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

March 3, 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.: $\mathbf{v}$ o $\mathbf{w}$). Además, serán vectores columnas. Aquellos normalizados los representaremos con un gorrito: $\hat{\mathbf{v}} = \frac{\mathbf{v}}{\|\mathbf{v}\|}$. Las matrices, por otra parte, vendrán dadas por letra mayúscula en negrita, como $\mathbf{M}$. También son columna.

El producto escalar vendrá dado por $\mathbf{v} \cdot \mathbf{w}$, y el vectorial por $\mathbf{v} \times \mathbf{w}$. Cuando escribamos $\mathbf{v}^2$, lo entenderemos como $\mathbf{v} \cdot \mathbf{v}$

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 Los fundamentos

Empecemos por definir lo que es un rayo.

Un rayo es una función $P(t) = O + tD$, donde O es el origin, D la dirección, y $t \in \mathbb{R}$. Podemos considerarlo una interpolación entre dos puntos en el espacio, donde t controla la posición en la que nos encontramos.

Por ejemplo, si $t = 0$, obtendremos el origen. Si $t = 1$, obtendremos el punto correspondiente a la dirección. Usando valores negativos vamos hacia atrás.

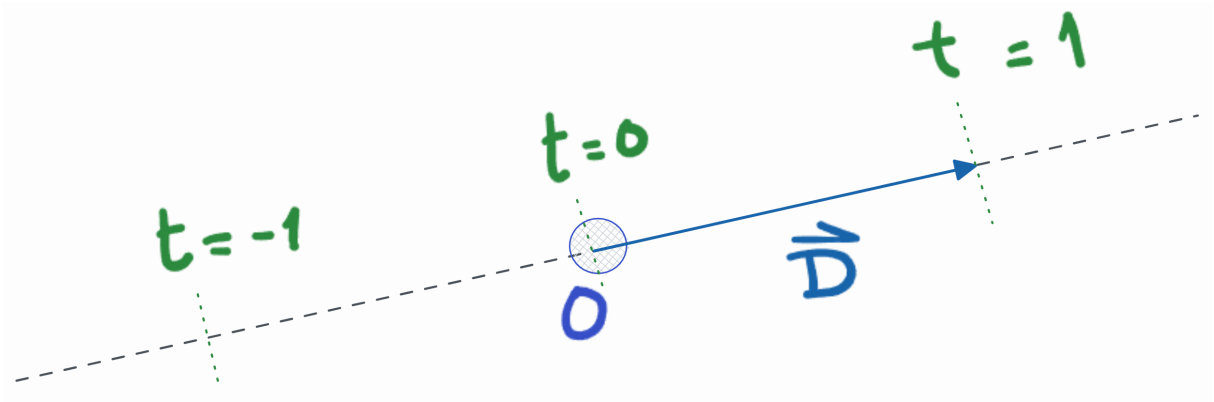

Figure 5.1

Dado que estos puntos estarán generalmente en $\mathbb{R}^3$, podemos escribirlo como

$$P(t) = (O_x, O_y, O_z) + t(D_x, D_y, D_z)$$

Estos rayos los dispararemos a través de una cámara virtual, que estará enfocando a la escena. De esta forma, los haremos rebotar con los objetos que se encuentren en el camino del rayo. A este proceso lo llamaremos ray casting.

> TODO foto de cámara - pixel - rayo - objeto.

Generalmente, nos quedaremos con el primer objeto que nos encontremos en su camino. Aunque, a veces, nos interesará saber todos con los que se encuentre.

Cuando un rayo impacta con un objeto, adquirirá parte de las propiedades lumínicas del punto de impacto. Por ejemplo, cuánta luz proporciona la lámpara que tiene encima la esfera de la figura anterior.

Una vez recojamos la información que nos interese, aplicaremos otro raycast desde el nuevo punto de impacto, escogiendo una nueva dirección determinada. Esta dirección dependerá del tipo de material del objeto. Y, de hecho, algunos serán capaces de invocar varios rayos.

Por ejemplo, los espejos reflejan la luz casi de forma perfecta; mientras que otros elementos como el agua o el cristal reflejan y refractan luz, así que necesitaremos generar dos nuevos raycast.

Usando suficientes rayos obtendremos la imagen de la escena. A este proceso de ray casting recursivo es lo que se conoce como ray tracing.

Como este proceso puede continuar indefinidamente, tendremos que controlar la profundidad de la recursión. A mayor profundidad, mayor calidad de imagen; pero también, mayor tiempo de ejecución.

