Visualisation, Rendering and Animation

2 VO / 1 KU (2001-2004)

Heinz Mayer, Franz Leberl & Andrej Ferko

ferko@icg.tu-graz.ac.at

Short podcast version 2020

Compare Reality - Synthesis

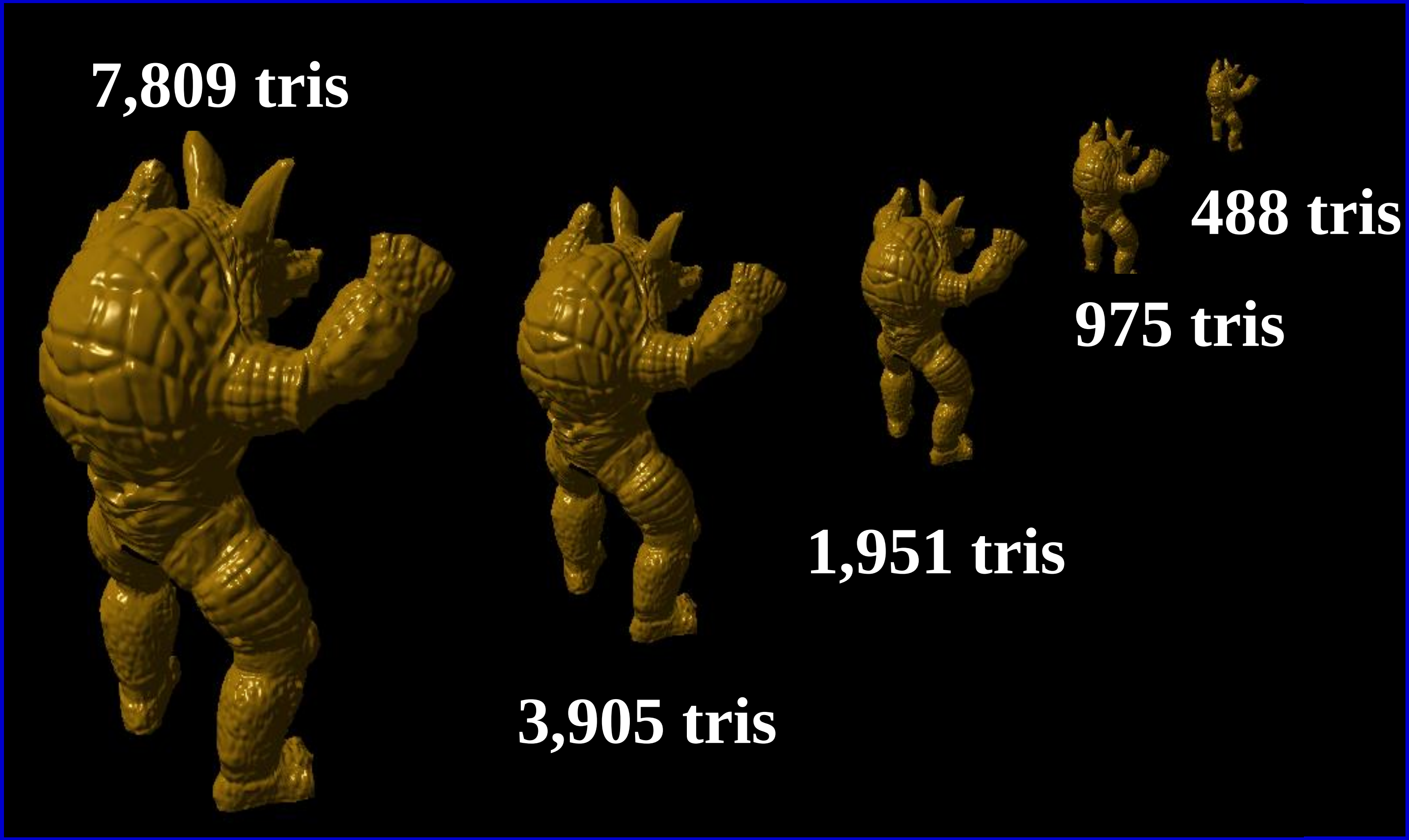
Photograph

Rendering using the deterministic method

Agenda - Photorealism

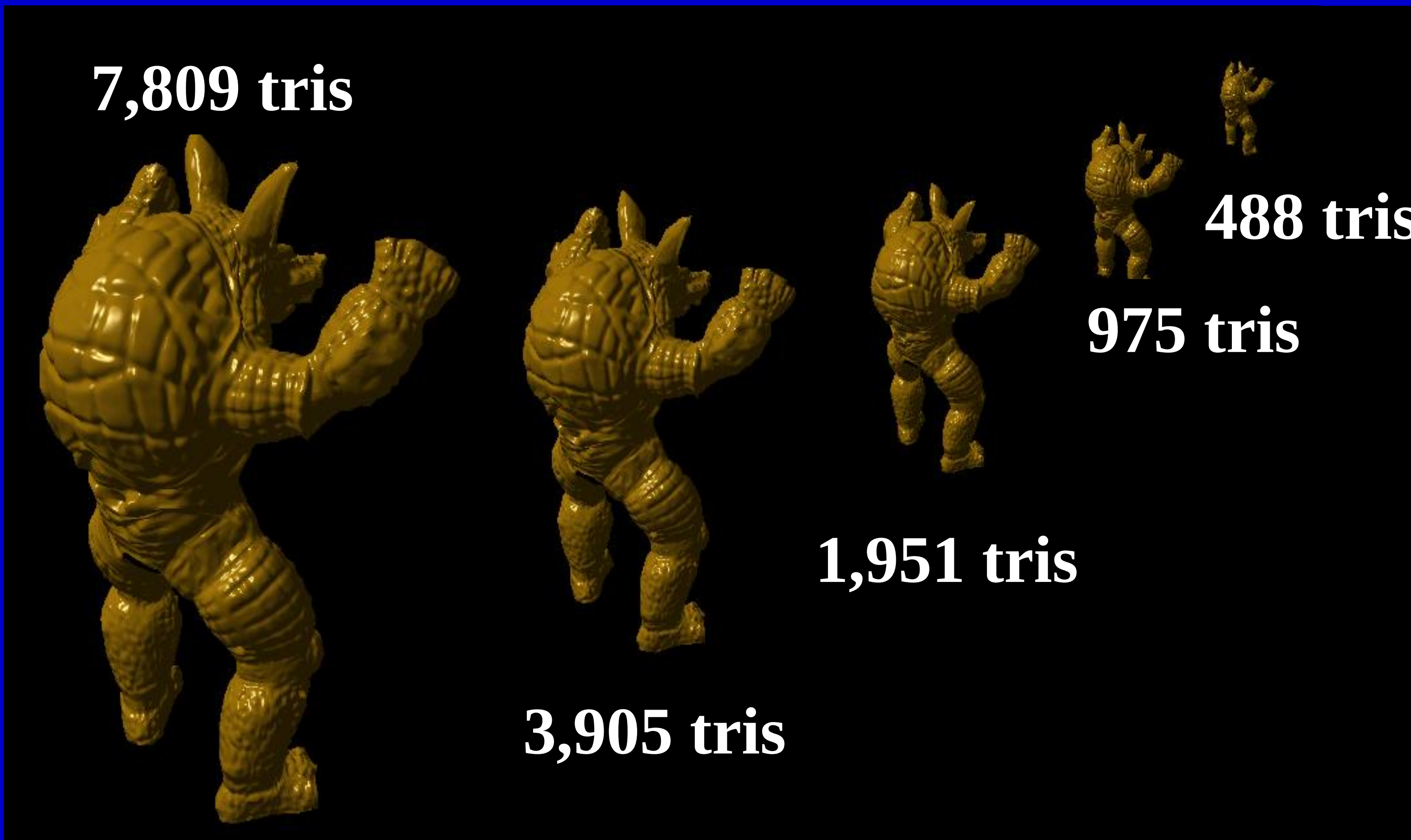
- (Polygonize, generate the mesh)
- Classic Local Illumination Models
- Definition of Light Sources
- Rendering & Light Simulation
- Material & Light Interaction
- Global Illumination, shadows included

APS Level-of-detail Hierarchy



model courtesy of Stanford and Caltech, J. Cohen, S2002 CN 14, modified by AF

APS Level-of-detail Hierarchy



model courtesy of Stanford and Caltech

Original

250,000 Tris

Phong Shading

62,000 Tris

3 pixel error

Original

250,000 Tris

Normal Map

62,000 Tris

3 pixel error

Original

250,000 Tris

Phong Shading

8,000 Tris

15 pixel error

Original

250,000 Tris

Normal Map

8,000 Tris

15 pixel error

Original

250,000 Tris

Phong Shading

1,000 Tris

78 pixel error

Original

250,000 Tris

Normal Map

1,000 Tris

78 pixel error

Big Models:
Plant Ecosystem Simulation

- 16.7 million polygons (sort of, by Luebke, S2002 CN14)

