

REALISMO

"El verdadero realismo consiste en revelar las cosas sorprendentes que se mantienen cubiertas por el hábito y nos impiden ver"

Jean Cocteau



1

REALISMO

1. Ray tracing

- 1.1. Modelo de iluminación transicional
- 1.2. Descripción del algoritmo
- 1.3. Software para ray tracing

2. Síntesis de imágenes basada en la física (Physically-based Rendering o PBR)

- 2.1. Radiancia
- 2.2. Ecuación de rendering
- 2.3. Radiosidad
- 2.4. Photon mapping

2

1

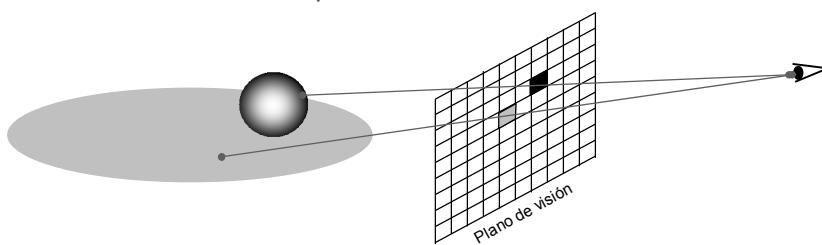
1. Ray tracing

- **Gauss:** tres ideas sobre la luz:
 - La Luz viaja en línea recta
 - Los rayos de luz no interfieren entre sí cuando se cruzan
 - Los rayos de luz viajan desde las fuentes de luz al ojo (pero la física es invariante en una situación inversa)
- El ray tracing es una extensión del algoritmo de ray casting de **Appel** (1968) y se basa en la óptica geométrica
- Ray tracing recursivo (reflexiones y refracciones) de **Whitted** 1979
- Efectos físicos que se simula :
 - Superficies difusas y especulares, sombras arrojadas, transmisión y reflexión

3

1. Ray tracing

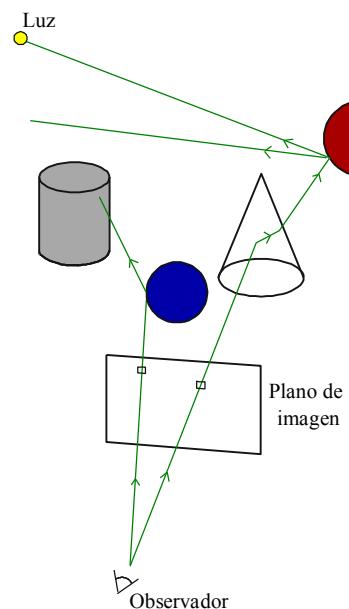
- Algoritmo de eliminación de partes ocultas de Ray Casting.
 - El método funciona en el espacio imagen
 - Para cada pixel de la imagen se crea un rayo (un par formado por un punto y un vector) con origen en el observador y que pase por el centro de dicho pixel.
 - Se encuentran las intersecciones con los objetos de la escena y nos quedamos con la más cercana al observador. Esta intersección determina el color del píxel.



4

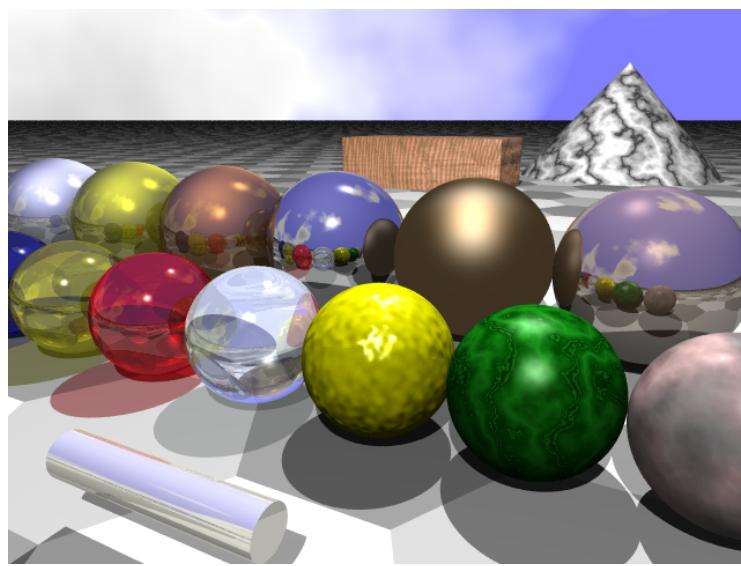
1. Ray tracing

- Permite de una sola vez generar imágenes con eliminación de partes ocultas, proyección en perspectiva, sombras y simular modelos de iluminación empíricos globales.
- Consiste en simular el recorrido de un rayo de luz en sentido inverso, desde el ojo de un observador hacia una escena



5

1. Ray tracing



6

1. Ray tracing

1.1. Modelo de iluminación transicional

- Están basados en el transporte de la luz. Son modelos de iluminación globales al considerar que las superficies de los objetos pueden estar iluminados por la luz procedente de otros por reflexión especular y refracción.
- De forma general:

$$I(\lambda) = \text{ambiente} + S(\text{difusa} + \text{especular}) + \text{reflexión} + \text{transmisión}$$

7

1. Ray tracing

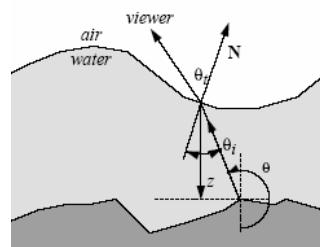
1.1. Modelo de iluminación transicional (reflexiones)



8

1. Ray tracing

1.1. Modelo de iluminación transicional (refracciones)



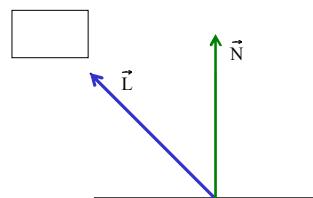
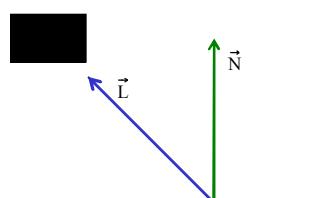
9

1. Ray tracing

1.1. Modelo de iluminación transicional (sombras)

- Cada fuente de luz se considera como punto de observación

$$I_{\lambda} = I_{a\lambda} K_{a\lambda} + \sum S_i f_{at} I_{L\lambda(i)} \left(K_{d\lambda} (\vec{N} \cdot \vec{L}_i) + K_s (\vec{N} \cdot \vec{H}_i)^n \right) \quad \text{con } S_i \in [0,1]$$



10

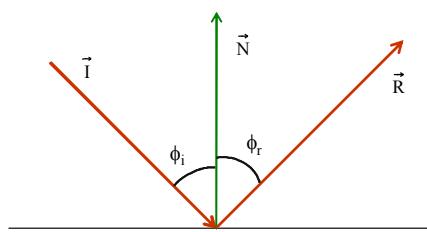
1. Ray tracing

1.1. Modelo de iluminación transicional (reflexiones)

- Se añade un término de intensidad reflejada que se calcula lanzando un rayo reflejado con respecto al incidente (el punto de intersección se considera como punto de observación)

$$I_{\lambda} = I_{a\lambda} K_{a\lambda} + \sum S_i f_{at} I_{L\lambda(i)} (K_{d\lambda} (\vec{N} \cdot \vec{L}_i) + K_s (\vec{N} \cdot \vec{H}_i)^n) + I_{r\lambda} K_s$$

