Sombreado Diferido

(Deferred Shading)

Héctor Barreiro Cabrera

Sombreado Diferido (Deferred Shading)

PRESENTACIÓN

Sombreado directo: ¿Lo cualo?

- Antes de hablar del sombreado diferido, necesitamos hablar del sombreado directo.
- *Forward shading*.
  - Método que hemos usado hasta ahora.
- Una única etapa.
  - Por cada objeto:
    - Buscar luces que lo afecten.
    - Renderizar todas las luces y materiales con un único *shader*.
    - Usar modo aditivo para sumar la contribución de todas las luces después de la primera.
  - Los objetos se dibujan directamente al *framebuffer* final.

Sombreado directo: Un vistazo

http://vimeo.com/73178894

Cortesía de **Christopher Riccio**

Sombreado directo: ¡Ups!

- ¡Oclusión!
  - En una escena típica es muy probable que existan oclusiones.
  - Todo ese valioso tiempo empleado en iluminar fragmentos ocluidos se ha desperdiciado.
  - Encima, el coste de dibujar cualquier objeto de esta forma es O(nm).
    - n – número de objetos
    - m – número de luces

  - Existen algunas soluciones.

Solución 1: Z-Prepass

- Realizar un paso previo dibujando toda la geometría con un *shader* simple, sólo escribiendo la profundidad.
- El *Z-Buffer* se encargará de que a la hora de la verdad sólo se iluminen los fragmentos visibles.
  - Early Fragment Test.
- Aun así, tenemos que redibujar la geometría múltiples veces para poder calcular la contribución de todas las luces.

Solución 2: Sombreado diferido

- *Deferred shading.*
- Esperemos hasta el final del dibujado para saber que fragmentos iluminar.
  - Para ello, almacenar la información geométrica del fotograma en un *Framebuffer* (*G-Buffer*).
  - El *G-Buffer* consistirá de varias texturas que contendrán la información necesaria para iluminar la escena.
    - Posiciones, normales, albedo, parámetros…
- Una vez toda la información esté en el *G-Buffer* se procede a iluminar.
  - Para ello se dibuja un *quad* que lea esta información y produzca la imagen final.

Sombreado diferido: Esquema

- *Inicialización*
  - Generar texturas del *G-Buffer.*
  - Generar *Framebuffer* y asociar texturas (MRT).
  - Generar VBO y VAO para el *quad*.
- Dibujado
  - Enlazar *Framebuffer*.
  - Dibujar escena (enlazar VAOs, programas, etc.)
  - Desenlazar framebuffer, enlazar texturas.
  - Activar programa de composición.
  - Dibujar *quad.*
  - Mostrar en pantalla.

Sombreado diferido: ¿Magia?

- Ventajas:
  - El coste de calcular la iluminación pasa a ser O(n+m).
    - Independiente de la complejidad de la escena.
    - **Puede** traducirse en una mejora de rendimiento importante.
  - Además produce la información necesaria para algunos efectos de post-procesado.
- Desventajas
  - ¡Requiere mucho ancho de banda!
  - El G-Buffer puede ocupar mucha memoria.
  - Limitado a un único modelo de iluminación.
    - Existen técnicas para permitir múltiples modelos.
  - No funciona con las técnicas de antialiasing tradicionales.
  - No funciona con geometría transparente.

Sombreado diferido: Pues vaya leche

- Soluciones:
  - Ancho de banda / memoria / modelo de iluminación:
    - Técnica derivada: Iluminación diferida.
    - Reduce el G-Buffer y permite modelos de iluminación distintos para cada tipo de material.
  - Antialiasing:
    - Calcular el antialiasing como efecto de post-procesado.
      - Algoritmos de detección de bordes + desenfoque.
  - Geometría transparente:
    - Combinar sombreado directo con sombreado diferido.
      1. Usar sombreado diferido para superficies opacas.
      2. Usar sombreado directo para superficies transparentes.

Sombreado Diferido (Deferred Shading)

IMPLEMENTACIÓN

Implementación: La he liado parda

- Implementado en C# con *OpenTK*.
- Arquitectura similar a la típica de los motores gráficos.
  - Capa de abstracción OO simple.
    - *Shaders*, *Framebuffers*, Texturas, *Uniforms*, etc.
  - Gestor de recursos basados en conteo de referencias.
  - Grafo de escena simple.

Implementación: La he liado parda (II)

Implementación: La he liado parda (III)

- Los materiales se implementan mediante un único *übershader*.
  - Se generan las permutaciones necesarias en función del tipo de renderizado y parámetros disponibles.
- Modelo de iluminación basado en el Unreal Engine 4.
  - Parámetros inspirados en el artículo de Disney "*Principled Physically Based Shading*".

Implementación: La he liado parda (IV)

- Tipos de luces soportadas:
  - Puntual.
  - Direccional.
  - Esférica de área.
  - Foco.
- Todos los cálculos se hacen en espacio de color lineal.
  - Aproximado con $\gamma = 2$.
- La conversión al espacio *gamma* se realiza cuando se vuelca el resultado final a pantalla.