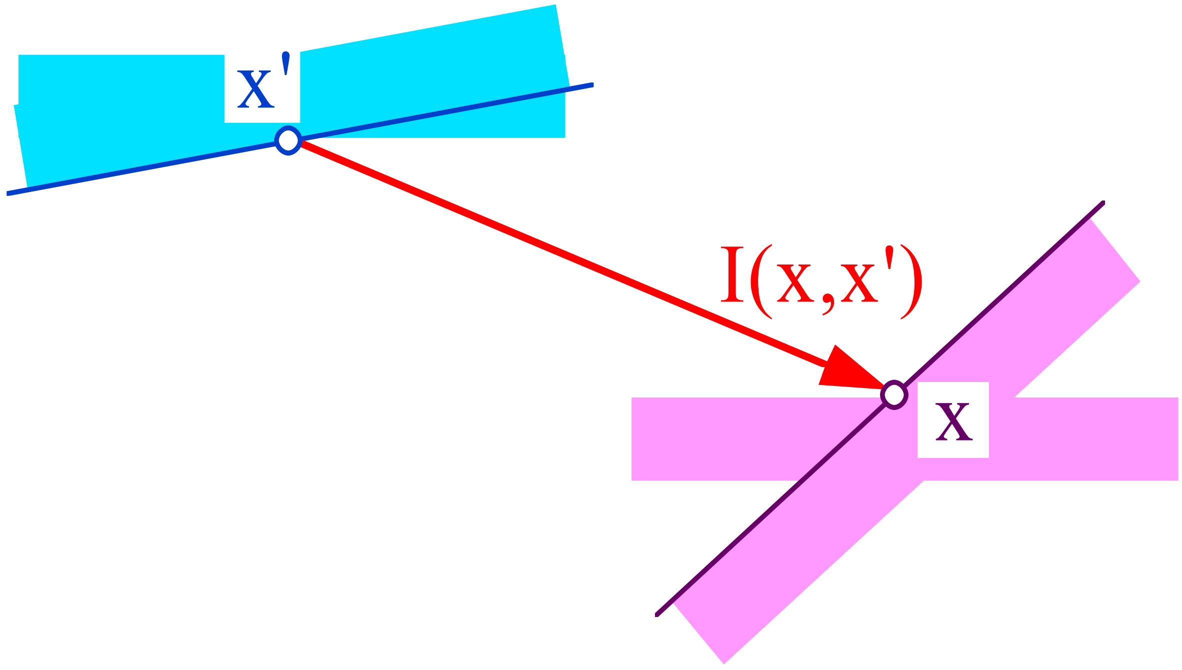
Deussen et al: *Realistic Modeling of Plant Ecosystems*

Agenda - Photorealism

- (Polygonize, generate the mesh)
- Classic Local Illumination Models
- Definition of Light Sources
- Rendering & Light Simulation
- Material & Light Interaction
- Global Illumination, shadows included

Rendering Equation (1)

I(x,x') … intensity arriving at x from x'

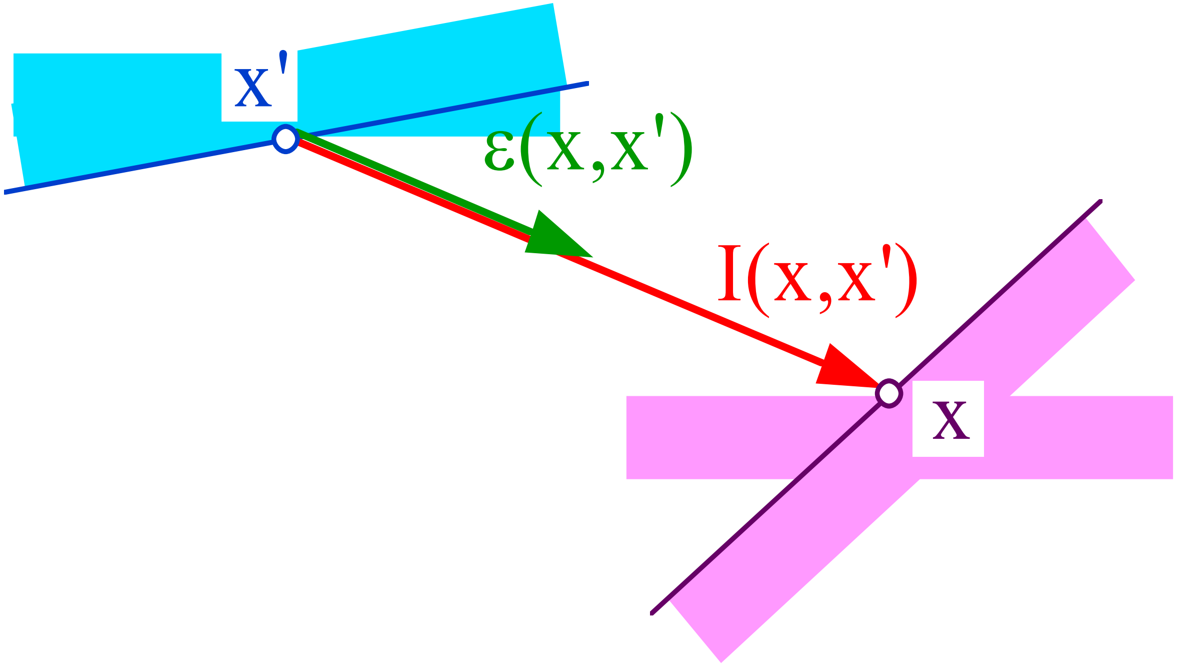


I(x,x') =

Rendering Equation (2)

$\varepsilon(x,x')$ ... ***self-emittance of x' in direction of x***

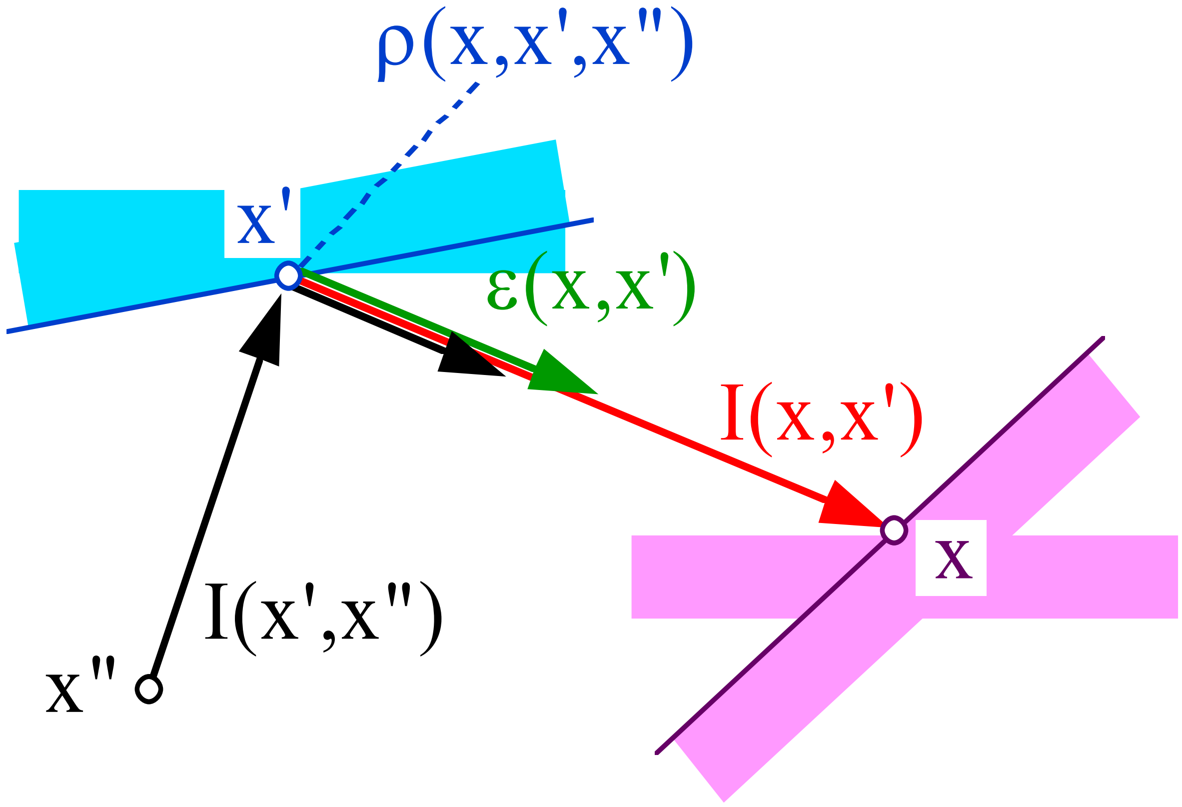
$\delta(x,x')$ ... ***visibility function between x and x'***



