UNIVERSIDAD DE GRANADA

Los fundamentos de Ray Tracing

Doble grado en ingeniería informática y matemáticas

asmilex.github.io/Raytracing

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

April 11, 2022

Contents

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

Dedicatoria

¡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/Ray-tracing en Github para ver el estado del desarrollo.

Aun así, hay mucha gente que me ha ayudado a sacar este proyecto hacia delante.

Gracias, en primer lugar, a mi familia por permitirme acabar la carrera. A Cristina, Jorge, Jose OC, Lucas, Mari, Marina y Sergio por ayudarme con el contenido, feedback del desarrollo y guía de diseño.

1. 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.

1.1. ¿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**.

1.2. Vale, ¿y qué vamos a hacer entonces?

> TODO: WIP

Estos son los objetivos del trabajo:

- Path tracer inspirado en Shirley's *RT In One Weekend* series.
  - Muestreo directo de fuentes de luz.
  - Comparativa de rendimiento en tiempo real *vs* offline renderer.

- Métodos de Monte Carlo. Estudio del error producido por cada uno. Comprobar cómo de "buena" es cada solución.
- Futuro del área.

Referencias

(Wikipedia: history of photography 2022), (Wikipedia: Kodak 2022), (Wikipedia: Computer 2022), (Wikipedia: rendering (computer graphics) 2022), (Caulfield 2020), (tracing 2022), ("Rendering" n.d.), (Haines and Akenine-Möller 2019)

- RT IOW series

2. Las bases

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*.

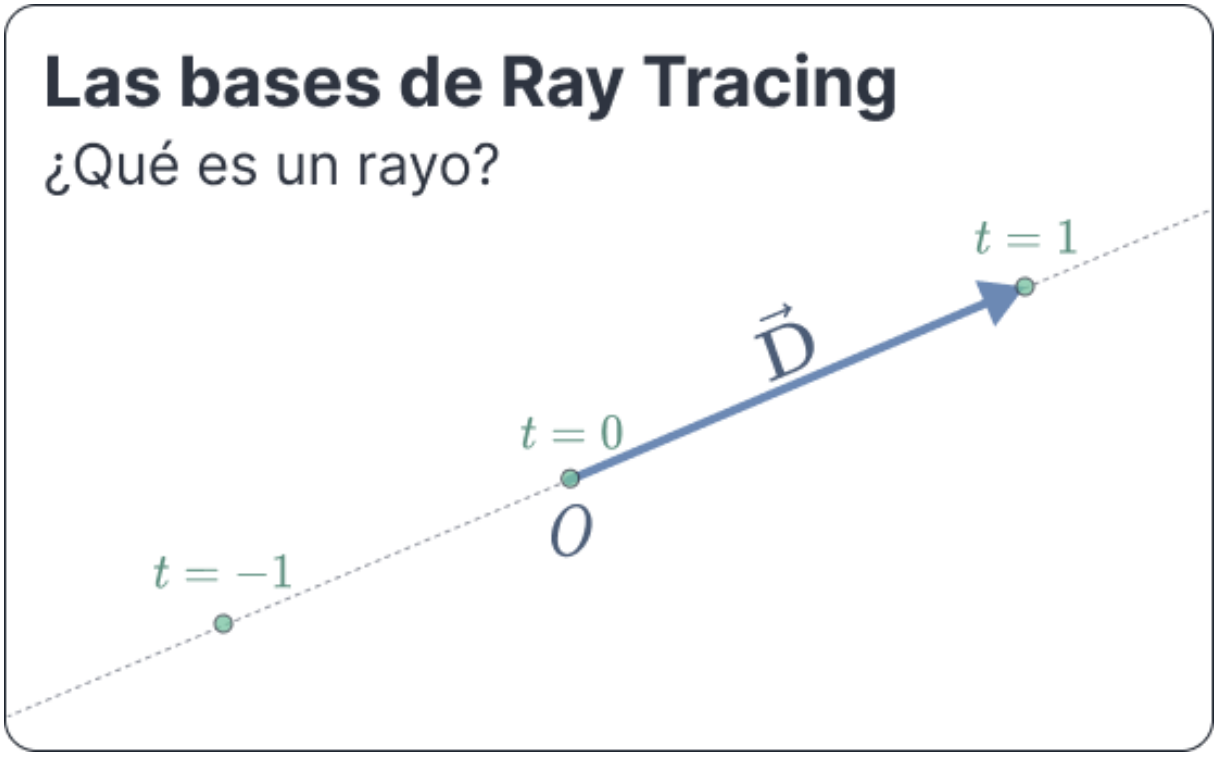

Figure 2.1.: El parámetro t nos permite controlar los puntos del rayo

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**.

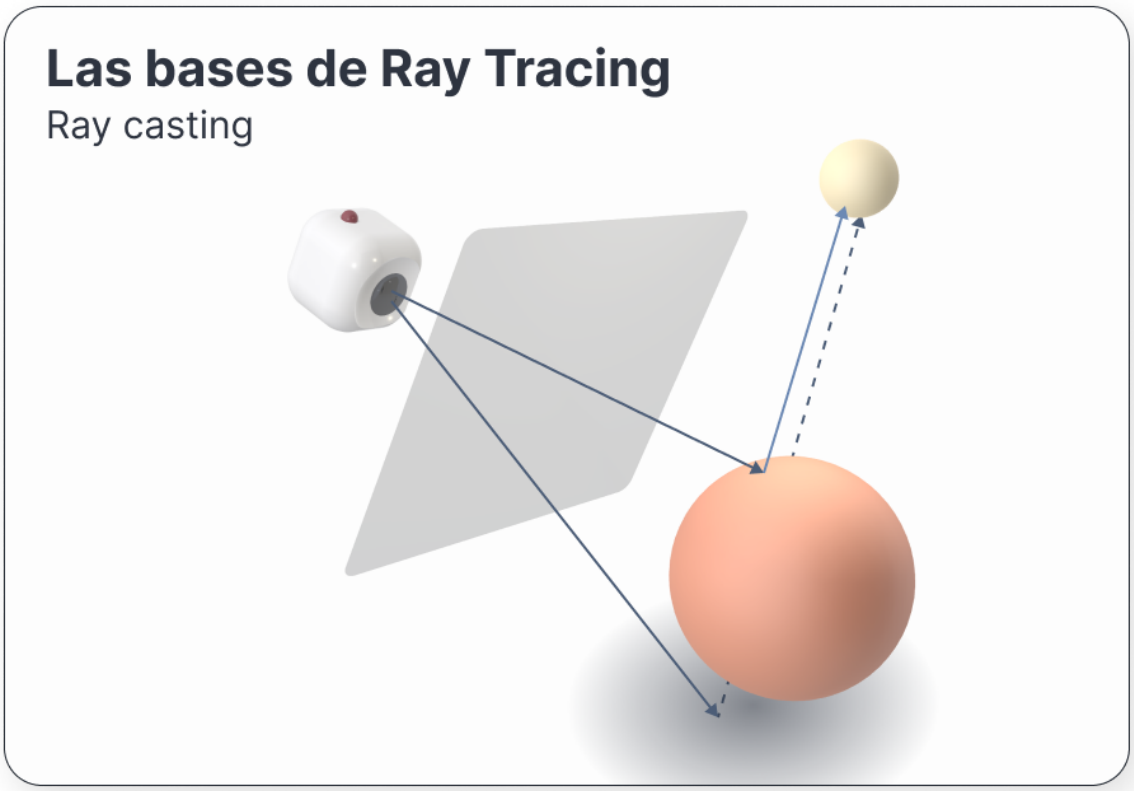

Figure 2.2.: Diagrama de ray casting

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.

2.1. 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.

2.2. Intersecciones rayo - objeto

Como dijimos al principio del capítulo, representaremos un rayo como

$$P(t) = (O_x, O_y, O_z) + t(D_x, D_y, D_z) = \\ = (O_x + tD_x, O_y + tD_y, O_y + tD_z)$$

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.

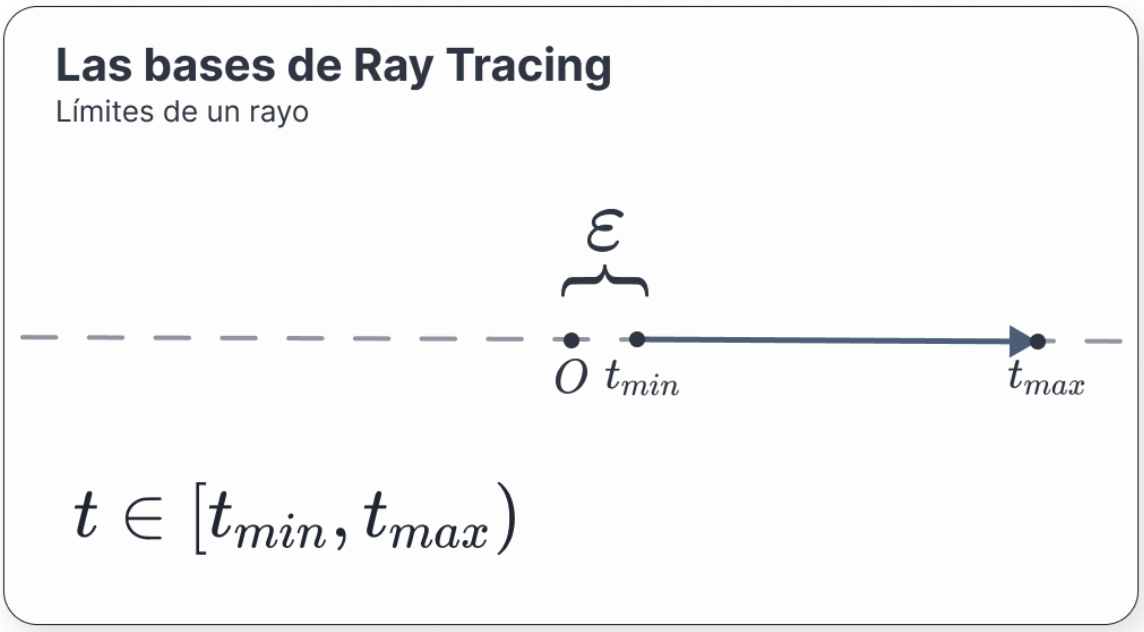

Figure 2.3.: Separarnos un poquito del origen evitará errores de coma flotante

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

2.2.1. Superficies implícitas

Generalmente, cuando hablemos de superficies, nos referiremos superficies diferenciables (Wikipedia: Differential geometry of surfaces 2022), 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 $\mathbf{n}$ a la superficie en ese punto como

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

> 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 $\mathbf{n}$.

La ecuación implícita del plano será

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

Si pinchamos nuestro rayo en la ecuación,

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

Resolviendo para t, esto se da si

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

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 $D \cdot \mathbf{n} = 0$. 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.

2.2.2. 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

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

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.

2.2.3. Intersecciones con esferas

Estudiemos ahora cómo intersecan una 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.

