

Computación Gráfica

Trabajo Práctico 3

Yet Another Tracer

Because there are not too many



Grupo 3

Del Giudice Gustavo

Menzella Facundo

Noriega Jose

Índice

[Depth of Field](#)

[Cámara "Fish Eye"](#)

[Environmental maps](#)

[Parseo de escenas](#)

[Escenas de prueba](#)

[Resultados](#)

[Depth of field](#)

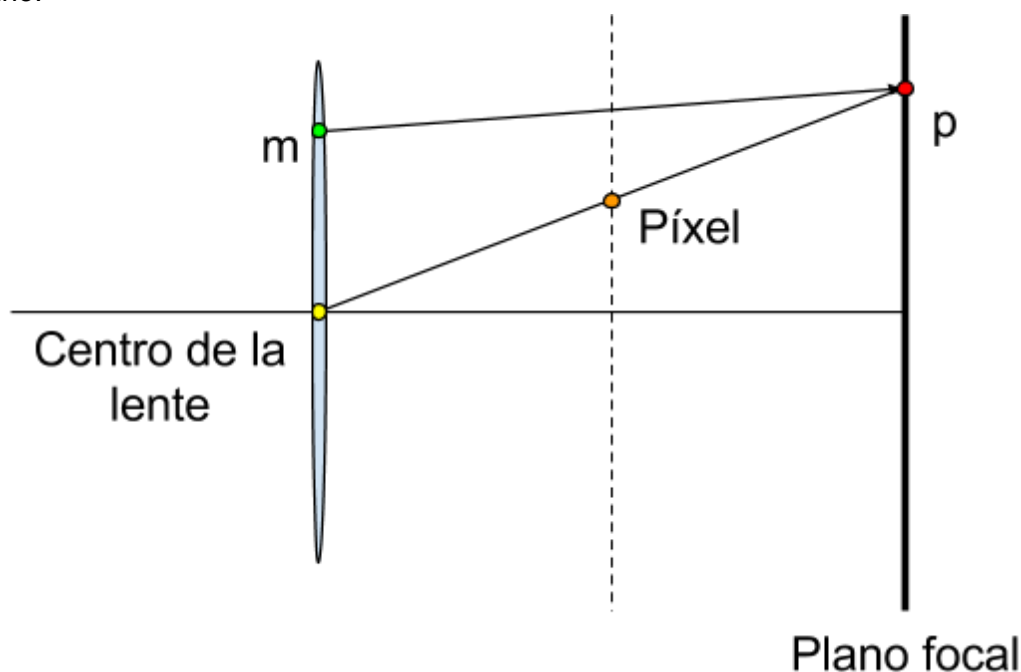
[Fisheye](#)

[Environmental maps](#)

[Otras escenas](#)

Depth of Field

Para simular Depth of Field, se simplificó el modelo real. En primer lugar, se utiliza un disco de radio variable y espesor despreciable para simular la lente de la cámara. El centro de este disco coincide con la posición de la cámara, y es perpendicular a la dirección de la misma. Luego, se define un plano focal, paralelo al disco y a una distancia ajustable por el usuario.