5.1 Eligiendo direcciones

Una de las partes más importantes de ray tracing, y a la que quizás dedicaremos más tiempo, es a la elección de la dirección.

Hay varios factores que entran en juego a la hora de decidir qué hacemos cuando impactamos con un nuevo objeto:

1. ¿Cómo es la superficie del material? A mayor rugosidad, mayor aleatoriedad en la dirección. Por ejemplo, no es lo mismo el asfalto de una carretera que una lámina de aluminio impecable.
2. ¿Cómo de fiel es nuestra geometría?
3. ¿Dónde se encuentran las luces en la escena? Dependiendo de la posición, nos interesará muestrear la luz con mayor influencia.

Estas cuestiones las exploraremos a fondo en las siguientes secciones.

5.2 Intersecciones rayo - objeto

Como dijimos al principio del capítulo, representaremos un rayo como

$$\begin{aligned} P(t) &= (O_x, O_y, O_z) + t(D_x, D_y, D_z) = \\ &= (O_x + tD_x, O_y + tD_y, O_y + tD_z) \end{aligned}$$

Por ejemplo, tomando $O = (1, 3, 2), D = (1, 2, 1)$:

- Para $t = 0$, $P(t) = (1, 3, 2)$.
- Para $t = 1$, $P(t) = (1, 3, 2) + (1, 2, 1) = (2, 5, 3)$.

Nos resultará especialmente útil limitar los valores que puede tomar t. Restringiremos los posibles puntos del dominio de forma que $t \in [t_{min}, t_{max})$, con $t_{min} < t_{max}$. En general, nos interesará separarnos de las superficies un pequeño pero no despreciable ε para evitar errores de redondeo.

> TODO dibujo origen - epsilon == tmin -> rayo -> t_max = 1-epsilon

Una de las principales cuestiones que debemos hacernos es saber cuándo un rayo impacta con una superficie. Lo definiremos analíticamente.

5.2.1 Superficies implícitas

Generalmente, cuando hablemos de superficies, nos referiremos superficies diferenciables, pues nos interesará conocer el vector normal en cada punto.

Una superficie implícita es una superficie en un espacio euclidiano definida como

$$F(x, y, z) = 0$$

Esta ecuación implícita define una serie de puntos del espacio $\mathbb{R}^3$ que se encuentran en la superficie.

Por ejemplo, la esfera se define como $x^2 + y^2 + z^2 - 1 = 0$.

Consideremos una superficie S y un punto regular de ella P; es decir, un punto tal que el gradiente de F en P no es 0. Se define el vector normal n a la superficie en ese punto como

$$\mathrm{n} = \nabla F(P) = \left(\frac{\partial F(P)}{\partial x}, \frac{\partial F(P)}{\partial y}, \frac{\partial F(P)}{\partial z} \right)$$

> TODO: dibujo de la normal a una superficie.

Dado un punto $Q \in \mathbb{R}^3$, queremos saber dónde interseca un rayo $P(t)$. Es decir, para qué t se cumple que $F(P(t)) = 0 \iff F(O + tD) = 0$.

Consideremos por ejemplo un plano, como en (Shirley and Morley 2003). Para ello, nos tomamos un punto Q_0 del plano y un vector normal a la superficie n.

La ecuación implícita del plano será

$$F(Q) = (Q - Q_0) \cdot \mathrm{n} = 0$$

Si pinchamos nuestro rayo en la ecuación,

$$\begin{aligned} F(P(t)) &= (P(t) - Q_0) \cdot \mathrm{n} \\ &= (O + tD - Q_0) \cdot \mathrm{n} = 0 \end{aligned}$$

Resolviendo para t, esto se da si

$$\begin{aligned} O \cdot \mathrm{n} + tD \cdot \mathrm{n} - Q_0 \cdot \mathrm{n} = 0 & \iff \\ tD \cdot \mathrm{n} = Q_0 \cdot \mathrm{n} - O \cdot \mathrm{n} & \iff \\ t = \frac{Q_0 \cdot \mathrm{n} - O \cdot \mathrm{n}}{D \cdot \mathrm{n}} & \end{aligned}$$

Es decir, hemos obtenido el único valor de t para el cual el rayo toca la superficie.

Debemos tener en cuenta el caso para el cual $D \cdot \mathrm{n} = 0$. Esto solo se da si la dirección y el vector normal a la superficie son paralelos.

> TODO: dibujo de dos rayos con un plano: uno corta a la superficie, mientras que el otro es paralelo.