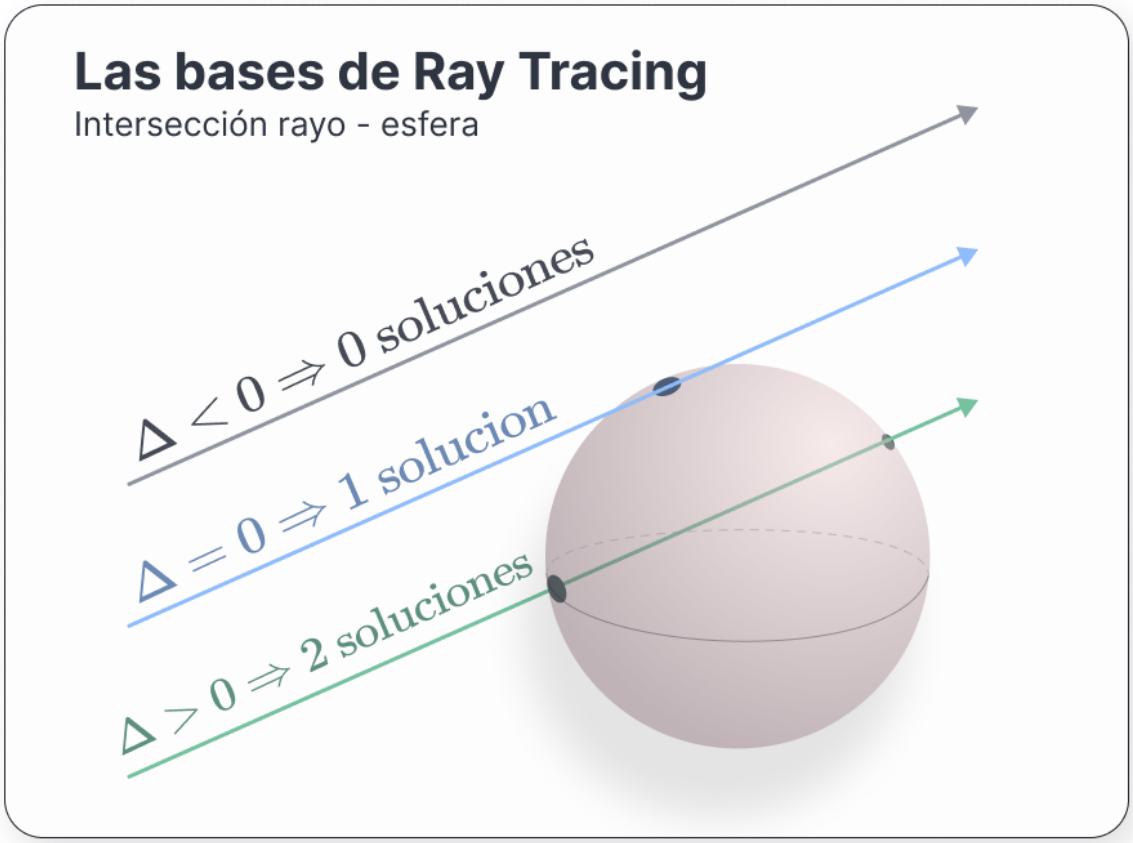

Figure 2.4.: Puntos de intersección con una esfera.

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

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

o, normalizando,

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

2.2.4. 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.

2.2.4.1. 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 solo 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)

2.2.4.2. 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

$$\begin{aligned} \beta + \gamma &< 1 \\ 0 &< \beta \\ 0 &< \gamma \end{aligned} \tag{2.1}$$

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 [2.1]. 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 (The Khronos® Vulkan Working Group 2022).

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

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

Referencias

(Wikipedia: Implicit surface 2022), (Wikipedia: Parametric surface 2021), (Wikipedia: Barycentric coordinate system 2022), (A. Romero Sarabia 2021), (Wikipedia: Differential geometry of surfaces 2022)

3. 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.

Una de las partes más importante de nuestro ray tracer es saber calcular la cantidad de luz en un punto de la escena. Para ello, necesitaríamos hallar la radianza en dicha posición mediante la *rendering equation*. Sin embargo, es *muy* difícil resolverla; tanto computacional 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 las integrales mediante una estimación. Utilizando muestreo aleatorio para evaluar puntos de una función, seremos capaces de obtener un resultado suficientemente bueno.

Una de las propiedades que hacen interesantes a este tipo de métodos es la **independencia del ratio de convergencia y la dimensionalidad del integrando**. Sin embargo, conseguir un mejor rendimiento tiene un precio a pagar. Dadas n muestras, la convergencia a la solución correcta tiene un orden de $\mathcal{O}\left(n^{-1/2}\right) = \mathcal{O}\left(\frac{1}{\sqrt{n}}\right)$. Es decir, para reducir el error a la mitad, necesitaríamos 4 veces más muestras.

En este capítulo veremos el fundamento de la integración de Monte Carlo, cómo muestrear distribuciones específicas y cómo afinar el resultado final.

3.1. 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 proceso de azar. Formalmente, 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 (conjunto de todas las posibilidades), $\mathcal{A}$ es una σ-álgebra de subconjuntos de Ω que refleja todas las posibilidades de eventos aleatorios, y 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 $R_X = \{x \in \mathbb{R} \,|\, \exists \omega \in \Omega : X(\omega) = x\}$:

3.1.1. 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

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

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 al lanzar los dados hemos obtenido $(1, 3)$, X tomará el valor 4. En total, X puede tomar todos los valores comprendidos entre 2 y 12. Este sería el **espacio muestral**. Cada pareja no está asociada a un único valor de X. 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 X tome un cierto valor x. Se denota por $P(X = x)$.

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

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

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

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

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

Muchas veces nos interesará conocer la probabilidad de que X se quede por debajo de un cierto valor x (de hecho, podemos caracterizar distribuciones aleatorias gracias a esto). Para ello, usamos la **función de distribución**:

$$F_X(x) = P(X \leq x) = \sum_{\substack{k \in \Omega \\ k \leq x}} P(X = k)$$

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

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

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

3.1.2. 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 definiremos las **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:

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

Intuitivamente, podemos ver esta última propiedad como *si acumulamos todos los valores que puede tomar la variable aleatoria, la probabilidad de que te encuentres en el conjunto debe*

ser 1. Si tratamos con un 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, definiendo correctamente una función de densidad conseguiremos mejorar el rendimiento del path tracer.

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

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

Es decir, dado un x, ¿cuál sería la probabilidad de que X se quede por debajo de x?

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

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

3.1.3. 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 dicha variable aleatoria.

En el caso de las variables discretas, se define como

$$E[X] = \sum_{x_i \in \Omega} x_i p_i$$

donde x_i son los posibles valores que puede tomar la v.a., y p_i la probabilidad asociada a cada uno de ellos; es decir, $p_i = P[X = x_i]$

Para una variable aleatoria continua real, 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 se puede generalizar como

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

Pongamos un par de ejemplos del cálculo de la esperanza. En el ejemplo de las variables discretas, la esperanza venía dada por

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

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. Esta propiedad hará que las variables aleatorias con distribución uniforme adquieran muchísima importancia. Generar números aleatorios en $[0, 1)$ es muy fácil, así que obtendremos otras v.a.s a partir de ξ.

Otra medida muy útil de una variable aleatoria es **la varianza**. 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.

3.2. El estimador de Monte Carlo

Tras este breve repaso de probabilidad, 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. Entonces, consideramos el estimador de μ (Owen 2013):

$$\hat{\mu}_N = \frac{1}{N}\sum_{i=1}^{N} Y_i \tag{3.1}$$

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 en [3.1] 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

$$\hat{F}_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p_X(X_i)} \tag{3.2}$$

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 [3.2]:

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

Elegir correctamente la función de densidad p_X será clave. Si conseguimos escogerla de-

bidamente, reduciremos mucho el error que genera el estimador. Esto es lo que se conoce como *importance sampling*.

> TODO: añadir enlace al capítulo de importance sampling.

Podemos calcular el error cuadrático medio de la estimación si volvemos al estimador de la media $\hat{\mu}_N$ [3.1]. 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.

Pongamos un ejemplo de estimador de Monte Carlo para una caja de dimensiones $[x_0, x_1] \times [y_0, y_1] \times [z_0, z_1]$. Si queremos estimar la integral de la función $f : \mathbb{R}^3 \to \mathbb{R}$

$$\int_{x_0}^{x_1}\int_{y_0}^{y_1}\int_{z_0}^{z_1} f(x, y, z)dxdydz$$

mediante una variable aleatoria $X \sim U([x_0, x_1] \times [y_0, y_1] \times [z_0, z_1])$ con función de densidad $p(x, y, z) = \frac{1}{x_1 - x_0}\frac{1}{y_1 - y_0}\frac{1}{z_1 - z_0}$, tomamos el estimador

$$\hat{F}_N = \frac{1}{(x_1 - x_0)\cdot(y_1 - y_0)\cdot(z_1 - z_0)}\sum_{i=1}^{N} f(X_i)$$

Otro ejemplo clásico de estimador de Monte Carlo es calcular el valor de π. Se puede hallar integrando una función que valga 1 en el interior de la circunferencia de radio unidad y 0 en el exterior:

$$f = \begin{cases} 1 & \text{si } x^2 + y^2 \leq 1 \\ 0 & \text{en otro caso} \end{cases} \implies \pi = \int_{-1}^{1} \int_{-1}^{1} f(x, y)\, dxdy$$

Para usar el estimador de [3.2], necesitamos saber la probabilidad de obtener un punto dentro de la circunferencia.

Bien, consideremos que una circunferencia de radio r se encuentra inscrita en un cuadrado. El área de la circunferencia es πr^2, mientras que la del cuadrado es $(2r)^2 = 4r^2$. Por tanto, la probabilida de obtener un punto dentro de la circunferencia es $\frac{\pi r^2}{4r^2} = \frac{\pi}{4}$. Podemos tomar $p(x, y) = \frac{1}{4}$, de forma que

$$\pi \approx \frac{4}{N} \sum_{i=1}^{N} f(x_i, y_i), \text{ con } (x_i, y_i) \sim U([-1, 1] \times [-1, 1])$$

3.3. Escogiendo puntos aleatorios

Una de las partes clave del estimador de Monte Carlo [3.2] es saber escoger la función de densidad p_X correctamente. En esta sección, veremos algunos métodos para conseguir distribuciones específicas partiendo de funciones de densidad sencillas.

3.3.1. Método de la transformada inversa

> **En resumen**: Para conseguir una muestra de una distribución específica F_X:
>
> 1. Generar un número aleatorio $\xi \sim U(0, 1)$.
> 2. Hallar la inversa de la función de distribución deseada F_X, denotada $F_X^{-1}(x)$.
> 3. Calcular $F_X^{-1}(\xi) = X$.

Este método nos permite conseguir muestras de cualquier distribución continua a partir de variables aleatorias uniformes, siempre que se conozca la inversa de la función de

distribución.

Sea X una variable aleatoria con función de distribución F_X[1]. Queremos buscar una transformación $T : [0, 1] \to \mathbb{R}$ tal que $T(\xi) \overset{\mathrm{d}}{=} X$, siendo ξ una v.a. uniformemente distribuida. Para que esto se cumpla, se debe dar

$$\begin{aligned} F_X(x) &= P[X < x] = \\ &= P[T(\xi) < x] = \\ &= P(\xi < T^{-1}(x)) = \\ &= T^{-1}(x) \end{aligned}$$

Este último paso se debe a que, como ξ es uniforme en $(0, 1)$, $P[\xi < x] = x$. Es decir, hemos obtenido que F_X es la inversa de T.

> TODO: dibujo similar a este: p.52

Como ejemplo, vamos a muestrear la función $f(x) = x^2,\ x \in [0, 2]$.

Primero, normalizamos esta función para obtener una función de densidad $p_X(x)$. Es decir, buscamos $p_X(x) = cf(x)$ tal que

$$\begin{aligned} 1 &= \int_0^2 p_X(x)dx = \int_0^2 cf(x)dx = c\int_0^2 f(x)dx = \\ &= \frac{cx^3}{3}\Big|_2^3 = \frac{8c}{3} \\ &\Rightarrow c = \frac{3}{8} \\ &\Rightarrow p_X(x) = \frac{3x^2}{8} \end{aligned}$$

A continuación, integramos la función de densidad para obtener la de distribución F_X:

$$F_X(x) = \int_0^x p_X(x)dx = \int_0^x \frac{3x^2}{8} = \frac{x^3}{8}$$

[1] En su defecto, si tenemos una función de densidad f_X, podemos hallar la función de distribución haciendo $F_X(x) = P[X < x] = \int_{x_{min}}^{x} f_X(t)dt$

Solo nos queda conseguir la muestra. Para ello,

$$\xi = F_X(x) = \frac{x^3}{8} \iff$$
$$x = \sqrt[3]{8\xi}$$

Sacando un número aleatorio ξ, y pasándolo por la función obtenida, conseguimos un elemento con distribución $f(x)$.