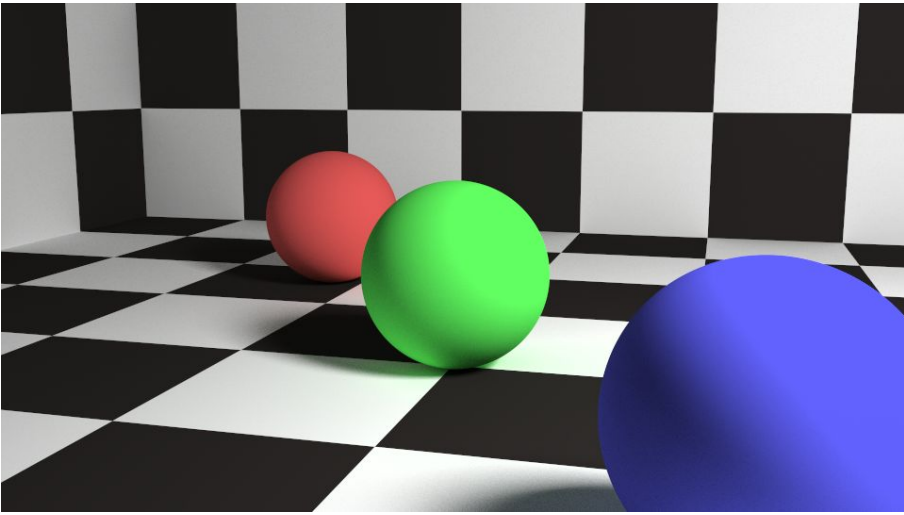
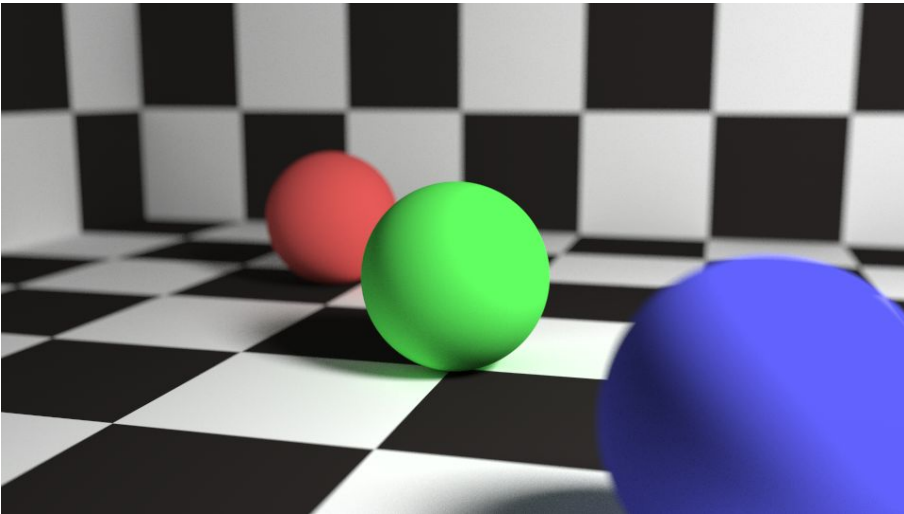
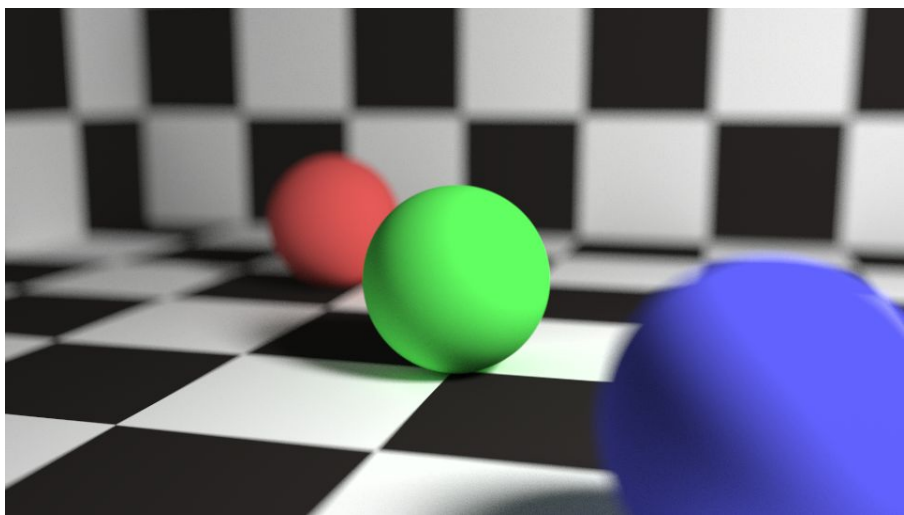
Finalmente, para producir una imagen que tenga depth of field, se realizan los siguientes pasos:

1. Se calcula la intersección **p** de un rayo que sale del centro de la lente y pasa por el píxel a dibujar, con el plano focal.
2. Se toma una muestra **m** en el disco de manera uniforme. Luego, se crea un rayo con origen en **m** y dirección hacia **p**.
3. Se realiza el path-tracing con el rayo generado en el paso anterior.

