

MÁSTER EN COMPUTACIÓN GRÁFICA, REALIDAD VIRTUAL Y SIMULACIÓN

RENDERIZADO DE TELAS BASADO EN ASPECTOS FÍSICOS (PBR) PARA UN ENTORNO INTERACTIVO WEB

AUTOR: ANXO BABÍO RODRÍGUEZ
TUTOR: DR. IVÁN ALDUÁN ÍÑIGUEZ

27 de septiembre de 2020

Resumen

Con la llegada del API gráfica WebGL y posteriormente WebGL 2.0 han aparecido librerías de alto nivel, como ThreeJs, Babylon, AFrame, Filament, etc., que proporcionan a los desarrolladores un interfaz que cubre la mayor parte de casos de uso del día a día y ofrece una experiencia de desarrollo intuitiva a costa de ocultar detalles de implementación de bajo nivel.

No obstante, el cometido de este proyecto es renderizar tejidos sobre un entorno web interactivo de la forma más realista posible y los modelos estándar basados en física no consiguen, por lo general, una representación lo suficientemente precisa para esta labor. Las imágenes generadas por este motor de renderizado sirven como previ-visualización de imágenes en alta fidelidad generadas por un motor de trazado de rayos, por lo que, además de realismo, debe buscarse la mayor coherencia posible entre estos dos motores.

Basado en el estudio teórico de los fundamentos técnicos del renderizado basado en aspectos físicos (PBR) y el análisis de diferentes modelos de Bidirectional Reflectance Distribution Function (BRDF) para la representación de materiales, este trabajo presenta un material especializado en tejidos integrado dentro de una arquitectura de servicios en la nube. Para dar soporte a este nuevo material en los servicios en la nube, se han extendido el modelo de datos de los materiales y desarrollado la interfaz que proporciona acceso a ellos. El modelo de BRDF para telas se ha implementado sobre ThreeJs, extendiendo la librería con un nuevo material que ofrece la misma interfaz que los materiales nativos de la librería.

Los resultados presentados ofrecen comparativas entre el material de telas y el material PBR estándar de ThreeJs y entre las imágenes generadas por este nuevo modelo e imágenes de referencia renderizadas por el motor de trazado de rayos. En la comparativa con el material de ThreeJs se aprecia un mayor grado de realismo en el nuevo modelo, además frente a las imágenes con el motor de trazados de rayos se aprecia poca discrepancia.

Marco colaborativo

Este proyecto educativo se desarrolla en estrecha colaboración con Seddi, empresa dedicada a la simulación gráfica que trabaja en la creación de gemelos digitales de tejidos. La empresa cuenta con departamentos de investigación e ingeniería para crear un conjunto de soluciones que permitan a diferentes agentes de la industria textil, como empresas de manufactura, diseñadores, patronistas o comerciales, trabajar y colaborar en tiempo real sobre una plataforma web.

Las tecnologías de digitalización de Seddi se sustentan sobre los departamentos de investigación, que se dividen en dos grupos según el tipo de propiedades a representar, ópticas o mecánicas y que se corresponden con los departamentos de captura óptica y renderizado, y captura mecánica y simulación respectivamente. El proceso de digitalización comienza con la captura, en el que se utiliza la tecnología desarrollada en los departamentos de captura óptica y mecánica para analizar muestras de tejido real, que luego se utilizarán en los modelos de simulación y renderizado desarrollados en sendos departamentos para dar vida a la réplicas digitales.

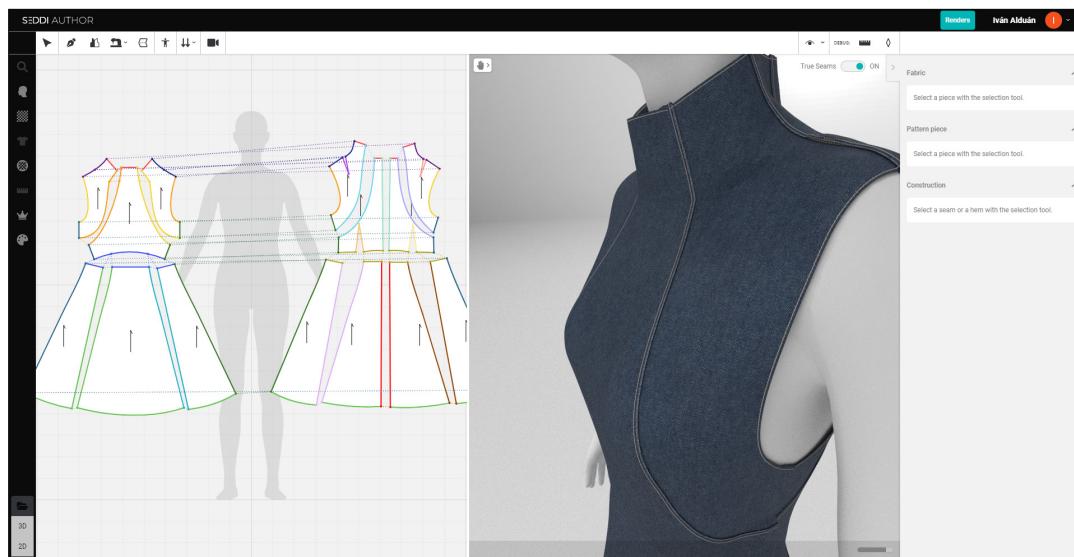


Figura 1: Vista de los entornos de trabajo 2D y 3D de Author

Los departamentos de ingeniería se encargan de crear la infraestructura e integrar las tecnologías desarrolladas en el área de investigación para exponer su funcionalidad al usuario en las diferentes plataformas de Seddi. Este Trabajo Fin de Máster (TFM) se enmarca bajo el contexto de Author, un departamento dentro del área de ingeniería dedicado a la creación de una plataforma que permita a los involucrados en el proceso de diseño, prototipado y construcción de una prenda, trabajar y colaborar en una plataforma en la nube, utilizando las réplicas digitales obtenidas durante el proceso de captura.

Author es una herramienta de edición que permite importar, crear y editar patrones, definir los detalles constructivos de la prenda, como el orden de costura, dobladillos, pinzas, etc., y ver en tiempo real una representación 3D realista del prototipo. Para ello proporciona al usuario principalmente dos contextos de trabajo: un espacio 2D en el que trabajar el patronaje y detalles de confección de la prenda, y un contexto 3D que permite previsualizar en tiempo real una simulación de la prenda sobre un avatar digital. La previsualización 3D es lo suficientemente precisa como para permitir trabajar con fluidez durante el proceso de diseño, no obstante, para fases que requieran un mayor grado derealismo y detalle en la construcción de la prenda, se ofrece la posibilidad de utilizar los servicios de simulación y renderizado en la nube de Seddi para generar una imagen de trazado de rayos renderizada progresivamente y transmitida en tiempo real al cliente web.

Índice general

Resumen	I
Marco colaborativo	II
1. Introducción	1
1.1. Motivación	1
1.2. Objetivos	2
1.3. Guía de la memoria	3
2. Estado del arte	4
2.1. Componentes y términos del BRDF basado en microfacetas	4
2.1.1. Componente difusa	4
2.1.2. Componente especular	5
2.2. Luz indirecta en sistemas de renderizado interactivos	6
2.2.1. Cálculo de la componente difusa	6
2.2.2. Cálculo de la componente especular	6
2.3. Modelos de renderizado interactivos de referencia	6
3. PBR	7
3.1. Unidades radiométricas	7
3.2. Ecuación de render	9
3.3. BRDF	10
3.3.1. Teoría de microfacetas	11
3.3.2. Tipos de BRDFs	12
3.4. Modelo de Cook-Torrance	13
3.4.1. Componente difusa	14
3.4.2. Componente especular	14
3.5. Modelo de Disney 2012	16
3.5.1. Componente difusa	18
3.5.2. Componente especular	19
3.6. Iluminación indirecta en entornos interactivos	21
3.6.1. Componente difusa	22
3.6.2. Componente especular	23
4. Desarrollo de infraestructura y servicios web	26
4.1. Servicios en la nube de Seddi	27
4.1.1. Implementación del modelo de datos	27
4.1.2. Actualización de las interfaces de servicios REST	29
4.2. Motor de renderizado de Author	30
4.2.1. Arquitectura del sistema de renderizado de ThreeJs	30

4.2.2. Materiales PBR en ThreeJs	31
5. Modelos de iluminación para tejidos	37
5.1. Integración de una nueva clase de material en ThreeJs	37
5.2. Integración del nuevo <i>shader</i> sobre ThreeJs	40
5.2.1. Componente difusa	40
5.2.2. Componente especular	42
5.2.3. Iluminación indirecta	43
5.3. Resultados	47
6. Conclusiones y trabajo futuro	51

Índice de figuras

1.1. Imágenes del motor de renderizado hiperrealista en el videojuego The Order: 1886 [DN13]	2
3.1. La superficie de la base del cono extraído de la esfera representa 1_{sr}	7
3.2. Esquema explicativo de un ángulo sólido diferencial.	8
3.3. La superficie de la base del cono extraído de la esfera representa 1_{sr}	8
3.4. La superficie de la base del cono extraído de la esfera representa 1_{sr}	9
3.5. Imagen publicada originalmente en el trabajo de Kajiya [Kaj86]	10
3.6. Diagrama de la luz incidente y luz reflectada	11
3.7. Superficie a escala microscópica	12
3.8. Esquema de diferentes modelos de BRDF según su origen	13
3.9. Diferentes tipos de materiales representados por el modelo de Cook-Torrance [Coo81]	14
3.10. Modelos de distribución de microfacetas frente a la muestra de referencia	15
3.11. Representación gráfica de la función de Fresnel	16
3.12. Representación gráfica de la sombra y el enmascaramiento en la función de geometría	16
3.13. Muestras de materiales MERL, junto a un esquema explicativo de las imágenes muestreadas	17
3.14. Modelo para metálicos y dieléctricos de Disney	17
3.15. Modelo de Disney	18
3.16. El cálculo de iluminación indirecta necesita hallar la irradiancia proveniente en todas direcciones w_i sobre la semiesfera Ω	21
3.17. Mapa de irradiancia como <i>cubemap</i>	23
3.18. Niveles de <i>mipmap</i> del mapa de entorno prefiltrado en función de la rugosidad del material	24
3.19. <i>BRDF integration map</i>	24
4.1. Editor de patrones 2D y editor 3D de Author	26
4.2. Esquema simplificado de comunicaciones y servicios en Seddi.	27
4.3. Modelo de datos de los recursos que soportan materiales.	28
4.4. Documentación de los recursos SurfaceMaterial y TextureStack en Swagger.	29
4.5. Documentación de los recursos SurfaceMaterial y TextureStack en Swagger.	30
4.6. Estructuras de datos de ThreeJs que se encargan de la gestión de programas de WebGL	31
5.1. Integración de la nueva clase <i>MeshClothMaterial</i>	38
5.2. Comparativa <i>MeshPhysicalMaterial</i> , <i>MeshClothMaterial</i>	38

5.3. Canales RG (BRDF GGX [Wal07]), B (BRDF para telas [Guy13]) y la suma de ellos, RGB, del <i>BRDF integration map</i>	45
5.4. Comparativa <i>MeshPhysicalMaterial</i> (fila superior) contra <i>MeshClothMaterial</i> (fila inferior) incrementando progresivamente la intensidad de luz global	47
5.5. Interpolación de un color de <i>sheen</i> aumentando su luminosidad frente a un color base constante.	48
5.6. Interpolación de un color de <i>sheen</i> aumentando su saturación frente a un color base constante.	48
5.7. Interpolación de <i>subsurfaceIntensity</i> utilizando un color base y color de <i>subsurface</i> blancos.	48
5.8. Interpolación de un color de <i>subsurface</i> frente a un color base constante.	49
5.9. Interpolación del color de <i>sheen</i> en el eje <i>x</i> frente a interpolación del color base en el eje <i>y</i>	49
5.10. Tejido de canvas renderizado con <i>MeshPhysicalMaterial</i> , <i>MeshClothMaterial</i> y el servicio de renderizado de trazado de rayos de Seddi	50
5.11. Tejido de sarga renderizado con <i>MeshPhysicalMaterial</i> , <i>MeshClothMaterial</i> y el servicio de renderizado de trazado de rayos de Seddi	50

Capítulo 1

Introducción

Los tejidos son estructuras compuestas por gran cantidad de elementos cuyas propiedades físicas e interacciones entre sí determinan las propiedades ópticas y mecánicas de su superficie. Estas características hacen que la simulación de tejidos sea, aún a día de hoy, un reto desafiante dentro del campo de los gráficos por ordenador.

Sectores como el del diseño industrial o la arquitectura llevan años beneficiándose de herramientas que permiten simular y previsualizar sus productos utilizando réplicas digitales. Sin embargo, en el sector textil, atributos determinantes para la evaluación de una prenda, como el brillo, la caída o su elasticidad requieren de una simulación hiper-realista que no ofrecen las soluciones dedicadas a tal fin existentes en el mercado.

Este TFM se desarrolla bajo el amparo de Seddi, empresa que desarrolla tecnología para la captura y simulación de tejidos con un alto nivel de detalle para la creación de réplicas digitales que permitan a la industria de la moda digitalizar su proceso de trabajo y beneficiarse de lo que esto implica: mayor velocidad de iteración, ubicuidad, disminución de los costes de producción, menor impacto medioambiental, entre otros beneficios.

