**INTRODUCCIÓN**

En este informe se va a exponer en detalle cada uno de los pasos realizados para completar un trazador de rayos. El trazador ha sido desarrollado en C++ y genera imágenes finales en formato “.ppm”. El trazador está formado por una cámara, que puede ser situada en cualquier punto, un plano con un ancho y alto que puede ser fijado por el usuario (se sitúa a una distancia de la cámara igual al ancho del plano), y finalmente, una escena compuesta por geometría.

El sistema implementado es capaz de generar geometría como esferas, así como simular la incidencia de la luz de focos puntuales situados en cualquier punto de la escena. Se permiten más de un foco de luz o esfera que son situados según un fichero de entrada elaborado por el usuario. En este archivo también se detallan aspectos como el color de cada una de las esferas o el tipo, ya que se puede indicar si son “lambertianas”, “phong”, “espejo” o “refracatada”.

Finalmente, además de la iluminación directa generada por los focos de luz puntuales, se simula la recibida indirectamente por cada elemento de la geometría generado. Esta aportación de luz, como ya se indicará posteriormente en su correspondiente apartado, se muestrea utilizando la técnica de Montecarlo, lanzando rayos mediante muestreo por importancia y promediando lo aportado cada uno de ellos. Al final de todo el proceso, se genera la imagen deseada, englobando todo lo indicado en el fichero de entrada.

**CÁMARA Y PLANO**

Para realizar un trazador de rayos, el primer paso es la generación de una cámara desde la que se lanzarán los rayos principales. Esta cámara se podrá localizar en cualquier punto del espacio marcado por tres coordenadas (x,y,z). Además, se podría indicar la dirección hacia la que apunta, o lo que es lo mismo, la parte del espacio hacia la que lanzará los rayos que intentarán intersectar con la geometría. En nuestro caso, se ha puesto que por defecto se lanzarán los rayos hacia el sentido positivo del eje “z”.

Una vez fijada la cámara, el siguiente paso era generar el plano de píxeles a través del cual se irían lanzando los rayos. Es plano de píxeles estaría compuesto por dos medidas: ancho y alto (estos dos tamaños irían también indicados en el fichero de entrada). Además de las medidas, era necesario fijar otro valor como es el tamaño de cada píxel, o el número de píxeles que compondrían el plano. En nuestro caso, decidimos que se aportaría el primero de los dos valores; es decir, el tamaño de cada píxel siendo este, como es lógico, un divisor de las dimensiones del plano para asegurar medidas exactas.

**TRAZAR RAYOS**

Con la cámara fijada en las coordenadas, y el plano situado con todas las medidas necesarias, el siguiente paso era la generación de rayos a través de cada píxel. En primer lugar, un rayo está formado por un punto y una dirección. El punto inicial del rayo está claro que va a ser el de las coordenadas de la cámara, ya que ésta será el foco de los rayos principales. Sin embargo, la dirección no era tan obvia de obtener, ya que para calcularla hacían falta un par de cosas.

La dirección del rayo se calculó a partir del punto inicial desde el que sería lanzado (coordenadas de la cámara) y el punto por el que tendría que pasar (coordenadas del píxel). El píxel estaba conformado por dos coordenadas (x,y) ya que el plano era de dos dimensiones, y esas coordenadas son calculadas independientemente para cada rayo. La coordenada “x” viene dada por el extremo horizontal del plano, que para que estuviera centrado respecto a la cámara, se fijaba en “cámarax-ancho/2” (análogamente para el extremo derecho). Para la coordenada “y” se realizó un proceso similar, fijando los extremos del plano en “cámaray-alto/2” y “cámaray+alto/2”.

Así pues, la dirección final del rayo se calculó mediante la resta a la coordenada del píxel en el plano menos el punto de origen de la cámara. Pero con esto tan sólo se tenían dos de las tres coordenadas necesarias para la dirección, faltando en último término la “z” por calcular. Esta sería simplemente la distancia de la cámara al plano, que como ya se comentó, sería igual a dos veces el ancho del plano. Con todo esto, ya se habían conseguido los dos datos necesarios para la generación del rayo, tanto el origen, como la dirección.

**INTERSECCIÓN GEOMETRÍA**

Tras haber obtenido todo lo necesario para lanzar el rayo, el siguiente paso era comprobar si intersectaba con algún tipo de geometría, en nuestro caso, con alguno de los distintos tipos de esfera. Para ello, se necesitaba partir de la ecuación para la intersección rayo esfera. La ecuación era la siguiente:

= 0

Esta ecuación podría dar cero, una o dos soluciones según los siguientes casos:

|  |  |  |
| --- | --- | --- |
| No intersecta | Tangente a la esfera | Atraviesa la esfera |

Con la ecuación se pueden obtener los resultados comentados, cuyo significado es la distancia de intersección respecto a la cámara. Como se indica en la imagen, el primer caso no se obtendría ninguna solución ya que el rayo no intersecta con la esfera. En el segundo caso se obtendría tan sólo una solución ya es tangente, y finalmente, en el último caso se obtienen dos soluciones ya que el rayo atraviesa la esfera, intersectando con ella a la entrada y a la salida.

A la hora de analizar la solución o soluciones de la ecuación de segundo grado había que tener un factor en cuenta. En caso de que el resultado fuera negativo, habría que ignorar dicha geometría, ya que significaría que la esfera está situada detrás de la cámara y no saldría en la escena. Si la solución es positiva, la esfera está delante de la cámara, por lo que se ha obtenido una geometría candidata a aparecer en la escena.

Una vez explicado lo que significa la solución obtenida para el sistema de ecuaciones, el siguiente paso es como saber cuál es la esfera más cercana para un cierto rayo. Para obtenerla el único procedimiento posible es recorrer todas las posibles esferas que han sido introducidas, e ir calculando la distancia de intersección para cada una de ellas. Si se obtiene una esfera se guarda, y si no se obtiene ninguna para el rayo, se sabe que el color que va a tener ese píxel es el mismo que el fijado para el fondo (negro).

**SOMBRAS**