De esta forma, cualquier objeto que se encuentre en el plano focal estará perfectamente focalizado, mientras que cualquier objeto que se encuentre fuera de dicho plano aparecerá fuera de foco (con mayor magnitud a medida que se aleje del plano).

Nuestra implementación permite variar la distancia entre la lente y el plano focal, y el radio de la lente. Los objetos fuera de foco se verán más difusos con mayores radios de lente. Si el radio de la lente es 0, la cámara se comporta como una Pinhole Camera y no habrá depth of field. Para acelerar la toma de muestras durante el rendering, se producen N muestras aleatorias antes de comenzar, y se toman de forma secuencial en una lista circular. Con un

N lo suficientemente grande (1000 muestras) no se encontraron fallas ni patrones en las imágenes generadas.

| Radio de lente | Resultado |
|----------------|--|
| 0 |  |
| 0.05 |  |
| 0.1 |  |

Cámara “Fish Eye”

Para implementar la proyección FishEye se utilizó la implementación del libro recomendado por la cátedra, “*Ray Tracing from the ground up*”.

En el modelo utilizado, el *Field of View* es independiente del tamaño de la pantalla. Además, sólo los píxeles centrados en el disco circular son los que se renderizan.

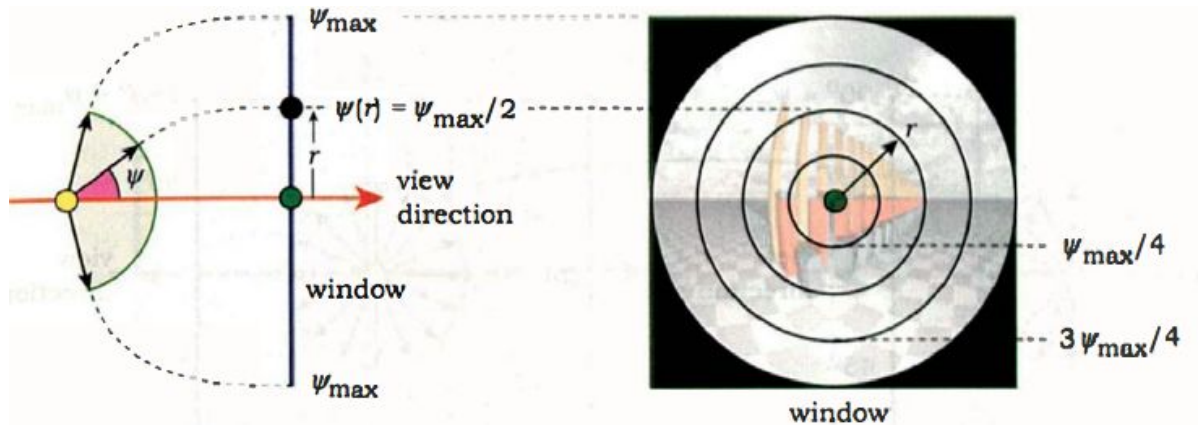


Imagen extraída de “*Ray Tracing from the ground up*”