El trabajo se integra dentro del contexto Author, una solución que utiliza los servicios en la nube de Seddi para ofrecer un entorno de diseño y prototipado de prendas que utiliza las réplicas digitales del tejido. En concreto, el propósito es tratar de aumentar el grado de realismo del motor de renderizado web y ofrecer una mayor coherencia con las imágenes generadas bajo demanda por el motor de renderizado progresivo en la nube de Seddi.

1.1. Motivación

Las técnicas de visualización sobre entornos interactivos han evolucionado desde algoritmos sencillos para representar el comportamiento de la luz en una escena 3D a algoritmos basados en física, más complejos y que permiten alcanzar un grado mayor de realismo.

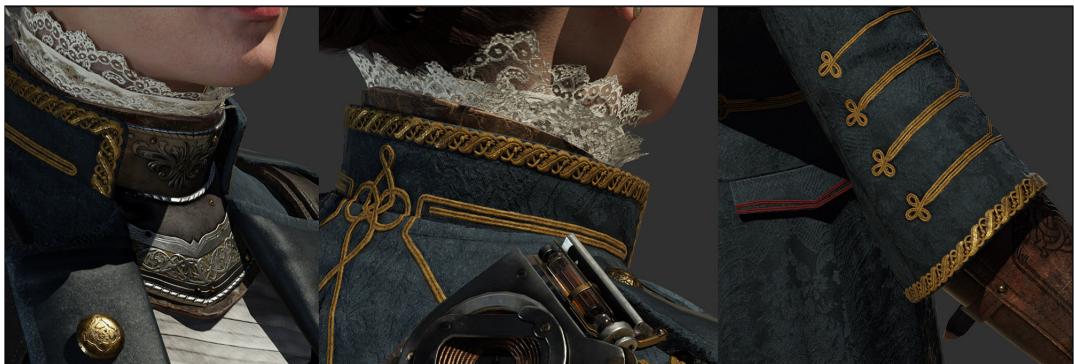


Figura 1.1: Imágenes del motor de renderizado hiperrealista en el videojuego The Order: 1886 [DN13]

La intención de este proyecto es profundizar en las base teóricas de estas técnicas y aprender a integrarlas dentro de un escenario real de trabajo, por lo que la colaboración con Seddi supone la ocasión perfecta para la puesta en práctica de los conocimientos teóricos adquiridos.

La infraestructura de servicios en la nube sobre la que se sostiene Author me ha ofrecido la oportunidad de aprender sobre la arquitectura, modelos de datos, sistemas de almacenamiento y protocolos de comunicación en sistemas distribuidos en red. Por otra parte, la implementación de esta nueva funcionalidad en ThreeJs, la librería de 3D utilizada en Author, me ha permitido entender como se integran estas técnicas dentro de la arquitectura de un motor de renderizado.

Además, como empleado de Seddi en el departamento de Author, entender mejor la infraestructura, el modelo de datos y las técnicas de renderizado hiperrealistas que se utilizan en la empresa además de tratar de mejorar la experiencia de uso del contexto 3D de la plataforma, son alicientes extra a las motivaciones citadas anteriormente.

1.2. Objetivos

La finalidad de este trabajo es proporcionar un nuevo modelo de material al contexto 3D de Author que represente mejor las propiedades ópticas de los tejidos y sea más coherente con las imágenes de referencia generadas por un motor de trazado de rayos. Para la consecución de esta tarea se trazan los siguientes objetivos generales:

- (I) Dar soporte al nuevo material bajo la infraestructura de servicios en la nube de Seddi.
- (II) Extender la librería ThreeJs para integrar un material específico para la reproducción de tejidos basado en aspectos físicos o *Physically Based Rendering* (PBR).

Que se componen de una serie de objetivos específicos:

- 1) Adquirir conocimiento teórico sobre los motores de renderizado basados en aspectos físicos.

- 2) Adquirir conocimiento teórico sobre diferentes modelos de reflectancia o *Bidirectional Reflectance Distribution Functions* (BRDFs) y sistemas de iluminación global en entornos interactivos de visualización hiperrealista.
- 3) Definición del modelo para dar soporte al nuevo material dentro de la arquitectura de servicios en la nube de Seddi.
- 4) Desarrollo de la interfaz de servicios en la nube para ofrecer soporte a este material.
- 5) Entender a bajo nivel los detalles de implementación del sistema de renderizado de ThreeJs.
- 6) Extender ThreeJs para integrar en su librería de materiales nativos el nuevo modelo de material orientado a la reproducción de tejidos.
- 7) Integrar el código GLSL dentro del sistema de renderizado de ThreeJs para dar soporte al modelo de BRDF utilizado por el nuevo material.

1.3. Guía de la memoria

Capítulo 1. Introducción En este capítulo se explican brevemente los objetivos y motivaciones del proyecto, así como las aportaciones de este trabajo.

Capítulo 2. Estado del arte El segundo capítulo ofrece una visión del estado del arte en cuanto a los trabajos sobre materiales e iluminación en tiempo real que guardan relación con la solución presentada en este proyecto.

Capítulo 3. PBR En este capítulo se explican las técnicas estándar en la actualidad para la representación de materiales realistas y sirve como base teórica para los desarrollos presentados en los capítulos 4 y 5.

Capítulo 4. Infraestructura y servicios web El capítulo 4 explica la integración de la propuesta sobre los servicios en la nube de Seddi y la integración de un nuevo material sobre la librería de WebGL ThreeJs.

Capítulo 5. Modelos de iluminación para tejidos En este capítulo se explica en detalle el modelo de BRDF a integrar para el material de tejidos y su implementación en ThreeJs. Además se presentan resultados en los que se demuestran los nuevos efectos modelados por el material, así como comparativas del material estándar PBR de la librería frente al nuevo.

Capítulo 6. Discusión y trabajo futuro Comentarios sobre el planteamiento, desarrollo, resultados obtenidos y posibles aplicaciones. También se establecen líneas de trabajo futuro.

Capítulo 2

Estado del arte

El primer modelo de BRDF basado en aspectos físicos fue presentado por Robert Cook y Ephraim Sparrow en 1967 [Tor+67] y presenta por primera vez el modelo de microfacetas en el que se basan todos los modelos de BRDF realistas presentados posteriormente. Ya en 1981, el modelo de Cook-Torrance [Coo81] combina las ideas del modelo de microfacetas y la función de distribución de las normales de Beckmann [Bec+63] y presenta nuevas ideas como que solo las microfacetas orientadas en dirección al vector medio entre la luz y el puntos de vista contribuyen a la reflexión especular, la conservación de la energía entre las componentes difusa y especular o la separación en el modelo de materiales según su conductividad: metálicos o dieléctricos.

2.1. Componentes y términos del BRDF basado en microfacetas

El modelo de Cook-Torrance [Coo81], explicado en detalle en el Capítulo 3, permite representar una amplia variedad de materiales y, aunque con variaciones en sus términos, es el modelo de referencia de BRDF basado en aspectos físicos. A continuación se detallan los trabajos sobre el cálculo de sus componentes y términos.

2.1.1. Componente difusa

Las superficies lambertianas son superficies ideales que reflejan la luz incidente en todas direcciones por igual, por lo que su radiancia es independiente del punto de vista. Este modelo de superficie ideal de reflexión difusa fue presentado por Johann Heinrich Lambert 1760 [Lam60] y es el utilizado en los modelos de Torrance-Sparrow [Tor+67] y Cook-Torrance [Coo81].

El modelo lambertiano supone una aproximación lo suficientemente precisa para representar una amplia variedad de materiales y es utilizado todavía en la actualidad por gran cantidad de motores de renderizado realista, entre ellos, ThreeJs. Sin embargo, el término no tiene en cuenta la conservación de la energía y cuando se utiliza en conjunto con un BRDF basado en microfacetas, genera oscurecimiento en las zonas de mayor rugosidad, que se acentúa a medida que aumenta el ángulo de incidencia de la luz.

El modelo de Oren-Nayar [Ore+94], basado en la teoría de microfacetas, presenta por primera vez un término difuso físicamente plausible y es el modelo en el que se inspira el factor de corrección sobre la componente difusa que simula el efecto de absorción y reflexión interna presentado en este trabajo.

2.1.2. Componente especular

La aproximación de Cook-Torrance a la integral del modelo de microfacetas se fundamenta en tres términos, una función de distribución de las normales, una función de geometría y la ecuación de Fresnel, a continuación un breve repaso de las diferentes implementaciones de estos términos.

Término D

Mientras que el modelo de Torrance-Sparrow [Tor+67] presenta una distribución gaussiana, el modelo de Cook-Torrance [Coo81] utiliza el término presentado en 1963 por Beckmann [Bec+63].

El modelo de *Generalized-Trowbridge-Reitz* (GTR) [Bur12] es el utilizado por ThreeJs para modelar la componente especular de sus materiales PBR, mientras que el presentado por Bruce walter [Wal07] es uno de los más extendidos en la actualidad.

Por otra parte, el modelo de Estévez y Kulla [ACE17], basado en el trabajo de Ashikhmin y Premoze [Ash+00], presenta un nuevo parámetro que ofrece mayor control sobre la componente especular, muy útil para ciertos tipos de tejido y que se utiliza en el modelo para tejidos presentado en este trabajo.

Término F

El término F es una aproximación de la función de Fresnel. La primera aproximación a la función de Fresnel fue la presentada por Cook-Torrance. ThreeJs utiliza el término presentado en el modelo de Schlick [Sch94], más tarde Largarde [SL14] y Gotanda [Got12] presentaron sucesivas aproximaciones.

Término G

El término de geometría fue presentado en el modelo de microfacetas de Cook-Torrance [Coo81] y se utiliza para describir la cantidad de microfacetas que quedan en sombra y depende típicamente de los parámetros de rugosidad y distribución de las microfacetas.

En la actualidad los modelos más usados son los basados en el método de Smith [Smi67], que consiste en separar la fórmula en dos partes que utilizan la misma ecuación pero una utilizando el vector de vista y otra la de la luz, representando los fenómenos de sombreado y enmascaramiento respectivamente. De entre éstos términos de geometría, cabe destacar el presentado en el trabajo de Schlick-Beckmann [Sch94] o el utilizado en Unreal [Kar13], que remapea el valor de rugosidad para conseguir unos

valores más intuitivos para los artistas. El modelo presentado por Neubelt [DN13] es el utilizado para el BRDF de telas integrado sobre ThreeJs.

2.2. Luz indirecta en sistemas de renderizado interactivos

La ecuación para el cálculo de iluminación indirecta, analizada en detalle en el Capítulo 4, fue presentada por primera vez por Kajiya [Kaj86], sin embargo la parte integral de la ecuación supone un cálculo recursivo que no puede ser resuelto en tiempo real. Es por ello, que para los motores de renderizado en interactivos, se utilizan técnicas que permiten conseguir éste efecto, a través de precalculos o aproximaciones analíticas.

2.2.1. Cálculo de la componente difusa

Para calcular la irradiancia sobre un punto debida al entorno, Paul Debevec [Deb86] presentó la técnica de mapas de irradiancia, mientras que en 2001, Ravi Ramamoorthi y Pat Hanrahan [Ram+01] presentan la técnica de esféricos harmónicos, que permite la compresión de información utilizando una representación de frecuencias de color frente a espacio.

2.2.2. Cálculo de la componente especular

La aproximación de sumas parciales [Kar13], que se comenta en detalle en el Capítulo 3 permite el cálculo de la reflexión especular separando el cálculo de la integral en dos partes, la integral de la radiancia sobre un punto y el cálculo del BRDF. La técnica utiliza un mapa de entorno prefiltrado y un *look-up table* (LUT) que almacena los resultados precomputados del BRDF. Además existen funciones analíticas como la presentada por Epic Games [Kar14] y que se utiliza en ThreeJs como aproximación a la solución a la integral del BRDF.

2.3. Modelos de renderizado interactivos de referencia

Hoy día el modelo de referencia es el de Disney [Bur12] [Bur15], conocido como "*Principled*", y que se analiza en detalle en el Capítulo 4. Además los modelos de Electronic Arts [SL14], Unreal [Kar13] o Filament [Guy13] son implementaciones completas de modelos de BRDF que integran iluminación global y que sirven de inspiración a otros motores de renderizado en tiempo real.

Capítulo 3

PBR

PBR o *physically based rendering* es el término que se utiliza para nombrar al grupo de técnicas basados en un modelo físico. La intención de los sistemas de renderizado PBR en entornos interactivos es conseguir aproximaciones plausibles y poco costosas de la dispersión de la luz al golpear sobre una superficie. Las principales ventajas de este tipo de algoritmos son el realismo, la consistencia bajo diferentes condiciones de luz y su manejo intuitivo por parte de los artistas.

Aunque este trabajo se centra en el tratamiento de materiales basados en aspectos físicos, adicionalmente, otros elementos de la escena, como la cámara o las luces, pueden estar basados en modelos físicos para ofrecer un mayor grado de realismo.

3.1. Unidades radiométricas

Los modelos de BRDF analizados en este trabajo son basados en física y utilizan las unidades radiométricas para medir las interacciones físicas de la luz con el entorno. A continuación un listado de las unidades básicas de radiometría:

Ángulo sólido

El ángulo sólido (w) es la unidad de medida del área de una proyección sobre una esfera de radio 1 es el equivalente tridimensional a la medida en radianes de un arco sobre un círculo. Su unidad es el estereoradián (sr), que representa el área de un ángulo sólido sobre la esfera de radio unitario Ω que contiene 4π radianes.