- Dirección del rayo reflejado: $R = I - 2(N \cdot I) N$



11

1. Ray tracing

1.1. Modelo de iluminación transicional (refracciones)

- Si la luz alcanza un material translúcido o transparente se propaga por su interior
- La dirección del rayo transmitido se puede calcular de dos formas:

- Se ignora la refracción: los rayos no cambian de dirección:

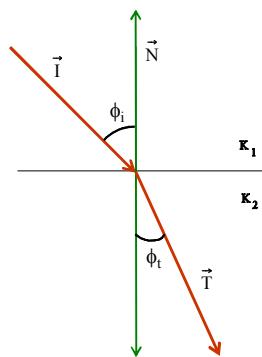
$$\bar{T} = \bar{I}$$

- Se usa la ley de Snell o Snelius:

$$\bar{T} = \frac{K_1}{K_2} \left\{ \sqrt{\left(\bar{N} \cdot \bar{I}\right)^2 + \left(\frac{K_1}{K_2}\right)^2} - 1 - \bar{N} \cdot \bar{I} \right\} \bar{N} + \bar{I}$$

Con solución cuando:

$$\left(\bar{N} \cdot \bar{I}\right)^2 > 1 - \left(\frac{K_1}{K_2}\right)^2$$



12

- Hay reflexión total con $K_2 < K_1$. El ángulo es $\phi_{rt} = \arcseno(K_2 / K_1)$

1. Ray tracing

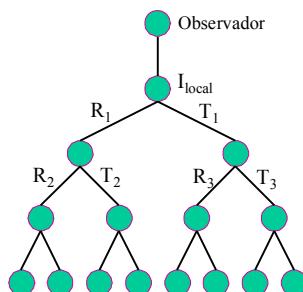
1.2. Algoritmo

- El uso de la recursión es la forma obvia de implementar un ray tracer
- En cada punto de intersección de un rayo con un objeto se consideran tres nuevos tipos de rayo: rayos hacia las fuentes de luz, un rayo reflejado y un rayo transmitido.

$$I = I_{\text{local}} + K_s I_{\text{reflejada}} + K_t I_{\text{transmitida}}$$

$$I_{\text{reflejada}} = I_{\text{local}} + K'_s I'_{\text{reflejada}} + K'_t I'_{\text{transmitida}}$$

$$I_{\text{transmitida}} = I''_{\text{local}} + K''_s I''_{\text{reflejada}} + K''_t I''_{\text{transmitida}}$$



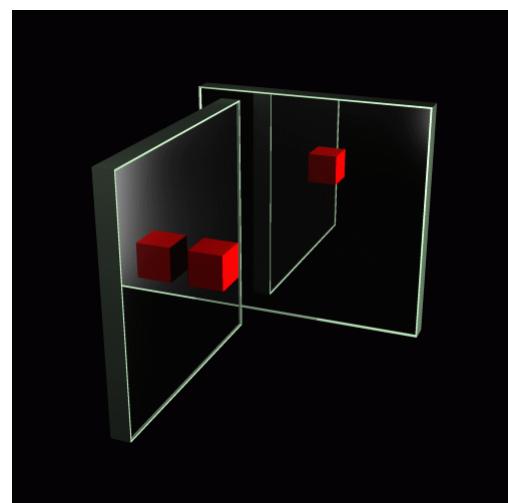
- Este proceso recursivo tiene cuatro mecanismos de parada

13

1. Ray tracing

1.2. Algoritmo

- Ejemplo de necesidad de mecanismos de parada



14

1. Ray tracing

1.2. Algoritmo. Función principal

```

Ray_Tracer (Punto3D inicio, Punto3D direccion,
Profundidad p, Color *c)
{
    Punto3D punto_inter, direccion_r, direccion_t;
    Color color_l, color_r, color_t, color_aux;
    Objeto objeto;
    int i;

    if (p > MAXPROFUNDIDAD) *c=NEGRO;
    else
        {If (Calcular_Interseccion(inicio, direccion,
                                    &objeto, &punto_inter))
            {color_l=NEGRO;
             for ( i=0; i<N_Luces; i++)
                 {Iluminacion_Local(objeto, punto_inter, color_aux);
                  color_l=color_l+color_aux;
                 }
             if (Traslucido(objeto) && !Mate(objeto))
                 {Direccion_Reflexion(objeto, punto_inter, &direccion_r);
                  Direccion_Transmision(objeto, punto_inter, &direccion_t);
                  Ray_Tracer (punto_inter, direccion_r, p+1, &color_r);
                  Ray_Tracer (punto_inter, direccion_t, p+1, &color_t);
                 }
            }
        }
}
    
```

```

if (Traslucido(objeto) && Mate(objeto))
    {Direccion_Transmision(objeto, punto_inter,&direccion_t);
     Ray_Tracer (punto_inter, direccion_t, p+1, &color_t);
     color_r=NEGRO;
    }
if (!Traslucido (objeto) && !Mate(objeto))
    {Direccion_Reflexion (objeto, punto_inter, &direccion_r);
     Ray_Tracer (punto_inter, direccion_r, p+1, &color_r);
     color_t=NEGRO;
    }
c->r=color_l+objeto.kr_r*color_r.r+objeto.kt_r*color_t.r;
c->g=color_l.g+objeto.kr_g*color_r.g+objeto.kt_g*color_t.g;
c->b=color_l.b+objeto.kr_b*color_r.b+objeto.kt_b*color_t.b;
}
else *c=COLOR_FONDO;
}
    
```

15

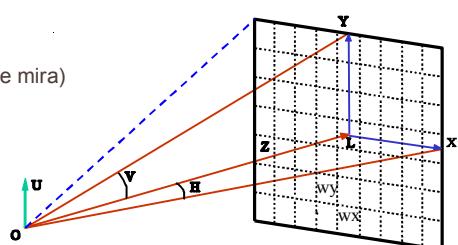
1. Ray tracing

1.2. Algoritmo.

- Función para calcular las direcciones de los rayos que se lanzan a la escena

- Cámara (pinhole)

- = O Posición del observador
- = Z Vector al centro de la imagen (punto de mira)
- = U vector de inclinación
- = L centro de la imagen
- = Y, X alto y ancho de la imagen
- = H, V ángulos focales
- = wx, wy tamaño de las celdilla



16

1. Ray tracing

1.2. Algoritmo.

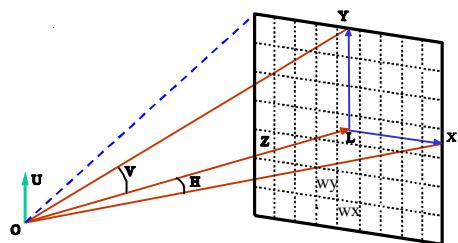
- Función para calcular las direcciones de los rayos que se lanzan a la escena

```

Viewing (int sizex, int sizey, float hfo, float vfo,
         Punto3D p_vision, Punto3D p_obs, Punto3D v_arriba,
         Punto3D *wx, Punto3D *wy, Punto3D *primer_rayo)
{
    int i, j;
    Punto3D vista;
    float distancia;

    vista=p_vision-p_obs;
    distancia=sqrt (vista.x*vista.x+vista.y*vista.y+vista.z*vista.z);
    Normalizar (&vista);
    Producto_Vectorial (vista, v_arriba, wx);
    Producto_Vectorial (*wx, vista, wy);
    distancia=2.0*distancia;
    distancia=distancia*tan(hfo*RADGRA)/sizex;
    *wy=distancia*tan(vfo*RADGRA)/sizey;
    *primer_rayo=p_vision-p_obs;
    primer_rayo->x= primer_rayo->x + (sizey*wy->x-sizex*wx->x)/2.0;
    primer_rayo->y= primer_rayo->x + (sizey*wy->y-sizez*wx.y)/2.0;
    primer_rayo->z= primer_rayo->z + (sizey*wy->z-sizez*wx_->z)/2.0;
}