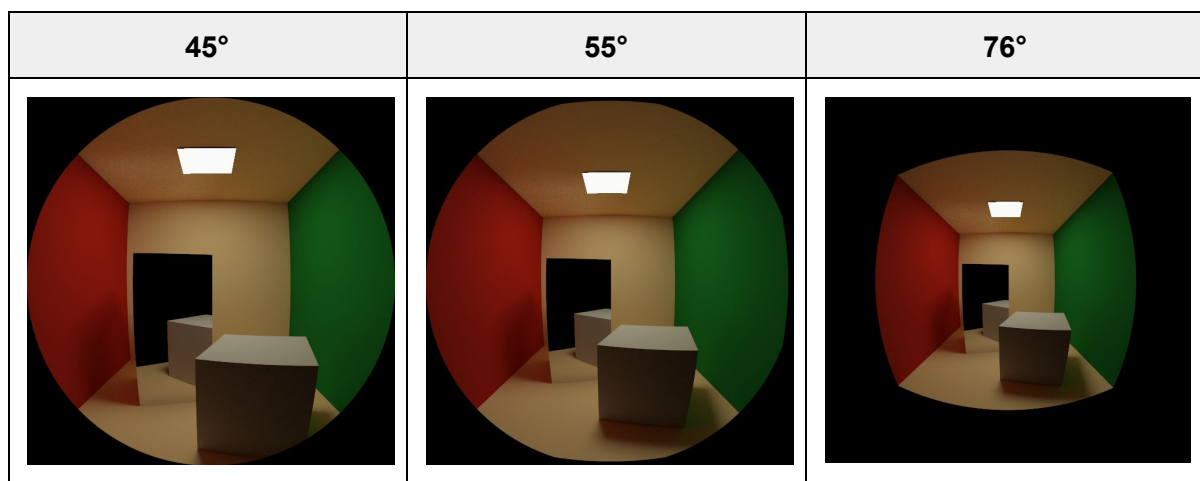
Para el cálculo se φ_{max} se usó la siguiente relación:

$$\varphi_{max} = 2 * fov$$

Es así como para cada sample se procede de la siguiente forma:

1. Se computa la distancia r .
2. Si el píxel se encuentra en dentro del disco, se computa $\varphi(r)$.
3. Se computa la dirección del rayo.

A continuación se muestran 3 imágenes con distinto Field of View:



Environmental maps

Para poder generar escenas de mayor realismo, se decidió implementar el uso de environmental maps. Los mismos consisten en una imagen que “rodea” a la escena. Cuando un rayo sale de la escena, en lugar de retornar el color de la luz ambiental, se toma una muestra de dicha imagen con coordenadas latitud/longitud. De esta forma, no solo se puede ver la imagen reflejada en materiales metálicos, sino que, gracias a la iluminación indirecta que resulta de utilizar path-tracing, la escena toma el color de la imagen de fondo. A continuación se puede ver la misma escena, pero con distintos environmental maps.



Para muestrear la imagen, se utiliza la dirección del rayo, normalizada.

Con el vector $\mathbf{r} = (x, y, z)$ como entrada, se calculan las coordenadas UV de la siguiente forma.

$$\theta = \text{acos}(z)$$

$$\varphi = \text{atan2}(y, x)$$

$$u = \frac{(\pi + \varphi)}{2 \times \pi}$$

$$v = \frac{\theta}{\pi}$$

Parseo de escenas

Debido a la flexibilidad que ofrecía el diseño del parser en el Trabajo Práctico anterior, no fue necesario realizar grandes cambios en el mismo para soportar las nuevas funcionalidades. Las mismas restricciones explicadas en el informe anterior siguen existiendo en esta versión.

Para obtener depth of field, se utilizan las propiedades *lensradius* y *focaldistance* definidas en Camera, según la especificación de LuxRender. Si alguno de los dos valores es 0, la cámara se comporta como Pinhole Camera y no habrá depth of field.

Para obtener una cámara fisheye, se debe definir una cámara con el tipo fisheye, en lugar de perspective. Las propiedades *lensradius* y *focaldistance* no están disponibles para este tipo de cámara.

Se puede definir un *environmental map* de la misma forma que en LuxRender: se define una luz del tipo *infinite*, con la propiedad *mapname* siendo el path a la imagen a utilizar. Las únicas restricciones son:

- La propiedad *mapping* será siempre *LatLong*.
- No se pueden cargar imágenes HDR, a pesar de ser lo recomendable. La librería ImageIO no permite cargar este formato, y se decidió evitar cambiar la librería a este punto del desarrollo.

