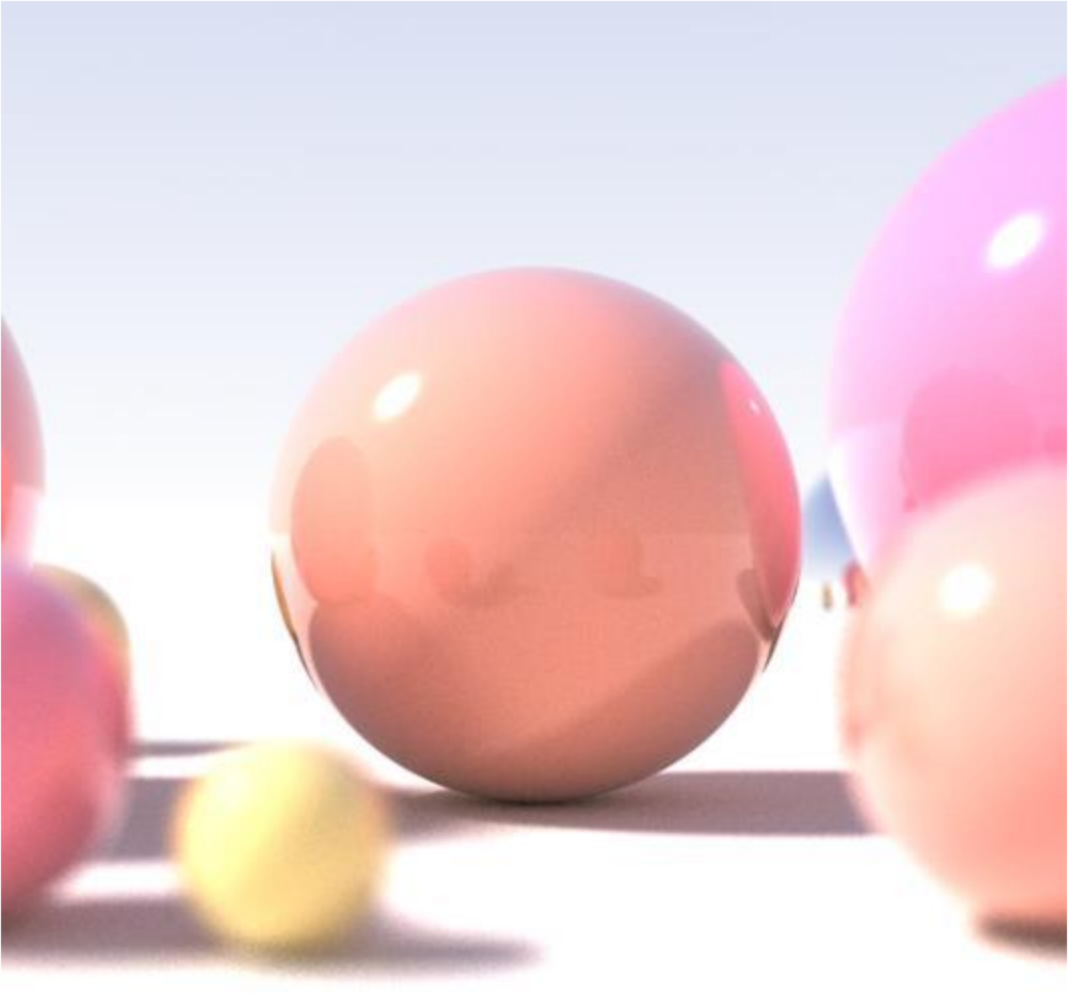
Computer Graphics



Eafit

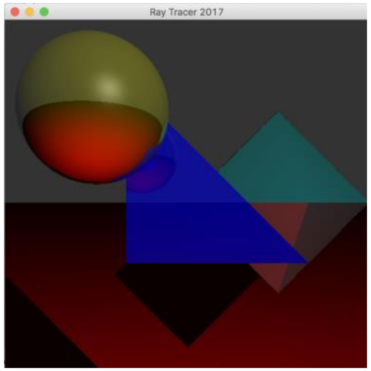
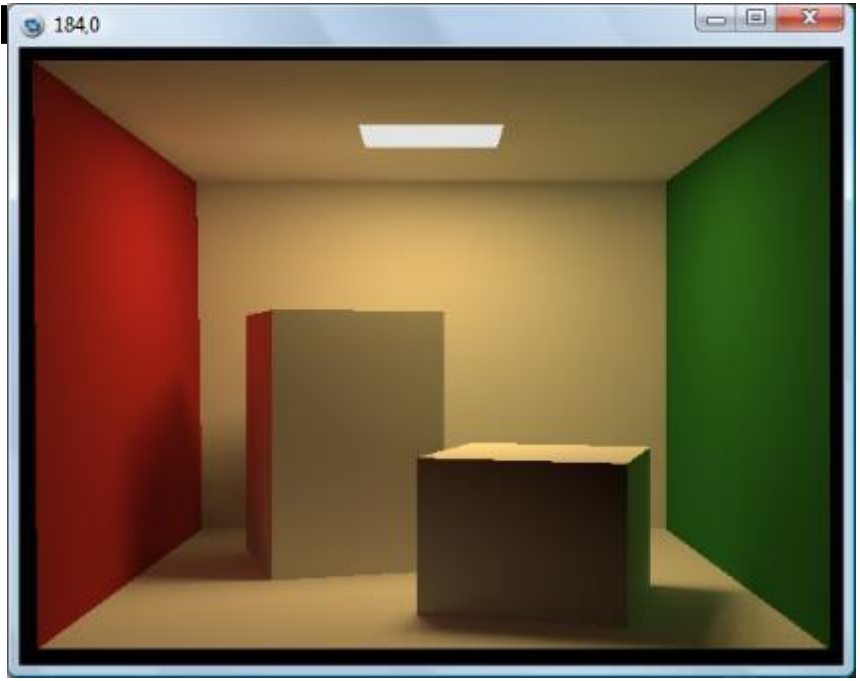


Imagen: http://en.wikipedia.org/wiki/Ray_tracing_(graphics)