```



17

1. Ray tracing

1.2. Algoritmo.

- Intersecciones. En ray tracing no solo se usan caras planas

— Ejemplo: objeto esfera

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

$$x = x_o + x_d t \quad y = y_o + y_d t \quad z = z_o + z_d t$$

$$t_{\pm} = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

$$A = x_d^2 + y_d^2 + z_d^2$$

$$B = 2[x_d(x_o - c_x) + y_d(y_o - c_y) + z_d(z_o - c_z)]$$

$$C = (x_o - c_x)^2 + (y_o - c_y)^2 + (z_o - c_z)^2 - r^2$$

18

1. Ray tracing

1.2. Algoritmo.

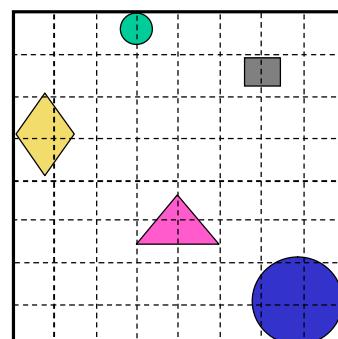
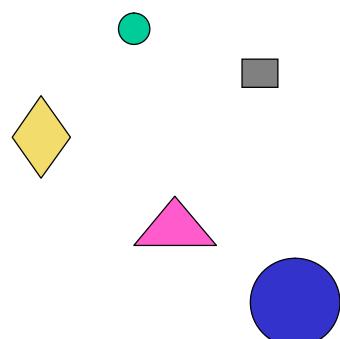
- Referencias sobre Intersecciones.
 - A. Glassner. "Introduction to Ray Tracing". Morgan Kaufmann 1989 (octava impresión 2000)
 - W. Stürzlinger. "Ray Tracing Triangular Trimmed Free-Form Surfaces". IEEE Trans. on Visualization and Computer Graphics, vol. 4, 1998 ([NURBS](#))
 - C. Benthin et al. "Interactive Ray Tracing of Free-Form Surfaces". Proceedings AFRIGRAPH 2004 ([Superficies bicubicas](#))
 - I. Wald and H. Seidel. "Interactive Ray Tracing of Point-based Models". Symposium on Point Based Graphics 2005 ([Puntos](#))
 - J. Kajiya. "New Techniques for Ray Tracing Procedurally Defined Objects". Computer Graphics, vol., 1983 ([Fractales](#))

19

1. Ray tracing

1.2. Algoritmo (técnicas de optimización)

- División uniforme del espacio (vóxeles)



20

1. Ray tracing

1.2. Algoritmo (técnicas de optimización)

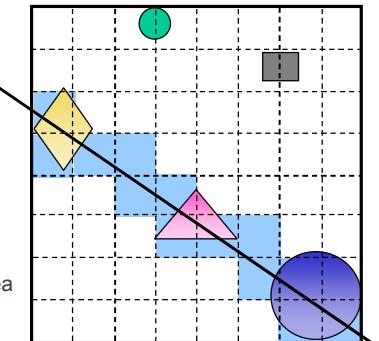
- División uniforme del espacio (vóxeles)

- Preprocesado

- = Calcular el volumen frontera de toda la escena
 - = Determinar la resolución
 - = Calcular la lista de objetos contenidos en todo o en parte en cada voxel

- Recorrido de la escena

- = Se usa un algoritmo rasterización de línea en 3D y 6-conectado



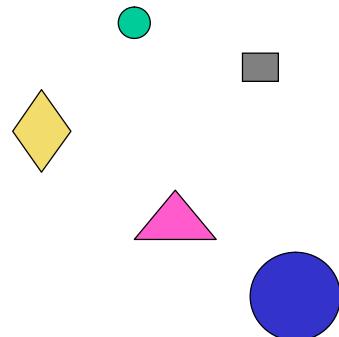
Recorrido de un rayo

21

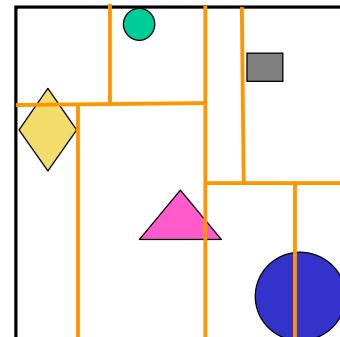
1. Ray tracing

1.2. Algoritmo (técnicas de optimización)

- Partición binaria del espacio con kd-tree



Escena inicial



Escena preprocesada

22

1. Ray tracing

1.3. Software

- POV-RAY
 - Un ejemplo de Ray Tracer de dominio público
 - Utiliza un lenguaje de descripción de escenas
 - Dirección:
<http://www.povray.org>



23

1. Ray tracing

1.3. Software (POV-RAY ejemplo)

```
#include "colors.inc"
#include "shapes.inc"

camera { location <0, 9, -60> angle 20 look_at <-2, 4, 0> }

light_source { <30, 40, -100> color White }

background { color SkyBlue }

#declare P = polygon { 12,
<0, 0>, <0, 6>, <4, 6>, <4, 3>, <1, 3>, <1, 0>, <0, 0>,
<1, 4>, <1, 5>, <3, 5>, <3, 4>, <1, 4>
}

#declare O = polygon { 10,
<0, 0, -3>, <0, 6, -3>, <4, 6, -3>, <4, 0, -3>, <0, 0, -3>,
<1, 1, -3>, <1, 5, -3>, <3, 5, -3>, <3, 1, -3>, <1, 1, -3> }

#declare V = polygon { 8,
<1, 0>, <0, 6>, <1, 6>, <2, 1>, <3, 6>, <4, 6>, <3, 0>, <1, 0>}
```



24

Realismo

IG

1. Ray tracing

1.3. Software (POV-RAY ejemplo. Continuación)

```
union {
    object { O translate 0*x }
    object { V translate 5*x }
    pigment { colour rgb<0.9,0.7,0.4> }
    finish { ambient 0.1 diffuse 0.2 reflection 1}
    translate -3*x
}

object {P translate -8.2*x
    pigment { colour rgb<0.9,0.7,0.4>}
    finish { ambient 0.1 diffuse 0.7}

plane { y, 0
    pigment { color rgb<0.3,1,0> }
}
```



