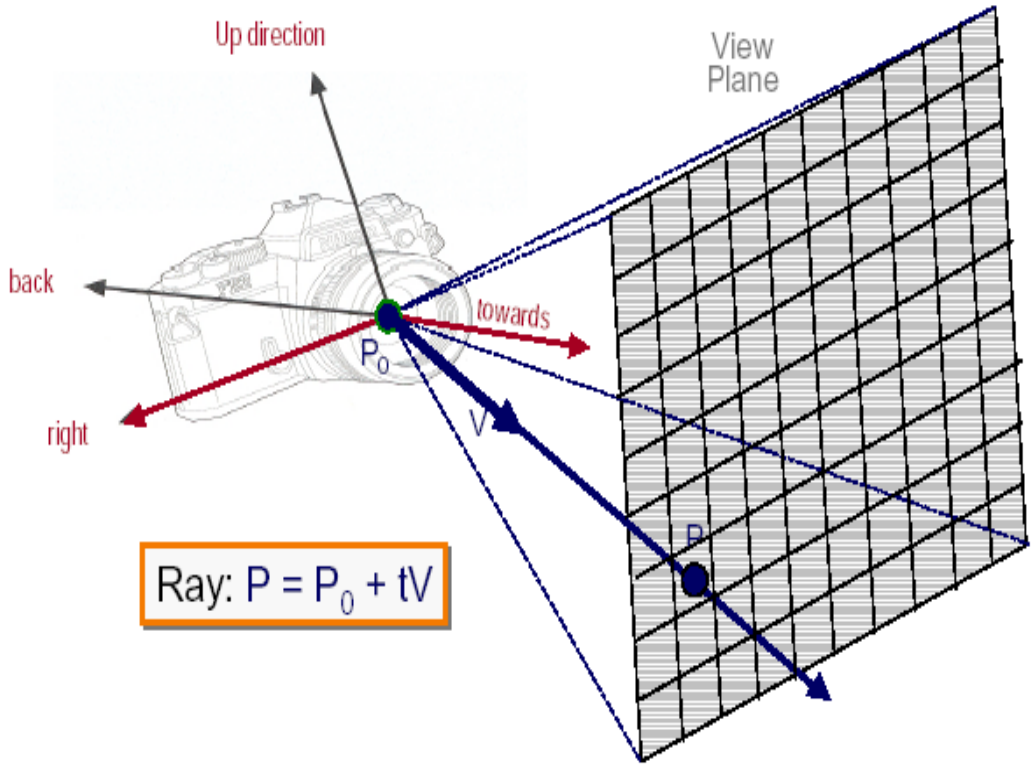
Técnicas avanzadas basadas en trazado de rayos

Trazado de rayos: Generación

- La cámara

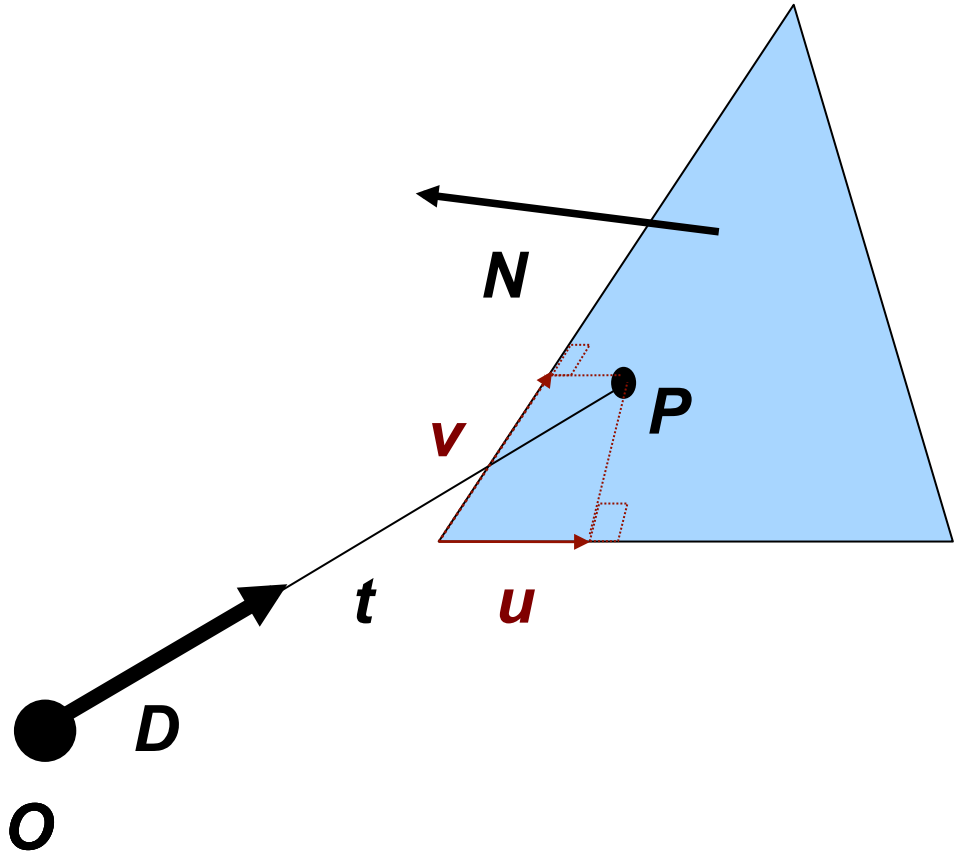
Rayo = f(x,y)



Graphics and Imaging Lab GIGA

Trazado de rayos: intersecciones

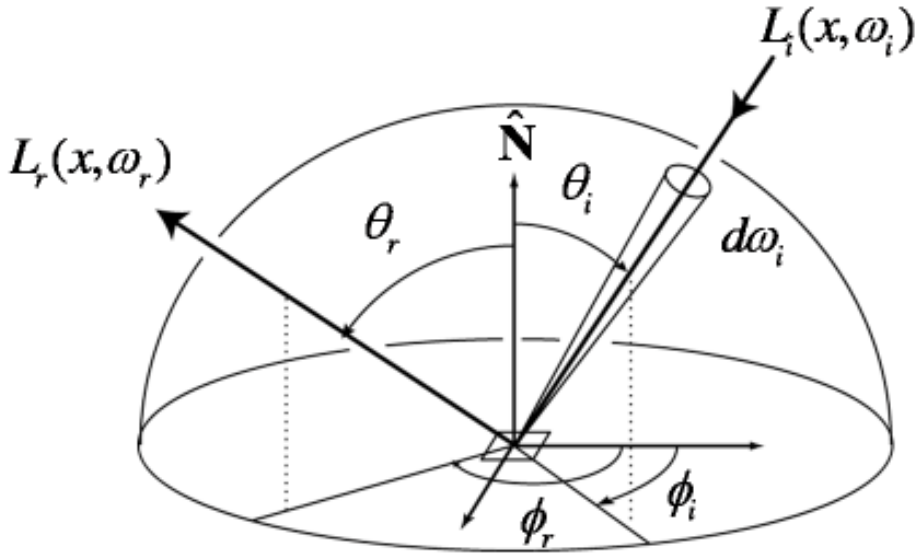
- Rayo-triángulo:
  - Intersección con plano.
  - Comprobación de rangos



Graphics and Imaging Lab GIGA

Trazado de rayos: color

- Ecuación de reflexión



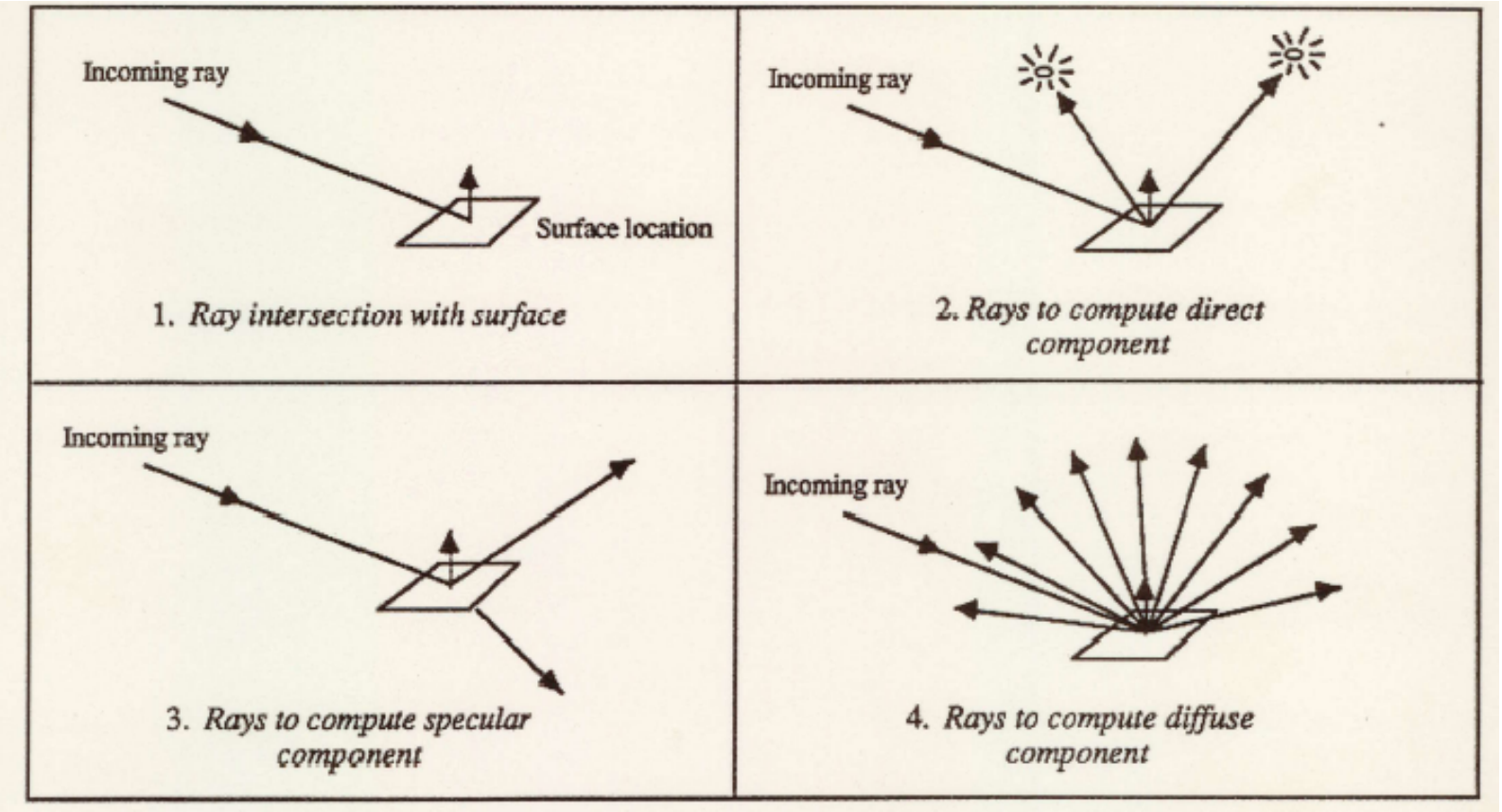
$$L_r(x,\omega_r) = \int_{H^2} f_r(x,\omega_i \rightarrow \omega_r) L_i(x,\omega_i) \cos\theta_i \, d\omega_i$$

BRDF

Graphics and Imaging Lab GIGA

Trazado de rayos

- Resumen



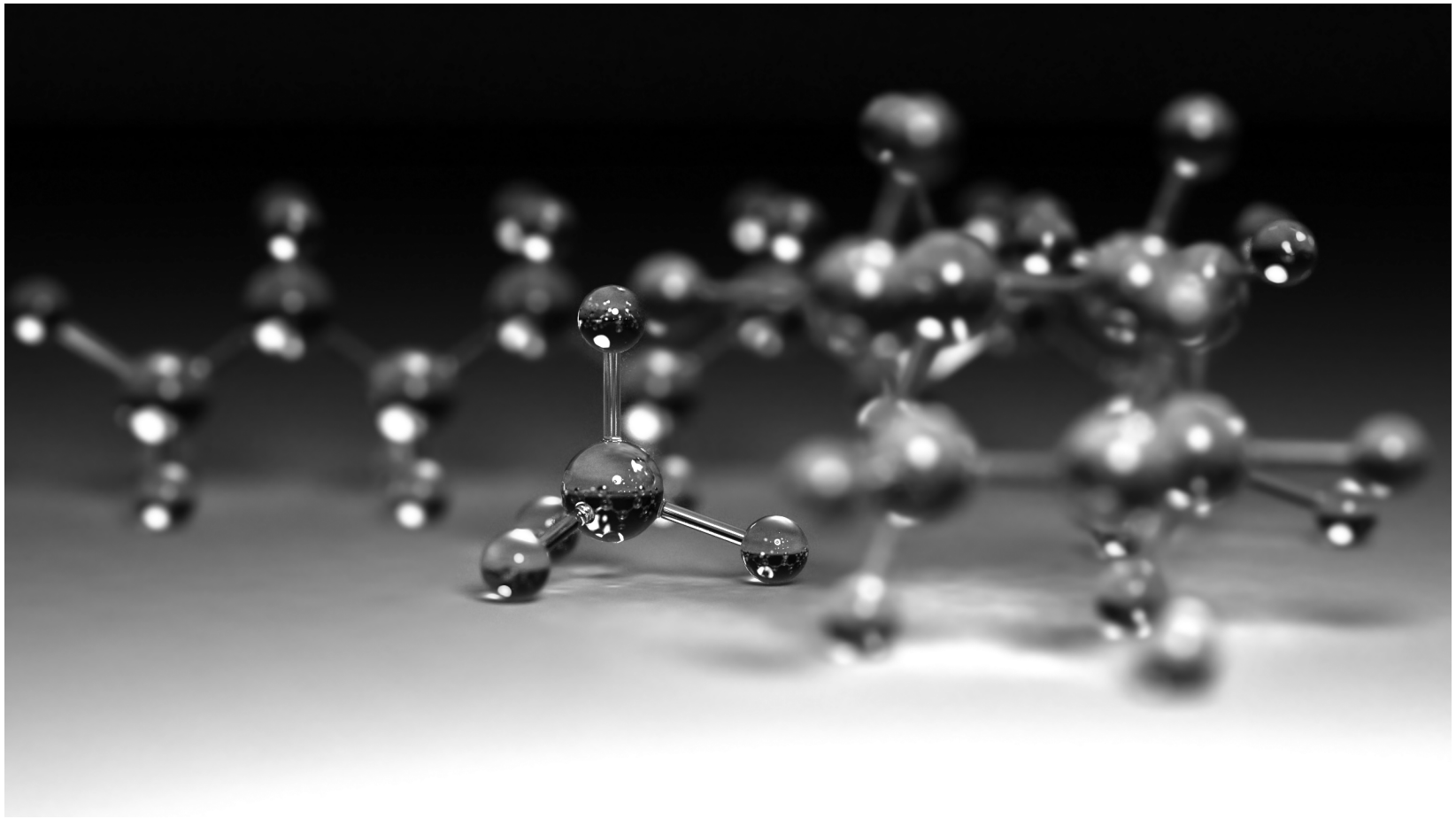
Trazado de rayos

- Tiempo de cálculo:

$$tinterseccion * npixels * (nobjetos * nsecundarios * nluces)^{nrebotes}$$

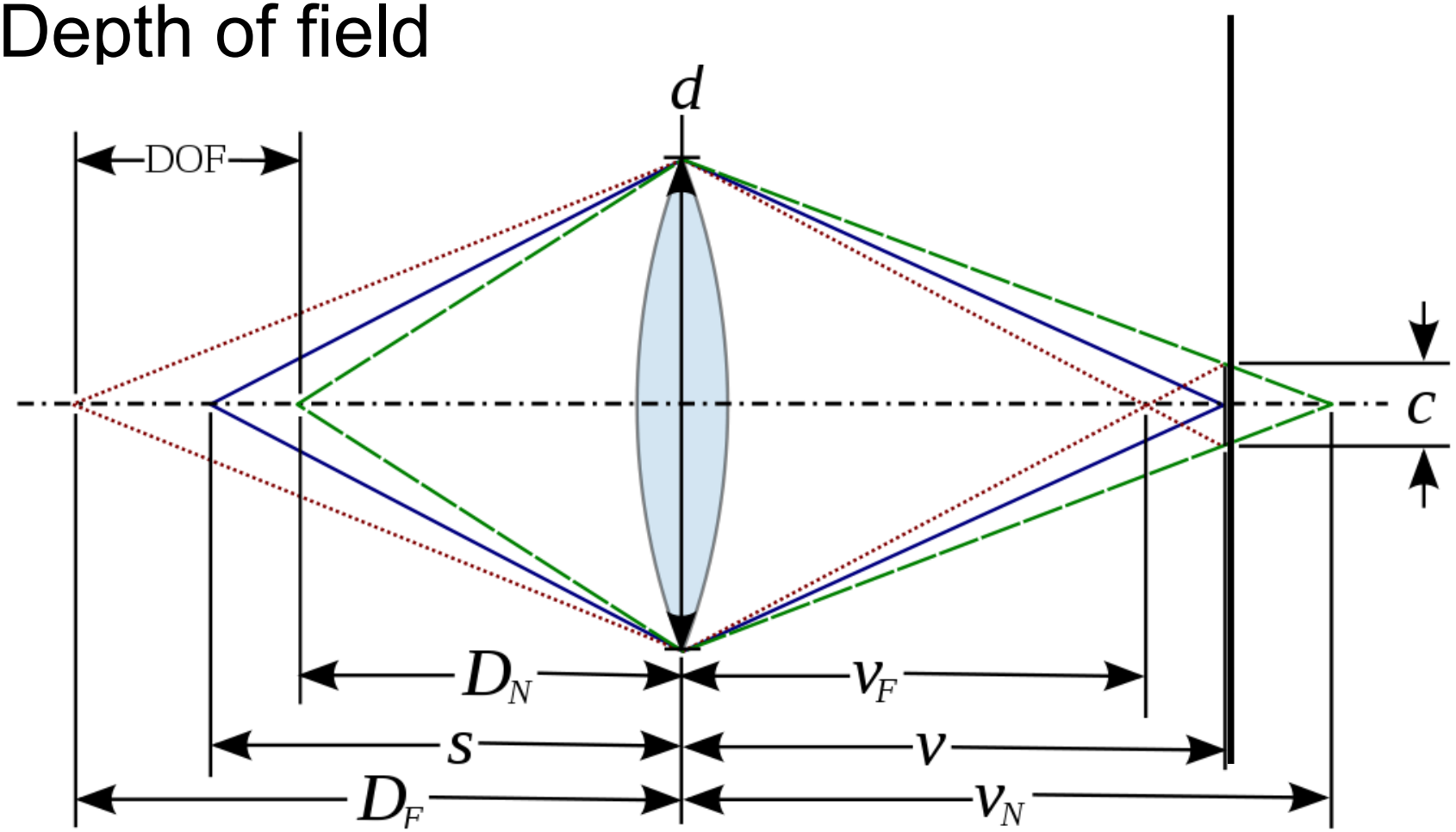
Trazado de rayos: ¿qué falta?