Modelo de iluminación difuso

- Lambert normalizado.

$$D(N, L) = \frac{N \cdot L}{\pi}$$

- Suficientemente bueno como para no molestarnos en implementar otros más avanzados.
- El factor de normalización es crítico para cumplir el principio de conservación de energía.

$$\int_{\Omega} \cos \alpha \leq 1$$

Modelo de iluminación especular

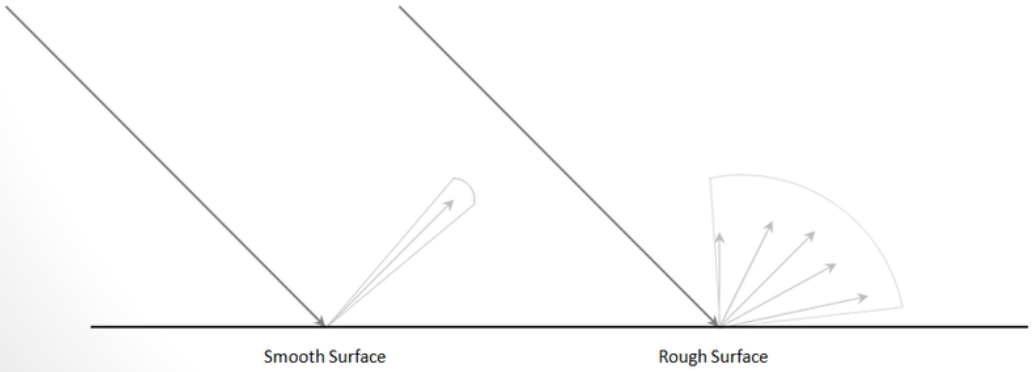
- Cook-Torrance.

$$S(L,V) = \frac{D(H)F(L,V,H)G(L,H)}{4\,(N \cdot L)\,(N \cdot V)}$$

  - Modelo de microfacetas.
    - Asume que cada faceta de la superficie es un espejo perfecto.
  - Tres términos:
    - Distribución especular (D).
    - Atenuación geométrica (G).
    - Término de Fresnel (F).

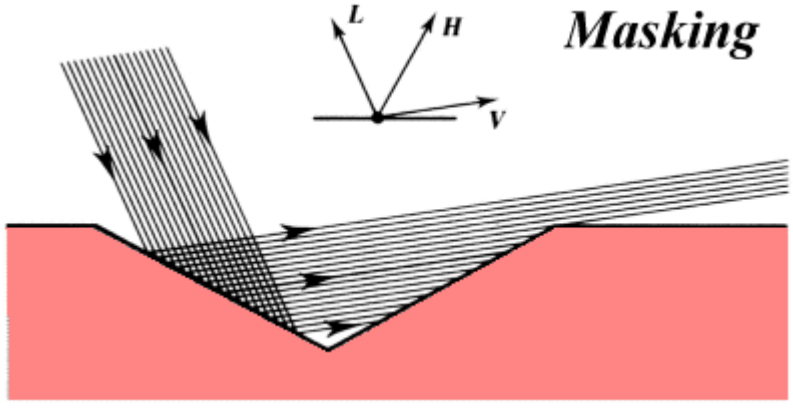
Modelo de iluminación especular (II)

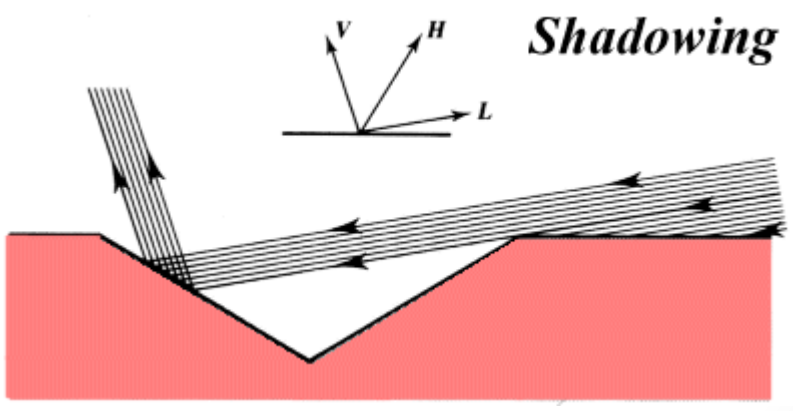
- Distribución especular.
  - Indica la probabilidad de que las facetas estén orientadas hacia el vector H.
    - Básicamente indica la forma del brillo especular.
  - Se emplea la aproximación de Trowbridge-Reitz (GGX).



Modelo de iluminación especular (III)

- Atenuación geométrica.
  - Indica la probabilidad de que las facetas estén siendo ocluidas por otras facetas.
- Se emplea la aproximación de Schlick.





Modelo de iluminación especular (IV)

- Término de fresnel.
  - Determina la cantidad de luz reflejada en función del punto de vista.
  - Se emplea la aproximación de Schlick.

