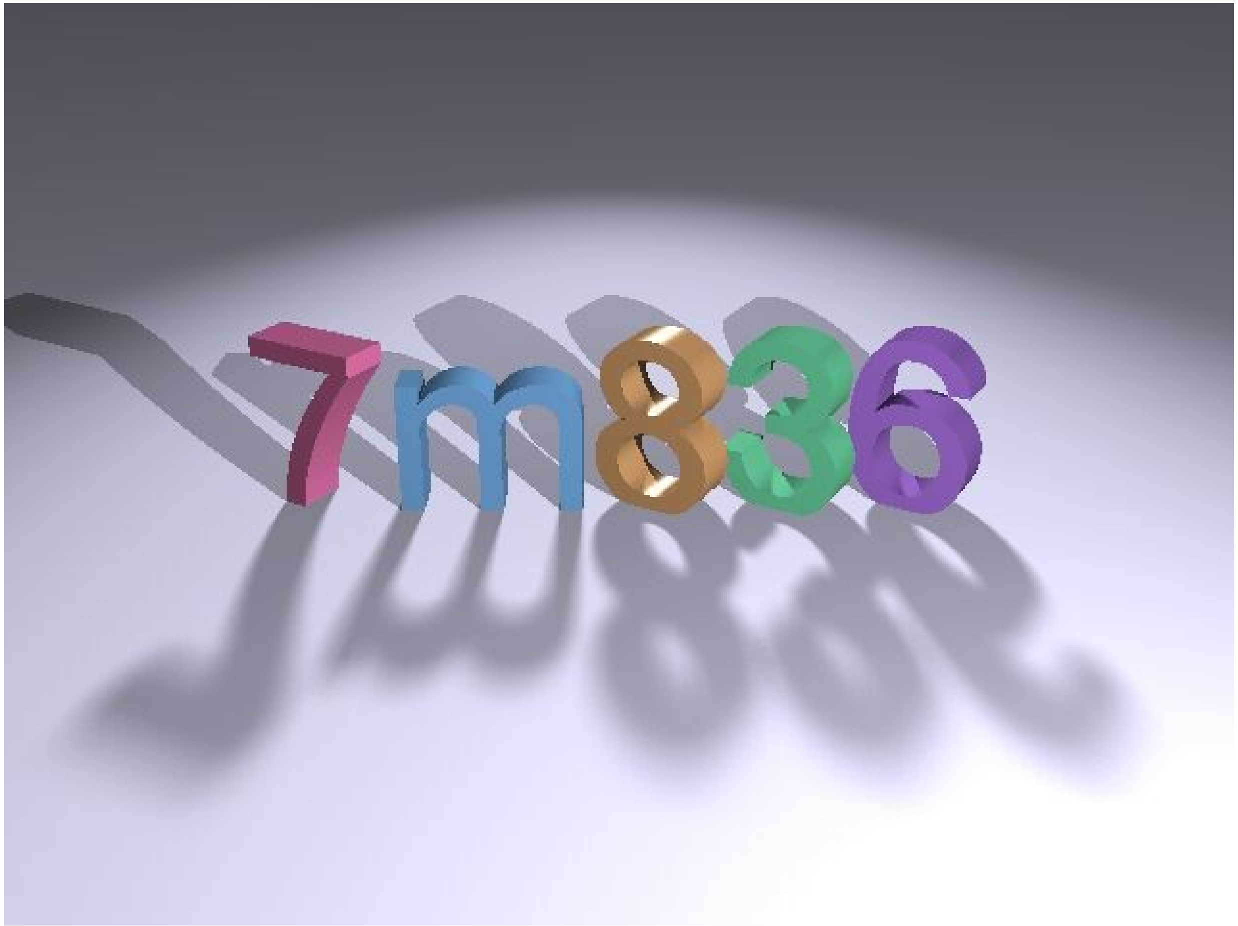
- More effects are possible:
  - Area light sources, soft shadows
  - Depth of field
  - Motion blur
  - ...