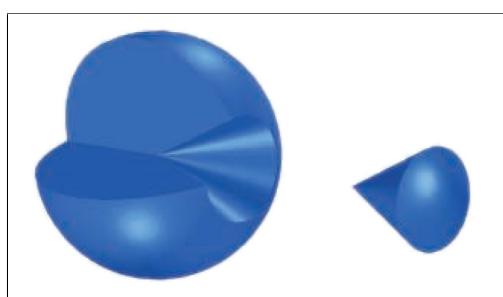


Figura 3.1: La superficie de la base del cono extraído de la esfera representa 1 sr .

Ángulo sólido diferencial

Es la unidad que se utiliza como área de medida en el BRDF para representar la cantidad de luz que incide en una superficie. Un ángulo sólido diferencial ($d\omega$) es el área contenida entre incrementos muy pequeños en coordenadas esféricas sobre los ángulos θ y ϕ de la semiesfera Ω .

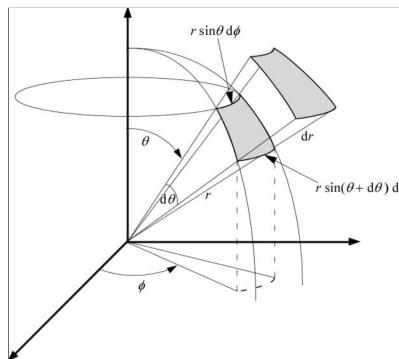


Figura 3.2: Esquema explicativo de un ángulo sólido diferencial.

Flujo radiante

El flujo radiante (Φ) es la unidad que mide la cantidad de energía de energía transportada por las ondas de luz por unidad de tiempo. En los motores de renderizado utilizan esta unidad para expresar la cantidad total de energía emitida por una fuente de luz y se mide en vatios (W).

Irradiancia

La irradiancia (E) es la cantidad de energía (flujo radiante) de tiempo por unidad de superficie y se mide en vatios por metro cuadrado (W/m^2). En los motores de render se utiliza para medir la cantidad de luz que incide sobre un punto desde todas las direcciones.

$$E = \frac{d\Phi}{dA} \quad (3.1)$$

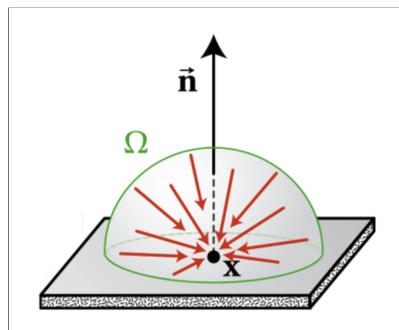


Figura 3.3: La superficie de la base del cono extraído de la esfera representa 1 sr.

Intensidad radiante

Representa el flujo radiante emitido por una fuente de luz en una dirección determinada y se mide expresa como vatios por estereoradián (W/sr).

$$I = \frac{d\Phi}{d\omega} \quad (3.2)$$

Radiancia

La radiancia (L) se puede considerar como la combinación de irradiancia e intensidad radiante y define la dirección y la intensidad radiante recibida por una superficie. La ecuación de render, igual que la cámaras, sensores o nuestros ojos miden la radiancia que llega desde las superficies del entorno al punto de vista. Su unidad son los vatios por metro cuadrado por estereoradián (W/m^2sr)

$$L = \frac{d^2\Phi}{dA_{proj} d\omega} \quad (3.3)$$

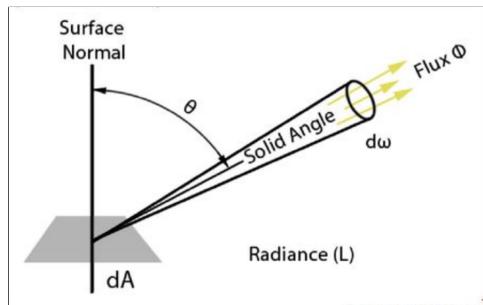


Figura 3.4: La superficie de la base del cono extraído de la esfera representa $1sr$.

3.2. Ecuación de render

Presentada por primera vez en el trabajo de Jim Kajiya [Kaj86], el propósito de la ecuación de render es conocer el valor de radiancia que llega a la cámara en una dirección por cada pixel de la cámara. Para ello se utiliza la ecuación de reflectancia, que depende de la luz que llega al punto, el coseno del ángulo con el que incide la luz y el BRDF, que modela el comportamiento de la luz al rebotar sobre la superficie.

$$L_o(p, w_o) = L_e(x, w_o) + \int_{\Omega} f_r(p, w_i, w_o) L_i(p, w_i) n \cdot w_i dw_i \quad (3.4)$$

Ecuación 3.4: Ecuación de render

Siendo $L_o(p, w_o)$ la luz reflejada por la superficie, $L_e(x, w_o)$ la luz emitida por la superficie, que está fuera de la integral porque es independiente de la luz reflejada por la

superficie. $\int_{\Omega} [...] dw$ que representa que la operación se repite para cada ángulo sólido dentro de la semiesfera $f_r(p, w_i, w_o)$, el BRDF, $L_i(p, w_i)$, la luz que llega al punto de la superficie que estamos evaluando y $n \cdot w_i$ el factor de atenuación de la luz con respecto a la normal.

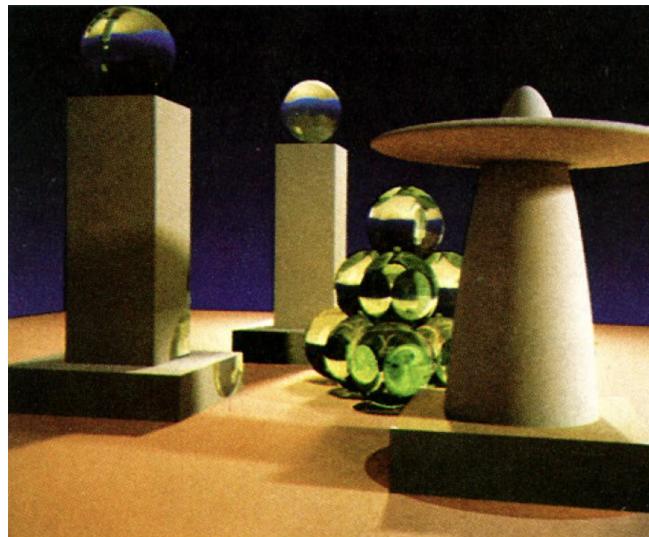


Figura 3.5: Imágen publicada originalmente en el trabajo de Kajiya [Kaj86]

El coste de computación de la radiancia debida al conjunto del entorno (iluminación indirecta o global) es muy alto, por lo que tradicionalmente los sistemas de renderizado interactivos calculaban únicamente la radiancia debida a los emisores de luz de la escena (iluminación directa o local). Sin embargo recientemente se han presentado técnicas que permiten la aproximación de este efecto que se analizan en la última sección de este capítulo.

3.3. BRDF

El BRDF, o función de distribución bidireccional de reflectividad, es la parte de la ecuación $f_r(p, w_i, w_o)$ que describe el comportamiento de la luz al golpear sobre la superficie y se utiliza para modelar propiedades físicas de los diferentes materiales.

Es una función estadística que calcula el ratio de luz que incide sobre un material que es reflejada en dirección a la cámara, para ello toma como argumentos la dirección de la luz, w_i , y la dirección de la vista, w_o , la normal de la superficie.

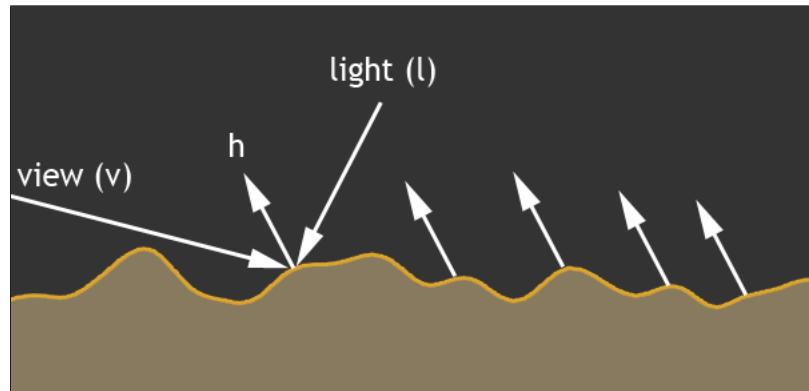


Figura 3.6: Diagrama de la luz incidente y luz reflectada

Los BRDFs basados en aspectos físicos, además de estar basados en la teoría de microfacetas, se caracterizan por otras dos propiedades: el principio de reciprocidad de Helmholtz y la ley de conservación de la energía.

- El principio de reciprocidad de Helmholtz establece que el BRDF debe de ser simétrico, lo que supone que invertir las direcciones de entrada y salida del BRDF no debe afectar al resultado.

$$f(x, w_i, w_o) = f(x, w_o, w_i) \quad (3.5)$$

- El principio de conservación de la energía determina que la energía dispersada por la superficie no puede ser mayor que la energía incidente.

$$\forall w_i \int_{\Omega} f(p, w_i, w_o) n \cdot w_i dw_o \leq 1 \quad (3.6)$$

Estos dos principios no siempre se cumplen en los sistemas de renderizado realista en entornos interactivos, que pueden presentar aproximaciones que sacrifiquen aspectos físicos del modelo, siempre y cuando no resulten en artefactos visuales apreciables, en favor de tiempos de respuestas menores.

3.3.1. Teoría de microfacetas

Aunque a escala macroscópica podamos considerar una superficie como lisa, ninguna superficie es completamente lisa a nivel microscópico. La teoría de microfacetas utiliza una representación estadística para modelar estas pequeñas irregularidades.

La teoría de microfacetas, considera la superficie de reflexión como una superficie compuesta por una matriz de superficies más pequeñas que la longitud de onda de la luz, completamente reflectantes, llamadas microfacetas, y cuyas diferentes orientaciones determinan la rugosidad de la superficie.

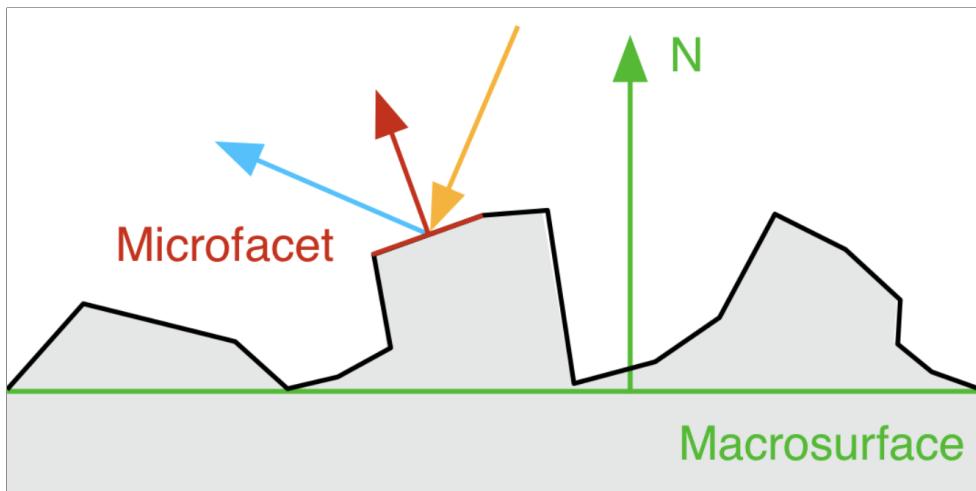


Figura 3.7: Superficie a escala microscópica

Si las normales de las microfacetas están completamente alineadas con el vector medio entre la vista y la luz (vector h), las reflexiones serán mas definidas, asimilándose a un espejo, mientras la variación de la normal de las microfacetas con respecto al vector h dará como resultado una reflexión más difusa.

3.3.2. Tipos de BRDFs

Existen diferentes tipos de BRDFs en función de las propiedades físicas del material que se quiera representar. Los materiales metálicos utilizan un BRDF diferente a los dieléctricos, esto es así porque los efectos de absorción y dispersión, son completamente diferentes en función de esta propiedad.

En los materiales metálicos, la reflexión especular es de color, la luz refractada es absorbida por el propio material y no existen efectos de dispersión de la luz bajo la superficie, mientras que en los materiales dieléctricos, o no metálicos, el aspecto final del material está determinado por una combinación de los fenómenos de absorción y dispersión de la luz.

Por otra parte, podemos diferenciar entre materiales anisotrópicos, como el metal pulido o el pelo, cuyas propiedades de reflexión cambian al rotar sobre la superficie alrededor de su vector normal. Aunque casi todos los materiales son, en cierta medida, anisotrópicos, la aportación de esta propiedad sobre la apariencia final suele ser tan baja que se desprecia y se utilizan modelos isotrópicos.

Además, según su origen, podemos distinguir principalmente entre tres tipos de modelos: empíricos, teóricos y experimentales. Los modelos empíricos no están basados en física, si no que son simplificaciones de menor coste que aproximan un determinado fenómeno, ejemplos de modelos empíricos son los modelos de Phong [Pho73] o el de Blinn-Phong [Bli77]. Los modelos teóricos, como los basados en microfacetas, se basan en observaciones sobre el comportamiento de la luz y su interacción sobre las superficies para crear un sistema de ecuaciones que describan este comportamiento sobre un modelo matemático. Por último, los modelos experimentales, el de Schlick

[Sch94] se basan en resultados obtenidos en estudios de campo con luz a partir de los cuales se crean funciones que se ajusten a los datos obtenidos.