Formato del G-Buffer

Nombre	Formato	Canal R	Canal G	Canal B	Canal A
1 – Posición	*RGBA32f*	Posición X	Posición Y	Posición Z	AO
2 – Normales	*RGBA16f*	Normal X	Normal Y	Normal Z	-
3 – Albedo	*RGBA16f*	Albedo R	Albedo G	Albedo B	-
4 – Material	*RGBA16f*	Metalness	Roughness	Gloss	Emission
5 – Depth	*Depth*	-	-	-	-

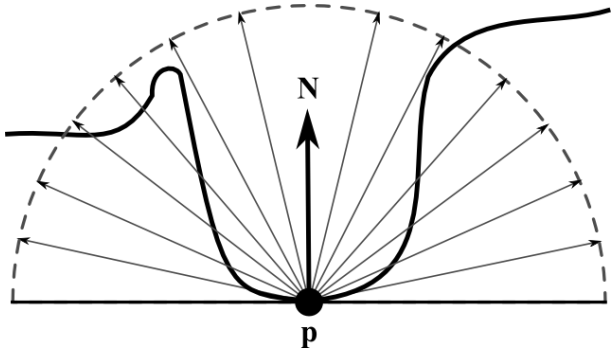
- Tamaño por pixel: 44 Bytes
  - Para resolución de 1280x720: ¡38.67 MB!
- Pero necesitamos esa precisión extra para evitar artefactos.
  - Se podría reducir empleando otros esquemas de codificación (YUV).

Efectos de Post-Proceso

- Por desgracia no me ha dado tiempo a implementar un sistema para gestionarlos.
- Así que sólo os puedo mostrar dos (y uno de ellos sin acabar).
  - SSAO: Oclusión ambiental en el espacio de imagen.
  - SSR: Reflejos especulares en el espacio de imagen.

Screen-Space Ambient Occlusion

- Trata de aproximar la oclusión producida por obstáculos cercanos.
- Para ello muestrea la información geométrica en el hemisferio alrededor del punto.
  - Distribución de muestreo aleatorio empleando un disco de Poisson.
  - $AO(p,n) = \frac{\sum_i^n 1-(A_i(p,n)B_i(p,n))}{n}^x$
  - $A_i(p,n) = 1 - smoothstep(0, k_d, |p - p_i| - k_b)$
  - $B_i(p,n) = \frac{p_i - p}{|p_i - p|} \cdot n$
- Si existen oclusores cercanos se produce una atenuación en la contribución de ese fragmento.