Area light sources

- Area light sources generate soft shadows (penumbrae)

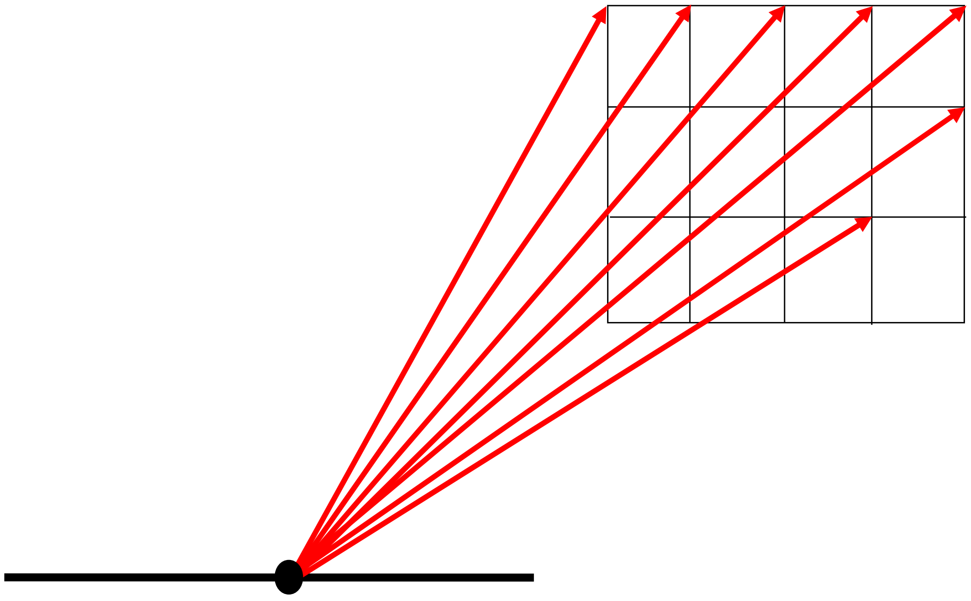


Area light source



Area light source

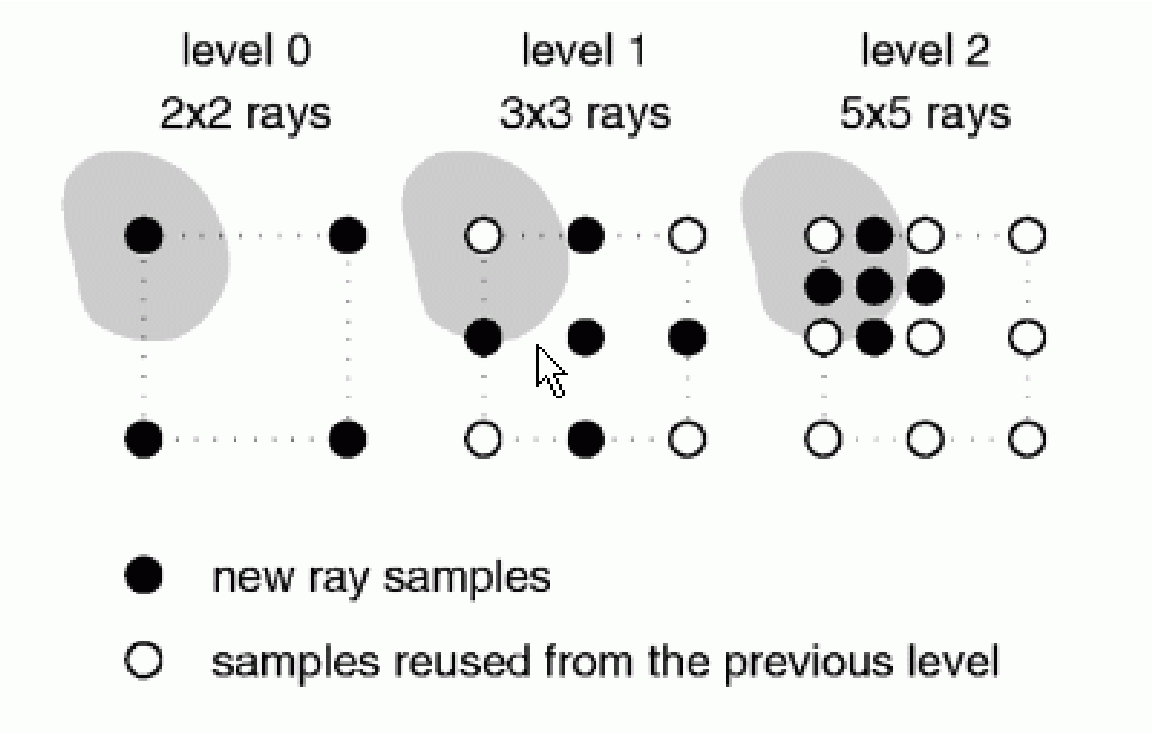
- Subdivide area light source in number of point light sources
- Use regular grid of points