- Depth of field



Trazado de rayos: ¿qué falta?

- Depth of field



Graphics and Imaging Lab GIGA

Trazado de rayos: ¿qué falta?

- Depth of field
  - Efectos chulos y realistas.
  - Se aumenta mucho el tiempo de cálculo (muchos rayos por píxel para simular el fenómeno).
  - Como siempre: es mejor idea elegir los ”más importantes”.

Trazado de rayos: ¿qué falta?

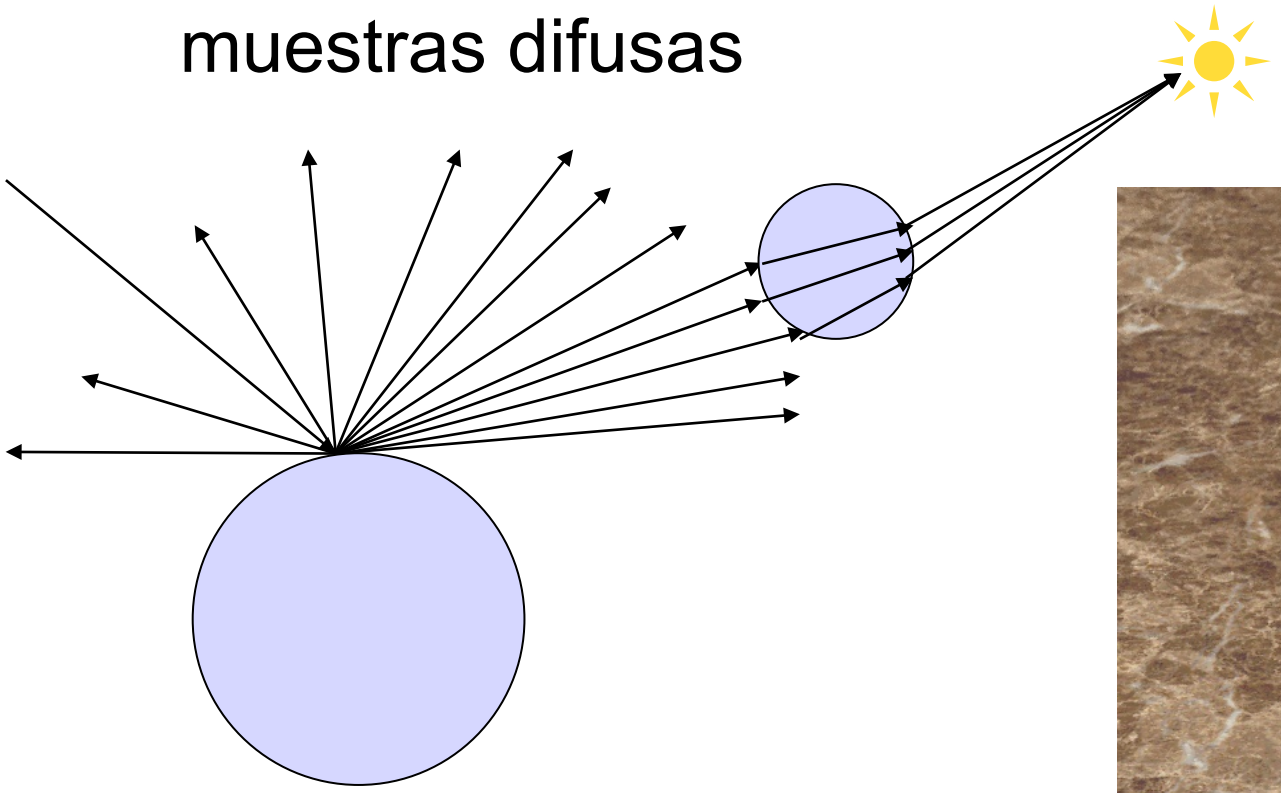
- Motion blur

Trazado de rayos: ¿qué falta?

- Motion blur:
  - Integral de la radiancia a lo largo del tiempo de exposición.
  - Coger varias muestras en ese tiempo: aumentar el tiempo de cálculo.

Trazado de rayos: ¿qué falta?

- Cáusticas
  - Caminos difíciles de encontrar, requieren muchas muestras difusas



Graphics and Imaging Lab GIGA

Trazado de rayos

- Tiempo de cálculo:

$$tinterseccion*npixels*ndof*nmotionblur*(nobjetos*nsecundarios*nluces)^{nrebotes}$$

Además, nsecundarios necesita ser muy grande (tirando a infinito) para encontrar cáusticas y fenómenos similares

Nuevos algoritmos

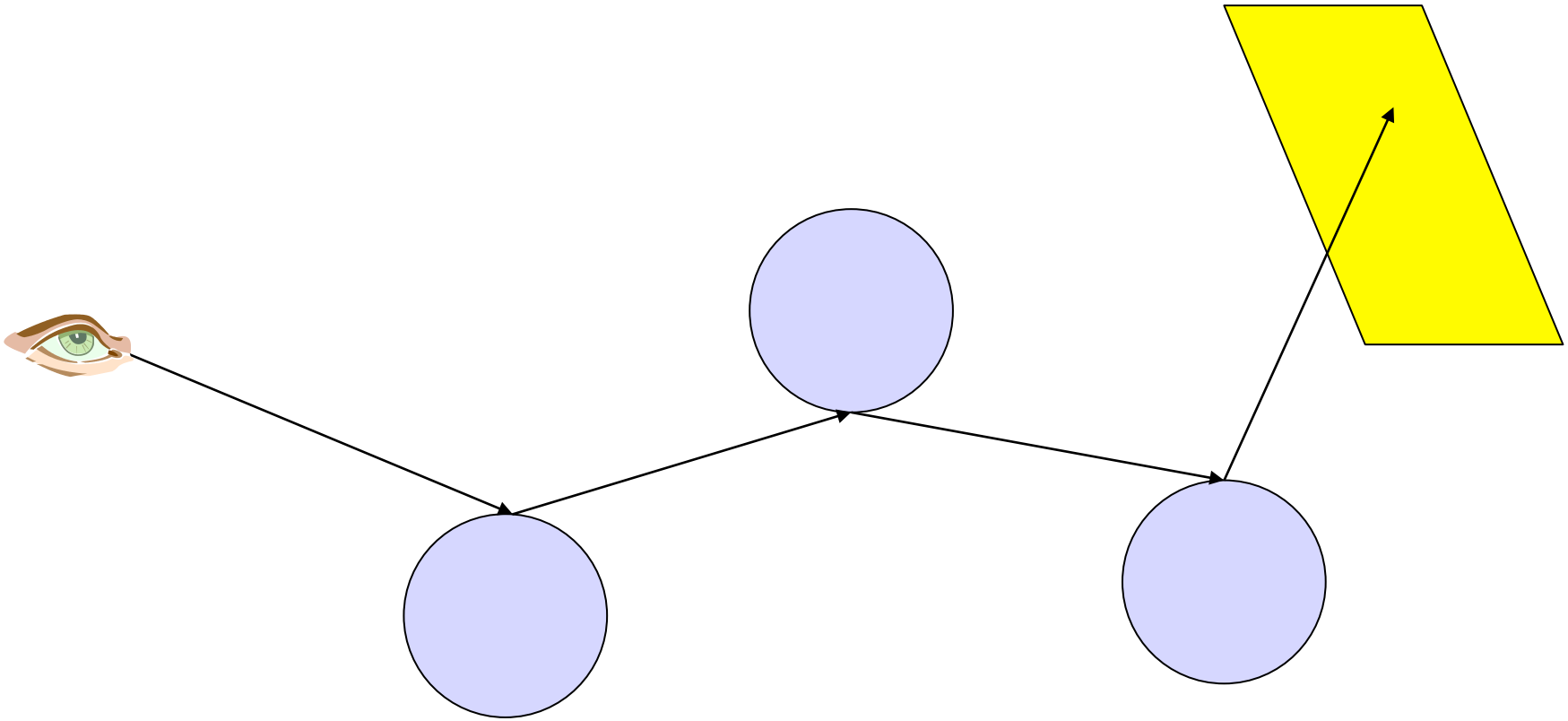
- Trazado de rayos necesita mucho tiempo, hay varios fenómenos que no son prácticos de simular.
- Cada nuevo fenómeno añadido, introduce mucha complejidad computacional en el algoritmo.

Nuevos algoritmos

- Nuestra ”unidad de medida” es el rayo.
- Idea: que nuestra ”unidad de medida” sea un camino entero de la cámara a la luz: *path-tracing*

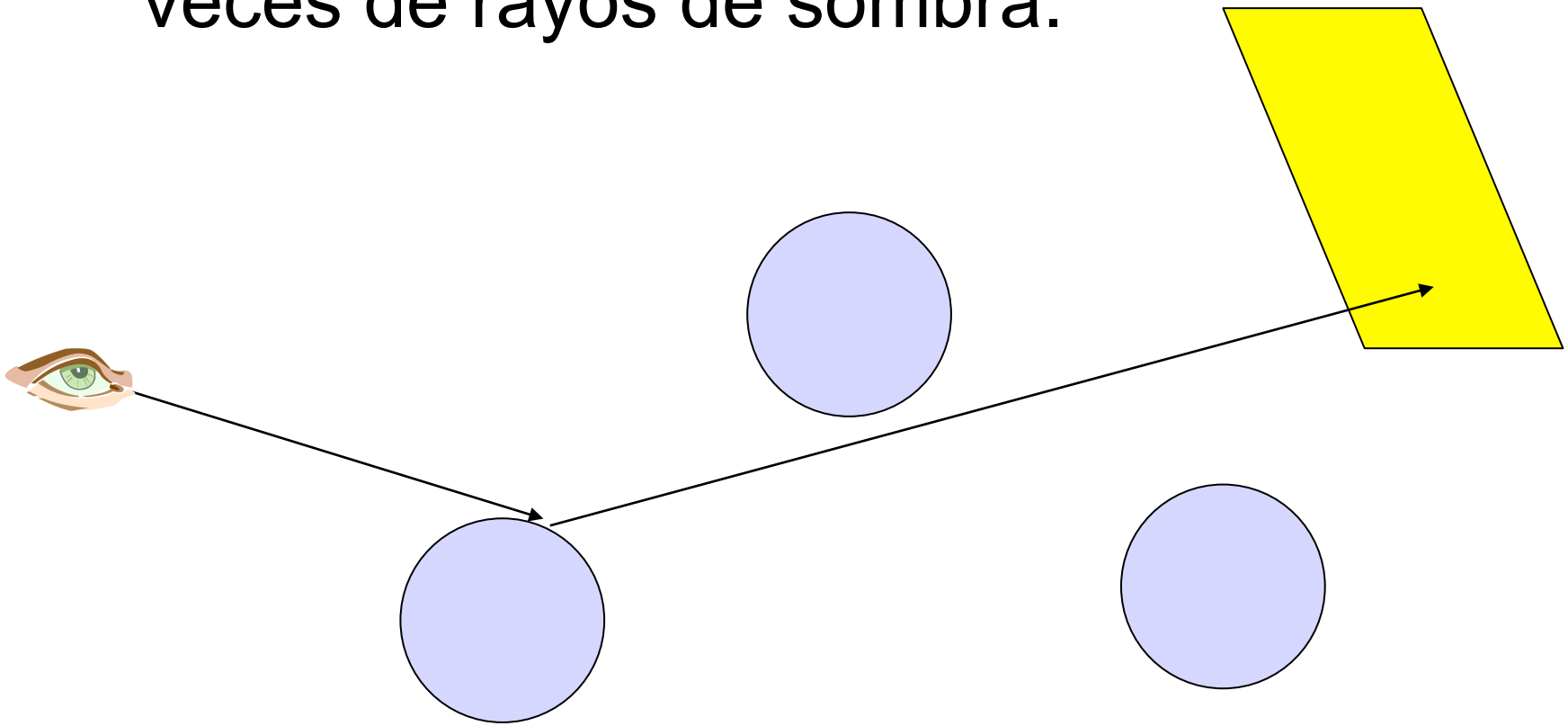
Path tracing

- Se pierde el concepto de ”rayo secundario”.



Path tracing

- Los propios rayos de un camino hacen las veces de rayos de sombra.



Path tracing

- ¿Qué pasa con el tiempo de cálculo?

$$t_{interseccion} * npixels * ndof * nmotionblur * (nobjetos * nsecundarios * nluces)^{nrebotes}$$

Path tracing

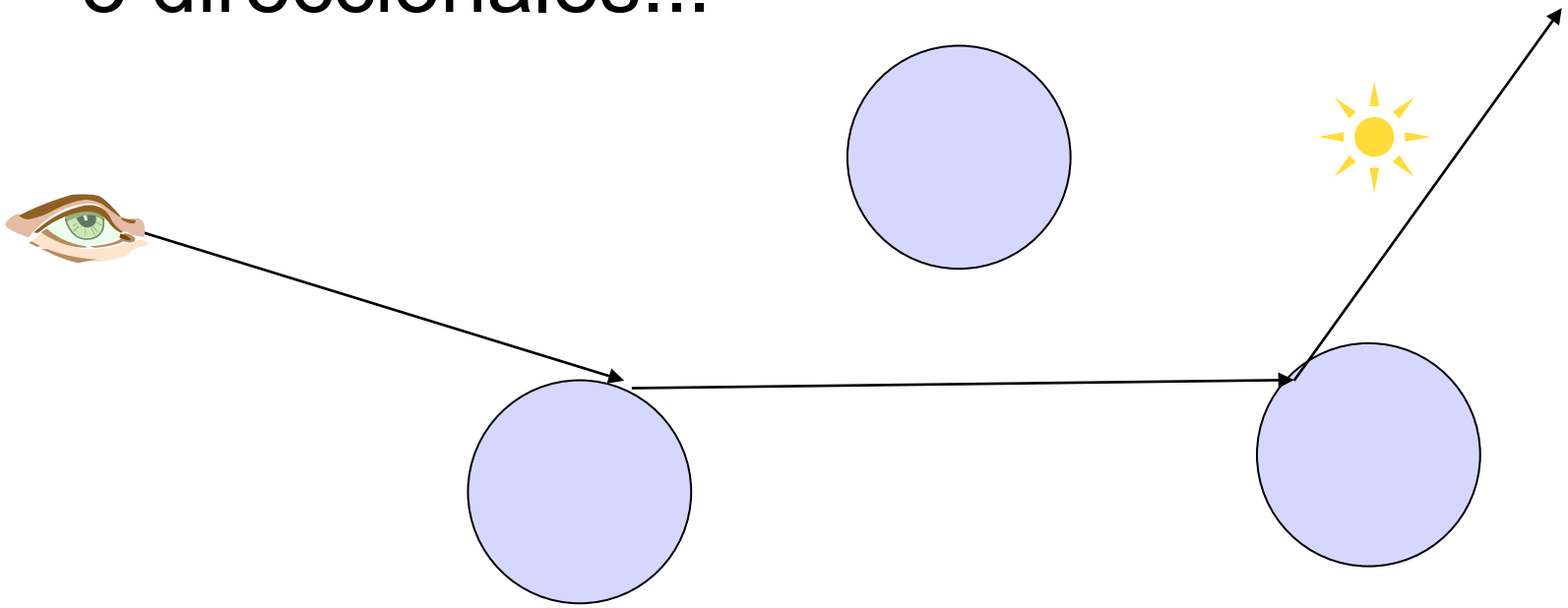
¿QUÉ OPINAS?

Path tracing

Ray tracing

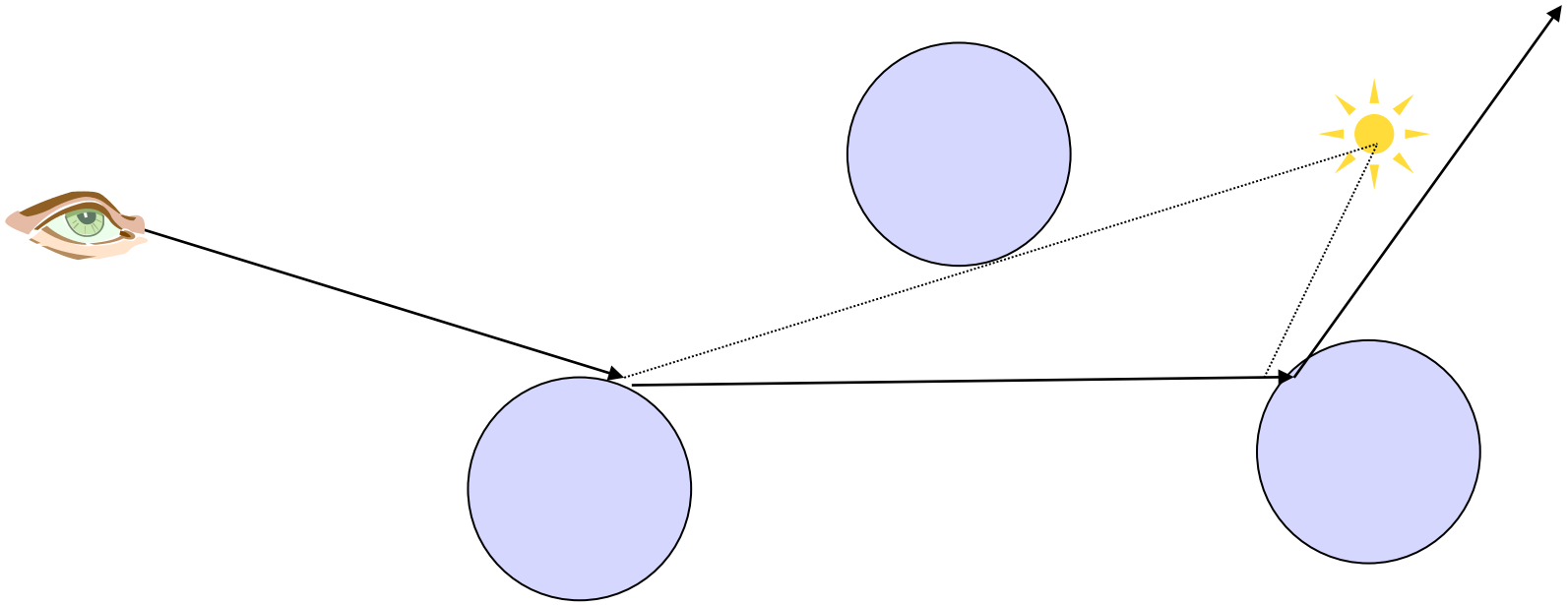
Path tracing

- La implementación clásica de path tracing tiene problemas localizando fuentes de luz puntuales o direccionales...