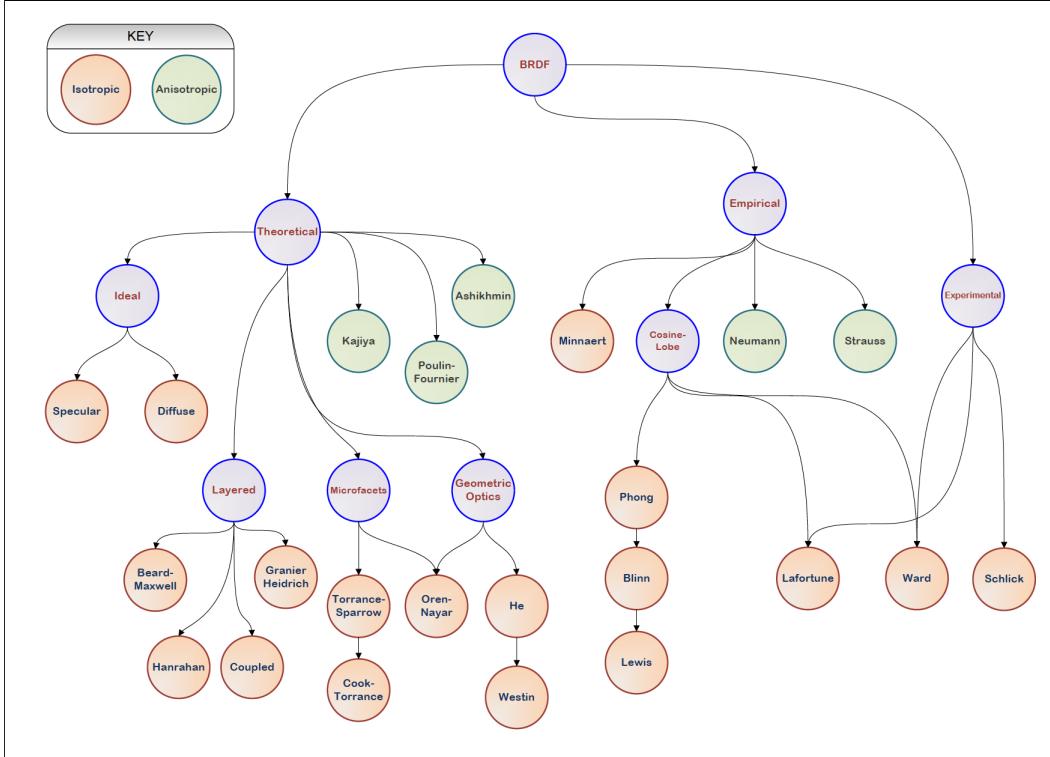


Figura 3.8: Esquema de diferentes modelos de BRDF según su origen

3.4. Modelo de Cook-Torrance

El modelo de BRDF de Cook-Torrance [Coo81] presenta la forma que conocemos a día de hoy: los términos de distribución de las normales, geometría y Fresnel y el factor de corrección en el denominador. Aunque con cambios en sus términos, ideas como la conservación de la energía, que solo las microfacetas cuyas normales apuntan en dirección h contribuyen a la reflexión especular o la diferencia entre materiales metálicos y dieléctricos, siguen presentes en la mayoría de modelos basados en aspectos físicos que se utilizan en la actualidad.

$$f_r = k_d f_{lambert} + k_s f_{cook-torrance}$$

$$\text{siendo } k_d + k_s = 1 \quad (3.7)$$

Ecuación 3.7: Ley de conservación de la energía en el modelo de Cook-Torrance [Coo81]

Este trabajo popularizó los modelos basados en aspectos físicos al permitir representar con gran realismo materiales eléctricos y dieléctricos con diferentes niveles de

rugosidad. Su principal desventaja es su parametrización, que no resulta intuitiva y dificulta el trabajo de los artistas de 3D.

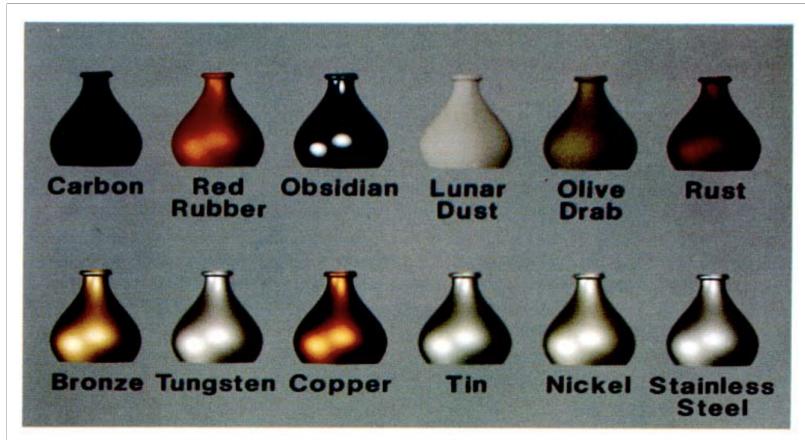


Figura 3.9: Diferentes tipos de materiales representados por el modelo de Cook-Torrance [Coo81]

3.4.1. Componente difusa

Típicamente la componente difusa, utiliza el modelo de Lambert, que asume una distribución completamente uniforme a lo largo de la superficie:

$$f_{Lambert} = \frac{diffuse}{\pi} \quad (3.8)$$

3.4.2. Componente especular

La componente especular es una función compuesta de otras tres términos y un factor de normalización en el denominador.

$$f(l, v) = \frac{F(w_i, h)G(w_i, h, w_o)D(h)}{4(n \cdot w_i)(n \cdot w_o)} \quad (3.9)$$

El denominador es una característica típica de los modelos de microfacetas y es un factor de corrección debido al cambio de espacio entre el espacio local de las microfacetas y el espacio local de la macrosuperficie. Para modelos que no incluyen este factor, se puede aplicar un factor de sombreado directamente multiplicando el término de geometría por el factor $\frac{1}{4(n \cdot w_i)(n \cdot w_o)}$.

D, término de distribución de las normales

Es la función de distribución de las normales y es un modelo estadístico que se encarga de representar la rugosidad de una superficie y es el término que afecta en mayor grado a la forma y tamaño del brillo especular. Típicamente el parámetro de rugosidad se utiliza para representar la cantidad de microfacetas alineadas con la normal

h , cuanta mayor sea la cantidad de microfacetas alineadas, más lisa parecerá la superficie.

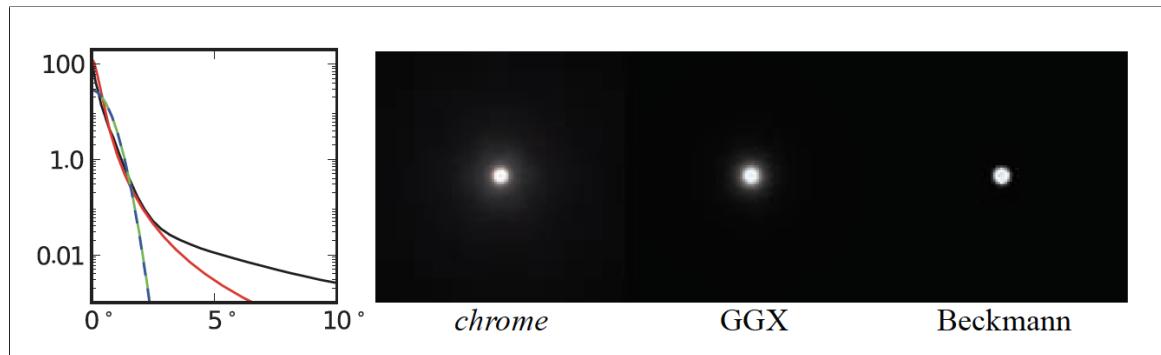


Figura 3.10: Modelos de distribución de microfacetas frente a la muestra de referencia

La función de distribución de microfacetas utilizada por este modelo es la presentada por Beckmann [Bec+63]:

$$D(h) = \frac{1}{\pi \alpha^2 (n \cdot h)^4} e^{\frac{(n \cdot h)^2 - 1}{m^2 (n \cdot h)^2}} \quad (3.10)$$

F, término de Fresnel

Representa la radiancia reflejada por un material según el ángulo de incidencia de la luz. Para la mayoría de los materiales no metálicos, las reflexiones son más intensas cuando el ángulo de incidencia es muy agudo.

En la fórmula original de Fresnel, la energía depende del índice de refracción η y el índice de refracción κ , coeficientes no siempre conocidos para un material. La aproximación a la función de Fresnel, otra de las principales contribuciones de este modelo, elimina κ de la parametrización.

$$F(\theta) = \frac{1}{2} \frac{(b - c)^2}{(b + c)^2} \left\{ 1 + \frac{[c(b + c) - 1]^2}{[c(b - c) + 1]^2} \right\}$$

siendo $b^2 = \eta^2 + c^2 - 1$ y $c = \cos(\theta)$ (3.11)

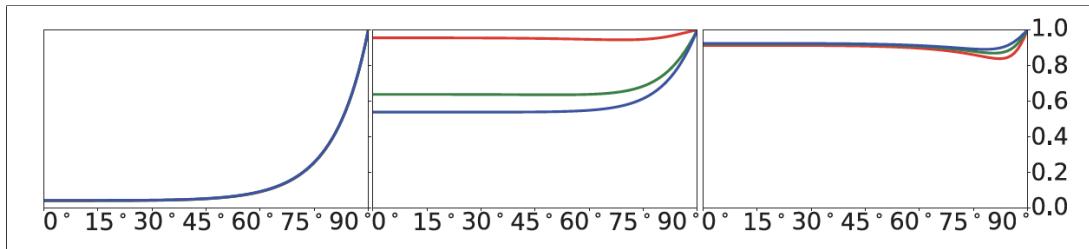


Figura 3.11: Representación gráfica de la función de Fresnel

G, término de geometría

Para estimar la cantidad de microfacetas que contribuyen a la reflexión especular, además de tener en cuenta la densidad de microfacetas cuyas normales están orientadas en dirección a h , se ha de calcular teniendo en cuenta la cantidad de rebotes ocluidos por las propias microfacetas. Este trabajo utiliza para ello el término presentado en el modelo de Torrance-Sparrow [Tor+67].

$$G(w_i, w_o) = \min \left\{ 1, \frac{2(n \cdot h)(n \cdot w_i)}{(w_i \cdot h)} \frac{2(n \cdot h)(n \cdot w_o)}{(w_o \cdot h)} \right\} \quad (3.12)$$

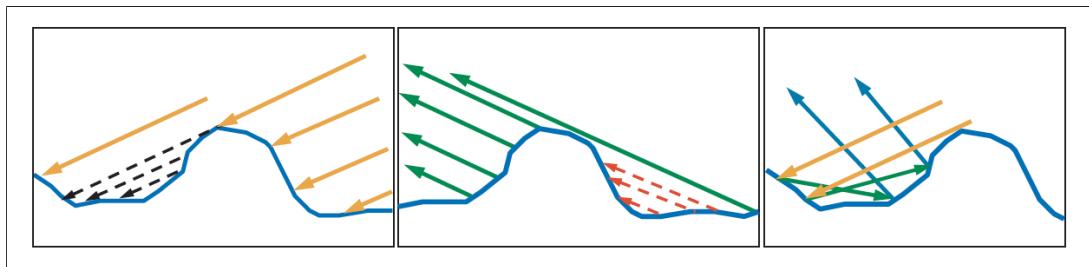


Figura 3.12: Representación gráfica de la sombra y el enmascaramiento en la función de geometría

3.5. Modelo de Disney 2012

Los modelos de BRDF basados en aspectos físicos ofrecen gran versatilidad al conseguir representar gran cantidad de materiales de una forma muy realista. Por contra, resultan en una definición del material compleja, con gran cantidad de parámetros que resultan poco intuitivos para los artistas.

En 2012, Disney presenta su modelo conocido como *Principled* [Bur12], que, a pesar de ser basado en física, sigue una serie de directivas como tener la menor cantidad de parámetros posibles, ofrecer nombres y rangos de parámetros amigables para los artistas u ofrecer combinaciones robustas y plausibles para todas sus posibles combinaciones. Desde entonces, los materiales basados en física han ganado popularidad y se han introducido progresivamente en la industria de las películas y videojuegos,

convirtiendo al modelo de Disney en el estándar de facto en la actualidad.

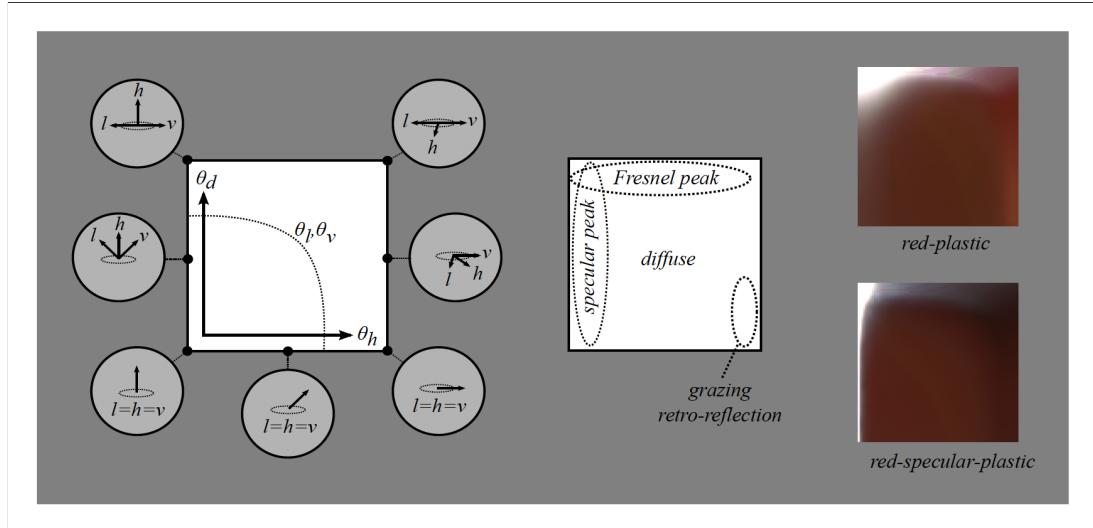


Figura 3.13: Muestras de materiales MERL, junto a un esquema explicativo de las imágenes muestreadas

El modelo teórico de Disney está basado en las observaciones sobre la tabla MERL [Mat+03], que provee una lista de 100 materiales muestreados y representados en una gráfica cartesiana, cuyo eje x representa pasos discretos de rotación alrededor de la tangente del material y en el y , la rotación alrededor de la normal de la superficie.