25

Realismo

IG

1. Ray tracing

1.3. Software (POV-RAY ejemplo. Continuación)

```
#declare O =
    polygon {
        id: 12,
        <0, 0, 0>, <0, 0, 1>, <0, 1, 1>, <0, 1, 0>, <0, 0, 0>, <0, 0, 0>, <0, 0, 0>, <0, 0, 0>
    }

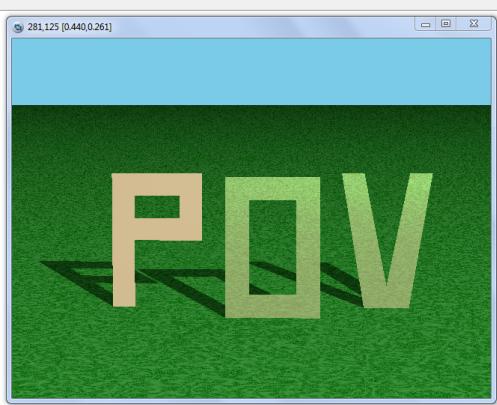
#declare P =
    polygon {
        id: 10,
        <-0.1, 0, -0.3>, <0, 0, -0.3>, <0, 0, -0.2>, <0, 0, -0.1>, <0, 0, 0>, <0, 0, 0>, <0, 0, 0>, <0, 0, 0>
    }

#declare V =
    polygon {
        id: 11,
        <0, 0, 0>, <0, 0, 1>, <0, 1, 1>, <0, 1, 0>, <0, 0, 0>, <0, 0, 0>, <0, 0, 0>, <0, 0, 0>
    }

union {
    object { O translate 0*x }
    object { V translate 5*x }
    pigment { colour rgb<0.9,0.7,0.4> }
    finish { ambient 0.1 diffuse 0.2 reflection 1}
    translate -3*x
}

object {P translate -8.2*x
    pigment { colour rgb<0.9,0.7,0.4>}
    finish { ambient 0.1 diffuse 0.7}

plane { y, 0
    texture{ pigment{ jadet } finish{ diffuse 1.0 } }
}
```



13

1. Ray tracing

1.3. Software

- Mental ray (NVIDIA 2007)
 - Utilizado en 3ds max hasta 2017
 - Permite simular luces fotométricas
 - Dirección: <http://www.nvidia-arc.com/mentalray.html>



27

1. Ray tracing

1.3. Software

- Arnold (Solid Angle)
 - Utilizado desde 3D Studio Max 2018
 - Ray tracing basado en Monte Carlo
 - Dirección: <https://www.solidangle.com/arnold/>

Solid Angle
Joins Autodesk

$$L_o = L_e + \int_{\Omega} L_i \cdot f_r \cdot \cos \theta \cdot d\omega$$

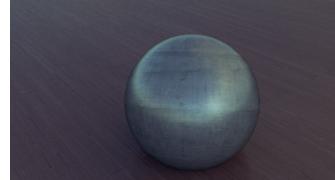


28

1. Ray tracing

1.3. Software

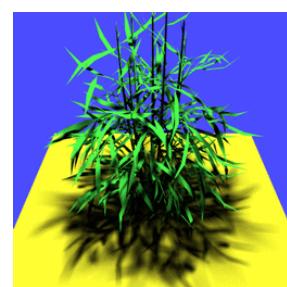
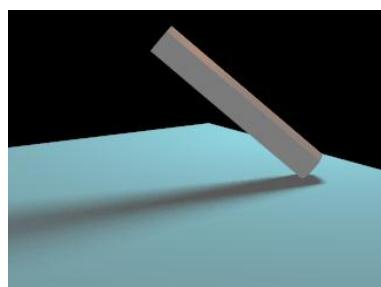
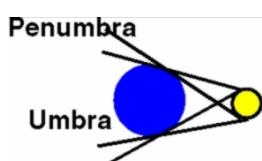
- Otros
 - Yafaray (usado por blender) <http://www.yafaray.org/>
 - Radiance (un clásico) <http://radsite.lbl.gov/radiance/>
 - 3Delight <https://www.3delight.com/>
 - Luxrender <http://www.luxrender.net/>
 - Appleseed <http://appleseedhq.net/>



29

1. Ray tracing (efectos)

- Penumbra (Soft Shadows)
P. Shirley. “[Realistic Ray Tracing](#)”. A. K. Peters LTD, 2000



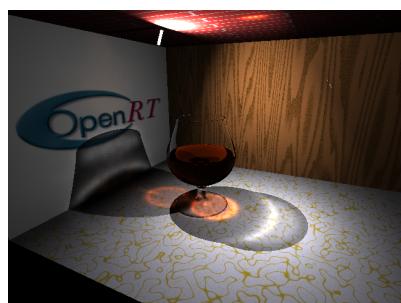
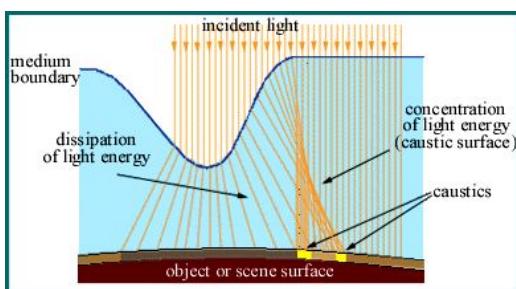
30

1. Ray tracing (efectos)

- Reflexión cáustica

J. Arvo, J. 1986. "Backward ray tracing". Proceedings of SIGGRAPH 1986

J. Günther, I. Wald, P. Slusallek. "Realtime Caustics using Distributed Photon Mapping". Eurographics Symposium on Rendering 2004

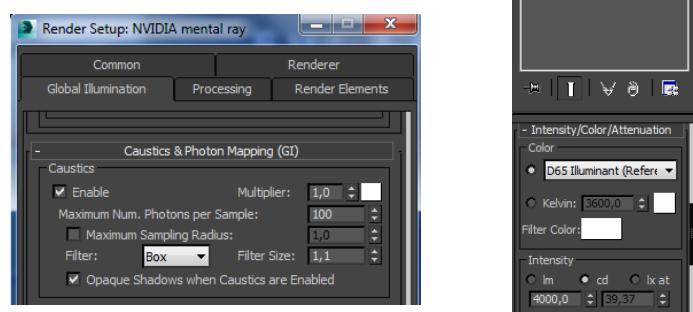


31

1. Ray tracing (efectos)

- Reflexión cáustica con Autodesk 3ds Max

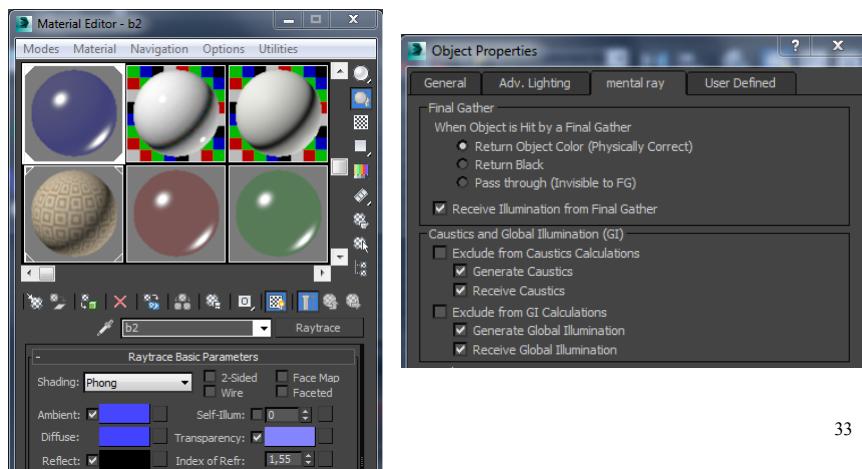
- Luces fotométricas que generen iluminación caústica
- Usar como render Mental ray
- Activar caústica en render setup



