Estimación de Densidades usando GPUs

M. Lastra[1], C. Ureña[1], J. Bittner[2], R. García[1], R. Montes[1]
[1] Universidad de Granada
[2] Czech Technical University in Prague

25 de julio de 2008

Objetivo final

Obtener imágenes realistas de escenas que se desean renderizar

Necesitamos

Calcular (estimar) la irradiancia en determinados puntos de la escena para poder calcular (estimar) la energía reflejada hacia el observador.

Monte Carlo. Fotosimulación

Una opción muy utilizada es la simulación del flujo de la energía luminosa en la escena de forma estocástica mediante la generación de *fotones* desde las fuentes de luz. Estas partículas recorrerán una serie de trayectorias en linea recta.

Estimación de densidades mediante mapas de fotones

Estimación de la irradiancia en un punto de la escena utilizando la energía de los impactos de fotones *cercanos* .

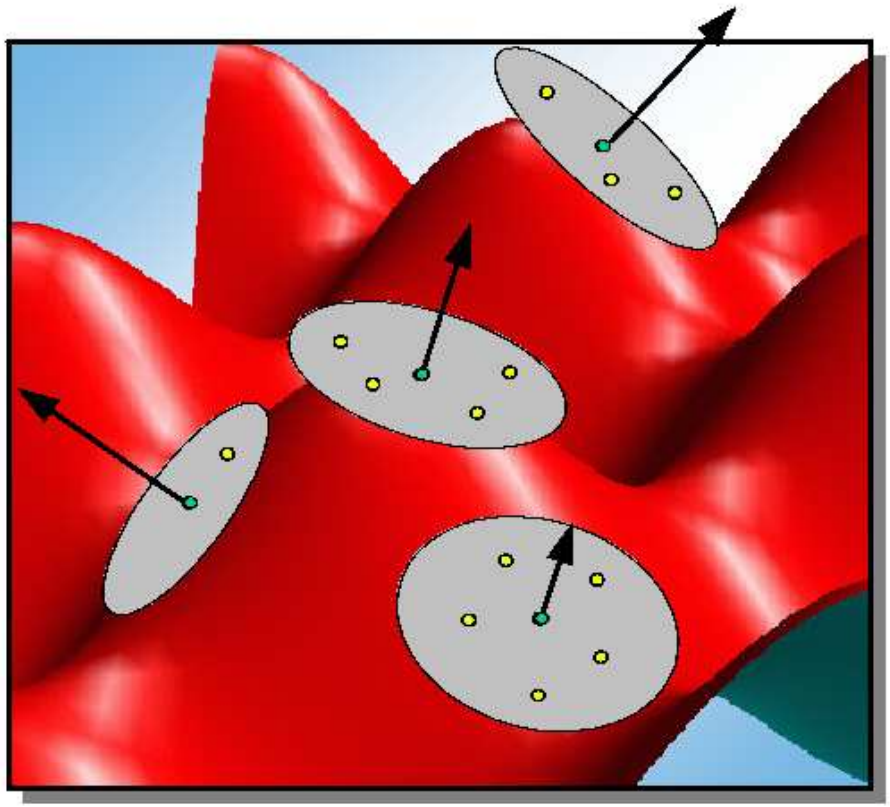
Estimación de densidades en el plano tangente (DETP)

Se utilizan las trayectorias de los fotones y un disco centrado en cada punto donde se desea estimar la irradiancia. La suma de la energía de los rayos que intersecan cada disco se usa para obtener un estimador de la irradiancia.

Estimación de la irradiancia

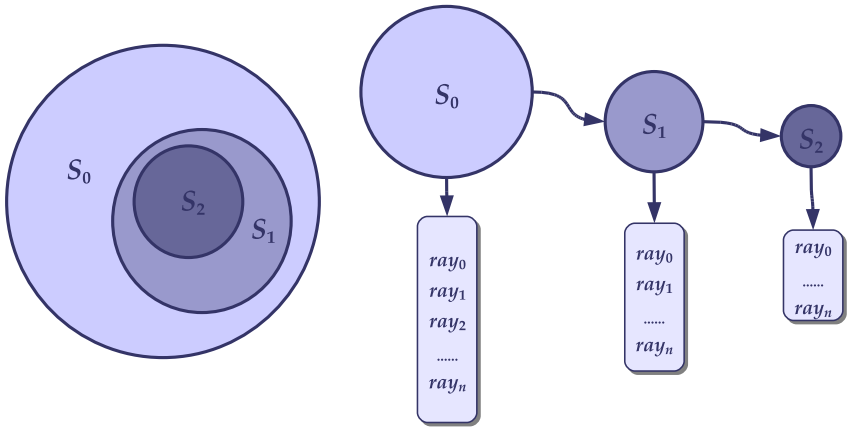
Utilizamos la energía de los rayos que intersecan el disco definido en el plano tangente y centrado en el punto considerado

