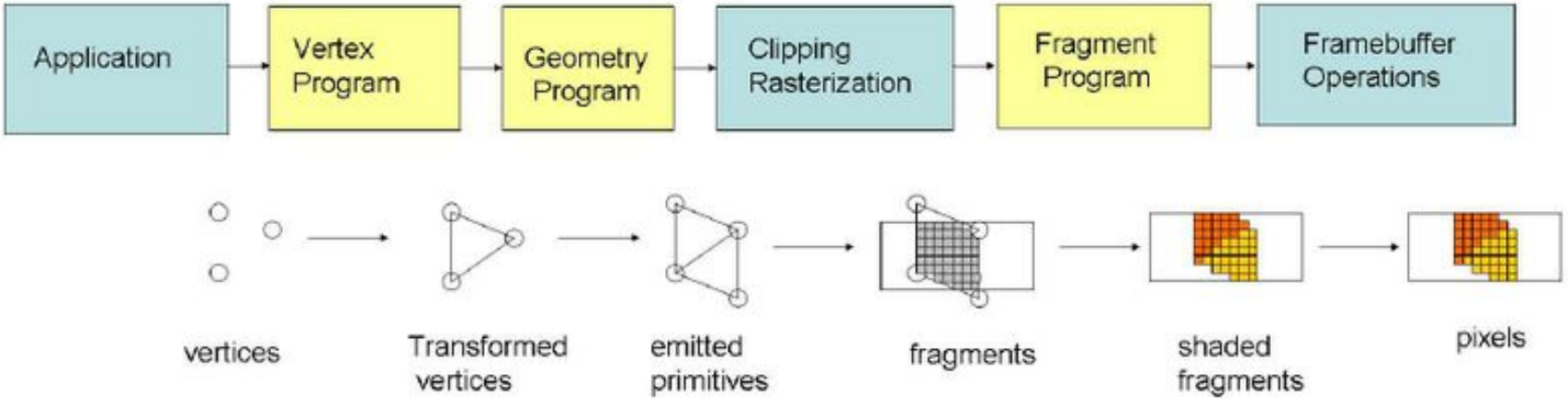
Framework para abstracción de entornos heterogéneos CPU/GPU

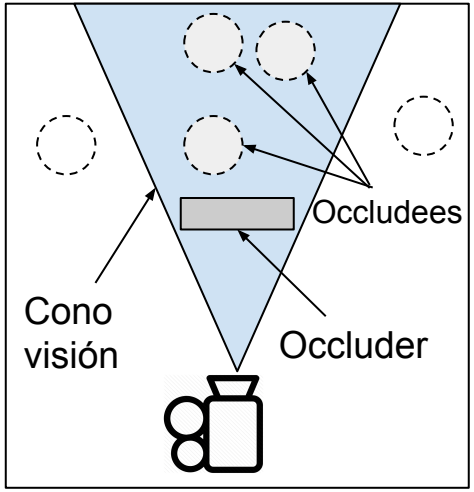
Optimización de renderización por cálculo de occlusion culling en paralelo

