técnicas avanzadas de gráficos

ingeniería multimedia

Seminario 10

Raytracing

Departamento de Ciencia de la Computación e Inteligencia Artificial
Universidad de Alicante

Raytracing

How does the light behave in a real environment?

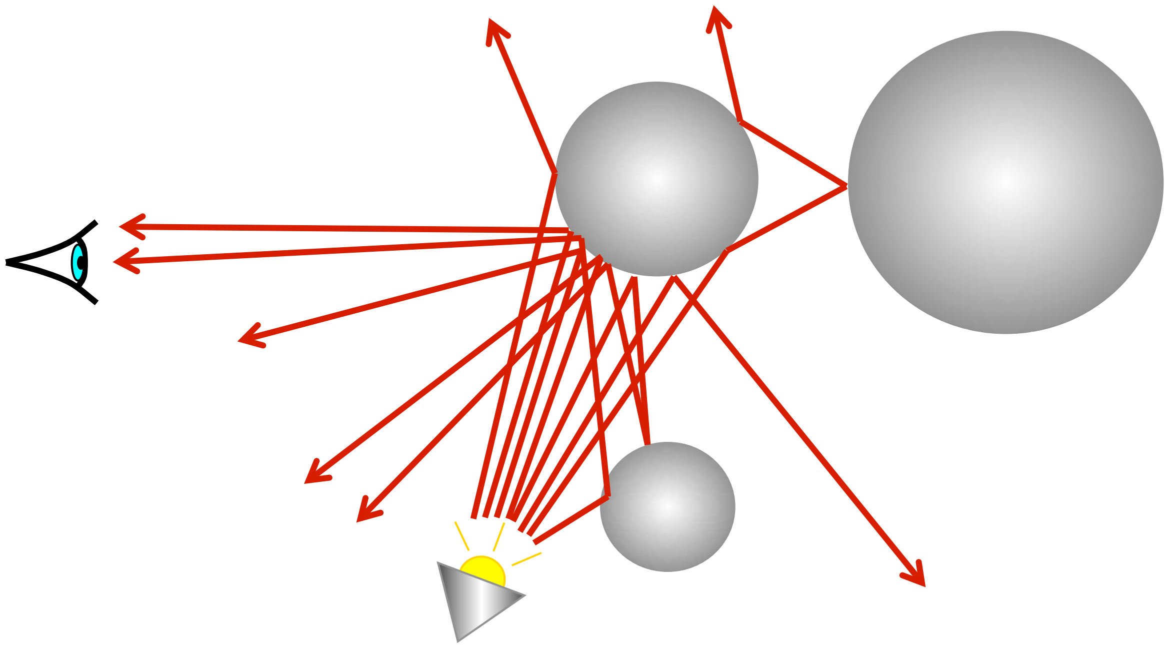
How can this behaviour be simulated?

Raytracing

- Very realistic rendering
- Global illumination model
  - Direct light from light sources
  - Light reflected by other objects
  - Light refracted by the object
- Incorporates
  - Hidden surfaces removal
  - Shadow calculation

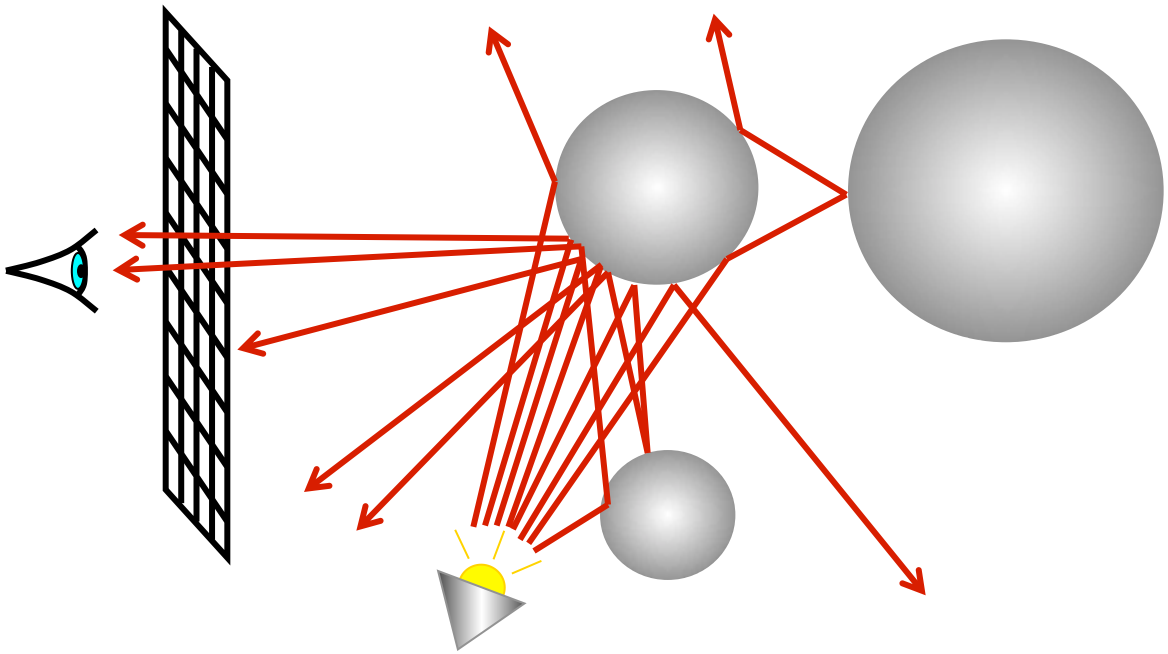
Introduction

- Ray Tracing is based on physical properties of light
- How is a real image created?



Introduction

- Ray Tracing is based on physical properties of light
- How is a real image created?



Forward Ray Tracing

- Forward Ray Tracing:
  - Every ray coming from the light source is traced
  - Every ray is reflected when it falls on the objects
  - Finally, rays falling on the observer are displayed
- Drawback of this approach
  - The number of rays is infinite, so only some of them can be considered
  - Despite this simplification, the temporal cost is so high that it is not affordable
  - Only a small portion of initial rays falls on the observer, so, a lot of calculations are wasted

Backward Ray Tracing

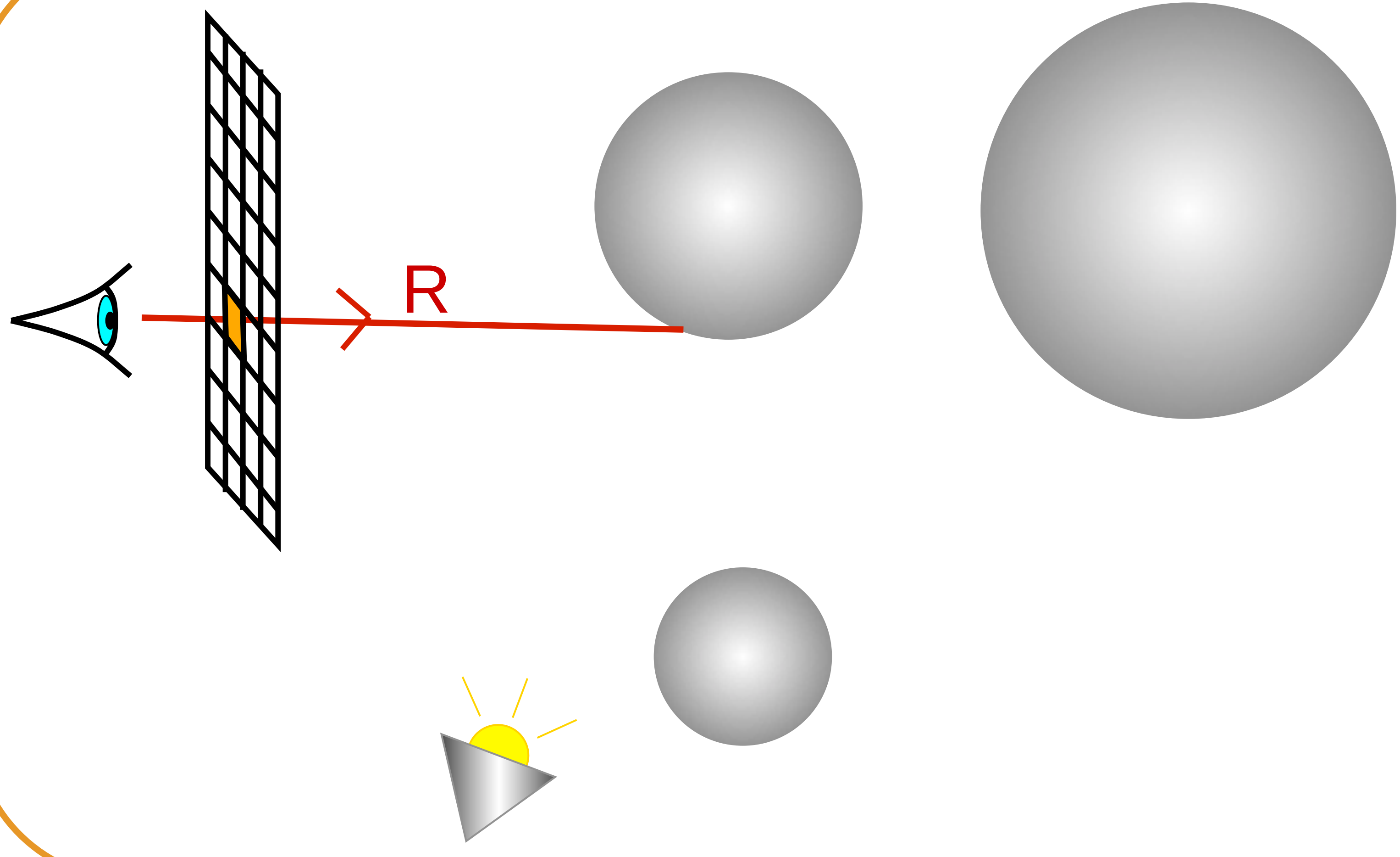
- Backward Ray Tracing
  - Only the rays falling on the observer are traced
  - The process is done inversely: the rays come from the observer and are traced backward
  - The light from which every ray comes is calculated
- Properties of this approach
  - Only a ray is traced for every pixel of the image (or a small set of rays)
  - No calculations are wasted, so the temporal cost goes on being high but it is affordable
  - It is the practical algorithm, that is always implemented