32

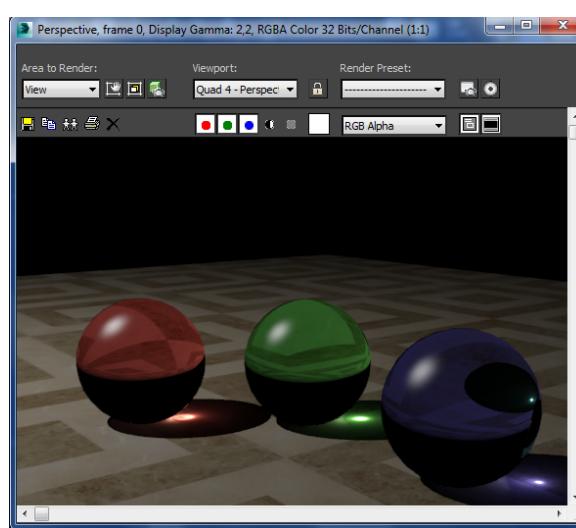
1. Ray tracing (efectos)

- Reflexión cáustica con Autodesk 3ds Max
 - Objetos con material tipo ray tracing que sean translúcidos y que generen y reciban iluminación caústica



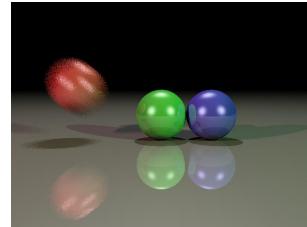
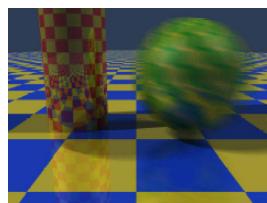
1. Ray tracing (efectos)

- Reflexión cáustica con Autodesk 3ds Max

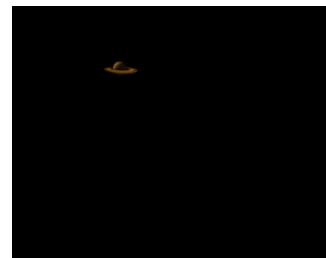
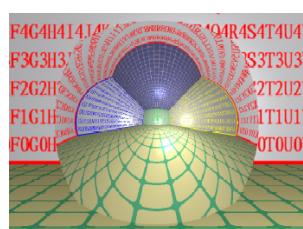


1. Ray tracing (efectos)

- Motion Blur



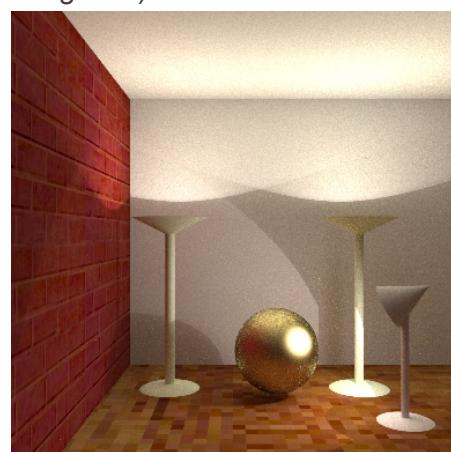
- Relatividad espacial (<http://www.vis.uni-stuttgart.de/>)



35

2. Síntesis de imágenes basada en la física

- Se pretende simular todas las reflexiones de la luz en una escena (iluminación local frente a iluminación global)



36

2. Síntesis de imágenes basada en la física

- La radiometría estudia la transferencia de energía radiante entre una fuente (radiador) y un receptor. Esta energía se puede considerar que se transporta como onda electromagnética o como fotones
- La fotometría estudia la energía radiante como estímulo de una sensación visual.
- Podemos decir que la fotometría es la radiometría cuando las longitudes de onda son las del espectro visible (entre 400-800 nm)

37

2. Síntesis de imágenes basada en la física

2.1. Radiancia

- **Flujo radiante, F** , energía emitida por un radiador por unidad de tiempo.
- **Flujo espectral, $F(\lambda)$** , El flujo radiante correspondiente a una longitud de onda
- **Intensidad radiante, I** , El flujo radiante emitido por un punto P por unidad de ángulo sólido

$$I = \frac{dF}{d\omega} \quad \text{W / esretorrad ianes (sr)}$$



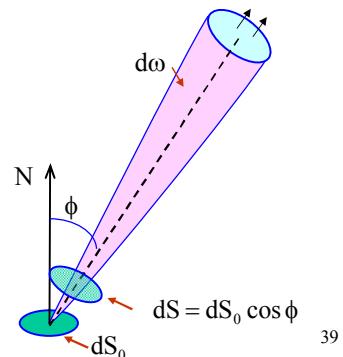
38

2. Síntesis de imágenes basada en la física

2.1. Radiancia

- **Radiancia, L** , la radiancia de un elemento radiante de superficie dS en una dirección dada es el cociente de la intensidad radiante en esa dirección por el cociente del área dS_0 proyectada sobre un plano normal a la dirección de radiación

$$L = \frac{dF^2}{d\omega dS} = \frac{dF^2}{d\omega dS_0 \cos \phi} \quad W /(\text{sr.m}^2)$$



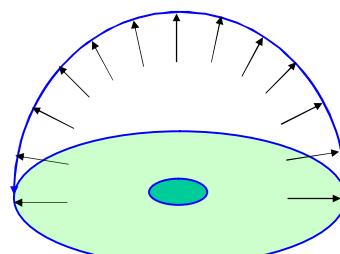
39

2. Síntesis de imágenes basada en la física

2.1. Radiancia

- **Excitancia radiante, E** , es la intensidad radiante emitida en todas direcciones por un área diferencial

$$E = \int_{\omega_{\text{out}}} L \cos \phi d\omega = \frac{dF}{dS_0} \quad W /(\text{m}^2)$$



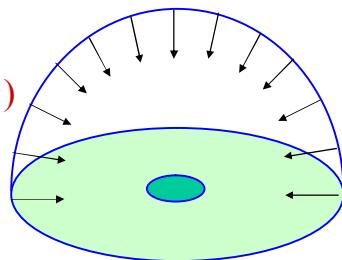
40

2. Síntesis de imágenes basada en la física

2.1. Radiancia

- Irradiancia, H , es la intensidad radiante recibida en todas direcciones por un área diferencial receptora

$$H = \int_{\omega_{out}} L \cos \phi d\omega = \frac{dF}{dS_0} \quad W / (m^2)$$



41

2. Síntesis de imágenes basada en la física

2.2. Ecuación de rendering

- Relaciona la radiancia total de una superficie con la radiancia debida a que la superficie sea un emisor (radiador) y la radiancia debida a la irradiancia (radiacia recibida por otras superficies)

$$L(\omega_0) = L_e(\omega_0) + \int_{\omega_i} \rho(\omega_0, \omega_i) L(\omega_i) \cos \phi_i d\omega_i$$

Radiancia total

Radiancia debida que la superficie sea emisora

Radiancia debida que la radiancia recibida por todas las superficies de una escena

Función de distribución de reflectancia bidireccional (BRDF)

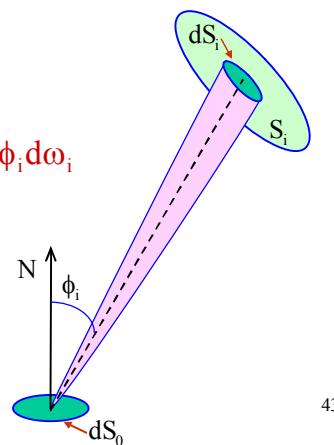
42

2. Síntesis de imágenes basada en la física

2.2. Ecuación de rendering

- La función de distribución de reflectancia bidireccional (bidirectional reflectance distribution function o BRDF) depende de las propiedades de las superficies.

