

# Práctica 2: Trazador de rayos clásico

---

*Novice programmers often neglect the design phase, instead diving into coding without giving thought to the evolution of a piece of software over time. The result is a haphazard, poorly modularized code which is difficult to maintain and modify. A few minutes of planning short-term and long-term goals at the beginning is time well spent.*

-- Paul Heckbert, "Writing a Ray Tracer," in An Introduction to Ray Tracing, edited by Andrew Glassner

## Introducción

En esta segunda práctica<sup>1</sup>, vamos a explorar las posibilidades de los métodos de iluminación global y para ello codificaremos un trazador de rayos cuya estructura de clases debe permitir añadir funcionalidades con cierta facilidad.

El trazado de rayos es un algoritmo de iluminación global muy potente capaz de crear imágenes tridimensionales de un alto nivel de complejidad y realismo.

Puesto que la pretensión de esta práctica es ayudarnos a familiarizarnos con los aspectos fundamentales, las imágenes resultantes pueden no resultar demasiado vistosas, sin embargo el trabajo que aquí se realice puede servir de base para crear efectos muy interesantes que se proponen en forma de partes opcionales.

El lenguaje en el que se propone el desarrollo de la práctica es C, ya que ha sido el lenguaje que en este curso se ha visto durante el primer cuatrimestre. Para facilitar la tarea a los alumnos y centrar los esfuerzos en los algoritmos de traza de rayos, se proporciona un conjunto mínimo de funciones para el parseo de los ficheros con la descripción de las escenas.

Los alumnos tienen la libertad de poder cambiar la estructura de este código si no encaja con sus necesidades, y de hecho se anima que partan de cero, si bien es importante que el trazador de rayos a implementar sea capaz de leer las escenas en el formato XML propuesto. Es posible elegir otro lenguaje de programación para realizar el trabajo, siempre que sea totalmente compatible con las especificaciones y ficheros de descripción de escena de la práctica, esto es, tendrán que escribir un *parser* que acepte la sintaxis de los ficheros de entrada.

El trazador de rayos que se implemente ha de tener las siguientes características:

- Algoritmo:
  - Trazador de rayos básico
  - Supermuestreo en el lanzamiento de rayos para reducir los bordes dentados
  - Sombras
  - Reflexiones
- Geometría:
  - Esferas

---

<sup>1</sup> Esta práctica está adaptada de la práctica 3 de las asignatura 15-462 de la Carnegie Mellon, y es un concepto original del equipo docente de Alexei Efros.

- Triángulos
- Modelos externos
- Materiales
  - Lambertianos (mate)
  - Phong
  - Mapeado de texturas básico (sin filtrado)
- Luces:
  - Luces puntuales

## Índice

|   |   |
|---|---|
| Introducción .....                          | 1 |
| Índice .....                                | 2 |
| Normativa del laboratorio .....             | 2 |
| Requisitos mínimos .....                    | 3 |
| Competición y Galería de imágenes .....     | 4 |
| Código inicial.....                         | 4 |
| Lectura recomendada.....                    | 5 |
| Partes opcionales.....                      | 5 |
| Cilindros .....                             | 6 |
| Refracción .....                            | 7 |
| Filtrado de texturas .....                  | 7 |
| Estructuras espaciales de aceleración ..... | 7 |
| Mallas normales interpoladas .....          | 7 |
| Sombras suaves .....                        | 7 |
| Reflexiones borrosas .....                  | 7 |
| Campo de profundidad.....                   | 7 |
| Muestreo circular o elíptico .....          | 7 |
| Estimadores combinados .....                | 8 |
| Iluminación indirecta .....                 | 8 |
| Otros .....                                 | 8 |
| Recomendaciones.....                        | 8 |
| Forma y fecha de entrega.....               | 9 |
| Nota aclaratoria.....                       | 9 |

## Normativa del laboratorio

- El criterio más importante es la funcionalidad: un programa que funciona siempre tiene más posibilidades de llevarse una buena puntuación; no se valorarán aquellos programas que no funcionen. Una práctica o proyecto modesto será evaluada mucho más favorablemente que un "proyecto" ambicioso que sólo da *core-dumps*. Los siguientes criterios que se tendrán en cuenta (y que hay que cuidar al realizar las prácticas) son:

