INFORMÁTICA GRÁFICA

Path Tracing

Curso 2019-2020

Autores:
Víctor Miguel Peñasco Estívalez 741294
Rubén Rodríguez Esteban 737215

Tabla de contenidos

Introducción

En este informe se va a describir el diseño desde cero del algoritmo de Path tracer, el cual es un método basado en el algoritmo de Monte Carlo para el renderizado de imágenes en tres dimensiones, de tal forma que la iluminación global y el transporte de la luz en general sean lo más fieles a la realidad en la medida de lo posible. A lo largo del documento se van a exponer las cuestiones y aspectos necesarios para conseguir el algoritmo del path tracer básico, así como las extensiones adicionales introducidas para renderizar las escenas como de conseguir que dichas imágenes sean visualmente mucho más impactantes.

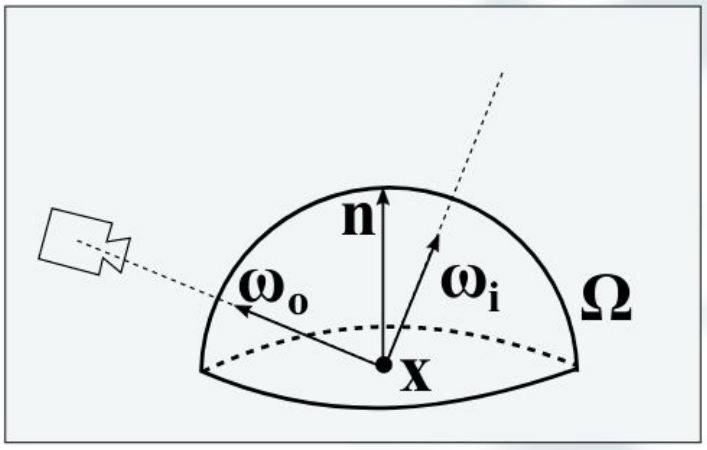
Adicionalmente, se van a describir las diversas decisiones de diseño que se han tomado, justificando las razones en las que se fundamentan y, finalmente, se van a proporcionar un conjunto de renders de muestra que han sido generados durante la elaboración del trabajo, donde para cada uno de ellos, se va a comentar la naturaleza de la escena (que elementos la forman) y los distintos comportamiento físicos de la luz al interactuar con los cuerpos de la escena.

Path tracer básico

En esta sección del documento se van a comentar distintos aspectos de la parte básica del path tracer, compuesta por dos únicas primitivas geométricas (plano y esfera), y que no dispone de más extensiones adicionales. Estas extensiones adicionales se detallan más adelante en la sección correspondiente.

Ecuación de render

La ecuación de render es, además de Monte Carlo, la pieza clave del algoritmo de Path Tracing. Es sobre lo que se fundamenta todo el transporte de luz que se simula en la implementación.



$$L_o(\mathbf{x}, \omega_o) = \underbrace{L_e(\mathbf{x}, \omega_o)}_{\text{EMISSION}} + \int_\Omega \underbrace{L_i(\mathbf{x}, \omega_i)}_{\text{INCOMING LIGHT}} \underbrace{f_r(\mathbf{x}, \omega_i, \omega_o)}_{\text{BRDF}} \underbrace{|\mathbf{n} \cdot \omega_i|}_{\text{GEOMETRY}} d\omega_i$$

Figura 1
Ecuación de Render

En primer lugar se va a proceder a explicar qué es lo que significan cada uno de los términos de la ecuación de Render a fin de cuentas de poder reflejar como se ha efectuado el cálculo de acuerdo al algoritmo del Path Tracing diseñado.

L_o(x, w_o) se define como la radiancia total que llega a un observador (cámara) procedente de la dirección w_o cuando se encuentra mirando a un punto concreto del espacio x. De acuerdo a la ecuación, para poder calcular esa magnitud de radiancia se debe disponer de la cantidad de luz que llega desde ese punto x en la dirección w_o definida como **L_e(x, w_o)**, la superficie del material donde se halla el punto x y la dirección del rayo respecto de la normal de la superficie. Además, teniendo en cuenta que se está considerando la iluminación que le llega al punto, la luz en x debe ser integrada en toda la hemiesfera definida por el punto y su superficie, definida como **L_i(x, w_i)**. El término **f_r(x, w_i, w_o)** se denomina BRDF, y es un factor que depende de las propiedades del material.

Por último, se encuentra el término geométrico **|n * w_i|**, definido como el producto escalar entre la dirección normal de la superficie del objeto y la dirección del rayo reflejado tras interceptar con la superficie del objeto. Es importante enfatizar que cuanto más perpendicular a la superficie llegue la luz, más intensa será porque el coseno entre ambas direcciones, normal y dirección del rayo de luz entrante, es más próximo a 1.

A la hora de implementar esta ecuación, se han tomado múltiples decisiones de diseño que se detallan a continuación:

1. Se ha asumido que un objeto que emite luz, no la recibe, y viceversa. Esta asunción estaba enunciada en las instrucciones del trabajo, y simplifica en gran medida la implementación del algoritmo.

2. Los diferentes caminos generados por el path tracer tienen como condición de terminación no intersectar con ningún objeto de la escena, por evento de absorción indicado por la ruleta rusa o por alcanzar un cuerpo que emita luz. En este último caso, se están tomando en consideración las luces de área. Para tener en cuenta el impacto de las luces puntuales en cada interacción con un objeto a lo largo de los caminos se hace uso de la técnica “Next event estimation”.

3. La ecuación de Render se ha calculado para cada uno de los canales RGB ya que la ecuación solo tiene en cuenta la radiancia y no los colores.

4. Dado que la implementación del algoritmo de Monte Carlo emplea la técnica de ruleta rusa, en cada interacción solo se calcula la BRDF propia del evento actual indicado por la ruleta rusa. Las probabilidades de dichos eventos están definidas para cada objeto en la construcción de la escena.

El algoritmo de Path Tracing comienza muestreando un punto aleatorio dentro del píxel. Cada uno de estos píxeles son secciones cuadradas del plano de proyección que se establece para renderizar la escena. Una vez se tiene este punto, se lleva a cabo el proceso descrito a continuación, repetido tantas veces como se haya establecido mediante el parámetro PPP (Paths per pixel), cuyo objetivo es reducir el nivel de ruido a costa de aumentar el tiempo de cálculo.

Se genera una dirección restando al punto muestreado la localización de la cámara. Esta dirección es la que tendrá el rayo inicial del algoritmo. Se procede a calcular si el rayo intersecta con los objetos de la escena, para finalmente quedarse con el más cercano de ellos. Para determinar si un objeto intersecta o no con el rayo, sólo se considera cierta la intersección si la distancia del inicio del rayo al objeto es mayor que un valor límite (muy cercano a 0). No se hace la comparación con 0 para evitar contar la intersección con el mismo objeto desde el que se lanza el rayo. Esto puede pasar debido a errores numéricos de precisión.

Es el momento de realizar ruleta rusa. Los coeficientes dados por las propiedades del material (difuso, especular de Phong, perfecto especular, refracción perfecta) indican la probabilidad de que suceda cada uno de los eventos, reservando siempre una determinada probabilidad para el evento de absorción. Dependiendo del evento seleccionado de forma aleatoria, se multiplica el resultado de evaluar la ecuación de render teniendo en cuenta sólo la BRDF propia del evento, por la iluminación conseguida hasta el momento a lo largo del camino del rayo, tal y como se ve en la Figura 2.

$$L_o(\mathbf{x_1}, \omega_{\mathbf{o1}}) \approx L_e(\mathbf{x_n}, \omega_{\mathbf{on}}) \prod_{j=1}^{n} \frac{f_r(\mathbf{x_j}, \omega_{\mathbf{ij}}, \omega_{\mathbf{oj}}) |\mathbf{n_j} \cdot \omega_{\mathbf{ij}}|}{p(\omega_{\mathbf{ij}})}$$

Figura 2
Aproximación de la ecuación de render mediante Path Tracing (Monte Carlo)

La BRDF de Phong (véase Figura 3) es un término en base al coeficiente difuso, el coeficiente especular, y la parte geométrica. El cálculo de la BRDF se lleva a cabo dependiendo de si el objeto con el que se ha intersectado es difuso o especular. En estos casos, para calcular la contribución de las luces puntuales en ese punto del objeto intersectado se efectúan de la siguiente manera.

La parte difusa de la BRDF se calcula multiplicando por el coeficiente difuso, K_d y dividiéndolo por π. La parte especular se calcula multiplicando el coeficiente especular por la brillantez del objeto intersectado sumandole 2 y dividido por 2π. Por último, la parte especular es multiplicada por el producto escalar entre la dirección del rayo que se originaría con reflexión perfecta, y el rayo real generado como resultado de la intersección, todo ello elevado a la brillantez del objeto.

$$f_r(\mathbf{x}, \omega_{\mathbf{i}}, \omega_{\mathbf{o}}) = \frac{k_d}{\pi} + k_s \frac{\alpha + 2}{2\pi} |\omega_{\mathbf{o}} \cdot \omega_{\mathbf{r}}|^{\alpha}$$

Figura 3
Cálculo de la BRDF de Phong

La BRDF de Phong sólo se evalúa de forma completa para el cálculo de la luz directa en Next event estimation, técnica que se explica más adelante. En el muestreo por Monte Carlo, se evalúa sólo de acuerdo al evento indicado por ruleta rusa. Como además se utiliza para el muestreo del siguiente rebote **Uniform Cosine Sampling**, el cálculo de la ecuación de render, por ejemplo en el caso de evento difuso, queda en multiplicar la luminancia (se obtendrá más adelante) por la componente difusa y dividir entre la probabilidad de evento difuso. Esta simplificación sucede al dividir la ecuación de render entre la pdf (función de distribución de probabilidad) utilizada para el muestreo.

$$L_o(\mathbf{x}, \omega_{\mathbf{o}}) = \int_0^{2\pi} \int_0^{\frac{\pi}{2}} L_i(\mathbf{x}, \omega_{\mathbf{i}}) f_r(\mathbf{x}, \omega_{\mathbf{i}}, \omega_{\mathbf{o}}) \cos\theta_i \sin\theta_i d\theta_i d\phi_i$$

Figura 4
Ecuación de Render

$$p(\theta_i) = 2 \sin \theta_i \cos \theta_i$$

Figura 5.1
Función de probabilidad (inclinación)

$$p(\phi_i) = \frac{1}{2\pi}$$

Figura 5.2
Función de probabilidad (azimuth)

$$L_o(\mathbf{x}, \omega_\mathbf{o}) \approx \sum_{i=1}^{N} \frac{2\pi L_i(\mathbf{x}, \omega_\mathbf{i}) f_r(\mathbf{x}, \omega_\mathbf{i}, \omega_\mathbf{o}) \cos\theta_i \sin\theta_i}{2 \sin\theta_i \cos\theta_i}$$

Figura 6
Ecuación de Render simplificada al usar Uniform Cosine Sampling

En los casos en los que el evento indicado por ruleta rusa es perfecto especular o refracción perfecta, al tratarse de un evento delta, la BRDF sólo se evalúa en una dirección. Es por esto que en estos casos no se realiza ningún cálculo salvo el muestreo del nuevo rayo, porque toda la luz obtenida en ese punto será la misma que la obtenida en el siguiente rebote.