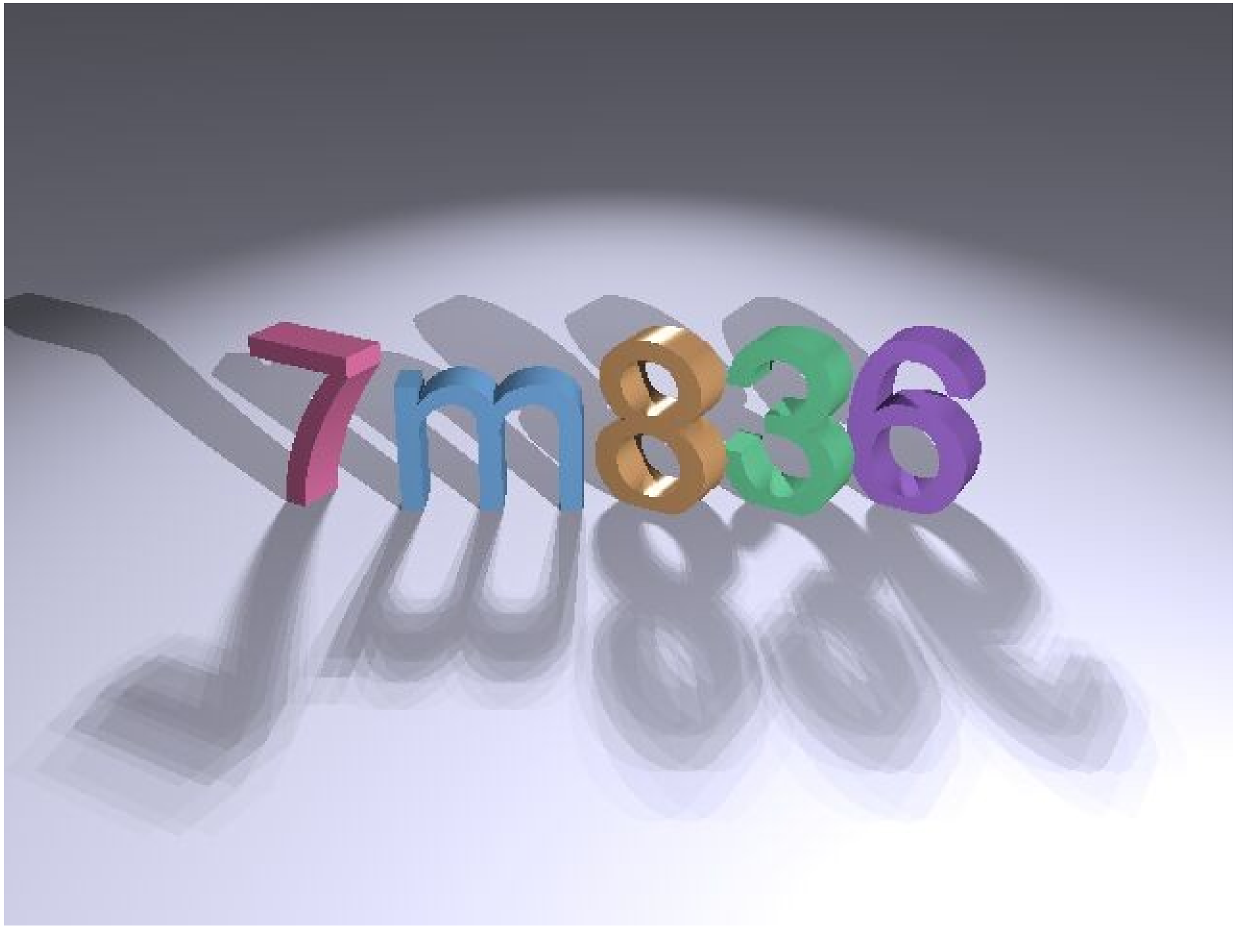
Area light source

- How many point light sources must be used to simulate area source?
  - Number depends on distance from point to light source
- Regular pattern of point sources generates shadow bands