Path tracing, next event estimation

- Idea: calcular el ”siguiente evento” (trazar rayos de sombra)

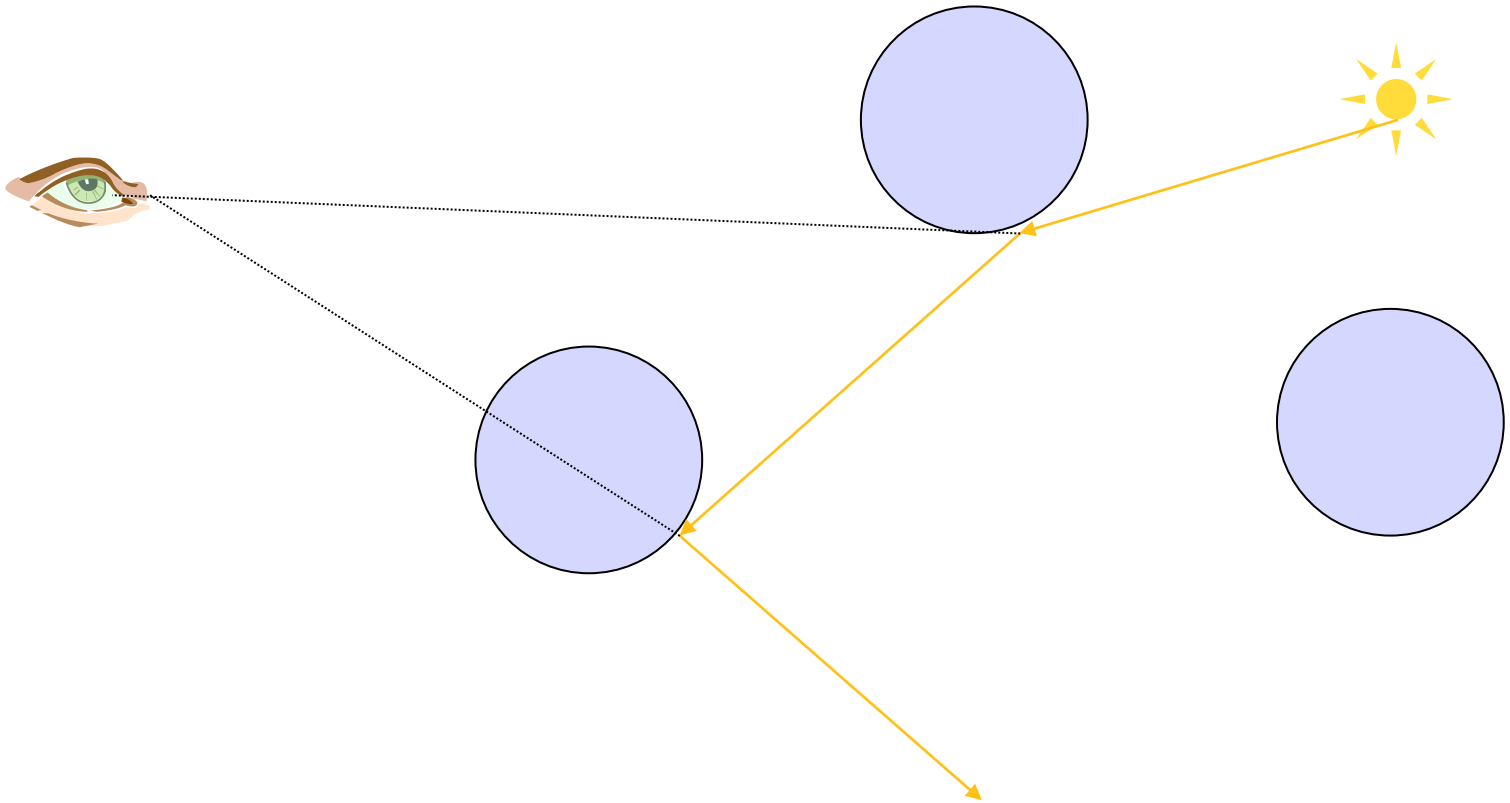


Path tracing, next event estimation

- … pero sigue teniendo problemas con las cáusticas.

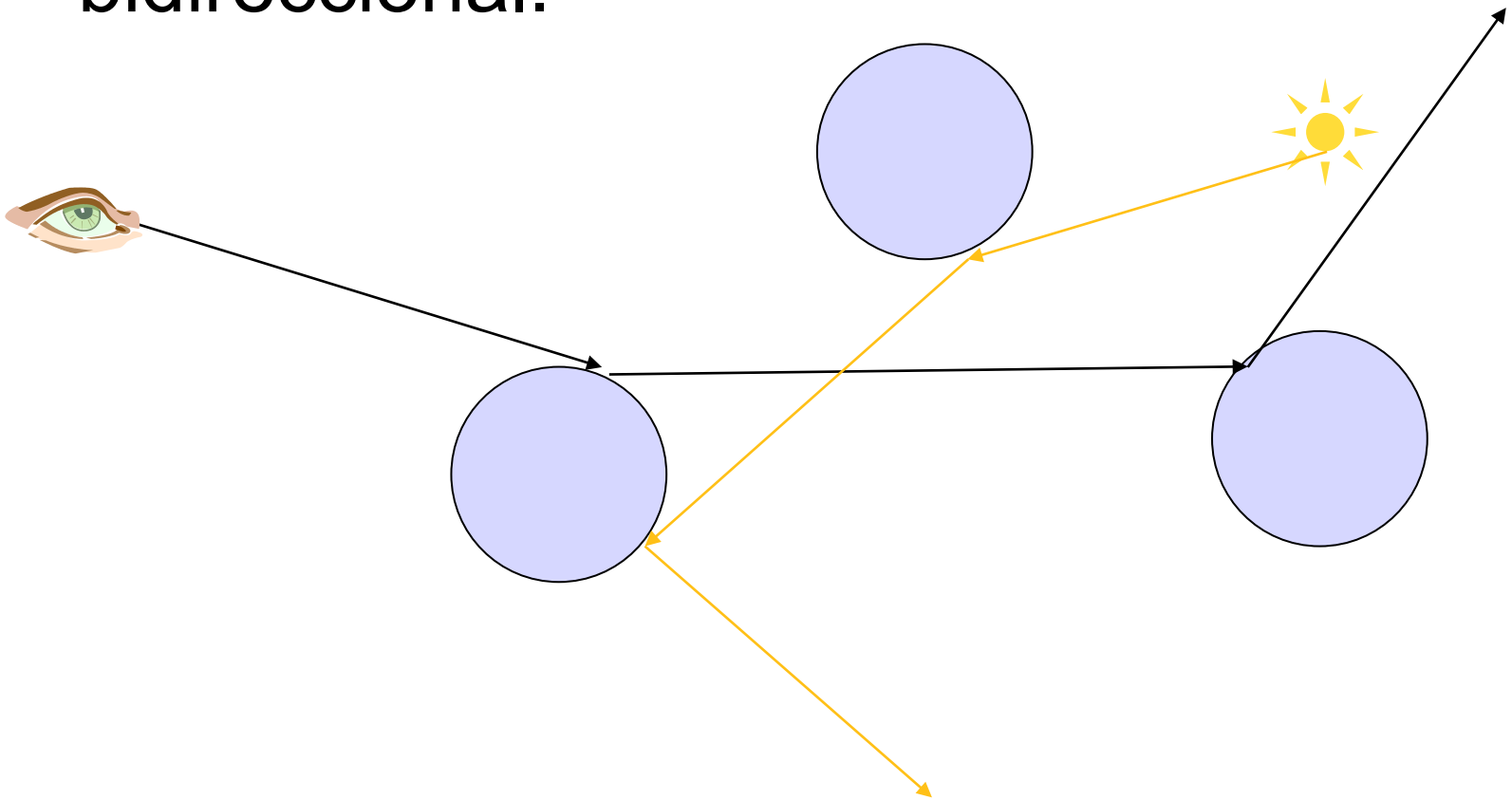
Light tracing

- Idea: Comenzar caminos desde la luz.
- Con siguiente evento: buscar la cámara.



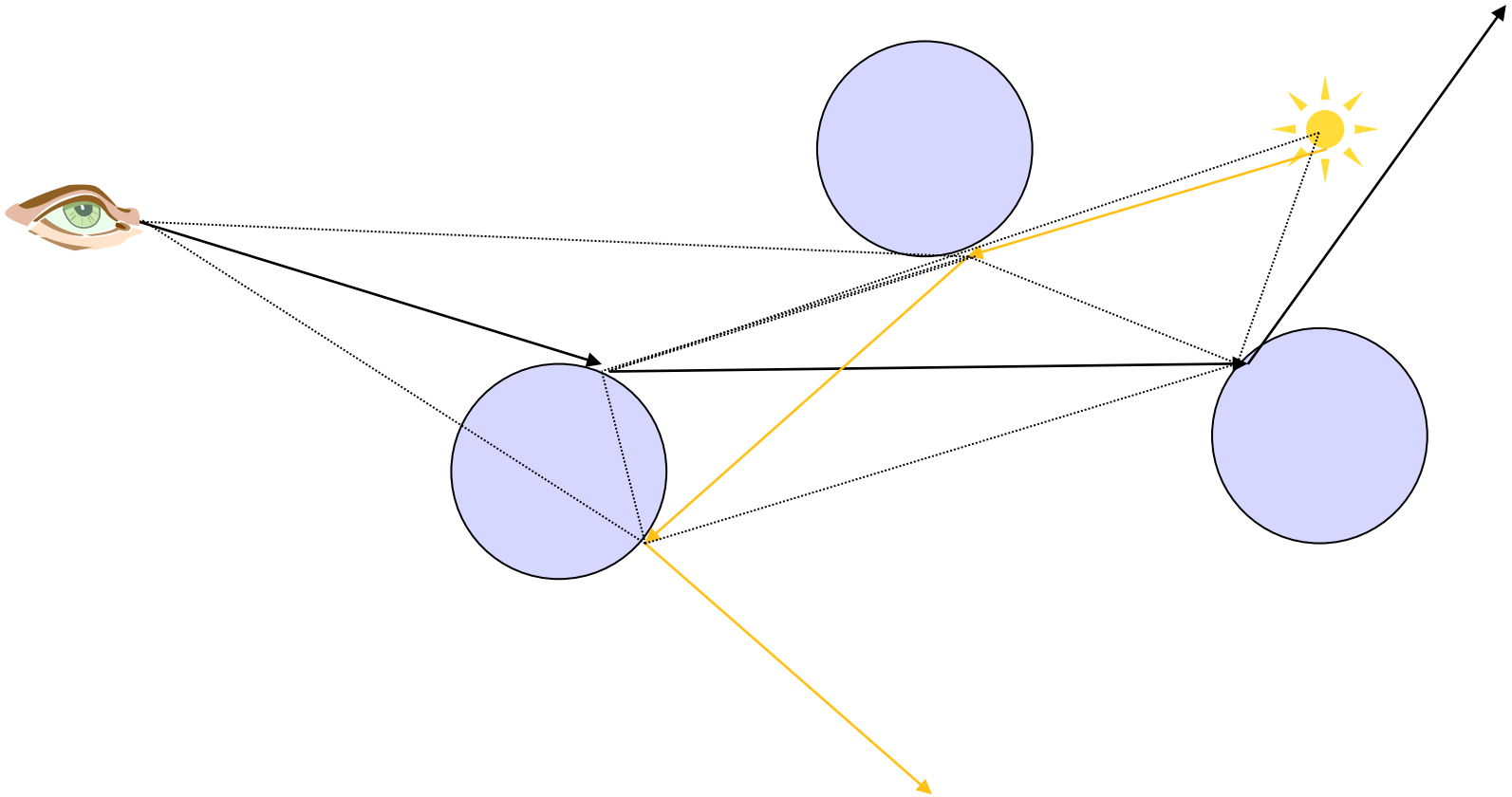
Path tracing bidireccional

- Idea: comenzar caminos tanto desde la cámara como desde las luces: path tracing bidireccional.



Path tracing bidireccional.

- Luego se trazan rayos de sombra.



Path tracing bidireccional

- ¿Qué pasa con el tiempo de cálculo?

tinterseccion*npixels*ndof*nmotionblur*
(nobjetos*nsecundarios*nluces)^nrebotes

Path tracing bidireccional

¿QUÉ OPINAS?

Path tracing

Ray tracing

Path tracing bidireccional

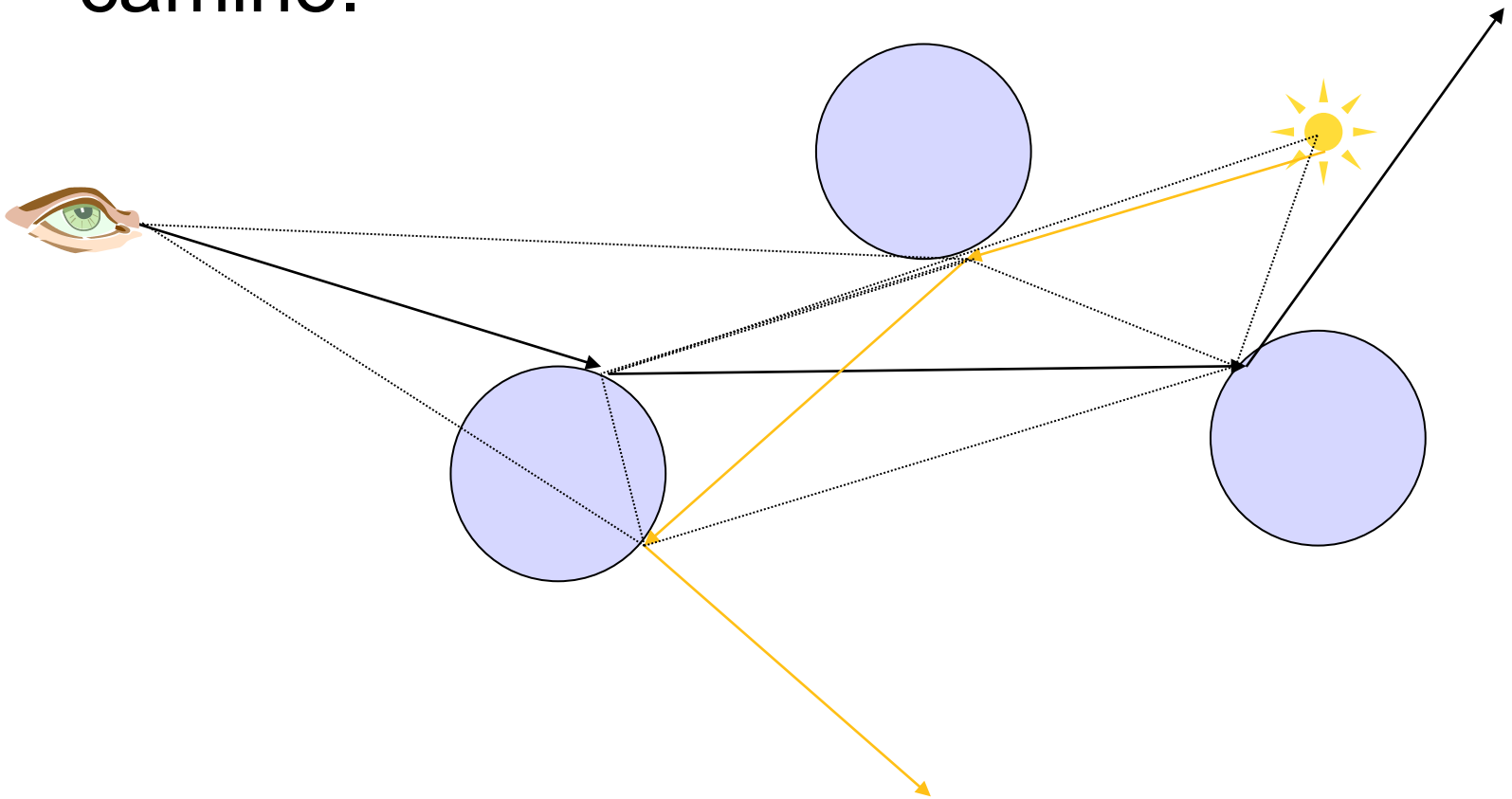
Graphics and Imaging Lab GIGA

Path tracing bidireccional

- Hacen falta muchos rayos por píxel.
- Convergencia lenta, tiempo de cálculo elevado.
- Muy dependiente de Montecarlo en cada rebote.