Basic Ray Tracing
Algorithm

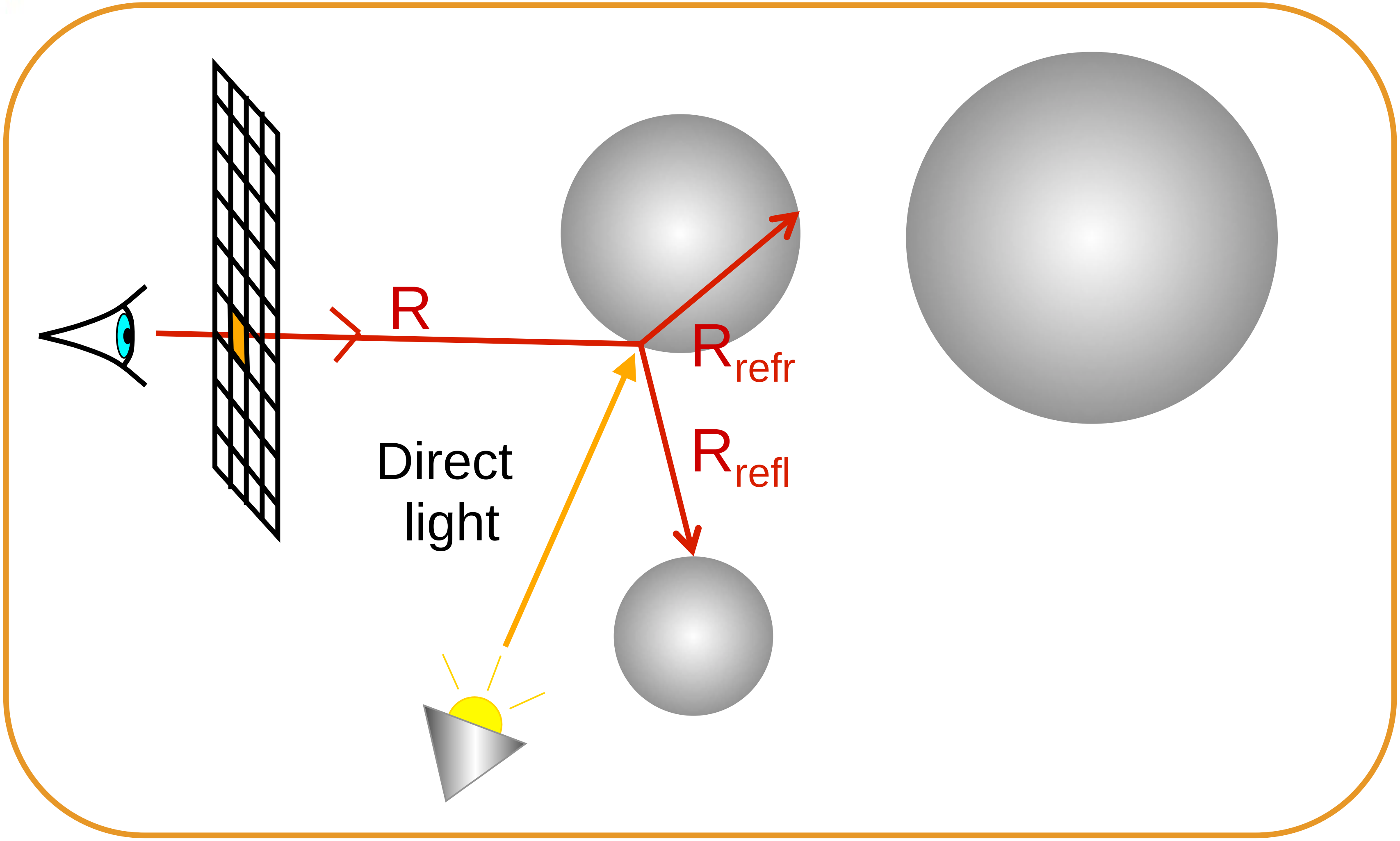
- Trace a ray from the observer through every pixel of the image
- Calculate the first intersection of the ray with any object of the scene
- The colour of the pixel is made up of 3 components:
  - Local component (Phong model or any other local model)
  - Reflection component → A new ray is generated (reflected ray)
  - Refraction component → A new ray is generated (refracted ray)
- Repeat this process with the new rays

- It is a recursive method. Base cases:
  - The ray does not intersect with any object
  - The contribution of this ray to the initial ray can be considered insignificant
- It can easily be parallelized
- The object occlusion is implicit

