UNIVERSIDAD
DE GRANADA

Raytracing (WIP name)

Doble grado en ingeniería informática y matemáticas

Presentado por: Andrés Millán Muñoz,
Tutorizado por: Carlos Ureña Almagro, María del Carmen Segovia García

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Facultad de Ciencias

February 17, 2022

Contents

1 Abstract

Se procederá a analizar los algoritmos modernos de visualización 3D realista usando métodos de Monte-Carlo, y su implementación en hardware gráfico moderno (GPUs) específicamente diseñadas para aceleración de Ray-Tracing. Se diseñará e implementará un sistema software de síntesis de imágenes realistas por path tracing y muestreo directo de fuentes de luz, que haga uso del hardware gráfico, y se analizará su eficiencia en tiempo en relación a la calidad de las imágenes y en comparación con una implementación exclusivamente sobre CPU.

Se realizará una revisión bibliográfica de los métodos de Montecarlo que se aplican de manera habitual para la visualización de imagenes 3D. Se examinarán los puntos fuertes y débiles de cada una de las técnicas, con el objetivo de minimizar el error en la recosntrucción de la imagen sin que esto suponga un alto coste computacional. Se investigarán las soluciones propuestas para el futuro del área.

Translation. It'll be left as is until there's a definitive abstract

2 Dedicatoria

Aquí es donde me pongo ñoño

¡Parece que has llegado un poco pronto! Si lo has hecho voluntariamente, ¡muchas gracias! Este proyecto debería estar finalizado en verano de 2022.

Mientras tanto, actualizaré poco a poco el contenido. Si quieres ir comprobando los progresos, puedes visitar Asmilex/Raytracing en Github para ver el estado del desarrollo.

3 Introducción

Ser capaces de capturar un momento.

Desde siempre, este ha sido uno de los sueños de la humanidad. La capacidad de retener lo que ven nuestros ojos comenzó con simples pinturas ruprestres. Con el tiempo, el arte evolucionó, así como la capacidad de retratar nuestra percepción con mayor fidelidad.

A inicios del siglo XVIII, se caputaron las primeras imágenes con una cámara gracias a Nicéphore Niépce. Sería una imagen primitiva, claro; pero era funcional. Gracias a la compañía Kodak, la fotografía se extendió al consumidor rápidamente sobre 1890. Más tarde llegaría la fotografía digital, la cual simplificaría muchos de los problemas de las cámaras tradicionales.

Hablando de digital. Los ordenadores personales modernos nacieron unos años más tarde. Los usuarios eran capaces de mostrar imágenes en pantalla, que cambiaban bajo demanda. Y, entonces, nos hicimos una pregunta...

¿Podríamos simular la vida real para mostrarla en pantalla?

Como era de esperar, esto es complicado de lograr. Para conseguirlo, hemos necesitado crear abstracciones de conceptos que nos resultan naturales, como objetos, luces y seres vivos. “Cosas” que un ordenador no entiende, y sin embargo, para nosotros funcionan.

Así, nació la geometría, los puntos de luces, texturas, sombreados, y otros elementos de un escenario digital. Pero, por muchas abstracciones elegantes que tengamos, no nos basta. Necesitamos visualizarlas. Y como podemos imaginarnos, esto es un proceso costoso.

La rasterización es el proceso mediante el cual estos objetos tridimensionales se transforman en bidimensionales. Proyectando acordemente el entorno a una cámara, conseguimos colorear un pixel, de forma que represente lo que se ve en ese mundo.

[TODO insertar imagen rasterización. NOTE quizás debería extender un poco más esta parte? Parece que se queda algo coja la explicación.]

Aunque esta técnica es bastante eficiente en términos de computación y ha evolucionado mucho, rápidamente saturamos sus posibilidades. Conceptos como shadow maps, baked lightning, o reflection cubemaps intentan solventar lo que no es posible con rasterización: preguntrarnos qué es lo que se encuentra alrededor nuestra.

En parte, nos olvidamos de la intuitiva realidad, para centrarnos en aquello computacionalmente viable.

Y, entonces, en 1960 el trazado de rayos con una simple idea intuitiva .

3.1 ¿Qué es ray tracing?

En resumidas cuentas, ray tracing (o trazado de rayos en español), se basa en disparar fotones desde nuestras luces digitales y hacerlos rebotar en la escena.

De esta forma, simulamos cómo se comporta la luz. Al impactar en un objeto, sufre un cambio en su trayectoria. Este cambio origina nuevos rayos, que vuelven a dispersarse por la escena. Estos nuevos rayos dependerán de las propiedades del objeto con el que hayan impactado. Con el tiempo necesario, lo que veremos desde nuestra cámara será una representación fotorealista de lo que habita en ese universo.

