Graphics and Imaging Lab

Procesamiento de imágenes

Informática Gráfica

Grado en Ingeniería Informática

Objetivos

- Aplicar conocimientos teóricos en imágenes rasterizadas y formatos de imagen.
- Comprender los problemas del rango dinámico de una imagen y dónde aplicar técnicas de tone mapping.
- Diseñar e implementar la carga y el guardado de imágenes (que se utilizarán en prácticas posteriores).

1 Planteamiento del problema

Las imágenes en la naturaleza normalmente tienen un alto rango dinámico (HDR, por sus siglas en inglés, High Dynamic Range) con una luminancia que varía desde la llama de una vela hasta un sol radiante. Cuando utilizamos un renderizador basado en física, las imágenes generadas pueden contener un rango de valores tan amplio como las imágenes naturales.

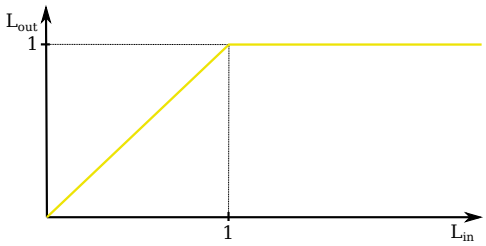
Este rango de valores para la luminancia no es aceptable para los formatos estándar de imagen ni para las pantallas, lo que provoca que las zonas más brillantes queden sobreexpuestas (volviéndose blancas) mientras que las zonas más oscuras quedan subexpuestas (volviéndose negras). Por estos motivos, existe la necesidad de aplicar el "tone mapping" a las imágenes de entrada, para que se ajusten al bajo rango dinámico convencional del formato de imagen/pantalla correspondiente, lo que permite visualizar todos los elementos de una imagen. Este problema es bastante importante, y esta práctica lo aborda ligeramente, pero hay libros de gran éxito en este tema [RWPD05].

2 Tone mapping

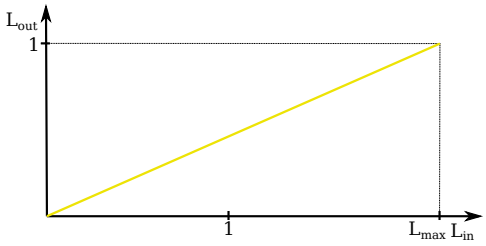
La tarea consiste en desarrollar una serie de estrategias de tone mapping como aplicaciones independientes, que tengan como parámetros la imagen de entrada HDR, la imagen de salida LDR y los parámetros del operador de tone mapping. Una vez cargada en la memoria de la computadora, una imagen se guarda como un vector 2D de tripletas RGB con precisión de punto flotante, para evitar la pérdida de información al abrir y guardar la imagen.

Los operadores de tone mapping son simples:

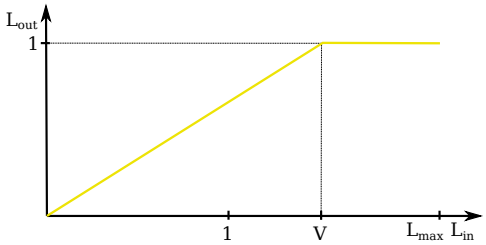
- **Clamping:** Acotar todos los valores superiores a 255 (1 en coma flotante).



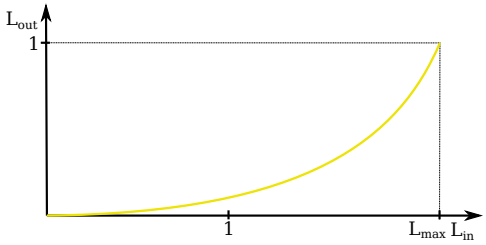
- **Ecualización:** Transformación lineal de los valores desde el mínimo hasta el máximo (normalización).



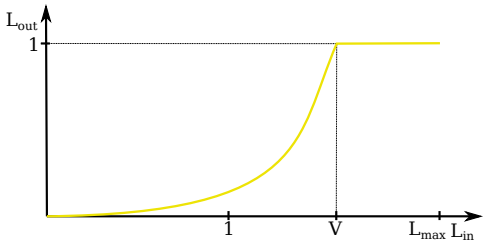
- **Ecualización y clamp:** Combinar los dos anteriores según un parámetro de "clamping." Nótese que los dos operadores anteriores pueden considerarse casos particulares de este operador.



- **Curva gamma:** Aplicar una curva gamma a todos los valores (necesita ecualización primero).



- **Clamp y curva gamma:** Aplicar una curva gamma después de una operación de clamping (necesita ecualización primero). Nótese que todos los operadores anteriores pueden verse como casos particulares de este operador.