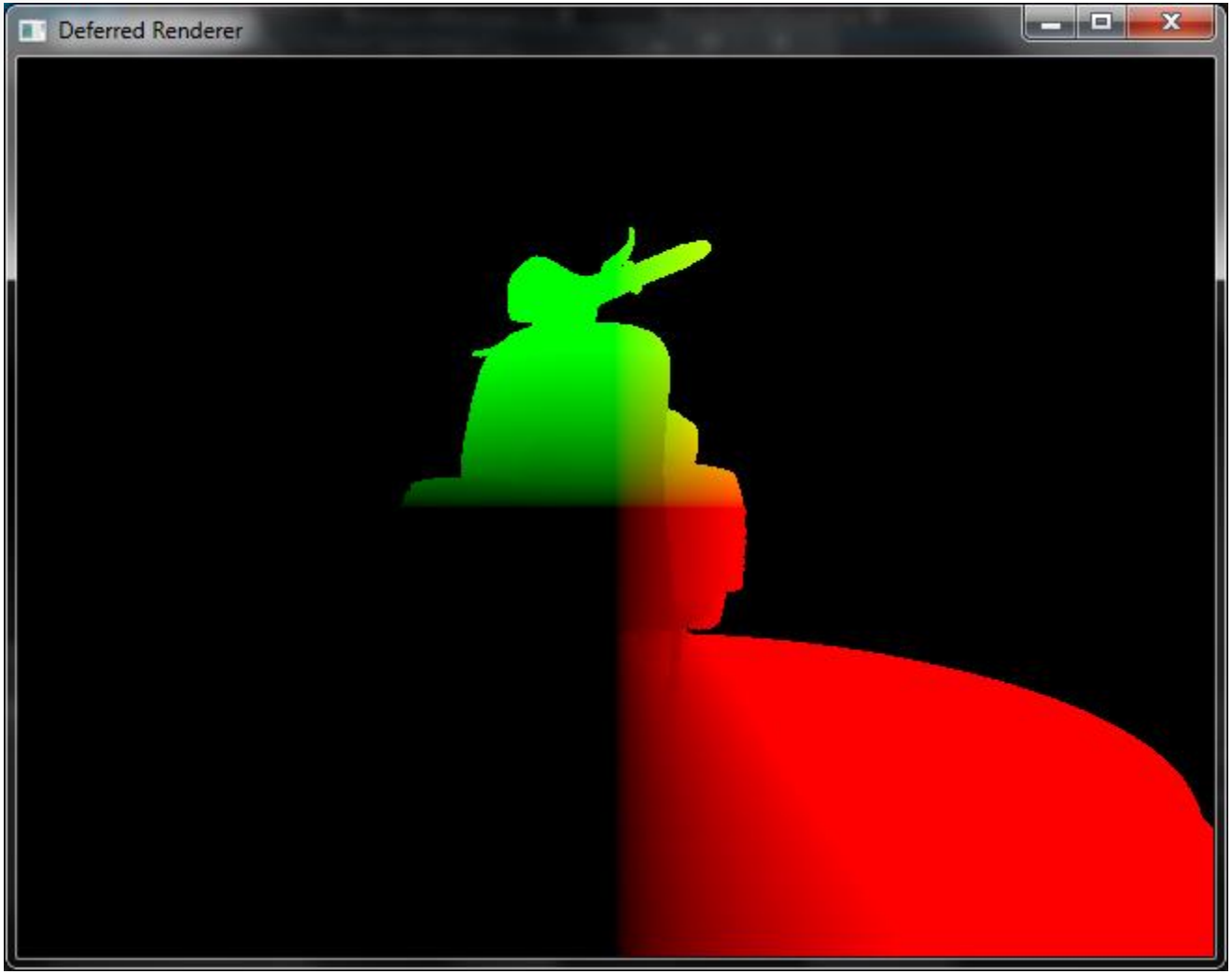


Screen-Space Reflections

- Se trata de aproximar el reflejo en las superficies empleando información en el espacio de imagen.
  - Para ello se marcha a lo largo de la dirección del vector reflejo.
  - En cuanto se encuentra una intersección, el color de ese fragmento es el reflejo.
- Los principales problemas de la técnica son su coste y ausencia de información para aquellas superficies que no están en pantalla.
- No he conseguido que funcione bien.
  - ☹

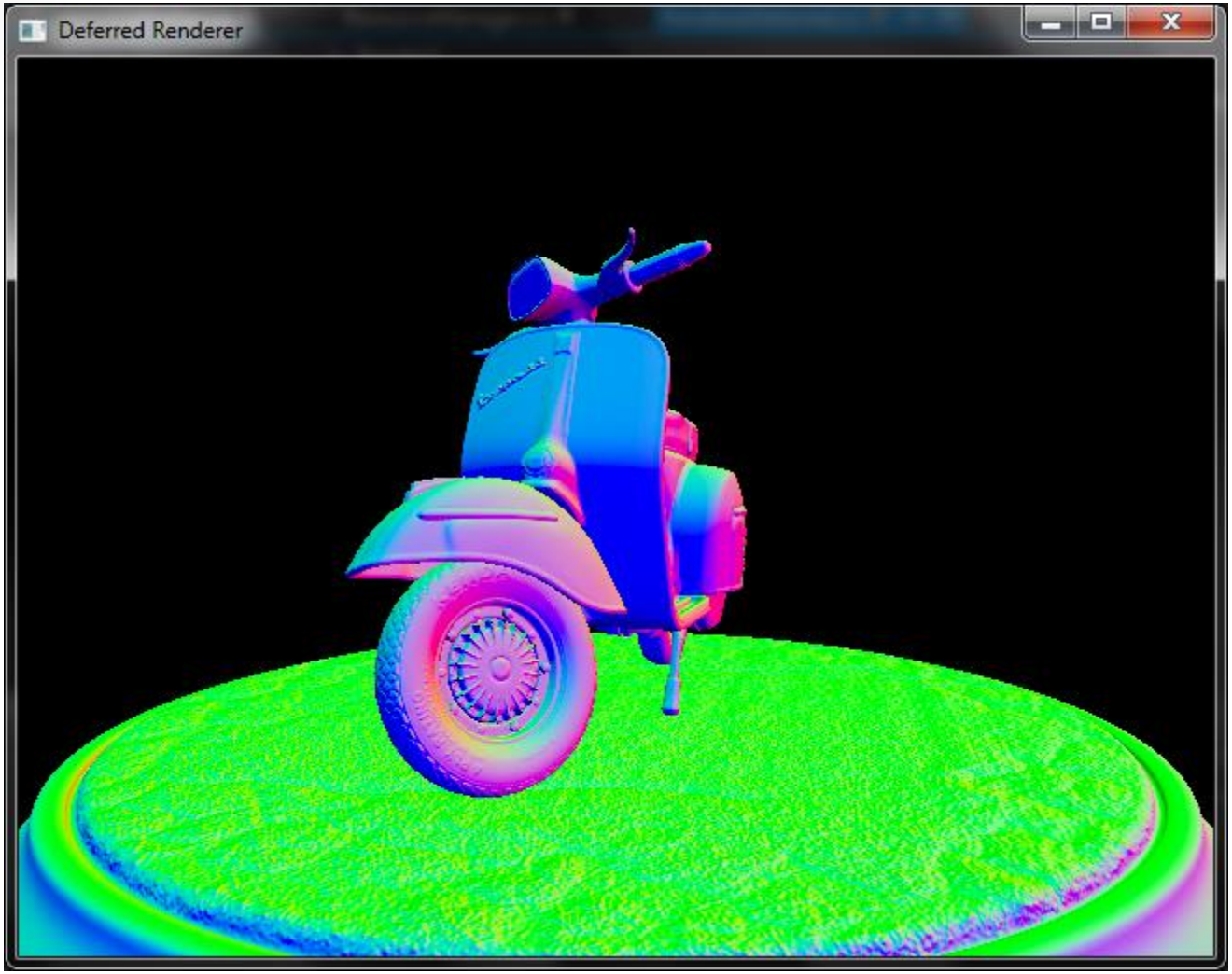
Sombreado Diferido (Deferred Shading)