$$L(\omega_0) = L_e(\omega_0) + \int_{\omega_i} \rho(\omega_0, \omega_i) L(\omega_i) \cos \phi_i d\omega_i$$



43

2. Síntesis de imágenes basada en la física

2.3. Radiosidad

- Es una solución de elementos finitos a la ecuación de rendering
- La escena se discretiza en áreas llamadas **pacthes**

$$B_i A_i = E_i A_i + \rho_i \sum_{j=1}^n B_j A_j F_{A_j A_i}$$

– Siendo

B_i la radiosidad del área i (A_i)

E_i la energía emitida desde A_i

ρ_i el coeficiente de reflectividad de A_i

$F_{dA_i dA_j}$ el factor de forma, que representa la fracción de energía que partiendo de la superficie A_j alcanza la superficie A_i

44

2. Síntesis de imágenes basada en la física

2.3. Radiosidad

- Relación de reciprocidad entre factores de forma:

$$F_{A_i A_j} A_i = F_{A_j A_i} A_j \rightarrow F_{A_i A_j} = F_{A_j A_i} \frac{A_j}{A_i} \rightarrow B_i = E_i + \rho_i \sum_{j=1}^n B_j F_{A_i A_j}$$

- Sistema de ecuaciones para las B_i incógnitas:

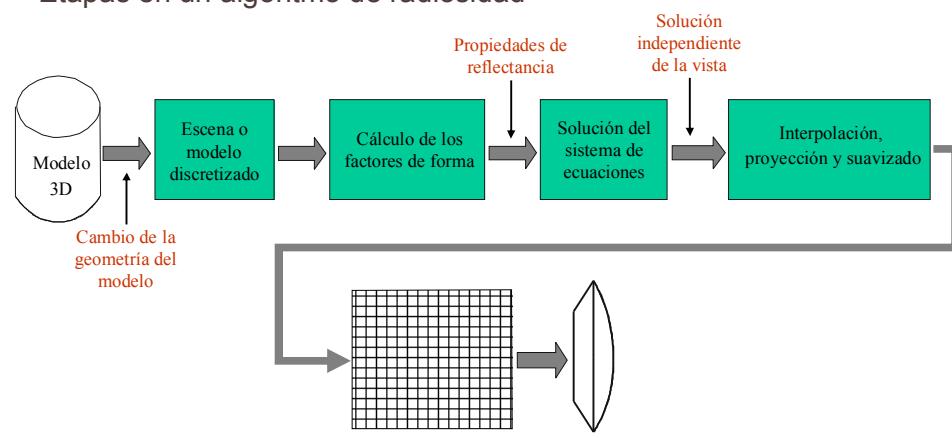
$$\begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{pmatrix} = \begin{pmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{pmatrix} + \begin{pmatrix} \rho_1 F_{A_1 A_1} & \rho_1 F_{A_1 A_2} & \dots & \rho_1 F_{A_1 A_n} \\ \rho_2 F_{A_2 A_1} & \rho_2 F_{A_2 A_2} & \dots & \rho_2 F_{A_2 A_n} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_n F_{A_n A_1} & \rho_n F_{A_n A_2} & \dots & \rho_n F_{A_n A_n} \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{pmatrix}$$

45

2. Síntesis de imágenes basada en la física

2.3. Radiosidad

- Etapas en un algoritmo de radiosidad

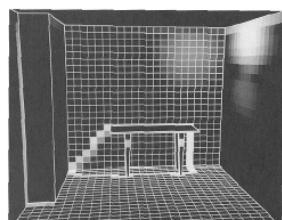
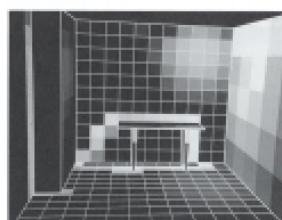
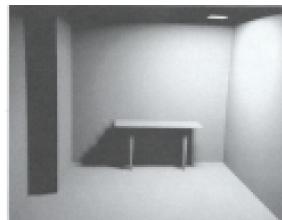
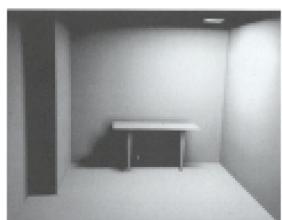


46

2. Síntesis de imágenes basada en la física

2.3. Radiosidad

- Discretización de la escena



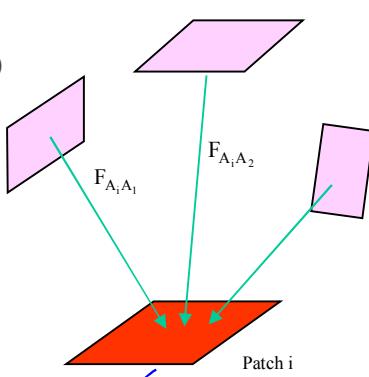
47

2. Síntesis de imágenes basada en la física

2.3. Radiosidad

- Solución del sistema de ecuaciones
 - Relajación de Gauss-Seidel (gathering)

$$\begin{pmatrix} B_1^t \\ B_2^t \\ \vdots \\ B_i^t \\ \vdots \\ B_n^t \end{pmatrix} = \begin{pmatrix} E_1 \\ E_2 \\ \vdots \\ E_i \\ \vdots \\ E_n \end{pmatrix} + \begin{pmatrix} \rho_1 F_{A_1 A_1} & \rho_1 F_{A_1 A_2} & \cdots & \rho_1 F_{A_1 A_n} \\ \rho_2 F_{A_2 A_1} & \rho_2 F_{A_2 A_2} & \cdots & \rho_2 F_{A_2 A_n} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_i F_{A_i A_1} & \rho_i F_{A_i A_2} & \cdots & \rho_i F_{A_i A_n} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_n F_{A_n A_1} & \rho_n F_{A_n A_2} & \cdots & \rho_n F_{A_n A_n} \end{pmatrix} \begin{pmatrix} B_1^{t-1} \\ B_2^{t-1} \\ \vdots \\ B_i^{t-1} \\ \vdots \\ B_n^{t-1} \end{pmatrix}$$



48

2. Síntesis de imágenes basada en la física

2.3. Radiosidad

- Solución del sistema de ecuaciones
 - Relajación de Southwell (shooting)

$$\begin{pmatrix} B_1^t \\ B_2^t \\ \vdots \\ B_n^t \end{pmatrix} = \begin{pmatrix} B_1^{t-1} \\ B_2^{t-1} \\ \vdots \\ B_n^{t-1} \end{pmatrix} + \begin{pmatrix} B_1^{t-1} \\ B_2^{t-1} \\ \vdots \\ B_n^{t-1} \end{pmatrix} \left(\frac{\rho_1 F_{A_1 A_i}}{B_1^{t-1}} + \frac{\rho_2 F_{A_2 A_i}}{B_2^{t-1}} + \frac{\rho_3 F_{A_3 A_i}}{B_3^{t-1}} + \frac{\rho_4 F_{A_4 A_i}}{B_4^{t-1}} \right) / (B_1^{t-1} + B_2^{t-1} + B_3^{t-1} + B_4^{t-1})$$