$$I(x,x') = \delta(x,x') \cdot [\ \varepsilon(x,x') \qquad]$$

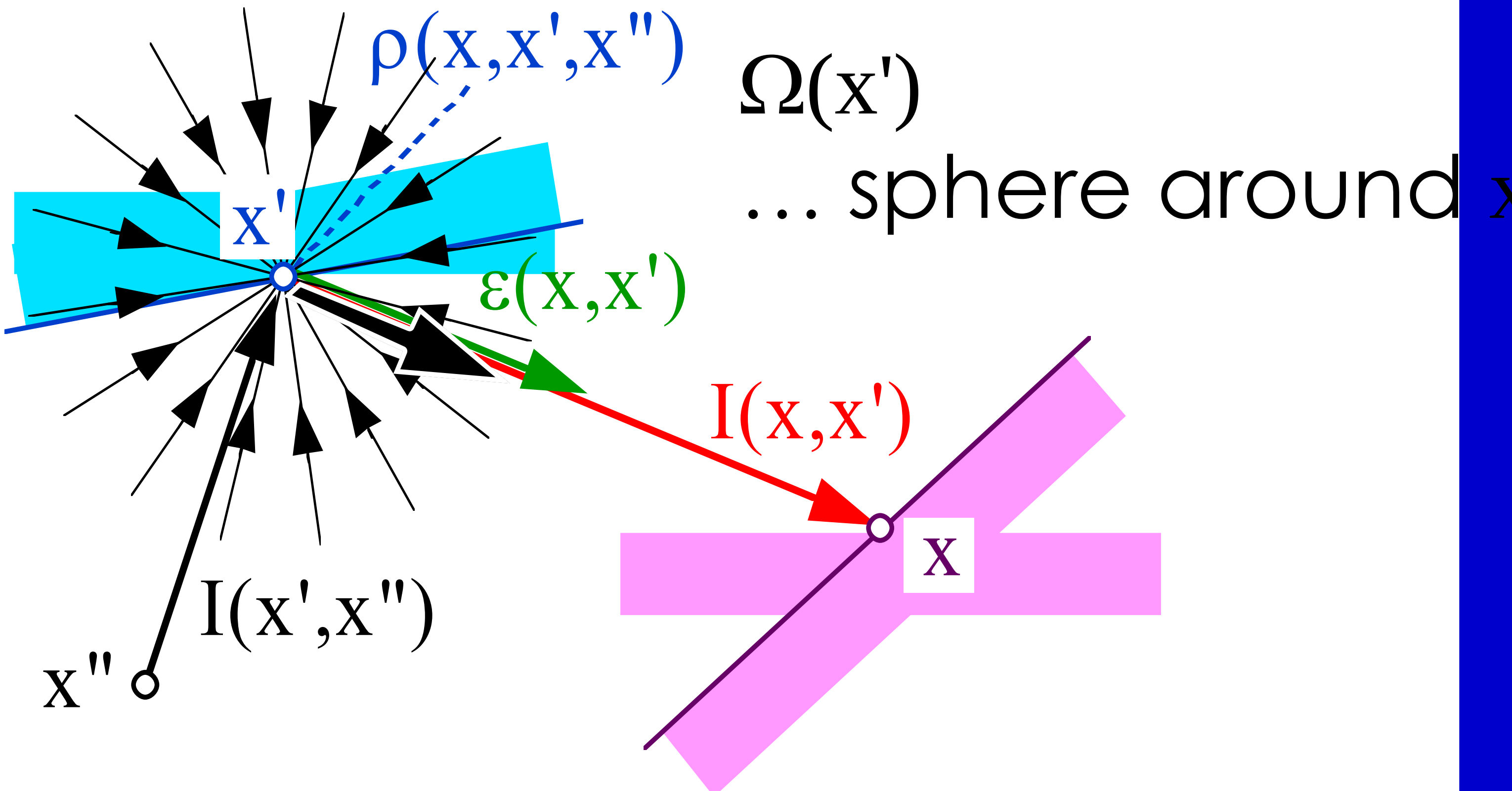
Rendering Equation (3)

$\rho(x,x',x'')$ ***… bidirectional reflection def. funct. at x'***



$$I(x,x') = \delta(x,x') \cdot [\ \varepsilon(x,x') + \quad \rho(x,x',x'') \cdot I(x',x'') \quad]$$

Rendering Equation (4)



$$I(x,x') = \delta(x,x') \cdot \left[\varepsilon(x,x') + \int_{\Omega(x')} \rho(x,x',x'') \cdot I(x',x'')\, dx'' \right]$$

Rendering Equation

describes the light exchange between surfaces in a closed system

$$I(x,x') = \delta(x,x') \cdot \left[\varepsilon(x,x') + \int_{\Omega(x')} \rho(x,x',x'') \cdot I(x',x'')\, dx'' \right]$$

$I(x,x')$ … ***intensity arriving at x from x'***

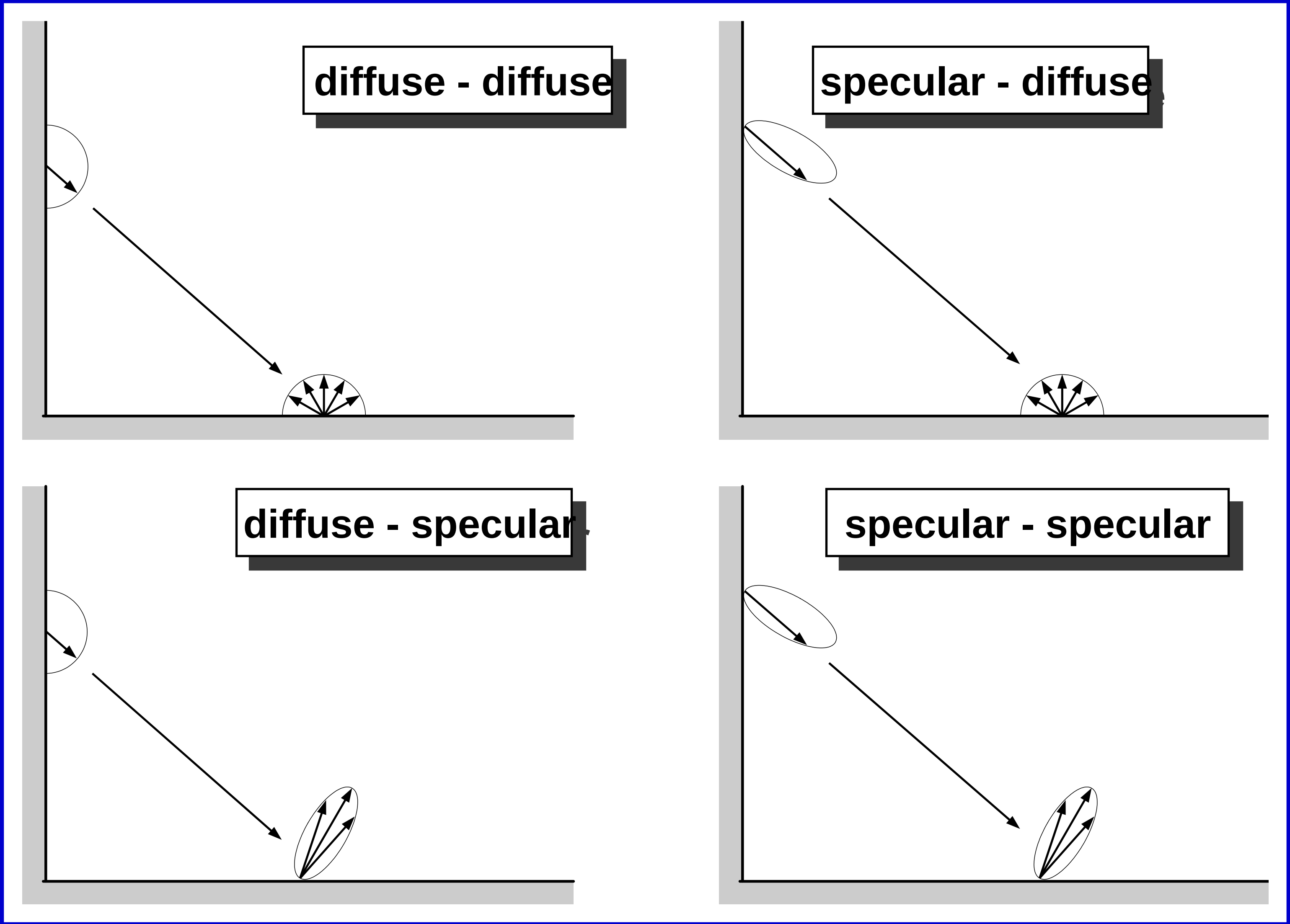
$\varepsilon(x,x')$ … ***self-emittance of x' in direction of x***