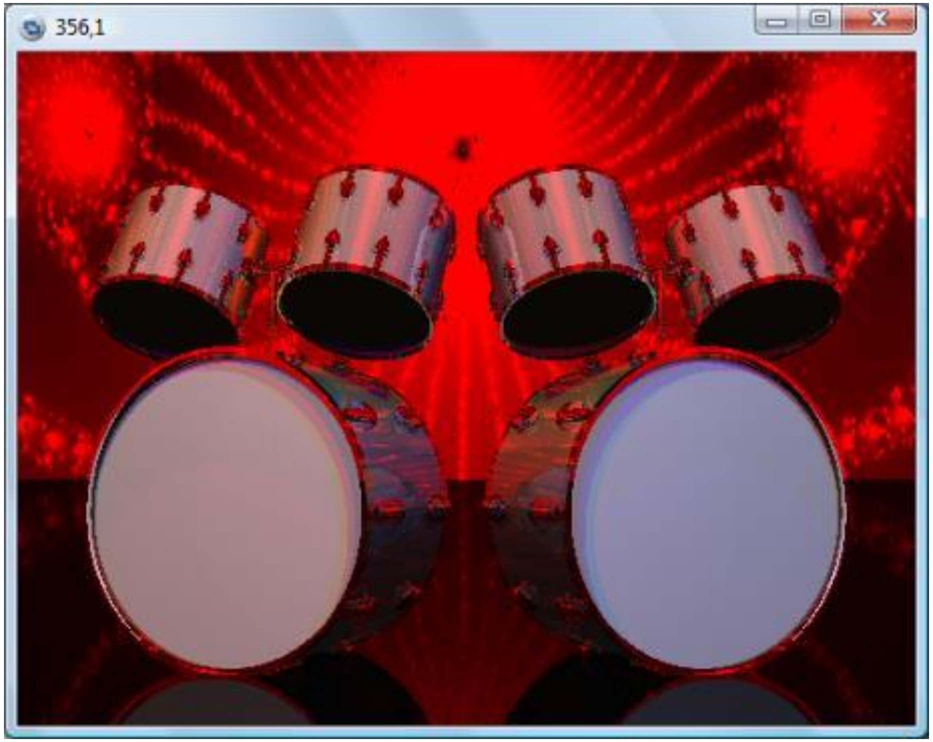
Overwiew

- Limitations of the local illumination model
- Introduction
- The recursive t
- Some examples
- Links



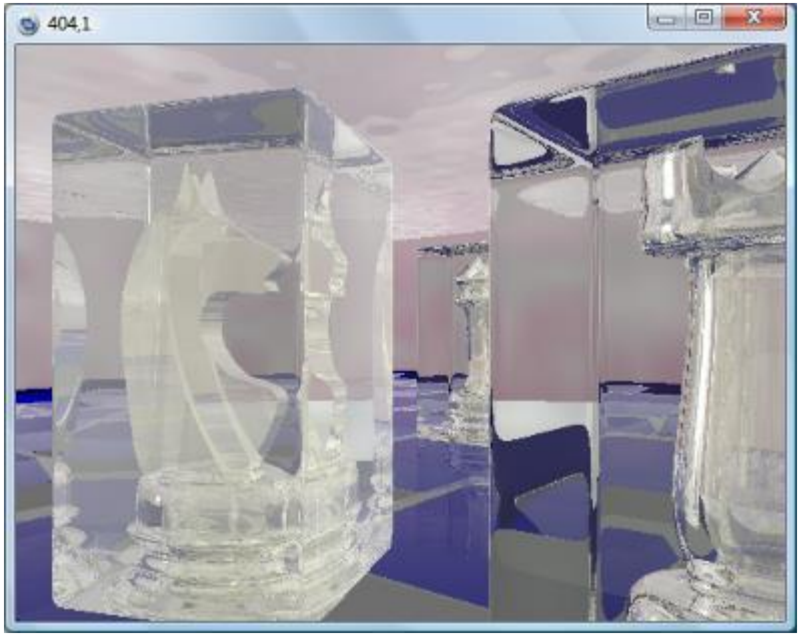
Limitations of local illumination

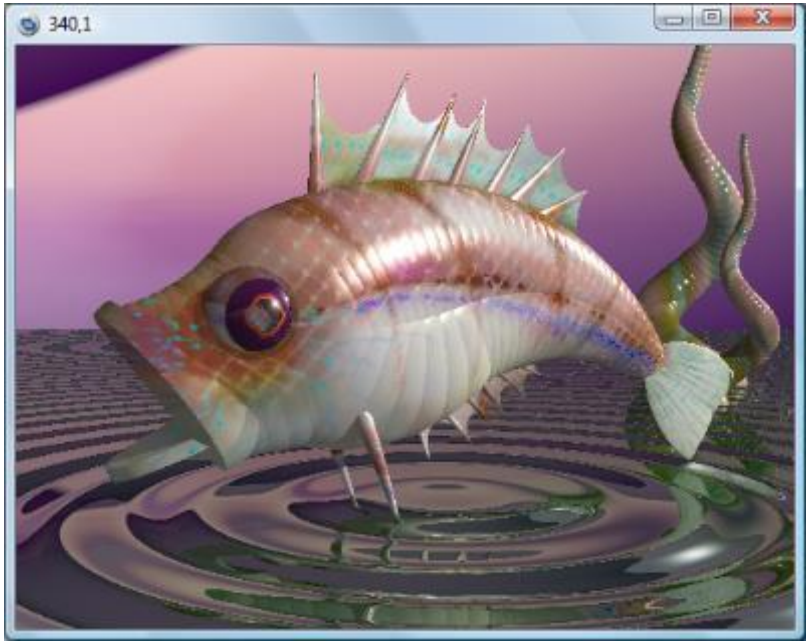
- The local illumination model cannot handle:
  - Shadows
  - Reflection
  - Refraction



Introduction

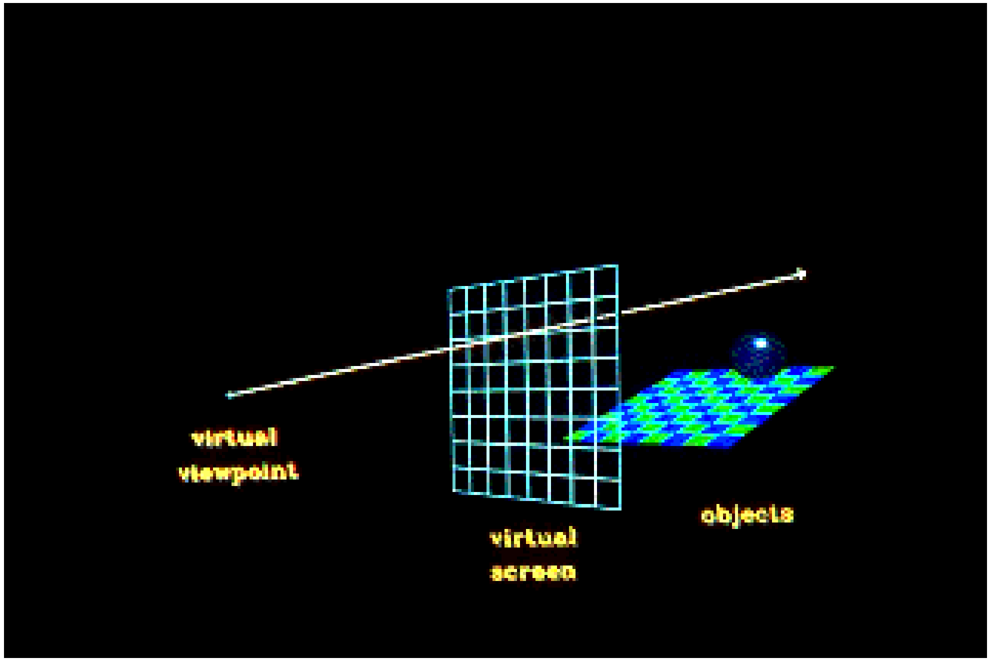
- In real life, photons originate at the light sources, bounce off objects and, finally, reach the oberver’s eye.
- The Ray-Tracing starts at the observer’s eye, is projected through the surface and, posibly, hits objects.