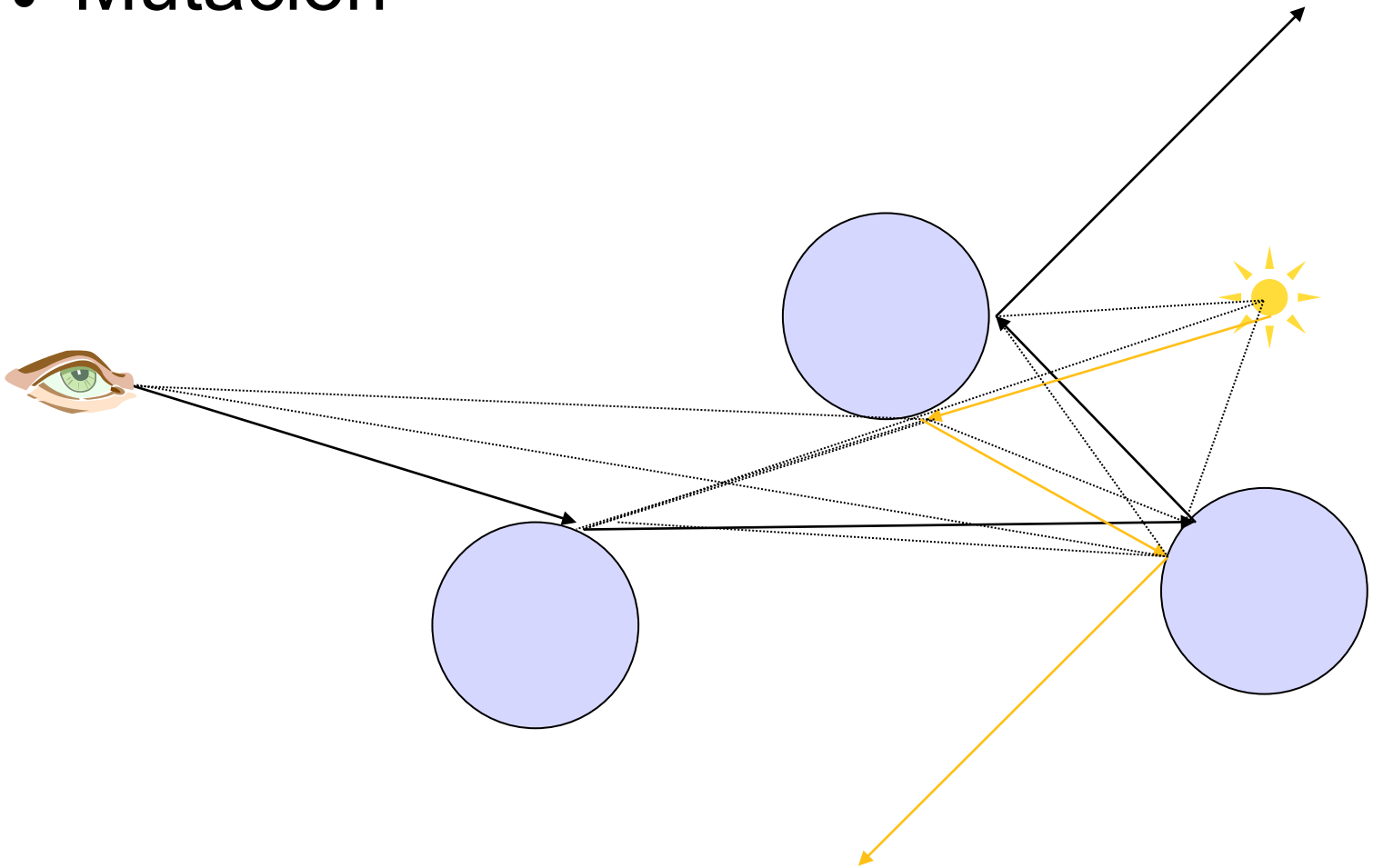
Metropolis Light Transport

- Idea: al localizar un camino importante, lo reaprovechamos (mutándolo) para el siguiente camino.



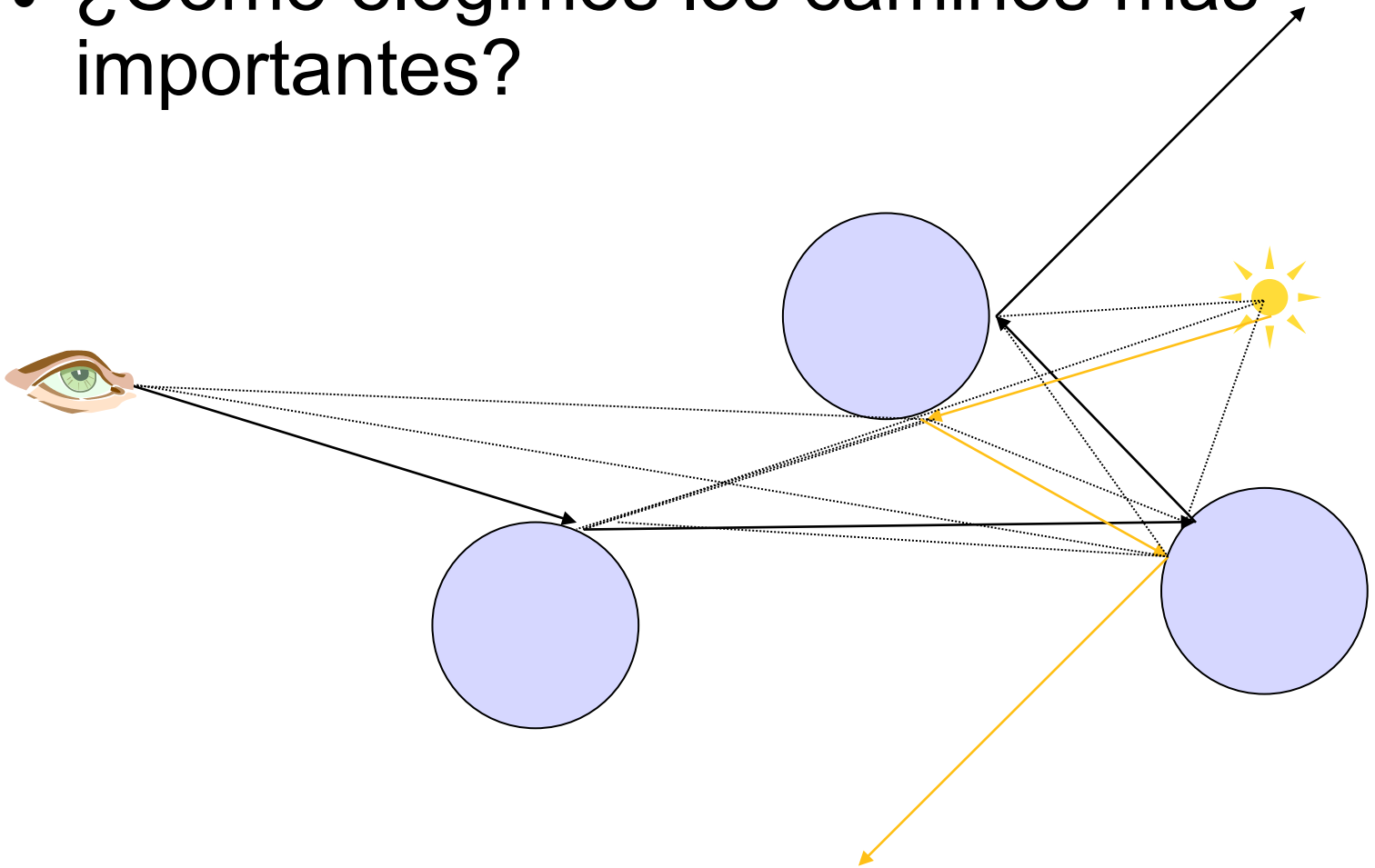
Metropolis Light Transport

- Mutación



Metropolis Light Transport

- ¿Cómo elegimos los caminos más importantes?

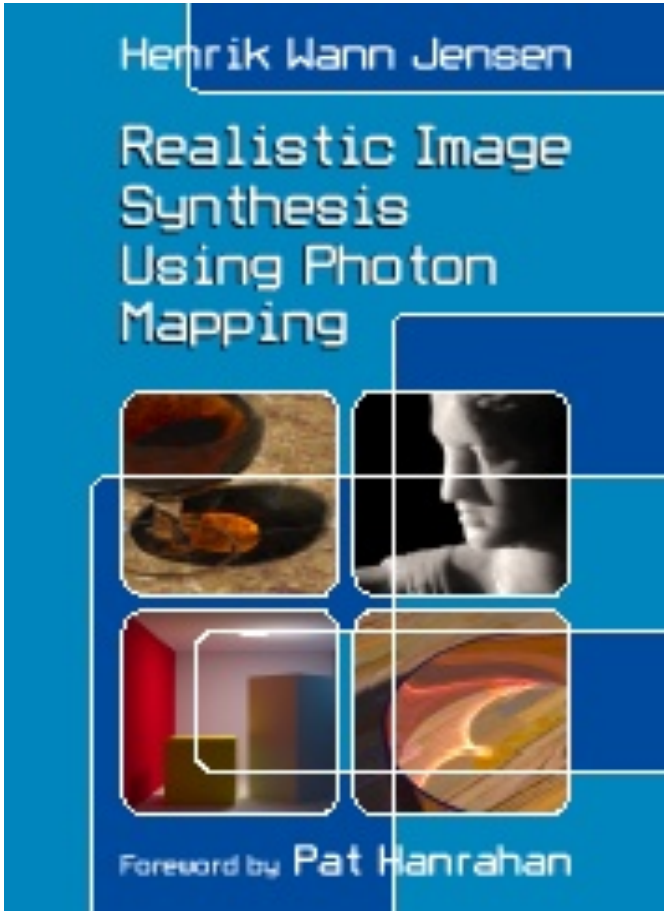


Graphics and Imaging Lab GIGA

Metropolis Light Transport

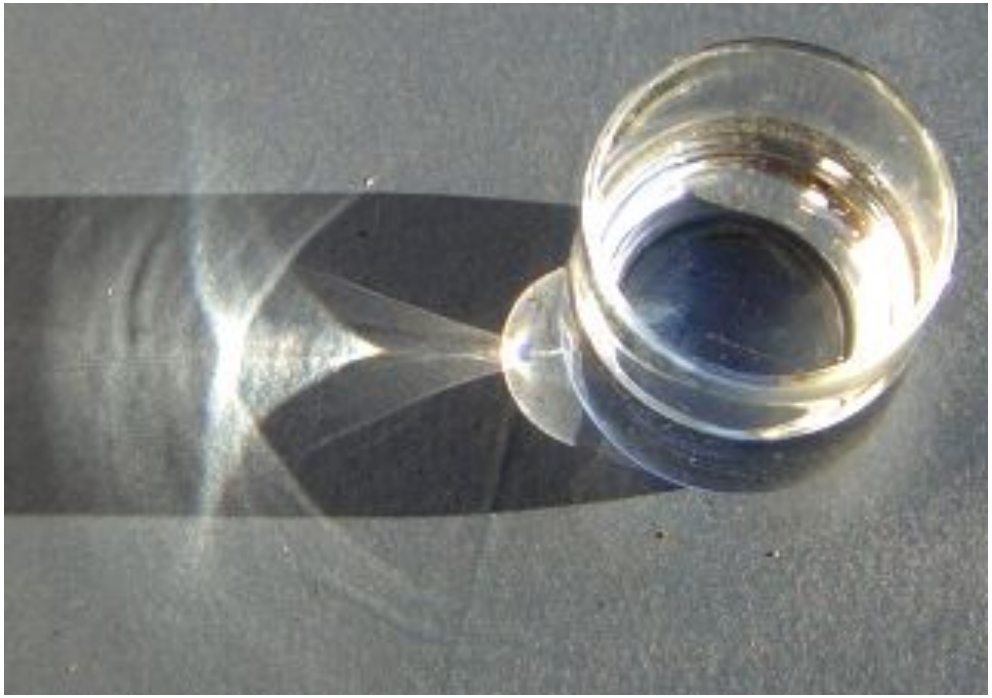
- Convergencia ligeramente superior al path tracing bidireccional clásico en muchos casos.

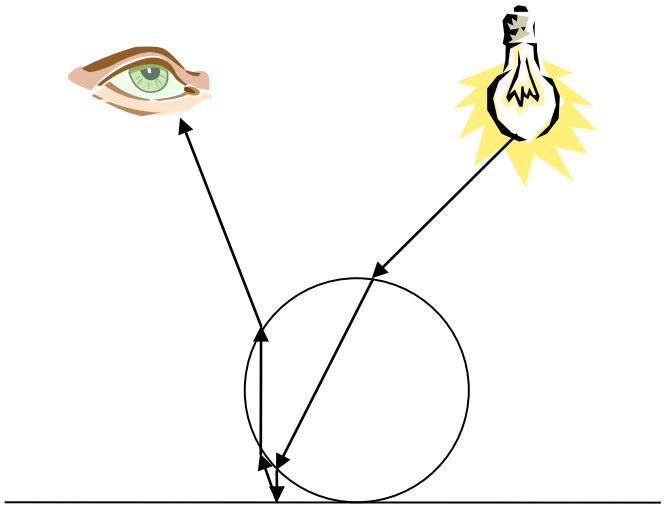
Mapeado de fotones



¿Para qué?

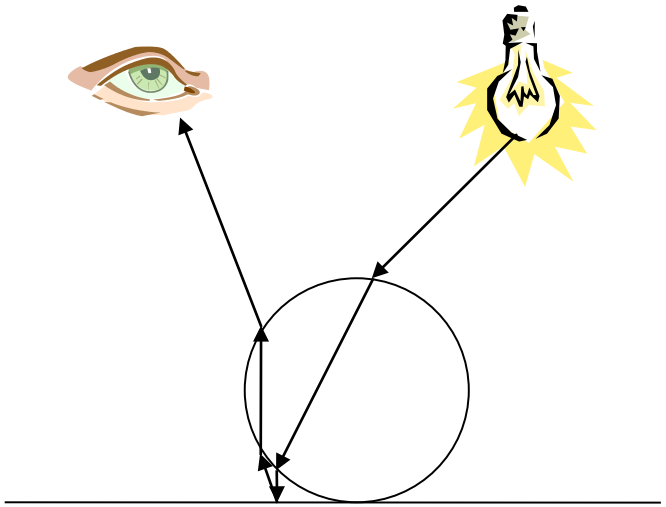
- Existen fenómenos visuales complicados que son costosos de simular con algoritmos de iluminación global convencionales.





¿Para qué?

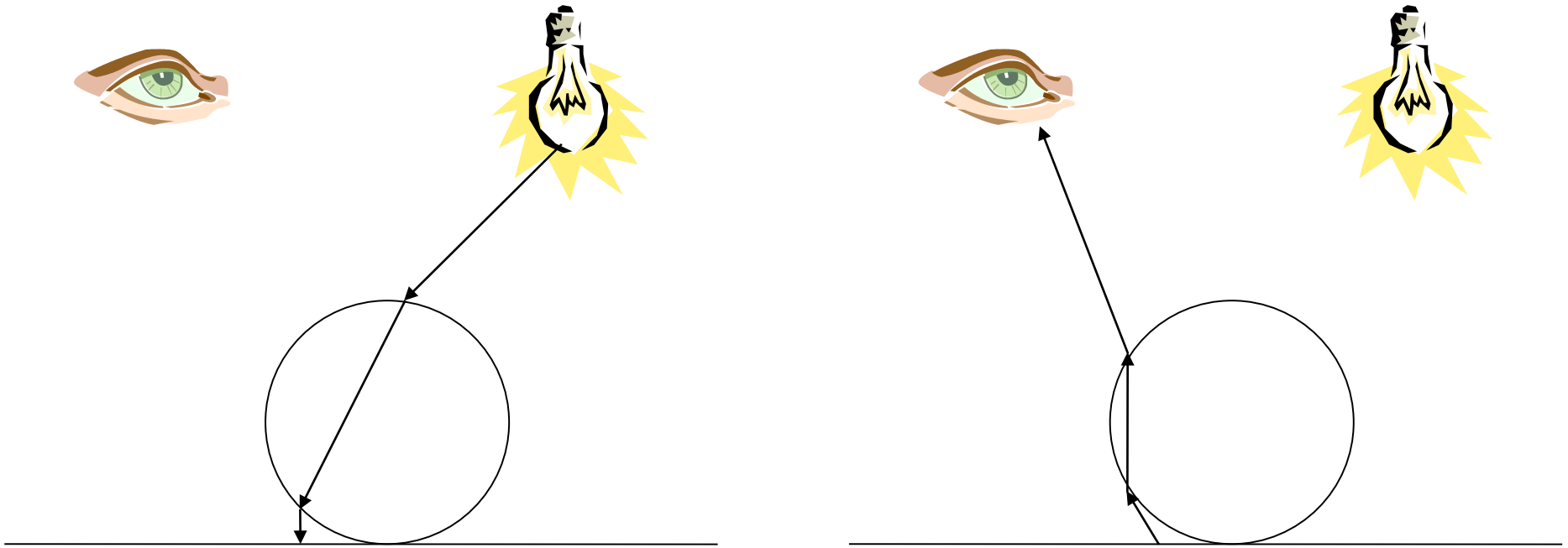
- En escenas complicadas es prácticamente imposible:
  - Path tracing requiere ”mucha suerte” para la cáustica.
  - Path tracing bidireccional y Metropolis requieren ”mucha suerte” para la reflexión de la cáustica



Graphics and Imaging Lab GIGA

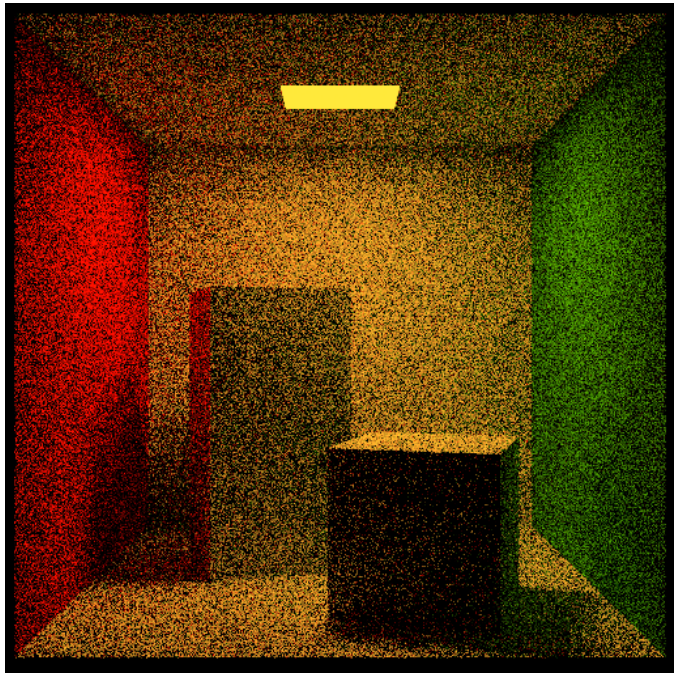
¿Para qué?

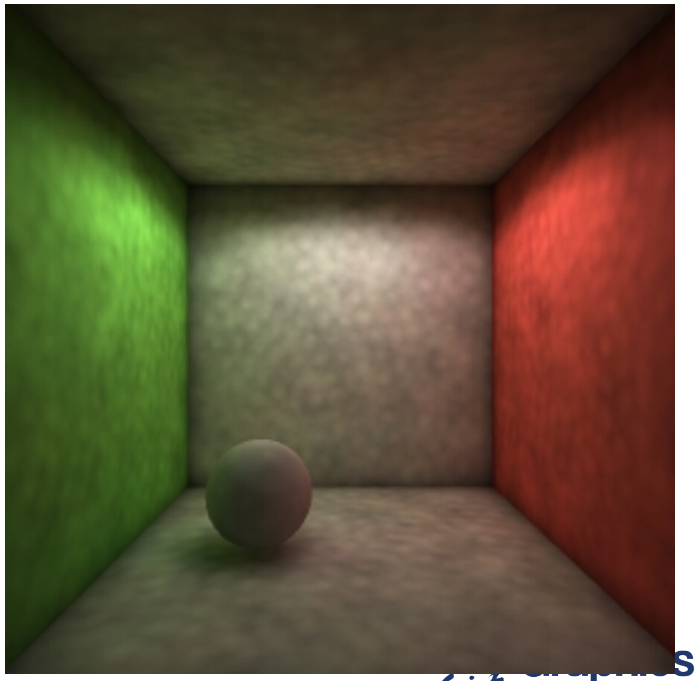
- Idea: desacoplar los dos ”caminos”



¿Para qué?