$\delta(x,x')$ … ***visibility function between x and x'***

$\rho(x,x',x'')$ … ***bidirectional reflection def. funct. at x'***

$\Omega(x')$ … ***sphere around x'***

Global Illumination Effects

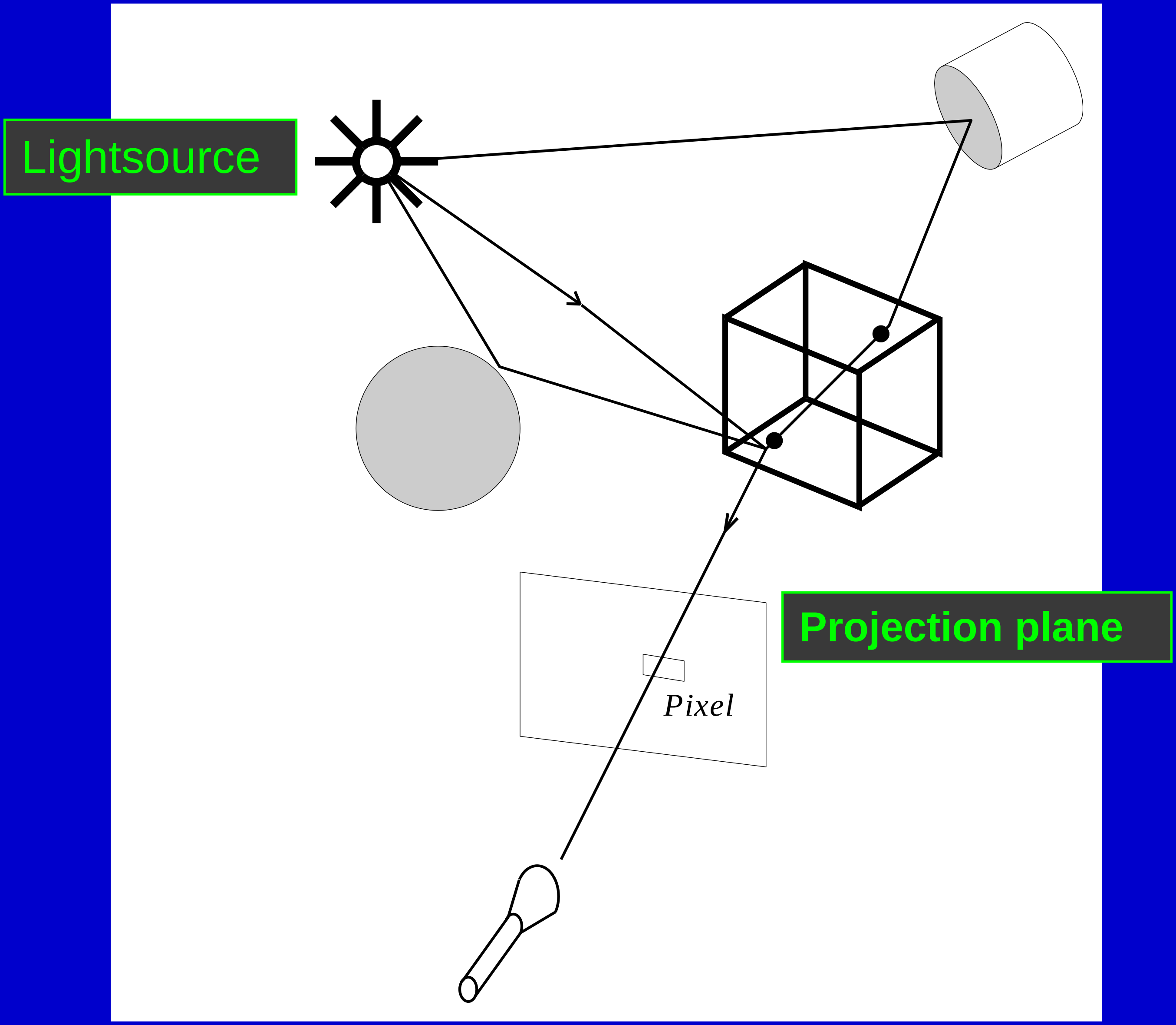


Ray Tracing

Standard Global Illumination Method

Ray Tracing Preview

- Early Use - Descartes (1637) - rainbow
- Optics, geometry for lens systems
- Reflection and refraction
- Three Ray Tracings:
  - Visibility method
  - Recursive Ray Tracing for Global Illumination
  - Volume Rendering Method
- http://www.acm.org/tog/resources/bib/


Lightsource
Projection plane
Pixel

Forward and Backward RT

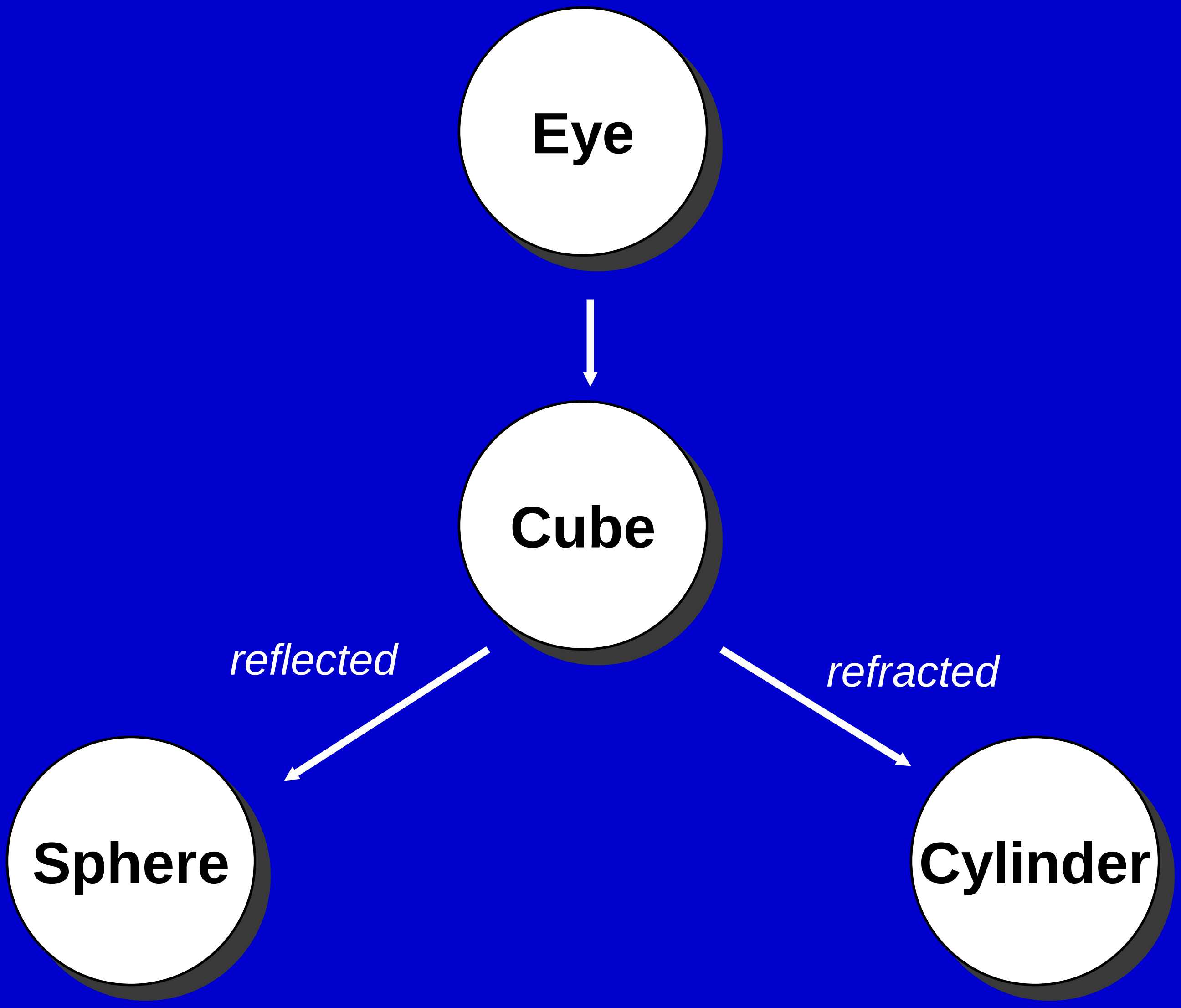
- 2D case by F. S. Hill, Jr.
- Pinhole camera model...
- … extended camera model (TU Wien)
- Pixels & rays (photon vibrations, RGB)
- Forward Ray Tracing
- Lightsource -> Image Plane, unfeasible
- Better one: Eye rays, pixel rays… light
- Shadow and Illumination Rays

