SÍNTESIS DE IMÁGENES USANDO RAY TRACING

PROCESOS PARALELOS Y DISTRIBUIDOS

Maximiliano Monterrubio Gutiérrez

Introducción

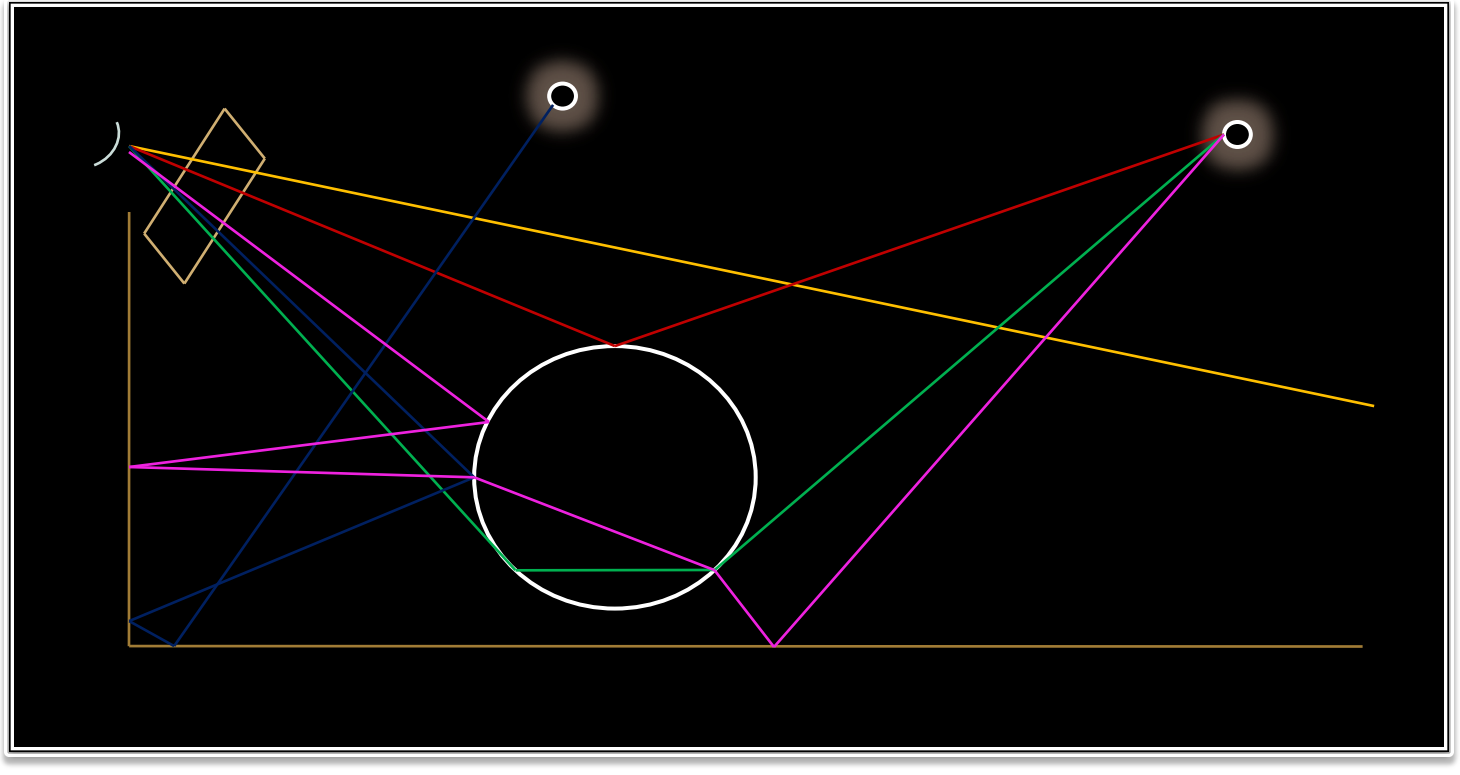
¿Qué es un *Ray Tracer*?

¿Cómo funciona?

¿Qué es un *Ray Tracer*?

- Es un algoritmo de síntesis de imágenes tridimensionales resultado de colocar un observador detrás de un plano focal.
- Capturamos los rayos de luz incidentes en el plano focal del observador por medio del concepto de rayo, el cual es en realidad una recta paramétrica , e intersecciones con objetos. Simula el viaje de la luz en sentido inverso.
- El algoritmo es sencillo y muy fácil de extender.

¿Cómo funciona un *Ray Tracer*?