Area source: adaptive subdivision

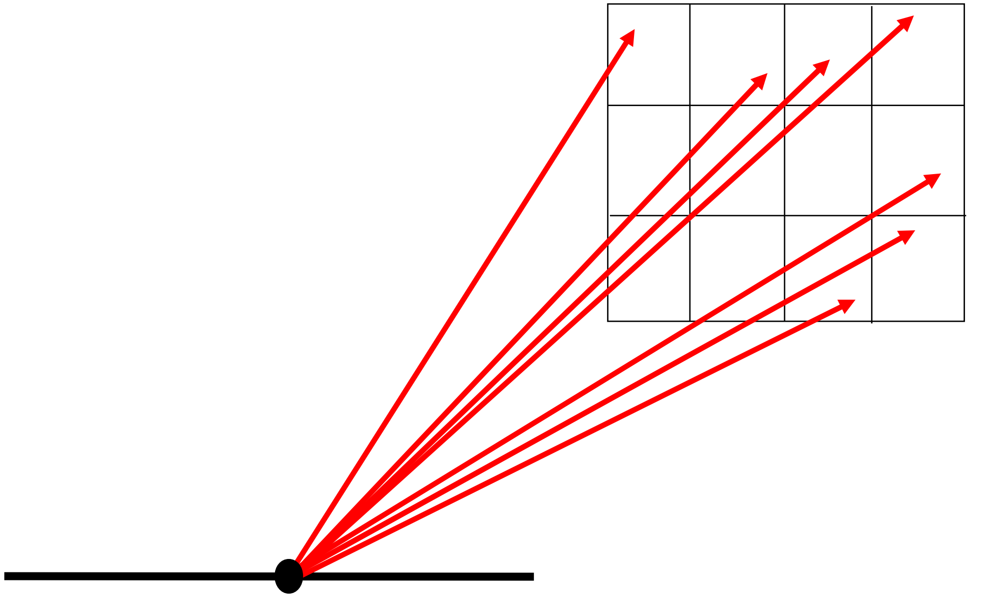


Shadow bands

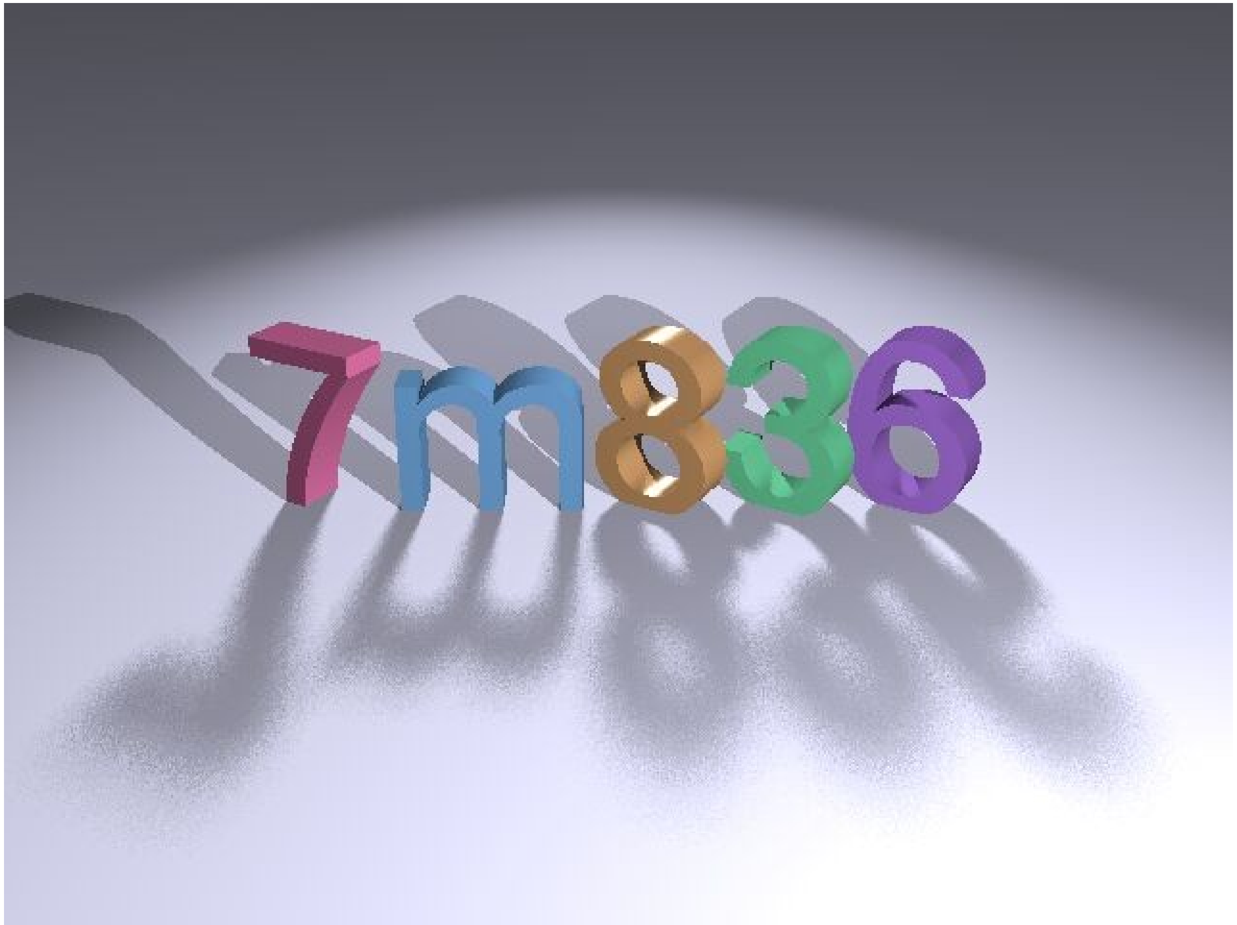


Area source: irregular pattern

- Trace shadow ray to random point on sub light source
- Degree of randomness often indicated with “jitter”
- Regular shadow patterns replaced by noise