Ray Classification & Numbers

- Primary Rays (Visibility <-> Shadow)
- Reflection and Refraction Rays
- Binary tree model
- 100 W bulb/sec about 10E42 photons
- Computer 10E7 initial particles :-(
- Time and memory (Terraflop Club)
- Standard free software is POVRay
- www.povray.org

Recursive Ray Tracing

- **Illumination Model**
  - Visibility/Shadows computation
  - Reflexion/Refraction of light
  - Global mirror reflection
- **Ray Distribution**
  - Indirectly through transparent object
  - Directly (local illumination)
  - Multiple reflexions


Eye
Cube
reflected
refracted
Sphere
Cylinder

Recursive RT Algorithm

1) Visibility algorithm for primary rays (eye -> pixel center)
   - Visible Object Intersection
   - Background (Color setting)

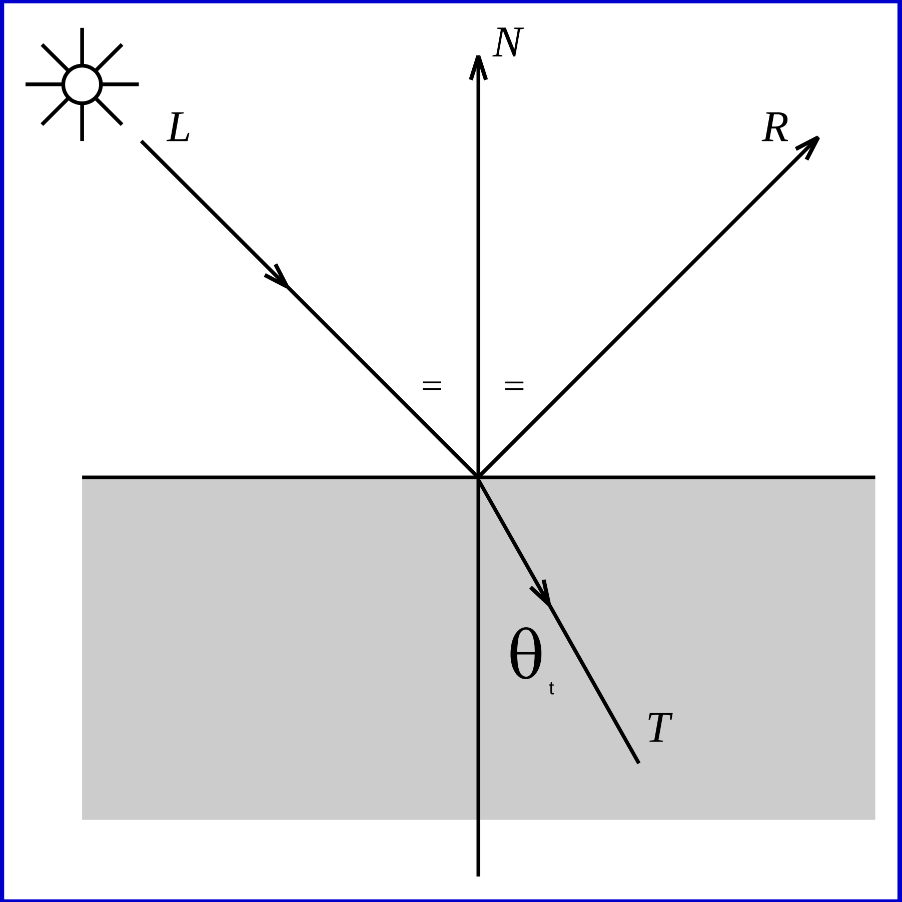
2) Recursive tracing of rays
   - Lightsource hit
   - Intensity increase until < ε (quality, see Optimization)

```
FOR every pixel p DO
1. trace primary ray
   find closest intersection s
2. FOR every light source l DO
     trace shadow feeler l -> s
     IF no intersection THEN
          illumination += influence of l
3. IF surface of s is reflective THEN
     trace secondary ray
     illumination += influence of reflection
   IF surface of s is transparent THEN
     trace secondary ray
     illumination += influence of refraction
```

Reflection & Refraction Vector

$$\vec{R} = 2\left(\vec{N} \cdot \vec{L}\right)\vec{N} - \vec{L}$$

$$\vec{T} = \frac{n_1}{n_2}\vec{L} - \left(\cos\theta_t + \left(\vec{L} \cdot \vec{N}\right)\right)\vec{N}$$



Illumination Model

- **Point Intensity:**

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

- **Local (Phong extended):**

$$I_{local} = I_a\, k_a + I_p\, [k_d\, (N \bullet L) + k_{rl}\, (N \bullet H)^n + k_{tl}\, (N \bullet H')^n]$$

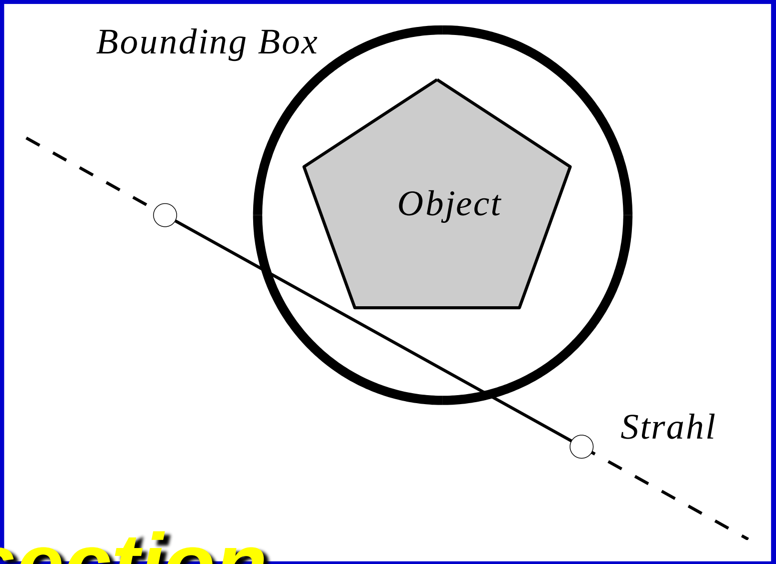
- **Recursive Definition:**

$$I(P) = I_{local} + k_{rg}\, I(P_r) + k_{tg}\, I(P_t)$$

Intersection Computations

- Ray-Scene Intersections (Sphere!)
- Multiple Intersections Possible
- Usable for B-Rep´s
  (95% of Time Consumption)
- Problem Formulation
  – Efficient Intersection Algorithm (stability)
  – Alternative Strategies
    (Bounding Box Checks, Space Subdivisions etc.)