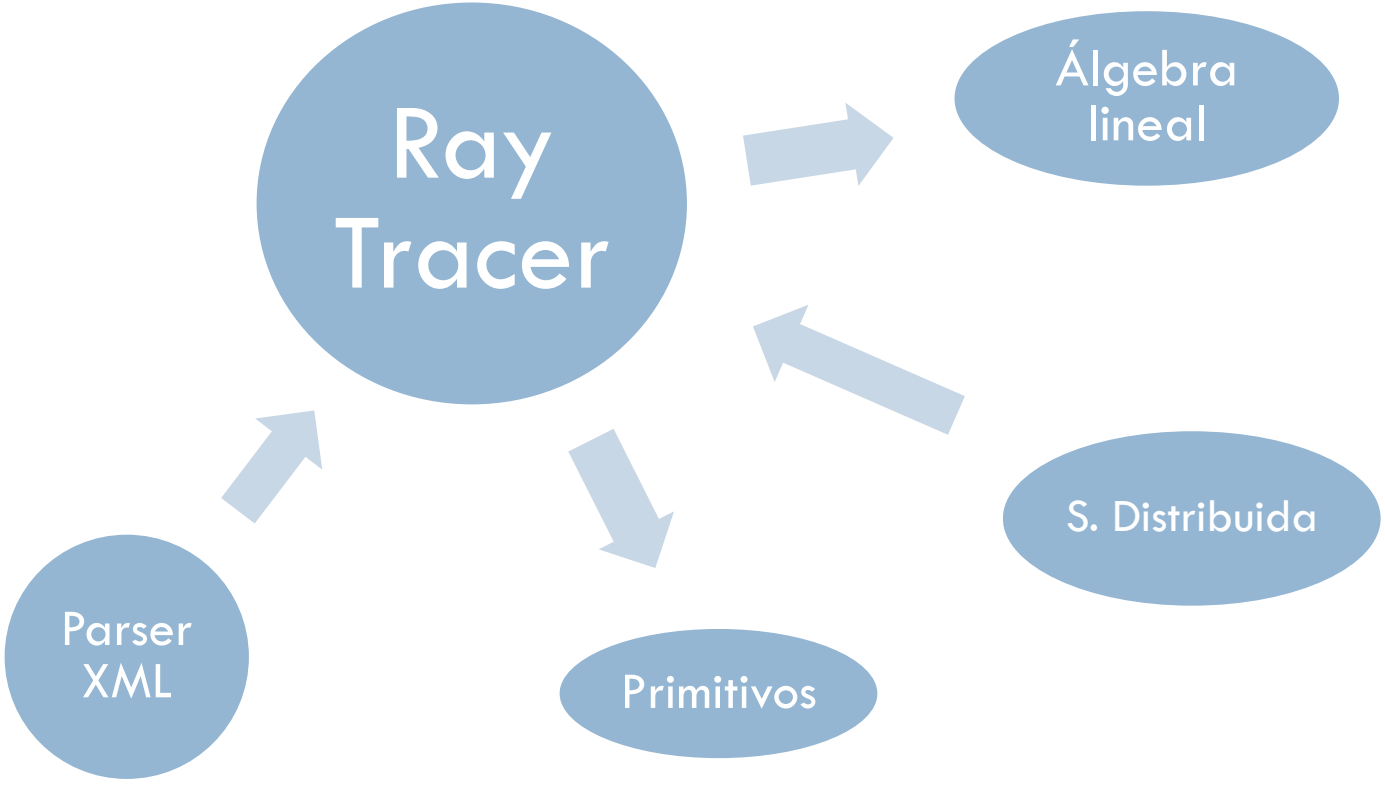


Características del Proyecto

¿Cómo está organizada la aplicación?

¿Qué se puede hacer con ella?

Diagrama de bloques



El parser de documentos de escena

- La entrada del programa es un documento XML que especifica la escena a sintetizar.
- El documento XML especifica parámetros globales como la posición de la cámara, resolución del plano focal y número de muestras por píxel.
- Adicionalmente se especifican todos los primitivos, tanto ubicación como geometría, así como el material de cada primitivo.

Características globales de una escena

- Resolución del plano focal expresado como ancho y altura medida en píxeles.
- Ubicación del observador en el espacio tridimensional.
- Vector de dirección del observador (hacia dónde voltea).
- Número de muestras (rayos a *disparar*) por píxel.

Primitivos soportados

- Objetos trazables:
  - Cubos alineados a los ejes *X*, *Y*, *Z*.
  - Planos infinitos (notación posición/normal).
  - Esferas.
- Iluminación
  - Luz puntual omnidireccional.
  - Luz de superficie.