3.3.2. Método del rechazo

> **En resumen**: Para conseguir una muestra de una variable aleatoria X con función de densidad p_X:
>
> 1. Obtener una muestra y de Y , y otra ξ de $U(0,1)$.
> 2. Comprobar si $\xi < \frac{p_X(y)}{Mp_Y(y)}$. Si es así, aceptarla. Si no, sacar otra muestra.

El método anterior presenta principalmente dos problemas:

1. No siempre es posible integrar una función para hallar su función de densidad.
2. La inversa de la función de distribución, F_X^{-1} no tiene por qué existir.

Como alternativa, podemos usar este método (en inglés, *rejection method*). Para ello, necesitamos una variable aleatoria Y con función de densidad $p_Y(y)$. El objetivo es conseguir una muestra de X con función de densidad $p_X(x)$.

La idea principal es aceptar una muestra de Y con probabilidad p_X/Mp_Y, con $1 < M < \infty$. En esencia, estamos jugando a los dardos: si la muestra de y que hemos obtenido se queda por debajo de la gráfica de la función $Mp_Y < p_X$, estaremos obteniendo una de p_X.

> TODO dibujo de la gráfica $\frac{p_X(y)}{Mp_Y(y)}$.
>
> ¿Quizás haga falta una demostración también? No estoy satisfecho con este apartado ahora mismo. Necesita trabajo.

El algoritmo consiste en:

1. Obtener una muestra de Y, denotada y, y otra de $U(0,1)$, llamada ξ.
2. Comprobar si $\xi < \frac{p_X(y)}{Mp_Y(y)}$.
    1. Si se cumple, se acepta y como muestra de p_X
    2. En caso contrario, se rechaza y y se vuelve al paso 1.

3.4. *Importance sampling*

Con la llegada de ray tracing en tiempo real surge una obligación por optimizar los pocos rayos que se pueden trazar. Una de las preguntas que nos debemos hacer es *hacia dónde* generamos el rayo.

En esta sección daremos respuesta a este dilema. Estudiaremos cómo las fuentes de luz afectan a la calidad de la imagen final. Veremos técnicas de reducción del error, las cuales nos permitirán acelerar enormemente el cómputo de la escena.

Referencias

(Wikipedia: Rendering equation 2021), (Wikipedia: Variable aleatoria 2021), (Wikipedia: Distribución de probabilidad 2022), (Wikipedia: Función de probabilidad 2022), (Wikipedia: Expected value 2022), (Galvin n.d.), (Wikipedia: Probability density function 2022), (Wikipedia: Estimador 2022), (Wikipedia: Método de la transformada inversa 2022), (Wikipedia: Rejection sampling 2022), (Shirley and Morley 2003), (Pharr, Jakob, and Humphreys 2016), (Owen 2013), (Berkeley cs184 2022, Monte Carlo Integration)

- *(berkeley-cs184)* https://cs184.eecs.berkeley.edu/public/sp22/lectures/lec-12-monte-carlo-integration/lec-12-monte-carlo-integration.pdf
- Gems I, p.284.

4. Transporte de luz

En este capítulo estudiaremos las bases de la radiometría. Esta área de la óptica nos proporcionará una serie de herramientas con las cuales podremos responder a la pregunta *cuánta luz existe en un punto*.

4.1. Unidades radiométricas básicas

> **Nota**: cuando usemos un paréntesis tras una ecuación, dentro denotaremos sus unidades de medida.

Antes de comenzar a trabajar, necesitamos conocer *qué entendemos* por luz. Aunque hay muchas formas de trabajar con ella (a fin de cuentas, todavía seguimos discutiendo sobre *qué es* exactamente la luz [1]), nosotros nos quedaremos con algunas pinceladas de la cuántica. Nos será suficiente quedarnos con la concepción de fotón. Una fuente de iluminación emite una serie de fotones. Estos fotones tienen(Shirley and Morley 2003) una posición, una dirección de propagación y una longitud de onda λ. Un fotón también tiene asociado una velocidad c que depende del índice de refracción del medio, n.

La unidad de medida de λ es el nanómetro (nm). También nos vendrá bien definir una frecuencia, f. Su utilidad viene del hecho de que, cuando la luz cambia de medio al propagarse, la frecuencia se mantiene constante.

$$f = \frac{c}{\lambda}$$

[1] No entraremos en detalle sobre la naturaleza de la luz. Sin embargo, si te pica la curiosidad, hay muchos divulgadores como (QuantumFracture 2021) que han tratado el tema con suficiente profundidad.

Un fotón tiene asociada una carga de energía, denotada por Q:

$$Q = hf = \frac{hc}{\lambda}(\text{J})$$

donde $h = 6.62607004 \times 10^{-34}\text{J} \cdot \text{s}$ es la constante de Plank y $c = 299792458\text{m/s}$ la velocidad de la luz.

En realidad, **todas estas cantidades deberían tener un subíndice** λ, puesto que dependen de la longitud de onda. La energía de un fotón Q, por ejemplo, debería denotarse Q_λ. Sin embargo, en la literatura de informática gráfica, **se ha optado por omitirla**. ¡Tenlo en cuenta a partir de aquí!

4.1.1. Potencia

A partir de la energía anterior, podemos estimar *la tasa de producción de energía*. A esta tasa la llamaremos(Pharr, Jakob, and Humphreys 2016) **potencia**, o **flujo radiante** Φ. Esta medida nos resultará más útil que la energía total, puesto que nos permite estimar la energía en un instante:

$$\Phi = \lim_{\Delta t \to 0} \frac{\Delta Q}{\Delta t} = \frac{dQ}{dt}(J/s)$$

Su unidad es julios por segundo, comúnmente denotado vatio (*watts*, W). También se utiliza el lumen. Podemos encontrar la energía total en un periodo de tiempo $[t_0, t_1]$ integrando el flujo radiante:

$$Q = \int_{t_0}^{t_1} \Phi(t)dt$$

4.1.2. Irradiancia

La **irradiancia** o **radiancia emitida** es el flujo radiante que recibe una superficie. Dada un área A, se define como

$$E = \frac{\Phi}{A}(\mathrm{W/m}^2)$$

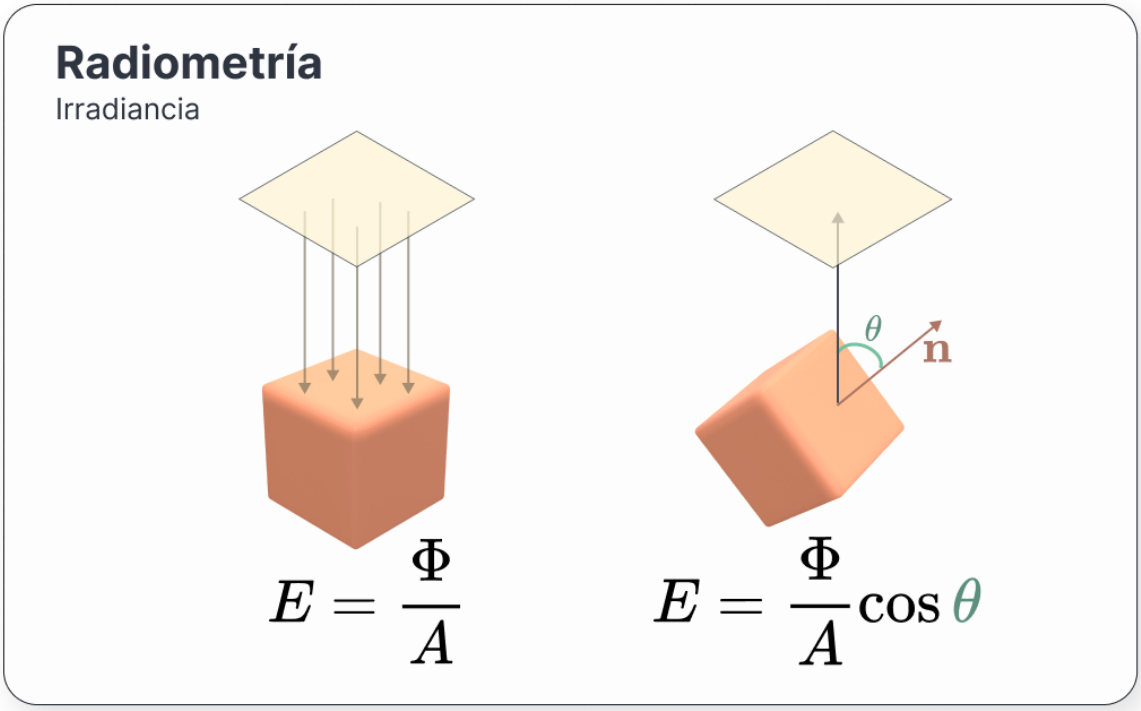

Figure 4.1.: La irradiancia es la potencia por metro cuadrado incidente en una superficie. Es proporcional al coseno del ángulo entre la dirección de la luz y la normal a la superficie.

Ahora que tenemos la potencia emitida en una cierta área, nos surge una pregunta: *¿y en un cierto punto p?*. Tomando límites en la expresión anterior, encontramos la respuesta:

$$E(p) = \lim_{\Delta A \to 0} \frac{\Delta \Phi}{\Delta A} = \frac{d\Phi}{dA}(\mathrm{W/m}^2)$$

De la misma manera que con la potencia, integrando $E(p)$ podemos obtener el flujo radi-

ante:

$$\Phi = \int_A E(p)dp$$

El principal problema de la irradiancia es que *no nos dice nada sobre las direcciones* desde las que ha llegado la luz.

4.1.3. Ángulos sólidos

Con estas tres unidades básicas, nos surge una pregunta muy natural: *¿cómo mido cuánta luz llega a una superficie?*

Para responder a esta pregunta, necesitaremos los **ángulos sólidos**. Son la extensión de los **ángulos planares**, en dos dimensiones.

Ilustremos el sentido de estos ángulos: imaginemos que tenemos un cierto objeto en dos dimensiones delante de nosotros, a una distancia desconocida. ¿Sabríamos cuál es su tamaño, solo con esta información? Es más, si entrara otro objeto en la escena, ¿podríamos distinguir cuál de ellos es más grande?

Parece difícil responder a estas preguntas. Sin embargo, sí que podemos determinar *cómo de grandes nos parecen* desde nuestro punto de vista. Para ello, describimos una circunferencia de radio r alrededor nuestra. Si trazamos un par de líneas desde nuestra posición a las partes más alejadas de este objeto, y las cortamos con nuestra circunferencia, obtendremos un par de puntos inscritos en ella. Pues bien, al arco que encapsulan dichos puntos le vamos a hacer corresponder un cierto ángulo: el ángulo planar.