- Otras ventajas:
  - Tiempo de cálculo menor
  - Ruido de baja frecuencia
- Desventajas
  - Sesgo





Índice

- Algoritmo
- Estructura de datos
- Mejoras
- Resultados

Índice

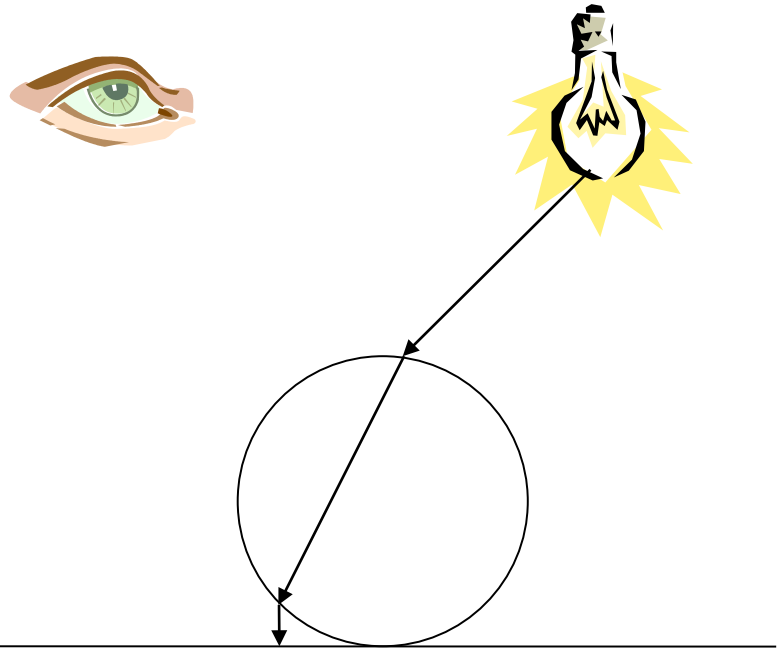
- **Algoritmo**
- Estructura de datos
- Mejoras
- Resultados

Algoritmo

1. Se lanzan fotones desde las fuentes de luz
   - Se almacenan en una estructura de datos.
2. Se trazan rayos desde la cámara
   - Se estima la radiancia a partir de los fotones almacenados.

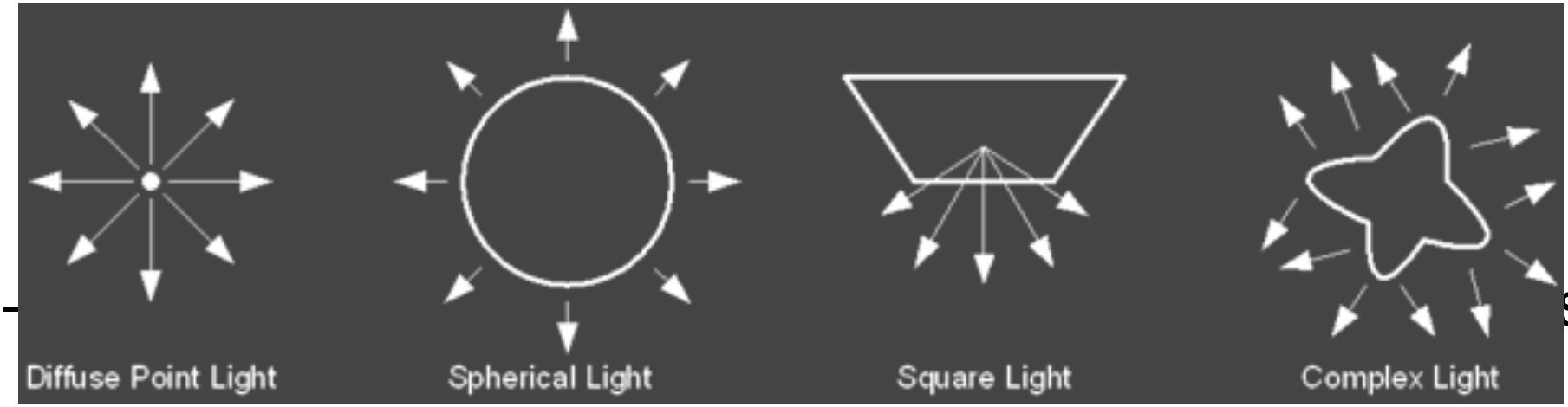
Algoritmo

- Fase 1: Trazado de fotones
  - Emisión
  - Interacción
  - Almacén



Algoritmo

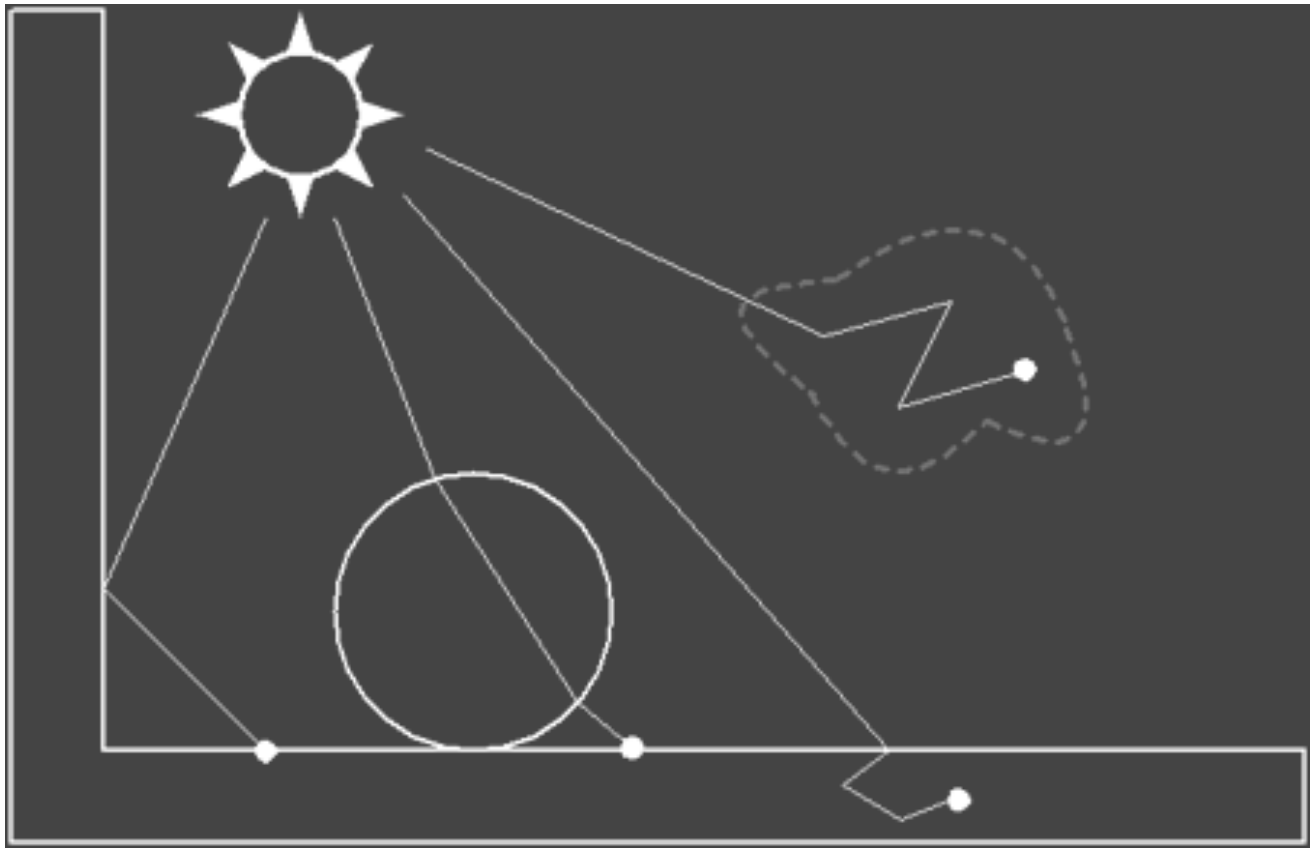
- Fase 1: Trazado de fotones
  - **Emisión**
  - Interacción
  - Almacén



s

Algoritmo

- Fase 1: Trazado de fotones
  - Emisión
  - **Interacción**
  - Almacén

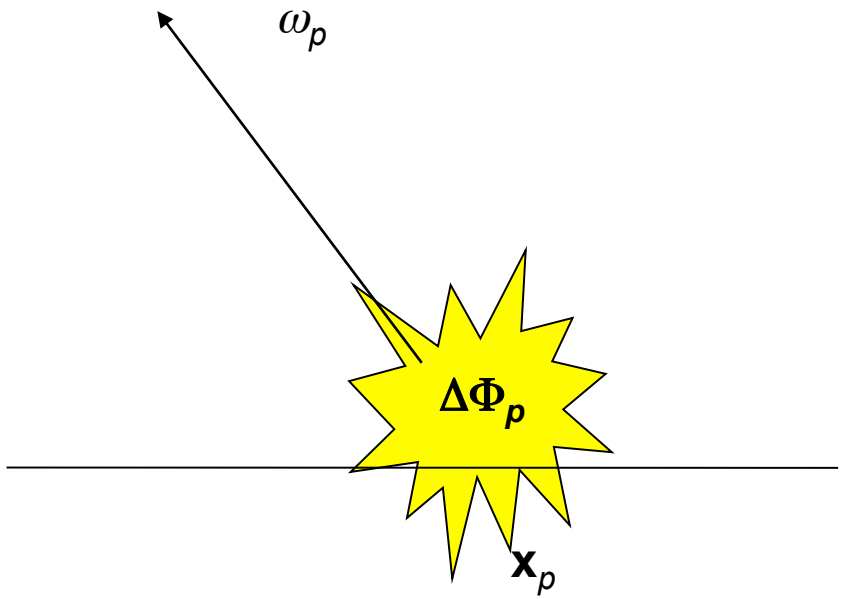


Algoritmo

- Fase 1: Trazado de fotones
  - Emisión
  - Interacción
  - **Almacén**
    - Que permita búsquedas de vecinos más cercanos

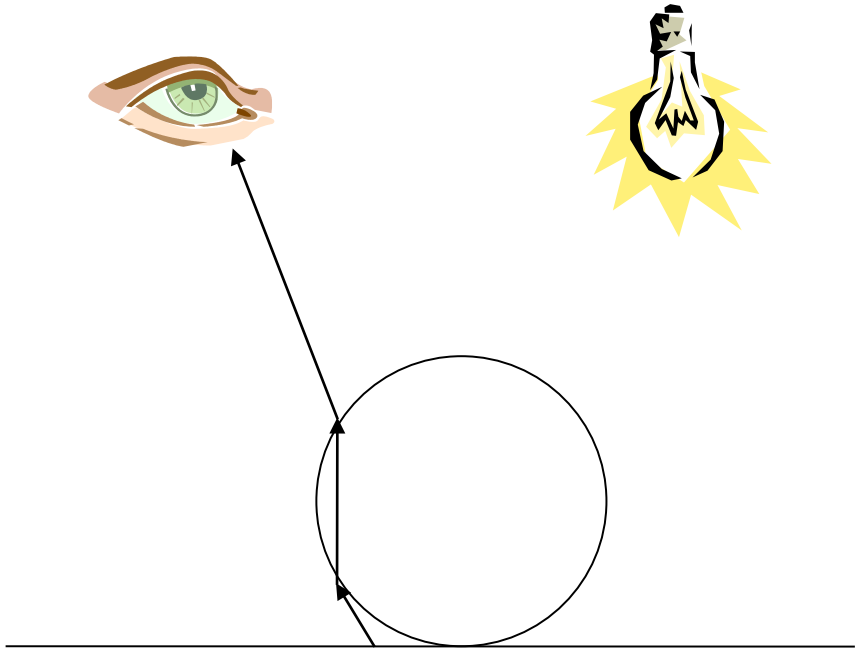
Algoritmo

- Un fotón
  - Flujo: $\Delta\Phi_p$ (W - color)
  - Posición: $\mathbf{x}_p$
  - Dirección: ω_p



Algoritmo

- Fase 2: Trazado de rayos
  - Trazado desde la cámara
  - Interacción
  - Estimación de radiancia



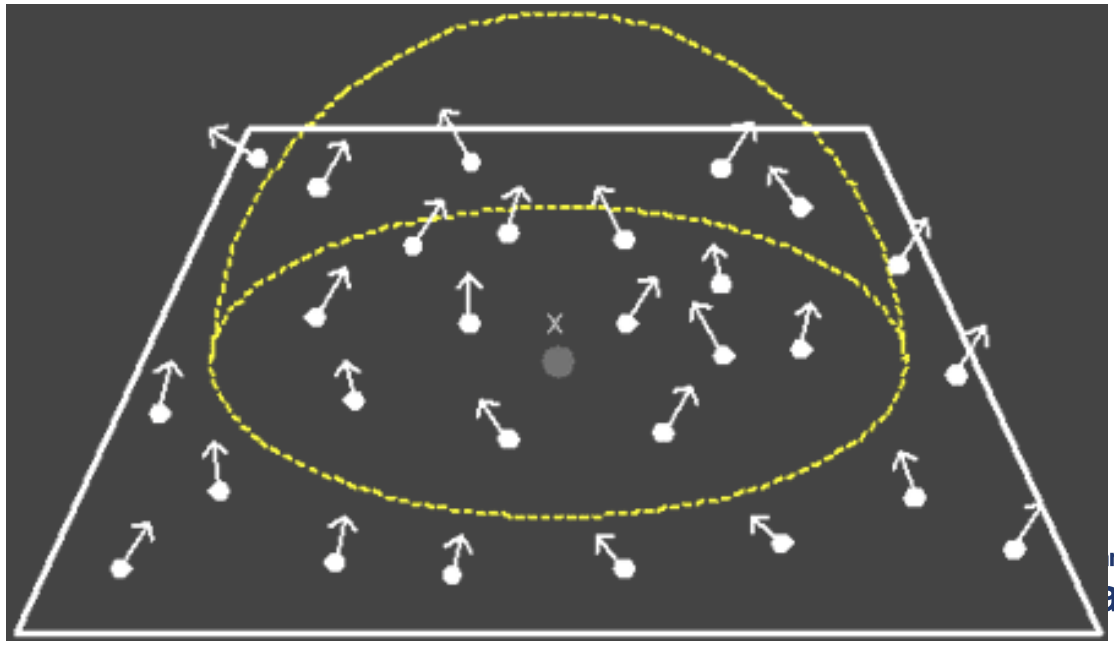
Algoritmo

- Fase 2: Trazado de rayos
  - **Trazado desde la cámara**
  - **Interacción**
  - Estimación de radiancia

  - **¡¡Igual que trazado de rayos estándar!!**

Algoritmo

- Fase 2: Trazado de rayos
  - Trazado desde la cámara
  - Interacción
  - **Estimación de radiancia**
    - N fotones más cercanos



Estimación de radiancia

- Ecuación de radiancia reflejada

$$L_r(\mathbf{x},\omega_r) = \int_{\Omega} f_r(\omega_i,\omega_r) L_i(\mathbf{x},\omega_i)(N\cdot\omega_i)d\omega_i$$

- Radiancia por flujo

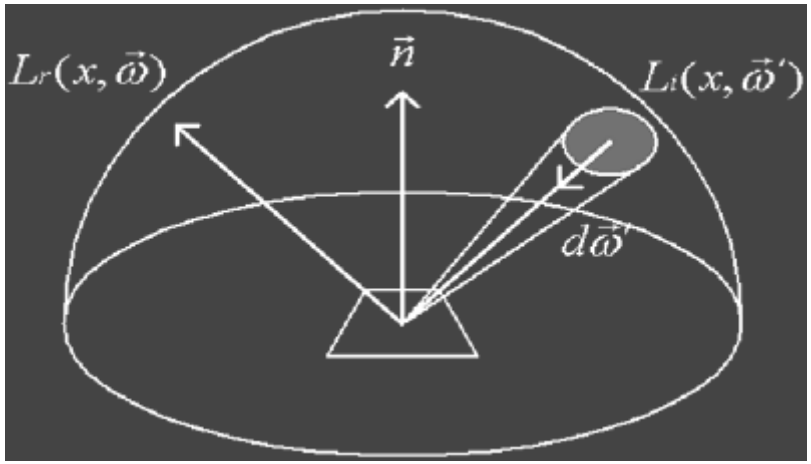
$$L_i(\mathbf{x},\omega_i) = \frac{d^2\Phi_i(\mathbf{x},\omega_i)}{(N\cdot\omega_i)d\omega_i dA_i}$$

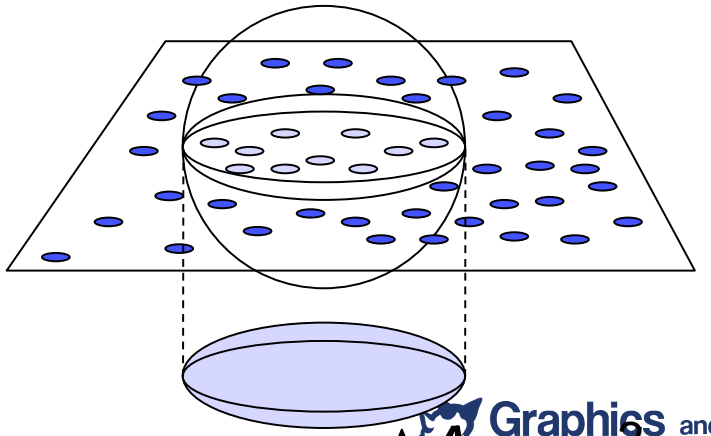
- Sustituimos

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

- Numéricamente

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



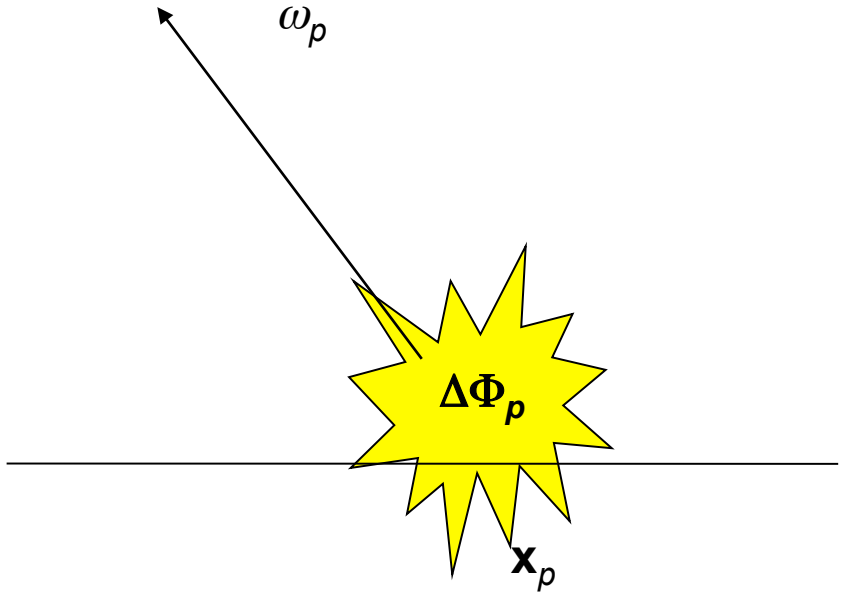


Índice

- Algoritmo
- **Estructura de datos**
- Mejoras
- Resultados

Estructura de datos

- Un fotón
  - Flujo: $\Delta\Phi_p$ (W - color)
  - Posición: $\mathbf{x}_p$
  - Dirección: ω_p



Estructura de datos

- Características:
  - Capaz de almacenar muchos datos
  - Capaz de búsquedas eficientes de los vecinos más cercanos (en 3D)

¿CUÁL ELEGIMOS?