El siguiente rebote del camino se genera de acuerdo al evento sucedido. Si el evento ha sido difuso o especular, el nuevo rayo se muestrea siguiendo la técnica Uniform Cosine Sampling. En el caso de perfecto especular el nuevo rayo es el producido de acuerdo a la ley de reflexión. En el caso de refracción perfecta el nuevo rayo es el producido de acuerdo a la ley de Snell. Los cálculos para la generación del nuevo rayo se realizan en coordenadas locales.

Cuando la condición de terminación del camino es no intersectar con un objeto o absorción debido a ruleta rusa, la luminancia obtenida (referente a luces de área) para ese camino es 0. Si la condición de terminación es haber encontrado un objeto emisor (luz de área), el producto acumulado a lo largo del camino se multiplica por la potencia de la luz.

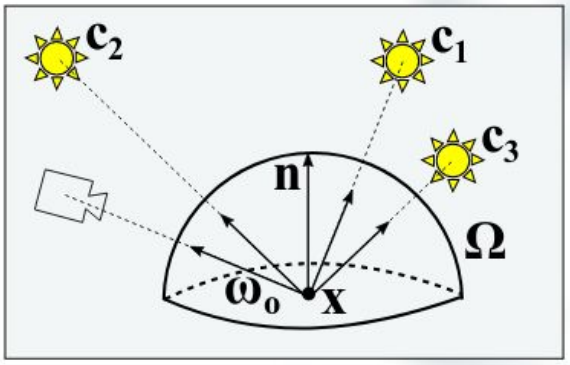
Como se traza un número dado de caminos por píxel (PPP), el color resultante en cada píxel consiste en la media de los obtenidos por cada uno de los caminos.

Next event estimation

Hasta ahora se ha visto cómo aproximar la ecuación de render para luces de área. Para las luces puntuales, se hace uso de la estimación de siguiente evento. Tras finalizar cada camino, la luz obtenida se puede simplemente sumar a la conseguida para luces de área gracias a la propiedad aditiva de la luz.

Para realizar la implementación de la técnica “Next event estimation” se ha hecho lo siguiente. En cada intersección (con un objeto no emisor) de un camino, si el evento seleccionado por ruleta rusa forma parte de Phong (difuso o especular), se calcula la contribución de las luces puntuales sobre el punto del espacio intersectado. Para ello se comprueba si el rayo entre el punto sobre la superficie del objeto y la luz puntual intersecta con algún objeto de la escena. Si existe esa intersección, entonces no se tendría en cuenta la contribución de esa luz puntual.

Esto se hace para cada uno de los puntos de luz de la escena. Si más de un punto de luz contribuye, los valores obtenidos se suman. Este resultado del sumatorio se correspondería con la luz directa sobre el punto intersectado (Figura 7). Para tener en cuenta también la luz indirecta, se lanza una muestra aleatoria. Esta muestra se corresponde con el siguiente rebote del camino.



$$L_o(\mathbf{x}, \omega_\mathbf{o}) = \sum_{i=1}^{n} \frac{p_i}{|\mathbf{c_i} - \mathbf{x}|^2} f_r\left(\mathbf{x}, \frac{\mathbf{c_i} - \mathbf{x}}{|\mathbf{c_i} - \mathbf{x}|}, \omega_\mathbf{o}\right) \left|\mathbf{n} \cdot \frac{\mathbf{c_i} - \mathbf{x}}{|\mathbf{c_i} - \mathbf{x}|}\right|$$

Figura 7
Contribución de luz directa de luces puntuales

Para conocer la contribución de las luces puntuales a lo largo del camino sobre la primera intersección (lo que se dibujará en el píxel finalmente), se hace uso de dos vectores. En el primero de ellos se almacena la luz directa que se ha mencionado anteriormente, y en el otro vector, un precálculo de la fórmula para la luz indirecta. Este segundo vector se utiliza ya que para la luz indirecta se tiene que tener en cuenta el valor de la BRDF del evento sucedido, pero se desconoce el valor que tomará la luz directa en la siguiente intersección.

Para el cálculo de la luz directa sobre el punto (o rayos de sombra) se evalúa la BRDF completa de Phong en el caso de que el evento de ruleta rusa haya difuso o especular.

En el caso del precálculo de esta componente indirecta (luz proveniente del rayo muestreado mediante Uniform Cosine Sampling), se utiliza la misma fórmula simplificada

que se usa para buscar la contribución de las luces de área (Monte Carlo con ruleta rusa), teniendo en cuenta únicamente la parte de la BRDF indicada por el evento.

Finalmente, cuando el camino ha acabado, se calcula la contribución de las luces puntuales de forma iterativa, desde la última intersección realizada hasta la primera. Este valor de radiancia obtenido se suma al obtenido por Monte Carlo (contribución de luces de área).

Convergencia

La velocidad del Path Tracer implementado depende de distintos factores a la hora de renderizar una imagen. Los factores que afectan son:

- El número de caminos por píxel (ppp). Esto afecta directamente al tiempo de ejecución de un render. Cuantos más caminos, menos ruido en la imagen, pero más tiempo de cálculo.

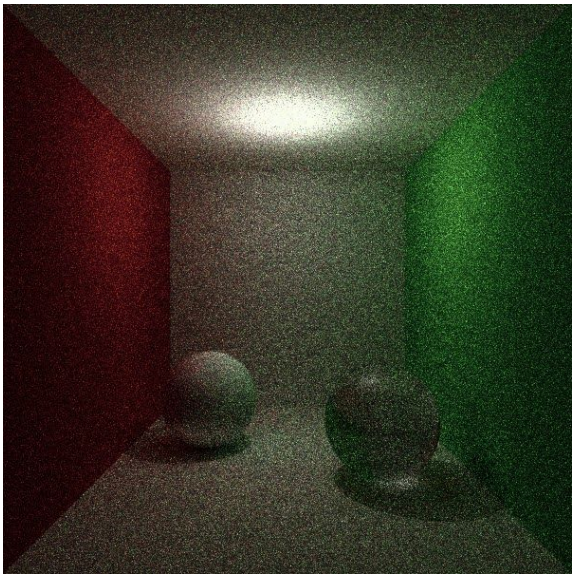


Figura 8.1
Cornell Box con 10 PPP
21 segundos

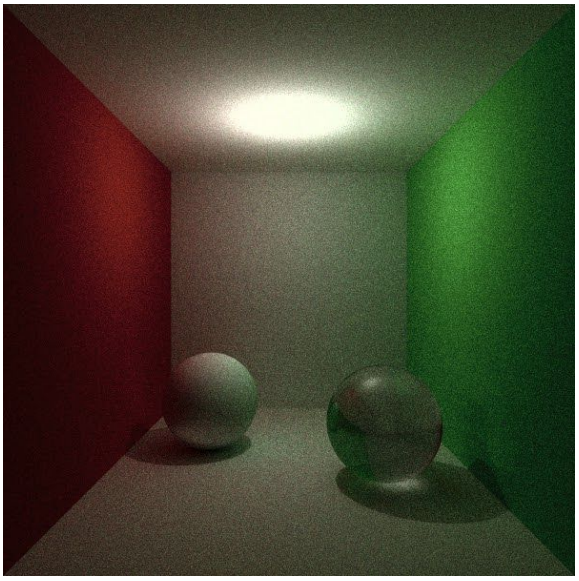


Figura 8.2
Cornell Box con 100 PPP
4 minutos y 24 segundos

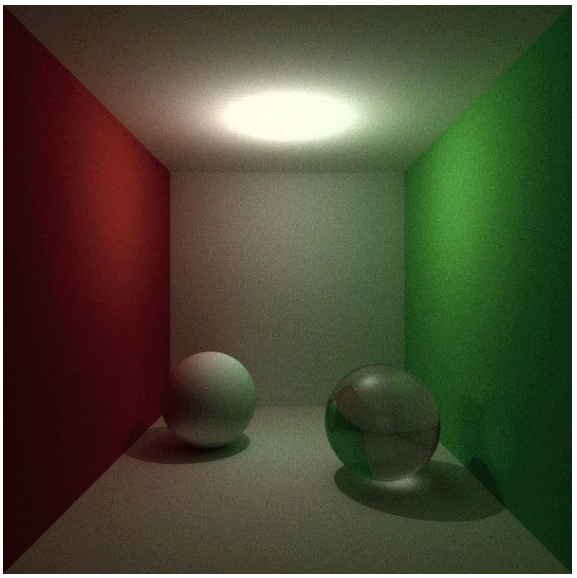


Figura 8.3
Cornell Box con 500 PPP
16 minutos y 42 segundos

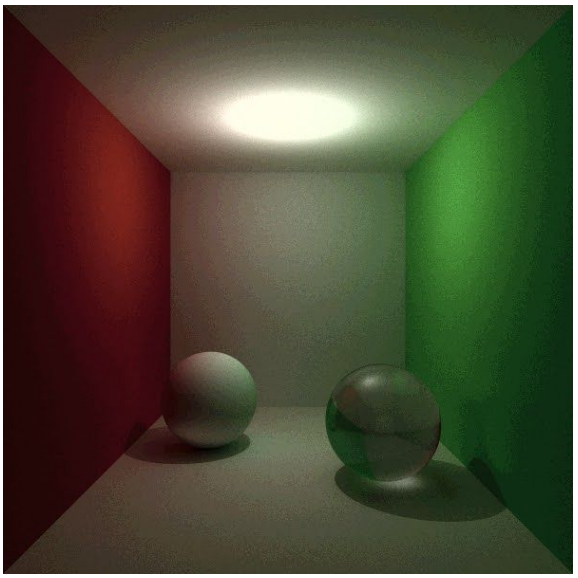


Figura 8.4
Cornell Box con 1000 PPP
34 minutos y 19 segundos

- La composición de objetos en la escena. Si es un espacio cerrado, se tardará el máximo tiempo posible, ya que ningún camino acabará por no intersectar con objetos. Por otro lado, cuanta más luz de área exista en una escena, mayor será la probabilidad de impactar contra ella, y por tanto finalizar el camino.