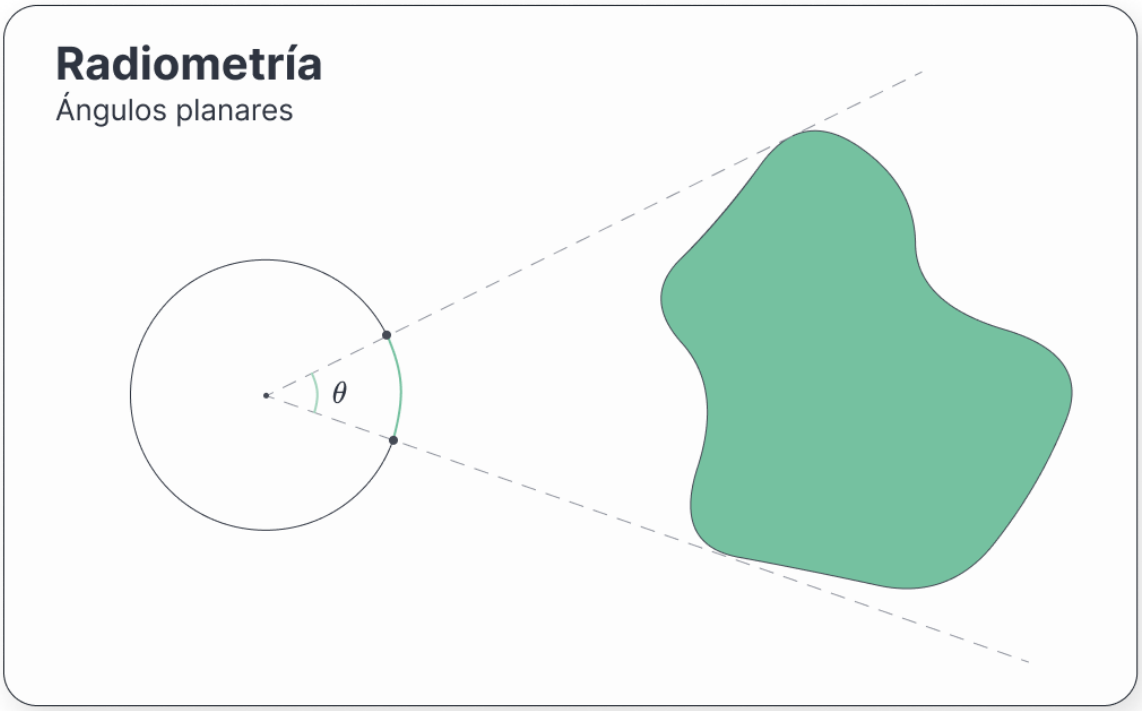

Figure 4.2.: La idea intuitiva de un ángulo planar

Llevando esta idea a las tres dimensiones es como conseguimos el concepto de **ángulo sólido**. Si en dos dimensiones teníamos una circunferencia, aquí tendremos una esfera. Cuando generemos las rectas proyectantes hacia el volumen, a diferencia de los ángulos planares, se inscribirá un área en la esfera. La razón entre dicha área A y el cuadrado del radio r nos dará un ángulo sólido:

$$\omega = \frac{A}{r^2}(\text{sr})$$

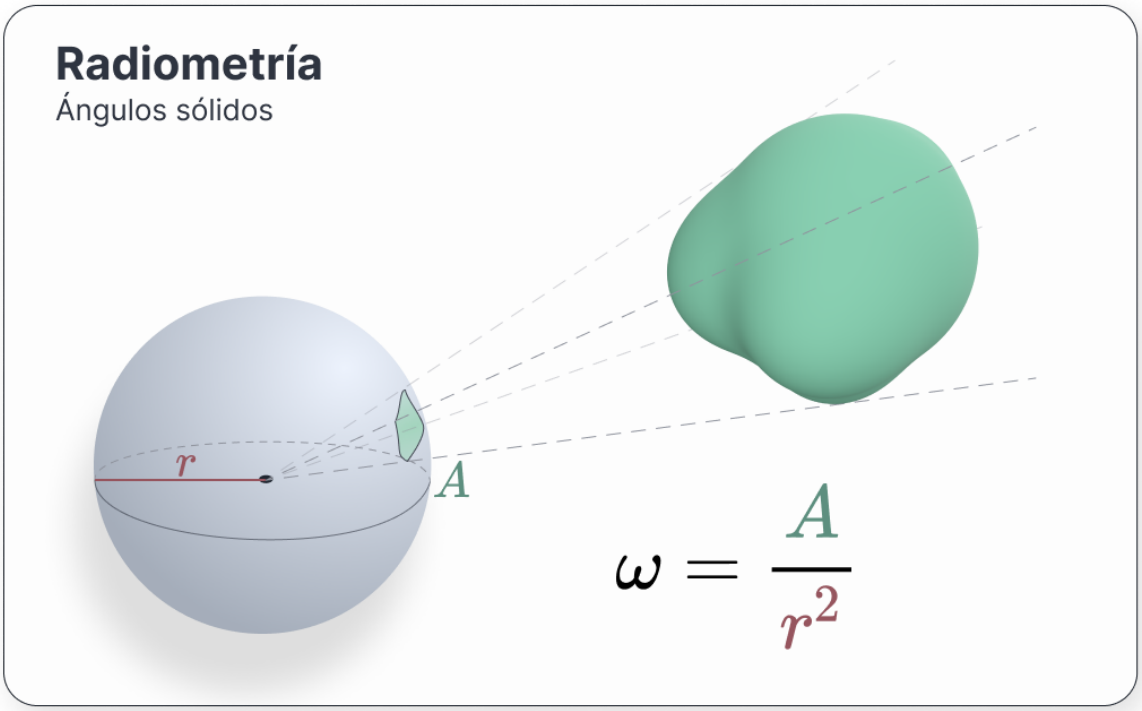

Figure 4.3.: Un ángulo sólido es la razón entre el área proyectada y el cuadrado del radio

Los denotaremos por ω. En física se suele usar Ω, pero aquí optaremos por la minúscula. Su unidad de medida es el estereorradián (sr). Se tiene que $\omega \in [0, 4\pi]$. Si 2π radianes corresponden a la circunferencia completa, para la esfera se tiene que 4π esteorradianes cubren toda la superficie de esta. Se tiene también que 2πsr cubren un hemisferio. Además, un esteorradián corresponde a una superficie con área r^2: $1\text{sr} = \frac{r^2}{r^2}$.

De vez en cuando, usaremos ω **un vector dirección unitario en la esfera**.

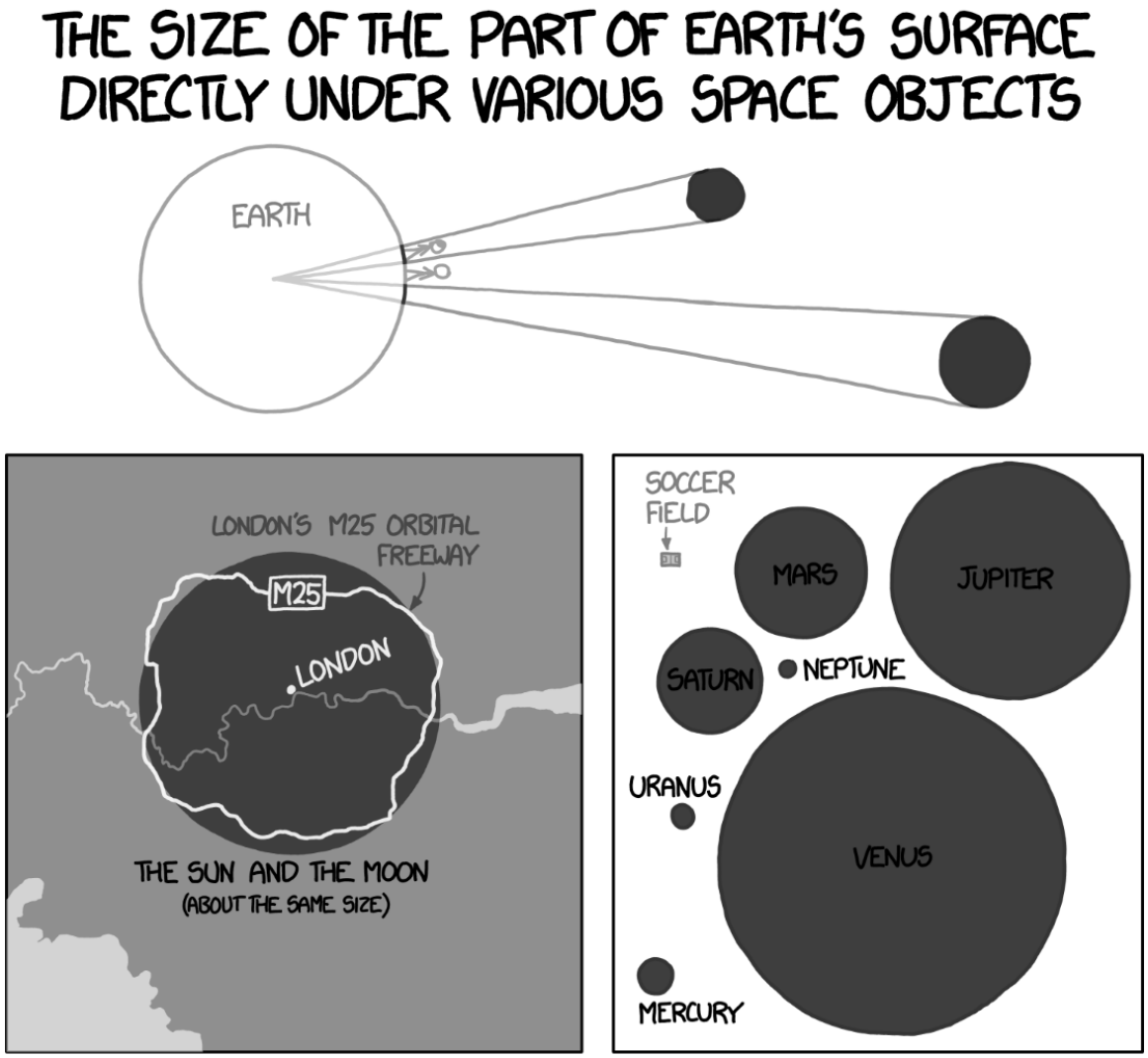

Figure 4.4.: Como de costumbre, hay un xkcd relevante. (Fuente)

Usualmente emplearemos coordenadas esféricas cuando trabajemos con ellos, dado que resulta más cómodo.

$$\begin{cases} x = \sin\theta\cos\theta \\ y = \sin\theta\sin\theta \\ z = \cos\theta \end{cases}$$

A θ se le denomina ángulo polar, mientras que a ϕ se le llama acimut. Imaginémonos un punto en la esfera de radio r ubicado en una posición (r, θ, ϕ). Queremos calcular un área chiquitita dA_h, de forma que el ángulo sólido asociado a dicha área debe ser $d\omega$. Así, $d\omega = \frac{dA_h}{r^2}$. Si proyectamos el área, obtenemos $d\theta$ y $d\phi$: pequeños cambios en los ángulos que nos generan nuestra pequeña área.

dA_h debe tener dos lados $lado_1$ y $lado_2$. Podemos hallar $lado_1$ si lo trasladamos al eje z de nuevo. Así, $lado_1 = r \sin d\theta$. De la misma manera, $lado_2 = rd\theta$.

> TODO: foto que explique todo esto, porque si no, no hay quien se entere. Quizás me sirva la de https://cs184.eecs.berkeley.edu/public/sp22/lectures/lec-11-radiometry-and-photometry/lec-11-radiometry-and-photometry.pdf, p.16 siempre que adapte ϕ.

Poniendo estos valores en $d\omega$:

$$\begin{aligned} d\omega = \frac{dA_h}{r^2} = \frac{lado_1 lado_2}{r^2} = \\ = \frac{r \sin\theta \, d\phi \, r \, d\theta}{r^2} = \\ = \sin\theta \, d\theta \, d\phi \end{aligned} \tag{4.1}$$

¡Genial! Acabamos de añadir un recurso muy potente a nuestro inventario. Esta expresión nos permitirá convertir integrales sobre ángulos sólidos en integrales sobre ángulos esféricos.