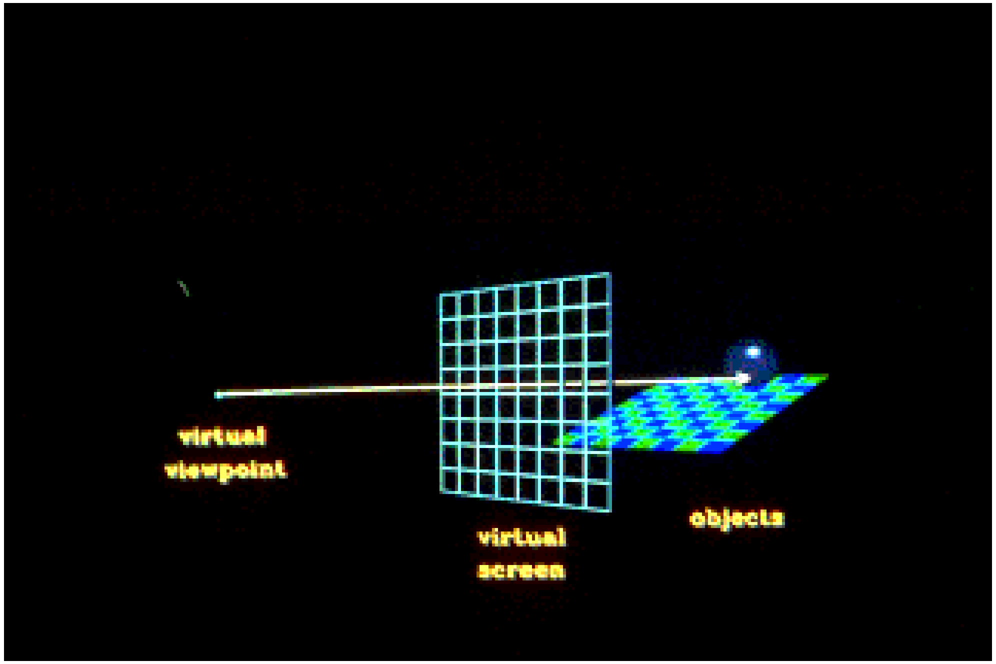
Introduction

- Some rays do not intersect objects
- => they take the background color



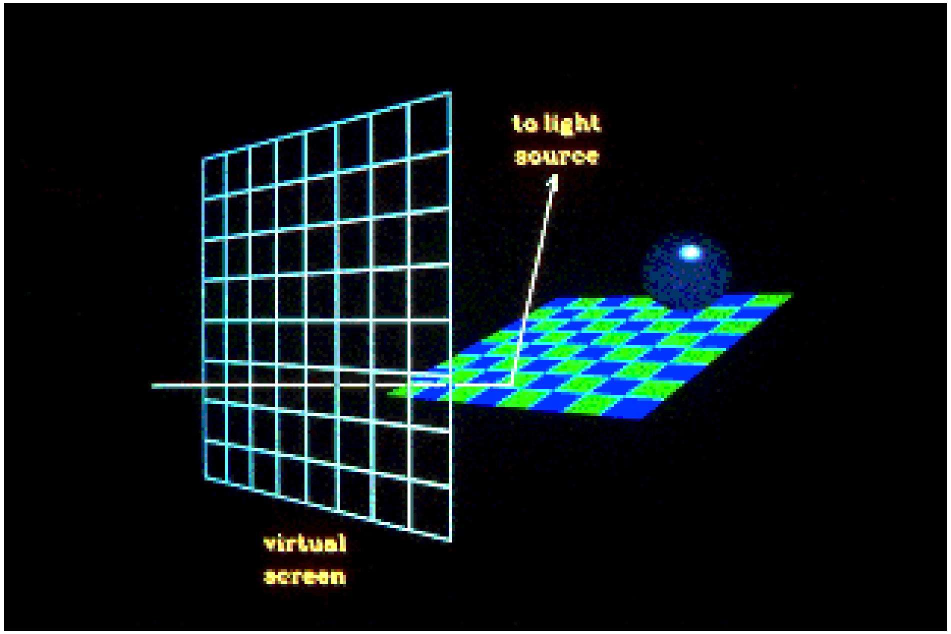
Introduction

- Other rays do hit objects.
- In this case, the local illumination model is used to calculate the color of the pixel.



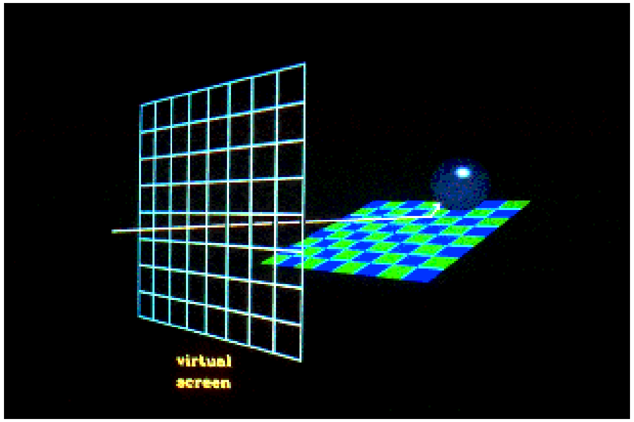
Introduction

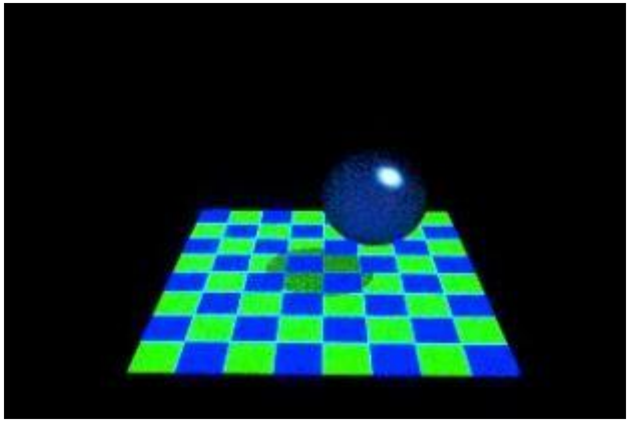
- If the ray hits and object, we want to know if it is in the shadow of another object, with respect to a source of light.