5.2.2 Superficies paramétricas

Otra forma de definir una superficie en el espacio es mediante un subconjunto $D \subset \mathbb{R}^2$ y una serie de funciones, $f, g, h : D \to \mathbb{R}^3$, de forma que

$$(x, y, z) = (f(u, v), g(u, v), h(u, v))$$

> En informática gráfica, hacemos algo similar cuando mapeamos una textura a una superficie. Se conoce como UV mapping

Demos un par de ejemplos de superficies paramétricas: - El grafo de una función $f : D \to \mathbb{R}^3$,

$$G(f) = \{(x, y, f(x, y)) \,|\, (x, y) \in D\}$$

define una superficie diferenciable siempre que f también lo sea. - Usando coordenadas esféricas (r, θ, ϕ), podemos parametrizar la esfera como $(x, y, z) = (\cos\phi \sin\theta, \sin\phi \sin\theta, \cos\theta)$

> TODO añadir imagen de coordenadas esféricas. U otro capítulo con coordenadas.

> NOTE: estoy usando (radial, polar, azimuthal). θ corresponde con la apertura con respecto a la vertical

El vector normal n a la superficie en un punto (u, v) del dominio viene dado por

$$\mathrm{n}(u,v) = \left(\frac{\partial f}{\partial u}, \frac{\partial g}{\partial u}, \frac{\partial h}{\partial u}\right) \times \left(\frac{\partial f}{\partial v}, \frac{\partial g}{\partial v}, \frac{\partial h}{\partial v}\right)$$

Encontrar el punto de intersección de una superficie paramétrica con un rayo es sencillo. Basta con encontrar aquellos puntos (u, v) y t para los que

$$\begin{aligned} O_x + tD_x &= f(u,v) \\ O_y + tD_y &= g(u,v) \\ O_z + tD_z &= h(u,v) \end{aligned}$$

Es posible que el rayo no impacte en ningún punto. En ese caso, el sistema de ecuaciones no tendría solución. Otra posibilidad es que intersequen en varios puntos.

5.2.3 Intersecciones con esferas

Estudiemos ahora cómo intersecan las esfera con nuestro rayo. Una esfera de centro C y radio r viene dada por aquellos puntos $P = (x, y, z)$ que cumplen

$$(P - C) \cdot (P - C) = r^2$$

Podemos reescribir esta ecuación en términos de sus coordenadas para obtener

$$(x - C_x)^2 + (y - C_y)^2 + (z - C_z)^2 = r^2$$

Veamos para qué valores de t de nuestro rayo se cumple esa ecuación:

$$\begin{aligned} (P(t) - C) \cdot (P(t) - C) = r^2 \quad &\iff \\ (O + tD - C) \cdot (O + tD - C) = r^2 \quad &\iff \end{aligned}$$

Aplicando las propiedades del producto escalar de la conmutatividad ($a \cdot b = b \cdot a$) y la distributiva ($a \cdot (b + c) = a \cdot b + a \cdot c$), podemos escribir

$$\begin{aligned} ((O - C) + tD) \cdot ((O - C) + tD) = r^2 \quad &\iff \\ (O - C)^2 + 2 \cdot (O - C) \cdot tD + (tD)^2 = r^2 \quad &\iff \\ D^2t^2 + 2D \cdot (O - C)t + (O - C)^2 - r^2 = 0 \quad &\iff \end{aligned}$$

Así que tenemos una ecuación de segundo grado. Resolviéndola, nos salen nuestros puntos de intersección:

$$t = \frac{-D \cdot (O - C) \pm \sqrt{(D \cdot (O - C))^2 - 4(D^2)((O - C)^2 - r^2)}}{2D^2}$$

Debemos distinguir tres casos, atiendiendo al valor que toma el discriminante $\Delta = (D \cdot (O - C))^2 - 4(D^2)((O - C)^2 - r^2)$:

1. Si $\Delta < 0$, $\sqrt{\Delta} \notin \mathbb{R}$, y el rayo no impacta con la esfera
2. Si $\Delta = 0$, el rayo impacta en un punto, que toma el valor $t = \frac{-D \cdot (O - C)}{2D \cdot D}$. Digamos que pegaría justo en el borde.
3. Si $\Delta > 0$, existen dos soluciones. En ese caso, el rayo atraviesa la esfera.

> TODO dibujo explicativo de la intersección con una esfera.