49

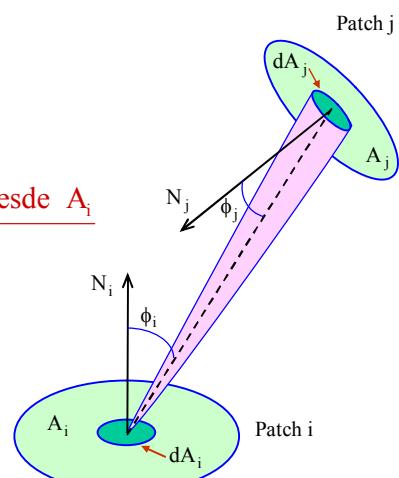
2. Síntesis de imágenes basada en la física

2.3. Radiosidad

- Cálculo de los factores de forma
 - Definición

$$F_{A_i A_j} = \frac{\text{Energía radiante que alcanza } A_j \text{ desde } A_i}{\text{Energía radiante total de } A_i}$$

$$F_{A_i A_j} = \frac{1}{A_i} \iint_{A_i} \iint_{A_j} \frac{\cos \phi_i \cos \phi_j}{\pi r^2} dA_j dA_i$$



50

2. Síntesis de imágenes basada en la física

2.3. Radiosidad

- Interpolación
 - Se necesitan conocer las intensidades en los vértices de cada patch

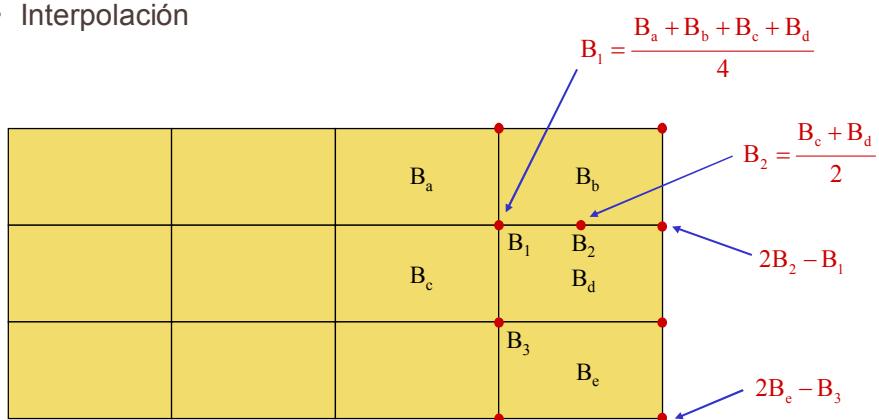


51

2. Síntesis de imágenes basada en la física

2.3. Radiosidad

- Interpolación



52

2. Síntesis de imágenes basada en la física

2.3. Radiosidad

- Suavizado de intensidades

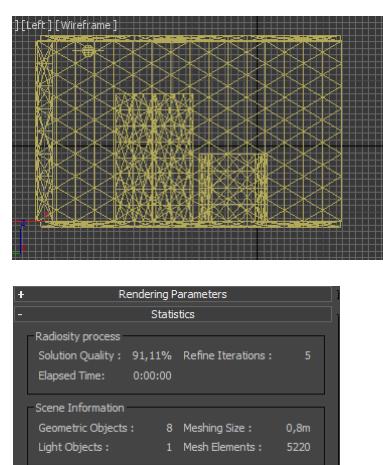
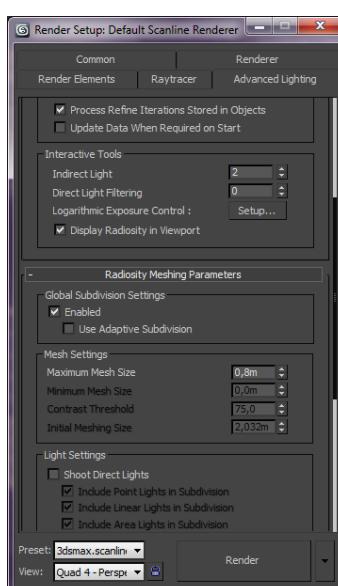


53

2. Síntesis de imágenes basada en la física

2.3. Radiosidad

- 3ds max

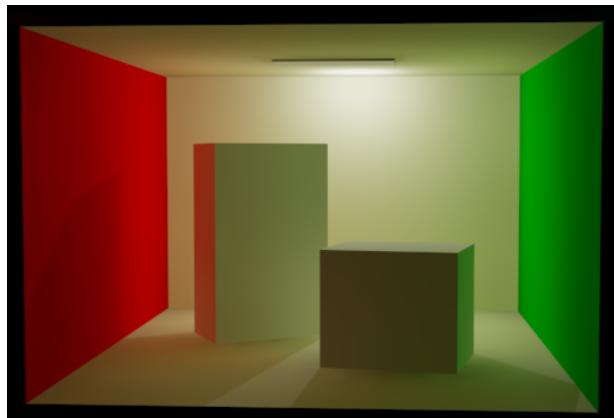
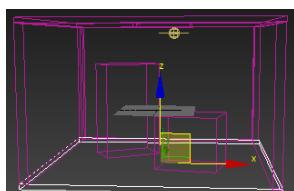


54

2. Síntesis de imágenes basada en la física

2.3. Radiosidad

- 3ds max



55

2. Síntesis de imágenes basada en la física

2.3. Radiosidad

- Combinando Radiosidad y Ray Tracing. Algoritmo híbrido
- Primera etapa: Radiosidad
 - Calcular factores de forma y la radiosidad
- Segunda etapa: Ray Tracing
 - Realizar un Ray Tracing tradicional
 - Sustituir las componentes ambientales y difusas de la iluminación local por la radiosidad

56

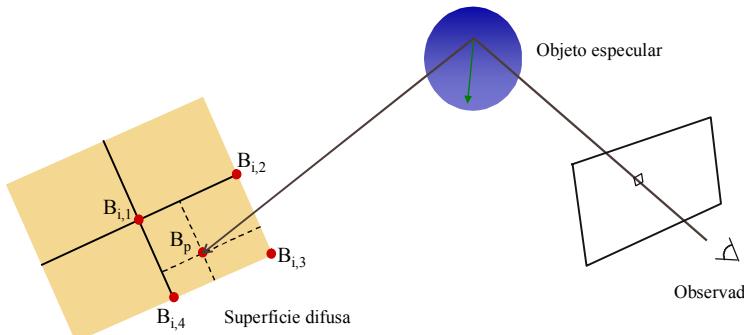
2. Síntesis de imágenes basada en la física

2.3. Radiosidad

- Combinando Radiosidad y Ray Tracing. Algoritmo híbrido

$$I_{\lambda} = I_{a\lambda} K_{a\lambda} + \sum S_i f_{at} I_{L\lambda(i)} (K_{d\lambda} (\vec{N} \cdot \vec{L}_i) + K_s (\vec{N} \cdot \vec{H}_i)^n) + I_{r\lambda} K_s + I_{t\lambda} K_t$$

$\Rightarrow = B_p(\lambda)$



57

2. Síntesis de imágenes basada en la física

2.4. Photon Mapping

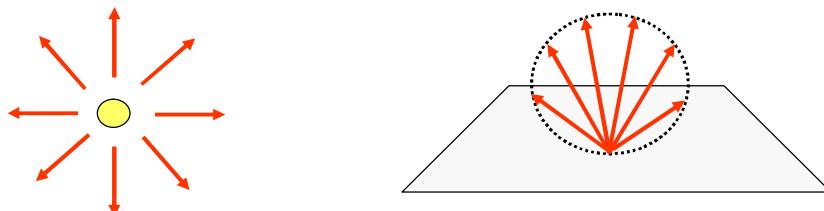
- Desarrollado por Jensen y Christensen en 1996. Puede simular caustica y medios participativos.
- Se basa en Monte Carlo ray tracing, pero con una solución más óptima
- Consta de dos pasos:
 - Photon tracing
 - Emisión de fotones
 - Seguimiento de fotones
 - Mapa de fotones
 - Síntesis de la imagen