Introduction

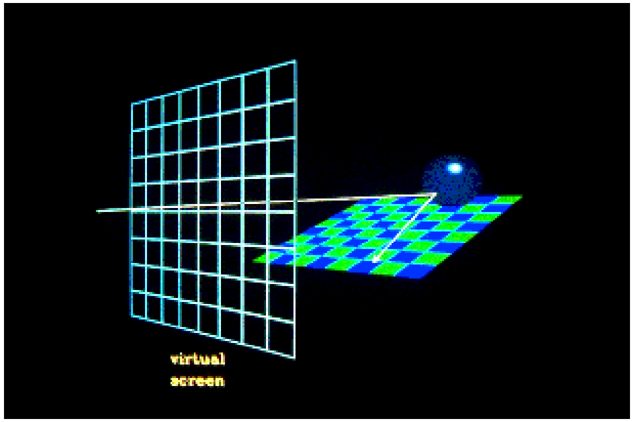
- If the object is in shadow, only ambient light is applied
- Looks like this:

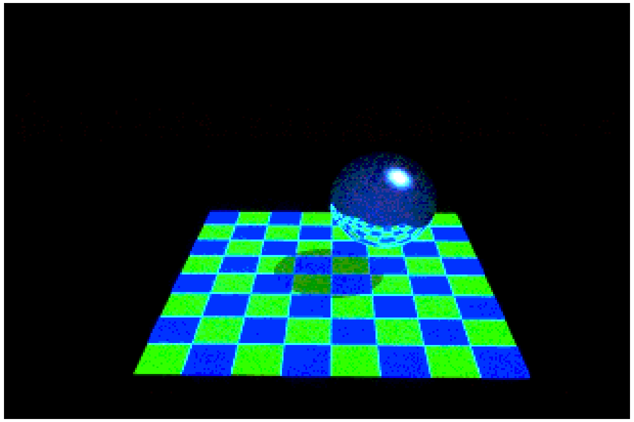




Introduction

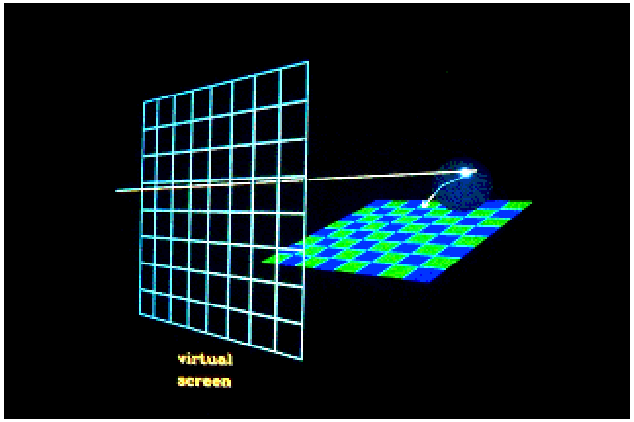
- If the object reflects light, a new "reflection" ray is casted and will pick up the color of some other object.
- Looks like this:

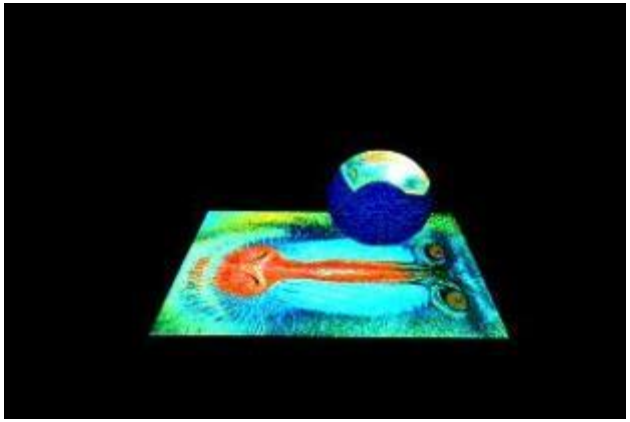




Introduction

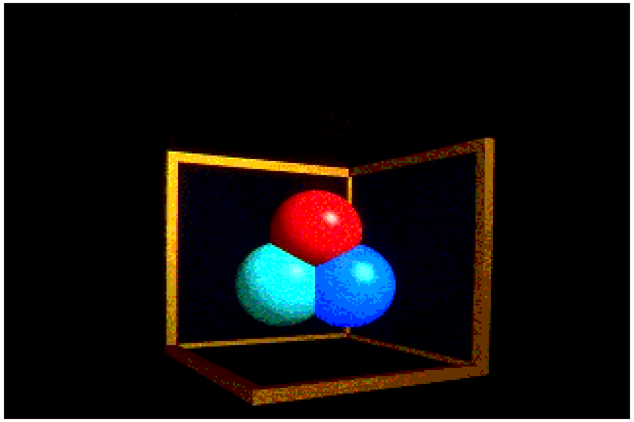
- If the object is transparent, a new "transmission" ray is casted.
- Looks like this.

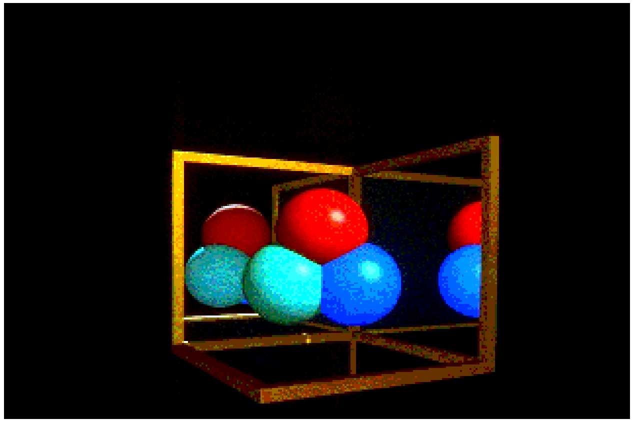




Introduction

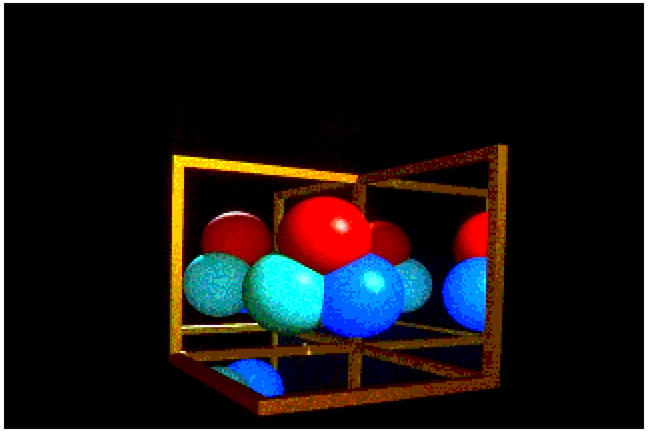
- Limiting the depth of the tree has impact on the result:
- 1
- 2
- …