Para estos dos últimos, si consideramos t_0 cualquier solución válida, el vector normal resultante viene dado por

$$\mathrm{n} = 2(P(t_0) - C)$$

o, normalizando,

$$\hat{\mathrm{n}} = \frac{(P(t_0) - C)}{r}$$

5.2.4 Intersecciones con triángulos

Este tipo de intersecciones serán las más útiles en nuestro path tracer. Generalmente, nuestras geometrías estarán compuestas por mallas de triángulos, así que conocer dónde impacta nuestro rayo será clave. Empecemos por la base:

Un triángulo viene dado por tres puntos, A, B, y C; correspondientes a sus vértices. Para evitar casos absurdos, supongamos que estos puntos son afinmente independientes; es decir, que no están alineados.

5.2.4.1 Coordenadas baricéntricas

Podemos describir los puntos contenidos en el plano que forman estos vertices mediante coordenadas baricéntricas. Este sistema de coordenadas expresa cada punto del plano como una combinación convexa de los vértices. Es decir, que para cada punto P del triángulo existen α, β y γ tales que $\alpha + \beta + \gamma = 1$ y

$$P = \alpha A + \beta B + \gamma C$$

> TODO: triángulo con coordenadas baricéntricas.

Debemos destacar que existen dos grados de libertad debido a la restricción de que las coordenadas sumen 1.

Una propiedad de estas coordenadas que nos puede resultar útil es que un punto P está contenido en el triángulo si y sólo si $0 < \alpha, \beta, \gamma < 1$.

Esta propiedad y la restricción de que sumen 1 nos da una cierta intuición de cómo funcionan. Podemos ver las coordenadas baricéntricas como la contribución de los vértices a un punto P. Por ejemplo, si $\alpha = 0$, eso significa que el punto viene dado por $\beta B + \gamma C$; es decir, una combinación lineal de B y C. Se encuentra en la recta que generan.

Por proponer otro ejemplo, si alguna de las coordenadas fuera mayor que 1, eso significaría que el punto estaría más allá del triángulo.

> TODO: dibujo con explicación de cómo funciona (libreta Shinrin - Yoku)

5.2.4.2 Calculando la intersección

Podemos eliminar una de las varibales escribiendo $\alpha = 1 - \beta - \gamma$, lo que nos dice

$$\begin{aligned} P &= (1 - \beta - \gamma)A + \beta B + \gamma C \\ &= A + (B - A)\beta + (C - A)\gamma \end{aligned}$$

bajo la restricción

$$\begin{aligned} \beta + \gamma &< 1 \\ 0 &< \beta \\ 0 &< \gamma \end{aligned} \tag{5.1}$$

Un rayo $P(t) = O + tD$ impactará en un punto del triángulo si se cumple

$$P(t) = O + tD = A + (B - A)\beta + (C - A)\gamma$$

cumpliendo [5.1]. Podemos expandir la ecuación anterior en sus coordenadas para obtener

$$\begin{aligned} O_x + tD_x &= A_x + (B_x - A_x)\beta + (C_x - A_x)\gamma \\ O_y + tD_y &= A_y + (B_y - A_y)\beta + (C_y - A_y)\gamma \\ O_z + tD_z &= A_z + (B_z - A_z)\beta + (C_z - A_z)\gamma \end{aligned}$$

Reordenamos:

$$\begin{aligned} (A_x - B_x)\beta + (A_x - C_x)\gamma + tD_x &= A_x - O_x \\ (A_y - B_y)\beta + (A_y - C_y)\gamma + tD_y &= A_y - O_y \\ (A_z - B_z)\beta + (A_z - C_z)\gamma + tD_z &= A_z - O_z \end{aligned}$$

Lo que nos permite escribir el sistema en forma de ecuación:

$$\begin{pmatrix} A_x - B_x & A_x - C_x & D_x \\ A_y - B_y & A_y - C_y & D_y \\ A_z - B_z & A_z - C_z & D_z \end{pmatrix} \begin{pmatrix} \beta \\ \gamma \\ t \end{pmatrix} = \begin{pmatrix} A_x - O_x \\ A_y - O_y \\ A_z - O_z \end{pmatrix}$$

Calcular rápidamente la solución a un sistema de ecuaciones lineales es un problema habitual. En (Shirley and Morley 2003) se utiliza la regla de Cramer para hacerlo, esperando que el compilador optimice las variables intermedias creadas. Nosotros no nos tendremos que preocupar de esto en particular, ya que el punto de impacto lo calculará la GPU gracias a las herramientras aportadas por KHR.