Esta técnica, tan estúpidamente intuitiva, se ha hecho famosa por su simpleza y su elegancia. Pues claro que la respuesta a "¿Cómo simulamos fielmente una imagen en un ordenador?" es "Representando la luz de forma realista".

Aunque, quizás intuitiva no sea la palabra. Podemos llamarla natural, eso sí. A fin de cuentas, fue a partir del siglo XVIII cuando empezamos a entender que podíamos capturar la luz. Nuestros antepasados tenían teorías, pero no podían explicar por qué veíamos el mundo.

Ahora sí que sabemos cómo funciona. Entendiendo el por qué lo hace nos permitirá programarlo. Y, resulta que funciona impresionantemente bien.

Atrás se quedan los hacks necesarios para rasterización. Los cubemaps no son esenciales

para los reflejos, y no necesitamos cámaras virtuales para calcular sombras. Ray tracing permite simular fácilmente efectos como reflejos, refracción, desenfoque de movimiento, aberración cromática... Incluso fenómenos físicos propios de las particulas y las ondas.

> Espera. Si tan bueno es, ¿por qué no lo usamos en todos lados?

Por desgracia, el elefante en la sala es el rendimiento. Como era de esperar, disparar rayos a diestro y siniestro es costoso. Muy costoso.

A diferencia del universo, nosotros no nos podemos permitir el lujo de usar fotones de tamaño infinitesimal y dispersiones casi infinitas. Nos pasaríamos una eternidad esperando. Y para ver una imagen en nuestra pantalla necesitaremos estar vivos, claro.

Debemos evitar la fuerza bruta. Dado que la idea es tan elegante, la respuesta no está en el "qué", sino en el "cómo". Si disparamos y dispersamos rayos con cabeza seremos capaces de obtener lo que buscamos en un tiempo razonable.

Hace unos años, al hablar de tiempo razonable, nos referiríamos a horas. Quizás días. Producir un frame podría suponer una cantidad de tiempo impensable para un ordenador de consumidor. Hoy en día también ocurre esto, claro está. Pero la tecnología evoluciona.

Podemos bajarlo a milisegundos.

Hemos entrado en la era del real time ray tracing.

3.2 Vale, ¿y qué vamos a hacer entonces?

TODO hablar de los objetivos del trabajo.

Referencias que pasar después:

1. https://www.wikiwand.com/en/History_of_photography#/1816_to_1833:_Ni%C3%A9pce's_earlie
2. https://www.wikiwand.com/es/Kodak#/Historia
3. https://www.wikiwand.com/en/Computer#/Digital_computers
4. https://www.wikiwand.com/en/Rendering_(computer_graphics)#/Chronology_of_important_pu
5. Ray tracing gems I (p 16), gems II.

6. https://blogs.nvidia.com/blog/2018/03/19/whats-difference-between-ray-tracing-rasterization/
7. https://www.wikiwand.com/en/Ray_tracing_(graphics)
8. https://sciencebehindpixar.org/pipeline/rendering#:~:text=They%20said%20it%20takes%20at,
9.

4 Notación

Antes de comenzar, asentemos la notación que utilizaremos.

Para denotar a los puntos, usaremos letras mayúsculas como P o Q. Los escalares vendrán dados por letras minúsculas, como a o b; mientras que los vectores irán en letra minúscula negrita (p.e.: v o w). Además, serán vectores columnas. Aquellos normalizados los representaremos con un gorrito: $\hat{\mathrm{v}} = \frac{\mathrm{v}}{\|\mathrm{v}\|}$. Las matrices, por otra parte, vendrán dadas por letra mayúscula en negrita, como M. También son columna.

El producto escalar vendrá dado por $\mathrm{v} \cdot \mathrm{w}$, y el vectorial por $\mathrm{v} \times \mathrm{w}$.

La notación usada para las variables aleatorias será la habitual: mayúsculas como X. Su valor esperado vendrá dado por $E[X]$ y la varianza por $V[X]$.

> TODO: notación para las funciones de densidad y distribución. TODO: acceso a componentes de un vector/matriz?

5 ¿Cómo funciona ray tracing?

6 Metodología; o cómo se hizo este trabajo

TODO - hablar de las fases de desarrollo. Interpretación propia de Agile. Documentación y código desarrollado a la par, mediante issues. Adaptación de los requisitos conforme se avanza. Beneficios de una página web (seguramente debería ser su propia sección)

6.1 Github

TODO - Hablar de cómo se utiliza Github y sus tecnologías para agrupar todo el trabajo. Hablar de la guía de estilos, y cómo los emojis ayudan a identificar rápidamente secciones.

6.1.1 Github Actions

TODO - Hablar de cómo se usa el sistema de integración continua para construir la web y el pdf

6.1.2 Github Projects

TODO - Hablar de cómo se gestiona el trabajo mediante issues, recapitulados todos con Projects.