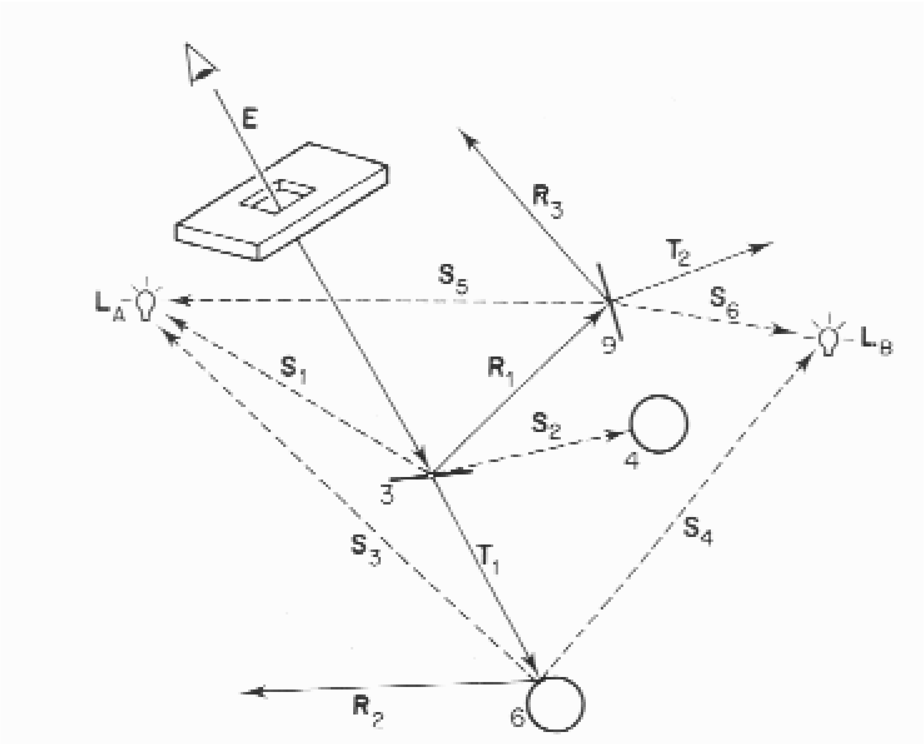
Introduction

- Or more



Recursive Tree

- Rays can be organized in a tree, as follows:
- R: Reflections
- T: Transparencies
- S: Lights or shadows



The recursive Tree

- In an abstract tree:

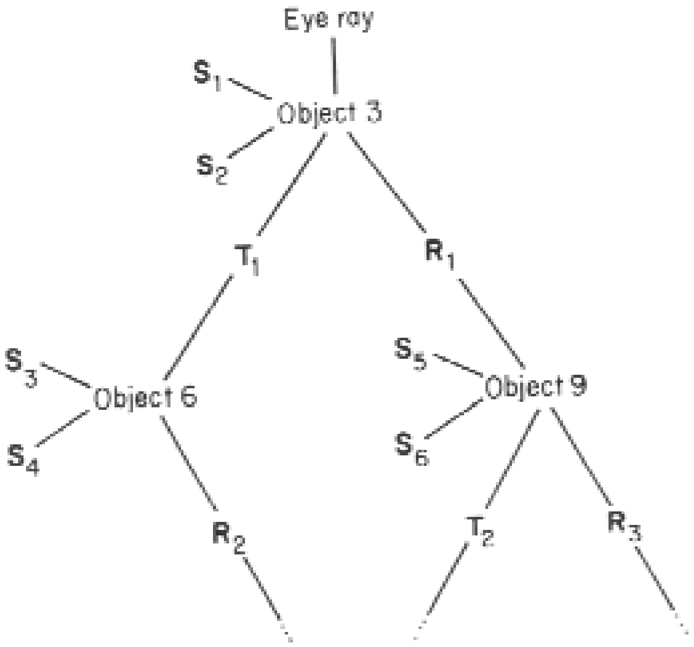


Fig. 12. The ray tree in schematic form.

How to compute the rays?

- Includes the following:
  - Parametric equations from the eye to the scene, passing through each pixel in the scene
  - Intersections
    - Ray - plane
    - Ray - sphere
    - Ray - polygon
    - Ray …
  - Reflection: incoming angle = reflection angle
  - Refraction: Snell's law.
  - More on these tomorrow.

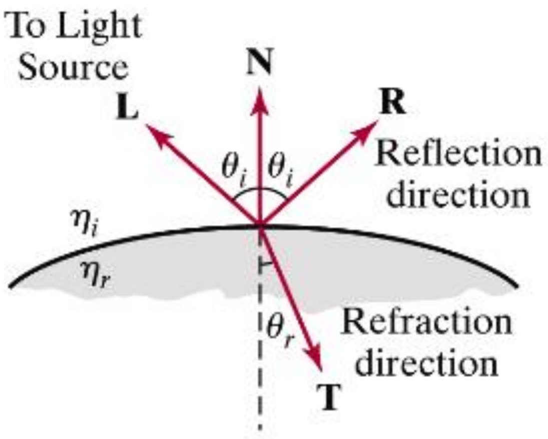


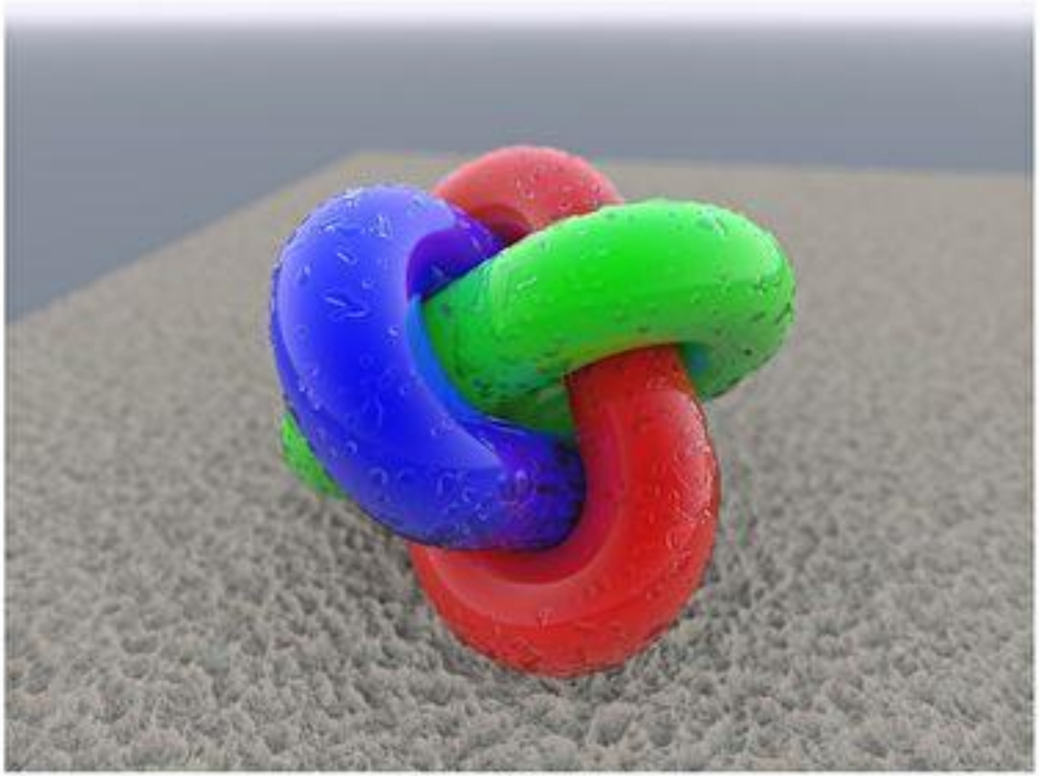
Figure 10-30

Reflection direction **R** and refraction (transmission) direction **T** for a ray of light incident upon a surface with index of refraction η_r.