- La manera de resolver el problema con el programa
- Estructuras de datos y diseño de los algoritmos
- Claridad y documentación en el código

- Eficiencia y elegancia en la implementación.
- ¡Por favor, no hagáis trampas! Se procura alentar el diálogo y el trabajo en equipo, pero por favor trabajad de forma independiente (a menos que el trabajo sea en grupos). Trabajos muy similares serán considerados como copias, a menos que la naturaleza lo pedido sea tan restrictiva que justifique las similitudes. Y una copia implica el suspenso automático. Simplemente piénsalo de esta manera: hacer trampas dificulta el aprendizaje y la diversión de conseguir hacerlo. Es vuestra responsabilidad proteger vuestro trabajo y asegurarnos que no se convierte en el de otro.
- Si se utiliza (o mejora) código fuente u otro material obtenido a través de internet, la biblioteca... debe darse el crédito a los autores y pedir permiso de ser necesario (si tiene una licencia restrictiva). Tomar código de un libro o de internet sin tener en cuenta estas consideraciones será considerado copia.
- Está terminantemente prohibido la práctica de técnicas de *overclocking* en las tarjetas gráficas del laboratorio, así como desbloquear los procesadores de vértices y fragmentos de los chips gráficos. Este tipo de acciones pueden dañar físicamente el equipo del laboratorio y los alumnos responsables serán amonestados severamente.

## Requisitos mínimos

- 1) Implementar un algoritmo básico de traza de rayos que muestree el plano focal de la “cámara virtual” y genere la imagen sintética. El algoritmo debe incluir sombras y reflexiones, aunque no es necesario que simule refracción.
- 2) Se ha implementar la función `trace_pixel (Scene * scene, int x, int y)` en el fichero `raytrace_RT.cpp` de modo que devuelva el color correspondiente a la coordenada de pantalla (x,y). Las coordenadas de pantalla comienzan en la esquina inferior izquierda (0,0) y la recorren hasta llegar a la esquina superior derecha en (WINDOW\_WIDTH-1, WINDOW\_HEIGHT-1)  
Lógicamente el código requerirá de métodos para:
- 3) Generar rayos desde la cámara
- 4) Tests correspondientes a las intersecciones con los elementos de la escena.
- 5) Interacción luz-material (al menos Lambert y Phong) para el cálculo del sombreado, la implementación debería seguir la ecuación mostrada en la asignatura de Informática Gráfica para cada uno de los tipos de materiales. Lambert y Phong no implementan el término de reflexión.  
CUIDADO: En el caso de un trazador de rayos no hay componente ambiental en la ecuación de iluminación.
- 6) Añadir el mapeado de textura al resultado de la interacción anterior
- 7) Luces puntuales sin atenuación por distancia.
- 8) Un mecanismo de *antialiasing* sencillo, mediante la toma de varias muestras por pixel (en la clase hay escrito un pseudocódigo con el que los alumnos se pueden guiar).

No es necesario preocuparse por un fichero de escena mal descrito. Por ejemplo, la cámara nunca estará dentro de una esfera. Tampoco es necesario preocuparse por los rayos que inciden en la parte posterior de las superficies.

El código inicial permite cambiar la posición de la cámara dinámicamente en el espacio 3D, utilizando el ratón y la tecla CTRL y hacer una previsualización de la escena en OpenGL, así pues es necesario hacer las comprobaciones pertinentes a la cámara y la perspectiva en cada frame renderizado.

Se ha de dar soporte al menos a tres tipos de objetos geométricos.

- 1) Esferas: definidas por el radio y su material. El centro de la esfera (0,0,0) está situado en el centro de su propio sistema de coordenadas objeto.
- 2) Triángulos: definidos por las 3 coordenadas de sus vértices, los vectores normales, el material y la coordenada 2D para el mapeado de texturas en cada uno de los vértices.
- 3) Modelos externos: compuestos de múltiples objetos triangulares. El código proporcionado ofrece una interfaz para leer ficheros de mallas sofisticados.