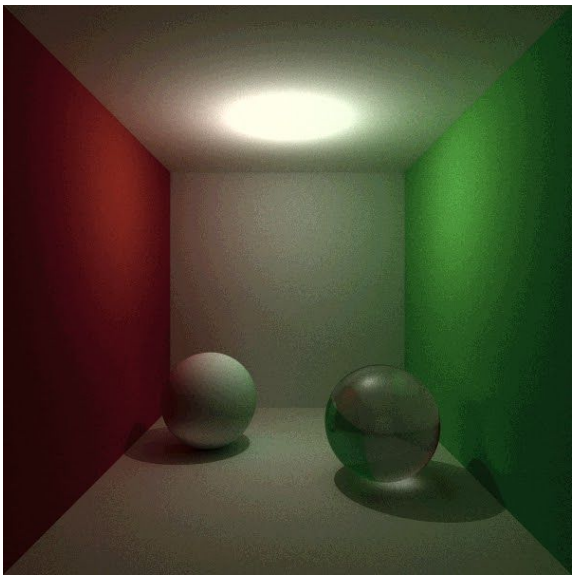


Figura 9.1
Cornell Box con 1000 PPP
y con luz puntual
34 minutos y 19 segundos

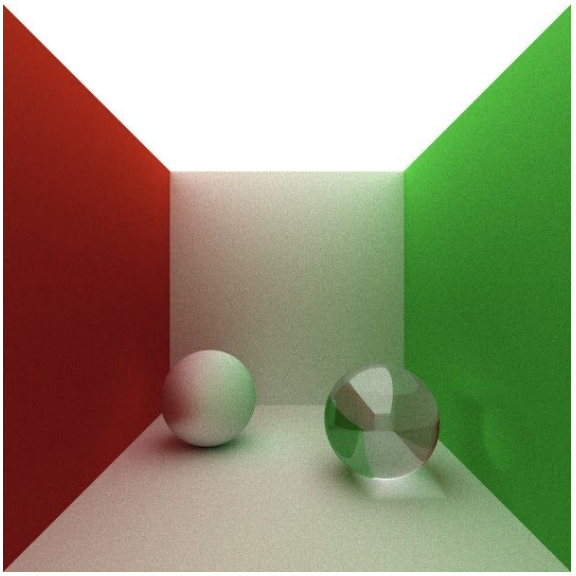


Figura 9.2
Cornell Box con 1000 PPP
y con luz de área
8 minutos y 53 segundos

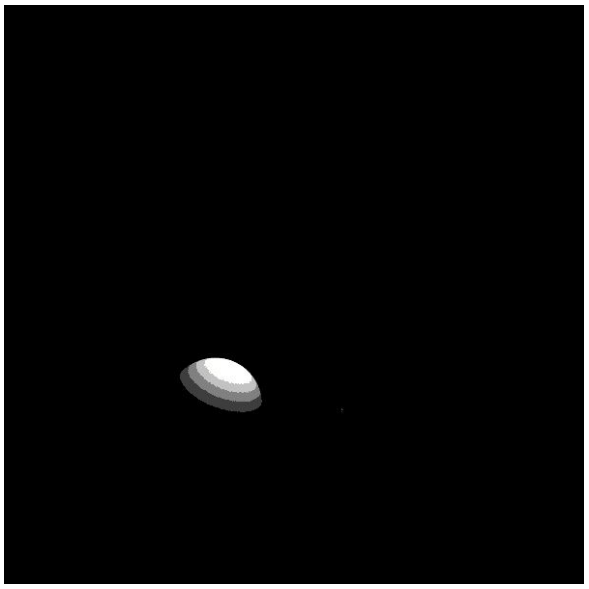


Figura 9.3
Cornell Box sin paredes con 1000 PPP
y con luz puntual
1 minuto y 39 segundos

- El número de objetos de la escena. Cuantos más objetos hay en la escena, más posibles intersecciones hay que calcular, por lo que el tiempo de cálculo aumenta de manera proporcional.

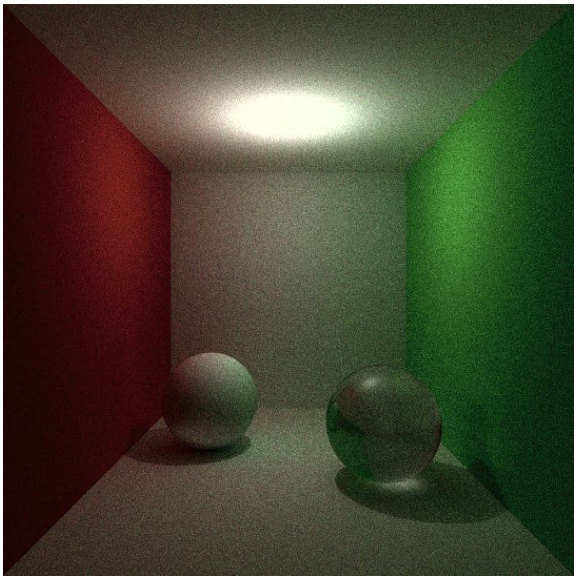


Figura 10.1
Cornell Box con 100 PPP
con dos esferas
4 minutos y 24 segundos

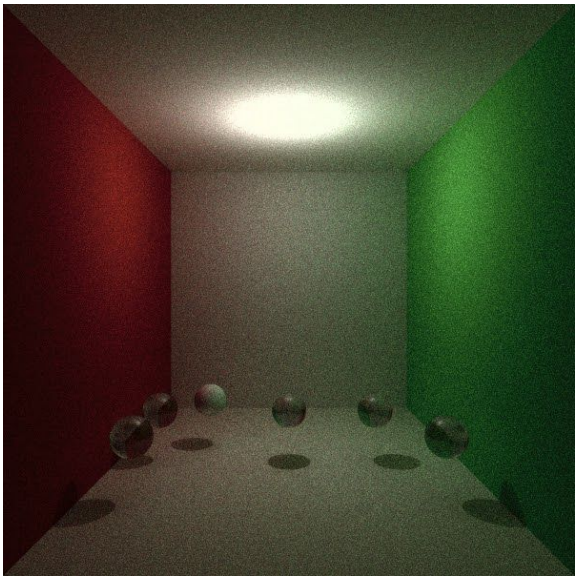


Figura 10.2
Cornell Box con 100 PPP
con seis esferas
7 minutos y 51 segundos

- El número de luces puntuales de la escena. Además del camino generado para encontrar luces de área, hay que tener en cuenta la contribución de las luces puntuales. En cada intersección (que ruleta rusa indique difusa o especular), hay

que iterar sobre todas las luces puntuales de la escena, y a su vez comprobar si el rayo trazado intersecciona con otros objetos, lo que hace que el tiempo de cálculo aumente.

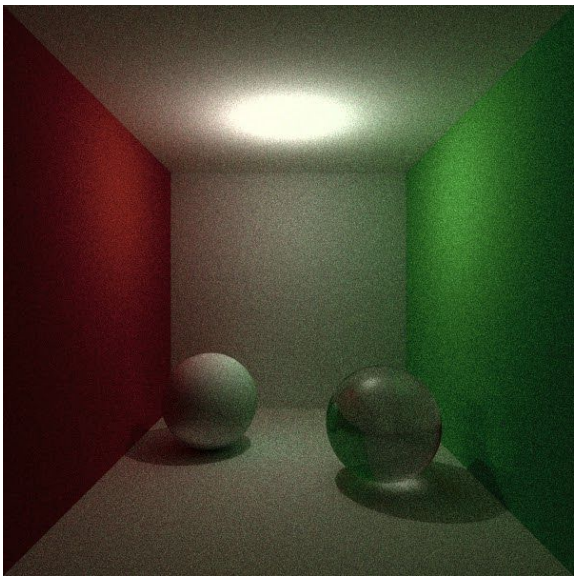


Figura 11.1
Cornell Box con 100 PPP
y una luz puntual
4 minutos y 24 segundos

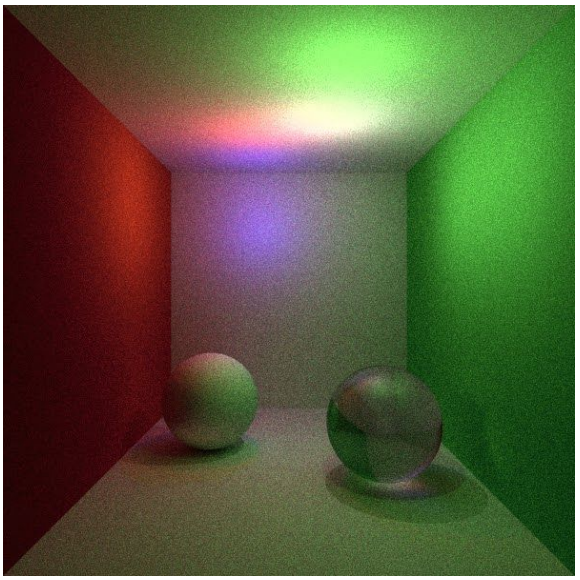
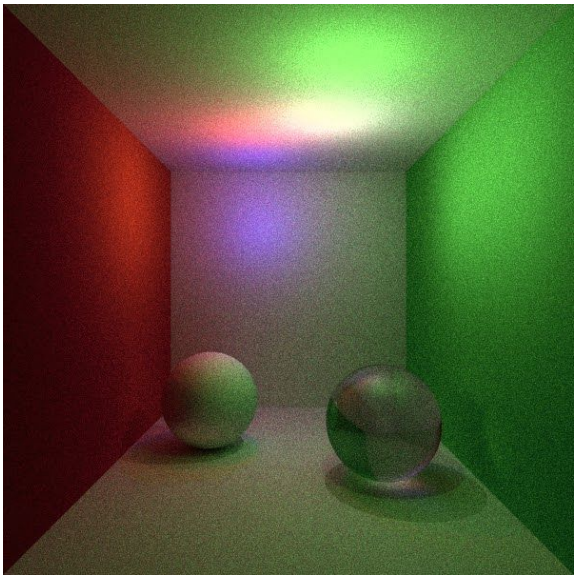


Figura 11.2
Cornell Box con 100 PPP
y cuatro luces puntuales
7 minutos y 26 segundos

- Los materiales de los objetos. Ya se ha mencionado que tener objetos que emitan en una escena reduce el tiempo de cálculo (ya que al impactar contra ellos el camino finaliza). Además de esto, hay determinados materiales que reducen el tiempo de cálculo. Esto sería el caso de los materiales con propiedad especular perfecta o refracción perfecta, ya que en estos casos no se realiza ningún cálculo iterativo sobre las luces puntuales. Claro está, que si una escena no tiene luces puntuales, esta condición del material de los objetos no afecta al tiempo de cálculo.



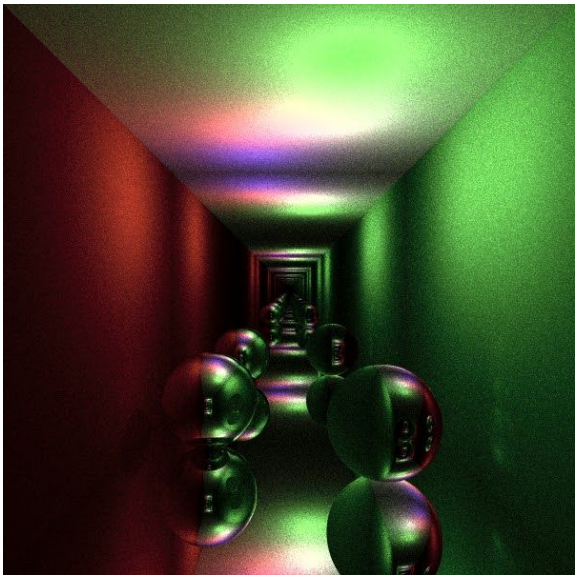


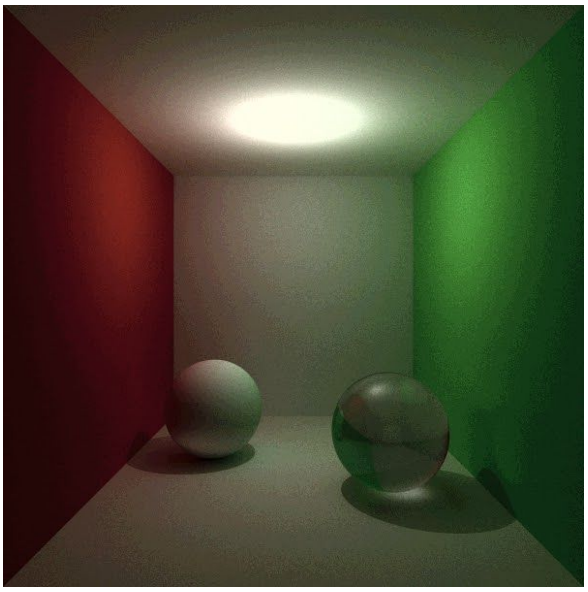
Figura 12.1
Cornell Box con 100 PPP
con cuatro luces puntuales
y mayoría de materiales difusos
7 minutos y 26 segundos

Figura 12.2
Cornell Box con 100 PPP
con cuatro luces puntuales
y mayoría de materiales delta
6 minutos y 10 segundos

Efectos de iluminación global

La iluminación global proporciona a una imagen efectos interesantes: sombras duras, sombras suaves, color bleeding y cáusticas. De estos cuatro efectos, son fácilmente obtenibles mediante path tracing:

- Sombras duras. (mostrar con imagen). Este efecto se puede conseguir fácilmente utilizando luces puntuales. Para ilustrarlo se ha empleado una escena con una luz puntual en el techo. En la imagen de la derecha se observa el efecto de sombra dura.