References

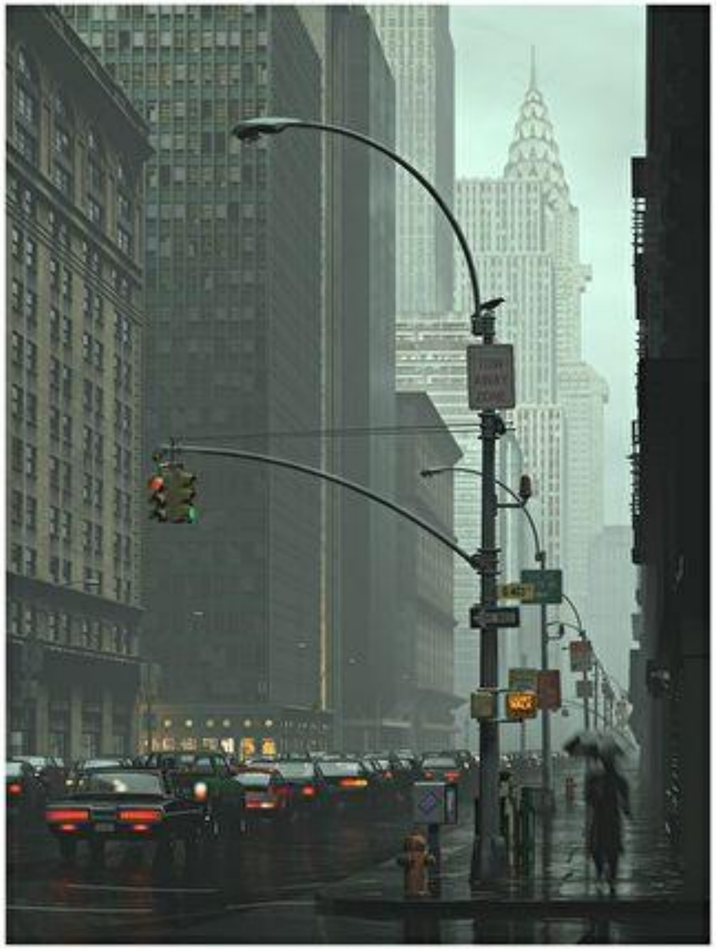
- Free Ray Tracing software: POV Ray (http://www.povray.org/)
- Images taken from Ray-Tracing tutorial at ACM's SIGGRAPH (http://www.siggraph.org/education/materals/HyperGraph/raytrace/rtrace0.htm)
- Other images generated with PovRay.
- Reading: Hearn&Baker, section 10-11.