4.1.4. Intensidad radiante

Los ángulos sólidos nos proporcionan una variedad de herramientas nuevas considerable. Gracias a ellos, podemos desarrollar algunos conceptos nuevos. Uno de ellos es la **intensidad radiante**.

Imaginémonos un pequeñito punto de luz encerrado en una esfera, el cual emite fotones en todas direcciones. Nos gustaría medir cuánta energía pasa por la esfera. Podríamos entonces definir

$$I = \frac{\Phi}{4\pi} (\text{W/sr})$$

Otra unidad de medida es el lumen por esterorradián, (lm/sr). La anterior definición mide cuántos fotones pasan por toda la esfera. ¿Qué ocurre si *cerramos* el ángulo, restringiéndonos así a un área muy pequeña de la esfera?

$$I = \lim_{\Delta\omega \to 0} \frac{\Delta\Phi}{\Delta\omega} = \frac{d\Phi}{d\omega}$$

De la misma manera que con los conceptos anteriores, podemos volver a la potencia integrando sobre un conjunto de direcciones:

$$\Phi = \int_{\Omega} I(\omega) d\omega$$

4.1.5. Radiancia

Finalmente, llegamos al concepto más importante. La **radiancia espectral** (o radiancia a secas[2]) es una extensión de la radiancia emitida teniendo en cuenta la dirección:

$$L(p, \omega) = \lim_{\Delta\omega \to 0} \frac{\Delta E_{\omega}(p)}{\Delta\omega} = \frac{dE_{\omega}(p)}{d\omega}$$

siendo $E_{\omega}(p)$ la radiancia emitida a la superficie perpendicular a ω.

> TODO: foto como la de https://cs184.eecs.berkeley.edu/public/sp22/lectures/lec-11-radiometry-and-photometry/lec-11-radiometry-and-photometry.pdf, página 10.

Podemos dar otra expresión de la radiancia en términos del flujo:

$$L(p, \omega) = \frac{d^2\Phi(p, \omega)}{d\omega \, dA^{\perp}} = \frac{d^2\Phi(p, \omega)}{d\omega \, dA \, \cos\theta} \tag{4.2}$$

donde $dA^{\perp}$ es el área proyectada por dA en una hipotética superficie perpendicular a ω:

[2] Recuerda que estamos omitiendo la longitud de onda λ.

> TODO: figura similar a pbr figura 5.10 https://www.pbr-book.org/3ed-2018/Color_and_Radiometry/I

Cuando un rayo impacta en una superficie, L puede tomar valores muy diferentes en un lado y otro de dicha superficie. Por ejemplo, si nos imaginamos un espejo, el valor un poco por encima y un poco por debajo de un punto del espejo es muy diferente. Para solucionarlo, podemos tomar límites para distinguir a ambos lados:

$$\begin{aligned} L^+(p,\omega) &= \lim_{t\to 0^+} L(p + t\mathbf{n_p}, \omega) \\ L^-(p,\omega) &= \lim_{t\to 0^-} L(p + t\mathbf{n_p}, \omega) \end{aligned} \tag{4.3}$$

donde $\mathbf{n_p}$ es la normal en el punto p.

Otra forma de solucionarlo (y preferible, puesto que simplifica entender lo que ocurre) es distinguir entre la radiancia que llega a un punto –la incidente–, y la saliente.

La primera se llamará $L_i(p,\omega)$, mientras que la segunda será $L_o(p,\omega)$. Es importante destacar que ω apunta *hacia fuera* de la superficie. Quizás es contraintuitivo en L_i, puesto que $-\omega$ apunta *hacia* la superficie. Depende del autor se utiliza una concepción u otra.

> **Nota**(ción): a L_o también se le conoce como la radiancia reflejada. Por eso, algunas veces aparece como L_r en algunas fuentes.

Utilizando esta notación y usando [4.3], podemos escribir L_i y L_o como

$$L_i(p,\omega) = \begin{cases} L^+(p,-\omega) & \text{si } \omega \cdot \mathbf{n_p} > 0 \\ L^-(p,-\omega) & \text{si } \omega \cdot \mathbf{n_p} < 0 \end{cases}$$

$$L_o(p,\omega) = \begin{cases} L^+(p,\omega) & \text{si } \omega \cdot \mathbf{n_p} > 0 \\ L^-(p,\omega) & \text{si } \omega \cdot \mathbf{n_p} < 0 \end{cases}$$

Hacemos esta distinción porque, a fin de cuentas, necesitamos distinguir entre los fotones que llegan a la superficie y los que salen.

> TODO: https://cs184.eecs.berkeley.edu/public/sp22/lectures/lec-11-radiometry-and-photometry/lec-11-radiometry-and-photometry.pdf, p.36

Una propiedad a tener en cuenta es que, si cogemos un punto p del espacio donde no existe ninguna superifcie, $L_o(p, \omega) = L_i(p, -\omega) = L(p, \omega)$

La importancia de la radiancia se debe a un par de propiedades:

La primera de ellas es que, dado L, podemos calcular cualquier otra unidad básica mediante integración. Además, **su valor se mantiene constante en rayos que viajan en el vacío en línea recta** (Fabio Pellacini 2017). Esto último hace que resulte muy natural usarla en un ray tracer.

Veamos por qué ocurre esto:

> TODO: https://pellacini.di.uniroma1.it/teaching/graphics17b/lectures/12_pathtracing.pdf, página 18.

Consideremos dos superficies ortogonales entre sí, S_1 y S_2 separadas una distancia r. Debido a la conservación de la energía, cualquier fotón que salga de una superficie y se encuentre bajo el ángulo sólido de la otra debe llegar impactar en dicha superficie opuesta.

Por tanto:

$$d^2\Phi_1 = d^2\Phi_2$$

Sustituyendo en la expresión de la radiancia [4.2], y teniendo en cuenta que son ortogonales (lo que nos dice que $\cos\theta = 1$):

$$L_1 d\omega_1 dA_1 = L_2 d\omega_2 dA_2$$

Por construcción, podemos cambiar los ángulos sólidos:

$$L_1 \frac{dA_2}{r^2} dA_1 = L_2 \frac{dA_1}{r^2} dA_2$$

Lo que finalmente nos dice que $L_1 = L_2$, como queríamos ver.

4.2. Integrales radiométricas

En esta sección, vamos a explorar las nuevas herramientas que nos proporciona la radiancia. Veremos también cómo integrar ángulos sólidos, y cómo simplificar dichas integrales.

4.2.1. Una nueva expresión de la irradiancia y el flujo

Como dijimos al final de la sección de la irradiancia, esta medida no tiene en cuenta las direcciones desde las que llegaba la luz. A diferencia de esta, la radiancia sí que las utiliza. Dado que una de las ventajas de la radiancia es que nos permite obtener el resto de medidas radiométricas, ¿por qué no desarrollamos una nueva expresión de la irradiancia?

Para obtener cuánta luz llega a un punto, debemos acumular la radiancia incidente que nos llega desde cualquier dirección.

> TODO: dibujo como el de la libreta roja. Me lo mandé por Telegram, por si no lo encuentro

Dado un punto p que se encuentra en una superficie con normal $\mathbf{n}$ en dicho punto, la irradiancia se puede expresar como

$$E(p, \mathbf{n}) = \int_{\Omega} L_i(p, \omega) |cos\theta| d\omega \tag{4.4}$$

El término cos θ aparece en la integral debido a la derivada del área proyectada, $dA^{\perp}$. θ es el ángulo entre la dirección ω y la normal $\mathbf{n}$.

Generalmente, la irradiancia se calcula únicamente en el hemisferio de direcciones asociado a la normal en el punto, $H^2(\mathbf{n})$.

Podemos eliminar el $\cos\theta$ de la integral mediante una pequeña transformación: proyectando el ángulo sólido sobre el disco alrededor del punto p con normal $\mathbf{n}$, obtenemos una expresión más sencilla: como $d\omega^{\perp} = |\cos\theta| d\omega$, entonces

$$E(p, \mathbf{n}) = \int_{H^2(\mathbf{n})} L_i(p, \omega) d\omega^{\perp}$$

Usando lo que aprendimos sobre la derivada de los ángulos sólidos [4.1], se puede reescribir la ecuación anterior como

$$E(p, \mathbf{n}) = \int_0^{2\pi} \int_0^{\pi/2} L_i(p, \theta, \phi) \cos\theta \ \sin\theta \, d\theta \, d\phi$$

Haciendo el mismo juego con el flujo emitido de un cierto objeto al hemisferio que encapsula la normal, conseguimos:

$$\begin{aligned} \Phi &= \int_A \int_{H^2(\mathbf{n})} L_o(p, \omega) \cos\theta \, d\omega dA = \\ &= \int_A \int_{H^2(\mathbf{n})} L_o(p, \omega) d\omega^{\perp} dA \end{aligned}$$

> TODO: a lo mejor merece la pena hacer un ejemplo sobre los diferentes tipos de luz, como en https://cs184.eecs.berkeley.edu/public/sp22/lectures/lec-11-radiometry-and-photometry/lec-11-radiometry-and-photometry.pdf p.41? O a lo mejor un capítulo para hablar de luces en general.

4.2.2. Integrando sobre área

Una herramienta más que nos vendrá bien será la capacidad de convertir integrales sobre direcciones en integrales sobre área. Hemos hecho algo similar en las secciones anteriores, así que no perdemos nada por generalizarlo.

Considera un punto p sobre una superficie con normal en dicho punto $\mathbf{n}$. Supongamos que tenemos una pequeña área dA con normal $\mathbf{n_{dA}}$. Sea θ el ángulo entre $\mathbf{n}$ y $\mathbf{n_{dA}}$, y r la distancia entre p y dA.

Entonces, la relación entre la diferencial de un ángulo sólido y la de un área es

$$d\omega = \frac{dA \cos\theta}{r^2}$$

> TODO: figura como la de pbr book 5.16.

Esto nos permite, por ejemplo, expandir algunas expresiones como la de la irradiancia [4.4] si partimos de un cuadrilátero dA:

$$E(p, \mathbf{n}) = \int_{\Omega} L_i(p, \omega)|\cos\theta|d\omega = \\ = \int_{A} L\cos\theta\,\frac{\cos\theta_o}{r^2}dA$$

siendo θ_o el ángulo de la radiancia de salida de la superficie del cuadrilátero.

4.3. Dispersión de luz: las familias de funciones de distribución bidireccionales

Cuando una fuente de luz emite fotones hacia una superficie impactando en ella, ocurren un par de sucesos: parte de la luz se refleja en ella, saliendo disparada hacia alguna dirección; mientras que otra parte se absorbe.

En informática gráfica se consideran tres tipos principales de dispersión de luz: **dispersión en superficie** (*surface scattering*), **dispersión volumétrica** (*volumetric scattering*) y **dispersión bajo superficie** (*subsurface scattering*)

En este capítulo vamos a modelar la primera. Estudiaremos qué es lo que ocurre cuando los fotones alcanzan una superficie, en qué dirección se reflejan, y cómo cambia el comportamiento dependiendo de las propiedades del material.

4.3.1. La función de distribución de reflectancia bidireccional (BRDF)

La **función de distribución de reflectancia bidireccional** (en inglés, *bidirectional reflectance distribution function*, BRDF) describe cómo la luz se refleja en una superficie opaca. Se encarga de informarnos sobre cuánta radiancia sale en dirección ω_o debido a la

radiancia incidente desde la dirección ω_i, partiendo de un punto p en una superficie con normal **n**. Depende de la longitud de onda λ, pero, como de costumbre, la omitiremos.