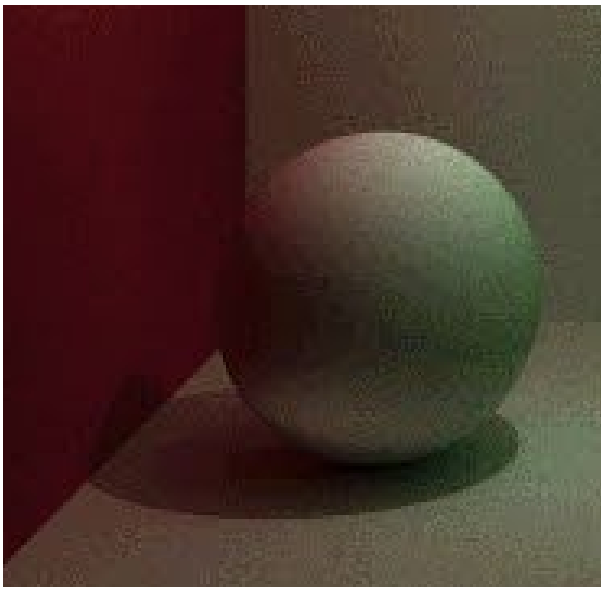


Figura 13.1
Cornell Box iluminada por luz puntual

Figura 13.2
Efecto de sombra dura ampliado

- Sombras suaves. Este efecto se puede obtener fácilmente utilizando fuentes de luz de área. Para ilustrarlo, se ha utilizado una escena en la que todo el techo es una luz de área. En la imagen ampliada, se observa que la sombra en el suelo generado por la esfera es suave, en contraposición a la sombra dura generada en el punto anterior al utilizar una luz puntual.

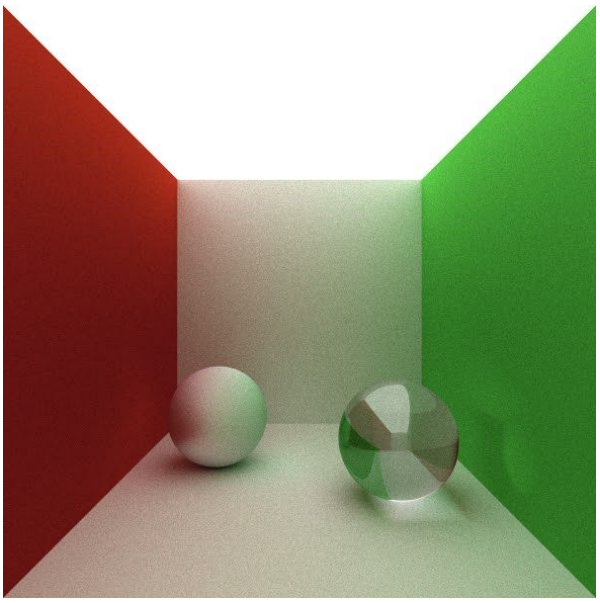


Figura 14.1
Cornell Box iluminada por luz de área

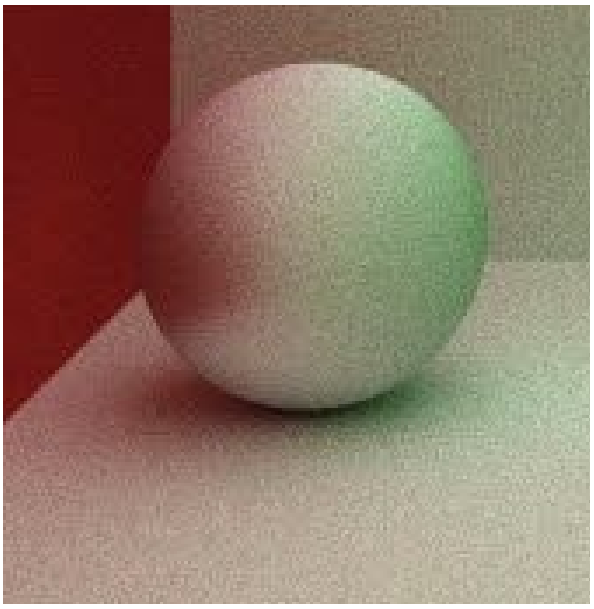


Figura 14.2
Efecto de sombra suave ampliado

- Color bleeding. (mostrar con imagen). El efecto color bleeding es muy fácil de generar (y visualizar) con una escena tipo cornell box. A continuación se muestran algunos ejemplos donde se observa el fenómeno de color bleeding como por ejemplo en la pared blanca del fondo de la escena donde se refleja color rojo y verde de las paredes laterales o en la esfera difusa localizada en la parte izquierda de la imagen.

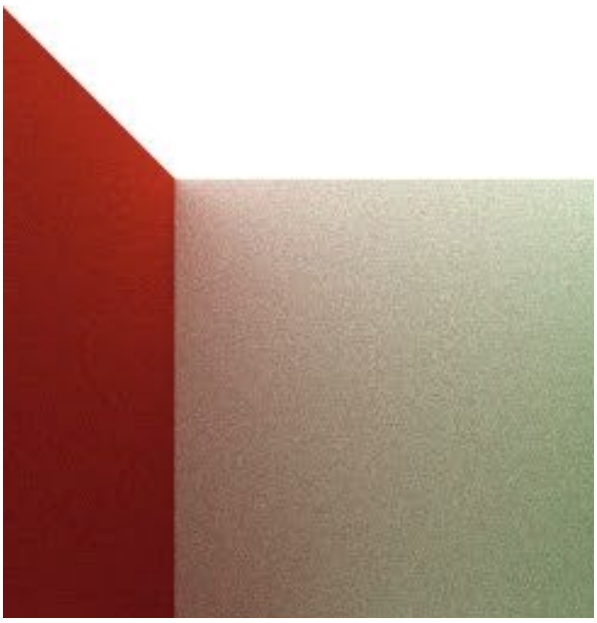


Figura 15.1
Color bleeding de la pared roja sobre la pared del fondo de la escena

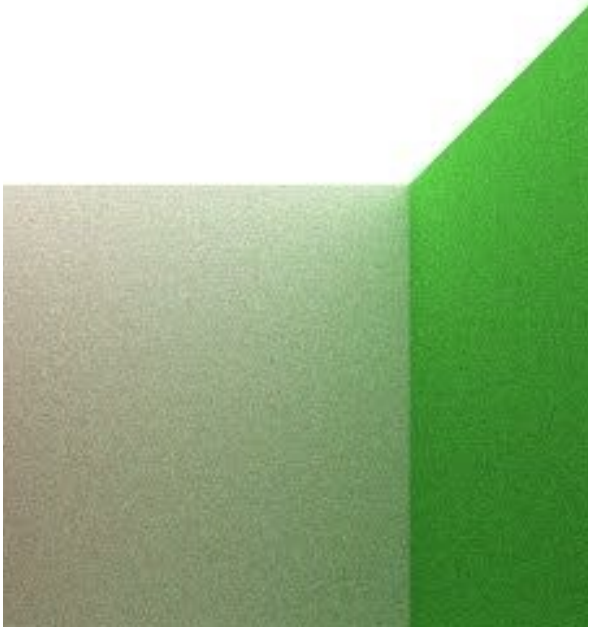


Figura 15.2
Color bleeding de la pared verde sobre la pared del fondo de la escena

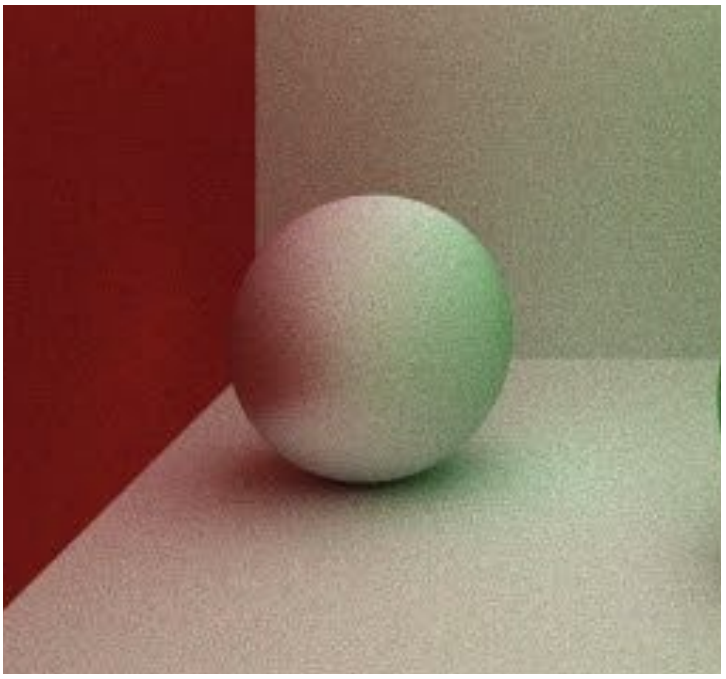


Figura 15.3
Color bleeding generado por las paredes laterales
sobre la esfera de Phong.

Sin embargo, las cáusticas son muy difíciles de conseguir, ya que son generadas por caminos de la luz muy específicos, y por tanto difíciles de encontrar utilizando un algoritmo de Path Tracing. Como se observa en la Figura 16, las cáusticas pueden generar formas muy definidas, y encontrar los caminos que lleven a esto requeriría muchas muestras difusas.