Introducción

Introducción

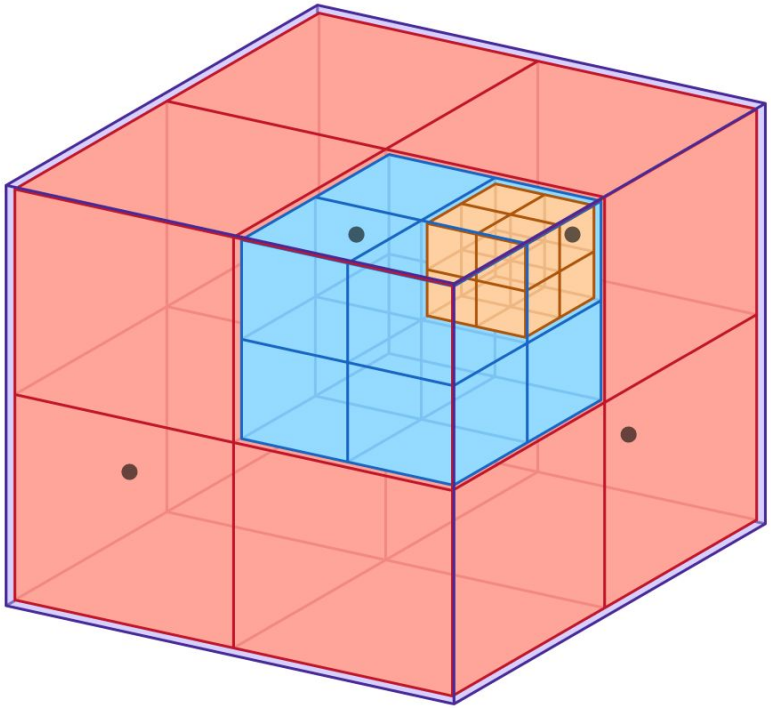


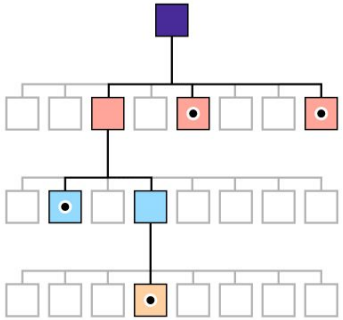
Introducción



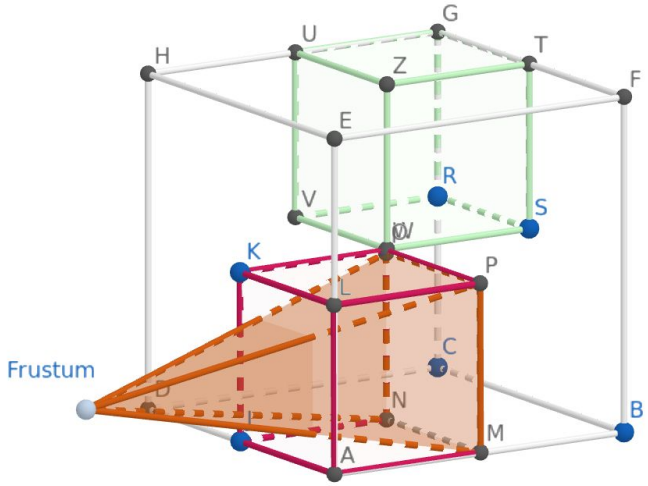
Diseño

Diseño de estructura de datos

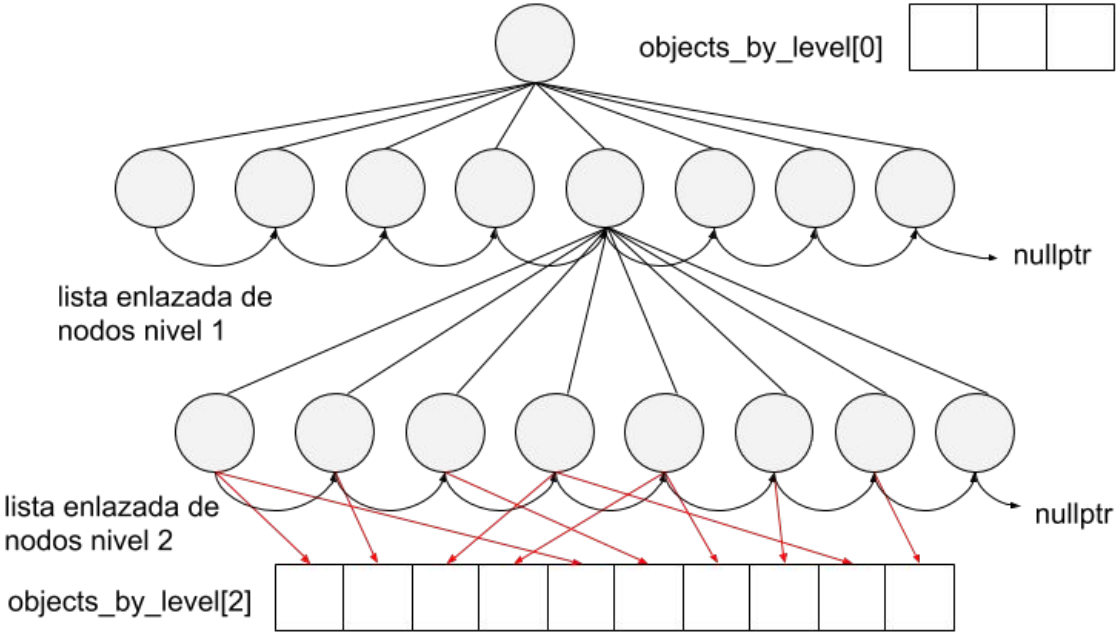




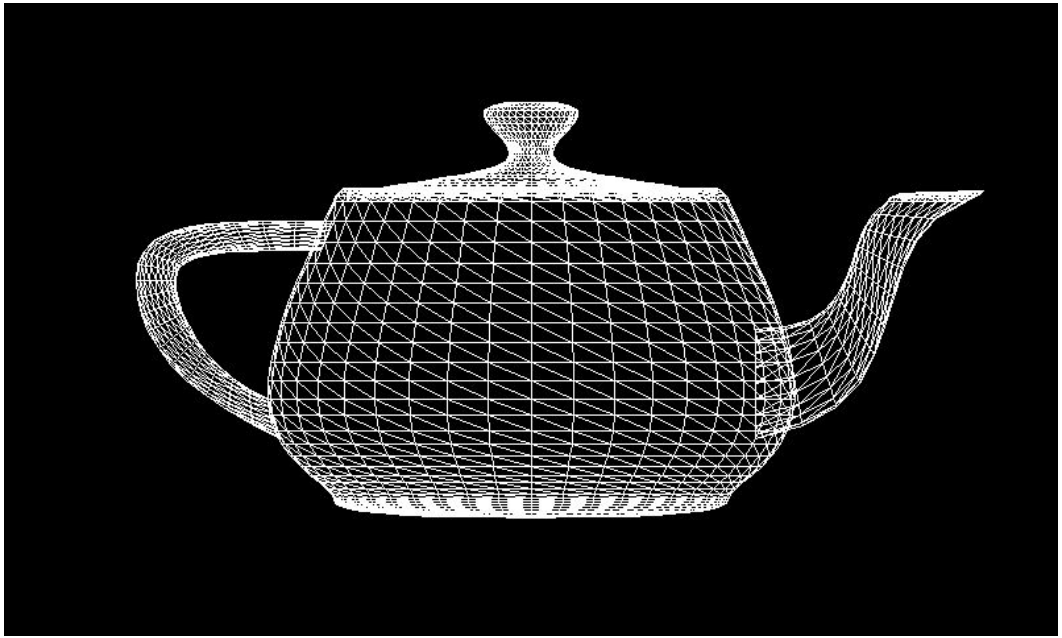
Diseño de estructura de datos



Diseño

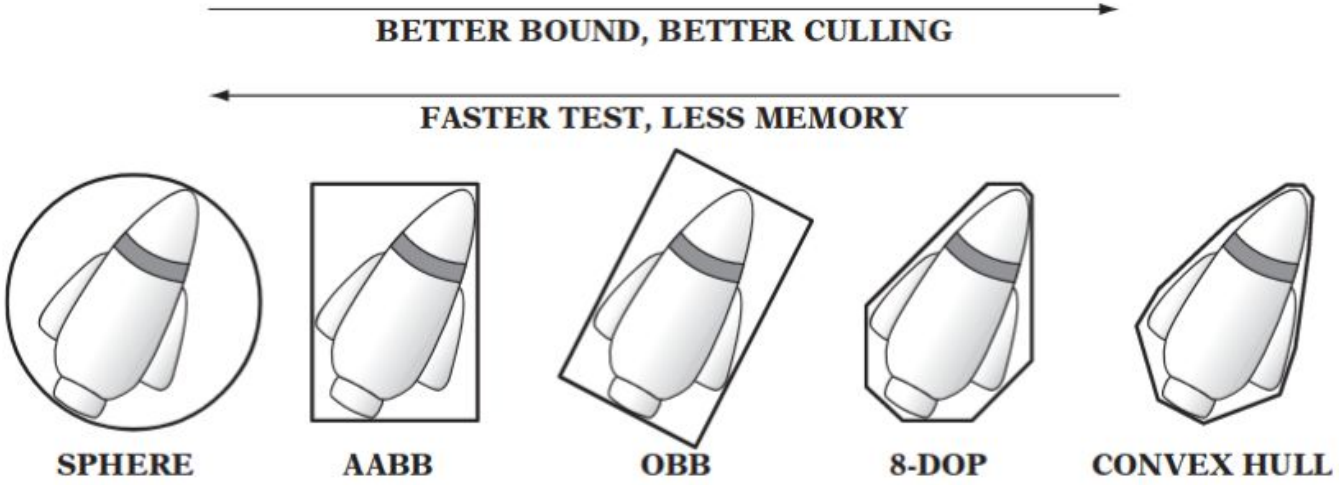

objects_by_level[0]
nullptr
lista enlazada de
nodos nivel 1
lista enlazada de
nodos nivel 2
nullptr
objects_by_level[2]

Simplificación de modelos

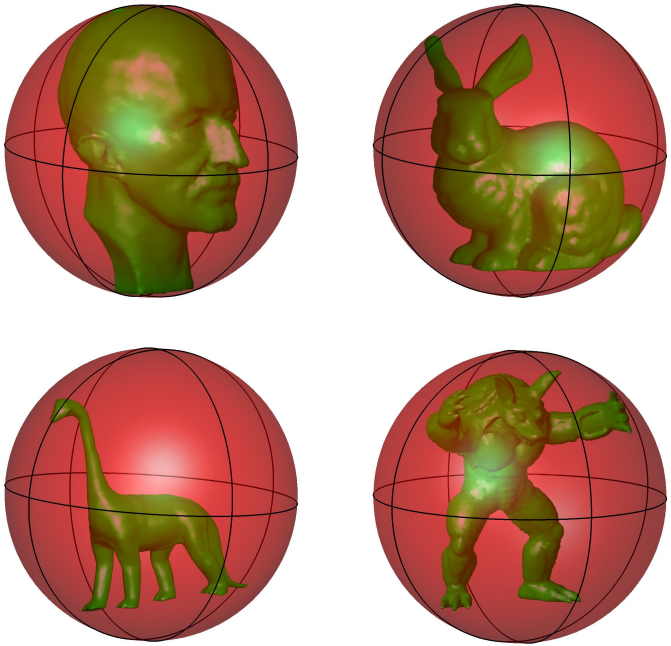


- Objetos son complejos
- Muchos triángulos por objeto
- Falta de consistencia/coherencia entre distintos objetos

Simplificación de modelos (Primitiva englobante)

