## Radiosidad

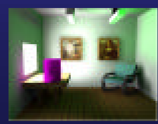
- Introducción
- Ecuación de Radiosidad
- Cálculo de factores de forma
- Refinamiento Progresivo
- Sub-división de superficies
- Monte Carlo Radiosity
- Métodos de dos pasos



## Iluminación global

Ray-tracing
Fuentes puntuales
Iluminación global especular
Texturas

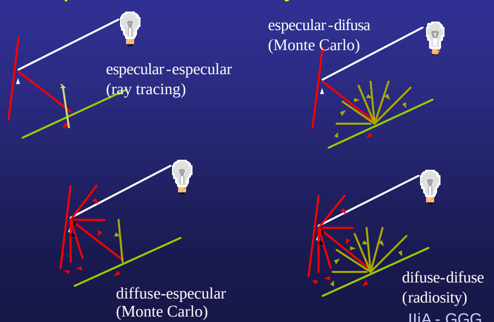
Radiosidad
Superfícies difusas



Texturas y *color bleeding*



## Tipos de reflectancia y transferencia



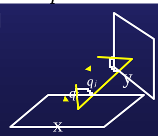



## Radiosidad

- Es la cantidad radiométrica y también el nombre del algoritmo
  - Radiosidad = flujo por unidad de area radiada desde una superfície (Watt/m²)
- Modela la iluminación considerando a todas las superfícies como idealmente difusas

$$B(x) = E(x) + r(x)\int B(y)\frac{\cos q \cos q'}{pr^2}V(x,y)dy$$

  - V - factor de visibilidad
    - 0 - no visble
    - 1 - visible



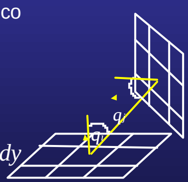



## Formulación discreta

- Subdivisión de las superficies en 'patches'
  - $B_i$ representa la radiosidad del patch i
  - $F_{ij}$ factor de forma geométrico

$$B_i = E_i + ?_i \sum_{j=1}^{n} B_j F_{ij}$$

$$F_{ij} = \frac{1}{A_i}\iint \frac{\cos q_i \cos q_j}{pr^2} V(x, y)dxdy$$





## Factor de forma

- $F_{ij}$ es la proporción de la energía lanzada por *i* que llega a *j*
- Es un factor puramente geométrico
- Propiedades:
  - reciprocidad

$$A_i F_{ij} = A_j F_{ji}$$

  - completitud

$$\sum_j F_{ij} = 1$$

## Solución de la ecuación

$$B_i = E_i + r_i \sum_{j=1}^{n} B_j F_{ij}$$

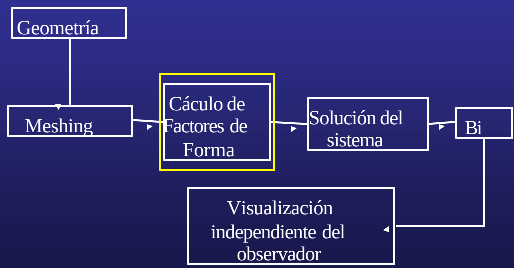
- $B_i$ incognitas, $F_{ij}$ deben calcularse
- Formulación matricial: se resuelve un sistema lineal

$$FB = E$$

donde:

$$B = \begin{bmatrix} B_1 \\ B_2 \\ \dots \\ B_n \end{bmatrix} \text{ and } E = \begin{bmatrix} E_1 \\ E_2 \\ \dots \\ E_n \end{bmatrix} \quad F = \begin{bmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \dots & -\rho_1 F_{1n} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & \dots & -\rho_2 F_{2n} \\ \dots & \dots & \dots & \dots \\ -\rho_n F_{n1} & -\rho_n F_{n2} & \dots & 1-\rho_n F_{nn} \end{bmatrix}$$