Images created with PovRay



© 2002 Tekno Frannansa
www.evilsuperbrain.com

Images created with PovRay



L'oiseau mouillé - The wet bird - Gilles Tran © 2000 www.oyonale.com

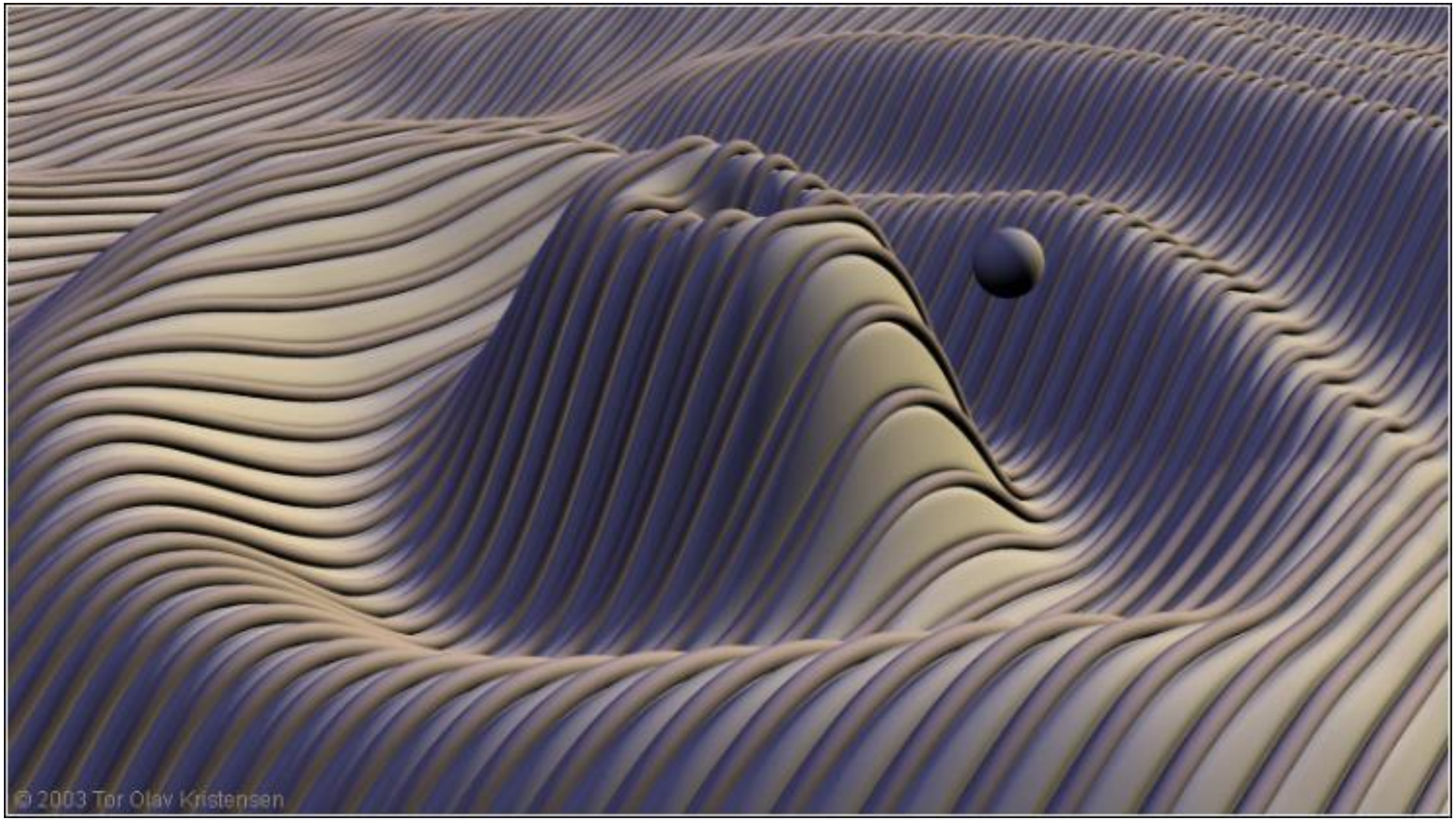
Images created with PovRay

- Christoph Hormann

Christoph Hormann <chris_hormann@gmx.de>

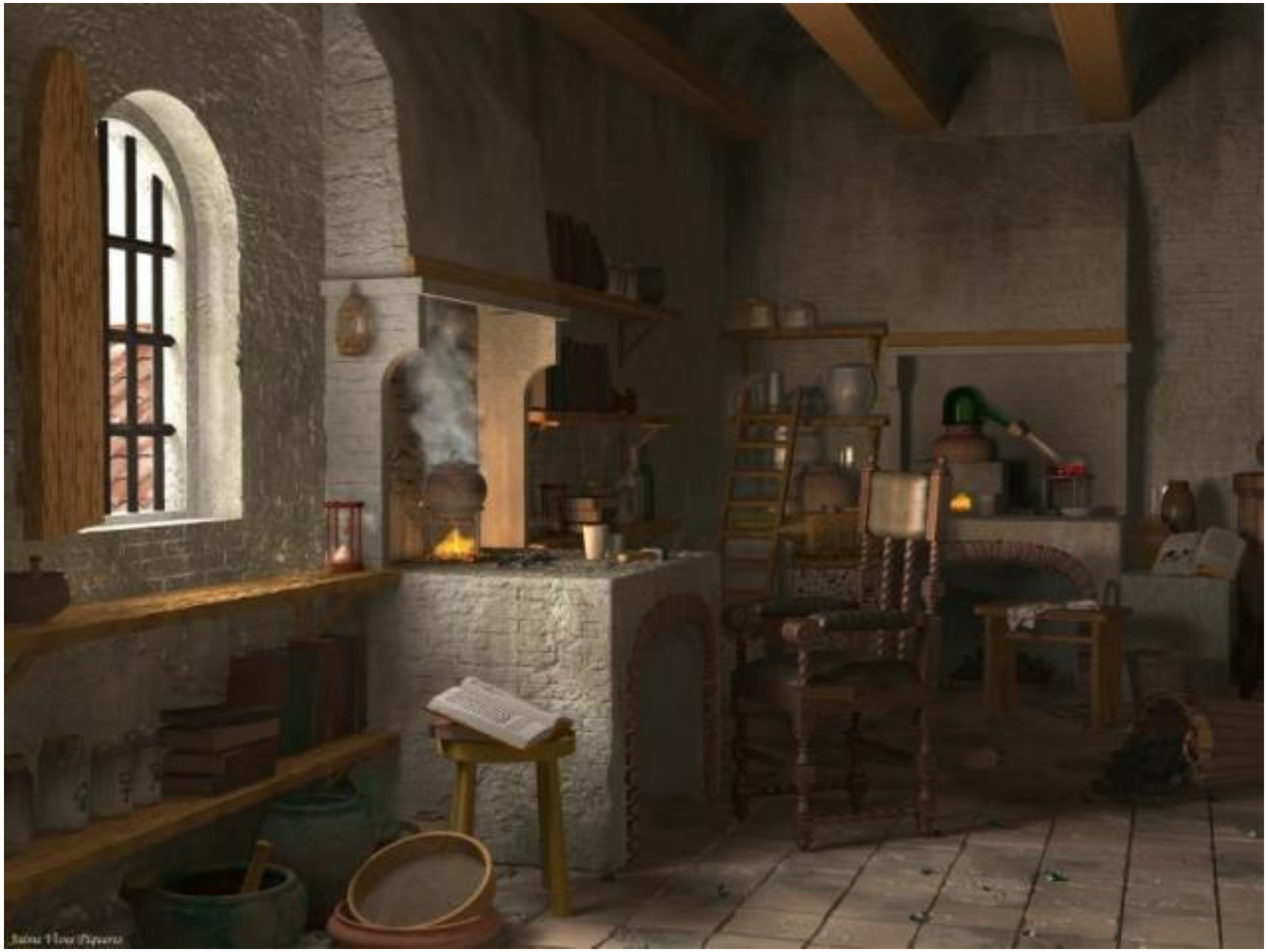
Images created with PovRay

- Tor Olav Kristensen


© 2003 Tor Olav Kristensen

Images created with PovRay

- Jaime Vives Piqueres

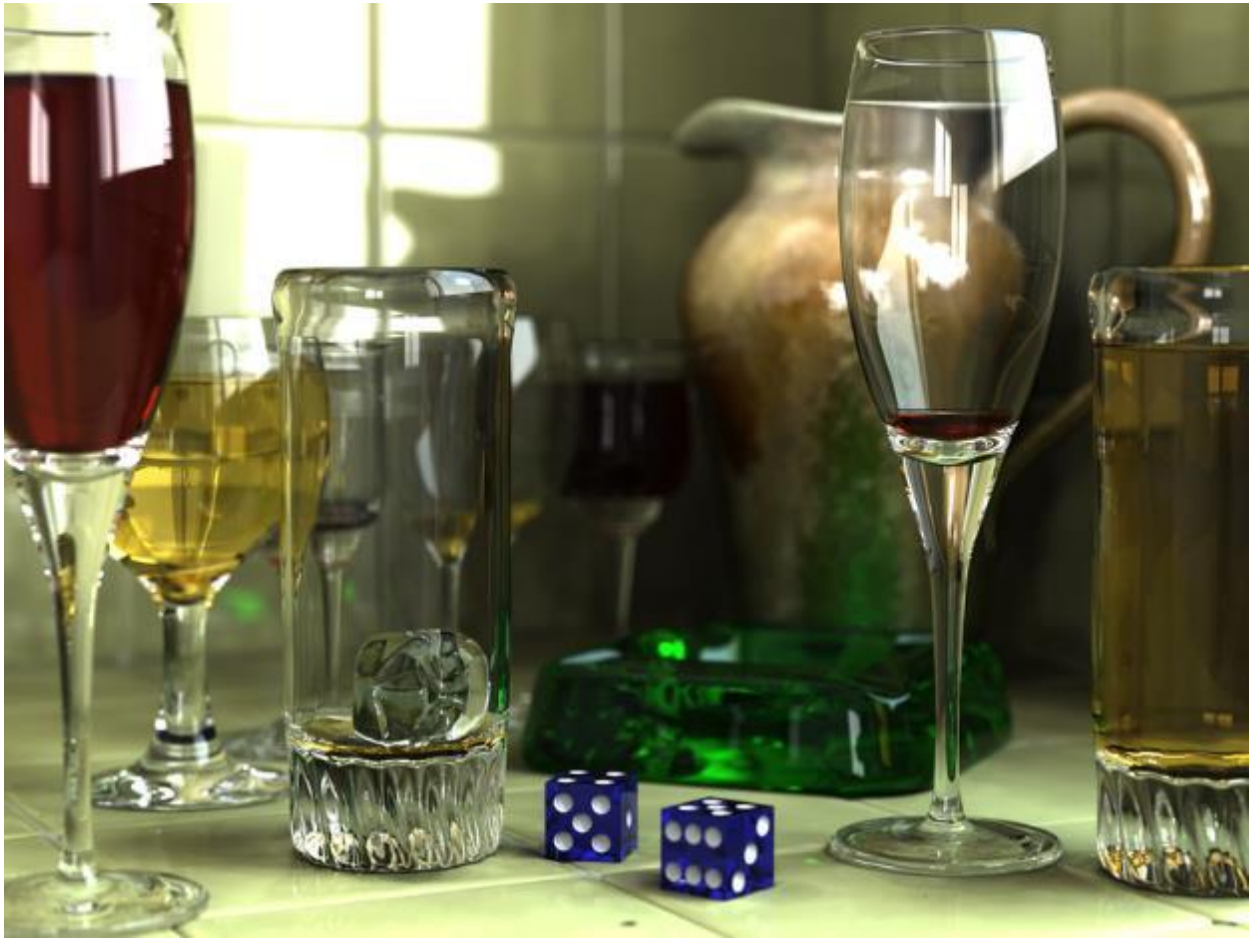


Images created with PovRay

- Jaime Vives Piqueres

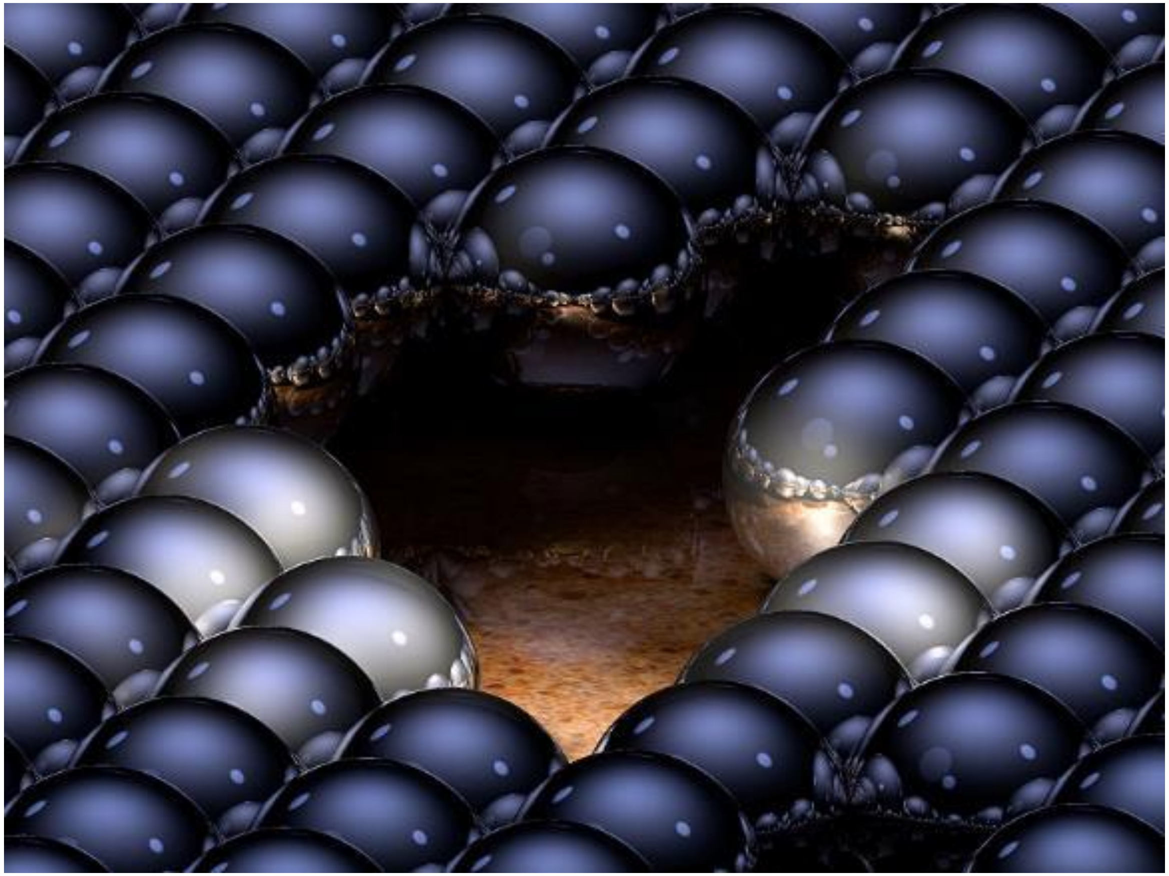
The
FROM
The PO
FROM
Team