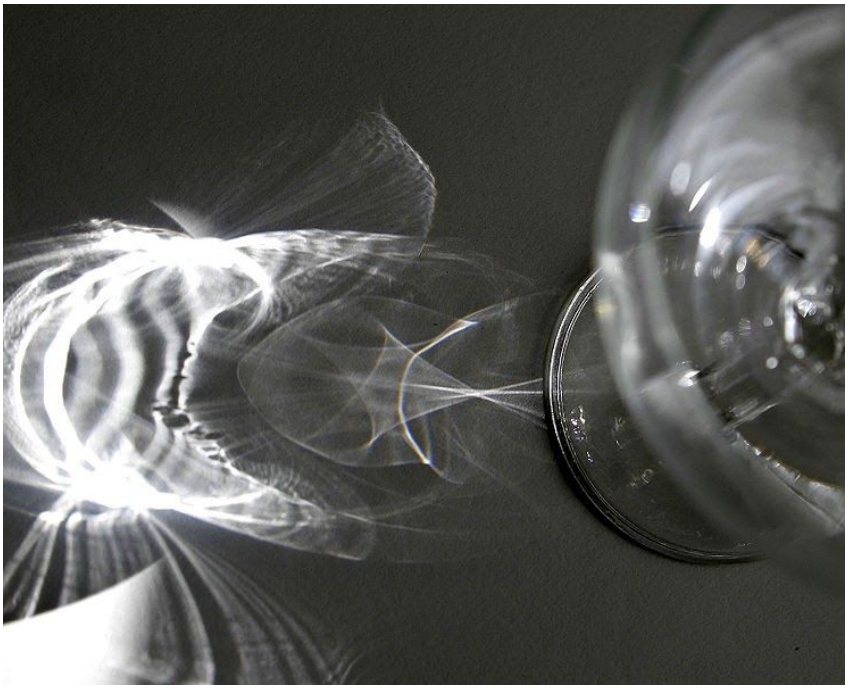


Figura 16

Ejemplo de cáusticas en la vida real

Con el Path Tracer implementado, es posible conseguir cáusticas (en escenas de luces de área), aunque se pierde mucha información. En la siguiente imagen ampliada se puede ver las cáusticas generadas sobre la pared y el suelo.

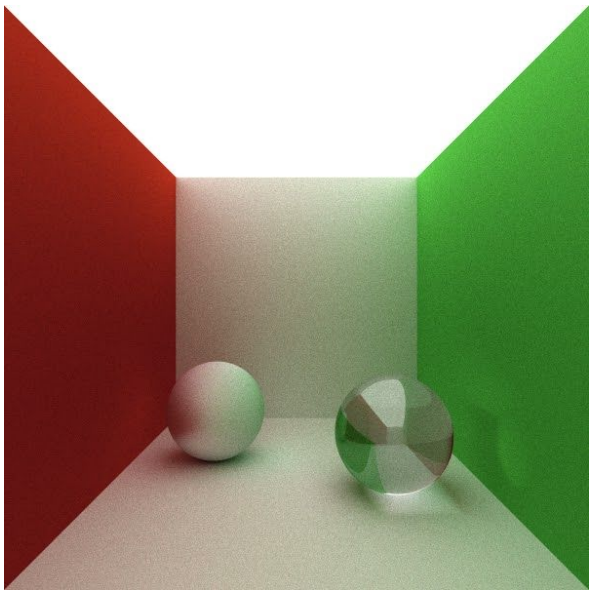


Figura 17.1
Cornell Box iluminada por luz de área

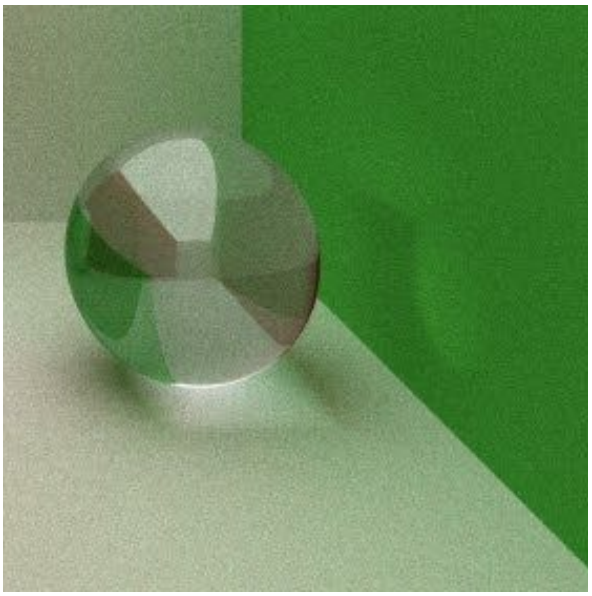


Figura 17.2
Efecto de cáusticas ampliado

Sin embargo, es posible conseguir el efecto de cáusticas con una luz puntual como se ve en la siguiente escena. Esto sucede porque, al reflejarse la luz sobre el techo, se simula un efecto de luz de área, lo que genera ligeras cáusticas.

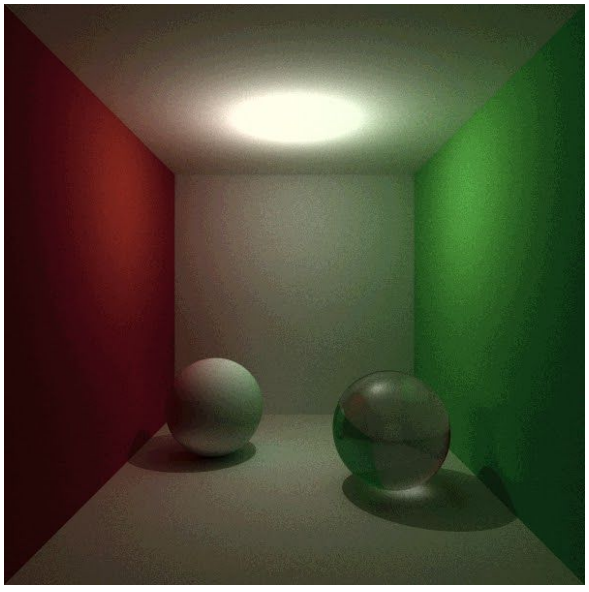


Figura 18.1
Cornell Box iluminada por luz puntual

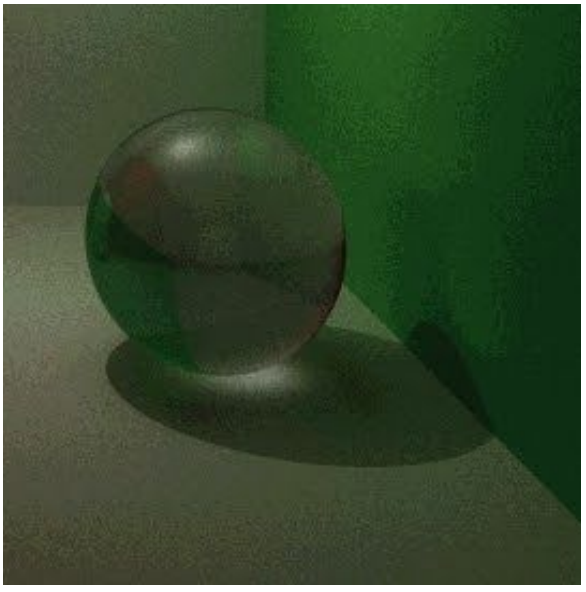


Figura 18.2
Efecto de cáusticas ampliado

Decisiones de diseño

Aunque ya se han comentado varias decisiones de diseño a lo largo de este documento, a continuación se incluyen otras no mencionadas hasta ahora.

- Para disponer de puntos y direcciones, se han implementado dos tipos de datos distintos, cada uno de ellos con 3 componentes (se ha omitido la componente homogénea). Junto con esto, también se han creado dos tipos de datos para representar matrices: de 3 y de 4 componentes, pensadas para direcciones y puntos respectivamente. Determinadas operaciones como la multiplicación entre matrices de 4x4 y puntos (3 componentes) han tenido que ser hardcodeadas. Esta división entre puntos y direcciones ha supuesto más tiempo de programación y algunas duplicidades en el código, pero ha permitido mayor control de errores debido a que se conocen posibles fallos en tiempo de compilación, reduciendo así el tiempo necesario para la depuración del código.

- Las propiedades de los materiales se han definido en tiempo de compilación (utilizando los constructores) como valores kd, ks, ksp, y krf, para las propiedades difusa, especular, perfecta especular y refracción. Para las k difusa y especular se ha incluido un valor k para cada canal RGB. También se ha introducido a través del constructor otros valores como shininess (propiedad especular) e índice de refracción (propiedad de refacción perfecta). Para este último se debe incluir en todos los objetos el valor 1, salvo que krf sea mayor que cero, caso en el que podría tomar otro valor.

- Para conocer las probabilidades de cada uno de los eventos en la ruleta rusa, se han utilizado las k de cada propiedad del material. En el caso de difuso y especular, en el que existe un k para cada valor RGB, se ha tomado como probabilidad la mayor de las tres k.

- Para que Monte Carlo combinado con ruleta rusa sea efectivo, tiene que existir siempre una probabilidad mayor que cero de que suceda el evento de absorción, sino no se estaría realizando una cadena de Markov. Actualmente no existe ningún control sobre las k introducidas en el constructor, por lo que perfectamente se podría introducir probabilidades mayores que 1 y cometer otro tipo de fallos. En este momento, es responsabilidad del creador de la escena asegurarse de que introduce valores físicamente correctos, y que siempre exista cierta probabilidad de evento de absorción. En todas las escenas realizadas, todos los objetos tienen una probabilidad de 10%, por lo que siempre las k de un objeto (tomando el valor máximo en el caso de difuso y especular) sumaban en total 0.9.

Extensiones

En este apartado del informe se describen las distintas extensiones que se ha realizado sobre el path tracer básico que se proponía en el enunciado del trabajo. Se han realizado dos extensiones: triángulos y texturas. Además se comenta el Tone Mapping desarrollado para corregir la luminancia de las imágenes.

Primitivas geométricas: triángulos

La primera mejora realizada sobre el path tracer básico propuesto ha sido la incorporación de una nueva primitiva geométrica: los triángulos. Para realizar esta extensión se ha tomado como referencia principalmente el material de la asignatura, tanto el tema de geometría (para la definición del tipo de dato), como el tema de ray tracing para el cálculo de las intersecciones rayo-triángulo.

Una de las decisiones de diseño tomadas ha sido guardar en el tipo de dato, además de la normal, dos aristas. Esto es útil para el posterior cálculo de las intersecciones rayo-triángulo, además de simplificar la creación de una base de coordenadas local cuando sea requerido en el algoritmo.

Disponer de triángulos frente a tener solo las primitivas de plano y esfera, resulta una ganancia de expresividad en las escenas, permitiendo crear objetos que antes no eran realizables. A pesar de haber incluido los triángulos como primitiva geométrica, no se ha realizado ninguna implementación que permita cargar mallas de triángulos a partir de un fichero. Se ha tomado esta decisión ya que incluir mallas de triángulos está relacionado con cargar objetos complejos (con un gran número de triángulos), lo cual implica indirectamente realizar alguna mejora opcional relacionada con estructuras de aceleración. Debido a la limitación del tiempo disponible, se ha preferido no seguir por ese camino, y utilizar los triángulos para generar figuras simples, aportando riqueza visual a la imagen por otras vías.

Se han tomado más decisiones de diseño sobre los triángulos, pero están más relacionadas con la segunda mejora opcional, que se explica a continuación.