## Pipeline clasico de radiosidad



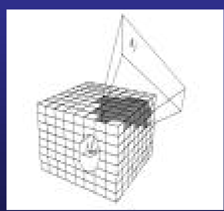

IIiA - GGG

## Cálculo de factores de forma

- Es el "cuello de botella" del método
- Se debe calcular para cada par de patches
  - Almacenamiento - N$^2$ ! (excesivo para escenas complejas)
- Métodos de cálculo
  - sin oclusión:
    - Integración directa
    - Integración de contorno (Teorema de Stoke)
    - Formulación angular
  - con oclusión:
    - métodos proyectivos
    - metodos de Monte Carlo

IIiA - GGG

## Método del Hemi-cube



- Aproxima el hemisfério superior de un patch por un hemi-cubo
- Proyecta el resto de patches de la escena en las caras
- El área proyectada es proporcional al FF
- Se puede utilizar z-buffer para las proyecciones (hardware)
- Tiene muchos problemas de precisión

IIiA - GGG

## Solución por Refinamiento Progresivo

- Resuelve el sistema por iteraciones sucesivas para cada fila
- No necesita almacenar la matriz de factores de forma
- Permite visualizaciones intermedias

$$B_i = E_i + r_i \sum_{j=1}^{n} B_j F_{ij}$$

$$B_j \text{ due to } B_i = \rho_j B_i F_{ji}$$

IIiA - GGG

## Refinamiento Progresivo

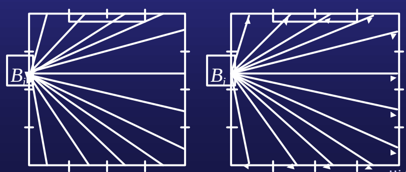
Utilizando $A_i F_{ij} = A_j F_{ji}$ e intercambiando *i* con *j*:

$$B_j \text{ due to } B_i = \rho_j B_i F_{ji} = \rho_j B_i F_{ij} \frac{A_i}{A_j}$$

Para todos los patches j: $B_j \text{ due to } B_i = \rho_j B_i F_{ij} \frac{A_i}{A_j}$

IIiA - GGG

- Estratégia de *shooting*
  - Lanza energía desde un patch *i*, calculando la contribución en todo el resto de patches *j*
- Estratégia de *gathering*
  - Recoge la energía lanzada en un patch *i* debido a la contribuición de todos los patches *j*






## Pseudo codigo

$B_i$ , $\Delta B_i = 0$ for all non-light sources

$B_i$ , $\Delta B_i = E_i$ for sources.

*$\Delta B_i$ = radiosidad no-lanzada*

```
for each iteration {
        sort in descending order of B_i ΔB_i ;
        for each patch  i { /*in the descending order*/
                compute the form factors F_ij using a hemi-cube at i;
                for each patch j≠i {
                        ΔRad = p_j * ΔB_i * F_ij * (A_i/A_j) ;
                        ΔB_j = ΔB_j + ΔRad ;
                        B_j = B_j + ΔRad;
                }
                ΔB_i = 0;
        }
        /*render scene here if desired*/
}/*until convergence of the B_i*/
```



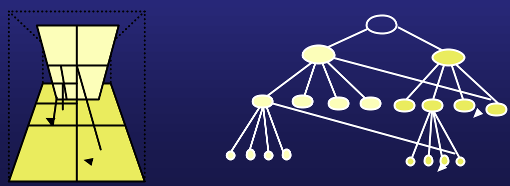
## Subdivisión en patches

- La subdivisión de las superficies es un punto crítico del algoritmo
- Poco fina - menos precisión, dificultad para representar variaciones fuertes (sombras)
- Demasiado fina - muy costoso y puede quedar sobremuestrado en algunas regiones
- Necesidad de métodos adaptativos
  - Subdivisión adaptativa proyectando fuentes
  - Radiosidad jerárquica



## Radiosidad Jerárquica

- Intercambio de energía a distintos niveles de subdivisión
- Refinamiento de acuerdo a la energía de intercambio
  - Se establecen enlaces (y se almacenan) segun un criterio de refinamiento