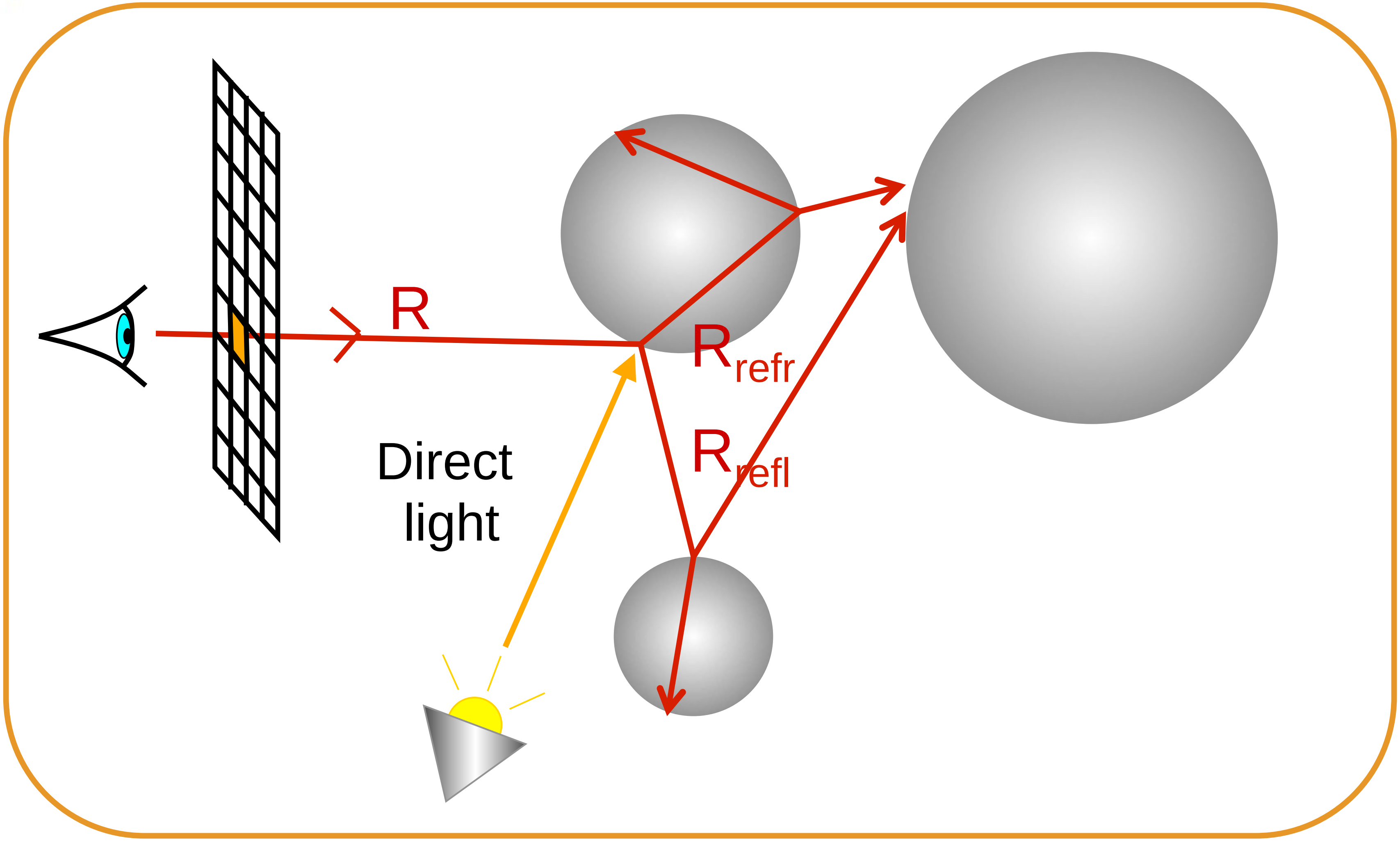
Basic Ray Tracing
Example



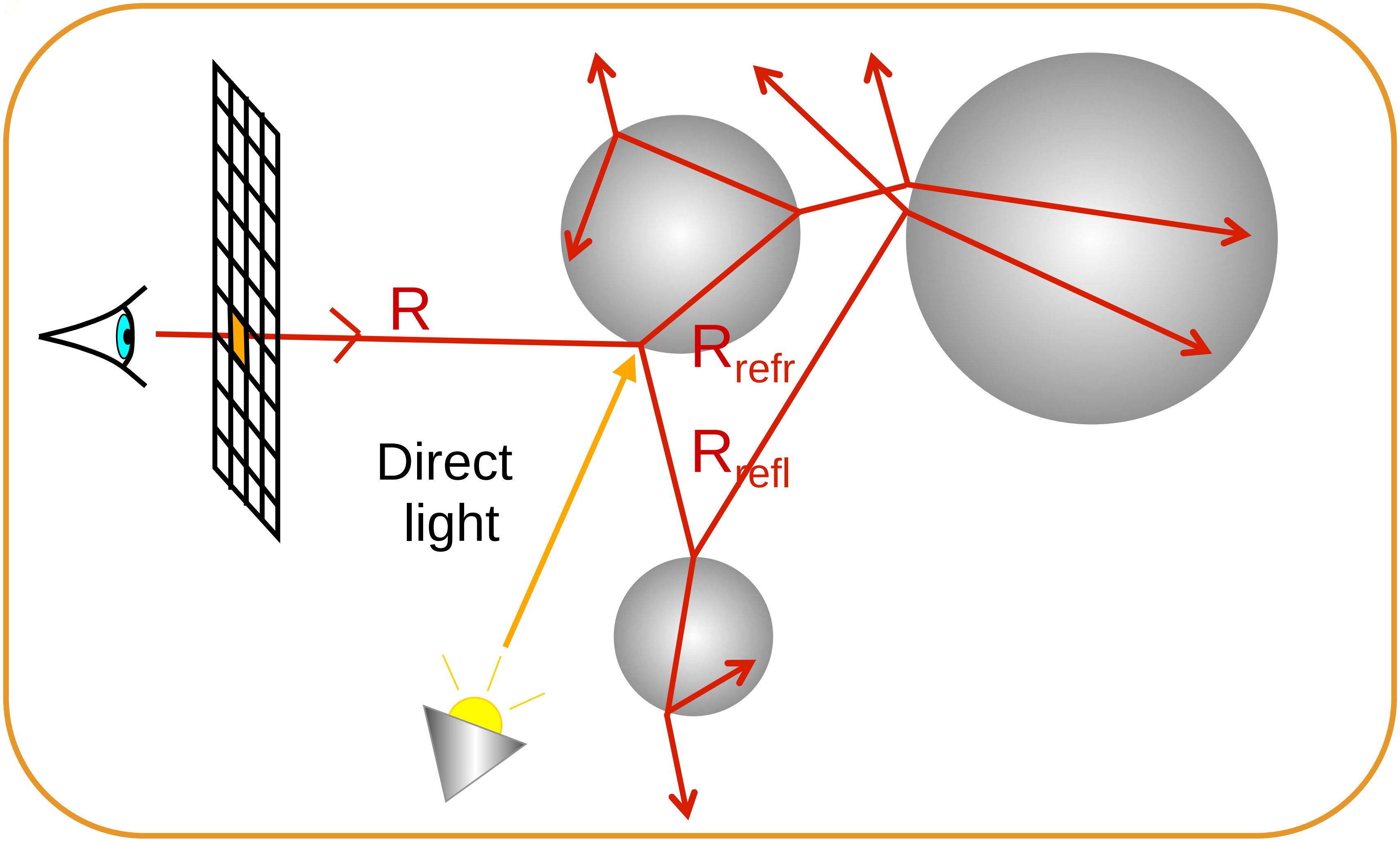
Basic Ray Tracing
Example



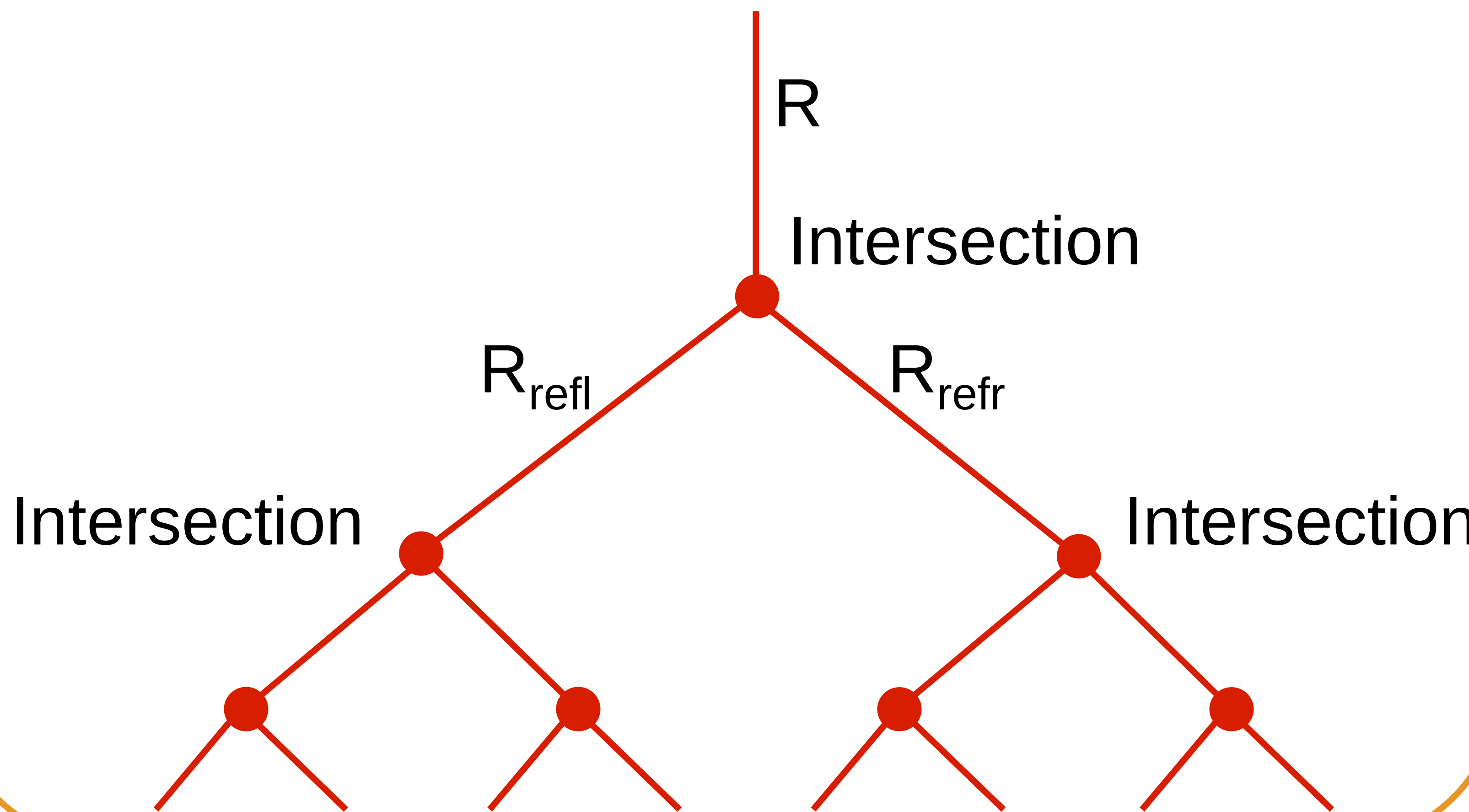
Basic Ray Tracing
Example



Basic Ray Tracing
Example



Basic Ray Tracing
Recursions tree



Intersection calculation

- This operation is the most frequent → It must be very quick
- Special cases
  - Ray-Sphere
  - Ray-Polyhedron
  - Ray-Parallelepiped
  - Other intersections

Intersection calculation
Ray-Sphere

- Parametric equation of ray

$$x = p_x + (q_x - p_x)\, t$$
$$y = p_y + (q_y - p_y)\, t$$
$$z = p_z + (q_z - p_z)\, t$$

- Replace in the sphere equation

$$(x-c_x)^2 + (y-c_y)^2 + (z-c_z)^2 = r^2$$

- First intersection = lower positive value for *t*

Intersection calculation
Ray-Polyhedron

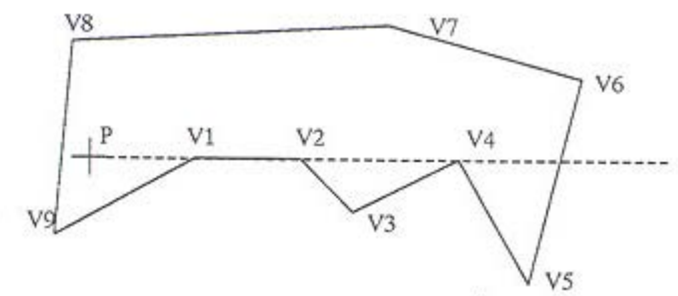
- General polyhedron:
  - Replace ray equations in the plane equation of every polyhedron face:

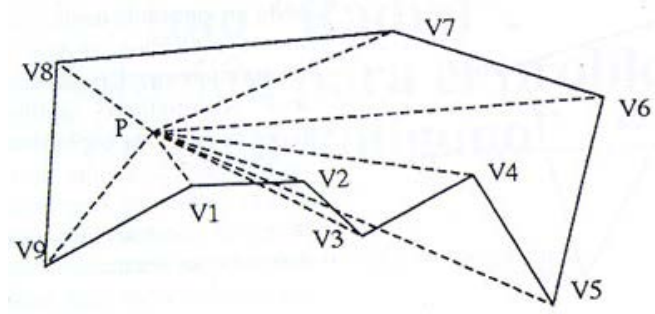
$$ax + by + cz + d = 0$$

  - An intersection ray-plane exists if any positive value for *t* exists
  - Check if the intersection point is inside the polygon

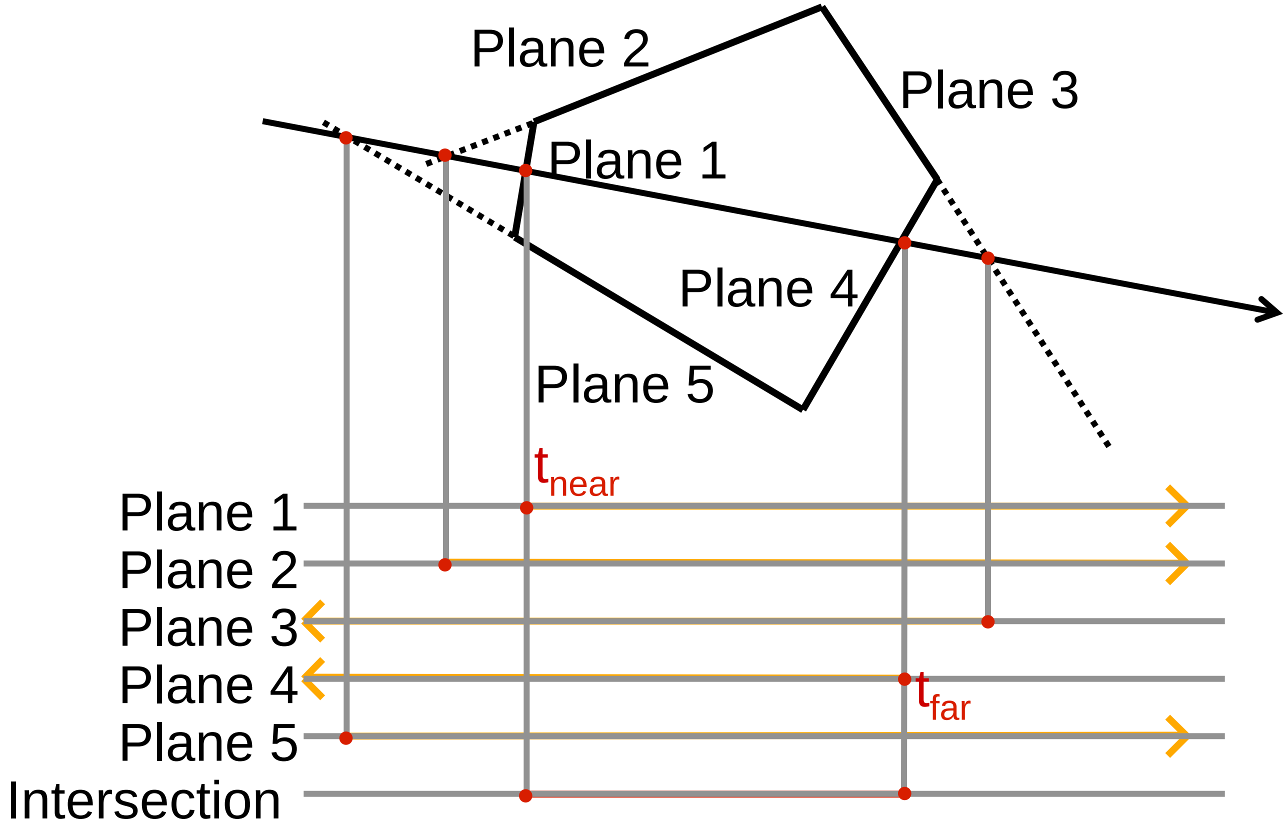
Intersection calculation
Ray-Polyhedron

- General polyhedron:
  - Check if the intersection point is inside the polygon
    - Jordan curve theorem algorithm
    - Radial algorithm
    - Left-right side algorithm (only convex polygons)





- Convex polyhedron



Intersection calculation

Ray-Parallelepiped

- Particular case of convex polyhedron
- t_{near} and t_{far} are:

$$t_{near} = \mathbf{max}\ (tc_x, tc_y, tc_z)$$

$$t_{far} = \mathbf{min}\ (tl_x, tl_y, tl_z)$$

- tc and tl are:

$$tc_x = \frac{a_x - p_x}{q_x - p_x} \quad tc_y = \frac{a_y - p_y}{q_y - p_y} \quad tc_z = \frac{a_z - p_z}{q_z - p_z}$$

$$tl_x = \frac{b_x - p_x}{q_x - p_x} \quad tl_y = \frac{b_y - p_y}{q_y - p_y} \quad tl_z = \frac{b_z - p_z}{q_z - p_z}$$