> **Intuición**: *¿cuál es la probabilidad de que, habiéndome llegado un fotón desde ω_i, me salga disparado hacia ω_o?*

> TODO: esquema como el de pbr fig. 5.18, o como https://pellacini.di.uniroma1.it/teaching/graphics17b p.20

Si consideramos ω_i como un cono diferencial de direcciones, la irradiancia diferencial en p viene dada por

$$dE(p, \omega_i) = L_i(p, \omega_i) \cos\theta_i \, d\omega_i$$

Debido a esta irradiancia, una pequeña parte de radiancia saldrá en dirección ω_o, proporcional a la irradiancia:

$$dL_o(p, \omega_o) \propto dE(p, \omega_i)$$

Si lo ponemos en forma de cociente, sabremos exactamente cuál es la proporción de luz. A este cociente lo llamaremos $f_r(p, \omega_o \leftarrow \omega_i)$; la función de distribución de reflectancia bidireccional:

$$f_r(p, \omega_o \leftarrow \omega_i) = \frac{dL_o(p, \omega_o)}{dE(p, \omega_i)} = \frac{dL_o(p, \omega_o)}{L_i(p, \omega_i) \cos\theta_i \, d\omega_i} \text{(1/sr)}$$

> **Nota**(ción): dependiendo de la fuente que estés leyendo, es posible que te encuentres una integral algo diferente. Por ejemplo, en tanto en Wikipedia como en (Shirley and Morley 2003) se integra con respecto a los ángulos de salida ω_o, en vez de los incidentes.
>
> Aquí, usaremos la notación de integrar con respecto a los incidentes, como se hace en (Pharr, Jakob, and Humphreys 2016).

Las BRDF físicamente realistas tienen un par de propiedades importantes:

1. **Reciprocidad**: para cualquier par de direcciones ω_i, ω_o, se tiene que $f_r(p, \omega_i, \omega_o) = f_r(p, \omega_o \leftarrow \omega_i)$.
2. **Conservación de la energía**: La energía reflejada tiene que ser menor o igual que la incidente:

$$\int_{H^2(\mathbf{n})} f_r(p, \omega_o \leftarrow \omega_i) \cos\theta_i \, d\omega_i \leq 1$$

4.3.2. La función de distribución de transmitancia bidireccional (BTDF)

Si la BRDF describe cómo se refleja la luz, la *bidirectional transmittance distribution function* (abreviada BTDF) nos informará sobre la transmitancia; es decir, cómo se comporta la luz cuando *entra* en un medio. Generalmente serán dos caras de la misma moneda: cuando la luz impacta en una superficie, parte de ella, se reflejará, y otra parte se transmitirá.

Puedes imaginarte la BTDF como una función de reflectancia del hemisferio opuesto a donde se encuentra la normal de la superficie.

Denotaremos a la BTDF por

$$f_t(p, \omega_o \leftarrow \omega_i)$$

Al contrario que en la BRDF, ω_o y ω_i se encuentran en hemisferios diferentes.

4.3.3. Juntando la BRDF y la BTDF en La función de distribución de dispersión bidireccional

Convenientemente, podemos unir la BRDF y la BTDF en una sola expresión, llamada **la función de distribución de dispersión bidireccional** (*bidirectional scattering distribution function*, BSDF). A la BSDF la denotaremos por

$$f(p, \omega_o \leftarrow \omega_i)$$

> **Intuición:** *la BSDF son todas las posibles direcciones en las que puede salir disparada la luz.*

Usando esta definición, podemos obtener

$$dL_o(p, \omega_o) = f(p, \omega_o \leftarrow \omega_i) L_i(p, \omega_i) |\cos \theta_i| d\omega_i$$

Esto nos deja a punto de caramelo una nueva expresión de la randiancia en términos de la randiancia incidente en un punto p. Integrando la expresión anterior, obtenemos

$$L_o(p, \omega_o) = \int_{\mathbb{S}^2} f(p, \omega_o \leftarrow \omega_i) L_i(p, \omega_i) |\cos \theta_i| d\omega_i \tag{4.5}$$

siendo $\mathbb{S}^2$ la esfera.

Esta forma de expresar la radiancia es muy importante. Generalmente se le suele llamar la *ecuación de dispersión* (*scattering equation*, en inglés). Dado que es una integral muy importante, seguramente tengamos que evaluarla repetidamente. ¡Los métodos de Monte Carlo nos vendrán de perlas! Más adelante hablaremos de ella.

Las BSDFs tienen unas propiedades interesantes:

- **Positividad**: como los fotones no se pueden reflejar "negativamente", $f(p, \omega_o \leftarrow \omega_i) \geq 0$.
- **Reciprocidad de Helmotz:** se puede invertir un rayo de luz: $f(p, \omega_o \leftarrow \omega_i) = f(p, \omega_i \leftarrow \omega_o)$.
- **Conservación de la energía**: todos los fotones que llegan a la superficie deben ser reflejados o absorbidos. Es decir, no se emite ningún fotón nuevo:

$$\int_{H^2(\mathbf{n})} f(p, \omega_o \leftarrow \omega_i) \cos \theta_i \, d\omega_i \leq 1 \; \forall \omega_o$$

4.3.4. Reflectancia hemisférica

Puede ser útil tomar el comportamiento agregado de las BRDFs y las BTDFs y reducirlo un cierto valor que describa su comportamiento general de dispersión. Sería Algo así como un resumen de su distribución. Para conseguirlo, vamos a introducir dos nuevas funciones:

La **reflectancia hemisférica-direccional** (*hemispherical-directional reflectance*) describe la reflexión total sobre un hemisferio debida a una fuente de luz que proviene desde la dirección ω_o:

$$\rho_{hd}(\omega_o) = \int_{H^2(n)} f_r(p, \omega_o \leftarrow \omega_i) |\cos\theta_i| \, d\omega_i$$

Por otra parte, la **reflectancia hemisférica-hemisférica** (*hemispherical-hemispherical reflectance*) es un valor espectral que nos proporciona el ratio de luz incidente reflejada por una superficie, suponiendo que llega la misma luz desde todas direcciones:

$$\rho_{hh} = \frac{1}{\pi} \int_{H^2(n)} \int_{H^2(n)} f_r(p, \omega_o \leftarrow \omega_i) |\cos\theta_o \, \cos\theta_i| \, d\omega_o \, d\omega_i$$

4.3.5. Reflejos

Una vez hemos definido las funciones de distribución bidireccionales, debemos encargarnos de modelar el comportamiento explícitamente. Para ello, veamos cómo los materiales modifican las distribuciones.

En esencia, los reflejos se pueden clasificar en cuatro grandes tipos:

- **Difusos** (*Diffuse*): esparcen la luz en todas direcciones casi equiprobablemente. Por ejemplo, la tela y el papel son materiales difusos.
- **Especulares brillantes** (*Glossy specular*): la distribución de luz se asemeja a un cono. La chapa de un coche es un material especular brillante.
- **Especulares perfectos** (*Perfect specular*): en esencia, son espejos. El ángulo de salida de la luz es muy pequeño, por lo que reflejan casi a la perfección lo que les llega.

- **Retrorreflectores** (*Retro reflective*): la luz se refleja en dirección contraria a la de llegada. Esto es lo que sucede a la luna.

Ten en cuenta que es muy difícil encontrar objetos físicos que imiten a la perfección un cierto modelo. Suelen recaer en un híbrido entre dos o más modelos.

Fijado un cierto modelo, la función de distribución de reflectancia, BRDF, puede ser **isotrópica** o **anisotrópica**. Los materiales isotrópicos mantienen las propiedades de reflectancia invariantes ante rotaciones; es decir, la distribución de luz es la misma en todas direcciones. Por el contrario, los anisotrópicos reflejan diferentes cantidades de luz dependiendo desde dónde los miremos. Los ejemplos más habituales de materiales anisotrópicos son las rocas y la madera.

4.4. La rendering equation

Y, finalmente, tras esta introducción de los principales conceptos radiométricos, llegamos a la ecuación más importante de todo este trabajo: la **rendering equation**; también llamada la **ecuación del transporte de luz**.

> **Nota**(ción): esta vez no traduciré el concepto. Es cierto que afea un poco la escritura teniendo en cuenta que esto es un texto en castellano. Sin embargo, la otra opción es inventarme una traducción que nadie usa.

Antes de comenzar, volvamos a plantear de nuevo la situación: nos encontramos observando desde nuestra pantalla una escena virtual mediante la cámara. Queremos saber qué color tomará un pixel específico. Para conseguirlo, dispararemos rayos desde nuestro punto de vista hacia el entorno, haciendo que reboten en los objetos. Cuando un rayo impacte en una superficie, adquirirá parte de las propiedades del material del objeto. Además, de este rayo surgirán otros nuevos (un rayo dispersado y otro refractado), que a su vez repetirán el proceso. La información que se obtiene a partir de estos caminos de rayos nos permitirá darle color al píxel.

La *rendering equation* se va a encargar de describir analíticamente cómo ocurre esto.

Un último concepto más: denotemos por $L_e(p, \omega_o)$ a **la radiancia producida por los materiales emisivos**. Por ejemplo, una luz emite radiancia por sí misma.

Bien, partamos de la ecuación de para la radiancia reflejada:

$$L_o(p, \omega_o) = \int_{H^2(\mathbf{n})} f(p, \omega_o \leftarrow \omega_i) L_i(p, \omega_i) \cos\theta_i \, d\omega_i$$

Vamos a buscar expresar la radiancia incidente en términos de la radiancia reflejada. Para ello, usamos la propiedad de que la radiancia a lo largo de un rayo no cambia.

Si a una superficie le llega un fotón desde alguna parte, debe ser porque *"alguien"* ha tenido que emitirlo. El fotón necesariamente ha llegado a partir de un rayo. La propiedad nos dice que la radiancia no ha podido cambiar en el camino.

Pues bien, consideremos una función $r : \mathbb{R}^3 \times \Omega \to \mathbb{R}^3$ tal que, dado un punto p y una dirección ω, devuelve el siguiente punto de impacto en una superficie. En esencia, es una función de *ray casting*.

> TODO: foto como la de https://pellacini.di.uniroma1.it/teaching/graphics17b/lectures/12_pathtracing p.29

Esta función nos permite expresar el punto anterior de la siguiente forma:

$$L_i(p, \omega) = L_o(r(p, \omega), -\omega)$$

Esto nos permite cambiar la expresión de L_i en la integral anterior:

$$L_o(p, \omega_o) = \int_{H^2(\mathbf{n})} f(p, \omega_o \leftarrow \omega_i) L_o(r(p, \omega_i), -\omega_i) \cos\theta_i \, d\omega_i$$

Finalmente, la radiancia total vendrá dada por la suma de la radiancia emitida y la reflejada:

$$L(p, \omega_o) = L_e(p, \omega_o) + \int_{H^2(\mathbf{n})} f(p, \omega_o \leftarrow \omega_i) L_o(r(p, \omega_i), -\omega_i) \cos\theta_i \, d\omega_i \tag{4.6}$$

Y con esto, ¡hemos obtenido la *rendering equation*!

Si quieres ver gráficamente cómo funciona, te recomiendo pasarte por (Arnebäck 2019). Es un vídeo muy intuitivo.

Si nos paramos a pensar, la ecuación de reflexión es muy similar a la de renderizado. Sin embargo, hay un par de matices que las hacen muy diferentes:

- La ecuación de reflexión describe cómo se comporta la luz reflejada en un cierto punto. Es decir, tiene un ámbito local. Además, para calcular la radiancia reflejada, se necesita conocer la radiancia incidente.
- La *rendering equation* calcula las condiciones globales de la luz. Además, no se conocen las radiancias de salida.