$$E(x) \approx \frac{1}{\pi r^2} \sum_{i \in I_r(x)} \phi_i$$



Cache de esferas

- Se utiliza para reducir el número de intersecciones rayo-disco
- Sólo se calculan las intersecciones con los rayos que intersecan una lista de esferas que contienen el disco.
- Las esferas se reutilizan para un gran número de discos
- Cuando se produce un fallo de cache (disco a procesar no está contenido en la ultima esfera) se reconstruye la parte de la cache que deja de ser *válida*



Motivación

El núcleo del algoritmo DETP es el cómputo de una serie de intersecciones rayo-disco. Este cómputo puede ser implementado usando GPUs ya que incluye una gran cantidad de operaciones de coma flotante que se pueden llevar a cabo en paralelo.

Modificaciones

Portar el algoritmo de una arquitectura de propósito general a una plataforma especializada requiere varias adaptaciones. Estas se refieren a la organización y almacenamiento de los datos y de la realización de los cálculos

Rayos

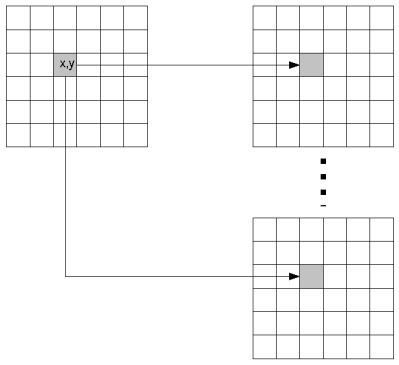
Los rayos se almacenan en tres texturas 2D: origen, vector director y energía

Discos

Los discos se almacenan en dos texturas: una para el centro y otra donde se empaqueta el vector normal y el radio

Índices 2D

Como es necesario referenciar los mismos discos y rayos multiples veces, una vez cargados, se usan índices 2D para referenciarlos



Cache de esferas

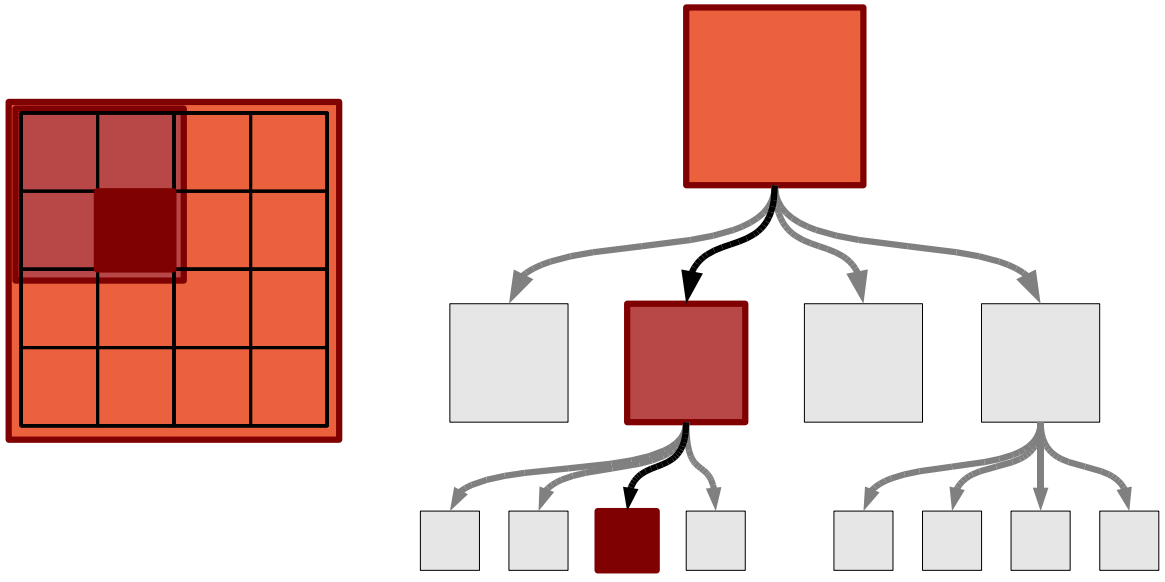
La cache de esferas no es adecuada para ser implementada en GPUs por su naturaleza dinámica

Octree

Optamos por el uso de una estructura estática tipo octree. El octree se construye bajo demanda y en cada momento solo se almacena la secuencia de nodos desde la raiz hasta el nodo hoja procesado en ese instante.