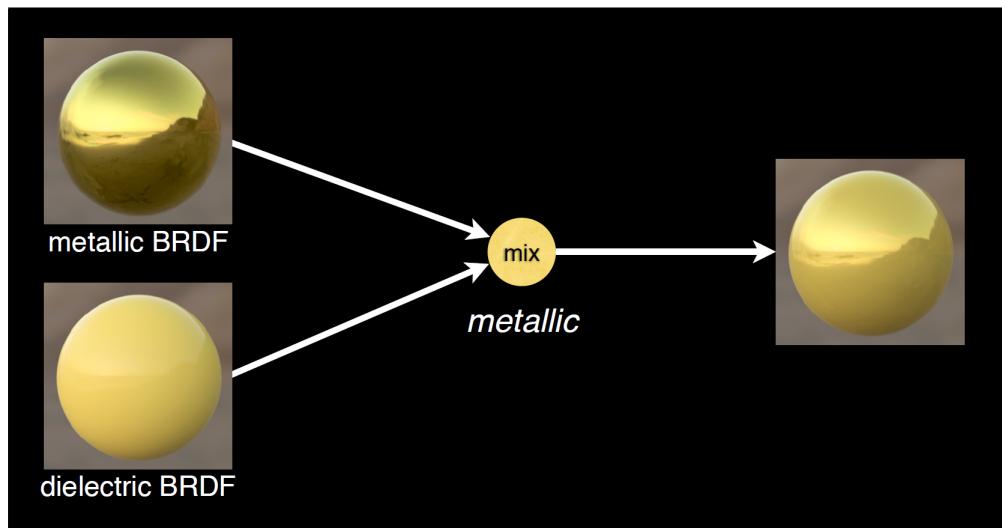


Figura 3.14: Modelo para metálicos y dieléctricos de Disney

En base a estas observaciones, el modelo *Principled* de Disney, ajusta los términos del BRDF de Cook-Torrance [Coo81] primando la expresividad artística sobre lo físicamente plausible. El modelo aporta dos nuevos lóbulos que, a diferencia del lóbulo primario pueden variar mucho en forma y tamaño, además tanto los materiales

isotrópicos como anisotrópicos se representan con el mismo modelo, que utiliza una interpolación entre los dos BRDFs, por lo que resulta muy atractivo para los artistas, pudiendo representar una enorme variedad de materiales con muy pocos parámetros.

La lista de parámetros final se compone de: *baseColor*, *subsurface*, *metallic*, *specular*, *specularTint*, *roughness*, *sheen*, *sheenTint*, *clearcoat*, *clearcoatGloss*

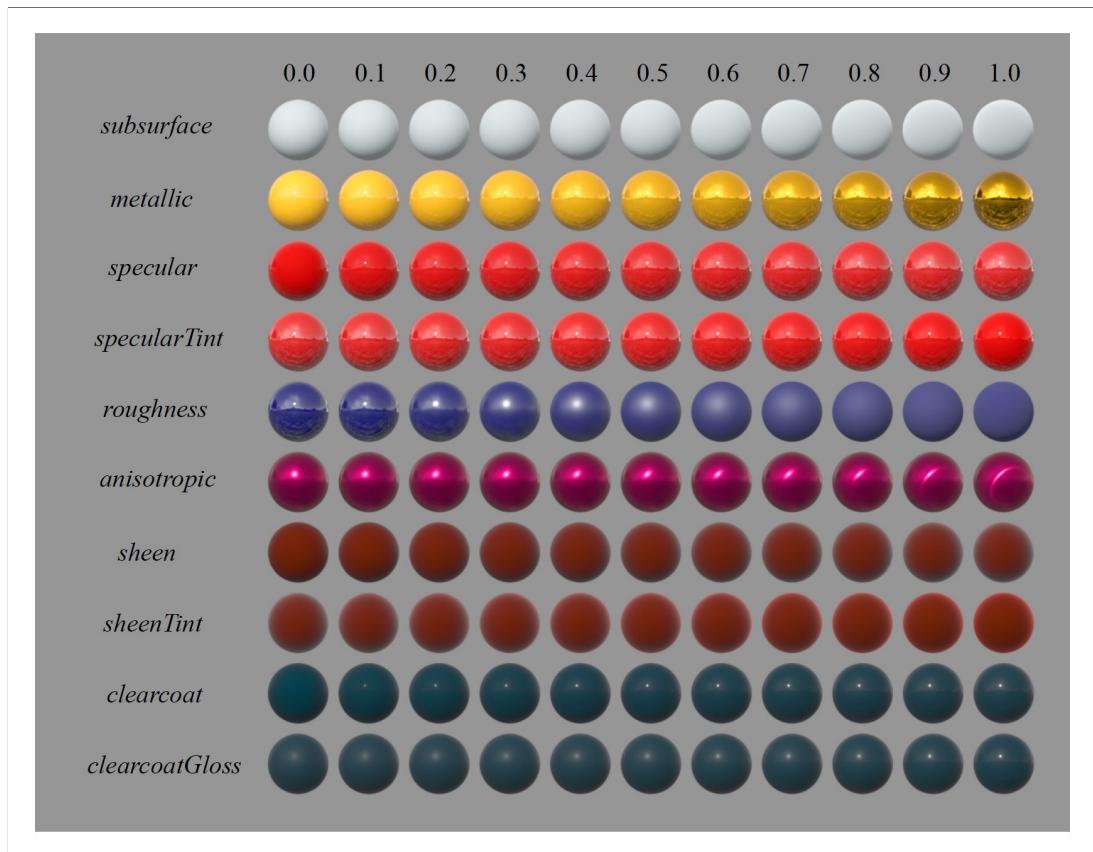


Figura 3.15: Modelo de Disney

3.5.1. Componente difusa

En base a las observaciones sobre los datos de MERL, el modelo difuso parece muy oscuro hacia los bordes, y aplicando la función de Fresnel, para intentar conseguir un modelo físicamente plausible, parece oscurecer más los bordes, acentuando el problema.

Disney []disney utiliza una modificación del término Fresnel-Schlick conseguir una dispersión hacia delante y hacia detrás que depende del valor de rugosidad del material y se adapta mejor a los datos obtenidos de MERL [Mat+03].

$$f_d = \frac{baseColor}{\pi} \left(1 + (F_{D90} - 1)(1 - \cos\theta_{wi})^5 \right) \left(1 + (F_{D90} - 1)(1 - \cos\theta_{wo})^5 \right)$$

$$\text{siendo } F_{D90} = 0,5 + 2roughness \cdot \cos^2\theta_d \quad (3.13)$$

3.5.2. Componente especular

El modelo de Disney [Bur12] utiliza términos conocidos del BRDF de Cook-Torrance [Coo81] (DFG) e incorpora dos nuevos lóbulos: el de *sheen* y el de *clearcoat*. El lóbulo de *sheen* permite mayor control sobre la reflexión especular en los ángulos críticos, mientras que el lóbulo de *clearcoat* permite simular una capa superficial para imitar efectos como el barniz o recubrimientos transparentes.

D, término de distribución de las normales

La distribución GGX es equivalente a la de Trowbridge-Reitz, no tiene una cola lo suficientemente larga para la mayoría de materiales.

$$D_{TR} = \frac{c}{(\alpha^2 \cos^2\theta_h + \sin^2\theta_h)^2} \quad (3.14)$$

El modelo de Disney utiliza un exponente en el denominador que permite mayor control sobre el radio de la reflexión especular, llamando a éste término Generalized Trowbridge-Reitz, o GTR:

$$D_{GTR} = \frac{c}{(\alpha^2 \cos^2\theta_h + \sin^2\theta_h)^\gamma} \quad (3.15)$$

Los dos lóbulos, primario y secundario utilizan la distribución GTR. El lóbulo primario, que representa la reflexión del material base, utiliza $\gamma = 2$ y puede ser dieléctrico o metálico, isotrópico o anisotrópico. Por otra parte, el lóbulo secundario representa la capa de *clearcoat* sobre el material base, utiliza $\gamma = 1$ y suele ser isotrópico y no metálico.

F, término de Fresnel

Para el especular, la aproximación de Fresnel-Schlick [Sch94] es lo suficientemente precisa y mucho menos costosa que la ecuación completa de Fresnel.

$$F_{Schlick}(w_o, h) = F_0 + (1 - F_0)(1 - (n \cdot w_o))^5$$

$$\text{siendo } F_0 = \frac{(n - 1)^2}{(n + 1)^2} \text{ y } n = IOR \quad (3.16)$$

F_0 representa la reflectancia de una incidencia del mismo ángulo que la normal. θ_d es el ángulo entre el vector h , y el de vista w_o . Es acromáctico para los dieléctricos y cromático para los metales.

G, término de geometría

El término de geometría utilizado en el modelo Disney [Bur12] es el Smith-GGX [Wal07], remapeando el valor de *roughness* para evitar ganar demasiada energía hacia los bordes de materiales brillantes. El lóbulo primario utiliza un valor α de rugosidad, mientras que para el lóbulo de *clearcoat* se utiliza un valor fijo de $\alpha = 0,25$.

Los términos de geometría basados en la técnica de Smith [Smi67] separan la función de geometría en dos partes. La oclusión puede deberse a la sombra, cuando una microfaceta no es visible desde la dirección de la luz o al enmascaramiento, cuando no es visible desde la dirección de la vista.

$$G(l, v, h) = G_{GGX}(l)G_{GGX}(v)$$

$$G_{GGX}(v) = \frac{2(n \cdot v)}{(n \cdot v) + \sqrt{\alpha^2 + (1 - \alpha)^2(n \cdot v)^2}}$$

$$\alpha = (0,5 + roughness/2)^2 \quad (3.17)$$

Lóbulo de *sheen*

El lóbulo de *sheen* utiliza dos parámetros para ajustar la intensidad y el color del brillo especular en los ángulos críticos. Este control es muy útil para modelar efectos de retroreflexión característicos de algunos tipos de tejidos.

$$f(\textit{sheen}, \theta_d) = \textit{sheen} * ((1 - \textit{sheenTint}) + \textit{sheenTint} * \textit{tint}) * (1 - \cos \theta_d)^5 \quad (3.18)$$

Lóbulo de *clearcoat*

El lóbulo de *clearcoat* es otro lóbulo adicional que se controla con los parámetros *clearcoat* y *clearcoatGloss*. Este lóbulo utiliza el mismo BRDF que el lóbulo primario pero con modificaciones en sus términos para evitar añadir complejidad a la definición del material.

La función de distribución de las normales también utiliza $\gamma = 1$ en el modelo GTR.

$$D_{GTR} = \frac{c}{(\alpha^2 \cos^2 \theta_h + \sin^2 \theta_h)} \quad (3.19)$$

La función de Fresnel utiliza el índice de refracción fijo de 0.04, o lo que es lo mismo, $F_0 = 0,04$, el valor del poliuretano.

$$F_{Schlick}(w_o, h) = F_0 + (1 - F_0)(1 - (n \cdot w_o))^5$$

siendo $F_0 = 0,04$ (3.20)

Finalmente, el término de geometría, utiliza la misma función que el lóbulo primario (Smith-GGX [Wal07]), pero con un rugosidad fija de 0.25 para no añadir complejidad a la definición del material.

$$G_{GGX}(v) = \frac{2(n \cdot v)}{(n \cdot v) + \sqrt{0,0625 + 0,5625(n \cdot v)^2}} \quad (3.21)$$

3.6. Iluminación indirecta en entornos interactivos

La ecuación de render describe la radiancia de salida sobre un punto.

$$L_o(p, w_o) = \int_{\Omega} f_r(p, w_i, w_o) L_i(p, w_i) n \cdot w_i dw_i \quad (3.22)$$

Siendo Ω la semiesfera centrada en el punto sobre el que se calcula la irradiancia, f_r representa el BRDF, L_i , la irradiancia de la escena, y $n \cdot w_i$, el coseno entre la normal de la superficie y el de la luz incidente.

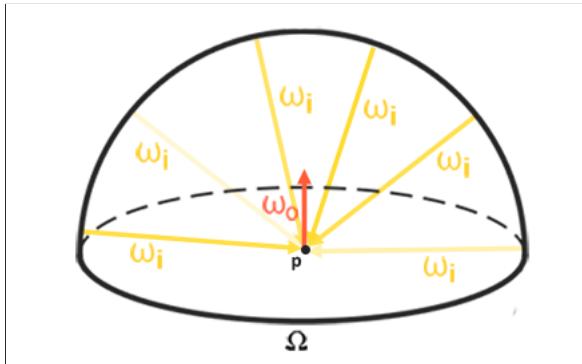


Figura 3.16: El cálculo de iluminación indirecta necesita hallar la irradiancia proveniente en todas direcciones w_i sobre la semiesfera Ω .