Estructura de datos

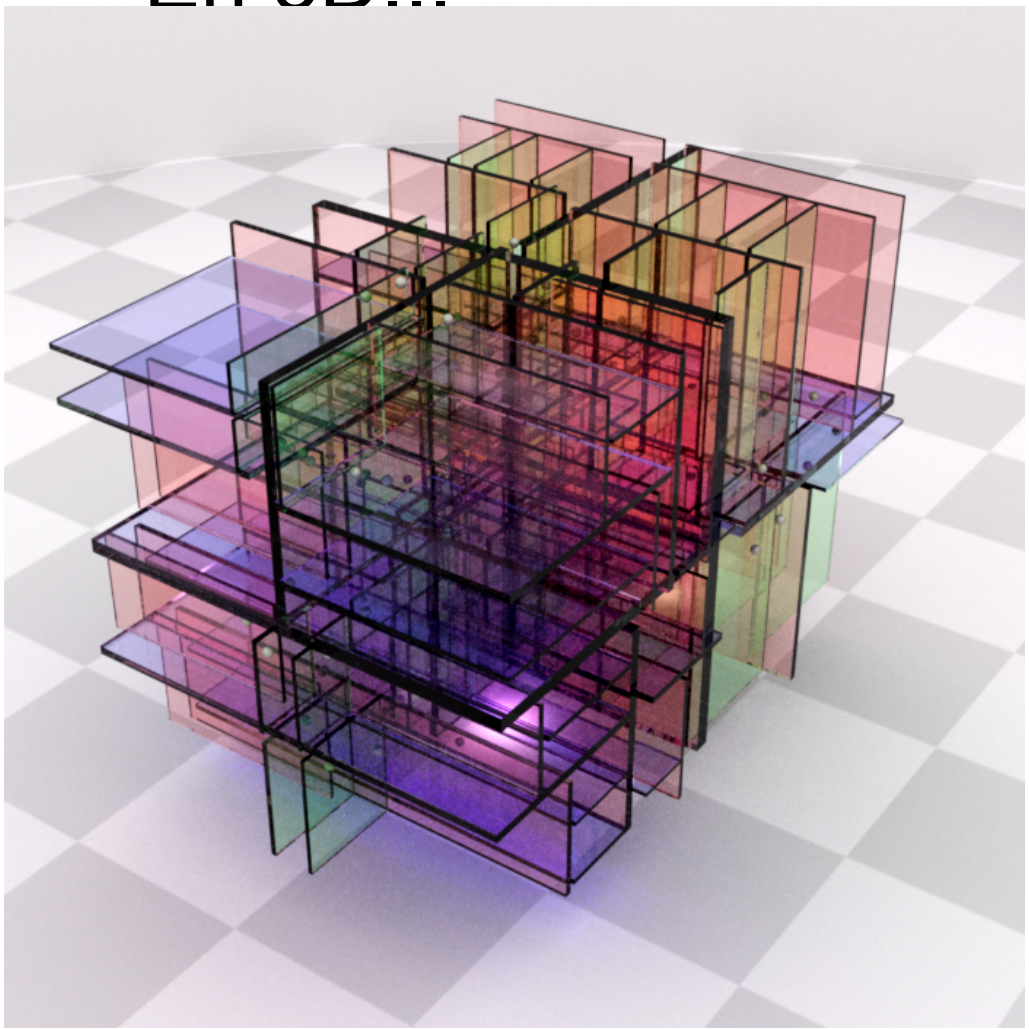
- Posibilidades
  - Diagrama de Voronoi
  - Voxels
  - Octree
  - Kd-tree

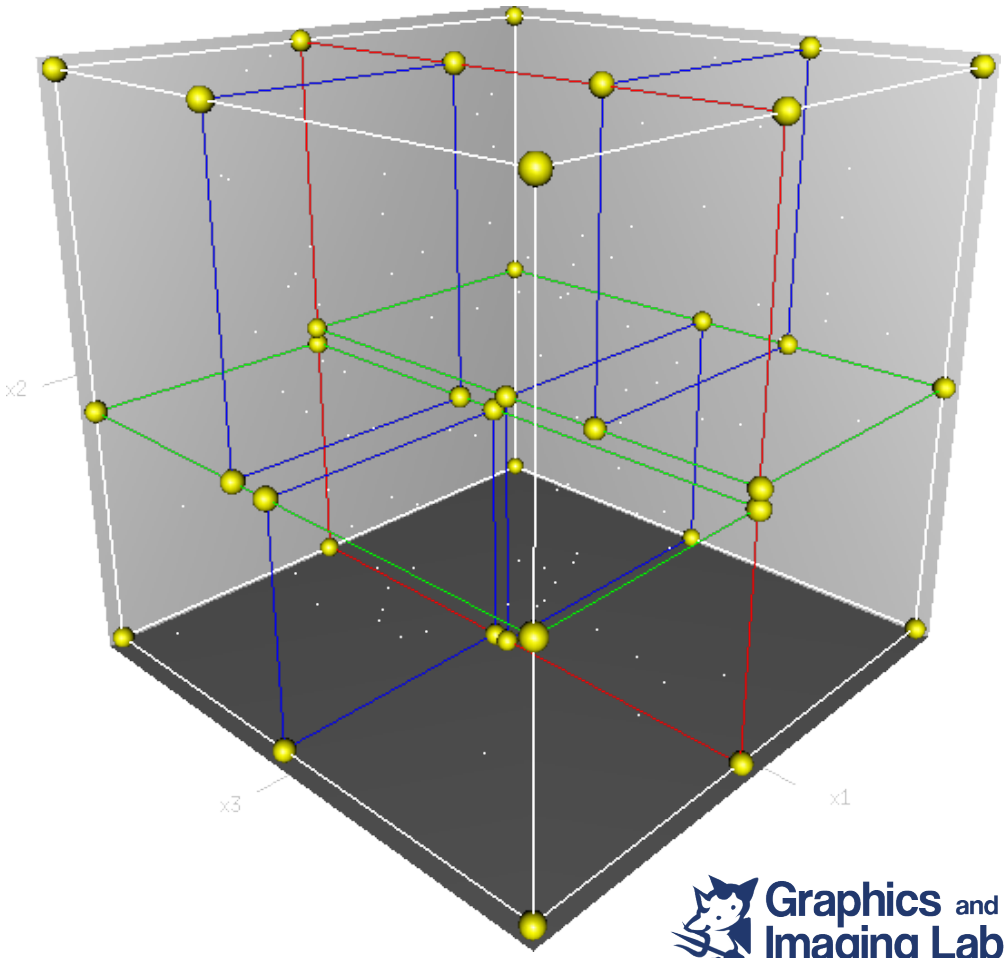
Kd-tree

- http://donar.umiacs.umd.edu/quadtree/points/kdtree.html
- http://homes.ieu.edu.tr/~hakcan/projects/kdtree/kdTree.html

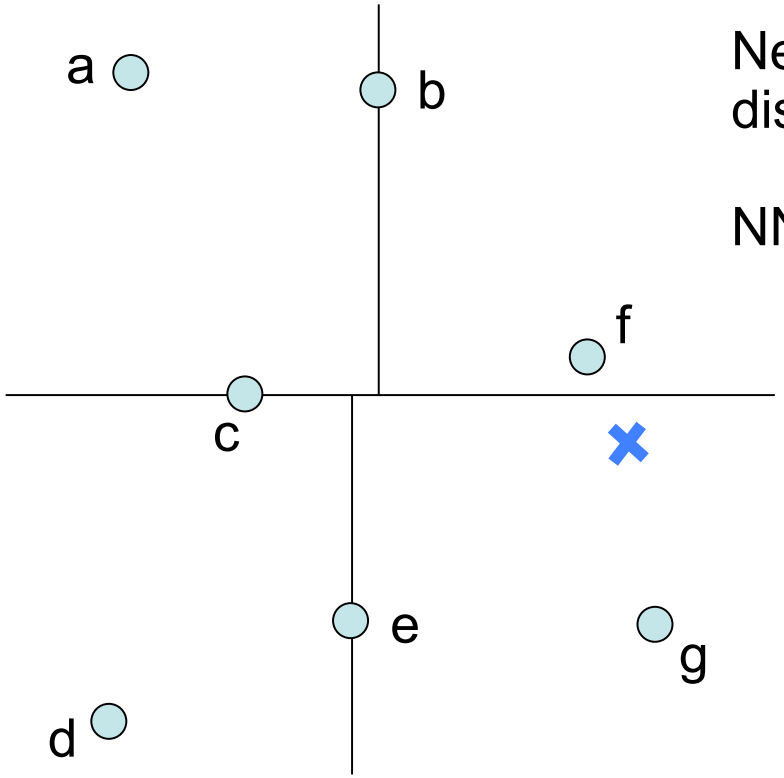
Kd-tree

- En 3D…



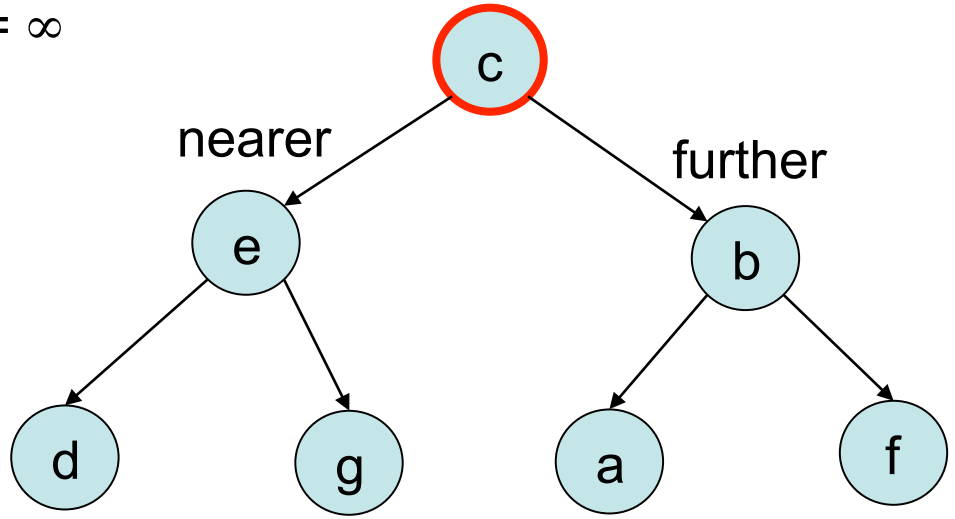


Graphics and Imaging Lab GIGA



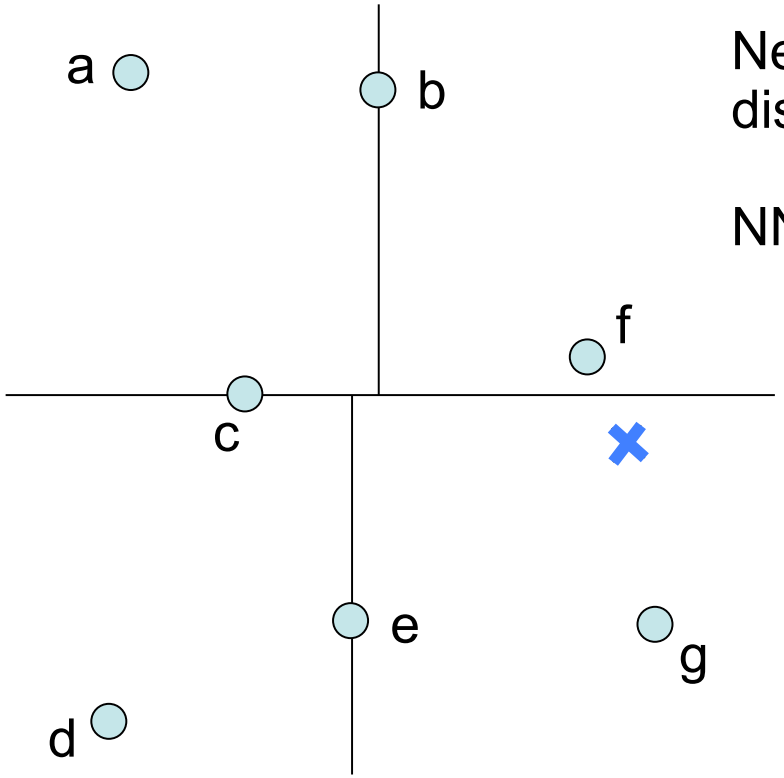
Nearest = ?
dist-sqd = ∞

NN(c, x)



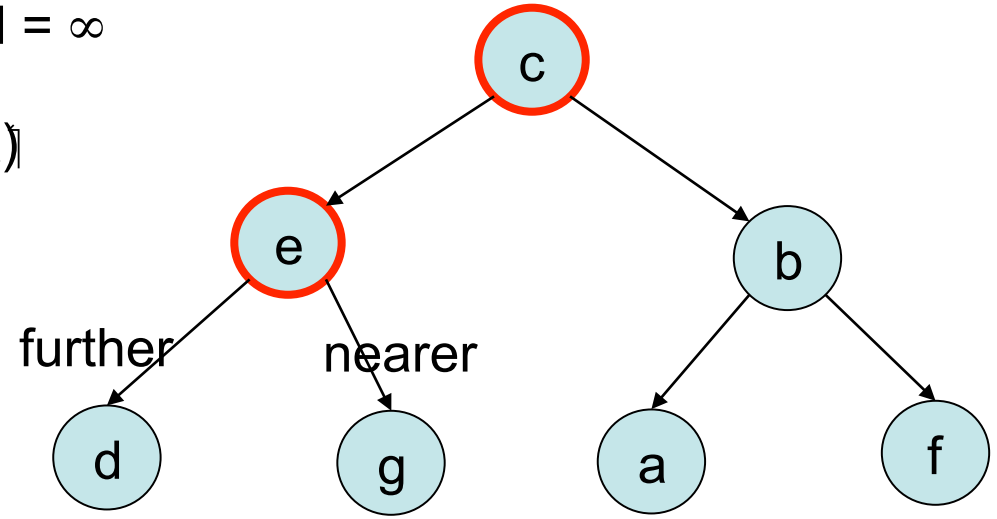
Nearer = e
Further = b

NN (e, x)



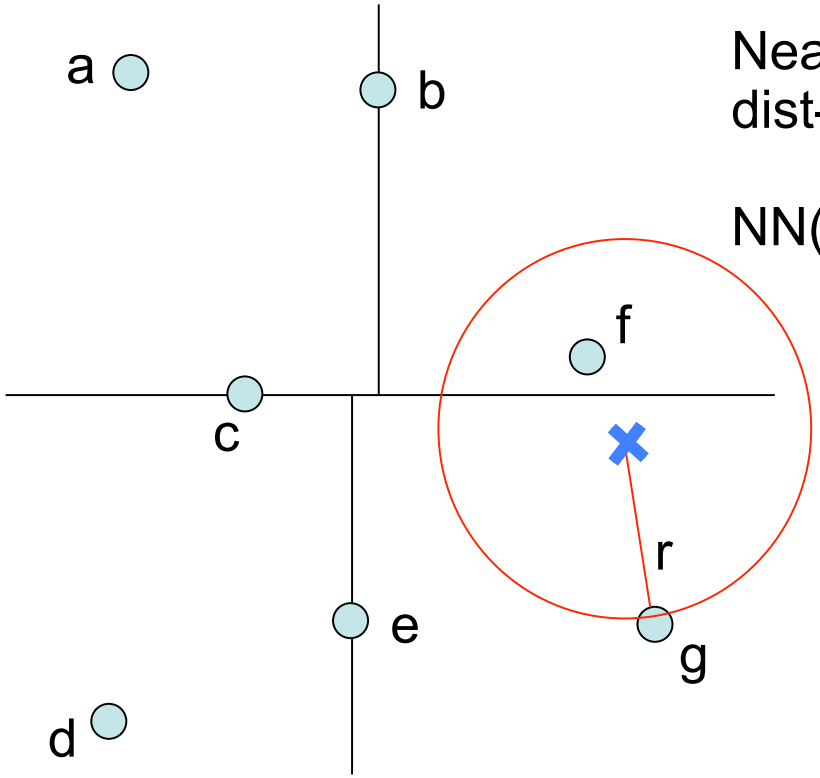
Nearest = ?
dist-sqd = ∞

NN(e, x)



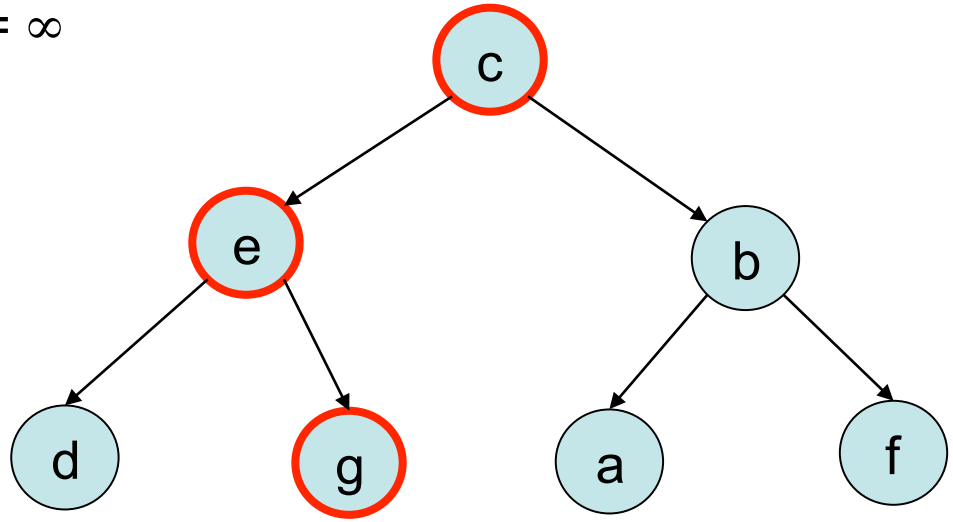
Nearer = g
Further = d

NN (g, x)