Ray Tracing images from Wikipedia



- Explanation: http://en.wikipedia.org/wiki/File:Glasses_800_edit.png

Ray Tracing images from Wikipedia



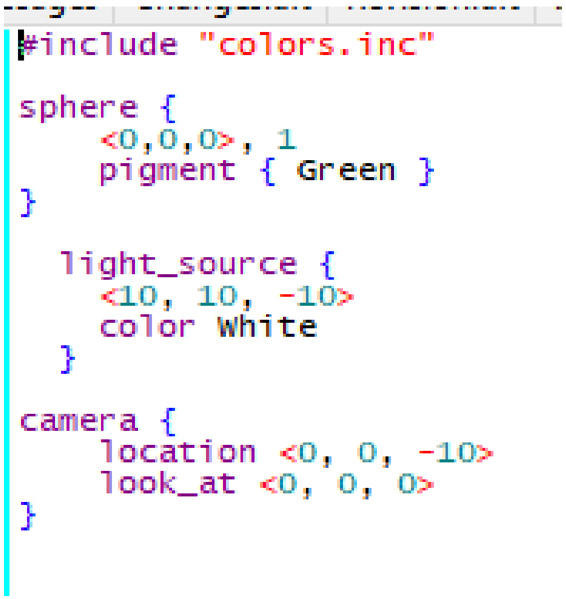
- Explanation: http://en.wikipedia.org/wiki/File:Ray-traced_steel_balls.jpg

Images created with PovRay

- Your images next…

Reto

- Instalar el POV Ray Tracer
- Partir del siguiente código de POVRay y modificar la escena



```
#include "colors.inc"

sphere {
    <0,0,0>, 1
    pigment { Green }
}

  light_source {
    <10, 10, -10>
    color White
  }

camera {
    location <0, 0, -10>
    look_at <0, 0, 0>
}
```

http://www.povray.org/documentation/3.7.0/t2_3.html#t2_3_5_3