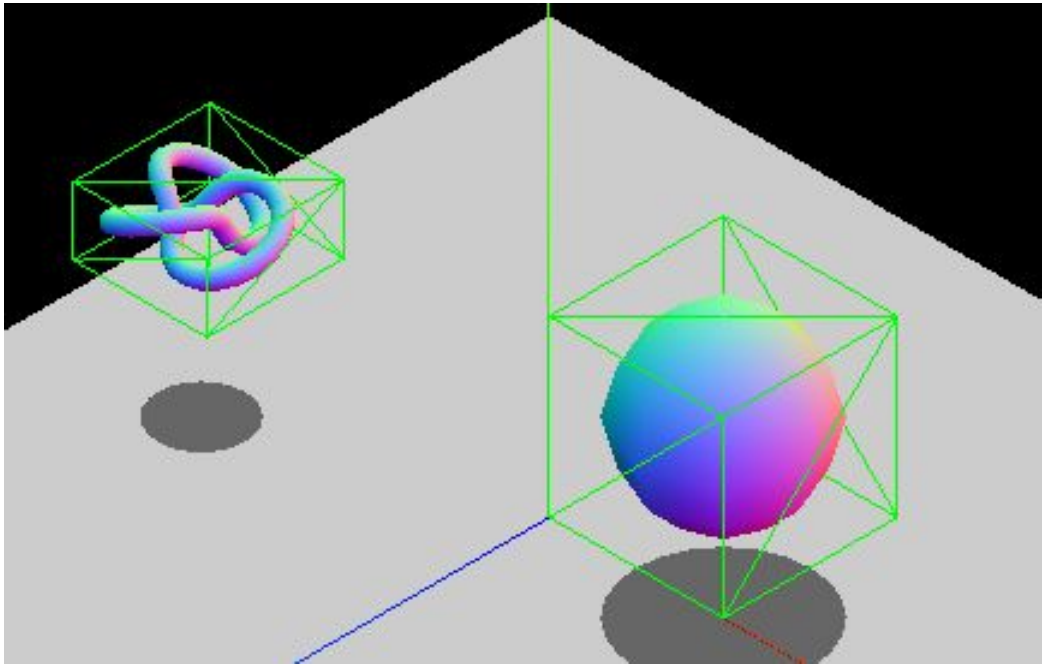


Simplificación de modelos (Bounding Sphere)



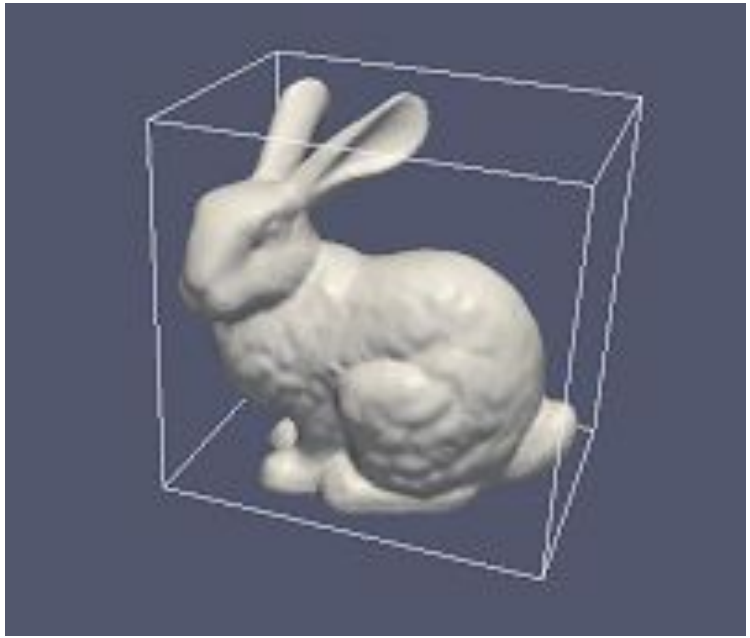
- Cómputo rápido
- Eficiente contra rotaciones
- Poca precisión

Simplificación de modelos (AABB)



- Simplificación de un problema 3D en 2D
- Se debe de recomputar cuando se rota la cámara!!

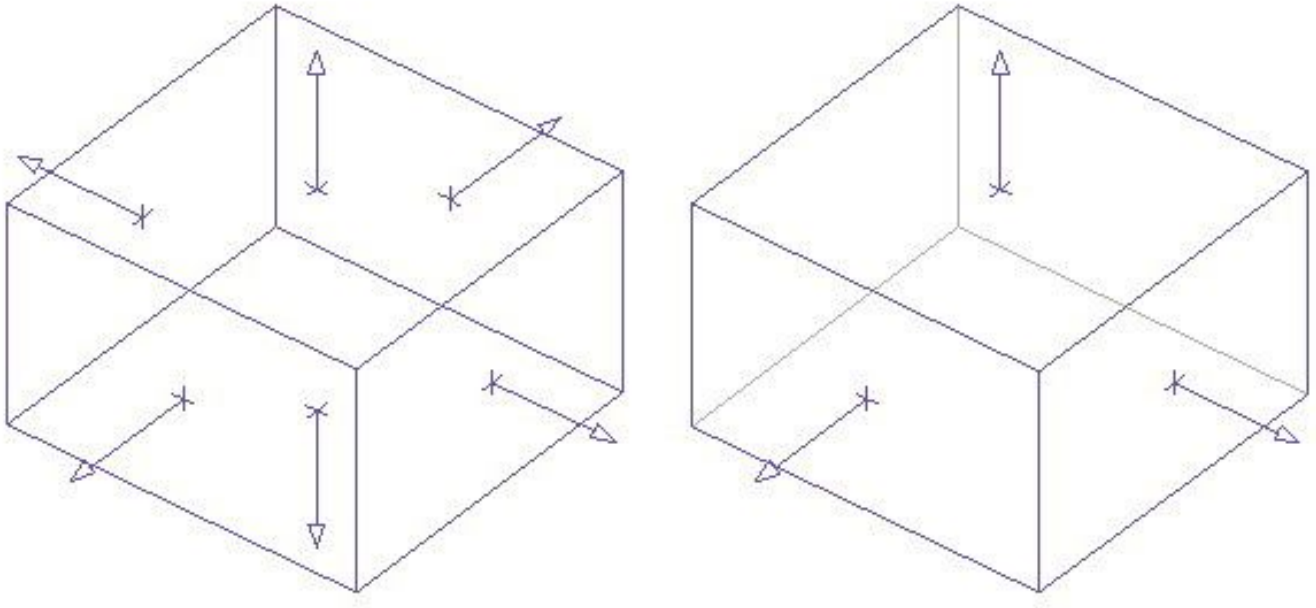
Simplificación de modelos (OBB)



- Eficiente con rotaciones
- Más precisión que la esfera
- Cómputo más complejo

Algoritmo de cálculo de oclusiones

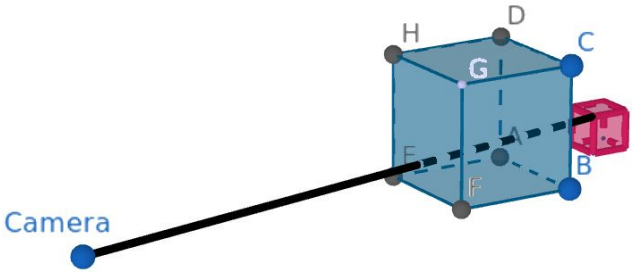
Cálculo de caras visibles (6 hebras por objeto)