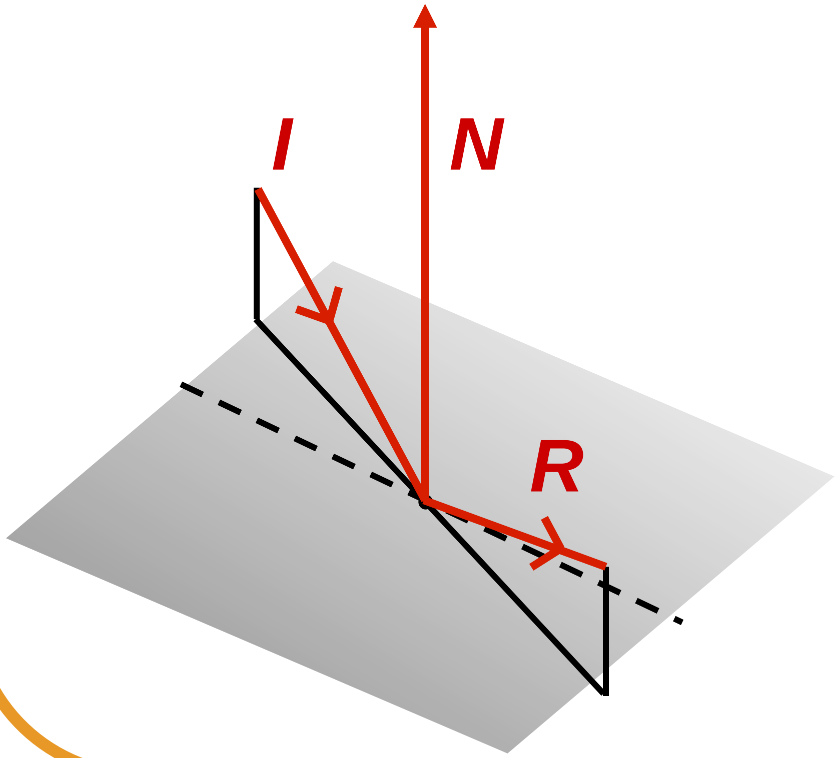
Intersection calculation

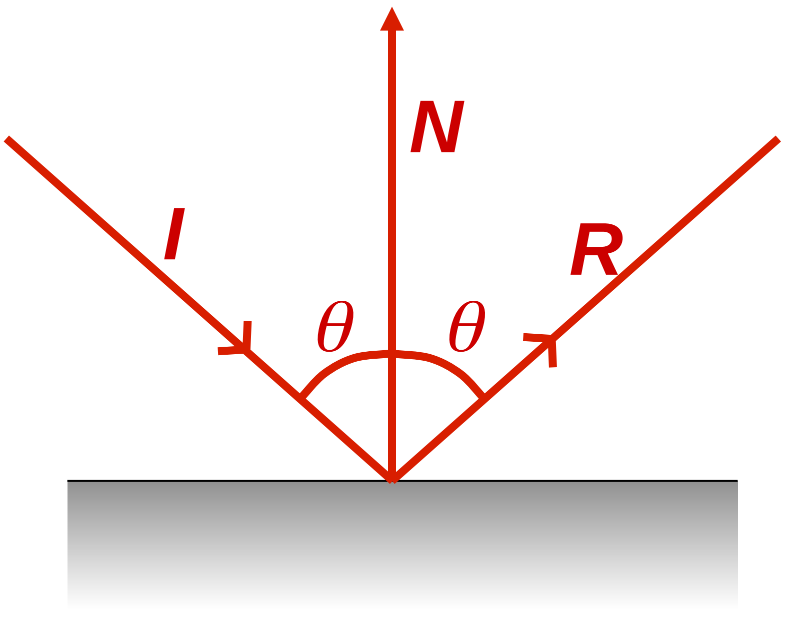
Other intersections

- Ray-Quadric: Similar process as with spheres, replace ray equations in quadric equation
- Bicubic surfaces:
  - Successive divisions methods
  - Analytic methods

Reflection

- Reflection direction: $\boldsymbol{R} = \boldsymbol{I} + 2\boldsymbol{N}\cos\theta$



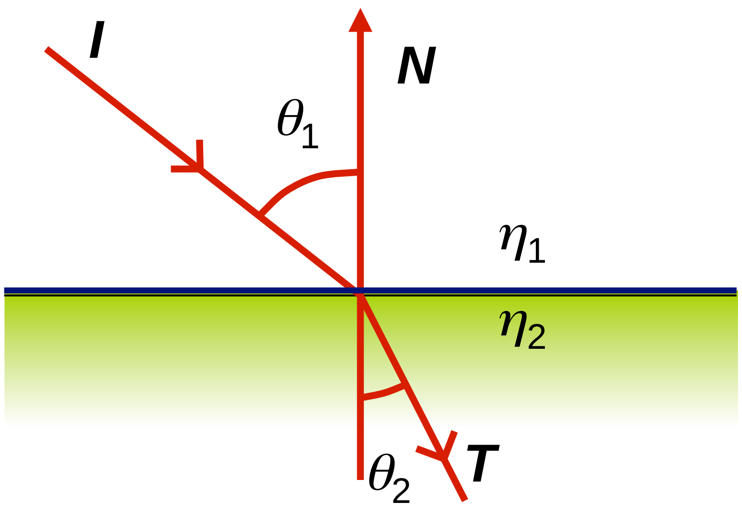


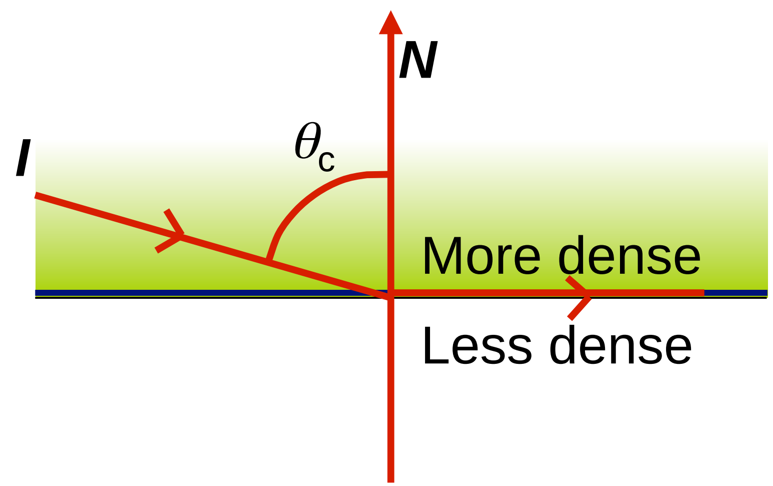
Refraction

- Refraction direction:

$$\boldsymbol{T} = \frac{\eta_1}{\eta_2}\boldsymbol{I} - \left(\cos\theta_2 - \frac{\eta_1}{\eta_2}\cos\theta_1\right)\boldsymbol{N}$$

$$\cos\theta_2 = \sqrt{1 - (\eta_1/\eta_2)^2(1 - \cos^2\theta_1)}$$



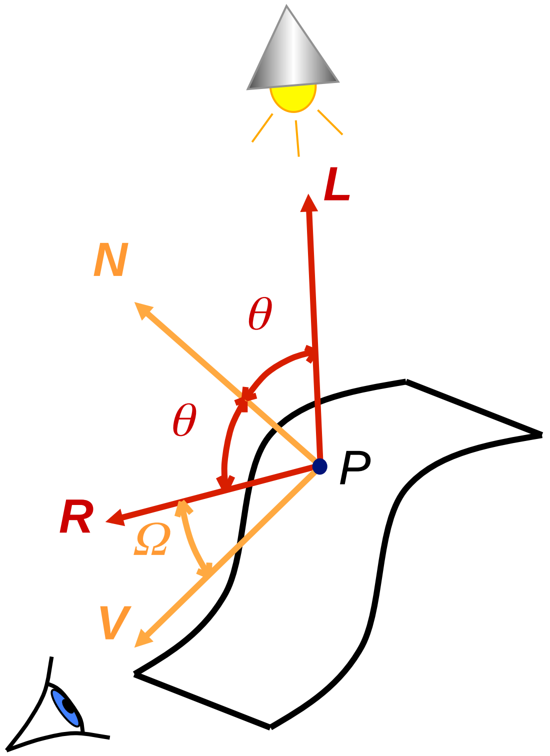


Local illumination

- The most frequent reflection model is Phong:

$$I_{local} = I_{ambient} + I_{difusse} + I_{specular}$$

$$I_{Phong} = \quad I_a k_a(\lambda) + I_d k_d(\lambda)(\boldsymbol{L \cdot N}) + I_e k_e(\lambda)(\boldsymbol{N \cdot H})^n$$



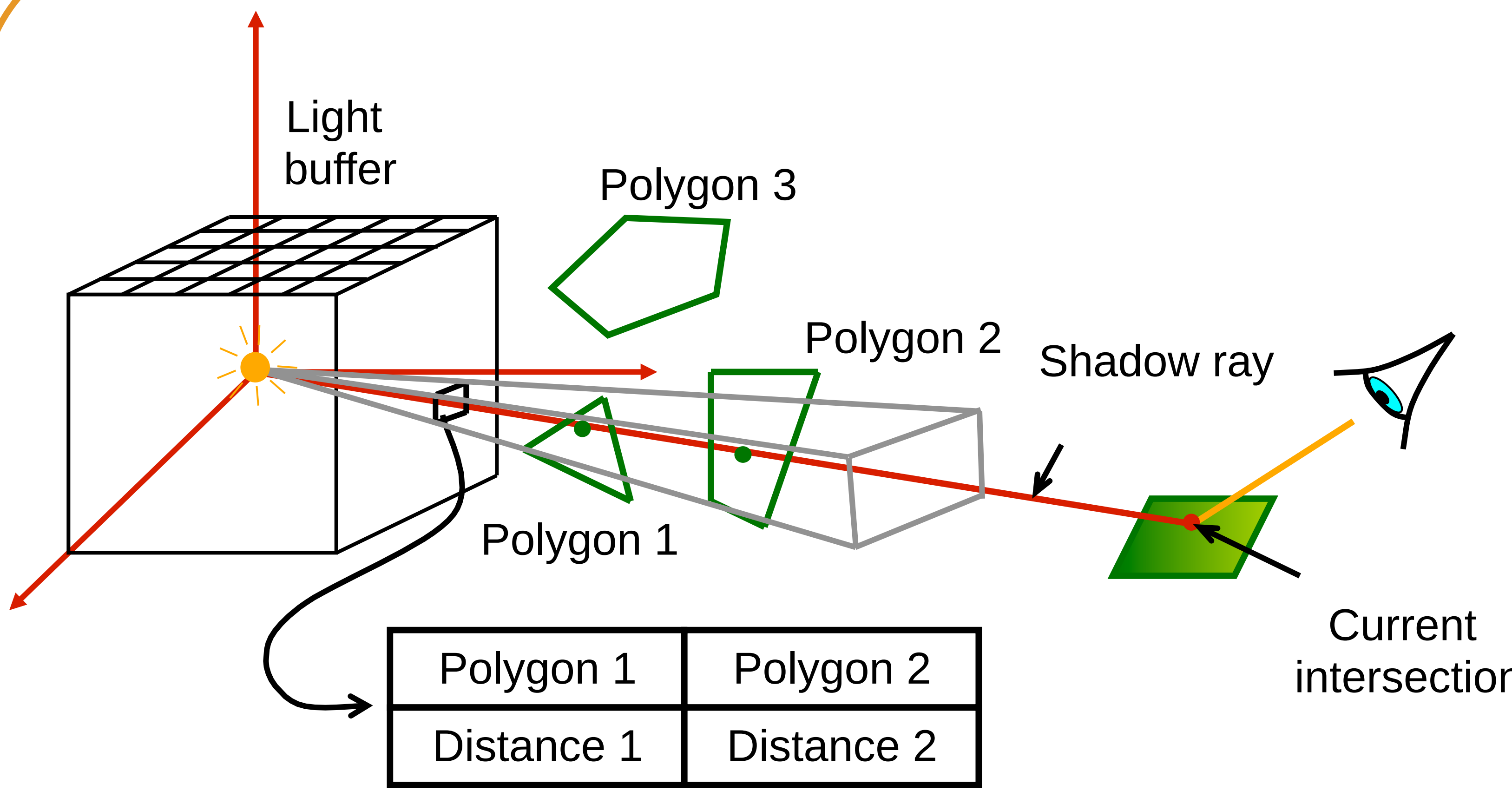
Local illumination

- Any local reflection model can be used. Usually it is Phong but it can be any other (Blinn, Cook&Torrance …)
- Using these models implies some contradictions
  - Ambient light (global) is considered in the local model
  - In the local model, reflections are blurred
  - In the global model, reflections are sharp

Shadows

- Shadow ray or shadow feeler: it is an additional ray from the object to the light source
- If any opaque object blocks the ray, this point is not illuminated
- If any semitransparent object blocks the ray, the light must be attenuated
- Improvement → ***Light buffer***