58

2. Síntesis de imágenes basada en la física

2.4. Photon Mapping

- Photon tracing. Emisión de fotones
- Se generan fotones desde la fuente desde las fuentes de luz. La distribución de los fotones dependerá del tipo de luz

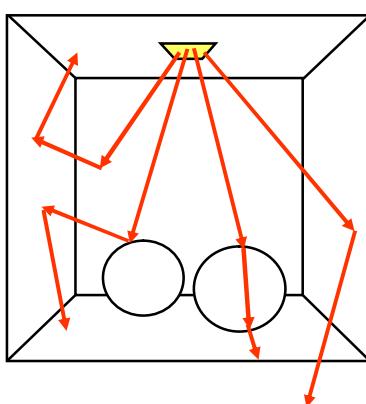


59

2. Síntesis de imágenes basada en la física

2.4. Photon Mapping

- Photon tracing. Seguimiento de fotones
- Cuando un fotón choca contra una superficie puede ser absorbido, reflejado (reflexión especular o difusa) o transmitido



60

2. Síntesis de imágenes basada en la física

2.4. Photon Mapping (Photon tracing. Seguimiento de fotones)

- Para decidir el tipo de interacción de un fotón y que camino seguir se realiza un sorteo (ruleta rusa)
- Suponiendo una superficie con todas las propiedades y siendo k_a , k_d , k_s y k_t los coeficientes de absorción, reflexión difusa, especular y transmisión respectivamente

$$K_a + K_d + K_s + K_t = 1 \quad \text{sea } v \text{ un número aleatorio entre } [0,1]$$

si

$v \in [0, K_a] \rightarrow \text{absorción}$

$v \in (K_a, K_a + K_d] \rightarrow \text{reflexión difusa}$

$v \in (K_a + K_d, K_a + K_d + K_s] \rightarrow \text{reflexión especular}$

$v \in (K_a + K_d + K_s, 1] \rightarrow \text{transmisión}$

61

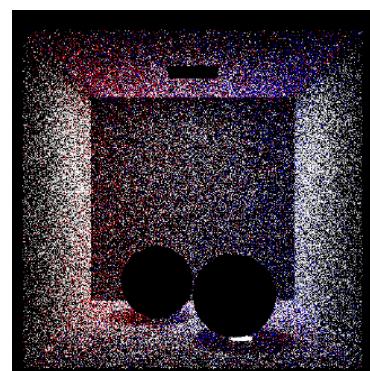
2. Síntesis de imágenes basada en la física

2.4. Photon Mapping

- Photon tracing. Mapa de fotones
- Los fotones se almacenan únicamente cuando golpean una superficie difusa. De esta forma se genera un mapa de fotones de la escena
- Información que se almacena

```
struct photon{
    float x,y,z; // posición
    char rgb[3]; // color
    char phi, theta; // dirección de incidencia
}
```

- Un fotón puede almacenar esta información varias veces. Se usa un kd-tree como estructura de datos

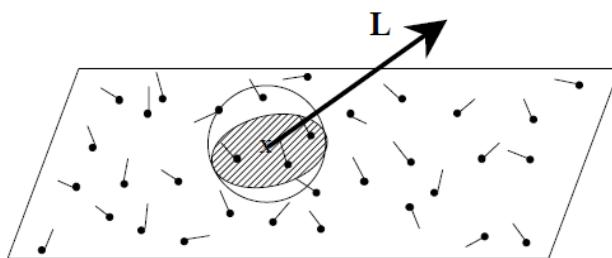


62

2. Síntesis de imágenes basada en la física

2.4. Photon Mapping

- Síntesis de la imagen
- Para encontrar el color en un punto se buscan los fotones más cercanos y se aplica la ecuación de rendering.

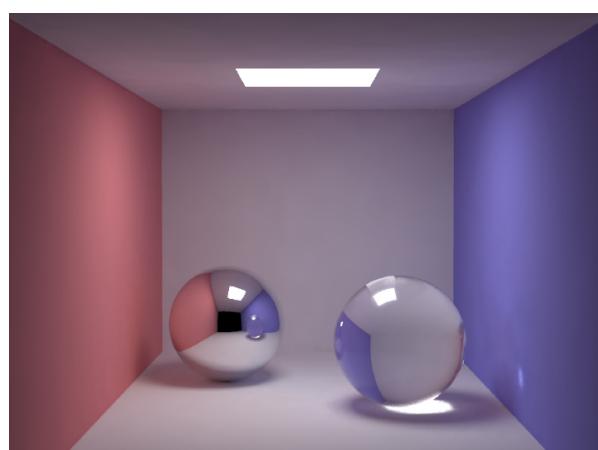


- Se aplica un filtro para disminuir el ruido de la imagen obtenida.

63

2. Síntesis de imágenes basada en la física

2.4. Photon Mapping



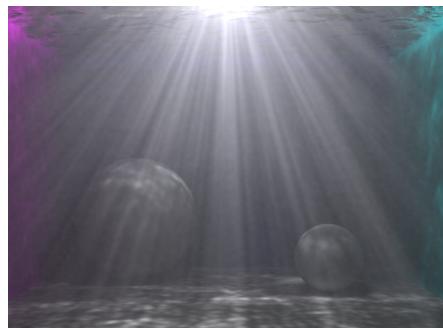
Jensen y Christensen Siggraph 2000

64

2. Síntesis de imágenes basada en la física

2.4. Photon Mapping

- Medios participativos. Los fotones pueden ser absorbidos o dispersados por el medio (volume photon map)
- Rayos crepusculares o rayos de Dios



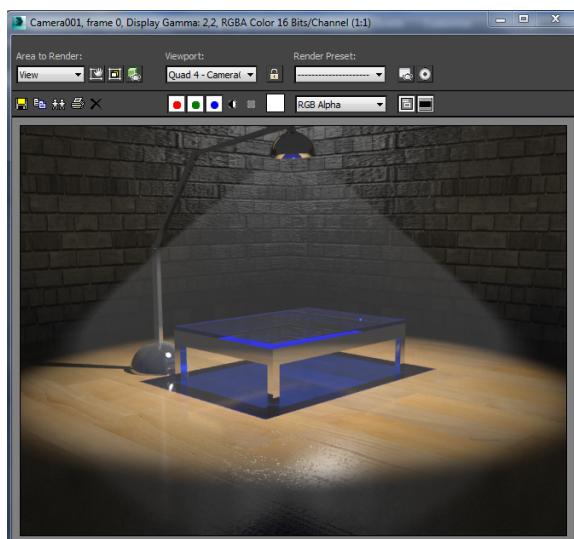
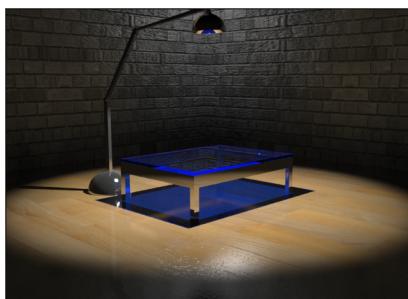
<https://chrisfontas.wordpress.com/>

65

2. Síntesis de imágenes basada en la física

2.4. Photon Mapping

- 3ds max



66