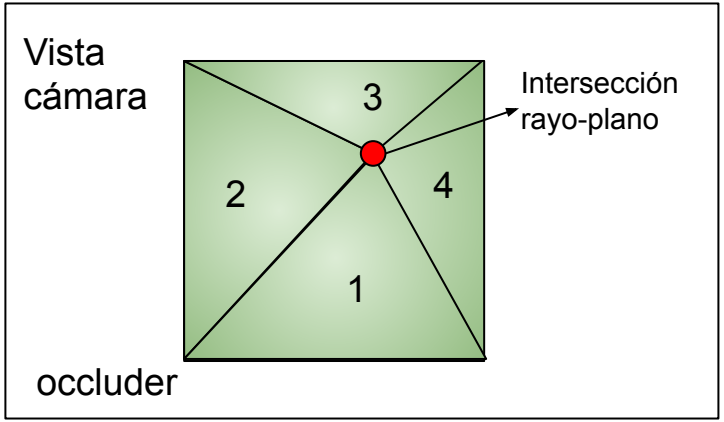


Cálculo de caras visibles

- $(V_0 - P) \cdot N \geq 0$
  - V_0 es un punto del plano N
  - P es la posición de la cámara
  - N es la normal del plano

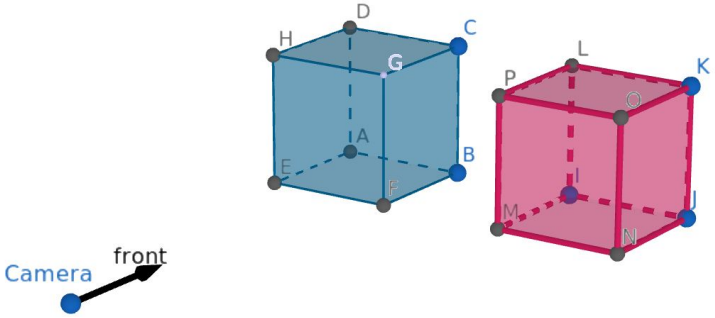
Cálculo de oclusiones

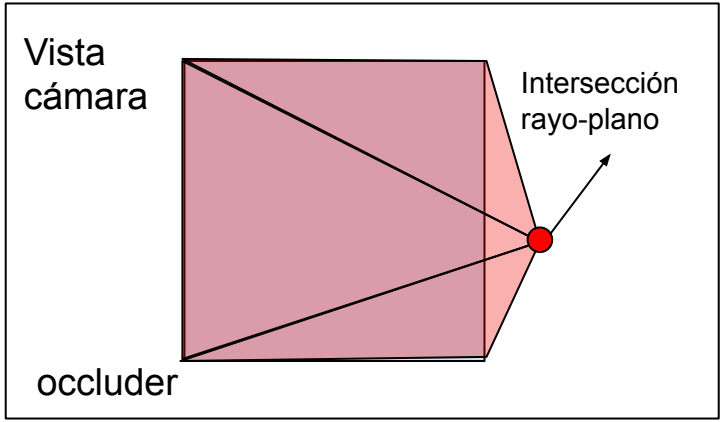




Punto Ocluido

Cálculo de oclusiones





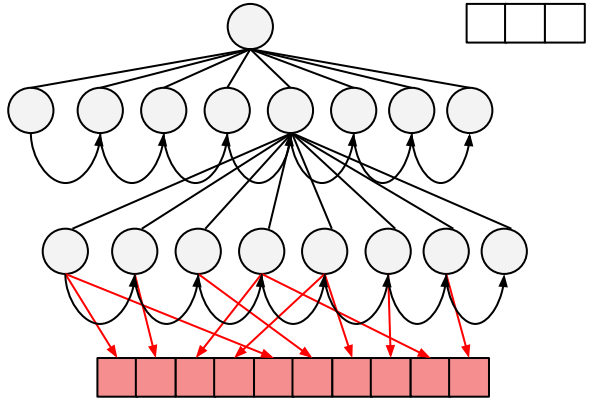
Punto NO ocluido

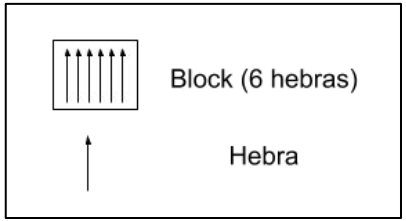
Cálculo de oclusiones

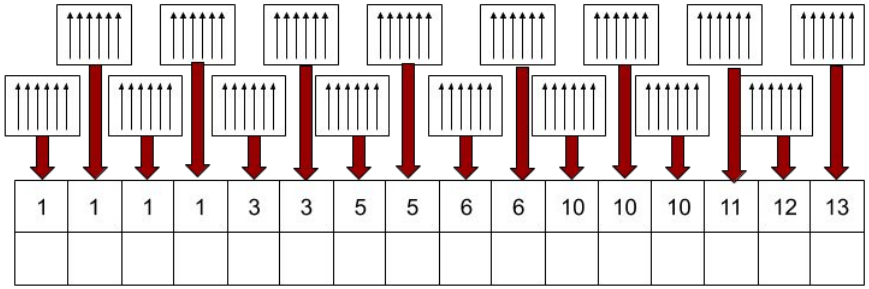
- $área = \frac{1}{2}|\overline{AB} \times \overline{AC}|$
  - A, B y C son los tres puntos que definen al triángulo
  - $\overline{AB}\ y\ \overline{AC}$ son los vectores que indican que empiezan en A y terminan en B y C respectivamente