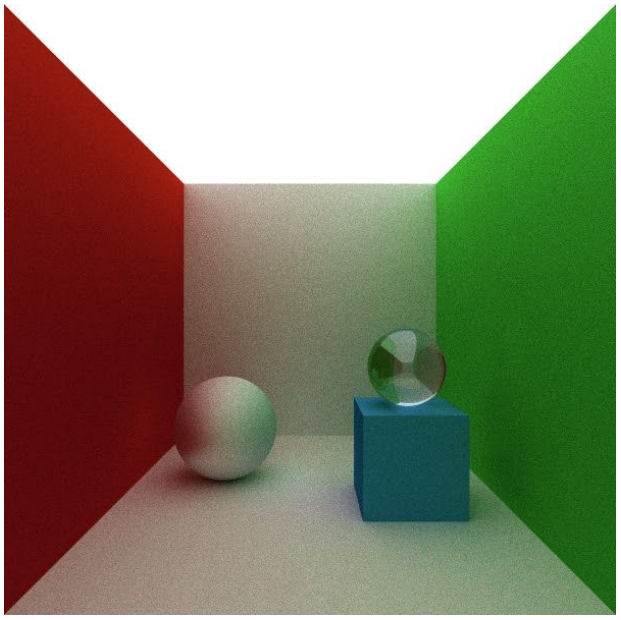


Figura 16
Prisma construido mediante triángulos

Texturas

La segunda extensión realizada se trata de texturas. Esta mejora se ha realizado únicamente sobre la primitiva geométrica del triángulo. Para el diseño e implementación se ha utilizado como material las diapositivas de la asignaturas relativas a texturas, así como otro material externo de diversas fuentes, citado en la sección de referencias del documento.

Para la introducción de texturas en un render, se carga la imagen de la textura al comienzo de la ejecución del programa y se referencia mediante un puntero en los triángulos que se desee. En el constructor de estos triángulos con textura, también se incluyen coordenadas s y t para cada uno de los vértices del triángulo. De esta manera, se conoce que punto de la textura se corresponde con cada uno de los vértices.

En tiempo de ejecución, cada vez que se intersecta un triángulo con textura, es necesario calcular sus coordenadas baricéntricas, para así obtener las coordenadas de la imagen de textura (y por tanto el píxel concreto con su valor RGB) a las que corresponde el punto intersectado del triángulo. A partir de este valor RGB obtenido, se calculan las k difusa y especular, ya que en el constructor de un triángulo con textura no se pueden incluir como tal, ya que a priori no se conoce el valor del color. Es por esto que en el constructor se introducen los valores máximos para estas dos k, y en tiempo de ejecución las k correspondientes a cada canal se calculan en base al RGB obtenido, tomando como referencia el valor máximo de la kd y ks.

Para ilustrar esta mejora opcional, se incluyen varios renders en los que se ha hecho uso de texturas.

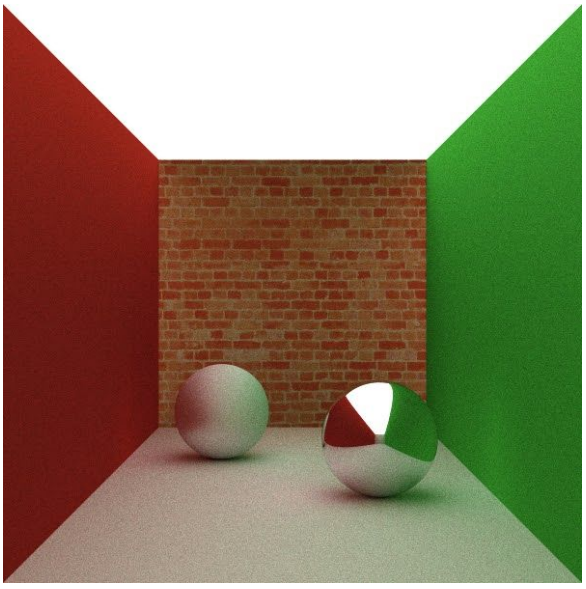


Figura 19.1
Cornell Box con texturas

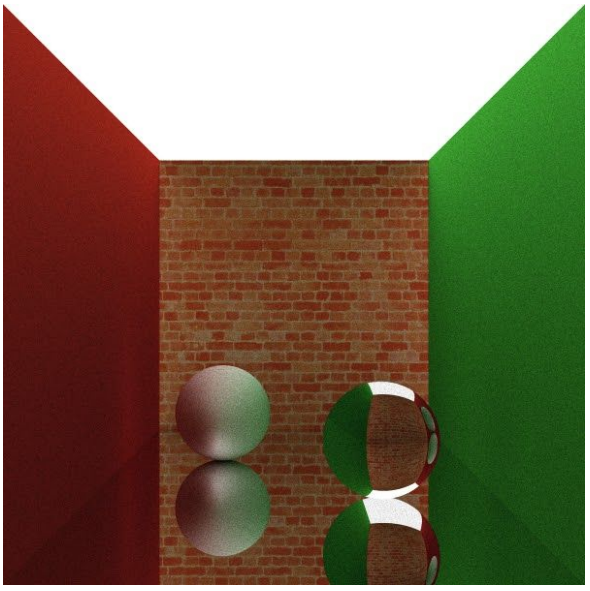


Figura 19.2
Cornell Box con texturas

Tone Mapping

Como extensión adicional se ha llevado a cabo la implementación de un tone mapper (mapeador tonal) con el fin de poder enriquecer el número de tonos de una imagen de modo que se pueda ver al mismo tiempo los tonos medios de la imagen, así como los tonos más oscuros y los más luminosos, permitiendo así reducir el contraste global de la imagen, que se mitiga ante tanta variedad de tonos.

El tone mapper implementado está preparado para poder trabajar con imágenes escritas en formato ppm (Portable PixMap). Las operaciones que es capaz de implementar el tone mapper diseñado son las siguientes:

- **Clampeo:** descartar (transformar a blanco) todos los valores superiores a 255 (1 en precisión de coma flotante).
- **Ecualización:** transformación lineal de valores desde el mínimo al máximo (normalización).
- **Ecualización y clampeo:** combinación de ambos operadores anteriores en base a un parámetro de clampeo.
- **Curva gamma:** aplicar una curva gamma para todos los valores. Requiere ecualizar la imagen de forma previa.
- **Clampeo con curva gamma:** aplicar una curva gamma después de clampear los valores de la imagen. Este proceso requiere previamente ecualización.

Todas la operaciones se realizan únicamente sobre la luminancia del píxel. Para esto el color se transforma de RGB a formato LAB, y las operaciones actúan sobre el canal L (luminosidad). Posteriormente, el color se convierte de nuevo a RGB y se escribe en la imagen resultante.

Separar el proceso de tone mapping del Path Tracer es muy útil, ya que al devolver una imagen HDR, luego se puede realizar mapeado de tonos todas las veces que se desee hasta encontrar la combinación de parámetro perfecta. De otro modo, se podrían echar a perder horas de renderizado únicamente por haber configurado el tone mapping de forma inadecuada. Todas las imágenes presentes en este informe han recibido tone mapping, en concreto utilizando ecualización y clampeo a nivel variable dependiendo de la escena.

Conclusiones

Tras realizar el desarrollo del Path Tracer se han llegado a diversas conclusiones. Una de ellas es que Path Tracing es un algoritmo muy robusto que permite generar renders con un gran realismo mediante un único parámetro de configuración, el número de caminos por píxel. Sin embargo, Path Tracing tiene problemas para representar efectos como las cáusticas, ya que al trazar rayos desde la cámara, los caminos que generan este tipo de

iluminación son muy difíciles de encontrar. Para simular estos efectos es mejor utilizar otros métodos como Photon Mapping.

De todas formas, Path Tracing es una buena técnica para aproximar la ecuación de render, y al tener un único parámetro (PPP), se puede controlar fácilmente la cantidad de ruido, y por tanto el tiempo de cálculo, que con la que se desea que se genere el render.

Organización del trabajo

El desarrollo del Path Tracer se ha llevado a cabo en todo momento con el trabajo simultáneo de los dos miembros del equipo. Se ha hecho con el fin de equiparar las horas trabajadas y, más importante, que todas las decisiones de diseño e implementación tuvieran el respaldo de ambos miembros.

En la siguiente tabla, se muestra el número de horas dedicadas a cada una de las fases del proyecto.

Tarea realizada	**Horas dedicadas**	
	Víctor Peñasco	**Rubén Rodríguez**
Puntos, direcciones, matrices y operaciones	20	20
Tone mapper	15	15
Comprensión y diseño del algoritmo de Path Tracing	7	7
Implementación del algoritmo	25	25
Triángulos	5	5
Texturas	5	5
Depuración y pruebas	50	50
Memoria (incluyendo renders finales)	20	20
Total	**147**	**147**