Procesamiento primero en profundidad

En cada nodo hoja se calculan las intersecciones de los discos y rayos contenidos en ese nodo. Se almacenan texturas con referencias a las texturas de los discos y rayos. El procesamiento primero en anchura sería mas eficiente pero requiere almacenar demasiadas texturas al mismo tiempo.



Octree en un momento del cómputo

Solo los nodos marcados existen cuando se procesa el nodo hoja señalado.

Cálculo de intersecciones

Todas los tests de intersección se calculan en la GPU: tanto las rayo-caja, disco-caja como las disco-rayo

Fragmentos

Todos las intersecciones se calculan en kernels de cálculo asociados a fragmentos. Se dibuja un rectangulo con tantos pixels como frgmentos sean necesarios.

Resultados

Los resultados de los test de intersección se obtienen por tanto en una textura de salida.

Intersecciones rayo-caja y disco-caja

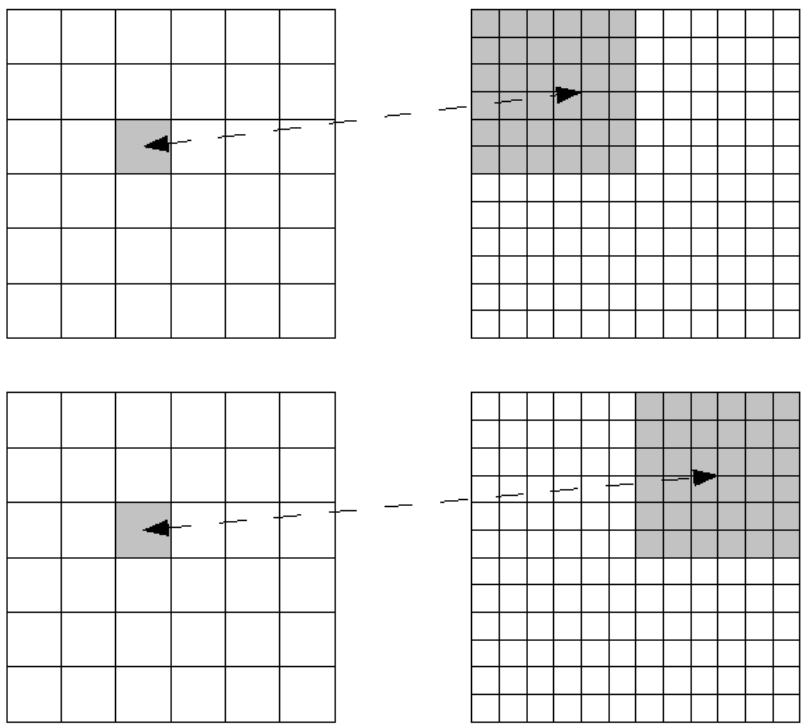
Se crea un fragmento por cada rayo o disco y el kernel de computación calcula en paralelo las intersecciones de todos los rayos o discos con la misma caja englobante.

Intersecciones disco-rayo

Se crea un fragmento por cada disco y en el kernel asociado se calcula la intersección de cada disco con todos los rayos (bucle)

Máximo número de instrucciones

Una vez que se llega a este límite el cómputo se detiene y el proceso termina. Para evitar esta limitación, no todos los rayos se procesan de una vez y se ejecutan varios pasos de rendering.