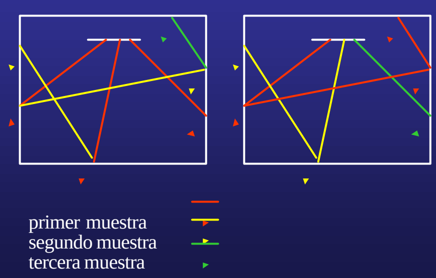


## ~~Monte Carlo Radiosity~~

- Simulate a stochastic process
- Can be used to:
  - Compute form factors
  - Estimate the radiosity solution as a Neumann series ➔ Random Walk
- Advantages:
  - Provide physically-based solution
  - No explicit form factor storage needed
    ➔ deals well with complex environments



## Estratégias: *breadth-first* y *depth-first*



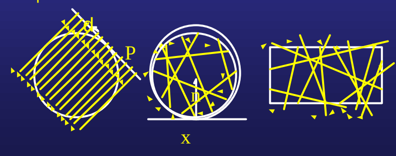
## Algoritmo Básico

Initialize power counters:
- the total accumulated power $P_i \leftarrow \phi_i$
- the unshot power $\Delta P_i \leftarrow \Phi_i$ \\
- and the receiving power $\delta P_i \leftarrow 0$
for all patches $i$.
- Compute the initial total unshot power $\Delta P_T = \sum_i \Delta P_i$
- While not converged, do
- Choose number of samples $N$
- For all patches $i$ do
  - $N_i \leftarrow N \frac{\Delta P_i}{\Delta P_T}$
  - Do $N_i$ times
  - Generate a local line at a random point $x$ on patch $i$ with cosine-distributed random direction $\theta$
  - Determine the nearest patch intersection $j$ of the line with the surfaces in the scene
  - $\delta P_j \leftarrow \delta P_j + \rho_j \frac{\Delta P_T}{N}$



## Monte Carlo Radiosity

- The way lines are sampled define two families:
  - Local lines
    - Cosine-distribution lines are built laying on surfaces
  - Global lines
    - Build a global uniform density of lines. Lines are cast irrespective of surfaces






## Monte Carlo Radiosity

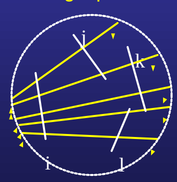
- Global Monte Carlo
  - Properties:

Estimate the number of lines crossing a patch

$$E(N_i) = N \frac{2A_i}{A_T}$$

Form factor computation

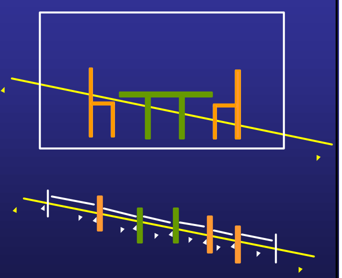
$$F_{ij} = \frac{N_{ij}}{N_i}$$






## Monte Carlo Radiosity

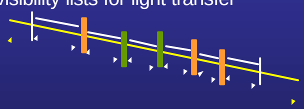
- Global Monte Carlo
  - Visibility lists






## Monte Carlo Radiosity

- Global Monte Carlo
  - Use visibility lists for light transfer



- Multi-path algorithm
  - Update paths of different levels of the random walk at each interaction
- Global transfer algorithm
  - Progressive refinement iterations



## Hierarchical Monte Carlo Radiosity

- Idea similar a Radiosidad jerárquica
- No necesita guardar *links* de refinamiento
- Optimiza memoria
  - Permite tratar entornos complejos


IIiA - GGG

## Rendering

- Radiosidad obtiene solutiones independientes del observador
- Se puede utilizar rendering básico de polígonos para visualización (hardware gráfico)
  - *Flat shading*
    - pinta cada patch con color uniforme (poco realista)
  - Gouraud shading
    - calcula las radiosidades en los vértices
    - interpolación lineal
    - método provisto en targetas gráficas