Para obtener el vector normal, podemos hacer el producto vectorial de dos vectores que se encuentren en el plano del triángulo. Como, por convención, los vértices se guardan en sentido antihorario visto desde fuera del objeto, entonces

$$\mathrm{n} = (B - A) \times (C - A)$$

Fuentes que he usado y que debería pasar:

- https://www.wikiwand.com/en/Implicit_surface
- https://www.wikiwand.com/en/Parametric_surface
- https://www.wikiwand.com/en/Barycentric_coordinate_system

6 Integración de Monte Carlo

> TODO: este capítulo seguramente debería ir más tarde. De esa forma, puedo introducir otros conceptos antes. De momento, se queda aquí.

La parte más importante de nuestro ray tracer es saber calcular la luz en un punto. Para ello, necesitaríamos hallar la radianza en dicho punto mediante la rendering equation. Sin embargo, es muy difícil resolverla. Tanto computacionalmente como analíticamente. Por ello, debemos atacar el problema desde otro punto de vista.

Las técnicas de Monte Carlo nos permitirán aproximar el valor que toman mediante una estimación. Utilizando muestreo aleatorio para evaluar integrales, seremos capaces de obtener un resultado suficientemente bueno.

Una de las propiedades más improtantes que tienen es la independencia del ratio de convergencia y la dimensionalidad del integrando. Sin embargo, dadas n nuestras, la convergencia a la solución correcta tiene un orden de $\mathcal{O}\left(n^{-1/2}\right)$. Es decir, para reducir el error a la mitad, necesitaríamos 4 veces más muestras.

6.1 Repaso de probabilidad

Necesitaremos unas cuantas nociones de variable aleatoria para poder entender la integración de Monte Carlo, así que vamos a hacer un breve repaso.

Una variable aleatoria X (v.a.) es, esencialmente, una regla que asigna un valor numérico a cada posibilidad de algún proceso de azar. Formalmente, una variable aleatoria es una función definida en un espacio de probabilidad $(\Omega, \mathcal{A}, P)$ asociado a un exprimento aleatorio:

$$X : \Omega \to \mathbb{R}$$

A Ω lo conocemos como espacio muestral (aquel conjunto de todas las posibilidades), $\mathcal{A}$ es una σ-álgebra de subconjuntos de Ω que refleja todas las posibilidades de eventos aleatorios. P es una función probabilidad, que asigna a cada evento una probabilidad.

> NOTE: no sé hasta qué punto debería meterme en la definición formal de variable aleatoria. Es una movida tremenda para poca cosa que necesitamos. De momento, voy con lo más interesante.

Una variable aleatoria X puede clasificarse en discreta o continua, dependiendo de cómo sea su rango $R_X = \{x \in \mathbb{R} \mid \exists \omega \in \Omega : X(\omega) = x\}$:

6.1.1 Variables aleatorias discretas

Las v.a. discretas son aquellas cuyo rango es un conjunto discreto.

Para comprender mejor cómo funcionan, pongamos un ejemplo.

Consideremos un experimento en el que lanzamos dos dados, anotando lo que sale en cada uno. Los posibles valores que toman serán

$$\begin{gathered}\{(1,1),(1,2),(1,3),(1,4),(1,5),(1,6),\\ (2,1),(2,2),(2,3),(2,4),(2,5),(2,6),\\ (3,1),(3,2),(3,3),(3,4),(3,5),(3,6),\\ (4,1),(4,2),(4,3),(4,4),(4,5),(4,6),\\ (5,1),(5,2),(5,3),(5,4),(5,5),(5,6),\\ (6,1),(6,2),(6,3),(6,4),(6,5),(6,6)\}\end{gathered}$$

Cada resultado tiene la misma probabilidad de ocurrir (claro está, si el dado no está trucado). Como hay 36 posibilidades, la probabilidad de obtener un cierto valor es de $\frac{1}{36}$.

La v.a. X denotará la suma de los valores obtenidos en cada uno. Así, por ejemplo, si los dados han dado $(1,3)$, X será 4. En total, X puede tomar todos los valores comprendidos

entre 2 y 12. Este sería el espacio muestral. Además, podemos observar que X puede obtener el mismo valor para dos resultados diferentes. Por ejemplo, $(1, 2)$ suma lo mismo que $(2, 1)$. Esto nos lleva a preguntarnos... ¿cuál es la probabilidad de que X adquiera un cierto valor?

La función masa de probabilidad nos permite conocer la probabilidad de que una variable aleatoria X tome un valor x. Se denota por $P(X = x)$.

