Graphics and Imaging Lab

Ray tracing

Informática Gráfica

Grado en Ingeniería Informática

Objetivos

Los objetivos de esta tarea son los siguientes:

- Desarrollar estructuras clave y funciones para algoritmos de *render*.
- Vincular conceptos matemáticos con código.
- Servir como punto de referencia para probar nuevas primitivas, sensores o estructuras de aceleración.

1 Sensor

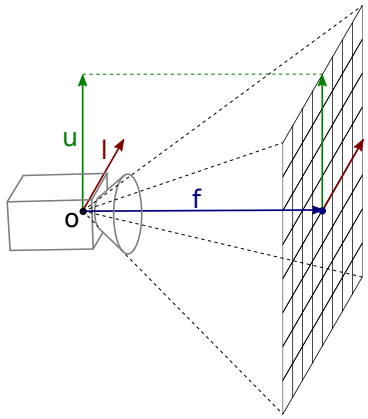

Figure 1: *Los elementos geométricos que componen una cámara* pinhole *o estenopeica.*

Los rayos serán emitidos desde un modelo de cámara. El modelo a seguir es una cámara *pinhole* o estenopeica (ver Figura 1). Ten en cuenta que el tamaño del plano de proyección (definido por los vectores hacia arriba u y hacia la izquierda l) debe ser proporcional a la resolución de la imagen (en píxeles). De lo contrario, la geometría aparecerá distorsionada según la relación de aspecto.

Es necesario conocer los límites de cada píxel (relacionados con el tamaño del plano de proyección y la resolución de la imagen) para emitir varios rayos por píxel. El número de rayos por píxel estará definido por un parámetro de línea de comandos.

2 Primitivas geométricas

La geometría consistirá en las siguientes primitivas:

- **Esfera:** definida por su centro y radio.
- **Planos:** plano infinito definido por su normal y su distancia al origen.

Cada primitiva puede ser intersectada por un rayo: esto da lugar a un sistema de ecuaciones en el cual el rayo tiene la ecuación de una línea y la geometría específica tiene su ecuación implícita. El sistema debe resolverse por sustitución para obtener el parámetro de un rayo.

Además de su información geométrica, cada primitiva también posee una propiedad de **emisión**, que se representa mediante una tupla de colores rojo, verde y azul (RGB), que define el color de la primitiva.

La aplicación que harás en esta tarea es una excelente referencia para poner a prueba estas nuevas primitivas.

3 Rendering

La aplicación será un *renderer* que emite varios rayos por píxel. Para cada pixel, promediará la emisión de la primitiva intersectada más cercana por cada rayo. Las escenas estarán codificadas en el código fuente, por lo tanto, no es necesario desarrollar un formato de archivo específico para cargar escenas. La imagen se guardará para su posterior visualización.

Se recomienda realizar varias pruebas cambiando el número de objetos y la resolución de la imagen para ver su efecto en el tiempo de *render*.

4 Extensiones opcionales

Los estudiantes pueden elegir una (o más) de las siguientes extensiones, que también servirán para las siguientes tareas evaluables:

Otras primitivas geométricas. Añade otras primitivas geométricas diferentes definidas por su ecuación implícita, como conos, cilindros, elipsoides, discos o triángulos. Deja volar tu imaginación.

Estructuras de aceleración. Para escenas con muchos objetos, puedes implementar *Bounding Volume Hierarchies* o *Kd-trees* para acelerar tu código. La mejora en el tiempo de *render* debe evaluarse si se implementa.

Paralelización. Paraleliza el algoritmo entre varios hilos de ejecución. Utiliza una estrategia de paralelización a nivel de imagen: subdivide la imagen en regiones (líneas / rectángulos / columnas / píxeles) y utiliza una cola de tareas (*thread-safe*) para configurar las regiones de *render*. Envía todas las tareas de *render* (un solo productor) y ten múltiples hilos extrayendo tareas y realizándolas (múltiples consumidores). Prueba diferentes estrategias: diferentes regiones de imagen y tamaños de regiones de imagen, diferentes implementaciones de colas *thread-safe* y diferentes combinaciones entre paralelización y estructuras de aceleración.

Texturas. Añade texturas que modulen la emisión de un objeto.

***Render* espectral.** En lugar de tener propiedades de emisión RGB para cada objeto, incluye una representación espectral del color que luego se convierte a RGB utilizando las curvas estándar de RGB de CIE. La representación espectral puede ser un vector de 8, 16 o 32 valores, o incluso una función espectral (una función que depende de la longitud de onda) que se muestrea en los valores correspondientes de los 8, 16 o 32 canales.

Otras extensiones. Los estudiantes deben discutir cualquier otra extensión que estén considerando con el profesor antes de proceder, con el fin de evitar aquellas que puedan requerir un esfuerzo excesivo.

5 Detalles de implementación

Lenguaje de programación. Los estudiantes pueden elegir cualquier lenguaje de programación que consideren. Sin embargo, recomendamos C++ debido a lo siguiente:

- El código compilado debería ser más eficiente que otros lenguajes.
- El código que se proporcionará para el segundo trabajo práctico está en C++ (a menos que elijas usar tu propio código para ambos).
- C++ permite definir operadores como + o * para cualquier nuevo tipo de datos. Esto hace que el código sea más legible para tipos de datos como vectores.

Formatos de archivo. Como has utilizado el formato de imagen `.ppm` para cargar y guardar imágenes en tareas anteriores, recomendamos el formato de imagen `.ppm` para escribir la salida del *ray tracer* en una imagen. Es un archivo de texto muy simple que puede ser leído por la mayoría de los visores y es muy fácil de leer y escribir. Puedes encontrar los detalles en `http://netpbm.sourceforge.net/doc/ppm.html`. También se puede utilizar para cargar imágenes como entrada si se incluye la extensión opcional de texturas.

Si los estudiantes deciden incluir la extensión opcional de mallas de triángulos y quieren car-

garlas, recomendamos el formato de archivo `.ply`, ya que es un archivo de texto muy simple. La descripción de dicho formato, junto con algunos modelos `.ply` gratuitos, se puede encontrar en `https://people.sc.fsu.edu/~jburkardt/data/ply/ply.html`.

Bibliotecas externas. El uso de cualquier tipo de biblioteca externa o código fuente descargado está prohibido. A los estudiantes solo se les permite utilizar la biblioteca estándar que viene con el lenguaje de programación elegido. La única excepción a esta regla es incluir código que sea propio y que haya sido creado previamente (quizás proveniente de asignaturas de informática anteriores). Si se incluye código que es propio y anterior, se debe documentar adecuadamente (cuándo se generó, para qué tarea y en qué asignatura).

Sumisión

No es necesario realizar una sumisión para esta tarea en particular. Sin embargo, se sugiere que se complete antes de la fecha de entrega recomendada para equilibrar la carga de trabajo.