IIiA - GGG

## Radiosidad vs Ray-Tracing

- RT lanza rayos para cada pixel de la imagen siguiendo su curso óptico
  - Tratamiento correcto de superfícies especulares
  - Dependiente del observador
- Radiosidad está basado en el intercambio de energía entre los objetos de la escena
  - Tratamiento correcto de superfícies difusas
  - Independiente del observador
  - Tratamiento de elementos finitos
    - representación discreta de la escena

IIiA - GGG

## Métodos combinados

- Estratégia de dos pasos
  - Primer paso: espacio objeto
    - Solución de radiosidad jerarquica
  - Segundo paso: espacio imagen
    - Path tracing desde el observador

IIiA - GGG

## ~~Overview of the Algorithm (Perez et al CGI2002)~~

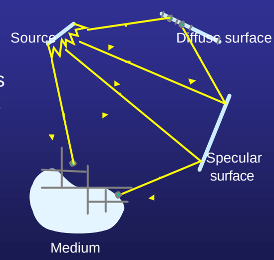
- 1st Pass: Extended HMCR
  - Bekaert et al.'s Hierarchical Monte Carlo Radiosity
  - Coarse solution: not for rendering
- 2nd Pass: Ray Tracing + Final Gather
  - Improved Importance Sampling with Link Probabilities
  - Use links as a representation of part of the irradiance

IIiA - GGG

## ~~First Pass: Coarse Global Illumination~~

Extended HMCR

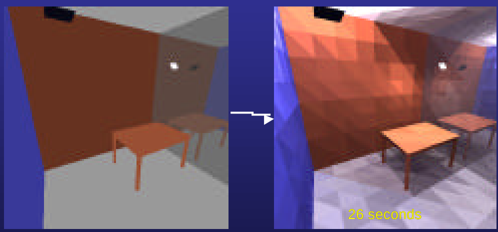
- Shoot particles from sources
  - Interact with media and surfaces
  - Bounce at specular surfaces (mirror, glass)
- Store energy hierarchically
  - Diffuse component of BRDF
- Iterate distributing energy




IIiA - GGG

## First Pass: Example

- Simple room (direct + indirect illumination)




Model

HMCR (Rough Global Illumination)

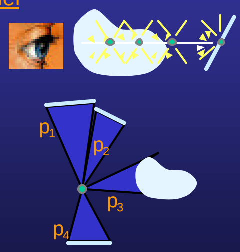
IIiA - GGG

## Second Pass: High Quality Image

RayTracer with Final Gather

For each line of sight (pixel)

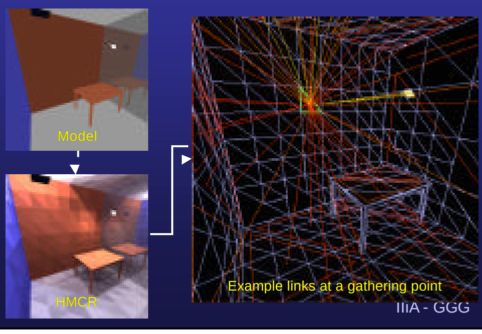
- Generate gathering points
  - 1 for the visible surface
  - N for the media traversed
- For each gathering point
  - Compute link probabilities (LP)
  - Estimate irradiance based on LP
- Estimate eye radiance using obtained estimations




IIiA - GGG

## Second Pass: Example

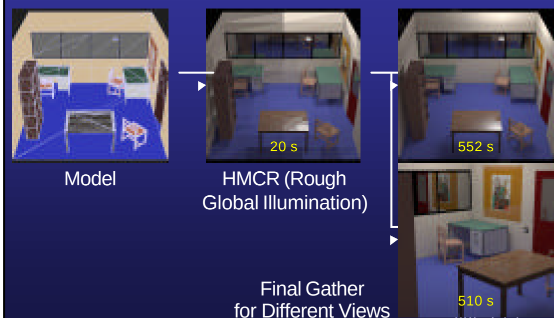
- Simple room (direct + indirect illumination)