En este ejemplo, la probabilidad de que X tome el valor 4 es

$$\begin{aligned} P(X = 4) &= \sum \text{nº parejas que suman 4} \cdot \text{probabilidad de que salga la pareja} \\ &= \frac{1}{36} \cdot 3 = \frac{1}{12} \end{aligned}$$

(Las parejas serían $(1, 3), (2, 2), (3, 1)$).

Por definición, si el espacio muestral de X es $x_1, \dots, x_n$, la función masa de probabilidad debe cumplir

$$\sum_{i=1}^{n} P(X = x_i) = 1$$

La función de distribución de una variable aleatoria X es

$$F_X(x) = P(X \leq x) = \sum_{k=-\infty}^{x} P(X = x) = 1$$

Es una función continua por la derecha y monótona no decreciente. Además, se cumple que $\lim_{x \to -\infty} F_X = 0$, $\lim_{x \to \infty} F_X = 1$.

En nuestro ejemplo, si consideramos $x = 3$:

$$\begin{aligned} F_X(x) &= \sum_{i=1}^{3} P(X = i) = P(X = 1) + P(X = 2) + P(X = 3) \\ &= \frac{1}{36} + \frac{2}{36} + \frac{3}{36} = \frac{1}{12} \end{aligned}$$

6.1.2 Variables aleatorias continuas

Este tipo de variables aleatorias tienen un rango no numerable; es decir, el conjunto de valores que puede tomar abarca un intervalo de números.

Un ejemplo podría ser la altura de una persona.

Si en las variables aleatorias discretas teníamos funciones masa de probabilidad, aquí definireoms funciones de densidad de probabilidad (o simplemente, funciones de densidad). La idea es la misma: nos permite conocer la probabilidad de que nuestra variable aleatoria tome un cierto valor del espacio muestral.

Es importante mencionar que, aunque la probabilidad de que la variable aleatoria tome un valor específico es 0, ya que nos encontramos en un conjunto no numerable, sí que podemos calcular la probabilidad de que se encuentre entre dos valores. Por tanto, si la función de densidad es f_X, entonces

$$P(a \leq X \leq b) = \int_a^b f_X(x)dx$$

La función de densidad tiene dos características importantes:

1. f_X es no negativa; esto es, $f_X(x) \geq 0 \; \forall x \in \Omega$
2. f_X integra uno en todo el espacio muestral:

$$\int_\Omega f_X(x) = 1$$

Esta última propiedad me gusta entenderla como si recoges todos los valores que puede tomar la variable aleatoria, la probabilidad de que te encuentres en el conjunto debe ser 1.. Si nos encontramos en el conjunto de números reales, podemos escribir la integral como $\int_{-\infty}^{\infty} f_X(x) = 1$.

Una de las variables aleatorias que más juego nos darán en el futuro será la v.a. con distribución uniforme en $[0, 1)$. La denotaremos como ξ, y escribiremos $\xi \sim U([0, 1))$. La probabilidad de que ξ tome un valor es constante, por lo que podemos definir su función de densidad como

$$f(\xi) = \begin{cases} 1 & \text{si } \xi \in [0,1) \\ 0 & \text{en otro caso.} \end{cases}$$

La probabilidad de ξ tome un valor entre dos elementos $a, b \in [0,1)$ es

$$P(\xi \in [a,b]) = \int_a^b 1dx = b - a$$

Como veremos más adelante, saber definir correctamente una función de densidad nos permitirá mejorar el rendimiento del path tracer.

La función de distribución $F_X(x)$ podemos definirla como:

$$F_X(x) = P(X \leq x) = \int_{-\infty}^{x} f_X(t)dt$$

Podemos concebir la función de distribución como ir acumulando los valores que pueda tomar la variable aleatoria: dado un cierto punto del espacio, x, ¿cuál sería la probabilidad de que cayéramos en algún valor que deja x por debajo?

El Teorema Fundamental del Cálculo nos permite relacionar función de distribución y función de densidad directamente:

$$f_X(x) = \frac{dF_X(x)}{dx}$$

6.1.3 Esperanza y varianza de una variable aleatoria

La esperanza de una variable aleatoria, denotada $E[X]$, es una generalización de la media ponderada. Nos informa del valor esperado de una variable aleatoria.

En el caso de las variables discretas, se define como

$$E[X] = \sum x_i p_i$$

donde x_i son los posibles valores que puede tomar la v.a., y p_i la probabilidad asociada a cada uno de ellos.