Este último matiz es importante. Para renderizar una imagen, se necesita calcular la radiancia de salida para aquellos puntos visibles desde nuestra cámara.

Referencias

(Pharr, Jakob, and Humphreys 2016), (Wikipedia: Radiometry 2021), (StudySession 2021), (Berkeley cs184 2022, Radiometry & Photometry), (Wikipedia: Función de distribución de reflectancia bidireccional 2022), (Wikipedia: Transmittance 2021)

- https://matmatch.com/learn/property/isotropy-anisotropy
- https://pellacini.di.uniroma1.it/teaching/graphics17b/lectures/12_pathtracing.pdf

5. ¡Construyamos un path tracer!

> TODO: una pequeña introducción. Hablar de los offline renderers disponibles. Tras esto, dar paso a los frameworks de real time RT. Esta sección estará fuertemente basada en el tutorial de Nvidia de nvpro samples + mini path tracer.

5.1. Requisitos de *real time ray tracing*

5.1.1. Arquitecturas de gráficas

> Turing, Ampere, RDNA2, Arc... NOTE: sería interesante enlazarlo con la sección de rendimiento.

5.1.2. Frameworks y API de ray tracing en tiempo real

> TODO: hablar de DXR, Vulkan, OptiX... Hablar de por qué he escogido Vulkan + Nvidia Designworks (nv-pro samples)

5.2. Setup del proyecto

5.3. RT pipeline

5.4. Estructuras de aceleración

> TODO: En esencia, hablar de la parafernalia que montamos con el BLAS y el TLAS.

5.5. Shaders

5.5.1. Tipos de shaders en RT

5.5.2. Shader binding table

5.6. Asmiray

5.7. Transporte de luz

> TODO: creo... que esto no debería ir aquí. Pero no quiero tampoco que el capítulo de radiometría sea un tocho impresionante.

Hemos llegado a una de las partes más importantes de este trabajo. Es el momento de poner en concordancia todo lo que hemos visto a lo largo de los capítulos anteriores.

Empecemos por la dispersión. ¿Recuerdas la ecuación de dispersión [4.5]? Podemos estimarla utilizando Monte Carlo:

$$\begin{aligned} L_o(p, \omega_o) &= \int_{\mathbb{S}^2} f(p, \omega_o \leftarrow \omega_i) L_i(p, \omega_i) |\cos \theta_i| d\omega_i \\ &\approx \frac{1}{N} \sum_{j=1}^{N} \frac{f(p, \omega_o \leftarrow \omega_j) L_i(p, \omega_j) |\cos \theta_j|}{p(\omega_j)} \end{aligned}$$

5.7.1. Materiales y objetos

> NOTE: esto son notas para el Andrés del futuro. Sí, lo sé, está bastante claro solo con leerlo (□_□;)

Si quiero meter las BxDFs en los materiales tal y como tenía pensado (es decir, unas cuantas flags que me indiquen la BxDF que tengo que usar), tengo que...

1. Modificar `common/obj_loader.h/MaterialObj` para meterle las flags necesarias.
2. Modificar acordemente `shaders/host_device.h/WaveFronMaterial`.
3. Secuestrar `ObjLoader::loadModel()` para indicarle los parámetros nuevos.
4. (*Creo que no hace falta tocar* `HelloVk::loadModel()` *de esta manera*)
5. Toquetear los shaders para que me saque las flags.

CREO que de esta manera no me va a hacer falta tocar framebuffers. Simplemente, todo dependerá de mi material y ya.

Creo.

5.8. Fuentes de luz

> TODO: point lights, area lights, ambient lights...
>
> TODO: estas son notas muy puntuales (*como las luces, jeje*). Ya las revisaré más adelante.

La interfaz se encuentra en `host_device.h`. Describe cómo comunicarse con la GPU.

Ahora mismo, tenemos 3 constantes: tipo de luz:

```
vec3  lightPosition;    // (x, y, z)
float lightIntensity;   // (Intensidad)
int   lightType;        // (0 => point light, 1 => area light)
```

Sería interesante añadir algunas constantes para controlar el tamaño (radio, posición, normal para las de área...)

5.8.1. Point lights + spotlights

> pbr-book, point lights: *"Strictly speaking, it is incorrect to describe the light arriving at a point due to a point light source using units of radiance. Radiant intensity is instead the proper unit for describing emission from a point light source, as explained in Section 5.4. In the light source interfaces here, however, we will abuse terminology and use Sample_Li() methods to report the illumination arriving at a point for all types of light sources, dividing*

> *radiant intensity by the squared distance to the point p to convert units. Section 14.2 revisits the details of this issue in its discussion of how delta distributions affect evaluation of the integral in the scattering equation. In the end, the correctness of the computation does not suffer from this fudge, and it makes the implementation of light transport algorithms more straightforward by not requiring them to use different interfaces for different types of lights."*

```
// https://github.com/mmp/pbrt-v3/blob/master/src/lights/point.h
// https://github.com/mmp/pbrt-v3/blob/master/src/lights/point.cpp

Spectrum sample_light(interaccion, vec2 u, vec3 wi, float pdf,
    visibility_tester) {
    wi = normalize(posicion_luz - interraccion.p);
    pdf = 1.f;
    // testeo de visibilidad. Opcional, I guess.

    return intensidad / distancia_al_cuadrado(posicion_luz, interraccion.p)
        ;
}
```

La potencia total emitida por la luz puede calcularse integrando la intensidad desprendida en toda su superficie. Asumiendo la intensidad constante:

$$\Phi = \int_{\mathbb{S}^2} I d\omega = I \int_{\mathbb{S}^2} d\omega = 4\pi I$$

Las spotlights son variaciones de las point lights iluminando en un cono.

5.8.2. Fuentes de área

Para simplificar la implementación, podemos asumir que son rectangulares.

Nos van a hacer falta técnicas de Monte Carlo para solucionar el problema de calcular integrales a lo largo de su superficie.

Primero, lo mejor es asumir un cuadrado, y después, extender la interfaz para meter otras formas (es decir, rectángulos. Porque lo otro sería mucha parafernalia innecesaria).

Código fuente

Referencias

- https://github.com/dannyfritz/awesome-ray-tracing
- https://www.wikiwand.com/en/Radeon
- https://www.wikiwand.com/en/List_of_Nvidia_graphics_processing_units#/GeForce_30_series

6. Análisis de rendimiento

> TODO: para completar esta parte, necesitamos ambas implementaciones (en CPU y GPU) listas. Hasta entonces, esto se queda vacío. NOTE: Podría preguntarle a Kako que tire benchmark en su 2060, y a Manu con su 3060 (¿ti?)

Referencias

7. El futuro de Ray Tracing

> TODO:

Referencias

A. Metodología; o cómo se hizo este trabajo

Cualquier proyecto de una envergadura considerable necesita ser planificado con antelación. En este capítulo vamos a hablar de cómo se ha realizado este trabajo: mostraremos las herramientas usadas, los ciclos de desarrollo, integración entre documentación y path tracer, y otras influencias que han afectado al producto final.

A.1. Influencias

Antes de comenzar con la labor, primero uno se debe hacer una simple pregunta:

> *"Y esto, ¿por qué me importa?"*

Dar una respuesta contundente a este tipo de cuestiones nunca es fácil. Sin embargo, sí que puedo proporcionar motivos por los que he querido escribir sobre ray tracing.

Una de las principales influencias ha sido Digital Foundry. Este grupo de divulgación se dedica al estudio de las técnicas utilizadas en el mundo de los videojuegos. El inicio de la era del ray tracing en tiempo real les llevó a dedicar una serie de vídeos y artículos a esta tecnología, y a las diferentes maneras en las que se ha implementado. Se puede ver un ejemplo en (Digital Foundry 2020).

Dado que esta área combina tanto informática, matemáticas y una visión artística, ¿por qué no explorarlo a fondo?

Ahora que se ha decidido el tema, es hora de ver cómo atacarlo.

Soy un fiel creyente del aprendizaje mediante el juego. Páginas como Explorable Explanations, el blog de Bartosz Ciechanowski, el proyecto *The napkin* o el divulgador 3Blue1Brown repercuten inevitablemente en la manera en la que te planteas cómo comunicar textos científicos. Por ello, aunque esto a fin de cuentas es un trabajo de fin de grado de una carrera, quería ver hasta dónde era capaz de llevarlo.

Otro punto importante es la *manera* de escribir. No me gusta especialmente la escritura formal. Prefiero ser distendido. Por suerte, parece que el mundo científico se está volviendo más informal (Nature 2016), así que no soy el único que aprueba esta tendencia. Además, la estructura clásica de un escrito matemático de "teorema, lema, demostración, corolario" no me agrada especialmente. He intentado preservar su estructura, pero sin ser tan explícito. Estos dos puntos, en conjunto, suponen un balance entre formalidad y distensión difícil de mantener.

A.2. Ciclos de desarrollo

Este proyecto está compuesto por 2 grandes pilares: documentación –lo que estás leyendo, ya sea en PDF o en la web– y software.

La metodología que se ha seguido es, en esencia, una versión de Agile muy laxa (Beck et al. 2001).

Para empezar, se implementaron los tres libros de Shirley de la "serie In One Weekend": In One Weekend (Shirley 2020a), The Next Week (Shirley 2020b), y The Rest of your Life (Shirley 2020c).

Tras esto, comenzó a desarrollarse el motor por GPU. Cuando se consiguió una base sólida (que se puede ver en este issue del repositorio), se empezó a alternar entre escritura de documentación y desarrollo del software. A fin de cuentas, no tiene sentido implementar algo que no se conoce.

Para apoyar el desarrollo, se ha utilizado Github. Más adelante hablaremos de cómo esta plataforma ha facilitado el trabajo.

A.3. Diseño

> TODO: hablar de paleta de colores, tipografía...

El diseño juega un papel fundamental en este proyecto. Todos los elementos visuales han sido escogidos con cuidado, de forma que se preserve la estética.

Se ha creado **un diseño que preserve el equilibrio entre la profesionalidad y la distensión**.

A.3.1. Bases del diseño

Para la documentación en versión PDF, usamos como base la *template* Eisvogel. Esta es una elegante plantilla fácil de usar para LaTeX. Uno de sus puntos fuertes es la personalización, la cual aprovecharemos para darle un toque diferente.

La web utiliza como base el estilo generado por Pandoc, el microframework de css Bamboo y unas modificaciones personales.

A.3.2. Tipografías

Se ha utilizado una combinación de las siguietes tipografías:

- Crimson Pro: una tipografía serif clara, legible y contemporánea. Funciona muy bien en densidades más bajas, como 11pt. Es ideal para la versión en PDF. Además, liga estupendamente con Source Sans Pro, utilizada para los títulos en la plantilla Eisvogel.
- Fraunces: de lejos, la fuente más interesante de todo este proyecto. Es una soft-serif *old style*, pensada para títulos y similares (lo que se conoce como *display*). Es usada en los títulos de la web. Una de sus propiedades más curiosas es que modifica activamente los glifos dependiendo del valor del *optical size axis*, el peso y similares. Recomiendo echarle un ojo a su repositorio de Github.
- Rubik: La elección de Rubik es peculiar. Por sí sola, no casa con el proyecto. Sin embargo, combinada con Fraunces, proporcionan un punto de elegancia y famil-

iaridad a la web. Su principal fuerte es la facilidad para la comprensión lectora en pantallas, algo que buscamos para la página web.
- Julia Mono: monoespaciada, pensada para computación científica. Llevo usándola bastante tiempo, y combia bien con Crimson Pro.
- Jetbrains Mono: otra tipografía monoespaciada open source muy sólida, producida por la compañía Jetbrains. Se utiliza en la web para los bloques de código.