Calcular la parte integral de la ecuación requiere lanzar rayos no en una dirección sino desde todas las direcciones posibles sobre la semiesfera Ω , por lo que no es posible en tiempo real. En esta sección se repasan técnicas que precomputan los cálculos o utilizan funciones analíticas para lograr aproximar este efecto en entornos interactivos.

El primer paso es separar el BRDF en sus componentes especular y difusa, que permitirá la combinación de diferentes técnicas para calcular la solución de la ecuación de render.

$$L_o(p, w_o) = \int_{\Omega} (k_d \frac{c}{\pi} + k_s \frac{DFG}{4(w_o \cdot n)(w_i \cdot n)}) L_i(p, w_i) n \cdot w_i dw_i \quad (3.23)$$

Siendo k_s y k_d términos independientes, el cálculo de la integral se puede separar en partes, la integral de la componente difusa más la integral de la componente especular.

$$L_o(p, w_o) = \int_{\Omega} (k_d \frac{c}{\pi}) L_i(p, w_i) n \cdot w_i dw_i + \int_{\Omega} k_s \frac{DFG}{4(w_o \cdot n)(w_i \cdot n)} L_i(p, w_i) n \cdot w_i dw_i \quad (3.24)$$

3.6.1. Componente difusa

La solución de la integral de la irradiancia sobre la semiesfera Ω requiere samplear el entorno en todas las direcciones posibles. Es por ello que en tiempo real, la solución consiste en precomputar este cálculo.

Habiendo separado la ecuación para la componente difusa y especular, podemos observar que el término del difuso de Lambert es constante y lo podemos sacar de la integral.

$$L_o(p, w_o) = \int_{\Omega} (k_d \frac{c}{\pi}) L_i(p, w_i) n \cdot w_i dw_i = k_d \frac{c}{\pi} \int_{\Omega} (L_i(p, w_i) n \cdot w_i) dw_i \quad (3.25)$$

Las técnicas IBL (*Image Based Lighting*), permiten precalcular la irradiacia del entorno tratando el entorno como un gran emisor de luz, utilizando una imagen de referencia para aproximar la luz emitida por el entorno sobre el punto que estamos calculando.

Mapas de irradiancia

Los mapas de irradiancia utilizan *cubemaps* para muestrear en coordenadas esféricas en pasos discretos la radiancia procedente del entorno. La irradiancia para cada punto de la superficie es el precálculo de la integral de la semiesfera Ω orientada en dirección a la normal N . Una vez preconvolucionado el mapa de entorno, el resultado, conocido como mapa de irradiancia se almacena en una textura a la que se accede en tiempo de ejecución para consultar los resultados precalculados de irradiancia debida al entorno.



Figura 3.17: Mapa de irradiancia como *cubemap*

Esféricos harmónicos

La técnica de esféricos harmónicos [Ram+01] supone una optimización sobre los mapas de irradiancia, que, pese a ser una buena aproximación, utilizan *look-up textures*, que pueden ser operaciones relativamente costosas en dispositivos con un rendimiento limitado. Los esféricos harmónicos permiten una aproximación a la irradiancia debida al entorno comprimiendo la información en una representación de espacio frente a frecuencias. De esta forma, la información se almacena en la función de esféricos harmónicos y para consultarla, se utiliza su transformada inversa, que devuelve los datos a su representación espacial.

3.6.2. Componente especular

El método split-sum approximation presentado por Epic [Kar13], permite separar la integral en dos partes, la integral de la radiancia de la escena y la del BRDF. De forma similar a la componente difusa, precomputar los resultados y consultarlos y para calcular el resultado de la integral durante en tiempo real.

$$L_o(p, w_o) \approx \int_{\Omega} L_i(p, w_i) dw_i * \int_{\Omega} fr(p, w_i, w_o) n \cdot w_i dw_i \quad (3.26)$$

De forma similar al mapa de irradiancia, el mapa de entorno prefiltrado se precomputa con una convolución sobre el mapa de entorno. La convolución utiliza la distribución de normales de Cook-Torrance [Coo81] en función de la dirección de la vista, la normal y la rugosidad del material.

Dado que se desconoce la dirección de la vista al precomputar la radiancia, la aproximación asume $w_i = w_o$, lo que implica perder las reflexiones especulares en el ángulo crítico, pero generalmente se considera una aproximación lo suficientemente buena para entornos interactivos. Los resultados para diferentes niveles de rugosidad

se almacenan en diferentes *mipmaps* de la textura.



Figura 3.18: Niveles de mipmap del mapa de entorno prefiltrado en función de la rugosidad del material

La segunda parte de la integral se precomputa asumiendo una radiancia blanca pura en todas direcciones, que permite calcular todos los posibles valores del BRDF en función de la normal de la superficie, el punto de vista y el valor de rugosidad del material. El resultado se almacena en un *look-up texture* (LUT) que se conoce como *BRDF integration map* y contiene valores de la escala del Fresnel en el canal rojo de la textura y valores de la desviación del Fresnel en el canal verde. Este LUT se consulta en tiempo de ejecución, accediendo a sus valores utilizando rangos normalizados entre 0 y 1 en los dos ejes, utilizando el coseno entre la vista y la normal de la superficie para el eje *x* y valores de rugosidad del material en el eje *y*.

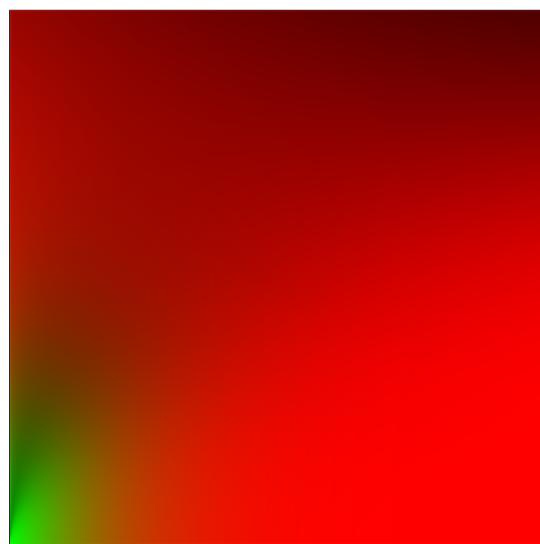


Figura 3.19: *BRDF integration map*

Finalmente el cálculo de la integral de la componente especular se ve reducido a

una consulta al mapa de entorno prefiltrado, otra la *BRDF integration map* y la combinación de sus resultados:

```
1 float lod = getMipLevelFromRoughness(roughness);
2 vec3 prefilteredColor = textureCubeLod(PrefilteredEnvMap, refVec, lod);
3 vec2 envBRDF = texture2D(BRDFIntegrationMap, vec2(NdotV, roughness)).xy;
4 vec3 indirectSpecular = prefilteredColor * (F * envBRDF.x + envBRDF.y)
```

Listing 3.1: Cálculo de la componente especular debida al entorno

Como alternativa al *BRDF integration map* existen función analíticas como la que utiliza ThreeJs presentada por Epic Games [Kar14] y basada en el trabajo de Dimitar Lazarov [Laz13]. Estas aproximaciones, aunque no cumplen con la ley de conservación de la energía, son computacionalmente baratas y suficientemente precisas como para ser utilizadas en soportes donde el rendimiento es prioritario.

Capítulo 4

Desarrollo de infraestructura y servicios web

Author es la plataforma de Seddi que permite a diseñadores, patronistas y demás agentes involucrados en el proceso de diseño y prototipado de una prenda trabajar con réplicas digitales de tejidos sobre un entorno colaborativo web.

Un API REST provee la información de los tejidos capturados por Seddi para su representación en el contexto de Author, que permite al usuario crear y editar patrones, detalles constructivos u ornamentos de una prenda. Este flujo permite el diseño de una prenda completamente sobre el cliente web, mientras que para las acciones más costosas, como la simulación o renderizado basado en trazado de rayos se utilizan los servicios en la nube de Seddi.

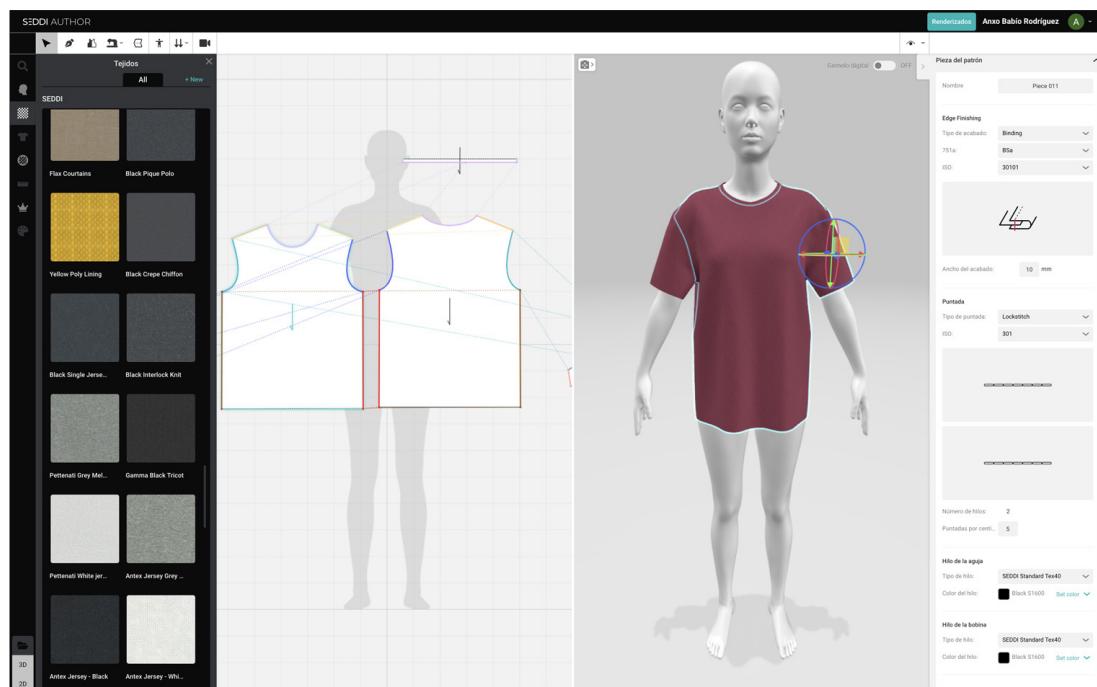


Figura 4.1: Editor de patrones 2D y editor 3D de Author

En la primera sección de este capítulo se detallan los cambios requeridos para dar soporte al nuevo modelo en la infraestructura de Seddi. Por otra parte, la segunda sección ofrece una visión general del sistema de renderizado de ThreeJs, el motor de renderizado que se utiliza en Author y sobre el que se integrará el nuevo material.

4.1. Servicios en la nube de Seddi

Los servicios en la nube utilizan una arquitectura distribuida, diferentes servicios que se comunican entre sí para realizar una tarea en conjunto. Esta arquitectura implica una mayor complejidad, teniendo que comunicar los diferentes servicios, pero ofrece una mayor escalabilidad, tolerancia a fallos y la posibilidad de compartir recursos entre las partes del sistema.

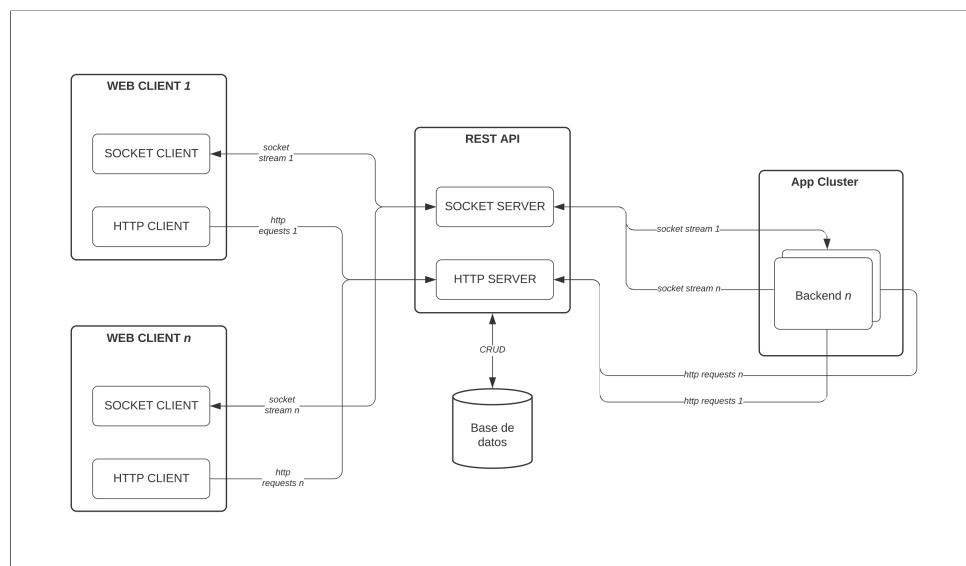


Figura 4.2: Esquema simplificado de comunicaciones y servicios en Seddi.

Seddi ofrece sus servicios a través de clientes web que exponen la funcionalidad al usuario y se comunican con un API REST para gestionar el acceso a recursos compartidos en la plataforma. Para utilizar un nuevo material especializado en tejidos, además de actualizar el motor de renderizado, ha de actualizarse la infraestructura de servicios de Seddi para crear un modelo de datos que lo represente en el sistema de almacenamiento de Seddi, así como proveer acceso a él a través del API REST. Por otra parte, las acciones más costosas, como la simulación o el renderizado basado en trazado de rayos, se computan en servicios de *High Performance Computing* (HPC) que se reservan bajo demanda.