Ray-Sphere Intersection



1) Define the Bounding Box (Sphere)
2) Ray&Sphere Query
3) If YES then Ray-Object Intersection (triangles: barycentric hint)

<u>Pros:</u>

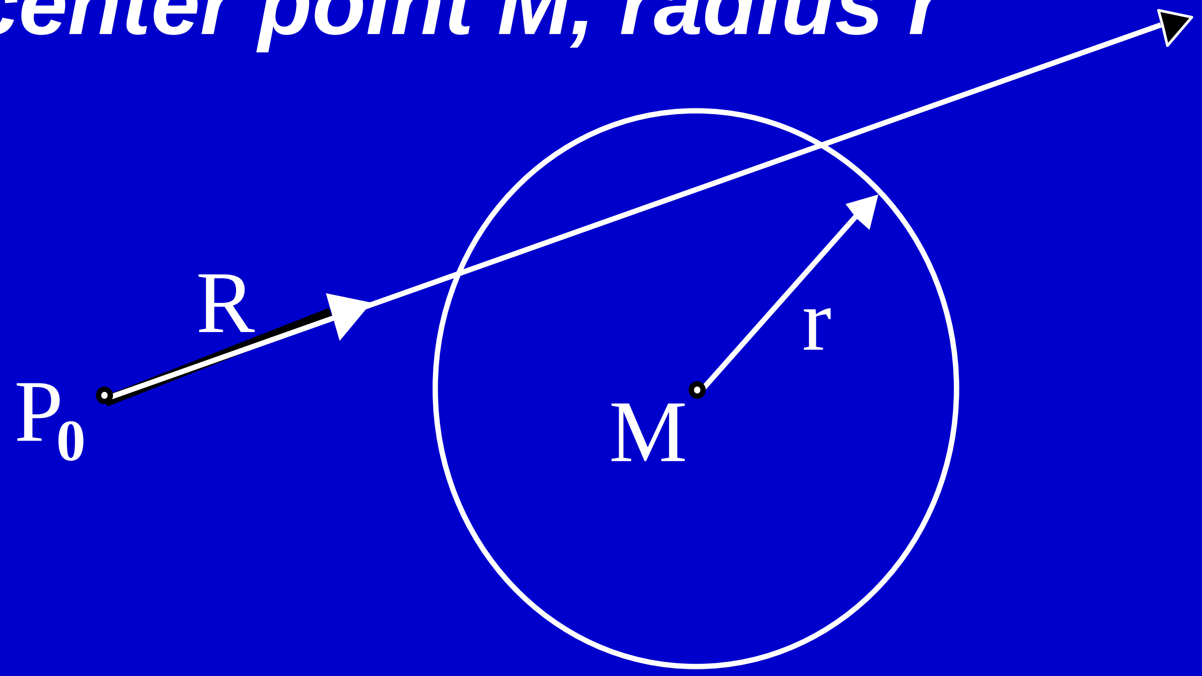
- Simple Bbox Definition
- Efficient Calculation of Intersections

Ray - Sphere Intersection (1)

ray: point P_0, direction vector R, i.e.

$$P(t) = P_0 + t * R \qquad (|R| = 1)$$

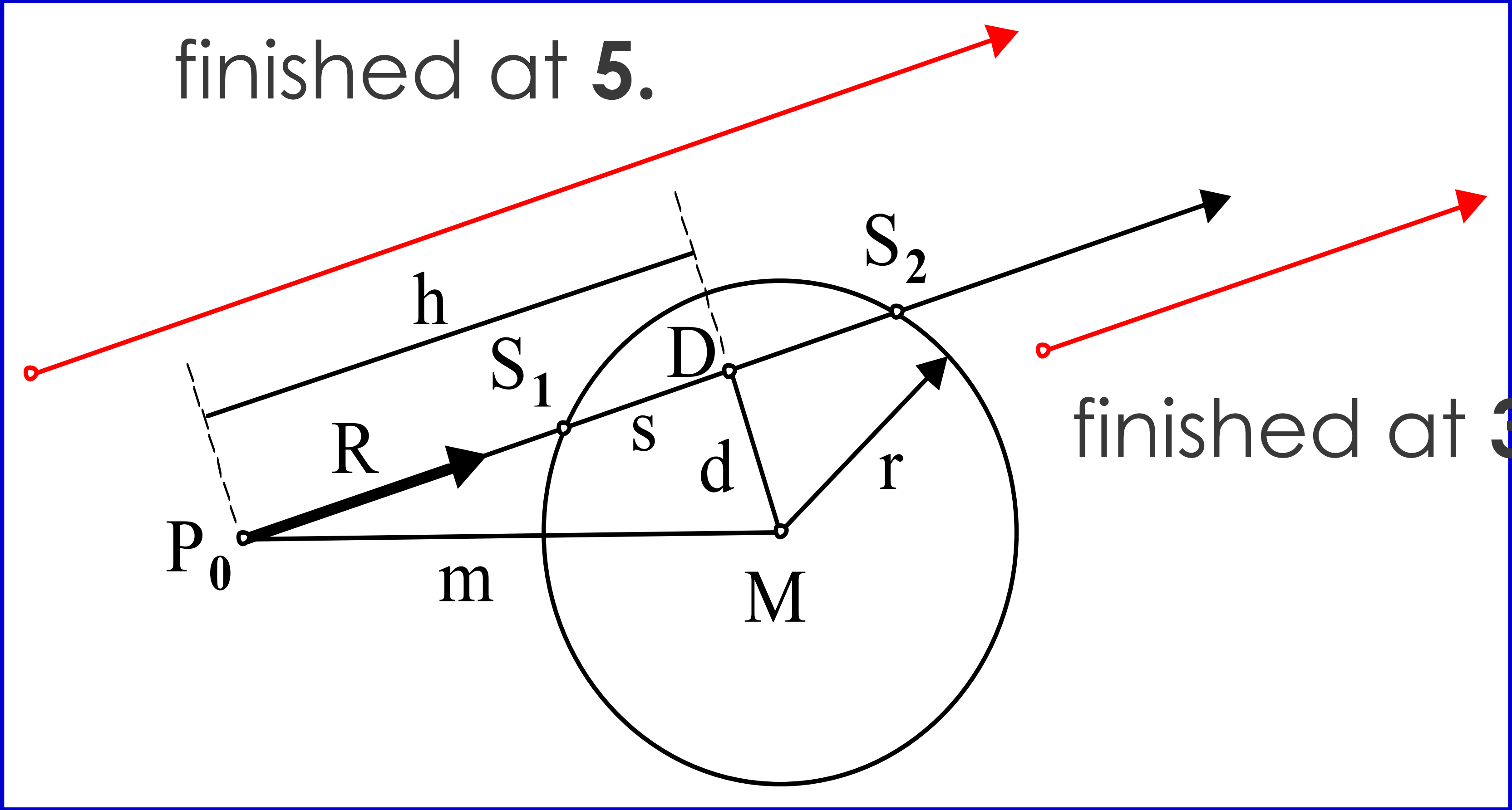
sphere: center point M, radius r



Ray - Sphere Intersection (2)

1. calc. distance² P_0M (=:m^2) $\Rightarrow$ outside or inside?
2. find nearest ray point D to M
3. P_0 outside + R points away from sphere
   $\Rightarrow$ finished
4. calculate distance² of MD =: d^2
5. $d^2 > r^2$ ray misses sphere, finished
6. calculate t values of intersection points =: t_1, t_2
7. calculate intersection points S_1, S_2
8. calculate surface normals N_1, N_2 , finished

Ray - Sphere Intersection (3)



Ray-Polygon Intersection (1)

ray: $P = P_0 + t \cdot R$ $\quad |R| = 1$

polygon plane:

$$A \cdot x + B \cdot y + C \cdot z + D = 0 \quad (A^2+B^2+C^2)=1$$

normal vector of the plane: $N = (A\ B\ C)$

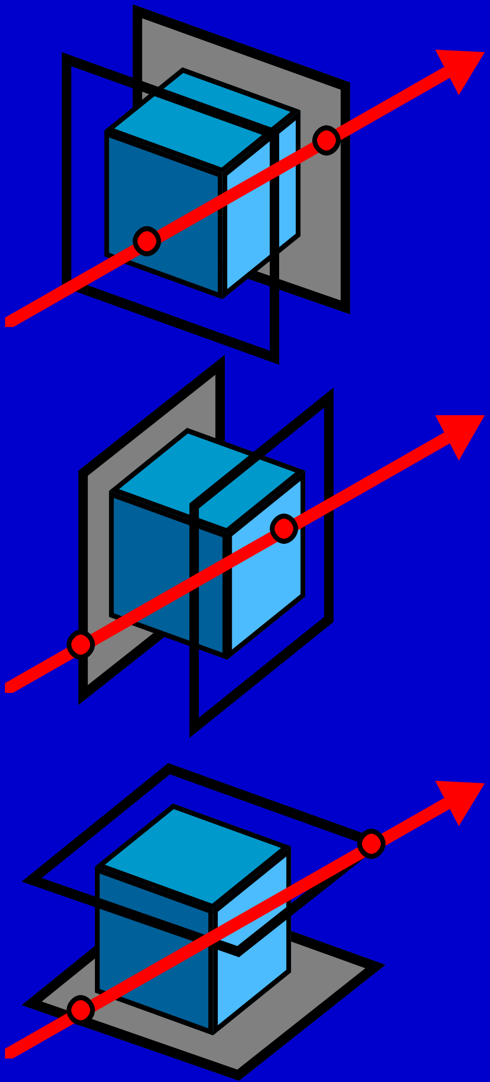
other notation for plane: $N \cdot P = -D$

$$N \cdot (P_0 + t \cdot R) = -D \Rightarrow \boxed{t = -(D + N \cdot P_0)/N \cdot R}$$