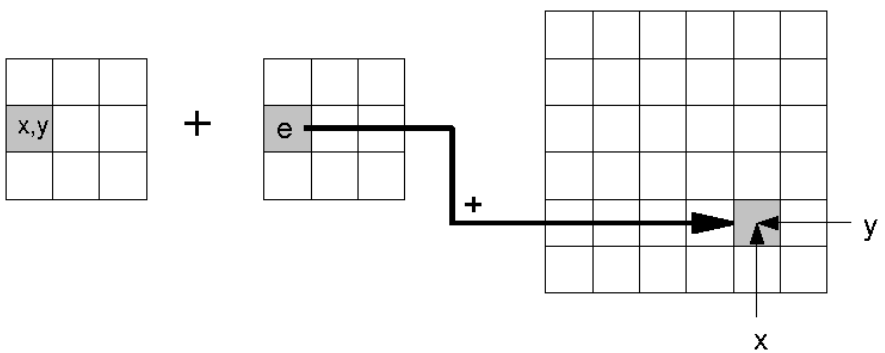


Distribución de la energía

En cada nodo hoja se obtiene, para cada disco, la cantidad de energía aportada por los rayos de ese nodo. Estos valores de energía es necesario distribuirlos a la posición correspondiente de la textura que almacena el resultado final.

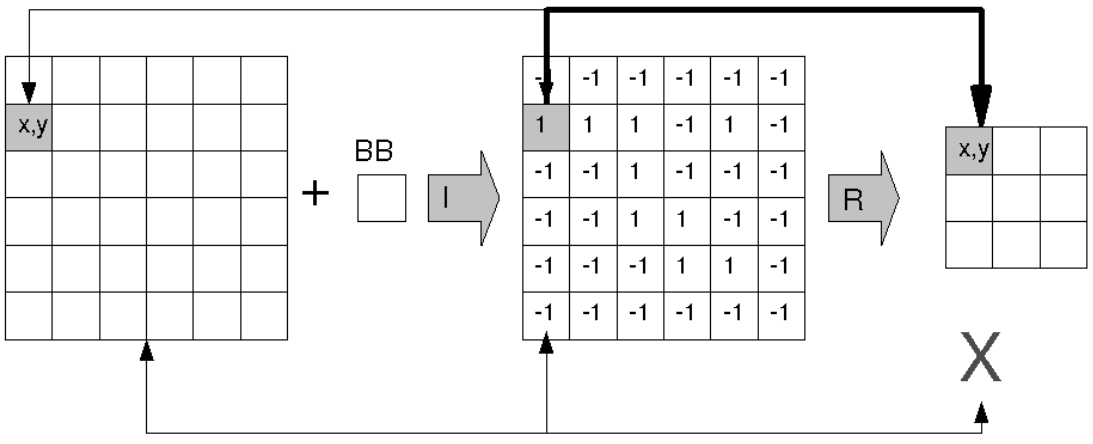
Scattering

Utilizamos un procesador de vértices para esta operación. Usando los indices de cada disco del nodo hoja se distribuye cada valor de energía a la posición adecuada de la textura resultado.



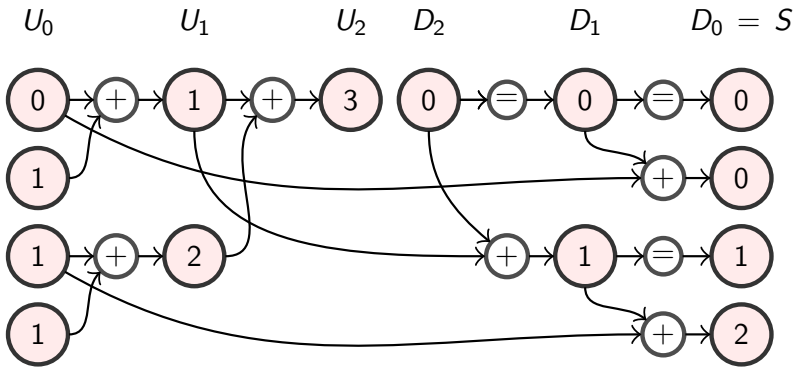
Reducción: necesaria al procesar nodos del octree

Al calcular los rayos o discos contenidos en la caja englobante de un nodo, es necesario *eliminar* los que no cumplen esta condición. Esto implica aplicar una reducción no uniforme del flujo representado por la textura.



Tres pasadas

- Calcular el número de entradas válidas. Esto se hace progresivamente aumentando sucesivamente el tamaño de las regiones consideradas
- Calcular el vector que indica a que posición de la textura de salida debe copiarse cada entrada válida
- Usando el vector anterior, realizar la recolección de los datos válidos desde la textura resultado