Area source: jittered subdivision



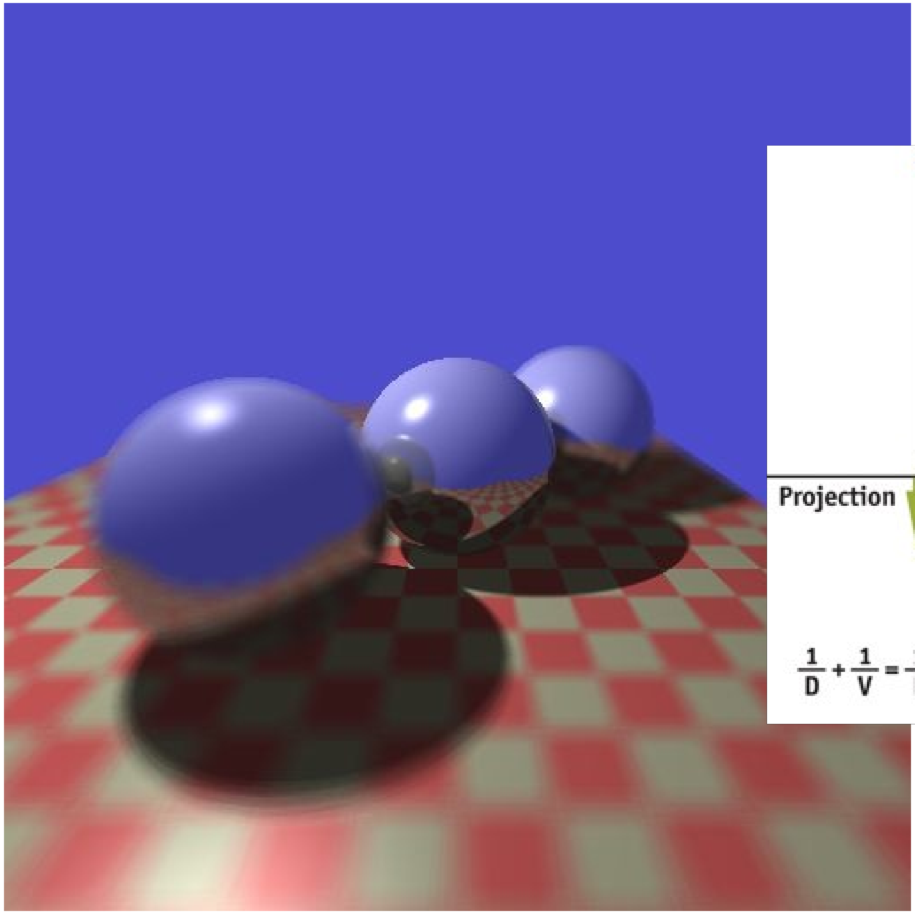
Ray tracing: conclusion

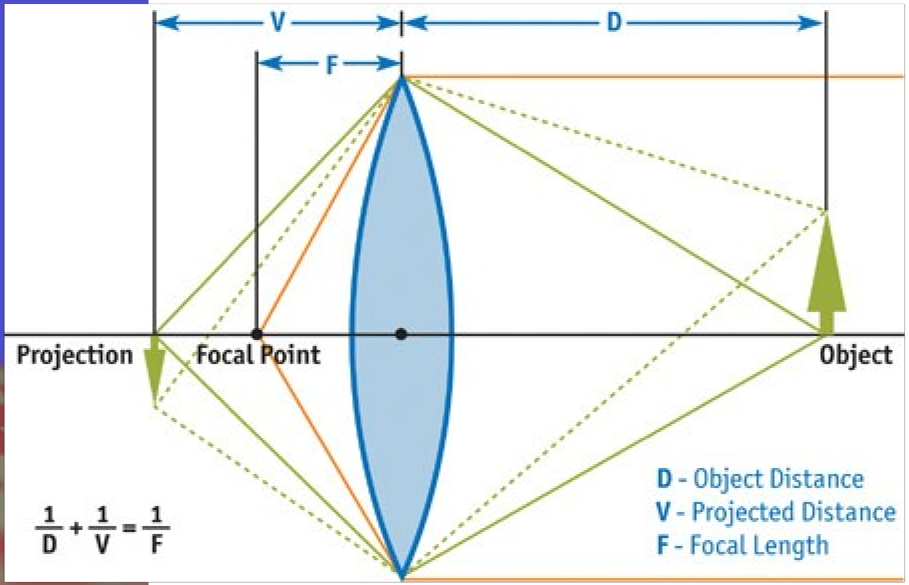
- Spectacular effects:
  - Shadows
  - Mirrors
  - Transparency, refraction
- Simple implementation