Para cada tipo de objeto, `Scene.h` define los parámetros de transformación de sólido rígido, esto es: la escala, el ángulo con el que esté rotado y la traslación.

Así el programa resultante tiene al menos que:

- 1) Trazar rayos y dibujar correctamente esferas, triángulos y modelos externos
- 2) Llevar a cabo correctamente el sombreado Phong y Lambertiano, con todas las componentes necesarias y la interpolación de las normales de los vértices para el cálculo.
- 3) Mostrar las sombras ahí donde las luces son ocluidas por los objetos
- 4) Llevar a cabo la reflexión de forma recursiva con una profundidad de al menos 7 iteraciones
- 5) Leer las propiedades de los materiales de la escena, incluyendo las coordenadas de textura y los ficheros con imágenes, añadir las texturas mapeadas a los objetos triangulares y modelos externos (se pueden utilizar coordenadas baricéntricas para calcular las propiedades de los materiales y las coordenadas de textura)

El código debe estar bien comentado y escrito de una forma entendible lo leeremos.

Por supuesto, el trazador de rayos puede mejorarse incluyendo una o varias características opcionales escritas en el apartado correspondiente.

## Competición y Galería de imágenes

Esta práctica tiene también un componente artístico, así pues se valorará la realización de escenas que demuestren las capacidades del trazador de rayos implementado. Los dos grupos con las escenas más impactantes a juicio de un conjunto de profesores del Máster, tendrán una bonificación en la nota final.

En la página web de la asignatura se publicará al menos una escena de cada trazador de rayos implementado, con el nombre de los alumnos que la han realizado, para su posterior votación.

## Código inicial

Se proporciona un código inicial, que es exactamente el mismo que se propone en la 3ª práctica de la asignatura de *Computer Graphics* de Nancy Pollard en la Carnegie Mellon University.

Básicamente se ofrecen las funciones de parsing y un visualizador de la escena en OpenGL. Los alumnos son libres de utilizarlo o no, pero los trazadores de rayos deben ser compatibles con el formato de los ficheros de descripción de la escena en XML.

- `raytracer.cpp/h`: Tiene la función `main` del trazador de rayos. No hay necesidad de cambiar muchas cosas en él, salvo quizás el tamaño de la ventana, ya que el trazador debe funcionar a cualquier resolución (no sólo la inicial, que se ha puesto para realizar los cálculos muy rápidamente).
- `Scene.h`: Contiene la librería para leer la escena, ofrece las funciones/variables para leer un fichero de entrada XML, las propiedades de la cámara, fondo, luces, los tres tipos de objetos geométricos, materiales... Es interesante leer los comentarios para entender cómo utilizarla.
- `test`
- `NormalRenderer.h` : Un renderizador de las escenas en XML basado en OpenGL. Puede servir como ejemplo para entender cómo hay que consultar la interfaz de `Scene.h`
- `RayTrace.h/c` : Como se ha indicado en el apartado de objetivos, es donde debería ir vuestro código.
- `Test.xml` y `scenes/*.xml` : Los ficheros de descripción de escenas de ejemplo. Cada grupo debe crear al menos uno para mostrar las posibilidades de su trazador de rayos y así participar en la Galería de imágenes y optar a la puntuación extra.

## Lectura recomendada

Si tenéis dudas en estos temas podéis consultar los libros de la bibliografía

- *Ray Tracing from the Ground Up*, *Kevin Suffern*
- *Physically Based Rendering : From Theory to Implementation* , *Matt Pharr y Greg Humphreys*
- *An Introduction to Ray tracing* , *Andrew S. Glassner*
- *Advanced Global Illumination*, *Philip Dutre, Kavita Bala, Philippe Bekaert y Peter Shirley*
- *Patterns for Parallel Programming*, *Timothy G. Mattson, Beverly A. Sanders y Berna L. Massingill*
- *Real Time Rendering*, *Tomas Akenine Möller*, 2ª edición. AK. Peters

## Partes opcionales

La calidad de las imágenes y el tiempo necesario para su generación es bastante mejorable en la implementación básica, por lo que se propone una serie de partes opcionales que serán tenidas muy positivamente en cuenta en la valoración de la práctica (siempre que funcione bien la versión básica).