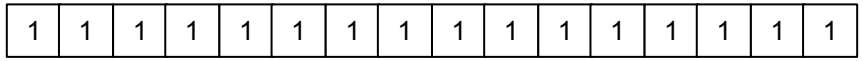
Implementación

Implementación en GPU



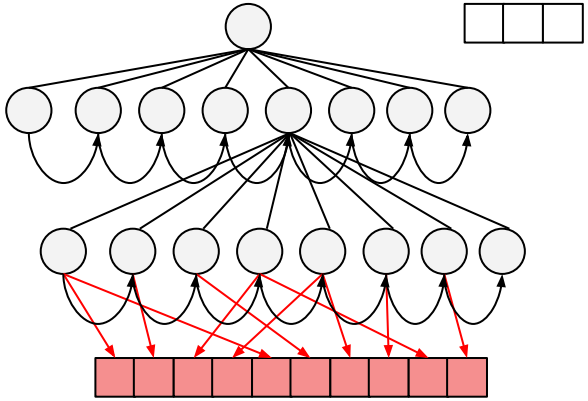


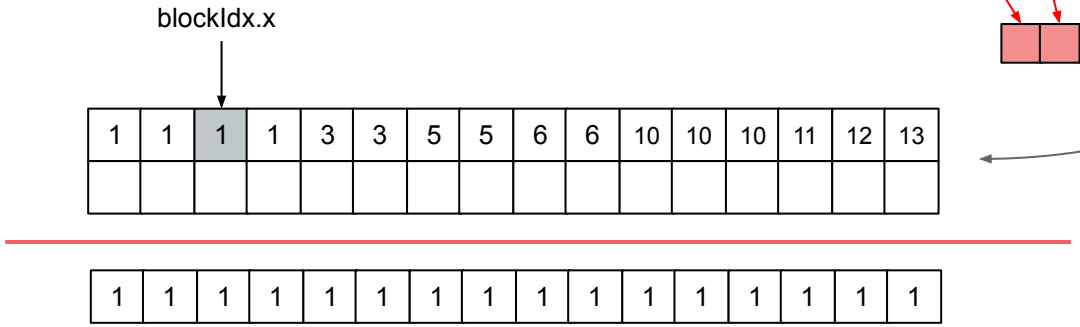




Array de visibilidad de los objetos

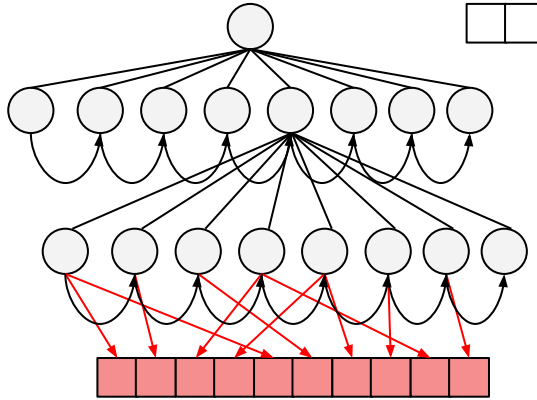
Implementación en GPU

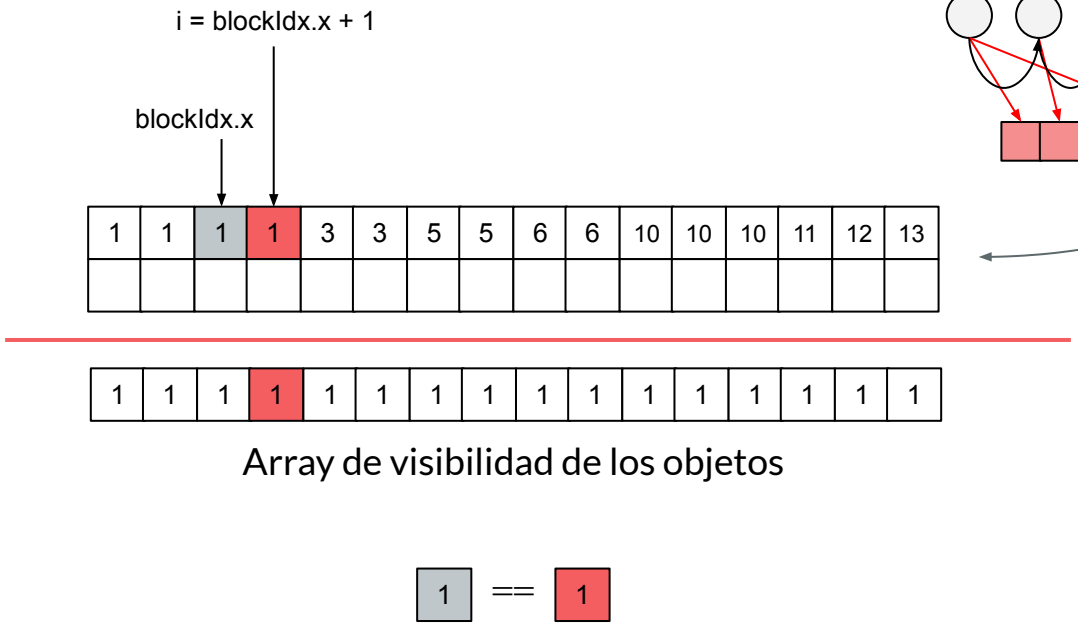




Array de visibilidad de los objetos

Implementación en GPU

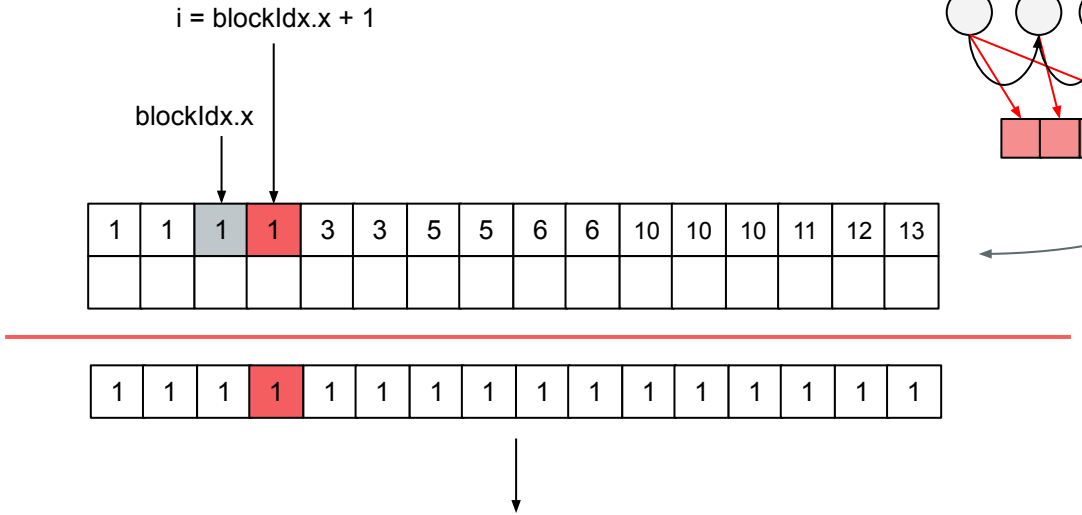


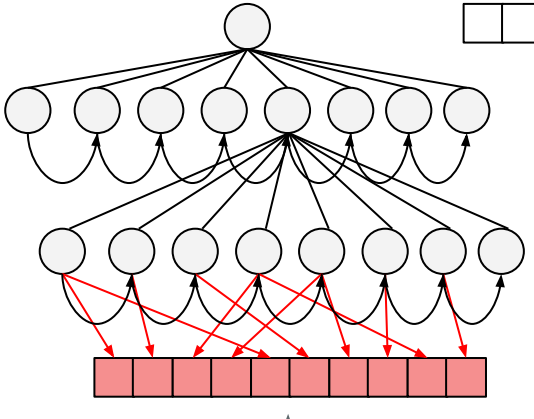


Array de visibilidad de los objetos

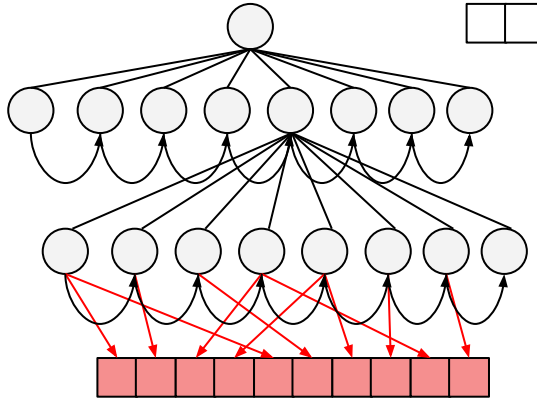
1 == 1

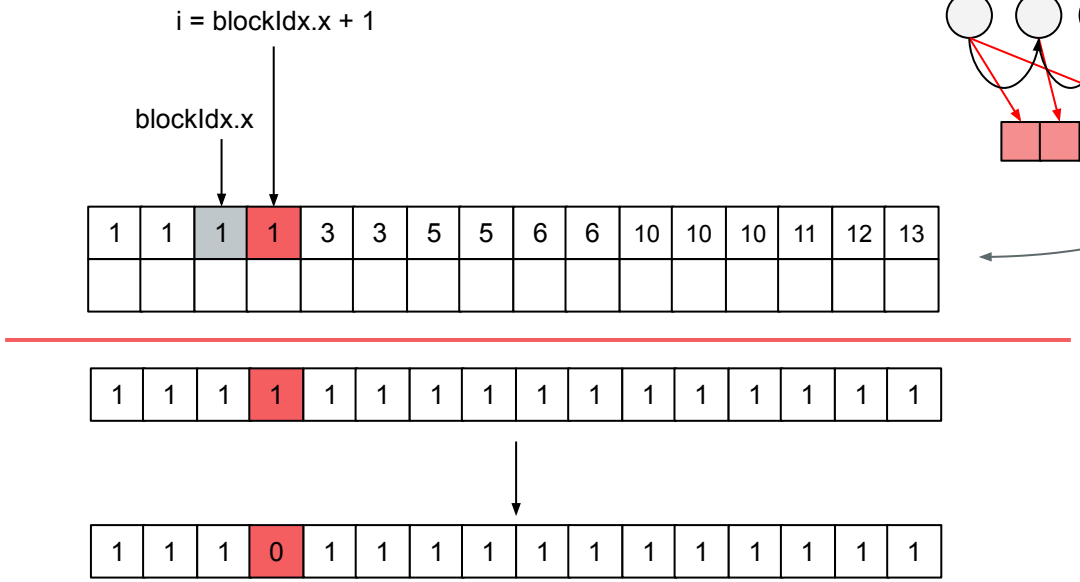
Implementación en GPU





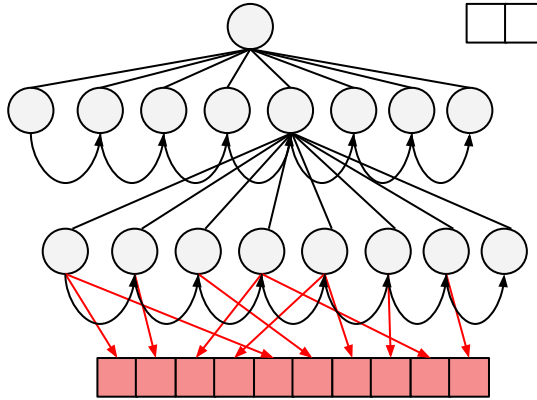
Implementación en GPU

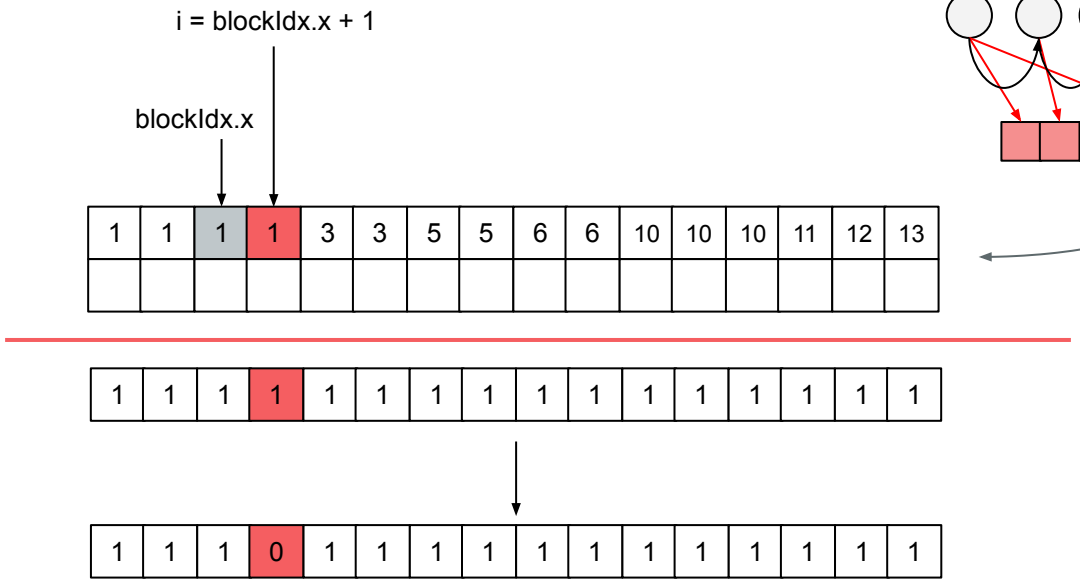




El objeto i no es visible

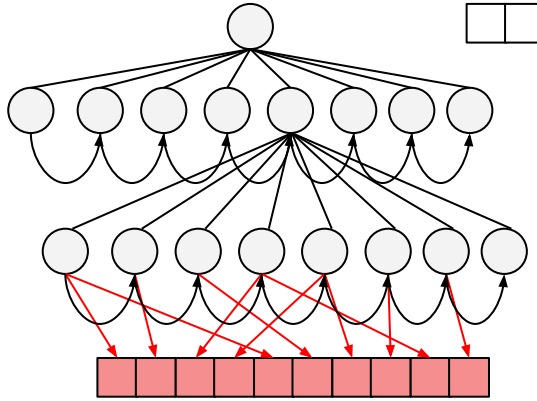
Implementación en GPU

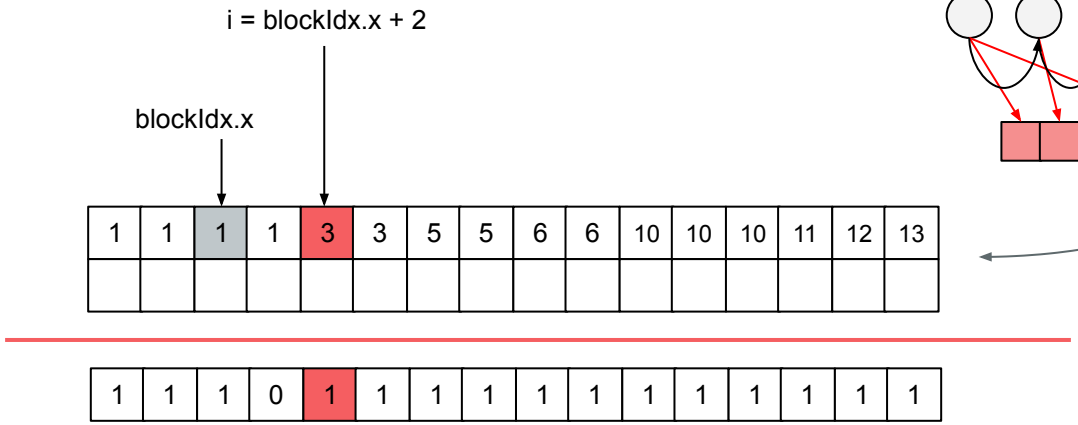




Actualizamos la visibilidad del objeto i de forma atómica

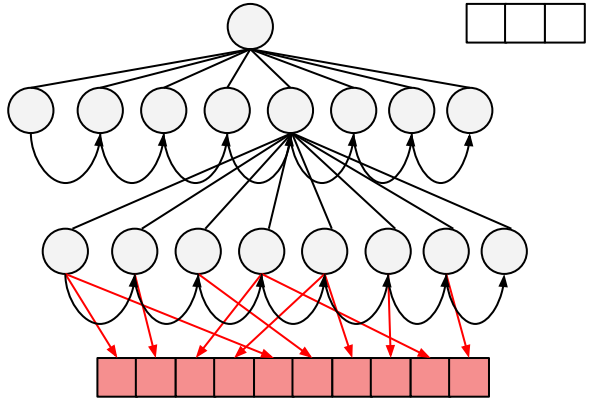