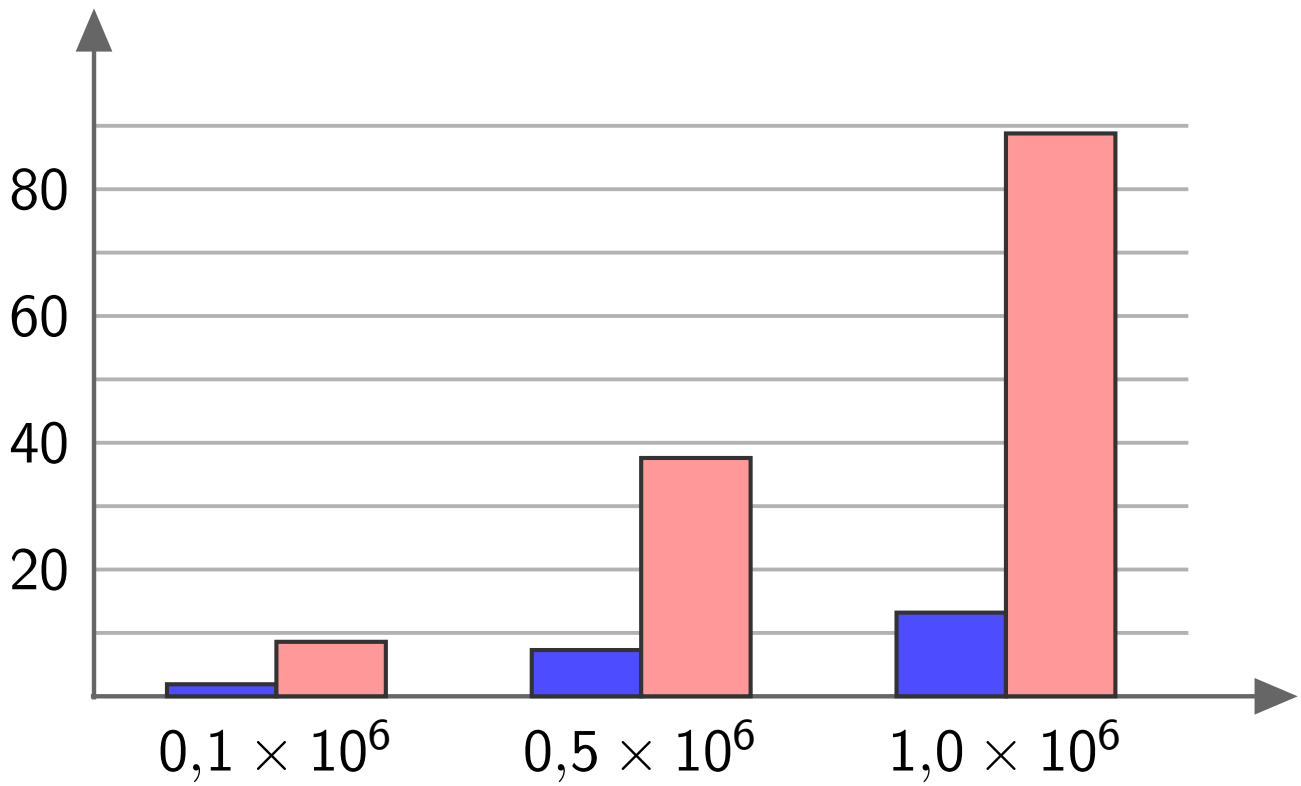
Entorno de prueba

- AMD Athlon 64 5200+ CPU (2.6Ghz)
- Nvidia 8800GTX
- Escena con distribución aleatoria de rayos y discos

#discos × #rayos	GPU	CPU(SSE)	CPU
$10^4 \times 10^4$	0.82s	0.18s	0.22s
$10^5 \times 10^5$	1.9s	8.6s	13s
$10^5 \times (5 \times 10^5)$	7.3s	37.6s	60.1s
$10^5 \times 10^6$	13.2s	88.8s	124.7s

Resultados

- Se consigue reducir los tiempos de cálculo en un orden de magnitud
- Se ha tenido en cuenta también el uso de instrucciones SIMD en la CPU (SSE)



Distribución del tiempo de cálculo en la GPU

Tiempo total (10^5 discos $\times$ 10^6 rayos)	13.2s
Intersecciones Rayo-Nodo octree	53ms
Intersecciones Disco-Nodo octree	17ms
Intersecciones Rayo-Disco	**11.8s**
Filtrado	380ms
Distribución de energía	32ms

Trabajo futuro

- Estudiar otras estrategias que permitan eliminar el uso del índice espacial ya que este requiere demasiada interacción con la GPU
- Utilizar directamente los valores de energía obtenidos en la GPU para generar una imágen directamente (presentación de resultados)