Las nuevas funcionalidades que se pueden implementar son:

- Intersección Rayo-Escena
  - Mapeado de texturas
    - Filtrado de texturas tipo *mipmap*

- Displacement mapping
    - Texturas procedurales (mármol, madera, agua...)
  - Estructuras espaciales y algoritmos de visibilidad para la aceleración de los algoritmos
    - Cajas contenedoras alineadas a los ejes
    - Mallas regulares (Octrees)
    - Árboles KD
- Geometría
  - Cilindros
- Simulación de cámara físicamente correcta
  - Lentes ->  
<http://www.cs.princeton.edu/courses/archive/fall98/cs426/assignments/ray/camera.pdf>
  - Motion blur
  - Efecto de campo de profundidad
  - Efectos de difracción
- Materiales
  - Refracción
- Trazado de rayos
  - Sombras suaves
  - Reflexiones borrosas
  - Efecto de campo de profundidad
  - Muestreo circular/elíptico
  - Estimación combinada
- *Path tracing*
  - Iluminación difusa indirecta
- Paralelización del código
  - Sistema de balanceo de carga entre PCs
  - Implementación en CUDA
- Otros

Es importante incluir ficheros de descripción de escenas que muestren claramente los efectos implementados, y en la memoria se ha de describir tanto lo que se ha buscado para implementarlos como una comparativa con imágenes de ejemplo entre el “antes y el después” junto con el tiempo extra que lleva el cálculo del efecto (en función del número de muestras tomadas).

Este tipo de efectos algo más avanzados requieren bastante tiempo de cálculo, por lo que se recomienda en las pruebas tomar un bajo número de muestras y resolución de la imagen. Sin embargo, por cada efecto deberá incluirse una imagen a 1024x1024 con bajo ruido del resultado (escenas más “impresionantes” serán mejor puntuadas y se colocarán en un “Hall of Fame” en la página web de la asignatura).

## Cilindros

Añadir una nueva primitiva de forma cilíndrica, compuesta de la parte lateral de la superficie y las dos “tapas” con forma de disco. Para demostrar la implementación incluir una imagen de una escena en la que aparezcan cilindros con material Phong (¡cuidado con las normales en la iluminación!).

## Refracción

Añade refracción al trazador de rayos ampliando la interfaz de materiales con: Glass.

Este material tendrá una interacción con la luz que incluya los efectos de reflexión, refracción y las del material Phong. El código utilizado para la reflexión puede ser un buen inicio para introducir las variaciones debidas a este efecto. Para demostrar la implementación que has realizado, ilústralo con una imagen de una escena en la que se vea (entre otras cosas) una esfera de este material.

## Filtrado de texturas

Implementar el filtrado trilineal para texturas tipo *mipmap*. En esta práctica puedes utilizar un simple algoritmo heurístico basado en la distancia para determinar los niveles del mipmap en que hacer la consulta. Evidentemente, antes de implementar esta mejora, debería codificarse la anterior.

## Estructuras espaciales de aceleración

Implementar una de las siguientes estructuras espaciales: jerarquía de cajas contenedoras alineadas a los ejes, regillas regulares (octrees) o árboles BSP. En lugar de incluir una imagen para mostrar la mejora, haz una pequeña demo en la que se note la aceleración conseguida.

## Mallas normales interpoladas

Añadir un nuevo tipo de malla como geometría, que permita interpolación de normales basada en coordenadas baricéntricas.

## Sombras suaves

Añadir sombras suaves mediante muestreo Monte Carlo uniforme. Se puede tomar la fuente de luz como un área cuadrada en la que se muestrea, o bien como una esfera o un disco que la aproxime.

## Reflexiones borrosas

Incluir reflexiones borrosas al trazador de rayos mediante muestreo Monte Carlo uniforme, de manera que la dirección de la reflexión cambie ligeramente de uno a otro.

## Campo de profundidad

Añadir el efecto de campo de profundidad muestreo Monte Carlo uniforme al generar los rayos (esto además disminuye la percepción del efecto de *aliasing*).