4.1.1. Implementación del modelo de datos

Como sistema de almacenamiento de datos Seddi utiliza MongoDB [Mon], un sistema NoSQL basado en documentos. Los sistemas no relacionales nacieron a mediados de los 90 frente a los sistemas SQL y sus principales ventajas son sus bajos tiempos

de respuesta, flexibilidad en el modelo de datos y escalabilidad horizontal.

Para asegurar la consistencia entre los diferentes motores de la compañía, la representación de los materiales en base de datos y las interfaces que proporcionan acceso a ellos son comunes. Además, los motores de renderizado utilizan un *Entity Component System* (ECS) [Jor], las referencias a materiales se serializan como parte de un componente.

Los componentes que permiten renderizar elementos con materiales en la escena son *GarmentPieceComponent* y *MeshRenderableComponent*. *GarmentPieceComponent* se especializa en representar tejidos en la escena y *MeshRenderableComponent* se utiliza para definir el resto de elementos con materiales como escenarios, avatares, fornitoras, etc.

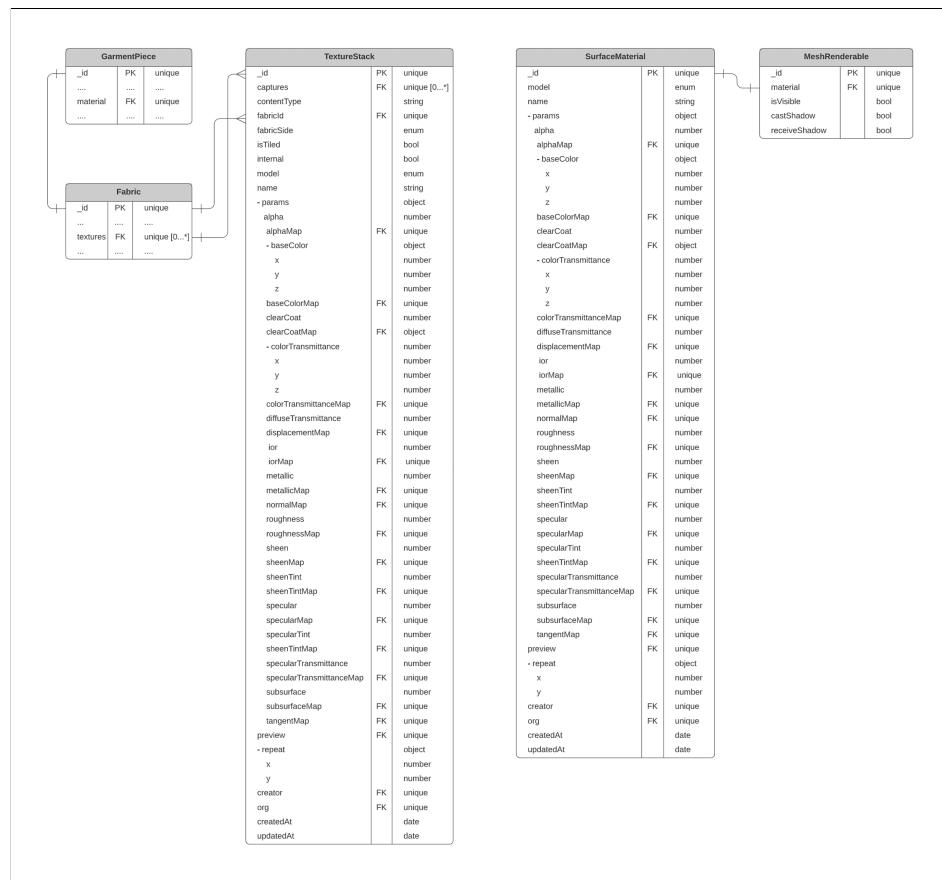


Figura 4.3: Modelo de datos de los recursos que soportan materiales.

Los materiales de un *GarmentPieceComponent* se almacenan en el campo `textures` de un recurso *Fabric*. El campo `textures` es un array que referencia recursos *TextureStack* que contienen las definiciones de los materiales para la parte frontal y trasera obtenidas durante el proceso de captura óptica. Por otra parte, los materiales de un *GarmentPieceComponent* se almacenan en recursos *SurfaceMaterial*. Estos recursos almacenan diferentes tipos de metadatos y pueden utilizar diferentes modelos de BRDFs pero comparten la parametrización de los materiales.

4.1.2. Actualización de las interfaces de servicios REST

El API REST que proporciona acceso a los recursos está montada sobre Node [Nod], un entorno de ejecución de Javascript montado sobre el motor V8 de Google que proporciona operaciones no bloqueantes de lectura y escritura (I/O) a disco o red. De entre la variedad de *frameworks* disponibles en el ecosistema de Node, el API REST de Seddi utiliza Express debido a su amplia comunidad, rendimiento, facilidad y velocidad de desarrollo.

The screenshot shows the Swagger UI interface for a REST API. It is organized into two main sections: 'Surface Materials' and 'Texture Stacks'. Each section contains a list of HTTP methods (GET, POST, PUT, PATCH, DELETE) with their corresponding URLs and descriptions. Most operations are marked with a lock icon, indicating they require authentication. The 'Surface Materials' section includes operations for listing, creating, retrieving, deleting, updating, exporting, and managing maps for surface materials. The 'Texture Stacks' section includes operations for listing, creating, retrieving, deleting, updating, promoting, and downloading texture stacks for fabrics.

Surface Materials	
GET	/surfaceMaterials List all of the Surface Materials available on your account
POST	/surfaceMaterials Create a new Surface Material
GET	/surfaceMaterials/{id} Retrieve an existing Surface Material by id
DELETE	/surfaceMaterials/{id} Delete a Surface Material
PATCH	/surfaceMaterials/{id} Update a Surface Material
GET	/surfaceMaterials/{id}/export Export an existing Surface Material
PUT	/surfaceMaterials/{id}/maps/{attribute} Update a Surface Material's Map
DELETE	/surfaceMaterials/{id}/maps/{attribute} Remove a Surface Material's Map
Texture Stacks	
GET	/fabrics/{id}/textures List all of the Texture Stacks for a Fabric available on your account
POST	/fabrics/{id}/textures Create a new Texture Stack for a Fabric
GET	/textureStacks/{id} Retrieve an existing Texture Stack by id
DELETE	/textureStacks/{id} Delete a Texture Stack
PUT	/textureStacks/{id}/maps/{attribute} Update a Texture Stack's map
POST	/textureStacks/{id}/promotions Promote a Texture Stack into its parent resource
GET	/textureStacks/{id}/stack Retrieve a Texture Stack's binary zip file

Figura 4.4: Documentación de los recursos SurfaceMaterial y TextureStack en Swagger.

Las operaciones permitidas sobre los recursos *SurfaceMaterial* y *TextureStack* son: crear, obtener, actualizar y borrar un recurso por *id*, además de permitir actualizar la textura de un mapa y descargar la definición del material y sus texturas. Adicionalmente, un *TextureStack* puede ser promocionado, lo que indica que el recurso indicado es el que se utiliza dentro del campo *textures* del *Fabric* asociado, que se indica en el campo *fabricId* del *TextureStack*.

Las citadas funcionalidades se documentan en Swagger [Sw], una herramienta de documentación automatizada que proporciona una guía a los consumidores del API.

4.2. Motor de renderizado de Author

Además de actualizar el modelo de datos y las interfaces que ofrecen acceso a ellos, el nuevo material ha de ser integrado dentro del contexto 3D basado en ThreeJs de Author. A continuación se analiza la arquitectura de su sistema de renderizado, su librería de materiales y los modelos matemáticos e implementaciones utilizadas por sus materiales PBR.

4.2.1. Arquitectura del sistema de renderizado de ThreeJs

El nuevo material extiende la librería de materiales para ofrecer una interfaz igual al resto de materiales nativos de ThreeJs. Para ello, estudiaremos la arquitectura del motor de renderizado de ThreeJs tomando como referencia los dos materiales disponibles en la librería, *MeshStandardMaterial* y *MeshPhysicalMaterial*.

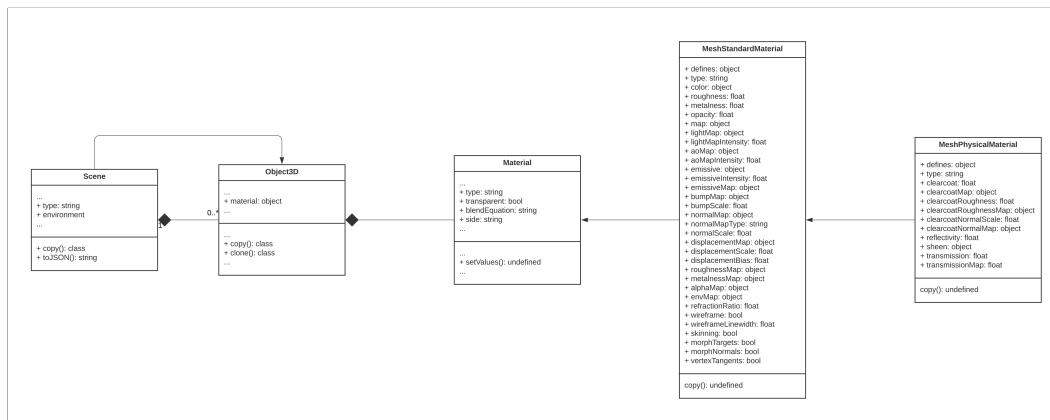


Figura 4.5: Documentación de los recursos SurfaceMaterial y TextureStack en Swagger.

La clase base de la que heredan es *Material* y almacena información común cualquier tipo de material: opacidad, ecuación de blending, la cara visible, etc. Además proporciona un método para configurar los valores de las propiedades del material, así como métodos para la copia, clonado o serialización del material.

MeshStandarMaterial hereda de *Material* y es la implementación base de un material PBR. Almacena propiedades comunes como el color el mapa de normales, mapa de desplazamiento, además de sobreescibir la parametrización de shaders y la función de *copy* de la clase *Material* para copiar correctamente estas nuevas propiedades.

El *MeshPhysicalMaterial* hereda de *MeshStandarMaterial* añadiendo el lóbulo de *clearcoat* y el BRDF alternativo para el *sheen*, utiliza sus propia parametrización de shaders y sobreescribe la función de copia.

Los elementos que se renderizan en la escena, heredan de la clase base *Object3D*, que guarda una referencia o varias referencias a materiales. El grafo de escena se gestiona desde la clase *Scene*, que hereda de *Object3D* y guarda información sobre el fondo o mapa de entorno, además de proporcionar sus propios métodos de copia y serialización a JSON.

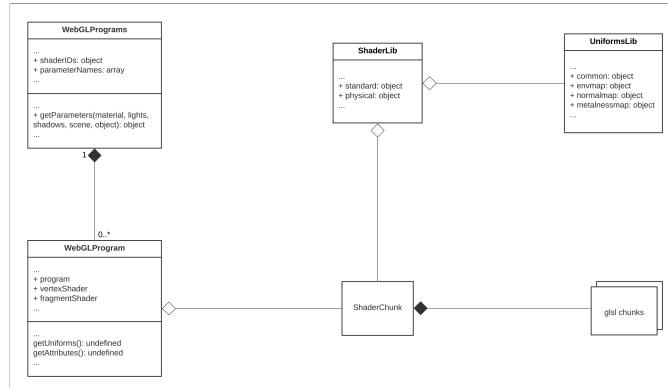


Figura 4.6: Estructuras de datos de ThreeJs que se encargan de la gestión de programas de WebGL

Para generar los *WebGLProgram* del API de WebGL, ThreeJs organiza el código GLSL en *chunks*, o trozos de código que se reutilizan y se componen en tiempo de ejecución para generar el código del programa de WebGL.

De la misma forma, los *uniforms* que se reutilizan entre programas se organizan en el objeto *UniformsLib*. El objeto *ShaderLib* almacena los materiales estándar de la librería, *MeshStandardMaterial*, *MeshPhysicalMaterial*, etc, utilizando los trozos de código GLSL de *ShaderChunk*, los uniforms definidos en *UniformsLib*, además de los *uniforms* únicos para cada tipo de material.

La clase *WebGLProgram* se encarga de juntar los trozos de código GLSL y generar la cadena de texto que se utilizará en el programa WebGL. Las instancias de la clase *WebGLProgram* se cachean en una lista de referencias en *WebGLPrograms*, que proporciona acceso a la lista, así como funciones para actualizarla u obtener los parámetros o los *uniforms* de un programa.

Finalmente, la clase *WebGLRenderer* es donde se gestiona la lógica del motor de render y se orquesta la relación entre todos estos componentes. *WebGLRenderer* recibe una referencia al grafo de escena, la clase *Scene*, que se encarga de inicializar y actualizar todos los materiales de la escena, referenciados en la clase base de cualquier elemento que se renderiza en la escena, *Object3D*. Tiene una referencia a *WebGLMaterials*, que ofrece un método para actualizar los *uniforms* de cualquier tipo de los materiales nativos de ThreeJs.

4.2.2. Materiales PBR en ThreeJs

ThreeJs presenta desde su versión 74 el material *MeshStandardMaterial*, basado en física y desde la versión 76, el *MeshPhysicalMaterial*, que extiende al anterior añadiendo parámetros presentes en el modelo de Disney. El modelo está basado en el de

Disney 2012 [Bur12], pero con algunas diferencias con intención de hacerlo más ligero para la plataforma web. A continuación se detallan los principales cambios entre los dos modelos:

4.2.2.1. Componente difusa

Al contrario que en Disney, el modelo de ThreeJs utiliza directamente el modelo de Lambert, considerando una retrodispersión uniformemente distribuida en todas direcciones.