Light buffer



Polygon 1	Polygon 2
Distance 1	Distance 2

Basic method properties

- Basic method restrictions
  - High cost intersections
  - Sharp reflections and shadows
  - Aliasing
- Some improvements
  - Efficiency improvement
  - Antialiasing
  - Distributed ray tracing

Efficiency improvement

- Reducing the number of rays
  - Adaptive depth control
- Reducing the cost of calculating the intersections
  - First-hit speedup
  - Bounding volumes
  - Spatial coherence

Efficiency improvement
Adaptive depth control

- In the basic algorithm, a branch in the recursions tree is cut when:
  - It reaches a maximum depth
  - An opaque non-reflecting surface is reached (e.g. the background)
- The tree depth can be adapted to the object properties
- Rays are attenuated as they go through the scene, depending on:
  - Reflection coefficient k_{refl}
  - Refraction coefficient k_{refr}
  - Transmission coefficient k_{trans} and distance
- The ray contribution can be considered insignificant if it does not reach a given threshold

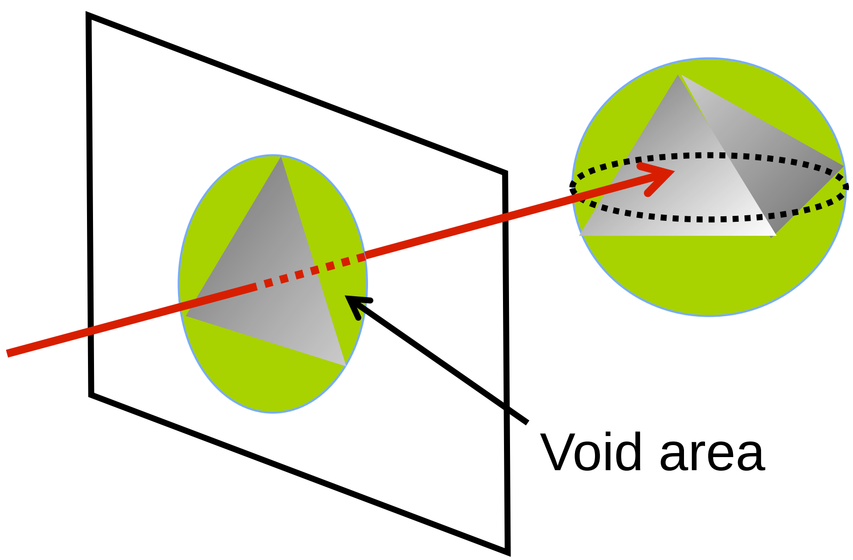
Efficiency improvement
First-hit speedup

- First hit (intersection) is always calculated
- The method cost can be improved if the first intersection is pre-calculated
- It is based on a Z-Buffer algorithm modification:
  - As well as depth, a reference to the nearest object is also stored in the Z-Buffer
  - This way, the first intersection is calculated

    $\underbrace{(x,y,}_{\text{pixel position}}\underbrace{z)}_{\text{Z-Buffer}}$, and the corresponding object can be obtained.

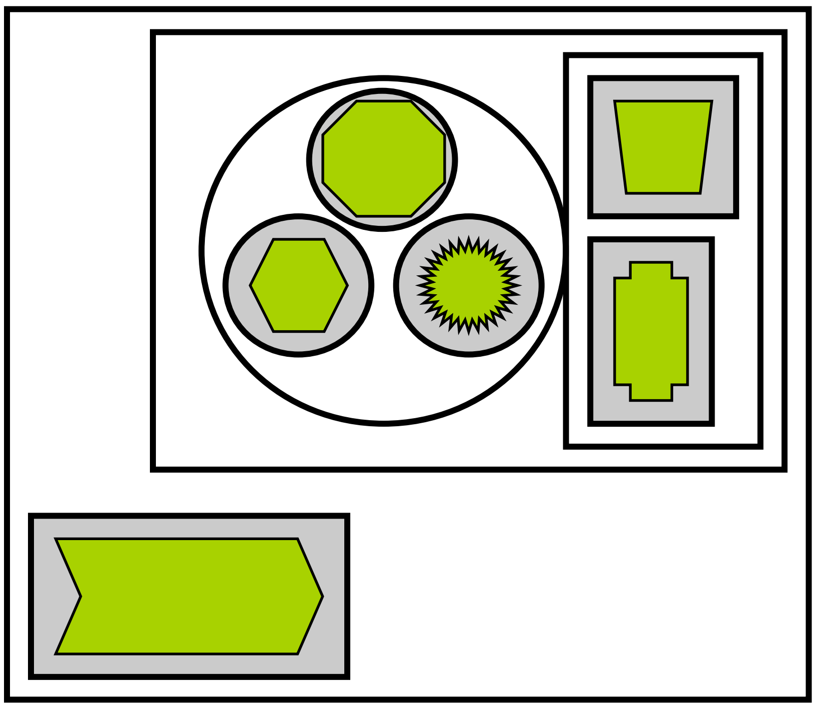
Efficiency improvement
Bounding volumes

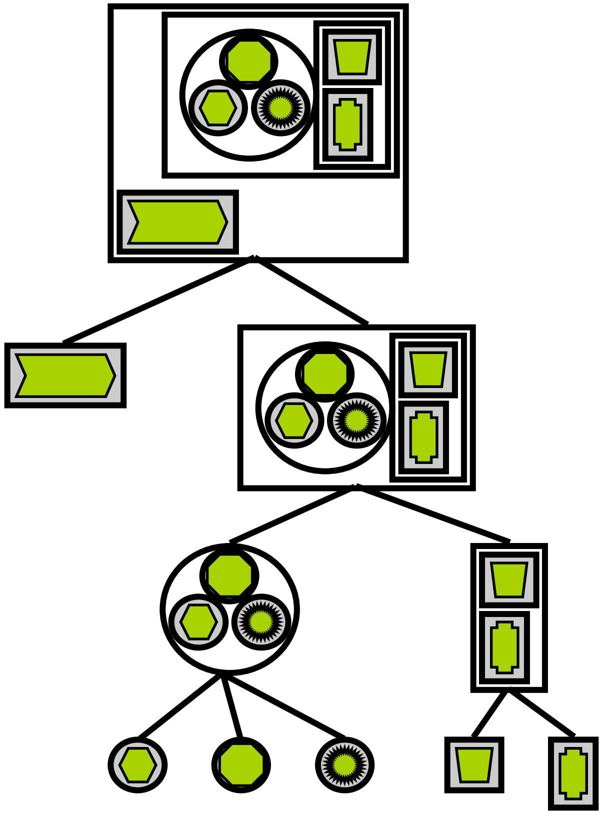
- Every object is bounded by a simple volume
- Bounding volumes features:
  - Simple intersection calculation
  - Void (or empty) area minimization



Efficiency improvement
Bounding volumes

- A hierarchy can be made using a tree

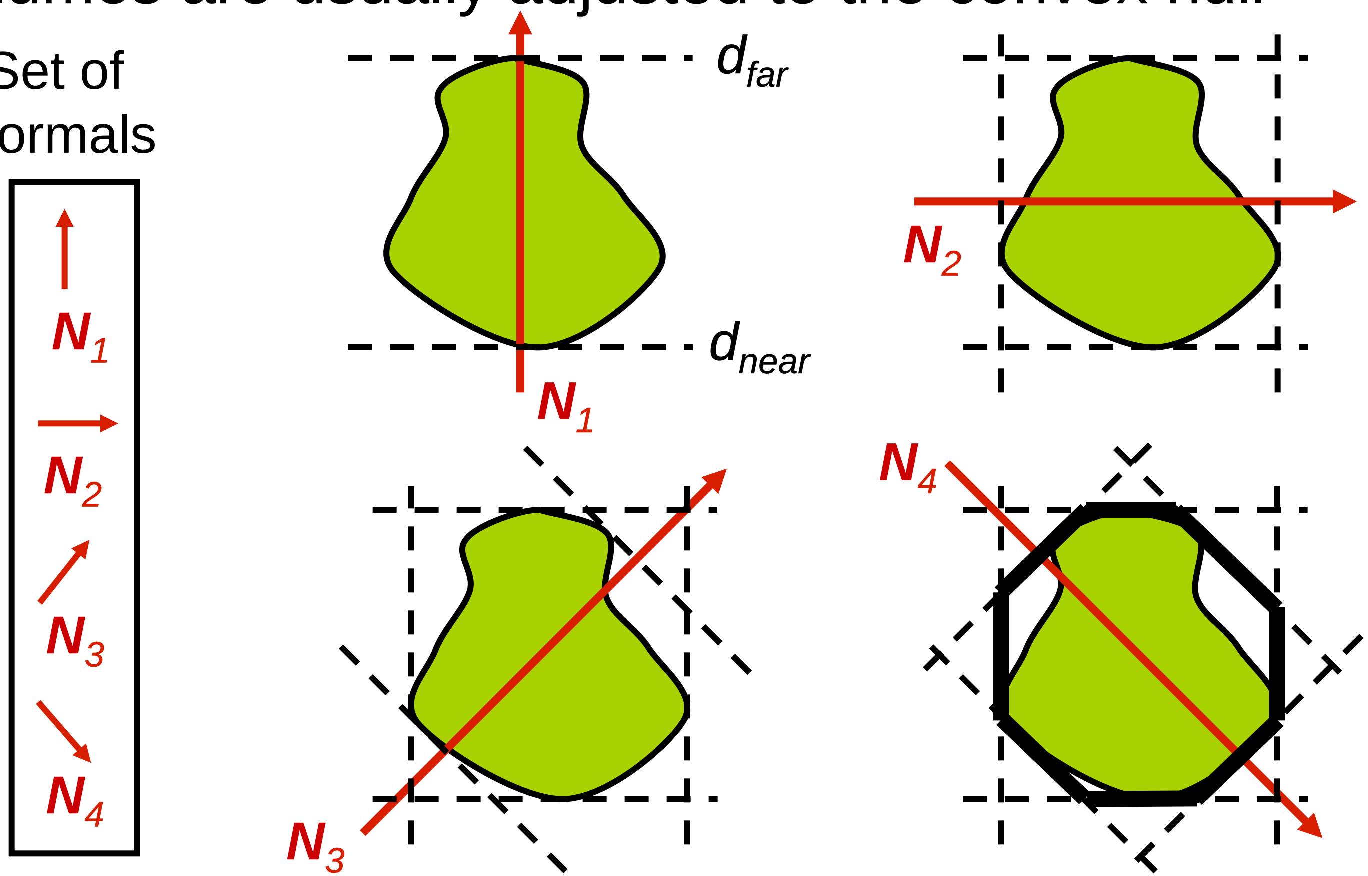




Efficiency improvement
Bounding volumes

- Volumes are usually adjusted to the convex hull

Set of normals

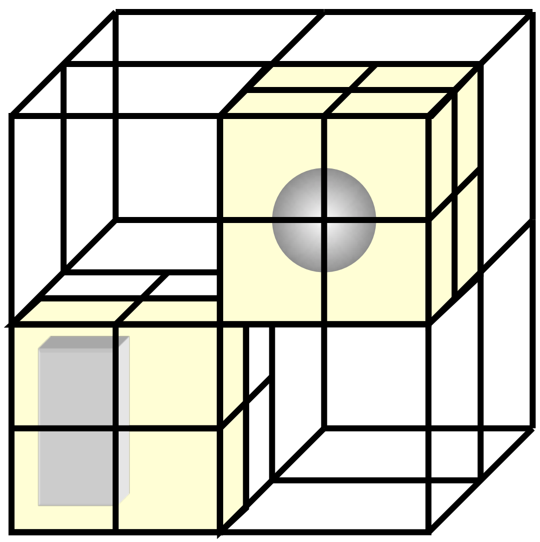


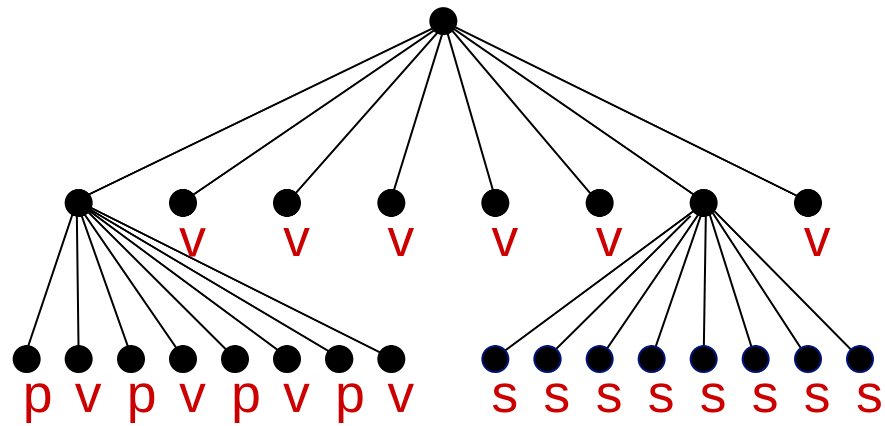
Efficiency improvement
Spatial coherence