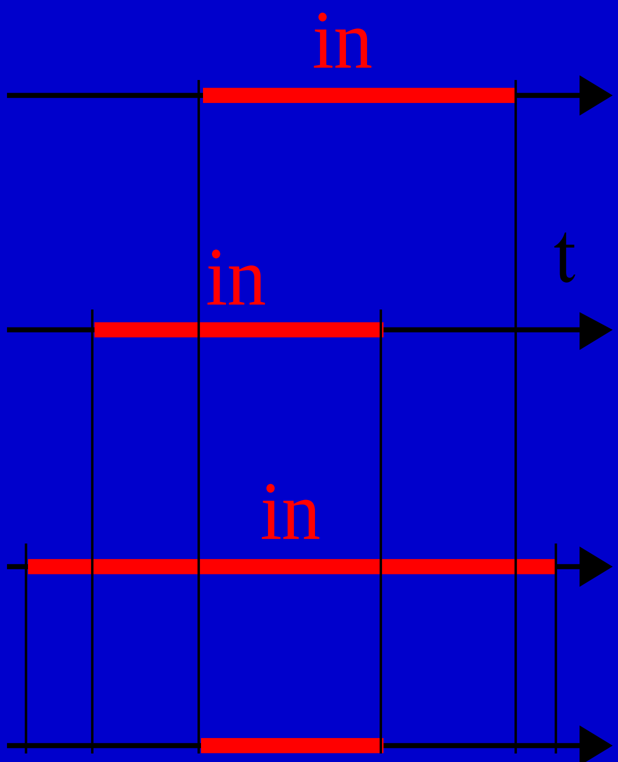
Ray-Polygon Intersection (2)

- $N \cdot R = 0 \Rightarrow$ ray is parallel to plane
- $t<0 \Rightarrow$ intersection point before ray origin
- intersection point: evaluate ray equation with t
- inside/outside polygon test for intersection point with even-odd-rule

Ray-Box Intersection



plane classifications:



box classification

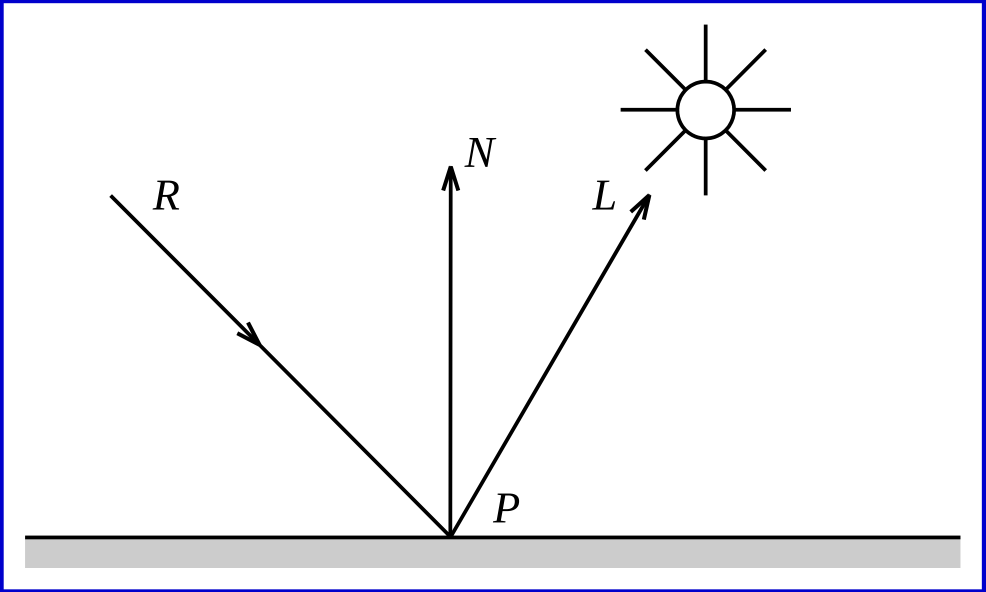
for *all* convex polyhydra: intersections only necessary with planes, no test "intersection point inside polygon?" necessary!

Shadow Feeler

<u>Ray = P + t (L - P)</u>

- P… Surface Point
- t … Parameter of Representation for Shadow Feeler

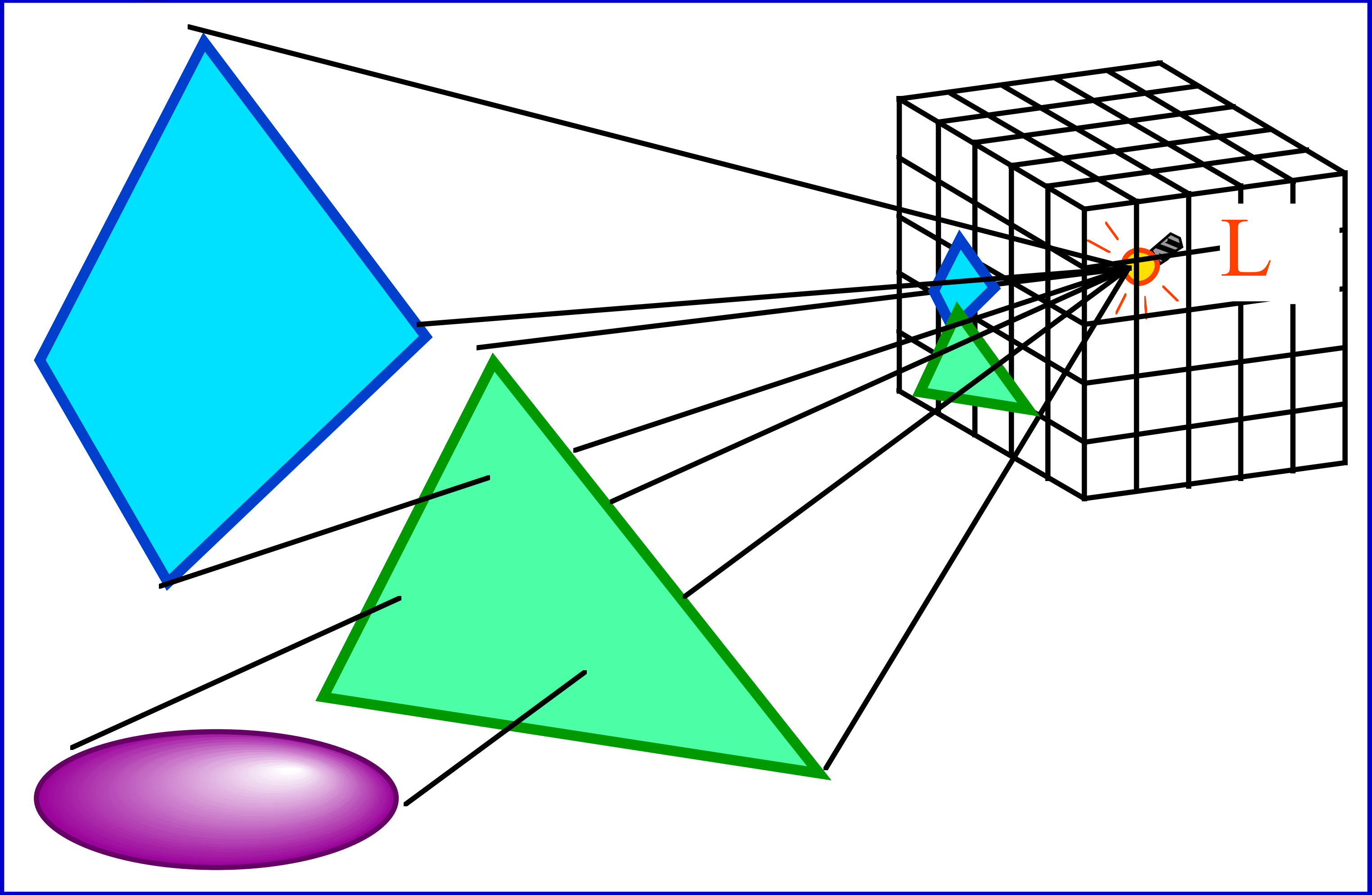
If (Intersection
for <u>0 < t < 1)</u>
then <u>no</u> Impact
of the given
Lightsource