Implementación en GPU

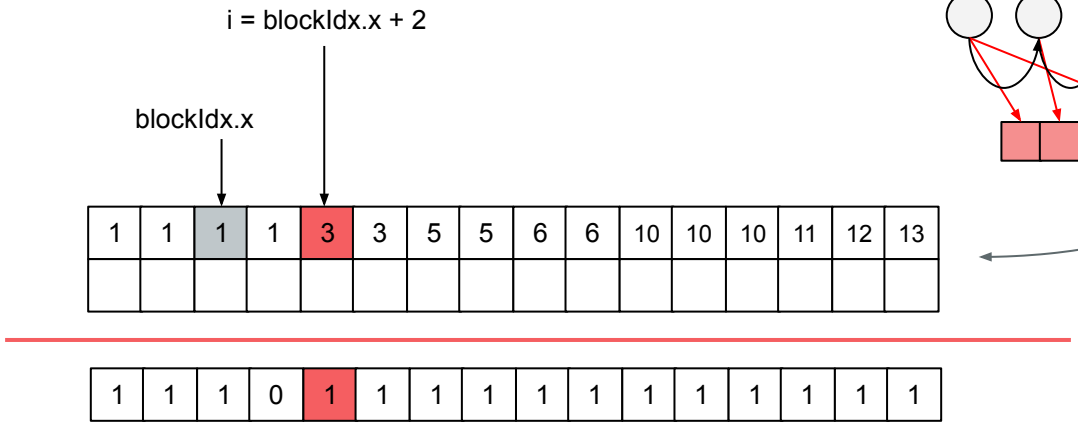




Array de visibilidad de los objetos

Implementación en GPU





Array de visibilidad de los objetos

1 ≠ 3

Se termina el bucle que itera hacia la derecha

Implementación en GPU

Ahora se realizaría el mismo proceso pero hacia la izquierda

Implementación en GPU

Finalmente

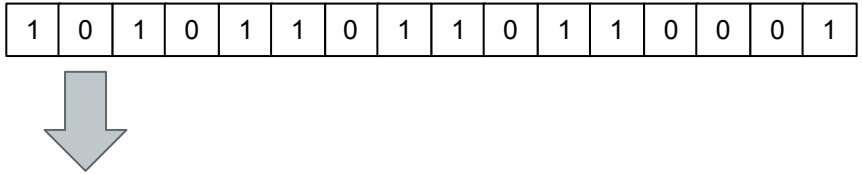
Se devuelve el array de visibilidad

1	0	1	0	1	1	0	1	1	0	1	1	0	0	0	1

Implementación en GPU

Finalmente

Se devuelve el array de visibilidad

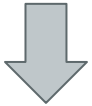


Se crea un nuevo array con los objetos visibles de este nivel y los objetos del nivel superior

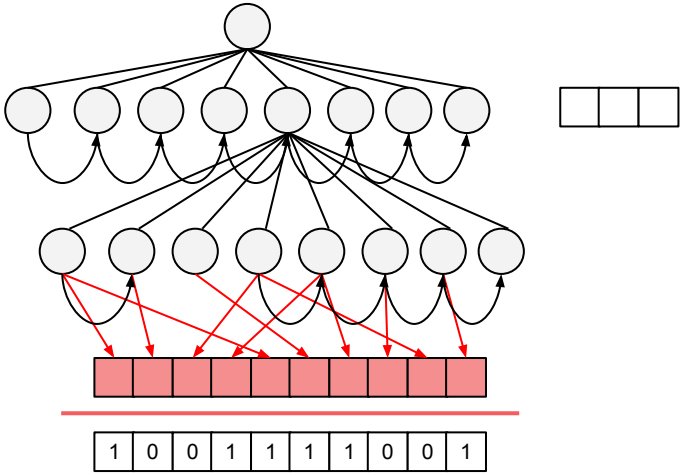
Implementación en GPU

Finalmente

Se devuelve el array de visibilidad



Se crea un nuevo array con los objetos visibles de este nivel y los objetos del nivel superior

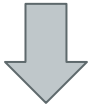


Array de visibilidad de los objetos