RESULTADOS Y CONCLUSIÓN



Sombreado Diferido

Posición

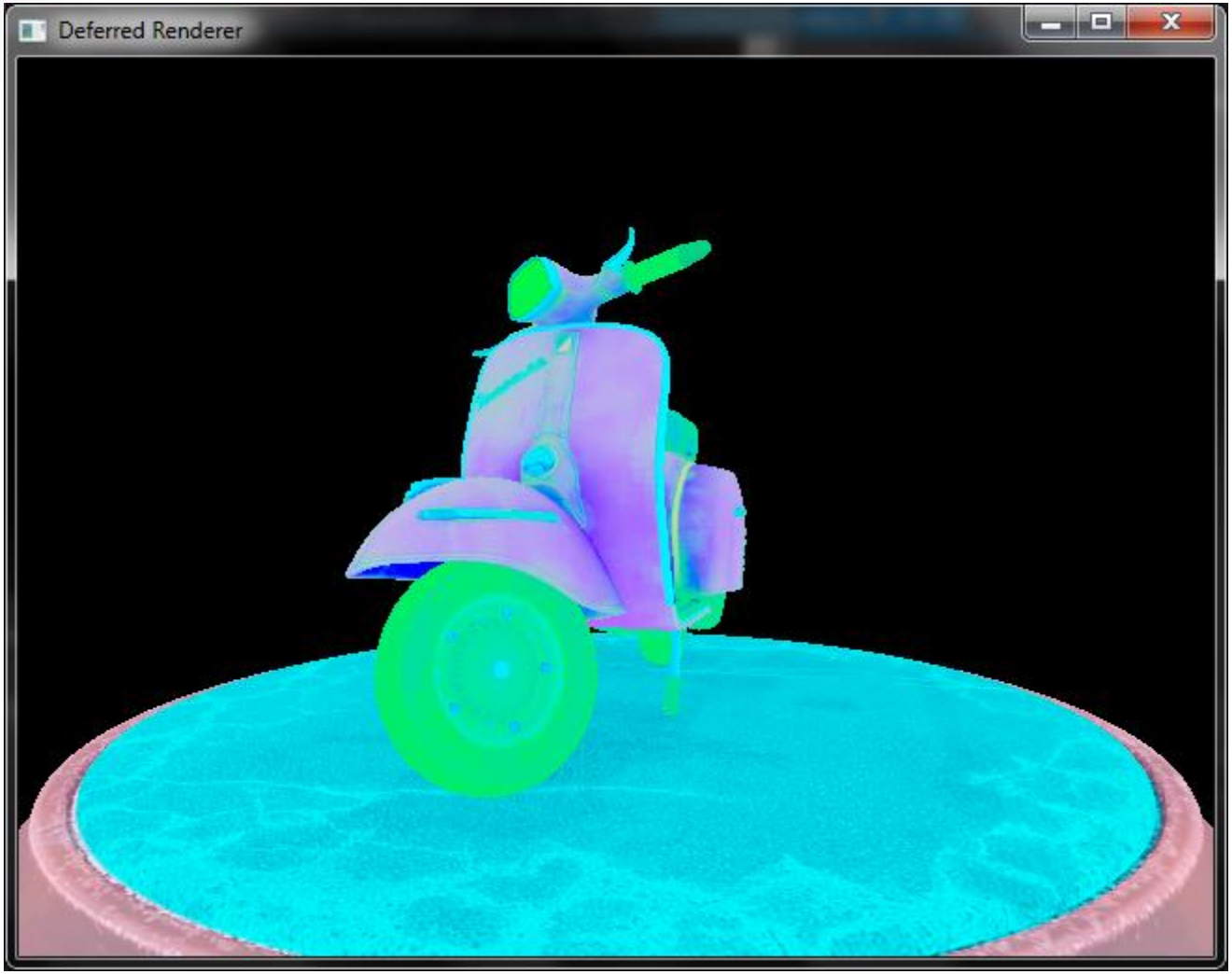


Sombreado Diferido

Normales

Sombreado Diferido

Albedo

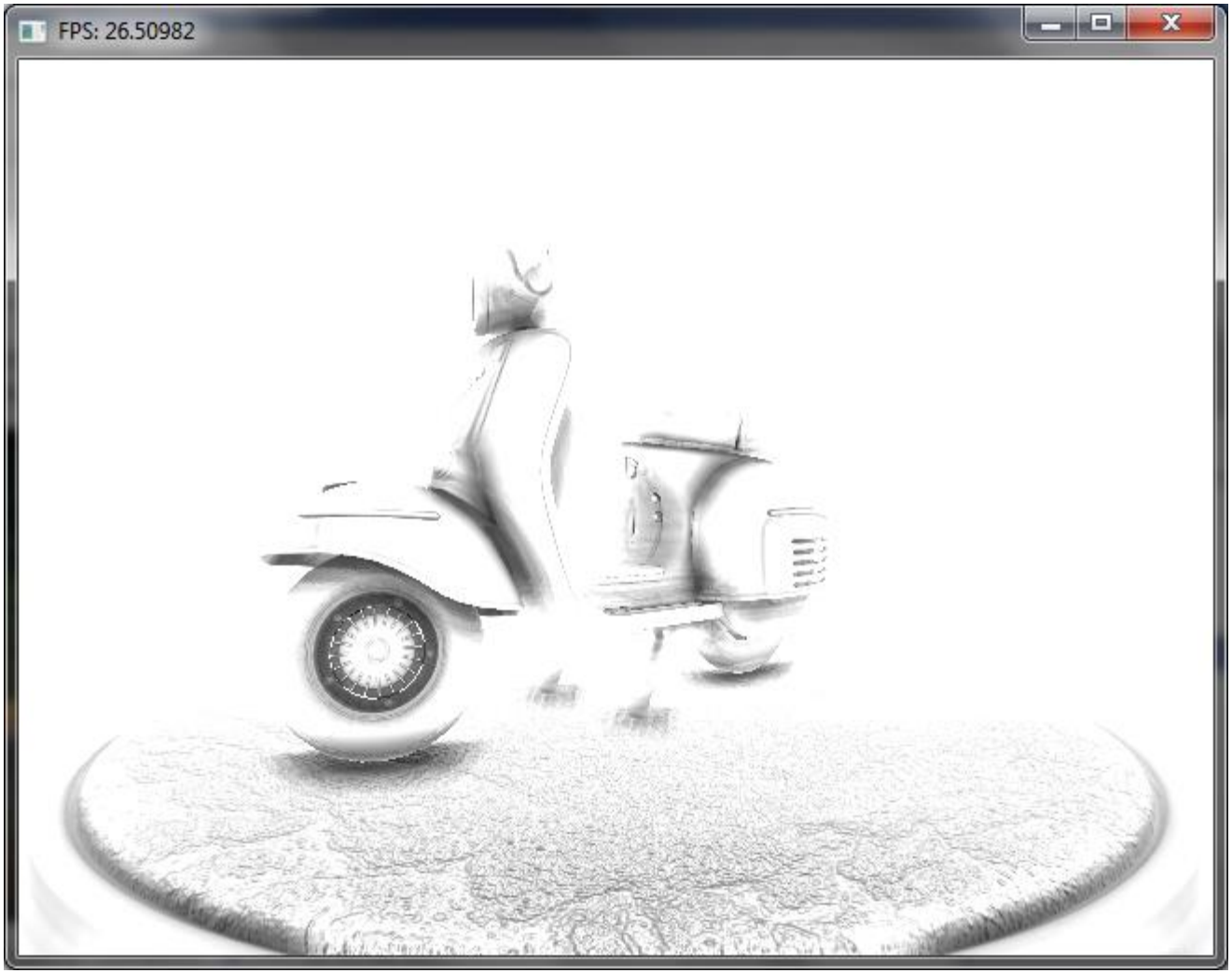


Sombreado Diferido

Parámetros del material

Sombreado Diferido

Composición final

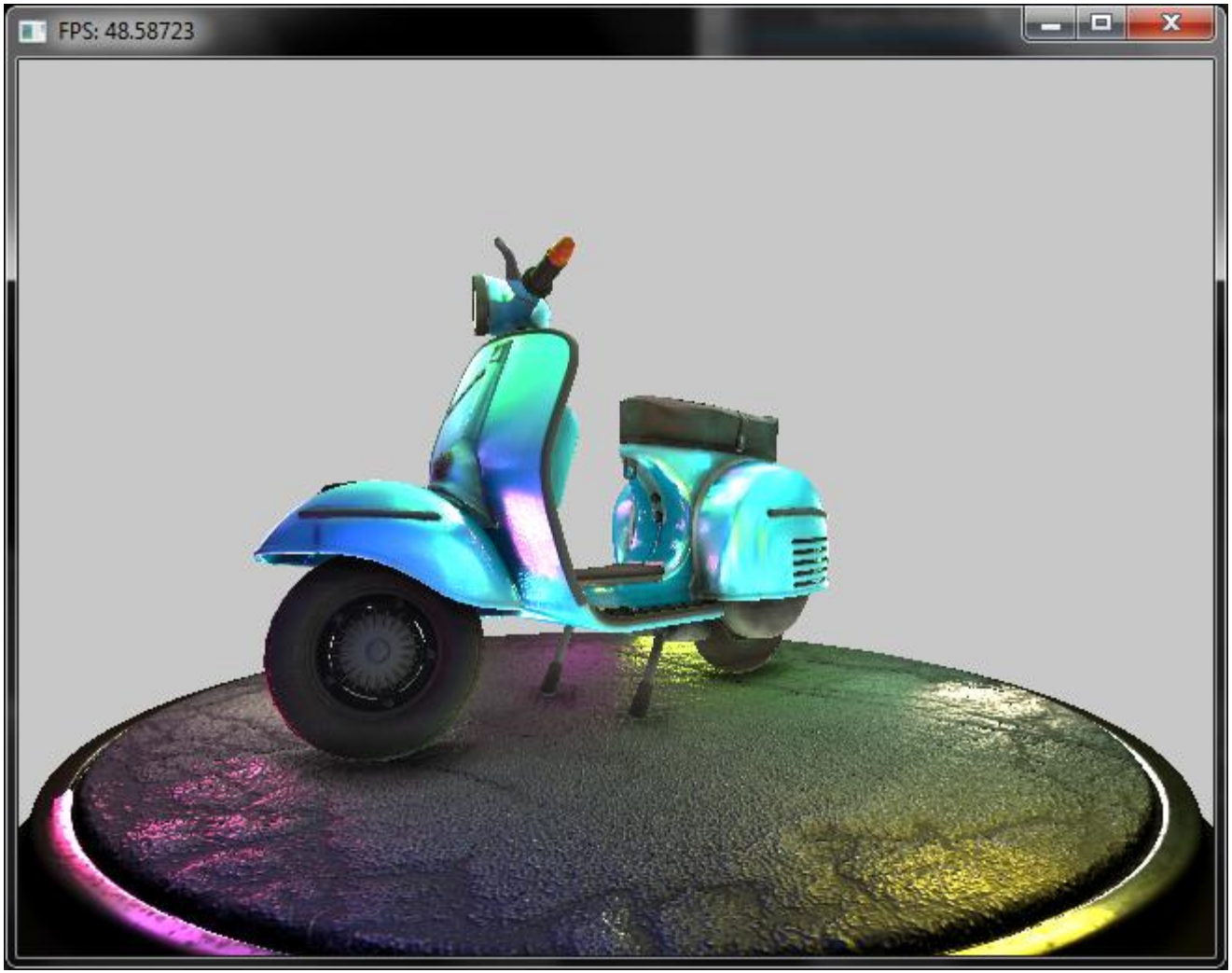