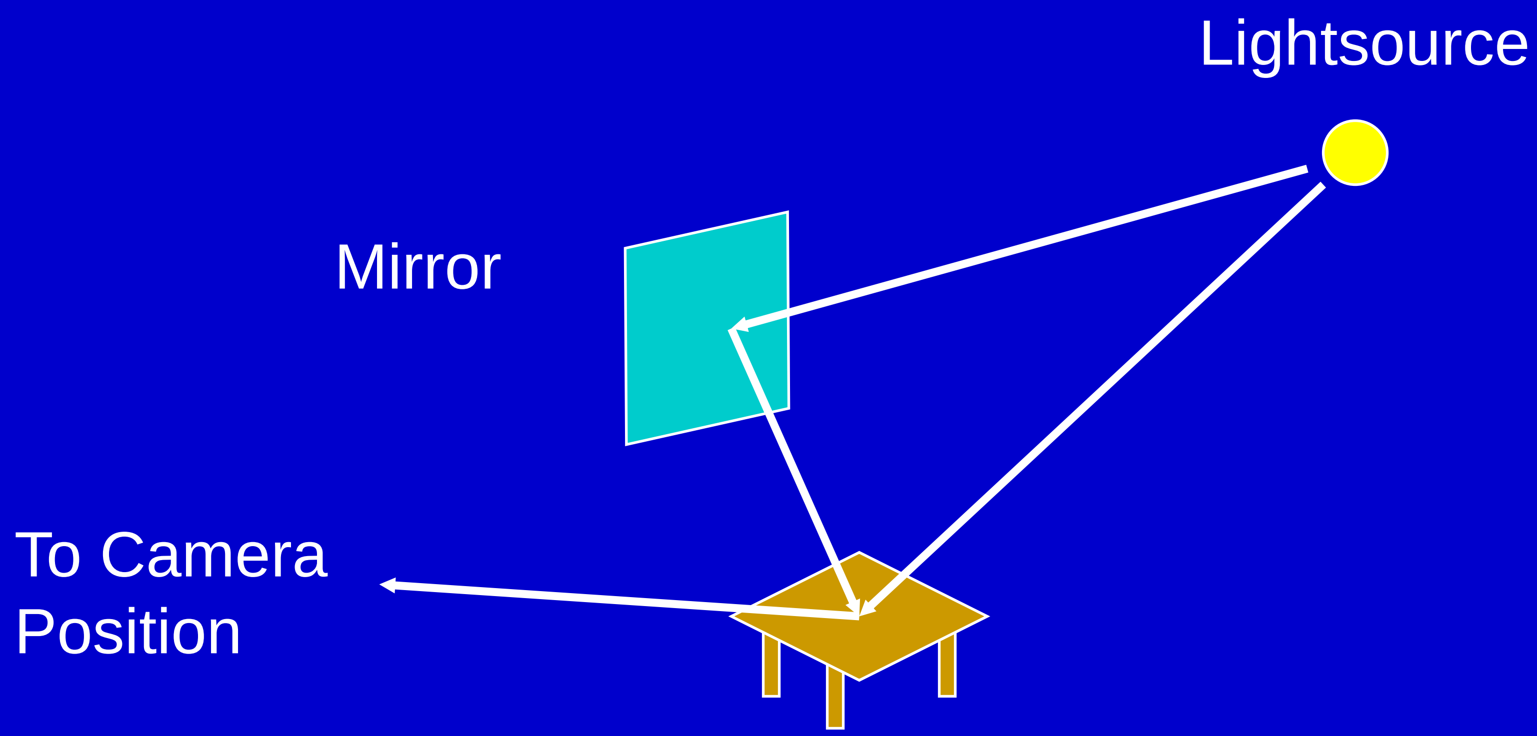
Disadvantages

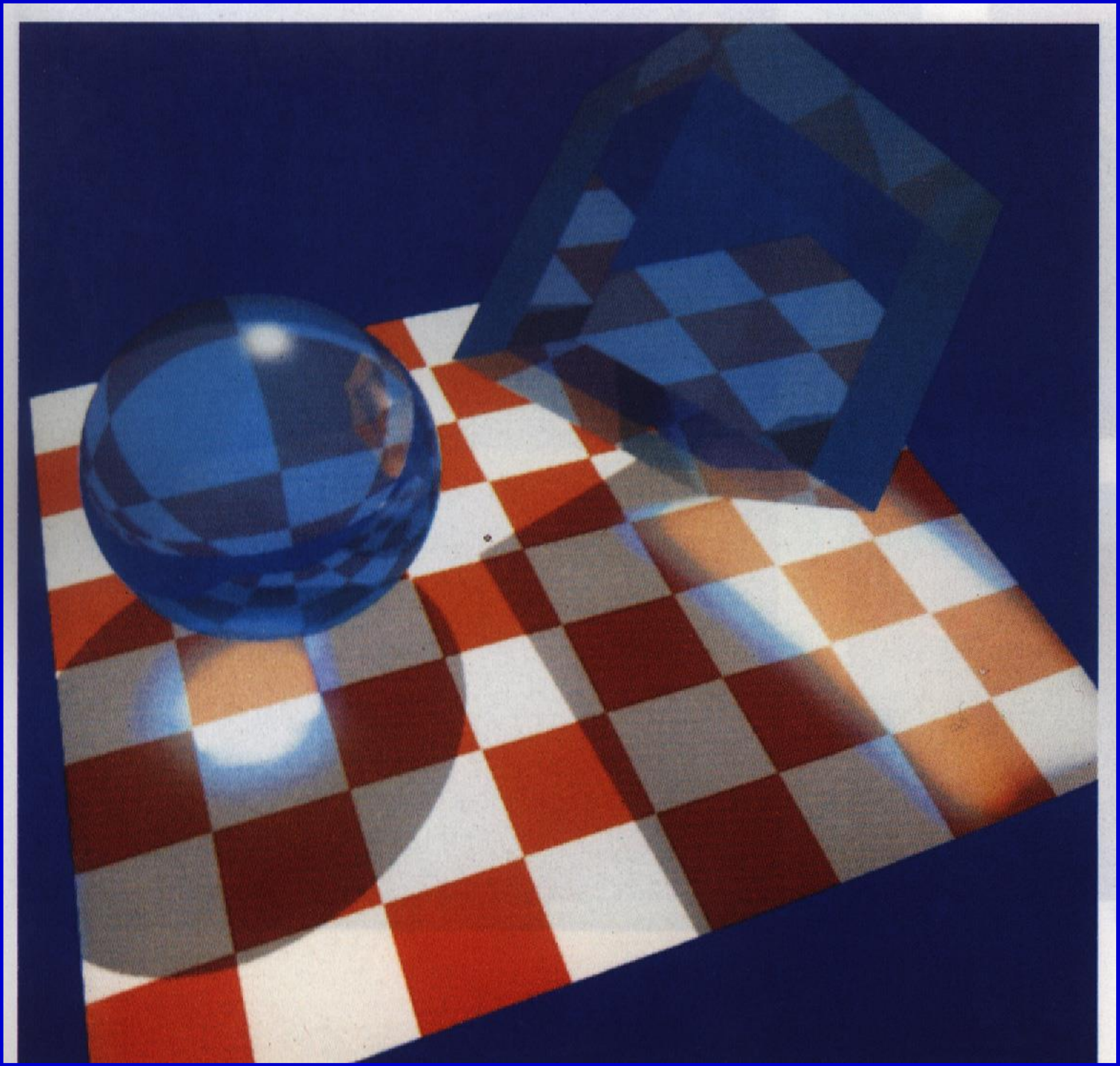
- High complexity, too many rays (and intersections)
- Restricted „globality“ for mirror reflection and refraction (no global diffuse illumination)
- View dependent & visual drawbacks
  - Anti-Aliasing
  - Sharp shadow borders
  - Depth of field problem

Light-Buffer by Haines&Greenberg



Backwards Ray Tracing





Ray Tracing Summary

- Very old geometric model
- Industrial standard and POVRay
- Computationally expensive
- Many improvements published:
- www.acm.org/tog/resources/bib/
- Parallelisation, ray space, random walk, two-pass methods, instant radiosity by Keller, … research...

Figure 12. A still life image showing examples of procedural and scanned textures and patterns.

Thank You...

... for Your attention.

Visualisation, Rendering and Animation

2 VO / 1 KU (2001-2004)

Heinz Mayer, Franz Leberl & Andrej Ferko

ferko@icg.tu-graz.ac.at

Short podcast version 2020