- The space is divided into regions
- The intersections are calculated only for the objects inside the region where the ray is
- Advantages
  - It dramatically decreases the cost
  - It introduces an object order
  - Subdivision in pre-process time → Rendering is not penalized
  - Constant rendering time → it depends on precision instead of scene complexity

Efficiency improvement
Spatial coherence

- Octrees (octal trees)

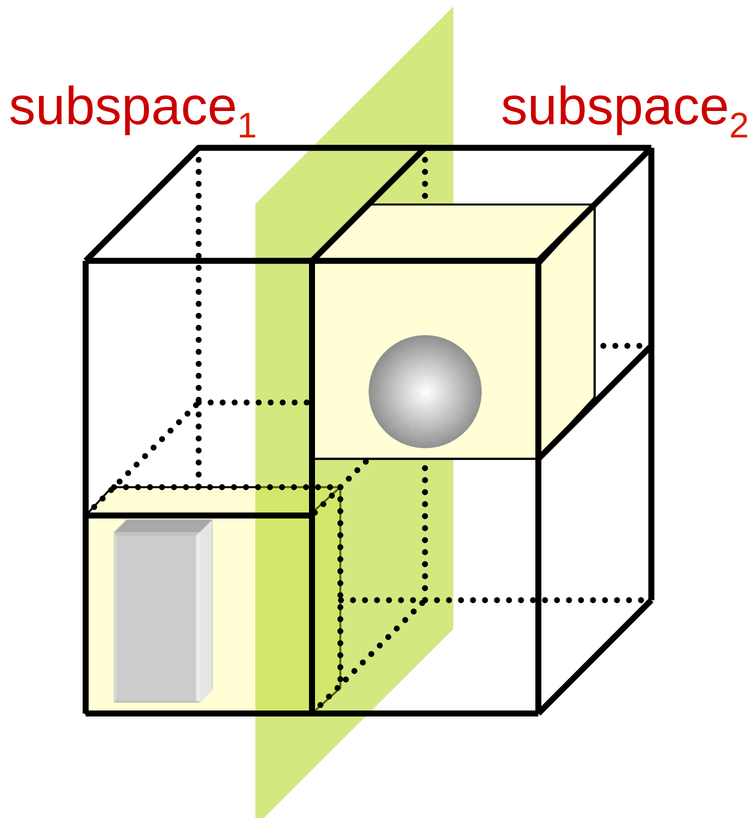


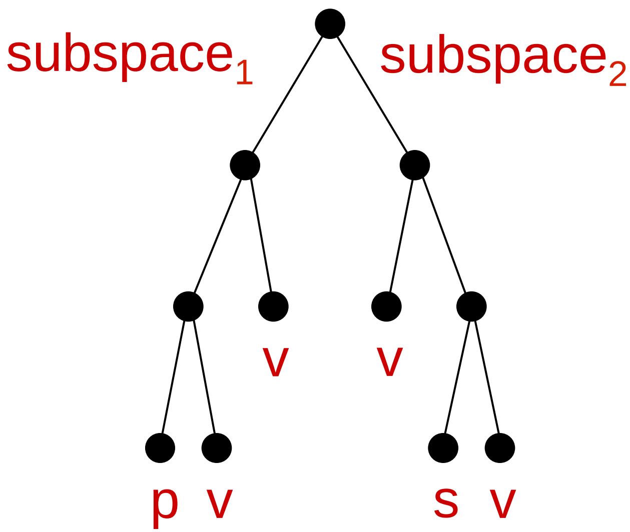


v: void
p: parallelepiped
s: sphere

Efficiency improvement
Spatial coherence

- BSP trees (binary space partitioning)





v: void
p: parallelepiped
s: sphere

Anti-aliasing

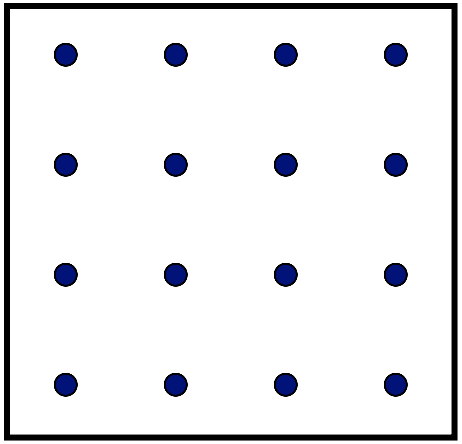
- Aliasing: phenomenon produced by equally spaced sampling of continuous information
- Some improvements
  - Super-sampling
    - Simple sampling: more than one sample is obtained for every pixel→ equally spaced super-sampling improves the aspect but does not eliminate the aliasing artefacts
    - Quincunx: the rays (samples) are traced through the pixel corners and are then averaged
    - Three steps anti-aliasing
  - Stochastic sampling
    - Distributed ray tracing

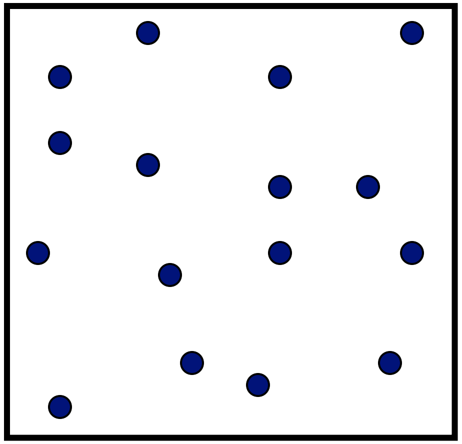
Distributed ray tracing

- Features of distributed ray tracing:
  - Several rays for each pixel (for example, 16)
  - Stochastic distributions of rays
- It is used to:
  - Avoid or improve aliasing artefacts
  - Blurred reflections (glossy objects)
  - Blurred refractions (translucent objects)
  - Penumbra
  - Depth of field
  - Motion blur

Distributed ray tracing

- Stochastic sample using controlled random samples:
  - Poisson distribution with minimum distance restriction
  - Jittering