Renders

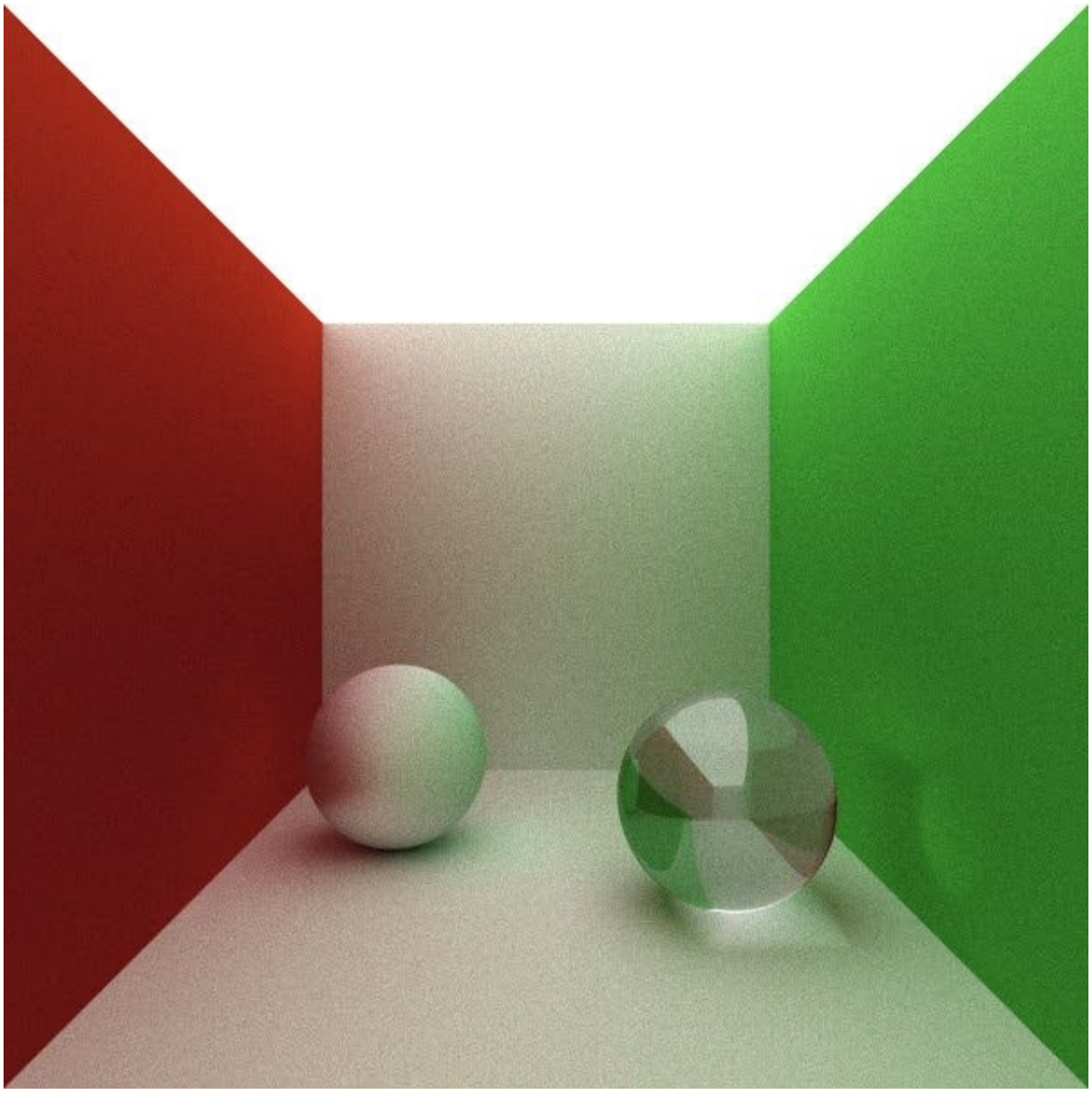


Figura 20
Cornell Box con luz de área
1000 PPP
8 minutos y 53 segundos

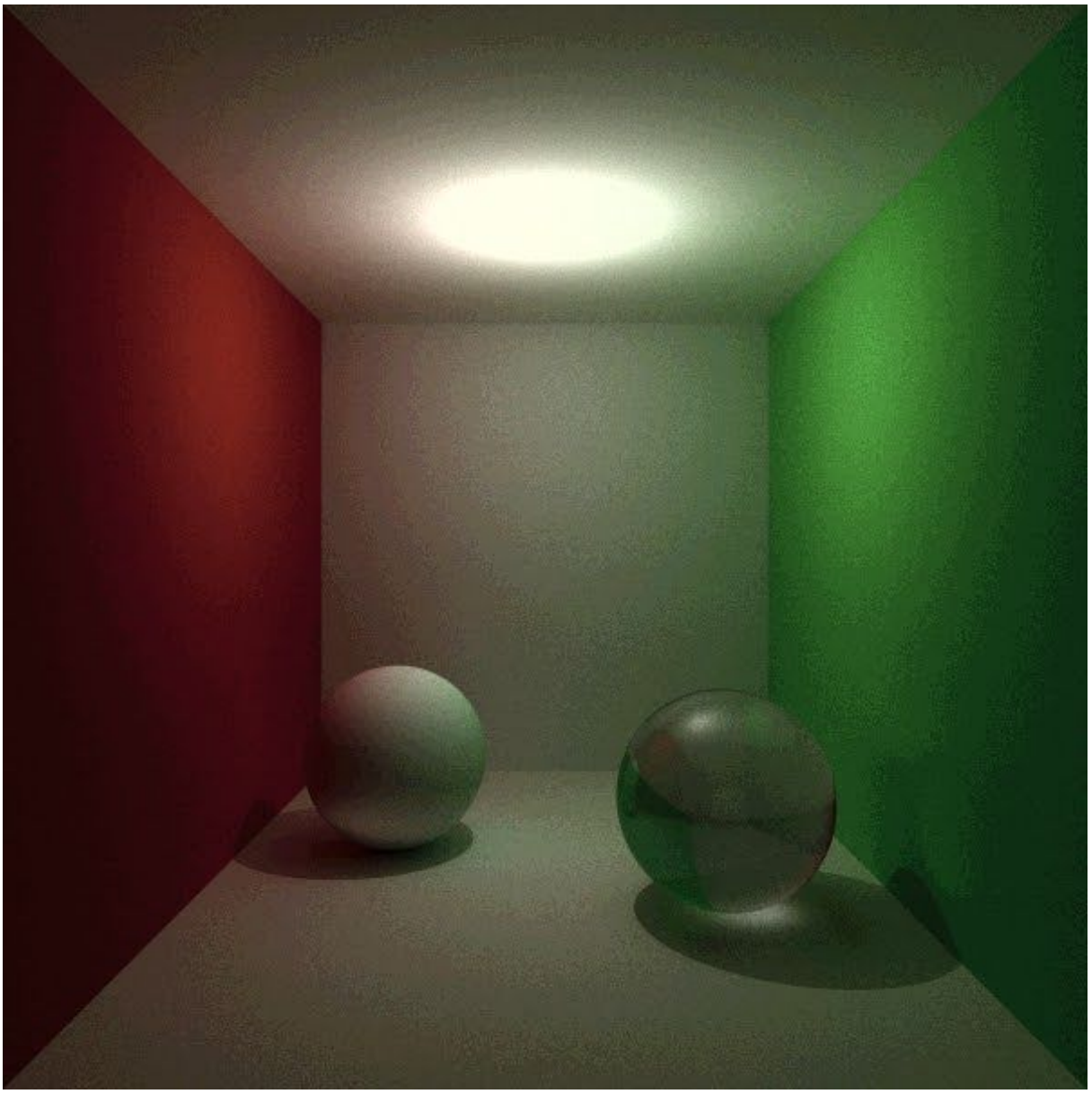


Figura 21
Cornell Box con luz puntual
1000 PPP
34 minutos y 19 segundos

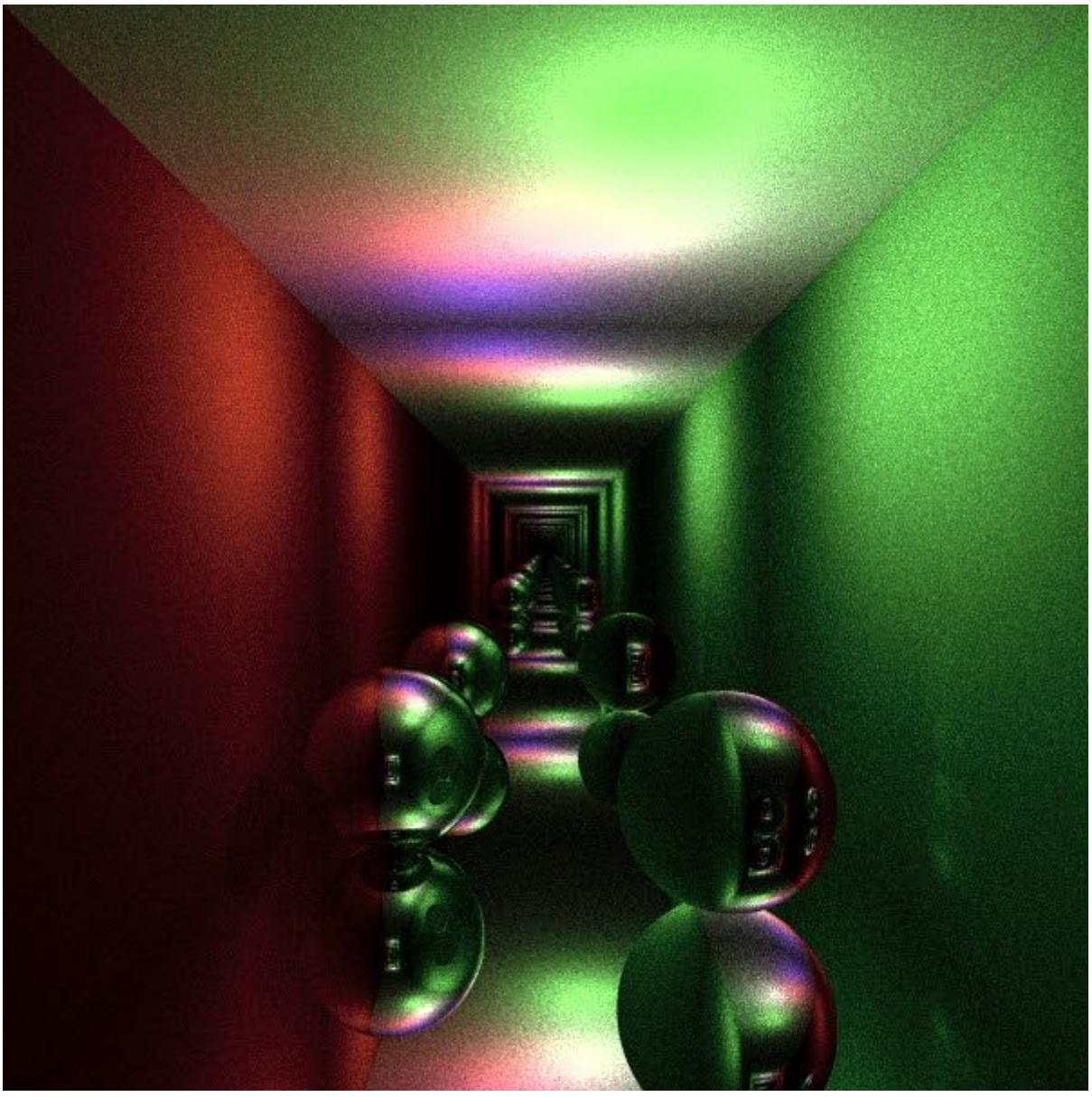


Figura 22
Cornell Box con cuatro luces puntuales y varios planes perfecto especulares
100 PPP
6 minutos y 10 segundos