IIiA - GGG

## Results

- Office room: indirect illumination




Model

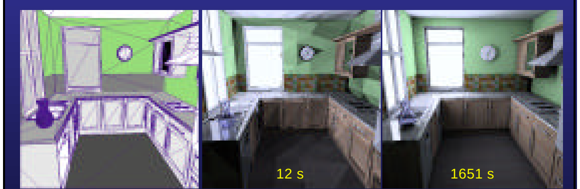
HMCR (Rough Global Illumination)

Final Gather for Different Views

IIiA - GGG

## Results

- Kitchen scene (direct illumination)
  - Lambertian light sources (windows)




Model (Courtesy of LightWork Design)

HMCR (Rough Global Illumination)

Final Gather

IIiA - GGG

## Results

- Kitchen scene: More views


IIiA - GGG

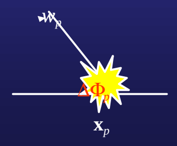
## Photon Mapping

- Jensen EGRW 95, 96
- Simulates the transport of individual photons
- Photons emitted from source
- Photons deposited on surfaces
- Photons reflected from surfaces to other surfaces
- Photons collected by rendering



## What is a Photon?

- A photon *p* is a particle of light that carries flux $\Delta\Phi_p(\mathbf{x}_p, w_p)$
  - Power: $\Delta\Phi_p$ – magnitude (in Watts) and color of the flux it carries, stored as an RGB triple
  - Position: $\mathbf{x}_p$ – location of the photon
  - Direction: $w_p$ – the incident direction $w_i$ used to compute irradiance
- Photons vs. rays
  - Photons propogate flux
  - Rays gather radiance



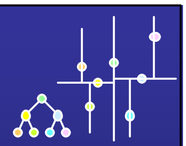



## Sources

- Point source
  - Photons emitted uniformly in all directions
- Power of source (W) distributed evenly among photons
- Flux of each photon equal to source power divided by total # of photons
- For example, a 60W light bulb would send out a total of 100K photons, each carrying a flux $\Delta\Phi$ of 0.6 mW
- Photons sent out once per simulation, not continuously as in radiosity



## Storing Photons



- Uses a kd-tree – a sequence of axis-aligned partitions
  - 2-D partitions are lines
  - 3-D partitions are planes
- Axis of partitions alternates wrt depth of the tree
- Average access time is $O(\log n)$
- Worst case $O(n)$ when tree is severely lopsided
- Need to maintain a balanced tree, which can be done in $O(n \log n)$
- Can find k nearest neighbors in $O(k + \log n)$ time using a heap



## Reflected Radiance

- Recall the reflected radiance equation

$$L_r(\mathbf{x}, w_r) = \int_\Omega f_r(w_i, w_r) L_i(\mathbf{x}, w_i)(N \cdot w_i)\, dw_i$$

- Convert incident radiance into incident flux

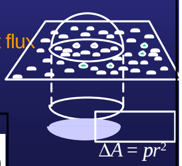
$$L_i(\mathbf{x}, w_i) = \frac{d^2\Phi_i(\mathbf{x}, w_i)}{(N \cdot w_i)\, dw_i\, dA_i}$$

- Reflected radiance in terms of incident flux

$$L_r(\mathbf{x}, w_r) = \int_\Omega f_r(w_i, w_r) \frac{d^2\Phi_i(\mathbf{x}, w_i)}{dA_i}$$

- Numerically

$$L_r(\mathbf{x}, w_r) \approx \frac{1}{\Delta A} \sum_{p=1}^{n} f_r(w_p, w_r)\Delta\Phi_p(\mathbf{x}, w_p)$$






## Rendering

- Photons in photon map are collected by eye rays cast by a distributed ray tracer
- Multiple photon maps
  - Indirect irradiance map
  - Caustic map
- Rays use the radiance constructed from reflected flux density from nearest neighbor photons

## Resultados