Para recuperar el material *MetaL2* definido en la primera entrega, se utilizó el material *Glossy*, según la especificación de LuxRender. Se aceptan las siguientes propiedades:

- Kd
- Ks
- roughness y vroughness (si ambas están definidas, se considera el mayor de los valores)

Escenas de prueba

Todas las escenas que se muestran en este informe fueron generadas con una PC con las siguientes características:

- Sistema operativo: Debian 8 de 64 bits
- CPU: AMD FX-8150 de 8 núcleos @ 3.60Ghz
- RAM: 16GB
- JVM: OpenJDK 8

Debido a que en esta entrega se intentó de hacer énfasis en nuevas funcionalidades, no se realizaron benchmarks para las escenas. Lo implementado no afecta significativamente la performance, por lo que se realizó solo una corrida por escena. Las mismas fueron generadas con un Trace Depth de 10 saltos y 1000 muestras por píxel.

| N° | Archivo | Comentario |
|----|---|---|
| 1 | DepthOfField/depth-of-field-control.lxs DepthOfField/depth-of-field-low.lxs DepthOfField/depth-of-field-high.lxs | Se proveen escenas con cámaras de distintos Lens Radius. |
| 2 | CornellBoxFisheye/cornell-box-fisheye.lxs CornellBoxFisheye/cornell-box-fisheye-low-fov.lxs CornellBoxFisheye/cornell-box-fisheye-high-fov.lxs | Se proveen escenas con cámaras fisheye de distintos FOV. |
| 3 | MetalMario/metal-mario-wall.lxs MetalMario/metal-mario-bg1.lxs MetalMario/metal-mario-bg2.lxs MetalMario/metal-mario-bg3.lxs MetalMario/metal-mario-bg4.lxs | Escenas con distintos environmental maps. |
| 3 | CaptainFalcon/captainfalcon.lxs | Gran tiempo de rendering debido a que la imagen final tiene una resolución mayor a las demás escenas. |
| 4 | GokuSelfie/goku-selfie.lxs | -- |
| 5 | Ornstein/ornstein.lxs | -- |
| 6 | Knight/knight.lxs | -- |

Resultados

A continuación, se muestran los tiempos de render de todas las escenas incluidas. Solo se realizaron las pruebas con 1000 muestras por píxel, ya que el impacto de este parámetro se discutió en la entrega anterior.

Depth of field

| Radio de lente | Tiempo de render |
|----------------|------------------|
| 0 | 9m 45s |
| 0.05 | 9m 36s |
| 0.1 | 9m 25s |

Se puede observar que habilitar depth of field no causa impactos en la performance. Esto se debe a que la única operación adicional durante el rendering es calcular la dirección de un segundo rayo. Si se quisiera utilizar con Raytracing, se necesitaría incrementar notablemente la cantidad de muestras por píxel. En ese caso, sí causaría un incremento notable en el tiempo de render.

Fisheye

| Field of View | Tiempo de render |
|---------------|------------------|
| 45° | 17m 36s |
| 55° | 14m 14s |
| 76° | 8m 11s |

Se puede observar que la escena elegida para probar Fisheye presenta menores tiempos de render a medida que incrementa el Field of view. Esto se debe a que la mayoría de los rayos no pasan por la escena. Como Fisheye no realiza operaciones extra y siempre se realiza path-tracing sobre la misma cantidad de rayos, se concluye que este modelo de cámara no produce impacto en la performance.

Environmental maps

| Escena | Tiempo de render |
|---------|------------------|
| Cerrado | 54m 29s |
| Fondo 1 | 10m 26s |
| Fondo 2 | 10m 18s |
| Fondo 3 | 10m 29s |
| Fondo 4 | 10m 23s |

Se puede apreciar cierta mejora en la performance al utilizar un environmental map, en lugar de cubrir la escena con geometría. Esto se debe a que la geometría produce más rebotes en el rayo, lo cual incrementa el tiempo de path-tracing. En cambio, un environmental map hace que corte el algoritmo y provee resultados realistas al mismo tiempo.

Otras escenas

| Escena | Tiempo de render |
|----------------|------------------|
| Captain Falcon | 2h 47m 55s |
| #DBZSelfie | 18m 18s |
| Ornstein | 30m 29s |
| Knight | 1h 3m 35s |