Post-Proceso

SSAO

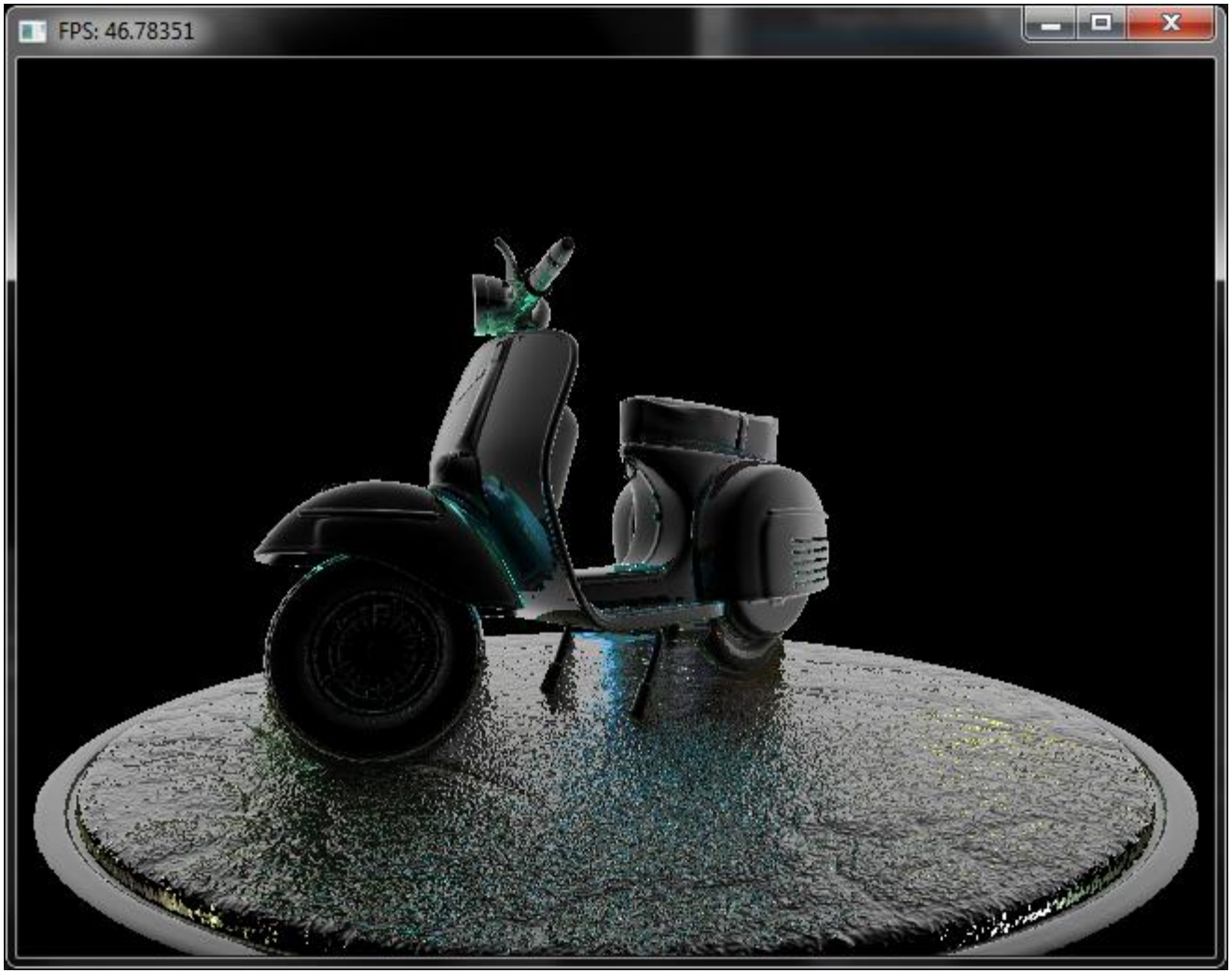
Post-Proceso

SSAO



Post-Proceso

SSAO



Post-Proceso

SSR

Post-Proceso

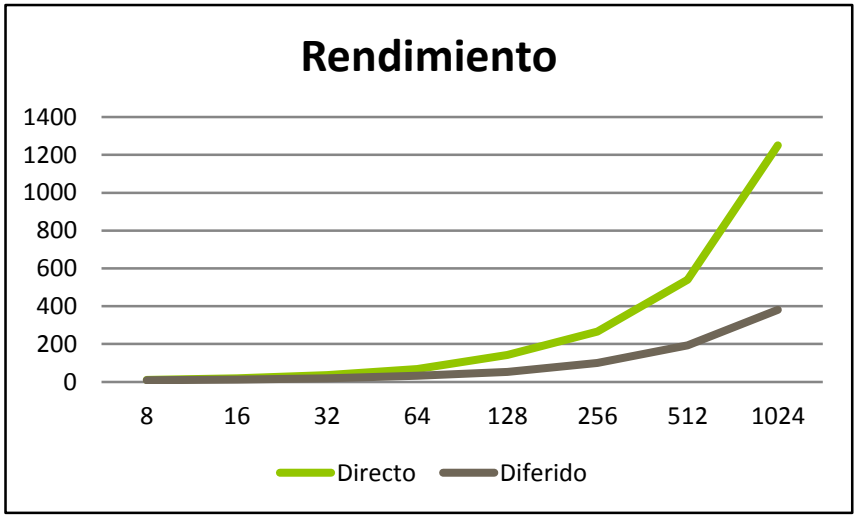
SSR

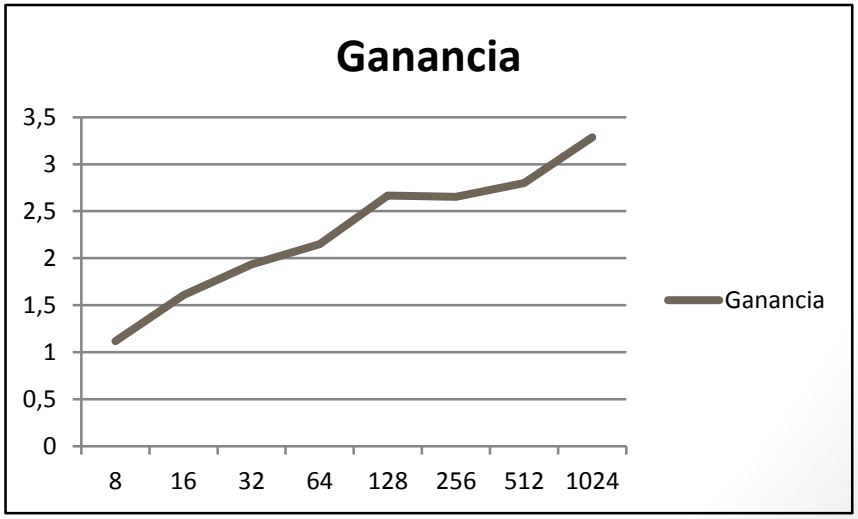
Post-Proceso

Todo junto

Comparación de rendimiento

N. Luces	Directo	Diferido
8	96.73	108.16
16	51.35	82.53
32	28.63	55.40
64	14.55	31.25
128	7.01	18.69
256	3.76	9.97
512	1.85	5.18
1024	0.8	2.63





Conclusiones

- Es una técnica relativamente sencilla de implementar.
  - Además tener directamente toda la información para los efectos de Post-Proceso es invaluable.
- Puede además suponer una mejora de rendimiento importante en escenas complejas.
- Por supuesto, se puede mejorar.
  - *Deferred Lighting*.
  - *Tile-Based Deferred Rendering*.
- Por lo general ha sido una experiencia gratificante.
  - Con las lecciones aprendidas, espero en un futuro portar el código a C++ y mejorarlo.

Gracias

Si queréis el código, sólo tenéis que pedírmelo.