Todas estas fuentes permiten un uso no comercial gratuito.

A.4. Herramientas

> TODO: hablar de Figma, LTeX... (Issue #40), bamboo.css

A.5. 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.

Github

A.5.1. Github Actions

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

A.5.2. Github Projects

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

A.5.3. Estilo de commits

Referencias

\end

B. Glosario de términos

> *It's dangerous to go alone, take this.*

Tener en mente *todos* los conceptos y sus expresiones que aparecen en un libro como este es prácticamente imposible. Tampoco hay necesidad de ello, realmente. ¡Vaya desperdicio de cabeza! Por eso, aquí tienes recopilada una lista con todos los elementos importantes y un enlace a sus secciones correspondientes.

B.1. Notación

Concepto	Notación
Puntos	Letras mayúsculas: $P, Q, \ldots$
Escalares	Letras minúsculas: $a, b, c, k, \ldots$
Vectores	Letras minúsculas en negrita: $\mathbf{v}, \mathbf{w}, \mathbf{n}, \ldots$. Si están normalizados, se les pone gorrito (por ejemplo, $\hat{\mathbf{n}}$)
Matrices	Letras mayúsculas en negrita: $\mathbf{M}$. Por columnas.
Producto escalar	$\mathbf{v} \cdot \mathbf{w}$. Si es el producto escalar de un vector consigo mismo, a veces pondremos $\mathbf{v}^2$
Producto vectorial	$\mathbf{v} \times \mathbf{w}$
Variables aleatorias	X, Y. ξ representa una variable aleatoria con distribución uniforme en $[0, 1)$.

B.2. Radiometría

Concepto	Expresiones
Ángulo sólido, derivada [4.1]	$\omega = \frac{A}{r^2}$ $d\omega = \sin\theta \, d\theta \, d\phi$
Hemisferio de direcciones alrededor de un vector	$H^2(\mathbf{n})$
Carga de energía	$Q = hf = \frac{hc}{\lambda}$
Flujo radiante, potencia	$\Phi = \frac{dQ}{dt}$ $\Phi = \int_A \int_{H^2(\mathbf{n})} L_o(p,\omega) d\omega^{\perp} dA$
Irradiancia, radiancia emitida	$E = \frac{\Phi}{A}$ $E(p) = \frac{d\Phi}{dA}$ $E(p,\mathbf{n}) = \int_\Omega L_i(p,\omega) \lvert cos\theta \rvert d\omega$ $E(p,\mathbf{n}) = \int_0^{2\pi} \int_0^{\pi/2} L_i(p,\theta,\phi) \cos\theta \ \sin\theta \, d\theta \, d\phi$ $E(p,\mathbf{n}) = \int_A L \cos\theta \frac{\cos\theta_o}{r^2} dA$
Intensidad radiante	$I = \frac{d\Phi}{d\omega}$

Concepto	Expresiones
Radiancia	$L(p,\omega) = \dfrac{dE_\omega(p)}{d\omega}$ $L(p,\omega) = \dfrac{d^2\Phi(p,\omega)}{d\omega\, dA^\perp} = \dfrac{d^2\Phi(p,\omega)}{d\omega\, dA\, \cos\theta}$ $L^+(p,\omega) = \lim_{t\to 0^+} L(p + t\mathbf{n_p}, \omega)$ $L^-(p,\omega) = \lim_{t\to 0^-} L(p + t\mathbf{n_p}, \omega)$
Radiancia incidente	$L_i(p,\omega) = \begin{cases} L^+(p,-\omega) & \text{si } \omega\cdot\mathbf{n_p} > 0 \\ L^-(p,-\omega) & \text{si } \omega\cdot\mathbf{n_p} < 0 \end{cases}$
Radiancia reflejada, radiancia de salida	$L_o(p,\omega) = \begin{cases} L^+(p,\omega) & \text{si } \omega\cdot\mathbf{n_p} > 0 \\ L^-(p,\omega) & \text{si } \omega\cdot\mathbf{n_p} < 0 \end{cases}$
Ecuación de dispersión	$L_o(p,\omega_o) = \int_{\mathbb{S}^2} f(p,\omega_o \leftarrow \omega_i) L_i(p,\omega_i) \lvert\cos\theta_i\rvert d\omega_i$
BRDF	$f_r(p,\omega_o \leftarrow \omega_i) = \dfrac{dL_o(p,\omega_o)}{dE(p,\omega_i)} = \dfrac{dL_o(p,\omega_o)}{L_i(p,\omega_i)\cos\theta_i\, d\omega_i}$
BTDF	$f_t(p,\omega_o \leftarrow \omega_i)$
BSDF	$f(p,\omega_o \leftarrow \omega_i)$

Bibliografía

A. Romero Sarabia. 2021. *Apuntes de La Asignatura Curvas y Superficies*.

Arnebäck. 2019. "An Explanation of the Rendering Equation." January 10, 2019. https://www.youtube.com/watch?v=eo_MTI-d28s.

Beck, Kent, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, et al. 2001. "Manifesto for Agile Software Development." http://www.agilemanifesto.org/.

Berkeley cs184. 2022. "Monte Carlo Integration Cs184/284a." 2022. https://cs184.eecs.berkeley.edu/sp22.

Caulfield, Brian. 2020. "What's the Difference Between Ray Tracing, Rasterization?" May 22, 2020. https://blogs.nvidia.com/blog/2018/03/19/whats-difference-between-ray-tracing-rasterization/.

Digital Foundry. 2020. "Cyberpunk 2077 PC: What Does Ray Tracing Deliver... And Is It Worth It?" December 19, 2020. https://www.youtube.com/watch?v=6bqA8F6B6NQ.

Fabio Pellacini, Steve Marschner. 2017. "Fundamentals of Computer Graphics." 2017. https://pellacini.di.uniroma1.it/teaching/graphics17b/.

Galvin. n.d. "Random Variables." Accessed March 20, 2022. https://www3.nd.edu/~dgalvin1/10120/10120_S16/Topic17_8p4_Galvin_class.pdf.

Haines, Eric, and Tomas Akenine-Möller, eds. 2019. "Ray Tracing Gems." Apress. 2019. http://raytracinggems.com.

Nature. 2016. "Scientific Language Is Becoming More Informal." November 8, 2016. https://doi.org/10.1038/539140a.

Owen, Art B. 2013. *Monte Carlo Theory, Methods and Examples*. https://artowen.su.domains/mc/.

Pharr, Matt, Wenzel Jakob, and Greg Humphreys. 2016. "Physically Based Rendering: From Theory to Implementation (3rd Ed.)." San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. November 2016. https://www.pbr-book.org/3ed-2018/contents.

QuantumFracture. 2021. "Ya, En Serio, ¿Qué Es La Luz?" December 10, 2021. https://www.youtube.com/watch?v=DkcEAz09Buo.

"Rendering." n.d. Accessed March 20, 2022. https://sciencebehindpixar.org/pipeline/rendering#:%7E:text=They%20said%20it%20takes%20at,to%20render%20that%20many%20frames.

Shirley, Peter. 2020a. "Ray Tracing in One Weekend." 2020. https://raytracing.github.io/books/RayTracingInOneWeekend.html.

———. 2020b. "Ray Tracing: The Next Week." 2020. https://raytracing.github.io/books/RayTracingTheNextWeek.html.

———. 2020c. "Ray Tracing: The Rest of Your Life." 2020. https://raytracing.github.io/books/RayTracingTheRestOfYourLife.html.

Shirley, Peter, and R. Keith Morley. 2003. *Realistic Ray Tracing*. 2nd ed. USA: A. K. Peters, Ltd. https://www.taylorfrancis.com/books/mono/10.1201/9780429294891/realistic-ray-tracing-peter-shirley-keith-morley.

StudySession. 2021. "Solid Angle Derivation & Intuition." March 4, 2021. https://www.youtube.com/watch?v=WrKsgBElPWA.

The Khronos® Vulkan Working Group. 2022. "Vulkan® 1.2.210 - KHR Extensions: 33. Ray Intersection." March 29, 2022. https://www.khronos.org/registry/vulkan/specs/1.2-khr-extensions/html/chap33.html#ray-intersection-candidate-determination.

tracing, Wikipedia: Ray. 2022. "Ray Tracing (Graphics)." March 7, 2022. https://en.wikipedia.org/wiki/Ray_tracing_(graphics).

Wikipedia: Barycentric coordinate system. 2022. "Barycentric Coordinate System." March 14, 2022. https://en.wikipedia.org/wiki/Barycentric_coordinate_system.

Wikipedia: Computer. 2022. “Computer.” March 13, 2022. https://en.wikipedia.org/wiki/Computer.

Wikipedia: Differential geometry of surfaces. 2022. “Differential Geometry of Surfaces.” January 14, 2022. https://en.wikipedia.org/wiki/Differential_geometry_of_surfaces.

Wikipedia: Distribución de probabilidad. 2022. “Distribución de Probabilidad.” February 28, 2022. https://es.wikipedia.org/wiki/Distribuci%C3%B3n_de_probabilidad.

Wikipedia: Estimador. 2022. “Estimador.” March 14, 2022. https://es.wikipedia.org/wiki/Estimador?oldformat=true.

Wikipedia: Expected value. 2022. “Expected Value.” March 14, 2022. https://en.wikipedia.org/wiki/Expected_value.

Wikipedia: Función de distribución de reflectancia bidireccional. 2022. “Función de Distribución de Reflectancia Bidireccional.” January 28, 2022. https://es.wikipedia.org/wiki/Funci%C3%B3n_de_distribuci%C3%B3n_de_reflectancia_bidireccional.

Wikipedia: Función de probabilidad. 2022. “Función de Probabilidad.” February 10, 2022. https://es.wikipedia.org/wiki/Funci%C3%B3n_de_probabilidad.

Wikipedia: history of photography. 2022. “History of Photography.” March 18, 2022. https://en.wikipedia.org/wiki/History_of_photography.

Wikipedia: Implicit surface. 2022. “Implicit Surface.” March 13, 2022. https://en.wikipedia.org/wiki/Implicit_surface.

Wikipedia: Kodak. 2022. “Kodak.” March 14, 2022. https://es.wikipedia.org/wiki/Kodak.

Wikipedia: Método de la transformada inversa. 2022. “Método de La Transformada Inversa.” February 20, 2022. https://es.wikipedia.org/wiki/M%C3%A9todo_de_la_transformada_inversa.

Wikipedia: Parametric surface. 2021. “Parametric Surface.” December 15, 2021. https://en.wikipedia.org/wiki/Parametric_surface.

Wikipedia: Probability density function. 2022. “Probability Density Function.” March 17, 2022. https://en.wikipedia.org/wiki/Probability_density_function.

Wikipedia: Radiometry. 2021. “Radiometry.” September 12, 2021. https://en.wikipedia.org/wiki/Radiometry.

Wikipedia: Rejection sampling. 2022. “Rejection Sampling.” February 4, 2022. https://en.wikipedia.org/wiki/Rejection_sampling.

Wikipedia: rendering (computer graphics). 2022. “Rendering (Computer Graphics).” March 8, 2022. https://en.wikipedia.org/wiki/Rendering_(computer_graphics).

Wikipedia: Rendering equation. 2021. “Rendering Equation.” March 9, 2021. https://en.wikipedia.org/wiki/Rendering_equation.

Wikipedia: Transmittance. 2021. “Transmittance.” January 3, 2021. https://en.wikipedia.org/wiki/Transmittance.

Wikipedia: Variable aleatoria. 2021. “Variable Aleatoria.” August 30, 2021. https://es.wikipedia.org/wiki/Variable_aleatoria.