En el caso de las variables aleatorias continuas reales, la esperanza viene dada por

$$E[X] = \int_{-\infty}^{\infty} x f_X(x) dx$$

aunque, generalizando a una v.a. con espacio muestral Ω, la esperanza viene dada por

$$E[X] = \int_{\Omega} x f_X(x) dx$$

En el ejemplo de las variables discretas, su esperanza venía dada por

$$E[X] = \sum_{i=2}^{12} i \cdot p_i = 2\frac{1}{36} + 3\frac{2}{36} + \cdots + 12\frac{1}{36} = 7$$

Para variables aleatorias uniformes en (a, b) (es decir, $X \sim U(a, b)$), la esperanza es

$$E[X] = \int_a^b x \cdot \frac{1}{b-a} dx = \frac{a+b}{2}$$

La esperanza tiene unas cuantas propiedades que nos resultarán muy útiles. Estas son:

- Linealidad:
  - Si X, Y son dos v.a., $E[X + Y] = E[X] + E[Y]$
  - Si a es una constante, X una v.a., entonces $E[aX] = aE[X]$
  - Análogamente, para ciertas $X_1, \dots, X_k$, $E\left[\sum_{i=1}^{k} X_i\right] = \sum_{i=1}^{k} E[X_i]$
  - Estas propiedades no necesitan que las variables aleatorias sean independientes. ¡Este hecho será clave para las técnicas de Monte Carlo!.
- La Ley del estadístico insconciente (Law of the unconscious statistician, o LOTUS): dada una variable aleatoria X y una función medible g, la esperanza de $g(X)$ se puede calcular como

$$E[g(X)] = \int_{\Omega} g(x) f_X(x) dx$$

Esta propiedad será clave en nuestro desarrollo.

Será habitual encontrarnos con el problema de que no conocemos la distribución de una variable aleatoria Y. Sin embargo, si encontramos una transformación medible de una variable aleatoria X de forma que obtengamos Y (esto es, $\exists g$ función medible tal que $g(X) = Y$), entonces podemos calcular la esperanza de Y fácilmente. La importancia de la variable aleatoria con distribución uniforme ξ se debe a esta propiedad. Generar números aleatorios en $[0, 1)$ es muy fácil, así que obtendremos otras v.a.s a partir de ξ. Pero no adelantemos acontecimientos, así que no variemos tanto.

Hablando de variar: la varianza de una variable aleatoria nos permitirá medir cómo de dispersa es la distribución con respecto a su media. La denotaremos como $Var[X]$, y se define como

$$Var[X] = E\left[(X - E[X])^2\right]$$

Si desarrollamos esta definición, podemos conseguir una expresión algo más agradable:

$$\begin{aligned} Var[X] &= E\left[(X - E[X])^2\right] = \\ &= E\left[X^2 + E[X]^2 - 2XE[X]\right] = \\ &= E\left[X^2\right] + E[X]^2 - 2E[X]E[X] = \\ &= E\left[X^2\right] - E\left[X\right]^2 \end{aligned}$$

Hemos usado que $E[E[X]] = E[X]$ y la linealidad de la esperanza.

Enunciemos un par de propiedades que tiene, similares a la de la esperanza:

- La varianza saca constantes al cuadrado: $Var[aX] = a^2 Var[X]$
- $Var[X + Y] = Var[X] + Var[Y] + 2Cov[X, Y]$, donde $Cov[X, Y]$ es la covarianza de X y Y.

– En el caso en el que X e Y sean incorreladas (es decir, la covarianza es 0), $Var[X+Y] = Var[X] + Var[Y]$.

La varianza nos será útil a la hora de medir el error cometido por una estimación de Monte Carlo.

6.2 El estimador de Monte Carlo

Tras este breve repaso, estamos en condiciones de definir el estimador de Monte Carlo. Primero, vamos con su versión más sencilla.

Los estimadores de Monte Carlo nos permiten hallar la esperanza de una variable aleatoria, digamos, Y, sin necesidad de calcular explícitamente su valor. Para ello, tomamos unas cuantas muestras $Y_1, \dots, Y_N$ que sigan la misma distribución que Y con media μ, y consideramos el estimador de μ (Owen 2013):

$$\hat{\mu}_N = \frac{1}{N}\sum_{i=1}^{N} Y_i \tag{6.1}$$

Haciendo la esperanza de este estimador, vemos que

$$\begin{aligned} E[\hat{\mu}_N] = E\left[\frac{1}{N}\sum_{i=1}^{N} Y_i\right] &= \frac{1}{N} E\left[\sum_{i=1}^{N} Y_i\right] \\ &= \frac{1}{N}\sum_{i=1}^{N} E\left[Y_i\right] = \frac{1}{N}\sum_{i=1}^{N} \mu = \\ &= \mu \end{aligned}$$

A este tipo de estimadores se les llama insesgados.