## Muestreo circular o elíptico

Los píxeles ¡no son pequeños cuadrados! Por tanto, en lugar de tomarlos como tales, utilizaremos un dominio circular en lugar de cuadrado. Esto puede mejorar sustancialmente la calidad de muchos otros efectos.

## Estimadores combinados

Combinar varios efectos para ver de que modo varía la calidad y velocidad del resultado en función del número de muestras y la resolución (por ejemplo, campo de profundidad y sombras suaves)

## Iluminación indirecta

Implementar iluminación indirecta para una escena simple compuesta de materiales difusos utilizando el algoritmo de *path tracing*. Utilizar el muestreo uniforme de una semiesfera y la ruleta rusa como criterio de parada.

## Otros

Añadir luces volumétricas (ó superficiales) utilizando *next event estimation* al path tracer ó hacer muestreo “adaptativo” en el cálculo de la iluminación indirecta en función del tipo de material. Tratar de acelerar parte de los cálculos con OpenGL y Cg...

Las posibilidades son muchas, pero preguntad antes al profesor, no vaya a ser que la posible nueva mejora opcional que inventéis sea excesivamente compleja o carezca de valor didáctico.

## Recomendaciones

Comienza con la práctica tan pronto como te sea posible. Es una práctica que requiere bastante dedicación y tiene muchos pasos intermedios. Si lo dejas para unos días antes de la fecha tope es poco probable que tengas margen para poder terminarla a tiempo.

Es aconsejable que estéis familiarizados con el algoritmo de traza de rayos básica antes de poneros manos a la obra. Esta práctica es mucho más sencilla si se hace un ejercicio previo de diseño a la hora de establecer las clases/funciones.

No lo dejéis para el último día. Esta práctica está pensada para que sea fácil y relativamente sencilla, pero la presión del tiempo puede fastidiar por completo esta sensación.

En principio, la práctica se puede hacer en parejas, si alguien no encuentra pareja o, por problemas de horario tiene complicado el poder compaginarlo con el trabajo, también es posible hacerla de forma individual, esta práctica es sencilla y no debería suponer una dificultad añadida el no tener compañero/a para realizarla.

¡Experimenta con tus propias ideas y divierte con la práctica!

Esta práctica no requiere de hardware especializado para su realización. Los alumnos pueden realizarla cuando le resulte más cómodo en el aula de libre acceso S09 del edificio de Laboratorios II (Campus de Móstoles) que dispone de 15 ordenadores con tarjeta GeForce7900. Indicadle al responsable del aula que necesitáis un ordenador os dirá el que podéis utilizar (procurad que sea uno de los ordenadores con las tarjetas para que vuestros trabajos vayan más rápido).

## Forma y fecha de entrega

Se entregará la memoria en el formato de cualquier procesador de textos estándar (ó PDF) debidamente identificada junto con los ficheros con los programas en C (u otro lenguaje) en formato texto. En dicha memoria se deben indicar las dificultades que se han encontrado y el cómo se han superado, haciendo especial hincapié en las mejoras y partes opcionales incluidas.

También es necesario incluir al menos una escena de ejemplo (fichero XML junto con las texturas y ficheros .3DS y .OB necesarios) y una imagen desde la posición de cámara definida en formato JPG, que muestren las capacidades del trazador de rayos para la Competición y la Galería.

Puede enviarse directamente mediante la aplicación correspondiente en el Campus Virtual dentro del plazo, e indicando “[TAG3D Practica02]”. No olvidéis indicar vuestros nombres y apellidos en el cuerpo del mensaje.

La fecha tope de entrega de esta práctica será el **10 de Mayo** de 2010.

Planificaros bien, pues se solapa con el periodo de realización del resto de las prácticas. Esta práctica puede llevar una cantidad considerable de tiempo, pero también es muy divertida.

## Nota aclaratoria

Todas las marcas y productos mencionados en este enunciado de prácticas están registradas por sus respectivas compañías, y su uso es de carácter descriptivo con fines docentes.

Esta práctica está adaptada de la práctica 3 de las asignatura 15-462 de la Carnegie Mellon, y es un concepto original del equipo docente de Alexei Efros.