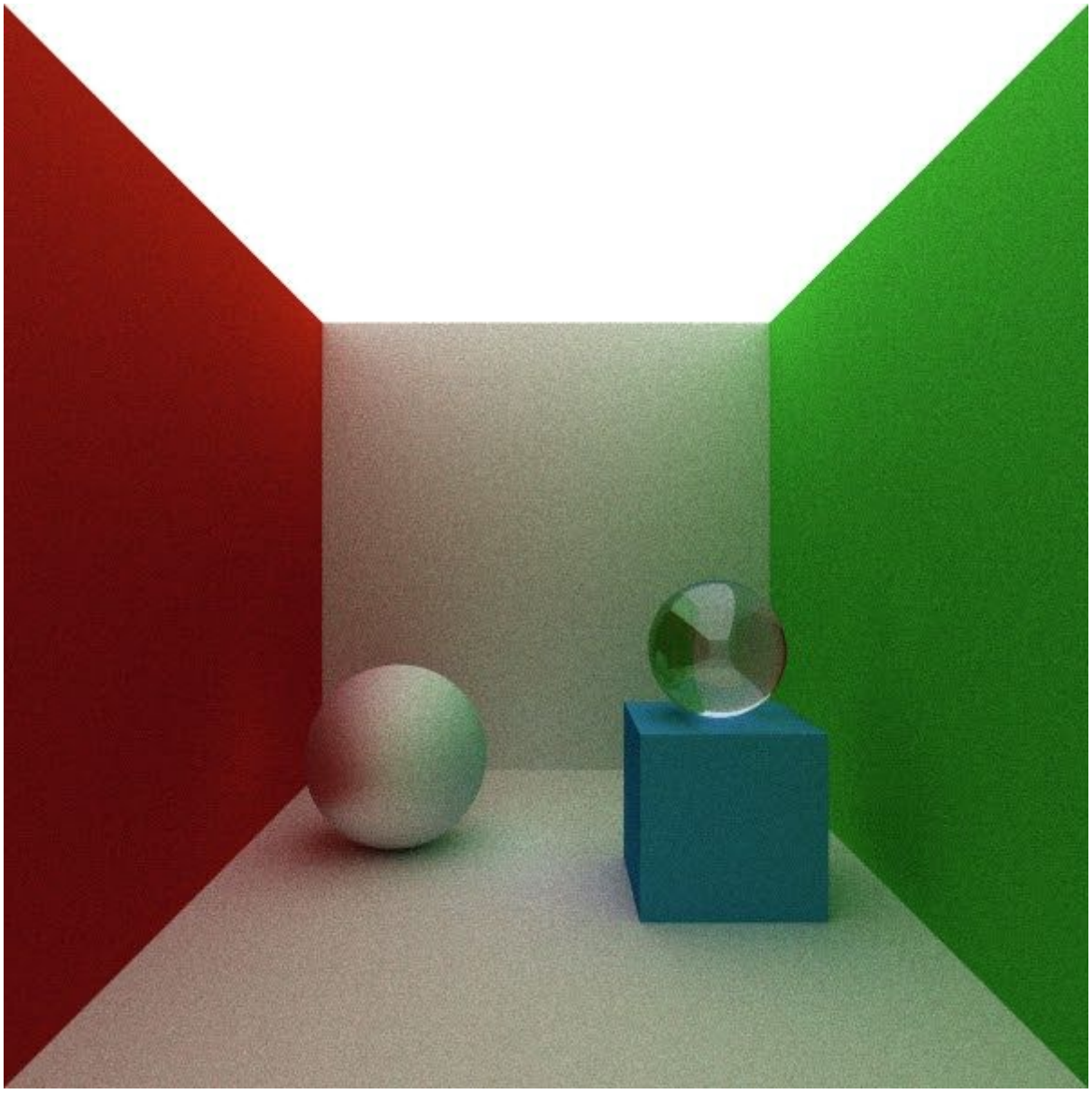


Figura 23
Cornell Box con luz de área y prisma formado con triángulos
500 PPP
10 minutos y 30 segundos

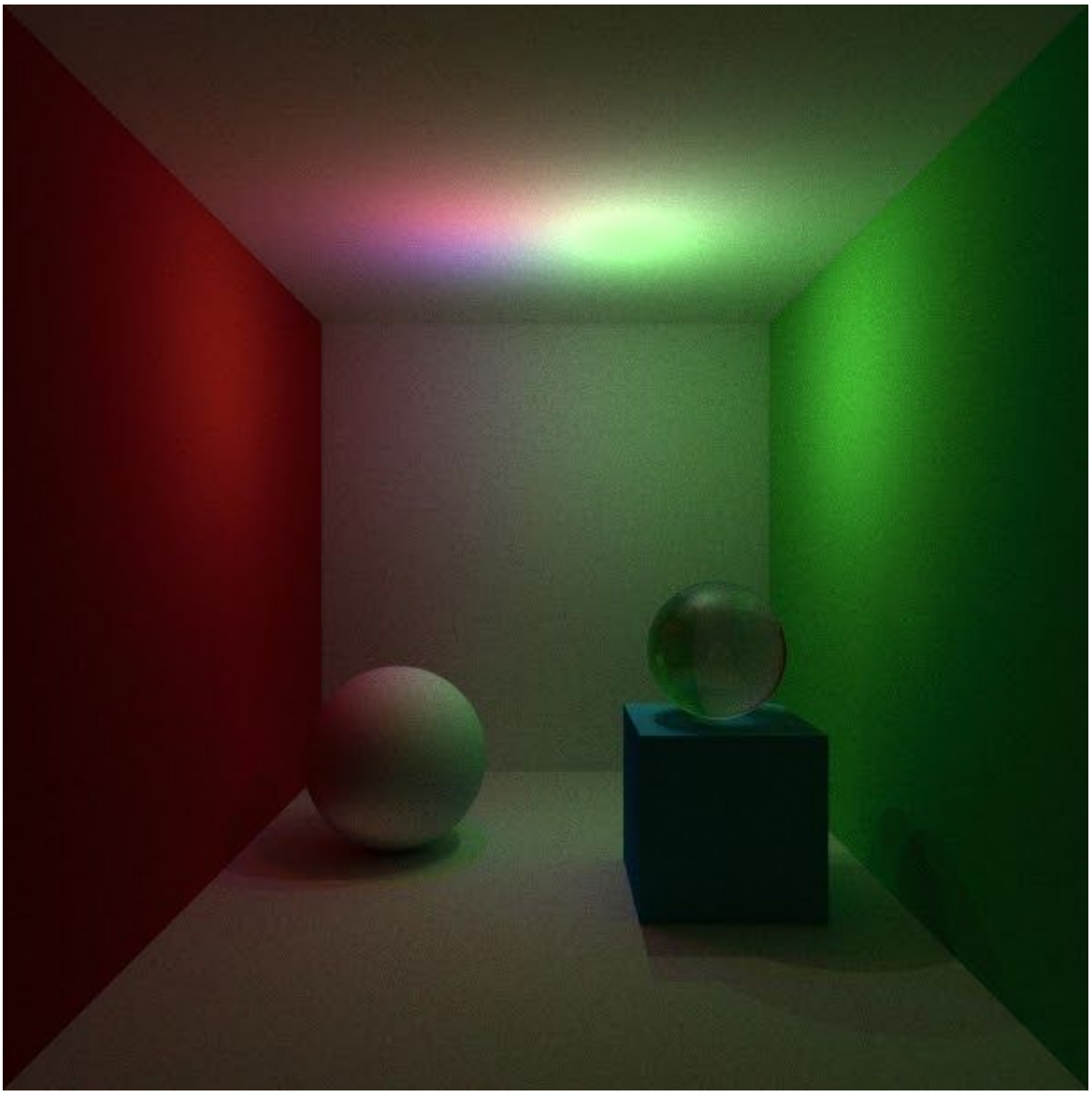


Figura 24
Cornell Box con luces puntuales y prisma formado con triángulos
500 PPP
91 minutos y 32 segundos

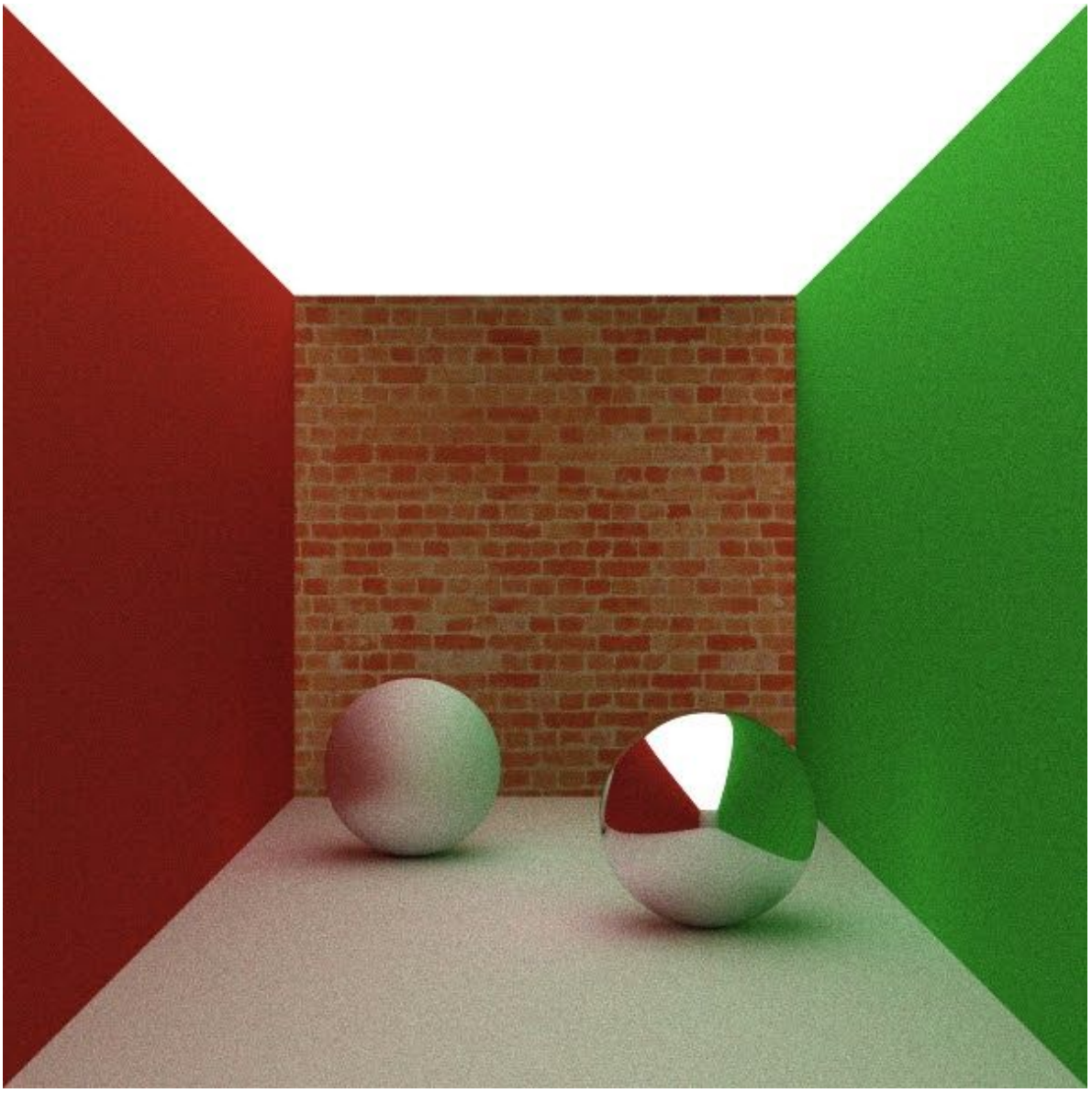


Figura 25
Cornell Box con luz de área y triángulos con textura
500 PPP
8 minutos y 50 segundos

Referencias

[1] "Lighting - Informática gráfica"
https://moodle.unizar.es/add/pluginfile.php/2303601/course/section/356530/03-lighting-handout.pdf.

[2] "Geometry - Informática gráfica"
https://moodle.unizar.es/add/pluginfile.php/2303601/course/section/356530/02-geometry-handout.pdf.

[3] "Ray Tracing - Informática gráfica"
https://moodle.unizar.es/add/pluginfile.php/2303601/course/section/356530/06-raytracing.pdf?time=1572262300716..

[4] "Texturas - Informática gráfica"
https://moodle.unizar.es/add/pluginfile.php/2303601/course/section/356530/09-texturas.pdf?time=1573038366755.

[5] “Intersections between rays and spheres”
http://viclw17.github.io/2018/07/16/raytracing-ray-sphere-intersection/

[6] “Barycentric coordinates calculation”
https://gamedev.stackexchange.com/questions/23743/whats-the-most-efficient-way-to-find-barycentric-coordinates