Generalmente nos encontraremos en la situación en la que $Y = f(X)$, donde X sigue una distribución con función de densidad $p_X(x)$, y $f : S \to \mathbb{R}$. En ese caso, sabemos que la esperanza de Y se puede calcular como

$$\mu = E[Y] = E[f(X)] = \int_S f(x)p_X(x)dx$$

Lo que estamos buscando es calcular $\int_S f(x)dx$. Entonces, ¿qué ocurre si intentamos compensar [6.1] con la función de densidad?

$$\begin{aligned} E\left[\frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p_X(X_i)}\right] &= \frac{1}{N}\sum_{i=1}^{N}E\left[\frac{f(X_i)}{p_X(X_i)}\right] = \\ &= \frac{1}{N}\sum_{i=1}^{N}\left(\int_S \frac{f(x)}{p_X(x)}p_X(x)dx\right) = \\ &= \frac{1}{N}N\int_S f(x)dx = \\ &= \int_S f(x)dx \end{aligned}$$

¡Genial! Esto nos da una forma de calcular la integral de una función usando muestras de variables aleatorias con cierta distribución. Llamaremos al estimador de Monte Carlo

$$\hat{F_N} = \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p_X(X_i)}$$

Es importante mencionar que $p_X(x)$ debe ser distinto de 0 cuando f también lo sea.

Podemos particularizar el caso en el que nuestras muestras X_i sigan una distribución uniforme en $[a, b]$. Si eso ocurre, su función de densidad es $p_X(x) = \frac{1}{b-a}$, así que podemos simplificar un poco nuestro estimador:

$$\hat{F_N} = \frac{b-a}{N}\sum_{i=1}^{N}f(X_i)$$

Elegir correctamente la función de densidad p_X será clave. Si conseguimos elegirla debidamente, reduciremos mucho el error que genera el estimador. Esto es lo que se conoce como importance sampling.

Podemos calcular el error cuadrático medio de la estimación si volvemos al estimador de

la media [6.1], $\hat{\mu}_N$. Para ello, necesitamos la varianza: como $\hat{\mu}_N$ es insesgado, tenemos que

$$\begin{aligned} Var[\hat{\mu}_N] = Var\left[\frac{1}{N}\sum_{i=1}^{N} Y_i\right] = \frac{1}{N^2} Var\left[\sum_{i=1}^{N} Y_i\right] = \\ = \frac{1}{N^2}\sum_{i=1}^{N} Var[Y_i] = \frac{1}{N^2} N Var[Y] = \\ = \frac{Var[Y]}{N} \end{aligned}$$

El error cuadrático medio es

$$\sqrt{(Var[\hat{\mu}_N])} = \sqrt{\frac{Var[Y]}{N}} = \frac{\sqrt{Var[Y]}}{\sqrt{N}}$$

así que, como adelantamos al inicio del capítulo, la estimación tiene un error del orden $\mathcal{O}(N^{-1/2})$. Esto nos dice que, para reducir el error a la mitad, debemos tomar 4 veces más muestras.

- https://www.wikiwand.com/en/Rendering_equation
- https://www.wikiwand.com/es/Variable_aleatoria
- https://www.wikiwand.com/es/Distribuci%C3%B3n_de_probabilidad#/Distribuciones_de_varia
- https://www.wikiwand.com/es/Funci%C3%B3n_de_probabilidad
- https://www.pbr-book.org/3ed-2018/contents
- https://www.wikiwand.com/en/Expected_value
- https://www3.nd.edu/~dgalvin1/10120/10120_S16/Topic17_8p4_Galvin_class.pdf
- https://www.wikiwand.com/en/Probability_density_function
- RTT Shirley.
- https://artowen.su.domains/mc/
- https://www.wikiwand.com/es/Estimador

7 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)

7.1 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.

7.1.1 Github Actions

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

7.1.2 Github Projects

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

Anexo: Cheatsheet

Owen, Art B. 2013. Monte Carlo Theory, Methods and Examples.

Shirley, Peter, and R. Keith Morley. 2003. Realistic Ray Tracing. 2nd ed. USA: A. K. Peters, Ltd.