3 Formato de imagen

Vamos a usar el formato de imagen plano `.ppm` para escribir la salida del ray tracer en una imagen. Es un formato de fichero muy simple que puede ser leído por la mayoría de visualizadores y es muy simple de leer y escribir. Puedes encontrar más detalles en `http://netpbm.sourceforge.net/doc/ppm.html`.

La idea es usar este formato de imagen tanto para almacenar imágenes HDR como LDR. El formato estándar de imagen no admite imágenes HDR, pero se puede modificar para lograrlo. Una imagen estándar sigue una estructura similar a esta:

```
1  P3
2  # feep.ppm
3  4 4
4  15
5   0  0  0    0  0  0    0  0  0   15  0 15
6   0  0  0    0 15  7    0  0  0    0  0  0
7   0  0  0    0  0  0    0 15  7    0  0  0
8  15  0 15    0  0  0    0  0  0    0  0  0
```

Esto incluye el identificador de formato P3, comentarios (líneas iniciadas por #) y resolución en términos de anchura y altura (en el caso de la imagen de arriba, ambos son 4) y la resolución en espacio de color, que se traduce en el número potencial de valores de color. Finalmente, se encuentran las tripletas RGB ordenadas. Después de comenzar la sección de tripletas RGB, no se permiten comentarios.

En memoria, las imágenes se almacenan en reales con precisión de coma flotante. Al leer de fichero, para cada valor s (R, G o B de una tripleta RGB), y dada la resolución de color c, el valor a almacenar cuando se carga este formato será $\frac{s}{c}$ (por lo tanto, estará comprendido entre 0 y 1). El estándar solo permite valores de $c \leq 65535$ como resolución en espacio de color, pero al almacenar imágenes HDR, se eliminará esa restricción. En general, cuando guardemos imágenes LDR, usaremos $c = 255$, y cuando guardemos imágenes HDR (cuando rendericemos en prácticas posteriores), usaremos, por ejemplo, $c = 2^{30}$ para una mayor resolución en el rango en espacio de color.

Aun así, con este formato HDR, el máximo de cualquier imagen sería solo 1. En su lugar, vamos a introducir un comentario opcional como `#MAX=18.35` que establece el máximo **real** como un número real. Como consecuencia, si c es la resolución de color, s es el valor almacenado en el archivo (R, G o B), m es el valor máximo de la imagen y v es el valor real almacenado en memoria de la imagen, al cargarla (del disco a la memoria) hay que aplicar la siguiente ecuación:

$$v = s \cdot \frac{m}{c} \tag{1}$$

y al guardar (de la memoria al disco) hay que aplicar la inversa:

$$s = v \cdot \frac{c}{m} \tag{2}$$

Como material suplementario, proporcionamos un conjunto de imágenes HDR en formato ppm con el objetivo de que podais validar la carga y el guardado, así como los diferentes operadores de tone mapping.

4 Opcionales

Los estudiantes pueden elegir una (o más) de las siguientes extensiones, que mejorarán las tareas evaluables posteriores:

Otros formatos de imagen LDR en los que guardar la imagen. Hay otros formatos LDR que son binarios y son razonablemente fáciles de guardar y cargar sin ninguna biblioteca externa. Esto incluye, por ejemplo, el formato bmp. Sin embargo, muchas aplicaciones de edición de imágenes pueden cargar archivos ppm estándar.

Operadores avanzados de asignación de tonos. Añade otros operadores de tone mapping más avanzados, incluyendo otros operadores globales o incluso locales. Se puede elegir cualquier operador de los discutidos durante la clase [RSSF02, MDK08].

5 Detalles de implementación

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

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

Librerías externas. Está prohibido el uso de cualquier tipo de biblioteca externa o código fuente descargado. Los estudiantes solo pueden utilizar la biblioteca estándar que viene con el lenguaje de programación elegido. La única excepción a esta regla es incluir código que sea tu propio trabajo previo (quizás proveniente de asignaturas anteriores de informática). Si incluyes código que es tu propio trabajo previo, deberás documentarlo adecuadamente (cuándo lo generaste, para qué tarea y en qué asignatura).

Sumisión

No es necesario realizar una entrega específica 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.

References

[MDK08] RafałMantiuk, Scott Daly, and Louis Kerofsky. Display adaptive tone mapping. *ACM Trans. Graph.*, 27(3):68:1–68:10, August 2008.

[RSSF02] Erik Reinhard, Michael Stark, Peter Shirley, and James Ferwerda. Photographic tone reproduction for digital images. *ACM Trans. Graph.*, 21(3):267–276, July 2002.

[RWPD05] Erik Reinhard, Greg Ward, Sumanta Pattanaik, and Paul Debevec. *High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting (The Morgan Kaufmann Series in Computer Graphics)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.