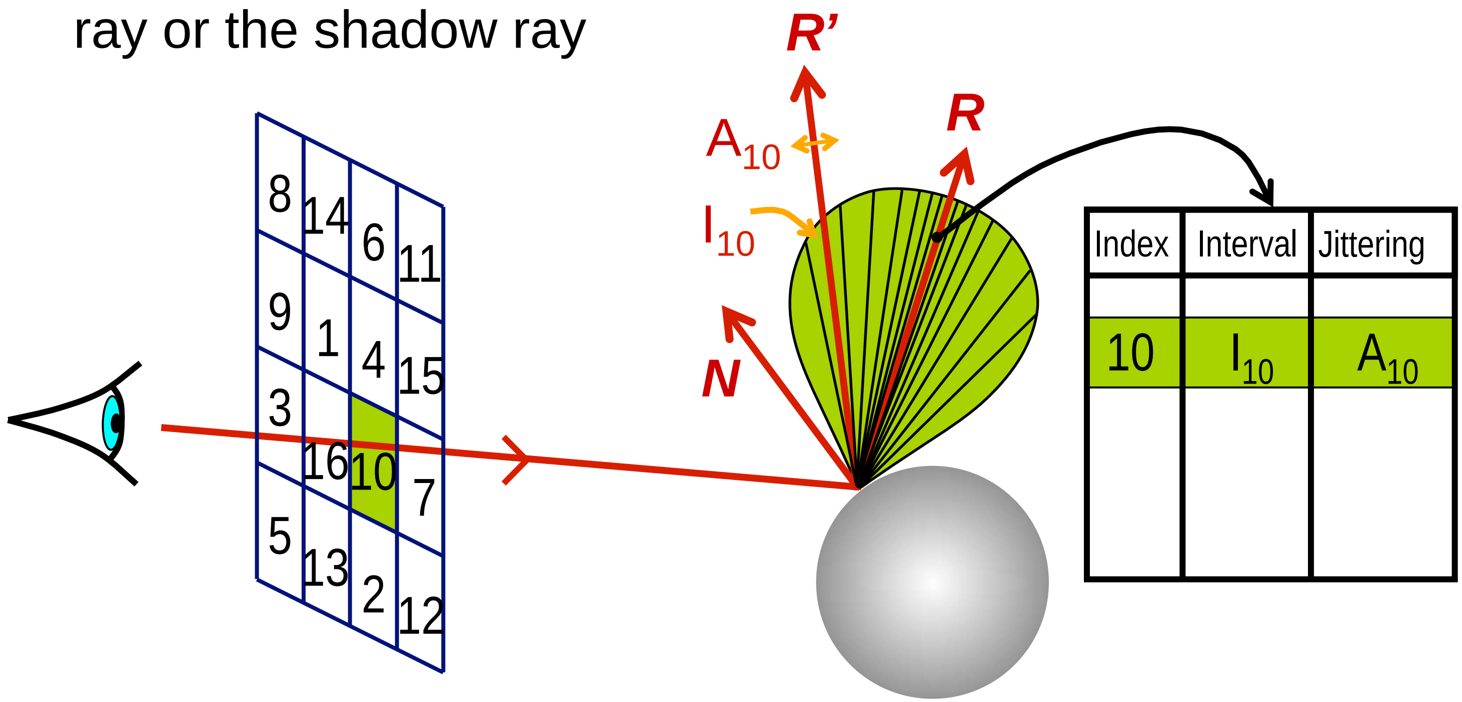




- It improves aliasing problems

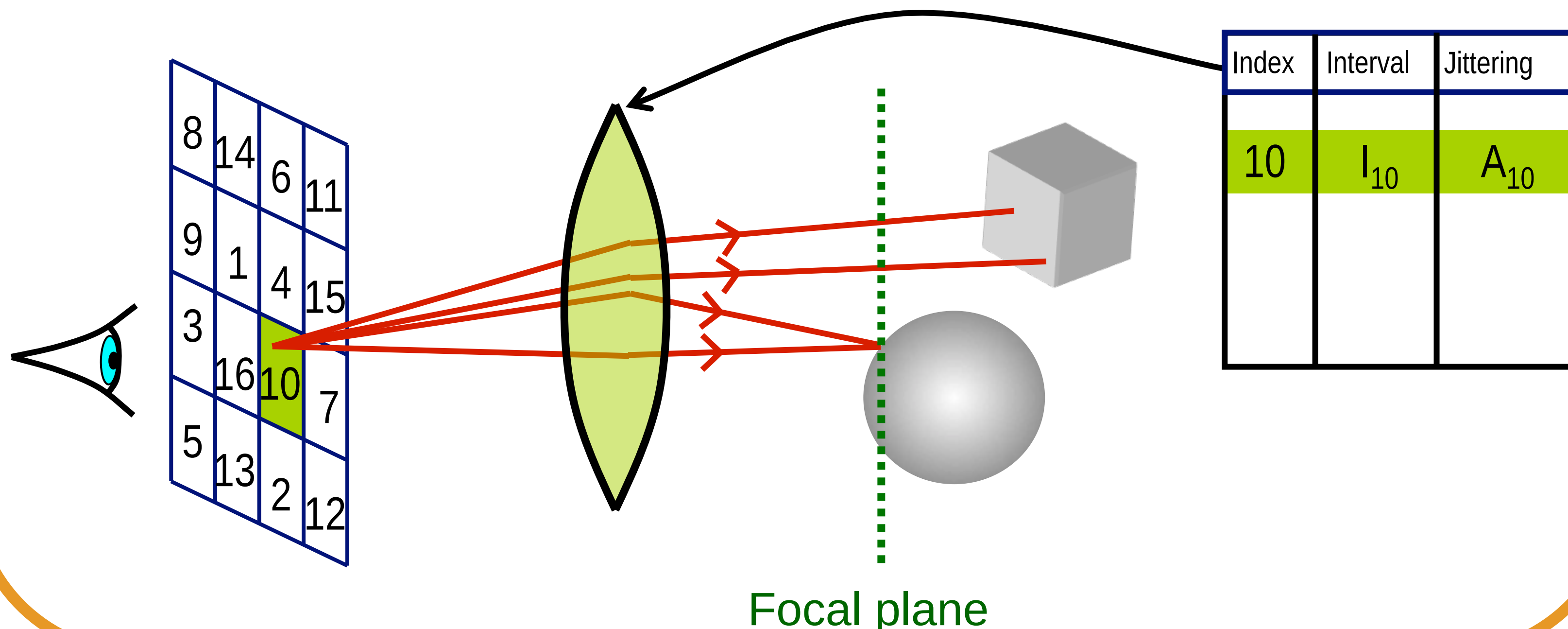
Distributed ray tracing

- Blurred reflections, blurred refractions and penumbra
  - They are produced by rough surfaces and distributed lights
  - Jittering method is applied to the reflected ray, the refracted ray or the shadow ray



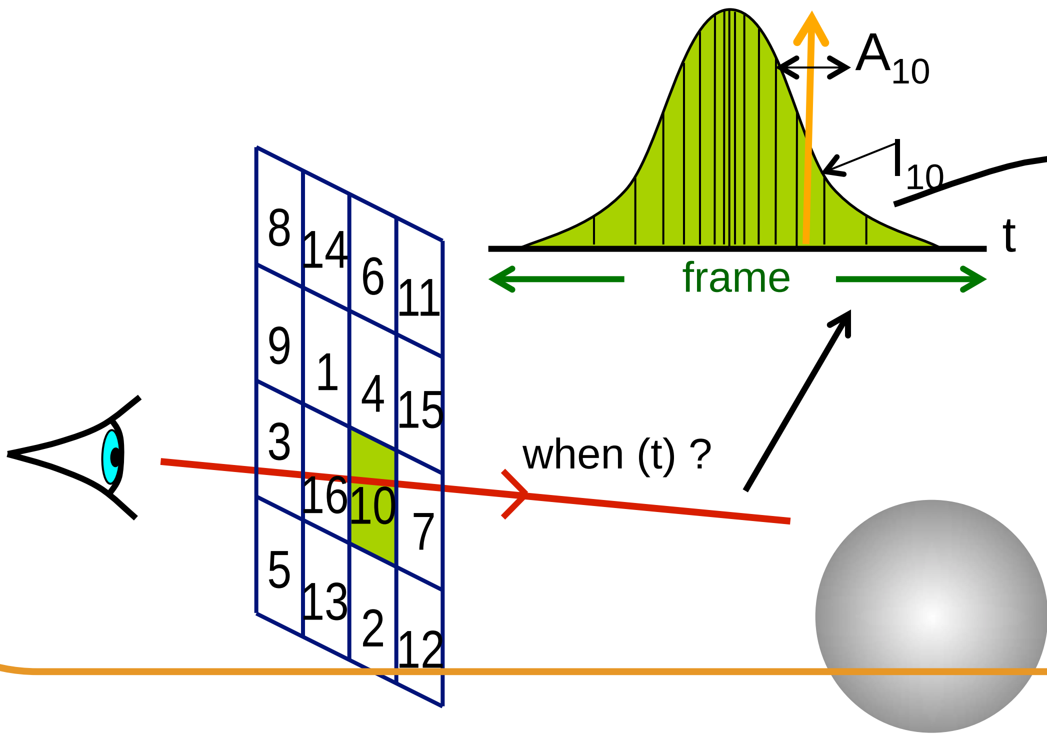
Distributed ray tracing

- Depth of field
  - In a real camera, only the objects on the focal plane are focussed
  - The jittering method is applied to a convex lens



Distributed ray tracing

- Motion blur:
  - Stochastic sample of time
  - Every ray is traced in a time instant during the frame
  - Time instants are stochastically distributed along the frame

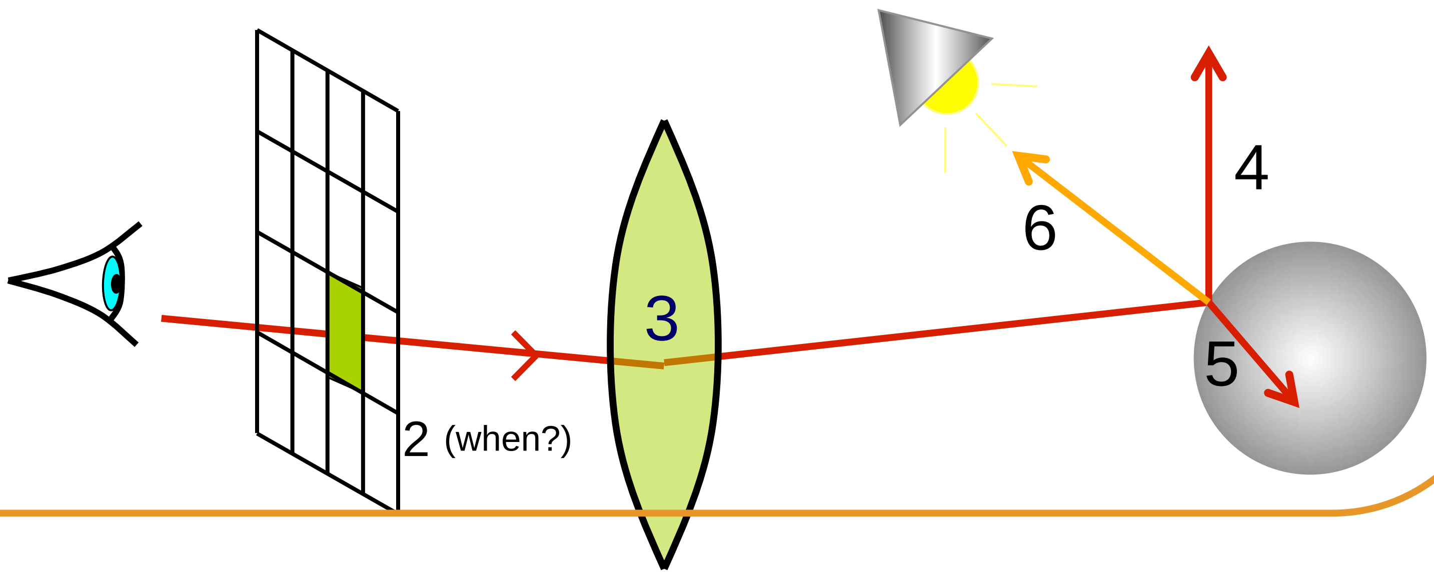


Index	Interval	Jittering
10	I_{10}	A_{10}

Distributed ray tracing

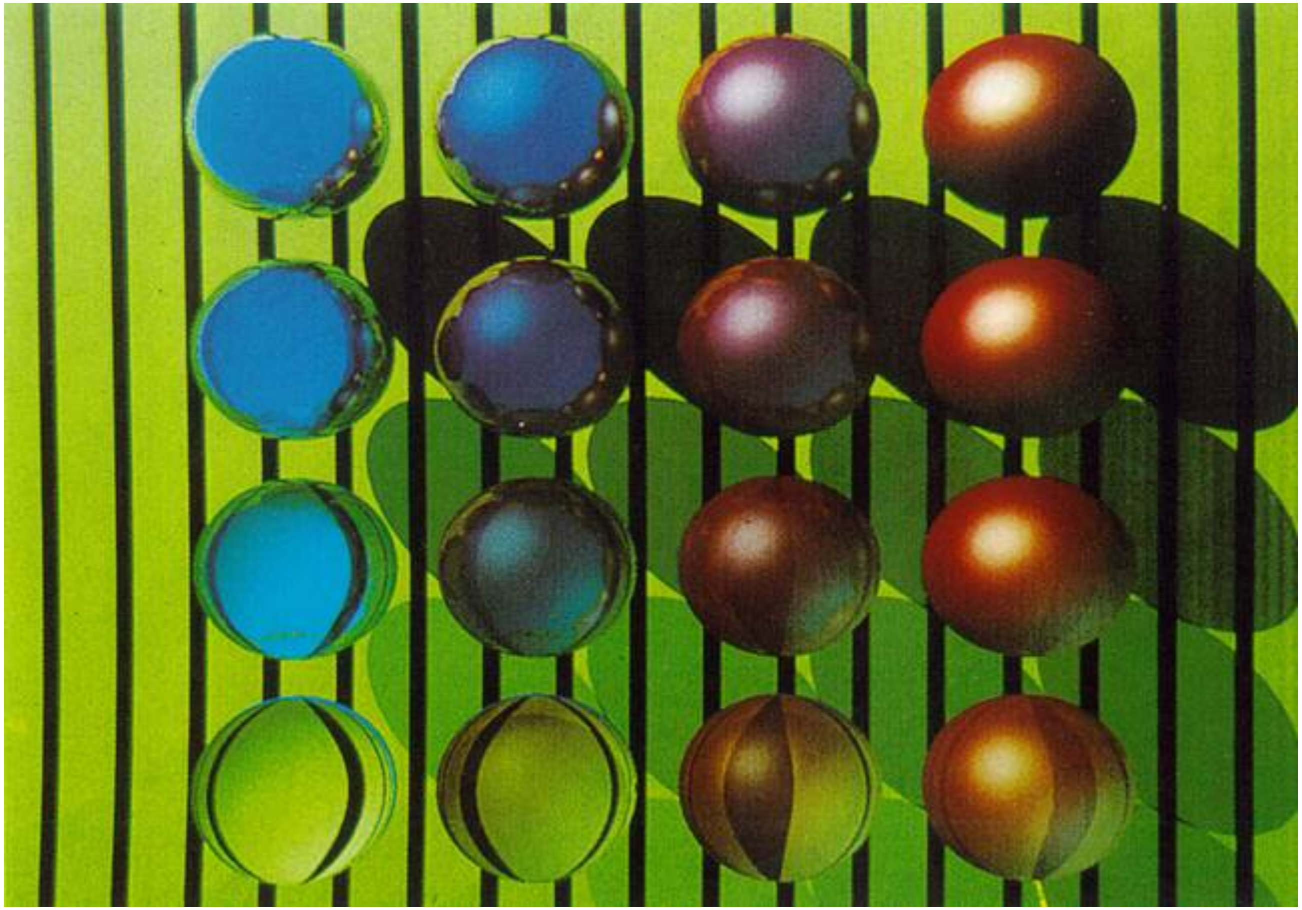
Steps to correctly apply the distributed ray tracing method:

1. Determine ray position using jittering→ **Anti-aliasing**
2. Determine the tracing time for the ray using jittering → **Motion blur**
3. Determine the lens effect using jittering → **Depth of field**
4. Obtain the reflected ray using jittering → **Blurred reflection**
5. Obtain the refracted ray using jittering → **Blurred refraction**
6. Obtain the shadow ray using jittering → **Penumbra**



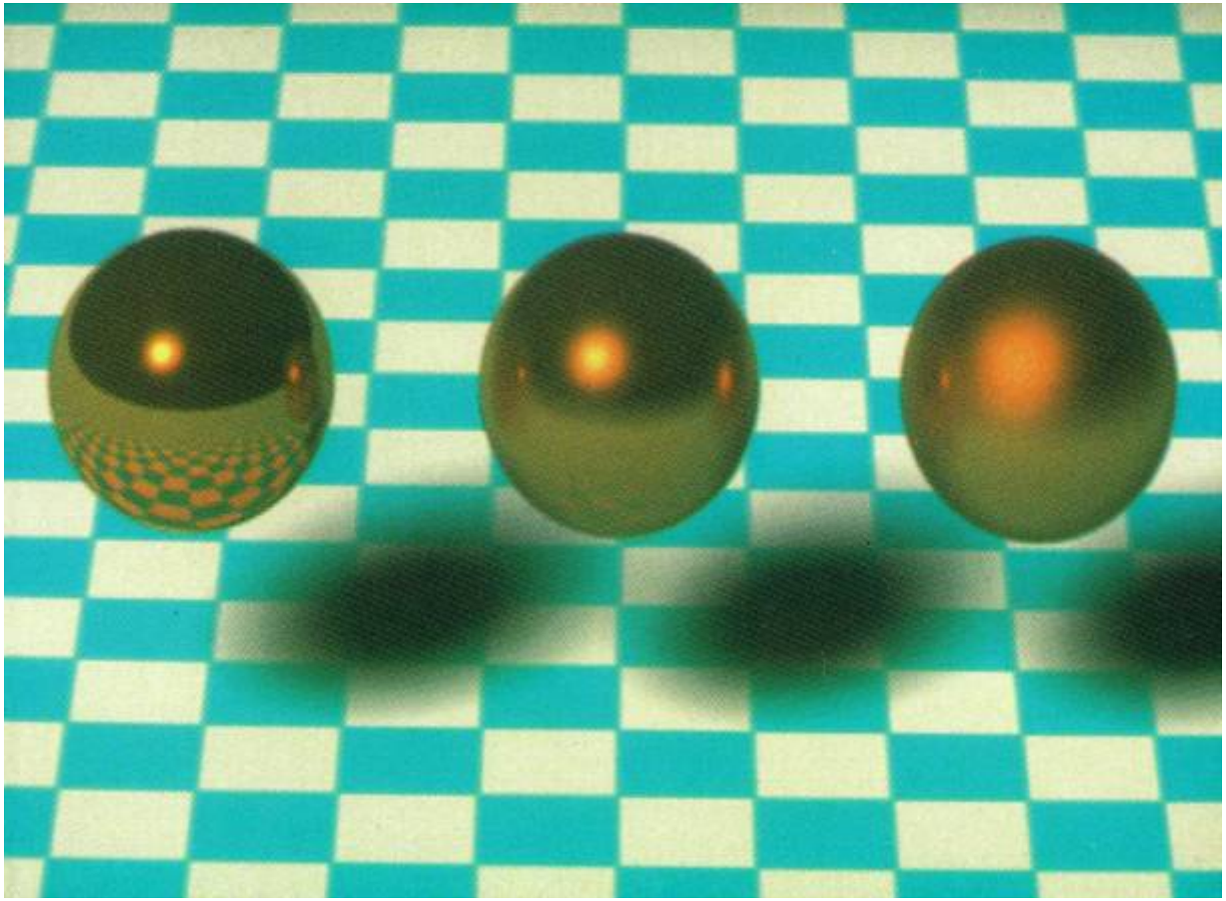
Examples

Effect of changing the contribution of the reflection and refraction components



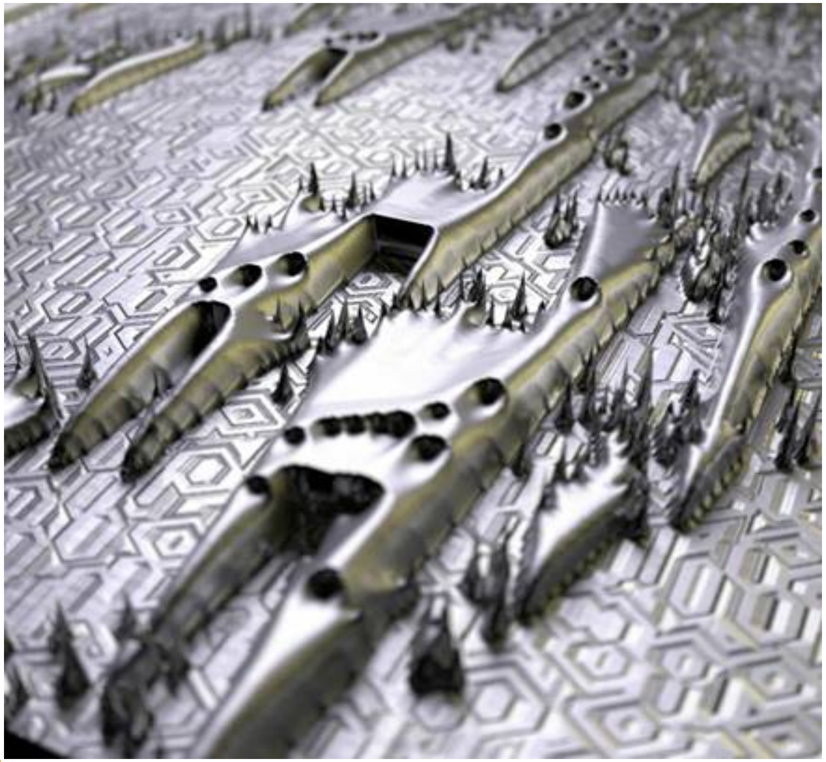
Examples

Effect of distributed ray tracing on blurred reflections and shadows



Examples

Effect of distributed ray tracing on field depth

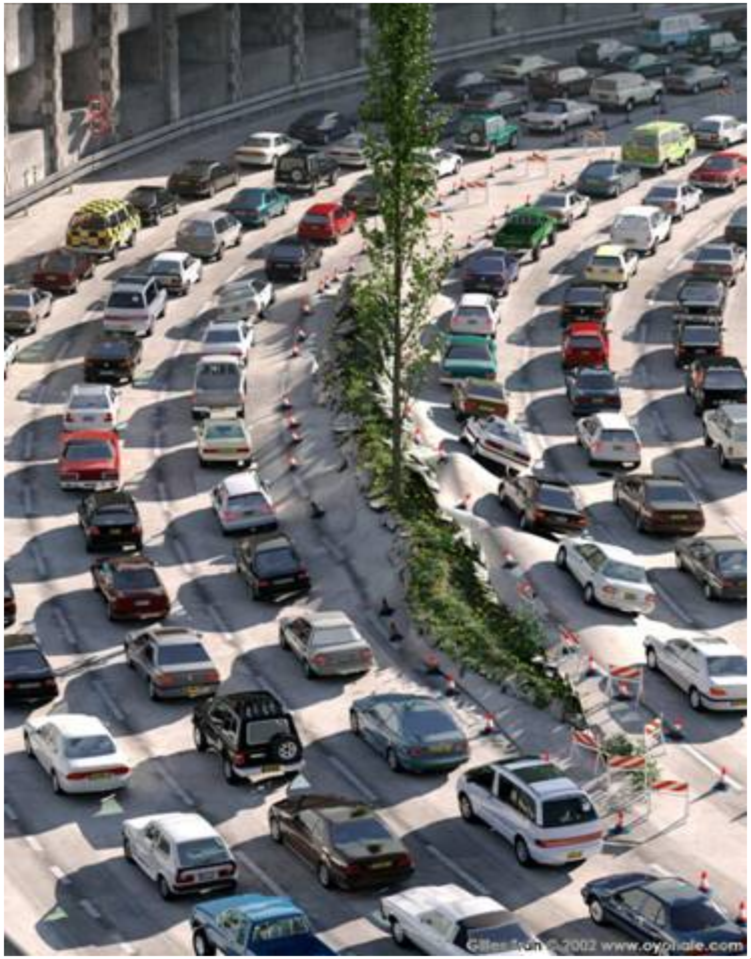


Examples

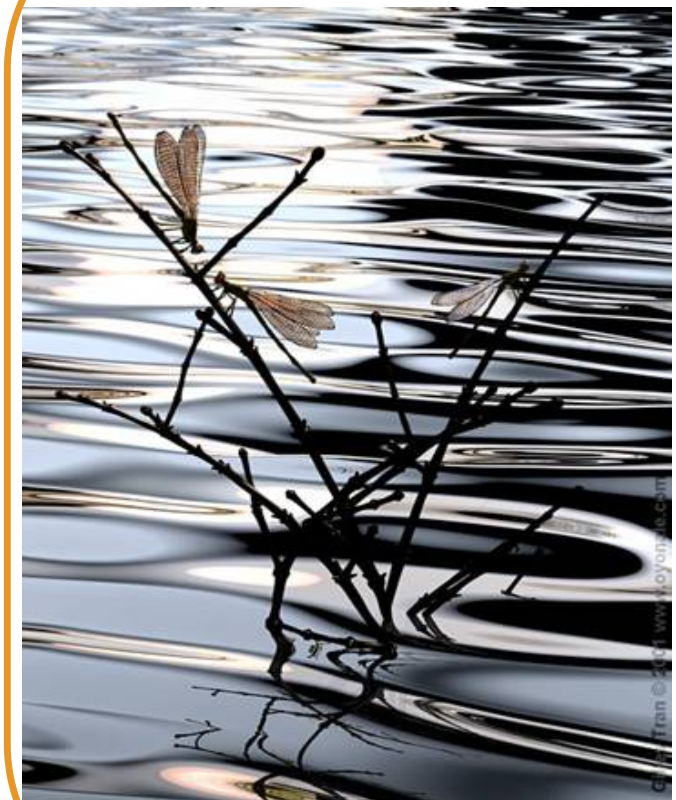
Effect of distributed ray tracing on movement

Examples


Gilles Tran © 2002 www.oyonale.com


Gilles Tran © 2002 www.oyonale.com

Examples



Conclusions

- Features
  - Very realistic
  - Recursive method
  - Global illumination
  - It can render non-polygonal objects (if the intersection ray object can be calculated)
- Restrictions
  - High computational cost
  - It mixes global and local illumination
- High impact improvements
  - Spatial coherence
  - Distributed ray tracing