- Limitations
  - Expensive
  - Not all light paths possible, missing diffuse interreflection
  - Area light sources possible, but at high price

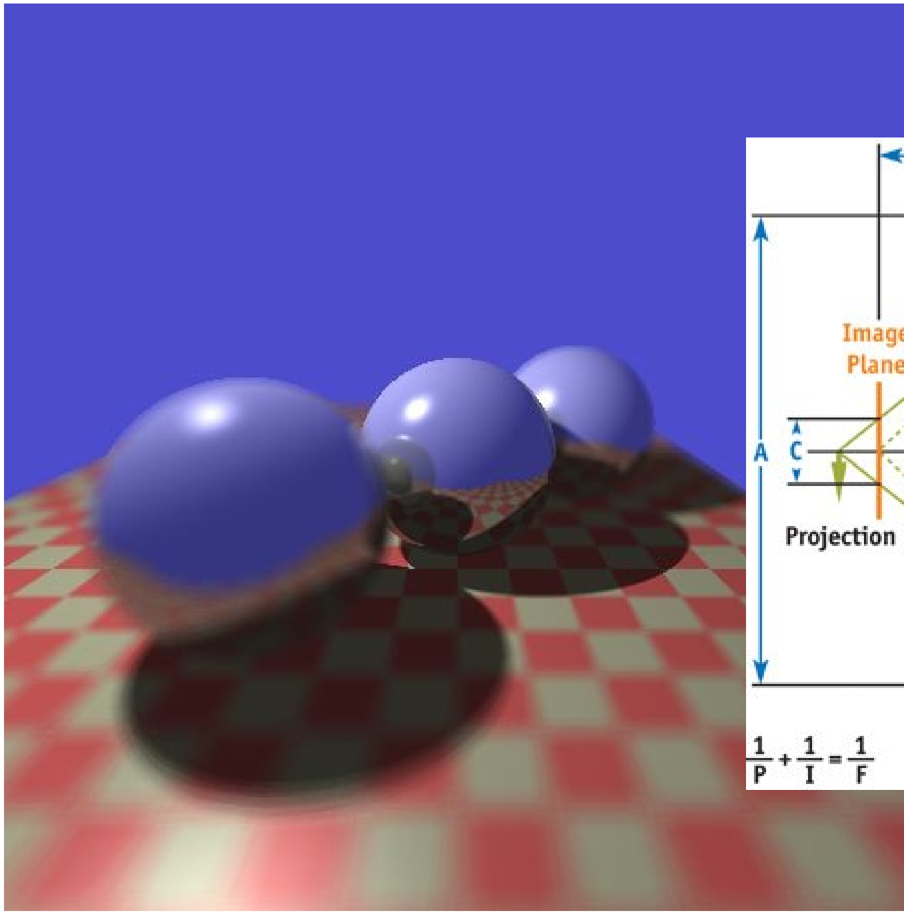
- => *Radiosity* method solves (parts of) limitations