Materiales

- Un material es un parámetro de un primitivo que especifica cómo refleja la luz (se le dice material porque especifica ***de qué está hecho***).
- Parámetros soportados:
  - Color del material.
  - Coeficiente difuso.
  - Coeficiente especular.
  - Dureza especular.
  - Coeficiente de reflexión.
  - Coeficiente de refracción.
  - Índice de refracción.

El subsistema de álgebra lineal

- Hay que hacer las cuentas.
- Se implementaron:
  - Vectores tridimensionales.
  - Matrices de transformación.
  - Representación de punto flotante de colores en las bandas RGB.

El motor de síntesis

Funcionalidad y características

Características generales del motor

- Algoritmo de *Ray Tracing* recursivo con rebote configurable.
- Soporte de multimuestreo por píxel.
- Implementación paralela libre de espera y libre de bloqueos.

Manejo de materiales

- *Shader* difuso con modelo de iluminación de **Lambert.**
- *Shader* especular.
- Soporte para reflexión y refracción haciendo *ray tracing* recursivo.

Ray tracing multithreaded

¿Cómo paralelizar este proceso?

¿Qué decisiones se tomaron para implementar la síntesis multithread?

Paralelizando el motor

- El proceso de paralelizar el motor de síntesis es **muy** sencillo, ya que cada muestra (rayo a disparar) es totalmente independiente de las demás.
- Simplemente hay que particionar el plano focal y distribuir uniformemente dicha partición entre el número total de hilos de ejecución.
- ¡No tan rápido! Existen consideraciones que hay que tomar.

Consideraciones a tomar.

- Balanceo de carga
  - ¿Qué hay que hacer para que se balancee lo más uniformemente posible la carga para cada hilo?
- Caché
  - Hay que sacarle jugo, una implementación ingenua puede quitarnos esta ventaja.

El balanceador de cargas

- Intenta resolver los problemas mencionados.
- Solución para el caché:
  - **Partir la imagen en pequeños cuadrados y sintetizar por localidades:** Esto intenta minimizar la penalización de fallo en el caché.
- Solución para el balanceo de carga por hilo:
  - **Aleatorizar la lista de cuadrados:** Así la probabilidad de que un hilo le toque una escena muy poblada de primitivos se distribuye más uniformemente.

Rendering Distribuido

¿Cómo se implementó?

¿Qué decisiones se tomaron para implementar este componente?

¿Cómo funciona?

- Arquitectura cliente/servidor.
- Por medio de **TCP/IP**.
- El servidor dice qué escena se va a sintetizar y qué le toca a cada quién.
- El cliente realiza el trabajo dado por el servidor y envía sus resultados.
- Se establece un protocolo sencillo de comunicación entre el servidor y los clientes.

El protocolo

- Servidor
  - Contesto enviando la escena en XML
  - Voy recibiendo velocidad/hilos por cada nodo.
  - Al terminar de recibirlos todos, envío las piezas que le toca a cada quien ya balanceadas.
  - Al ir recibiendo resultados, voy mostrando en pantalla
- Cliente
  - Me conecto al servidor
  - Recibo la escena y envío cuántos hilos voy a ejecutar y mi velocidad de reloj.
  - Recibo las piezas que me tocan y procedo a sintetizar.
  - Envío mis resultados progresivamente.

Cosas por hacer

¿Qué faltó?

¿Por qué faltó?

¿Qué habría que hacer para implementarlo?

¿Qué falto?

- Materiales:
  - Ley de *Beer* en refracción de luz (modela la atenuación a través del medio denso).
  - Texturas.
  - Ecuaciones de Fresnel.
- Algoritmos de iluminación global:
  - *Ambient Occlusion*.
  - *Path tracing*.
- Más primitivos:
  - Mallas de triángulos.
  - NURBS (Non-Uniform Rational B-Spline).

Resultados

¿Cómo se comportó?

¿Qué problemas hubo?

¿Cómo se comportó?

- Motor de síntesis
  - Calidad de imagen buena.
- Síntesis en paralelo
  - Una decepcionante mejora en desempeño entre 15% y 20% al usar los dos núcleos. (Blender3D lo hace muy cerca del doble de velocidad).
  - Introducción de ruido de sal y pimienta. Al parecer resultado de un problema de sincronización (NO ENTIENDO POR QUÉ).
- Síntesis distribuida
  - Buenos resultados, las cargas se balancean y no hubieron problemas de red.
  - Se probó en Internet y aunque jala muy lento por la red, no hubieron problemas de transmisión de datos.

Demostración