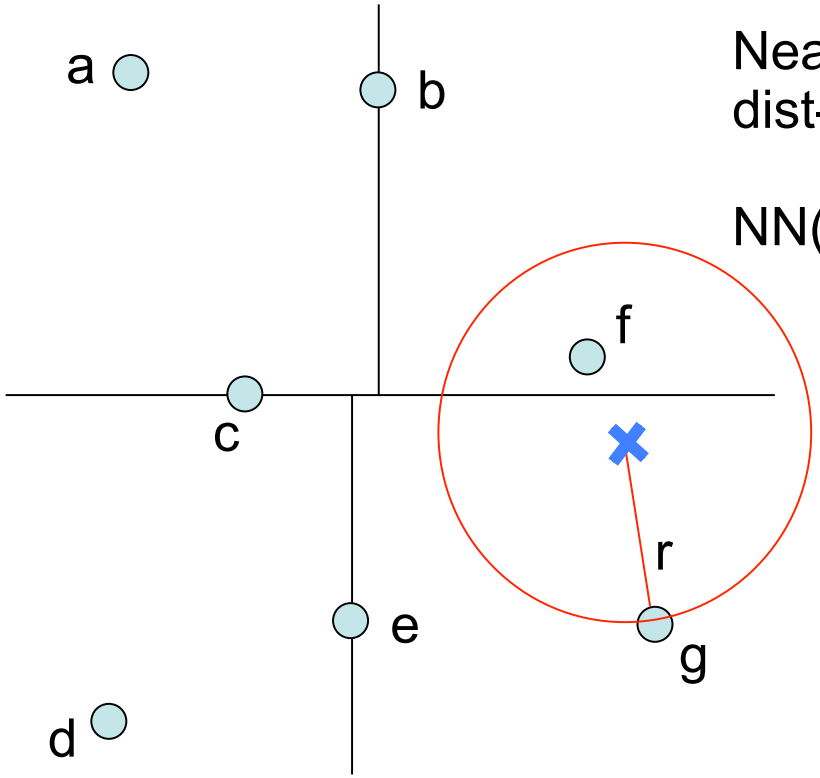


Nearest = ?
dist-sqd = ∞

NN(g, x)

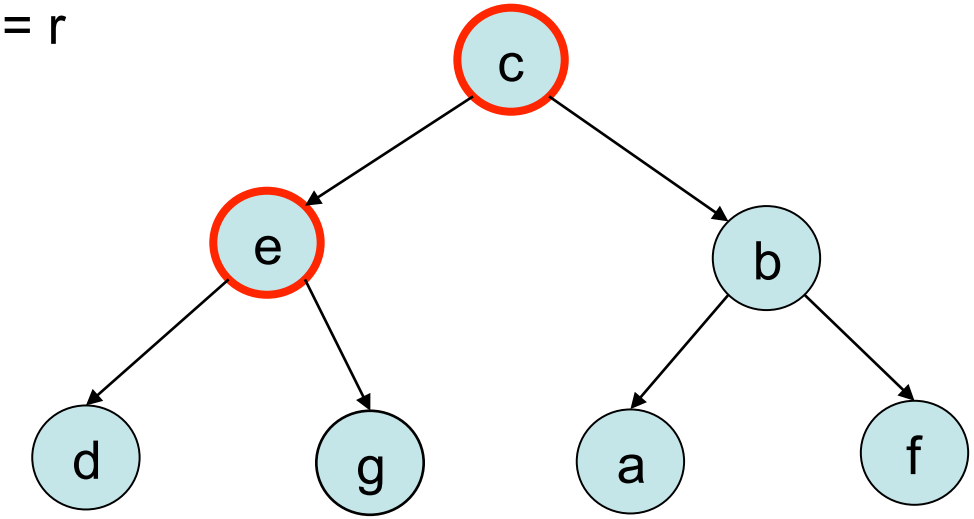


Nearest = g
dist-sqd = r

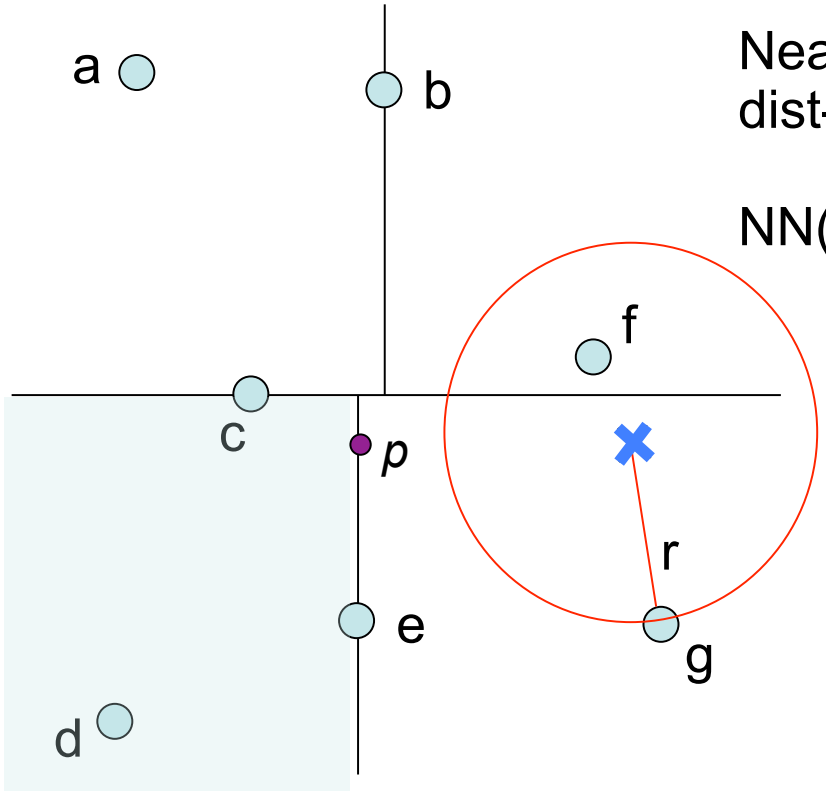


Nearest = g
dist-sqd = r

NN(e, x)

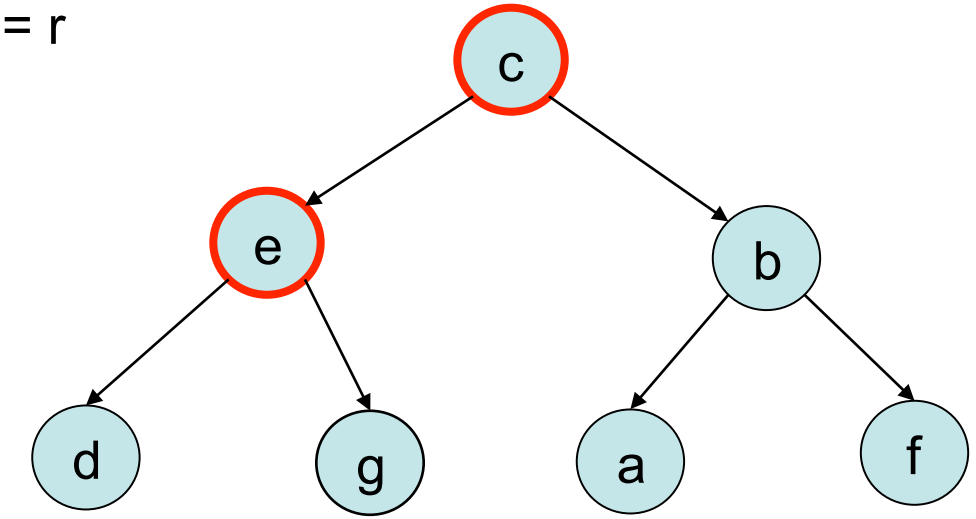


Check d2(e,x) > r
No need to update

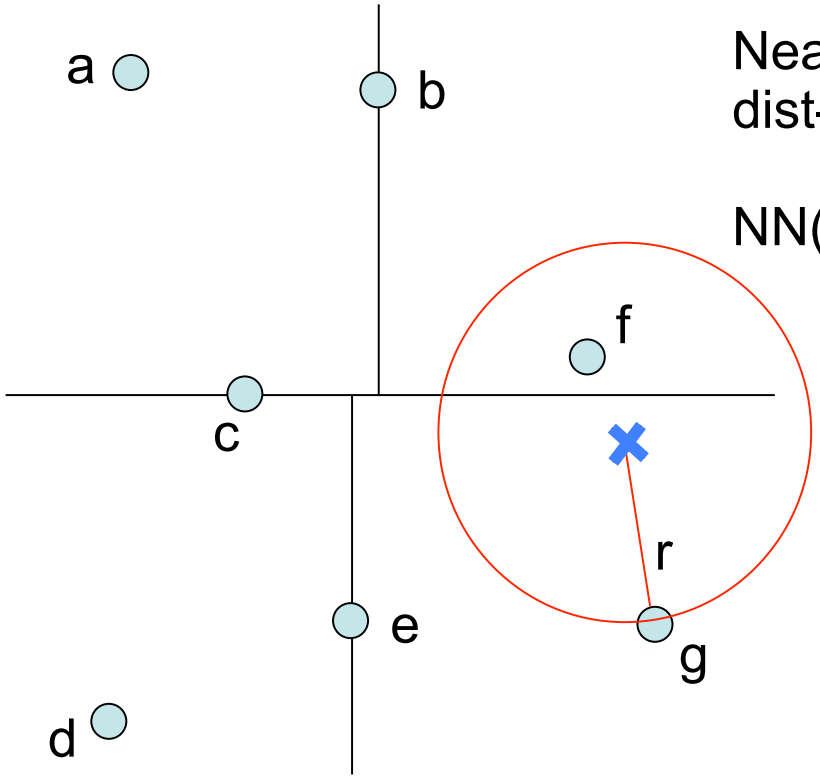


Nearest = g
dist-sqd = r

NN(e, x)

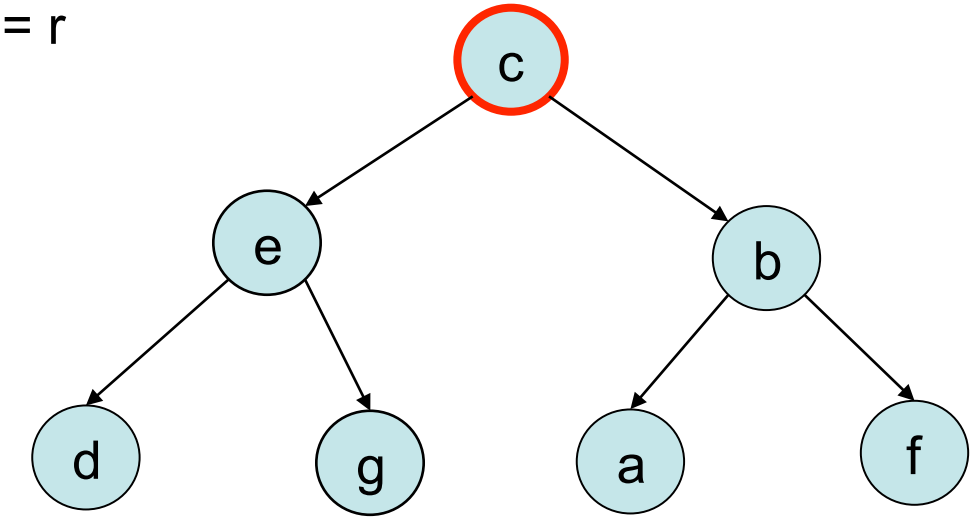


Check further of e: find *p*
d (p,x) > r
No need to update

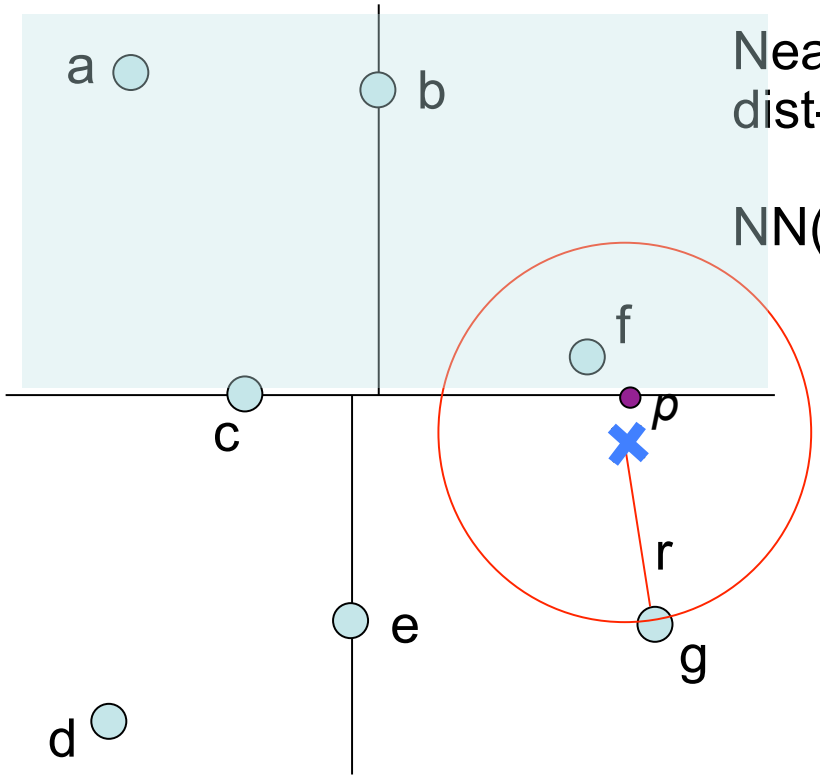


Nearest = g
dist-sqd = r

NN(c, x)

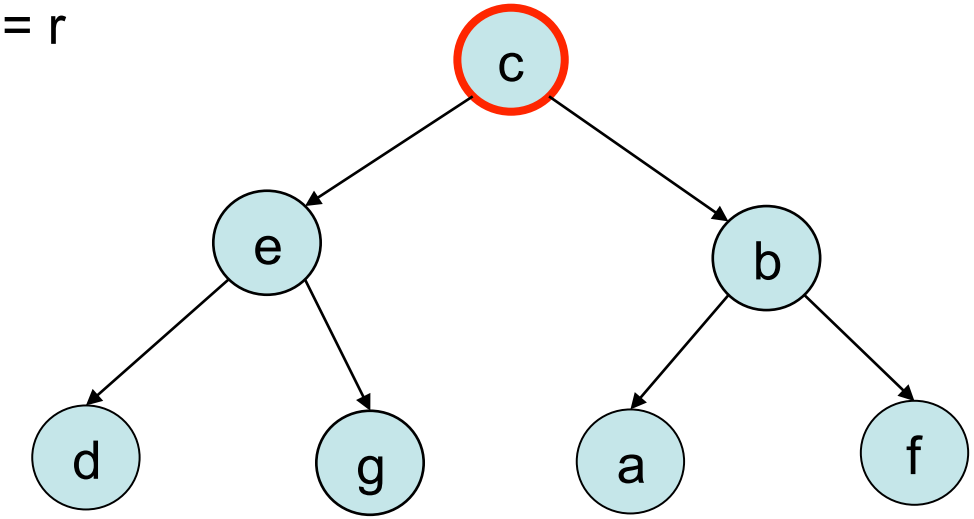


Check d2(c,x) > r
No need to update



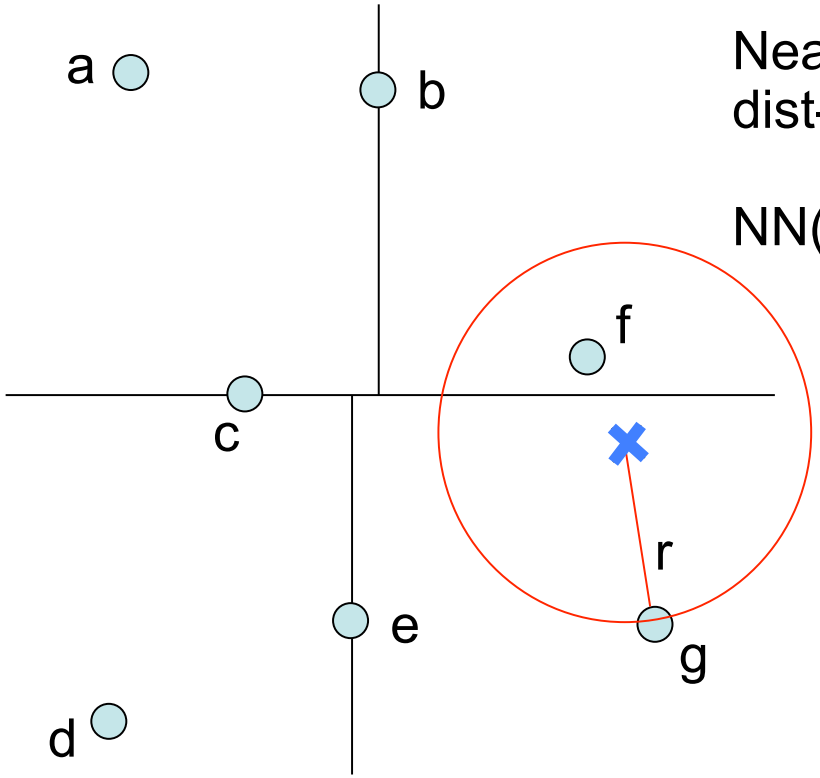
Nearest = g
dist-sqd = r

NN(c, x)



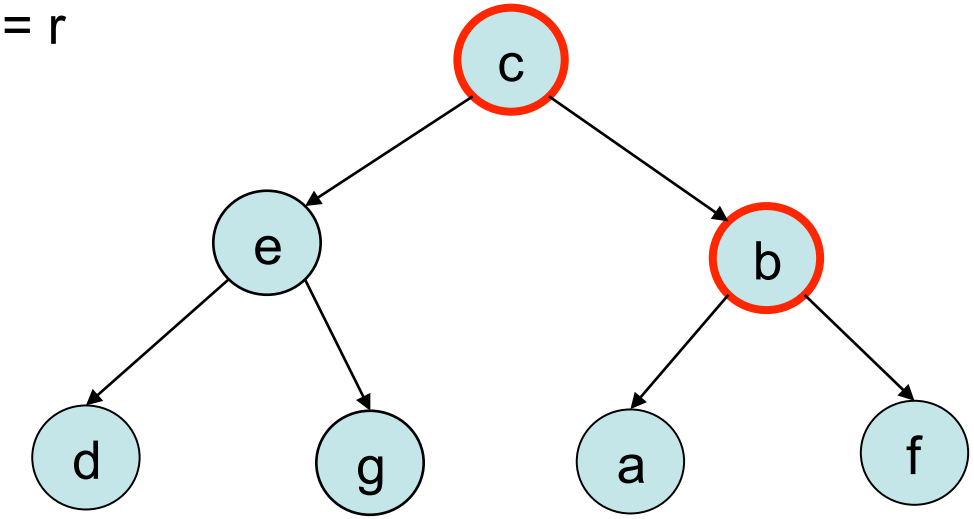
Check further of c: find p
d(p,x) < r !!