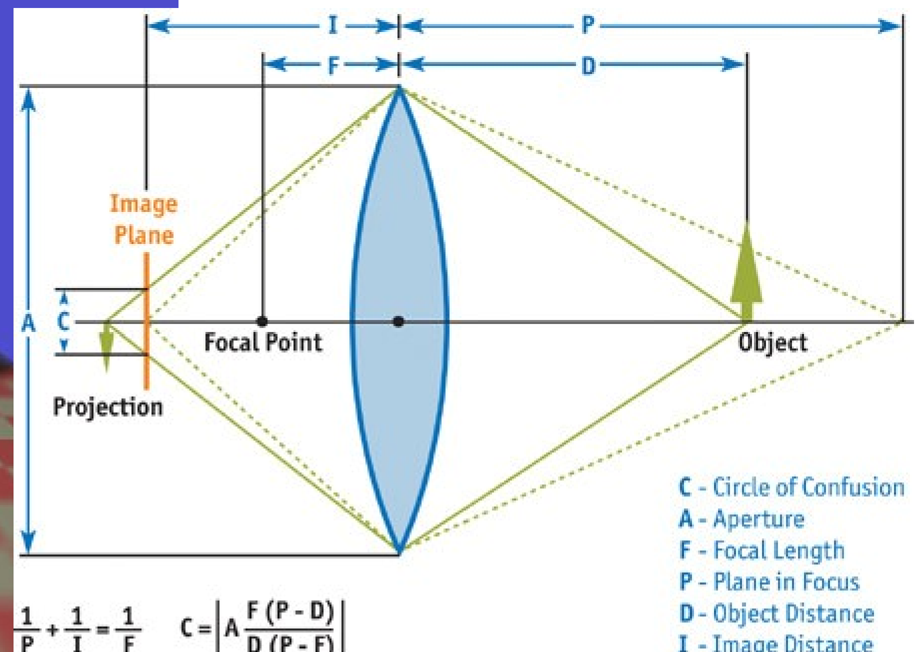
Depth of field





Depth of field

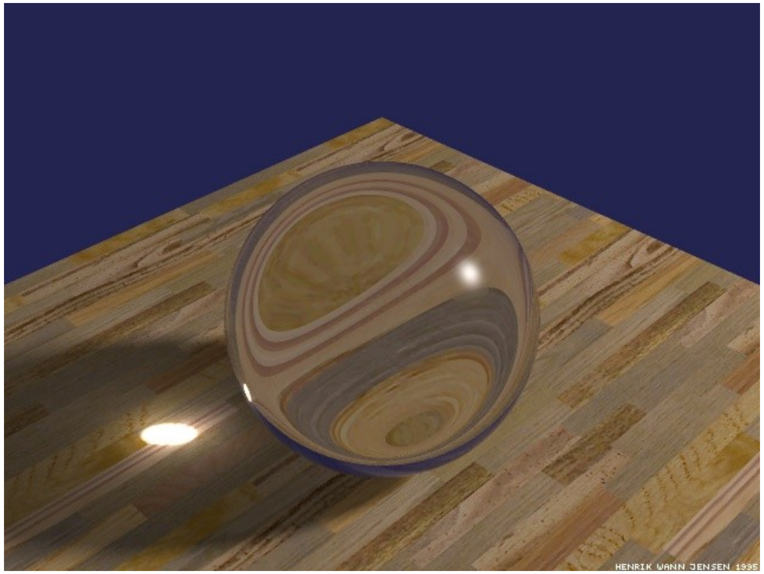




Depth of field effect results from :

Light does no converge to a single point on the image plane!

Caustics



Two-pass ray tracing

- Two-pass method
  - First pass: forward tracing (from lights into scene).
    - Limited to rays from light to reflective and transparent objects
    - When transparency ray hits surface, energy is stored at surface
  - Second pass: backward ray tracing
    - When local illumination applied, also check for stored intensity. Add this intensity to illumination
- Light paths ray tracing with caustics: LS*E and LS*DS*E