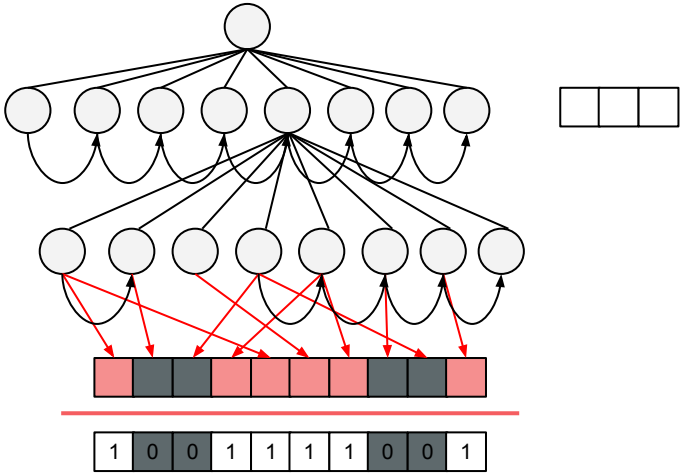
Implementación en GPU

Finalmente

Se devuelve el array de visibilidad



Se crea un nuevo array con los objetos visibles de este nivel y los objetos del nivel superior



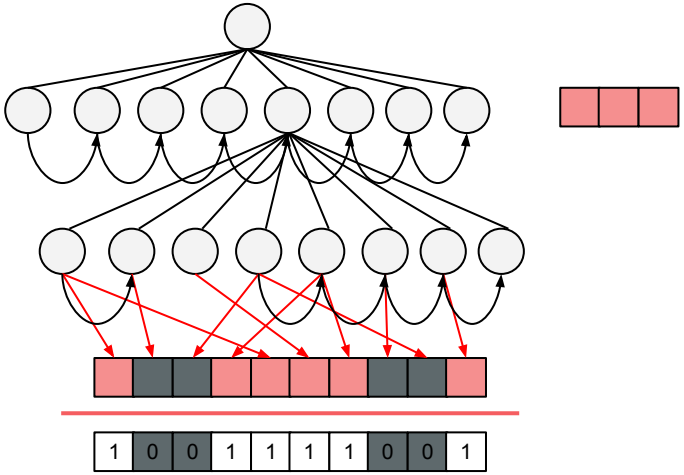
Array de visibilidad de los objetos

Implementación en GPU

Finalmente

Se devuelve el array de visibilidad

Se crea un nuevo array con los objetos visibles de este nivel y los objetos del nivel superior



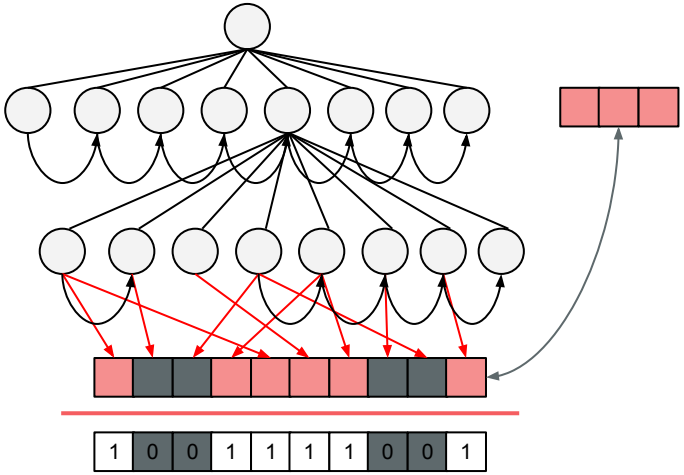
Array de visibilidad de los objetos

Implementación en GPU

Finalmente

Se devuelve el array de visibilidad

Se crea un nuevo array con los objetos visibles de este nivel y los objetos del nivel superior



Array de visibilidad de los objetos

Implementación en GPU

Finalmente

Se devuelve el array de visibilidad

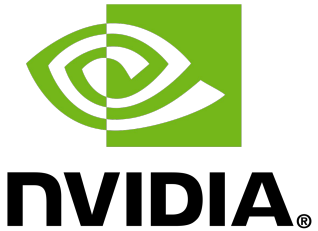
Se crea un nuevo array con los objetos visibles de este nivel y los objetos del nivel superior