NN (b,x)



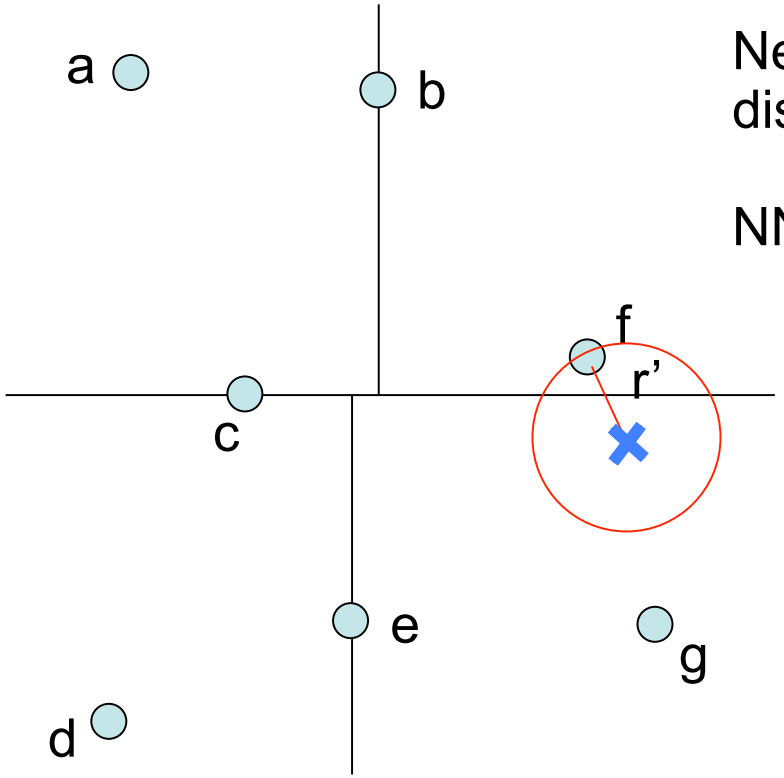
Nearest = g
dist-sqd = r

NN(b, x)



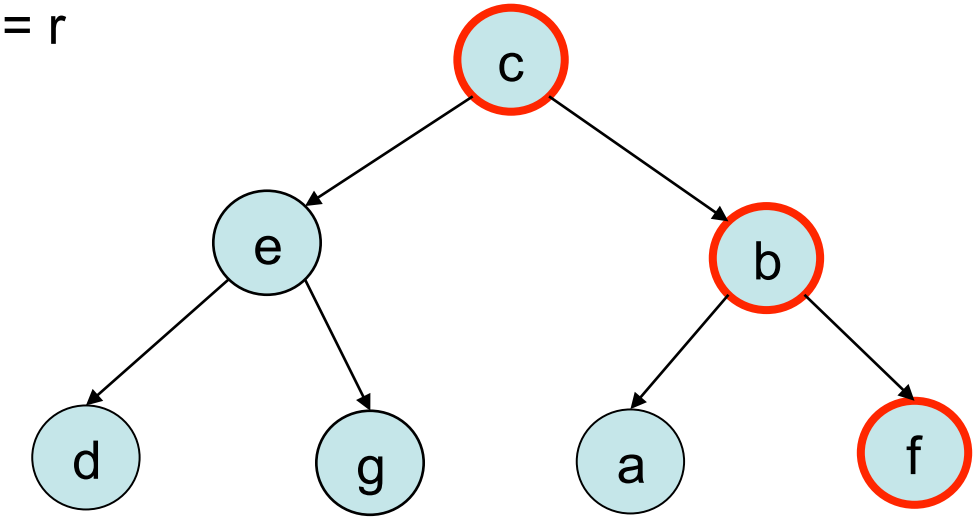
Nearer = f
Further = g

NN (f,x)



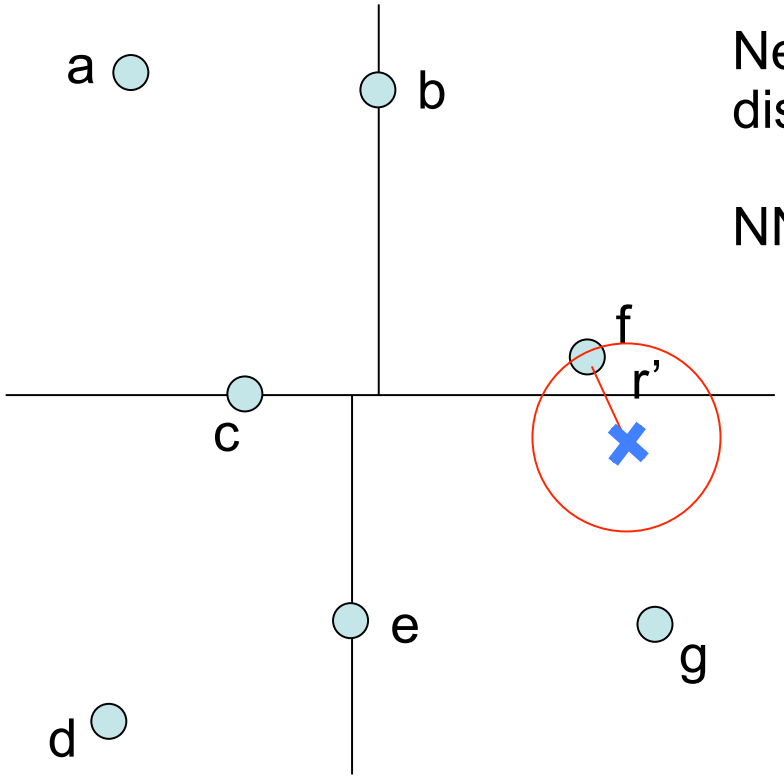
Nearest = g
dist-sqd = r

NN(f, x)



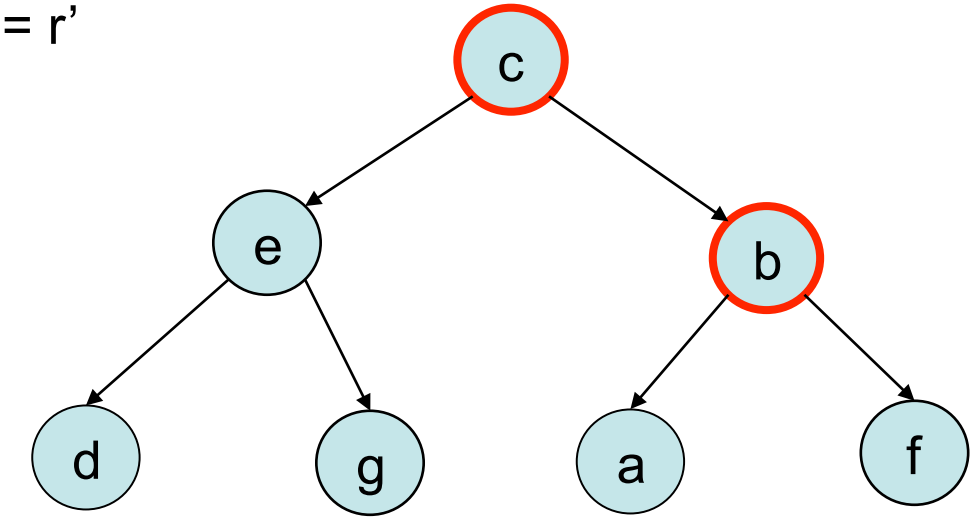
r' = d2 (f,x) < r

dist-sqd ← r'
nearest ←f



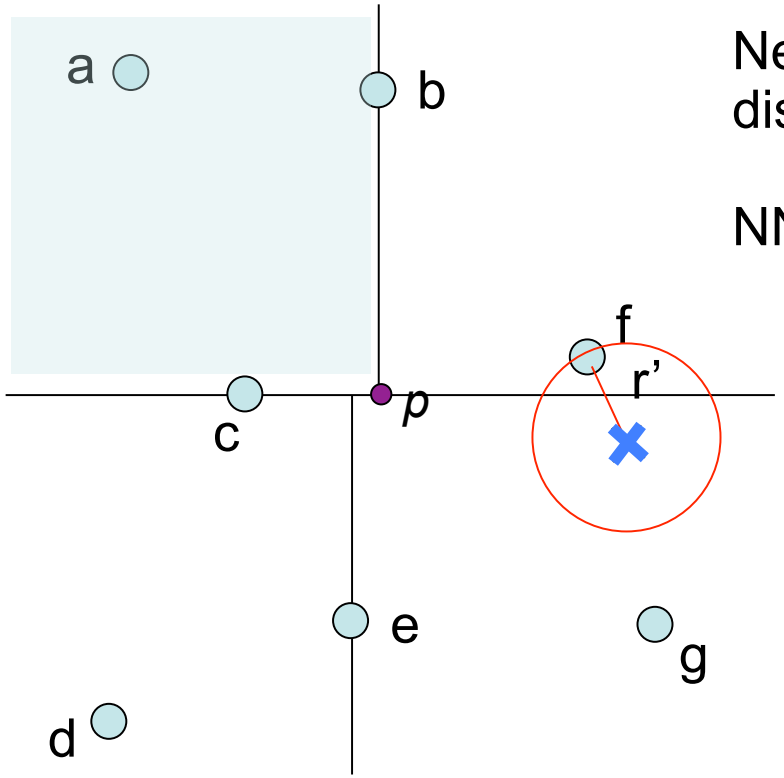
Nearest = f
dist-sqd = r'

NN(b, x)



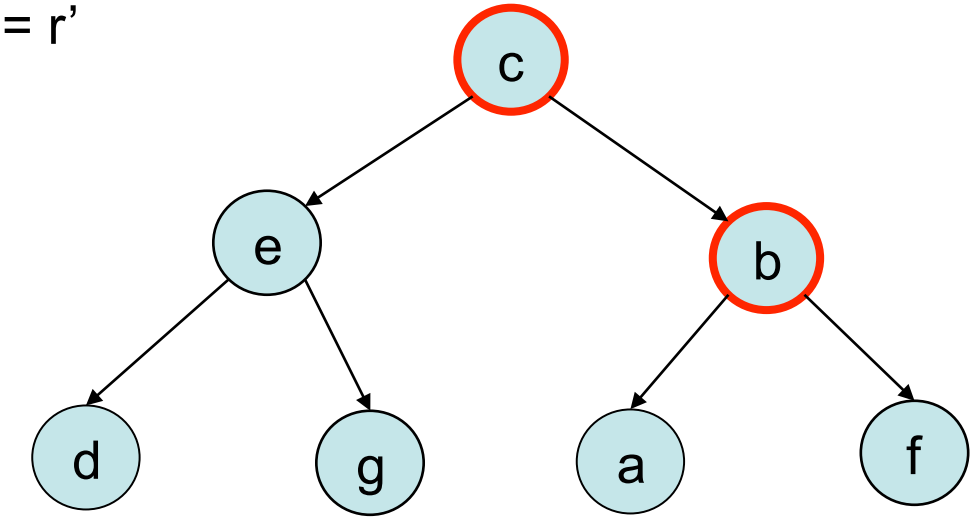
Check d(b,x) < r'

No need to update



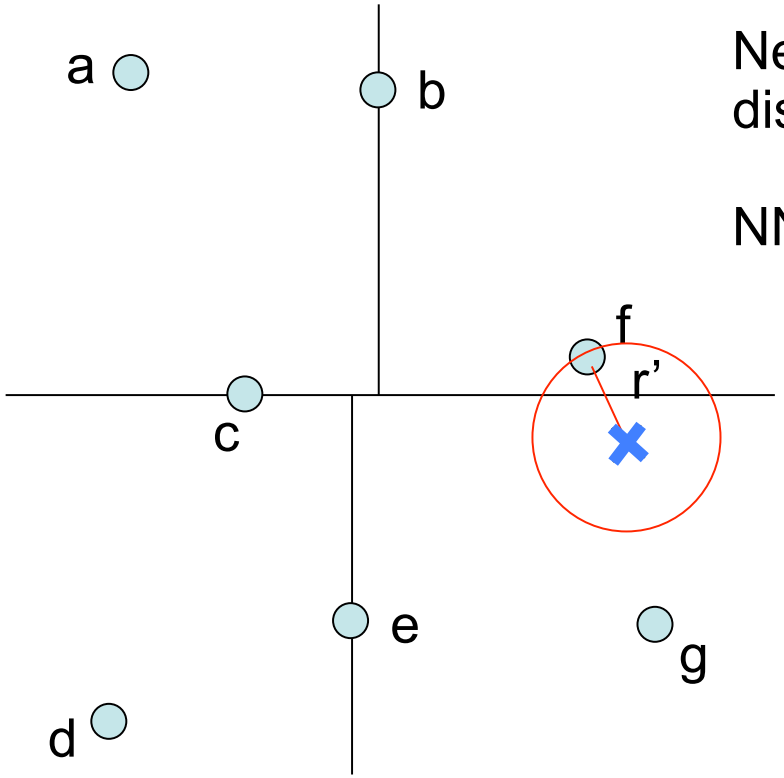
Nearest = f
dist-sqd = r'

NN(b, x)



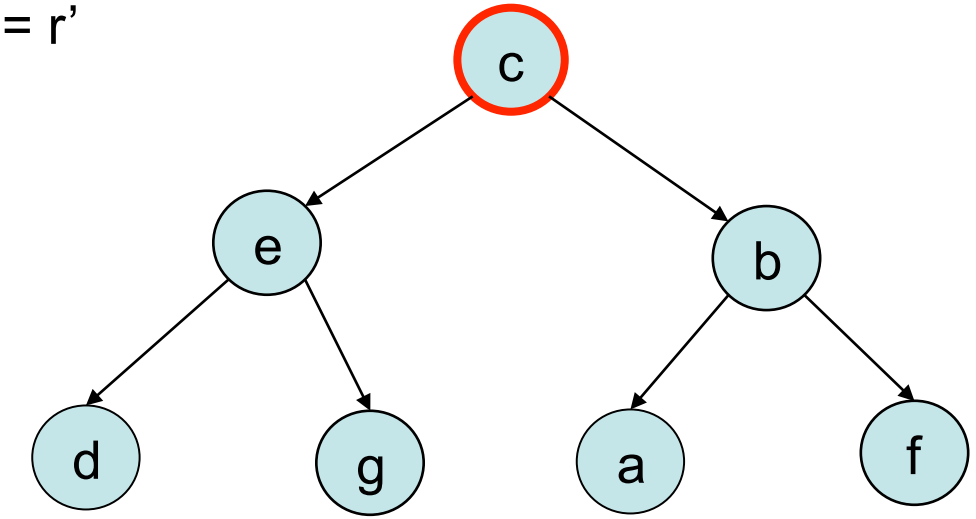
Check further of b; find p
d(p,x) > r'

No need to update



Nearest = f
dist-sqd = r'

NN(c, x)



Índice

- Algoritmo
- Estructura de datos
- **Mejoras**
- Resultados

Mejoras

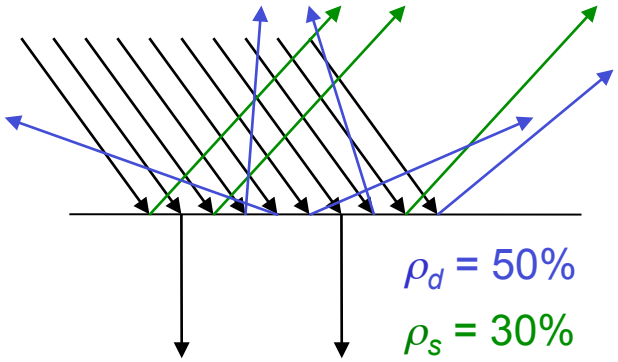
- A más fotones almacenados, más tiempo de cálculo.
  - Eliminar fotones que ”no se van a ver” (reflexión especular perfecta, cortar rebotes cuando la energía es cero).
  - Intentar que todos los fotones tengan aproximadamente la misma energía.

Mejoras

- El algoritmo funciona mejor cuando todos los fotones almacenados tienen la misma energía:
  - Muestreo por importancia (luces / BRDFs)
  - Ruleta Rusa

Mejoras

- El algoritmo funciona mejor cuando todos los fotones almacenados tienen la misma energía:
  - Muestreo por importancia (luces / BRDFs)
  - **Ruleta Rusa**
    - ζ aleatorio [0..1]
    - $\zeta < \rho_d \rightarrow$ difuso
    - $\zeta < \rho_d + \rho_s \rightarrow$ especular



Mejoras

- El camino ”directo” se calcula perfectamente con trazado de rayos de sombra estándar.
  - Primer rebote de fotones no se guarda.
  - Camino directo mediante trazado de rayos.
  - Algunos trabajos hacen incluso un rebote indirecto mediante rayos.

Mejoras

- Diferentes fenómenos se ven mejor con diferentes parámetros del mapa
  - Utilizar más de un mapa de fotones
  - Ejemplo: uno para luz indirecta y otro para cáusticas.
  - Ejemplo retorcido: un mapa de fotones por cada objeto.

Mejoras

- El cuello de botella del algoritmo es la búsqueda de elementos más cercanos.
- Puedes ahorrar un rebote haciendo que cada fotón sea se comporte como una luz puntual: *Instant Radiosity*
- ¿Cómo hacerlo eficiente? → Lightcuts

Índice

- Algoritmo
- Estructura de datos
- Mejoras
- **Resultados**