Repetimos el proceso hasta llegar a la raíz del árbol

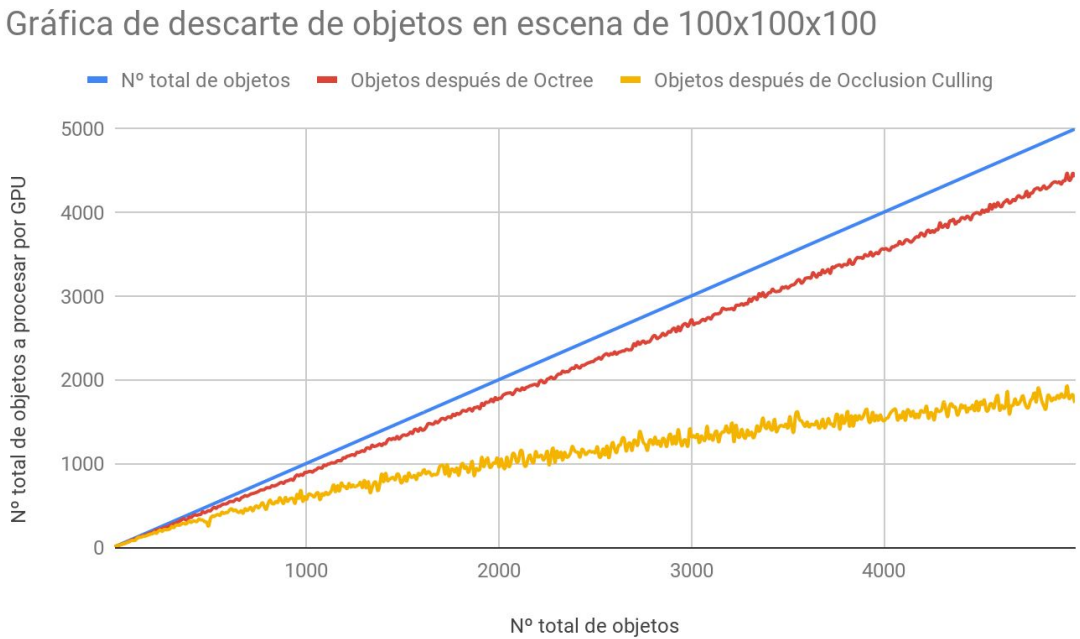
Análisis de resultados

Análisis de resultados

CPU	i7-7700HQ
Cantidad de RAM	16 GB
GPU	NVIDIA GTX 1050
Sistema Operativo	Ubuntu 18.04
GPU Drivers	NVIDIA drivers 390.116



Análisis de resultados

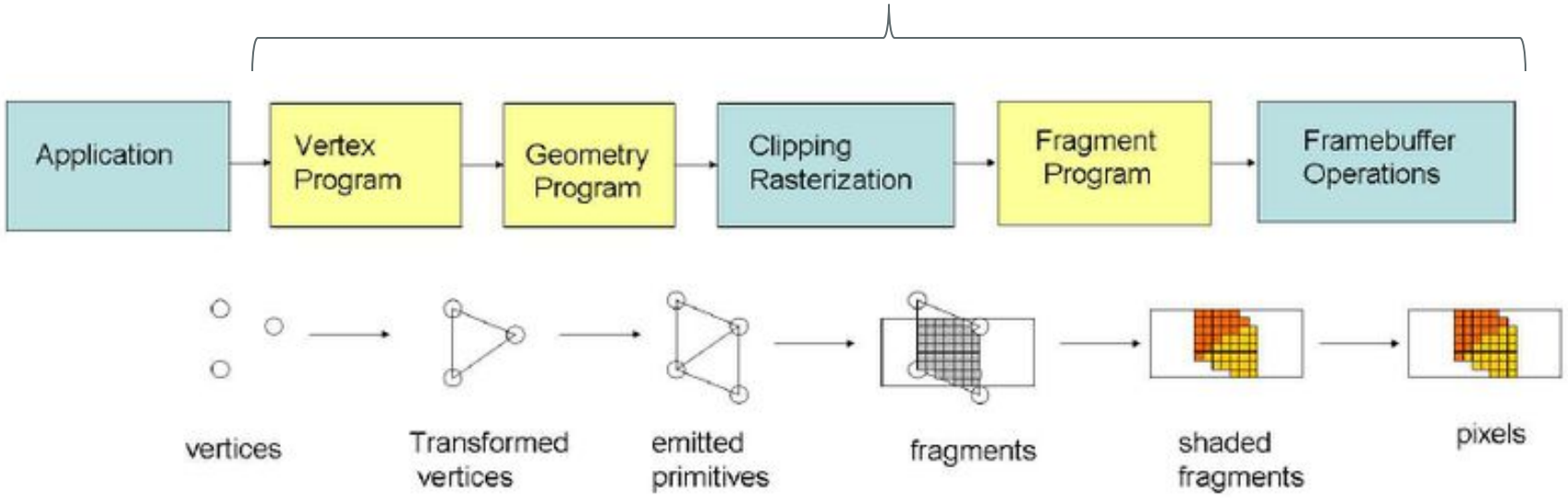


Análisis de resultados

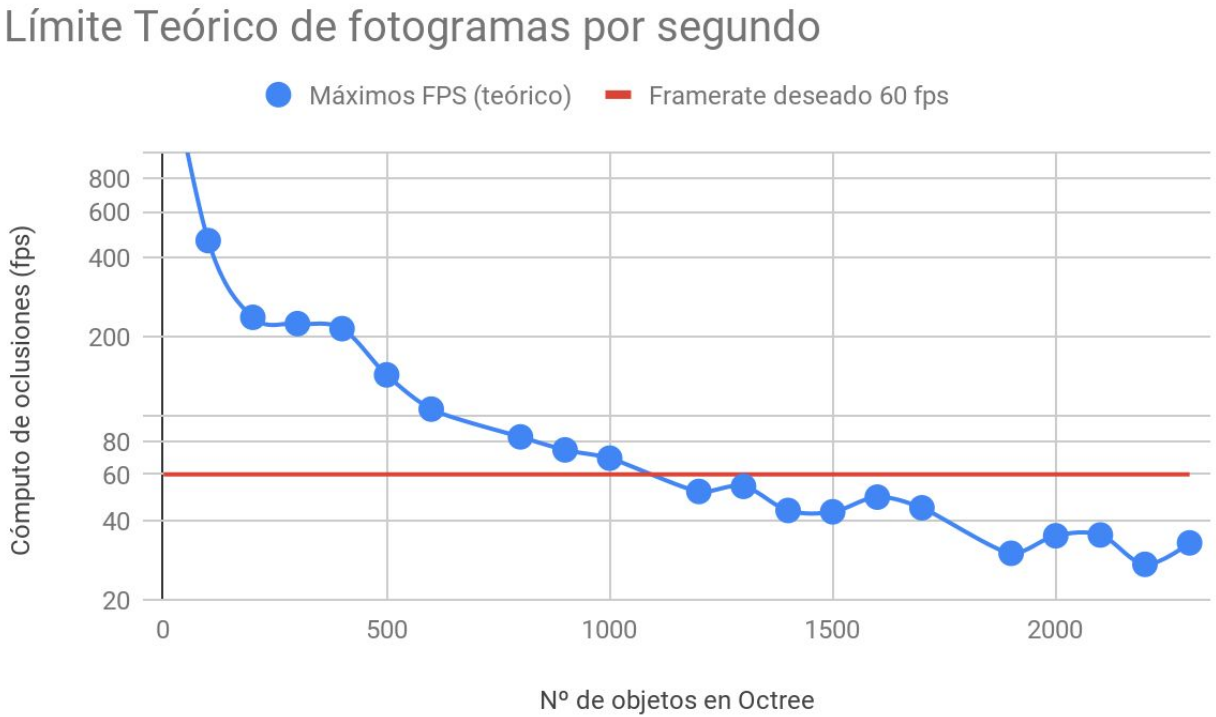
Vamos a estudiar el límite teórico de fotogramas posibles al usar nuestra solución

Análisis de resultados

Suponemos que este proceso es instantáneo y tarda 0 ms



Análisis de resultados



Demostración