Componente difusa en ThreeJs:

$$f_d = \frac{c}{\pi} \quad (4.1)$$

Componente difusa en Disney 2012:

$$f_d = \frac{\text{baseColor}}{\pi} \left(1 + (F_{D90} - 1)(1 - \cos\theta_{wi})^5 \right) \left(1 + (F_{D90} - 1)(1 - \cos\theta_{wo})^5 \right) \quad (4.2)$$

4.2.2.2. Componente especular

El BRDF del lóbulo primario es siempre isotrópico, a diferencia del de Disney 2012 [Bur12], en el que el lóbulo primario puede ser isotrópico o anistrópico, al contrario que el lóbulo de *clearcoat*, que siempre es isotrópico. Además el especular utiliza un BRDF GGX [Wal07], a diferencia del GTR [Bur12] del modelo de Disney. A continuación el código del BRDF especular en ThreeJs:

```

1 vec3 BRDF_Specular_GGX(
2   const in IncidentLight incidentLight ,
3   const in vec3
4   viewDir ,
5   const in vec3 normal ,
6   const in vec3
7   specularColor ,
8   const in float roughness
9 ) {
10   float alpha = pow2( roughness );
11   vec3 halfDir = normalize( incidentLight.direction + viewDir );
12   float dotNL = saturate( dot( normal, incidentLight.direction ) );
13   float dotNV = saturate( dot( normal, viewDir ) );
14   float dotNH = saturate( dot( normal, halfDir ) );
15   float dotLH = saturate( dot( incidentLight.direction, halfDir ) );
16   vec3 F = F_Schlick( specularColor, dotLH );
17   float G = G_GGX_SmithCorrelated( alpha, dotNL, dotNV );
18   float D = D_GGX( alpha, dotNH );
19   return F * ( G * D );
20 }
```

Listing 4.1: Clase MeshClothMaterial

Término D

Utiliza una aproximación de la distribución GGX, frente a la GTR utilizada en el modelo de Disney, por lo que presenta una cola más corta.

$$D_{GGX}(m) = \frac{\alpha^2}{\pi((n \cdot m)^2(\alpha^2 - 1) + 1)^2} \quad (4.3)$$

Ecuación 4.3: Función de distribución de las normales en ThreeJs

$$D_{GTR} = \frac{c}{(\alpha^2 \cos^2 \theta_h + \sin^2 \theta_h)^2} \quad (4.4)$$

Ecuación 4.4: Función de distribución de las normales en Disney 2012

```
1 float D_GGX( const in float alpha, const in float dotNH ) {
2
3     float a2 = pow2( alpha );
4     float denom = pow2( dotNH ) * ( a2 - 1.0 ) + 1.0;
5
6     return RECIPROCAL_PI * a2 / pow2( denom );
7 }
```

Listing 4.2: Implementación en ThreeJs del término de geometría

Término F

Igual que en el modelo de Disney 2012, se considera la función de Fresnel-Schlick lo suficientemente precisa para estimar el Fresnel, sin embargo, en éste caso se utiliza la aproximación presentada por Epic en el Siggraph de 2013.

$$F = F_0 + (1 - F_0)2^{(-5,55473\cos\theta_d - 5,55473\cos\theta_d)} \quad (4.5)$$

Ecuación 4.5: Aproximación de la función de Fresnel en ThreeJs

$$F = (1 - F(\theta_l)(1 - F(\theta_d)))^5 \quad (4.6)$$

Ecuación 4.6: Aproximación de la función de Fresnel en Disney 2012

```
1 vec3 F_Schlick( const in vec3 specularColor, const in float dotLH ) {
2
3     float fresnel = exp2( ( -5.55473 * dotLH - 6.98316 ) * dotLH );
4
5     return ( 1.0 - specularColor ) * fresnel + specularColor;
6
7 }
```

Listing 4.3: Implementación en ThreeJs de la aproximación a la función de Fresnel

Término G

Para definir las microfacetas en sombra o enmascaradas, ThreeJs utiliza el mismo término que Disney 2012, usa la formulación de Smith para separar la función en dos componentes, luz y vista, utilizando la misma ecuación para las dos.

$$G(w_i, w_o) = \frac{0,5}{G_1(w_i) + G_1(w_o)}$$

$$G_1(w_i) = (n \cdot w_o) \sqrt{((-n \cdot w_i)\alpha^2 + n \cdot w_i)n \cdot w_i + \alpha^2}$$

$$\text{siendo } k = \frac{(roughness + 1)^2}{8} \quad (4.7)$$

Ecuación 4.7: Función de geometría en ThreeJs

$$G(w_i, w_o) = G_1(w_i)G_1(w_o)$$

$$G_1(w_o) = \frac{n \cdot w_o}{(n \cdot)(1 - k) + k} \quad (4.8)$$

Ecuación 4.8: Función de geometría en Disney 2012

```

1 float G_GGX_SmithCorrelated(
2     const in float alpha,
3     const in float dotNL,
4     const in float dotNV
5 ) {
6
7     float a2 = pow2( alpha );
8
9     float gv = dotNL * sqrt( a2 + ( 1.0 - a2 ) * pow2( dotNV ) );
10    float gl = dotNV * sqrt( a2 + ( 1.0 - a2 ) * pow2( dotNL ) );
11
12    return 0.5 / max( gv + gl, EPSILON );
13
14 }
```

Listing 4.4: Clase MeshClothMaterial

Clearcoat

El efecto de *clearcoat*, se utiliza para simular efectos de barniz o recubrimientos, y se aplican tanto en Disney 2012 como un lóbulo secundario. Mientras que en ThreeJs se utiliza el mismo BRDF con la misma parametrización que para el lóbulo primario, en Disney 2012, varía la parametrización de los términos D y G.

$$G_{GGX}(v) = \frac{2(n \cdot v)}{(n \cdot v) + \sqrt{\alpha^2 + (1 - \alpha)^2(n \cdot v)^2}}$$

$$\text{siendo } \alpha = 0,25, G_{GGX}(v) = \frac{2(n \cdot v)}{(n \cdot v) + \sqrt{0,0625 + 0,5625(n \cdot v)^2}} \quad (4.9)$$

Ecuación 4.9: Función de geometría para el lóbulo de *clearcoat* en ThreeJs

$$D_{GTR} = \frac{c}{(\alpha^2 \cos^2 \theta_h + \sin^2 \theta_h)^\gamma}$$

$$\text{siendo } \gamma = 1,0, D_{GTR} = \frac{c}{(\alpha^2 \cos^2 \theta_h + \sin^2 \theta_h)^\gamma} \quad (4.10)$$

Ecuación 4.10: Función de distribución de las normales para el lóbulo de *clearcoat* en Disney 2012

```

1 void RE_Direct_Physical(
2     const in IncidentLight directLight,
3     const in GeometricContext geometry,
4     const in PhysicalMaterial material,
5     inout ReflectedLight reflectedLight
6 ) {
7
8     //...
9     #ifdef CLEARCOAT
10
11     // ...
12     reflectedLight.directSpecular += ccIrradiance * material.clearcoat
13         * BRDF_Specular_GGX(
14             directLight,
15             geometry.viewDir,
16             geometry.clearcoatNormal,
17             vec3( DEFAULT_SPECULAR_COEFFICIENT ),
18             material.clearcoatRoughness
19         );
20
21     #else
22     // ...
23
24 }
```

Listing 4.5: Implementación del lóbulo de *clearcoat* en ThreeJs

Sheen

El parámetro de *sheen* ofrece un mayor control sobre la reflectancia especular, permitiendo crear materiales con especulares en dos tonos.

ThreeJs utiliza un BRDF diferente cuando el parámetro de *sheen*, mientras que en el modelo de Disney el *sheen* se modela como otro lóbulo adicional además del de

clearcoat.

$$f(\text{sheen}, \theta_d) = \frac{D_{Charlie}}{4(n \cdot l + n \cdot v - (n \cdot l)(n \cdot v))}$$

$$\text{siendo } D_{Charlie}(\alpha) = \frac{(2 + \frac{1}{\alpha}) \sin(\theta)^{\frac{1}{\alpha}}}{2\pi} \quad (4.11)$$

Ecuación 4.11: Modelo de BRDF utilizando *sheen* en ThreeJs

$$f(\text{sheen}, \theta_d) = \text{sheen} * ((1 - \text{sheenTint}) + \text{sheenTint} * \text{tint}) * (1 - \cos \theta_d)^5 \quad (4.12)$$

Ecuación 4.12: Lóbulo adicional de *sheen* en Disney 2012

```

1 vec3 BRDF_Specular_Sheen(
2     const in float roughness,
3     const in vec3 L,
4     const in GeometricContext geometry,
5     vec3 specularColor
6 ) {
7
8     vec3 N = geometry.normal;
9     vec3 V = geometry.viewDir;
10
11    vec3 H = normalize( V + L );
12    float dotNH = saturate( dot( N, H ) );
13
14    return specularColor * D_Charlie( roughness, dotNH )
15        * V_Neubelt( dot(N, V), dot(N, L) );
16
17 }
```

Listing 4.6: BRDF del modelo de *sheen* de ThreeJs

Capítulo 5

Modelos de iluminación para tejidos

Los tejidos se componen de fibras entrelazadas con cierta separación entre sí, por lo que la metáfora de la rejilla compuesta por microespejos que se utiliza en la teoría de microfacetas no es la mejor aproximación para tratar de representar este material de forma realista. Los materiales PBR *MeshStandardMaterial* y *MeshPhysicalMaterial* de ThreeJs, basados en microfacetas, funcionan muy bien para gran parte de materiales, como plásticos o metales, sin embargo no consiguen una representación realista de tejidos, que suelen presentar un lóbulo especular más atenuado.

La estructura de fibras puede provocar los efectos de absorción y reflexión interna (*subsurface scattering*) y, dependiendo de la dirección de la luz, o los efectos de dispersión hacia delante ((*forward scattering*) que se produce cuando la luz viene en dirección opuesta al punto de vista y golpea las fibras que salen del tejido, o la dispersión hacia detrás (*backward scattering*), cuando la luz viene en dirección opuesta al punto de vista. El terciopelo es un caso de tejido en el que se acentúa este efecto y el primer trabajo en describirlo, el modelo de BRDF de Ashikmin [Ash07], es también conocido como *velvet BRDF* o *velvet distribution*.

Este trabajo implementa sobre ThreeJs el modelo de *Cloth* de Filament [Guy13]. La librería de Google presenta un BRDF especial para tejidos que, inspirado en los trabajos de Ashikmin [Ash07] y más tarde Estevez y Kulla [ACE17] para el término de distribución de las normales, y el de Neubelt [DN13] para el factor de geometría, permite una representación realista en entornos interactivos de los efectos mencionados.

5.1. Integración de una nueva clase de material en ThreeJs

ThreeJs ofrece la posibilidad de crear materiales a través de su API, sin embargo, para este trabajo se ha decidido crear un *fork* de la librería para soportar posibles nuevos materiales y funcionalidades que se requieran específicamente de Author.

En este capítulo se explica la implementación de una nueva clase de material que se integra en la librería de materiales de ThreeJs. Siguiendo la nomenclatura de la librería (*MeshBasicMaterial*, *MeshStandardMaterial*, etc.), el nuevo material se llamará *MeshClothMaterial*.

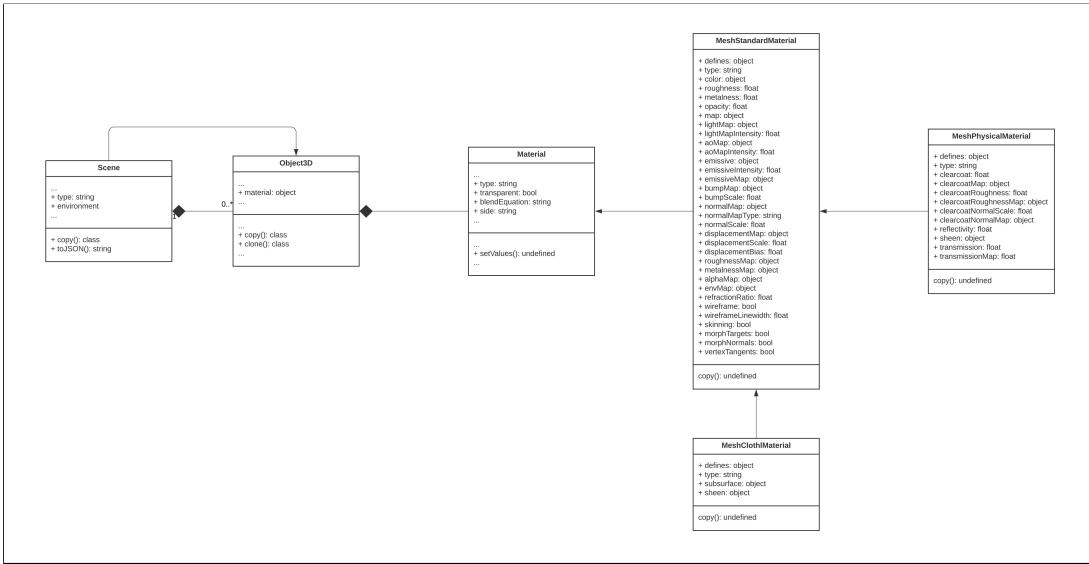


Figura 5.1: Integración de la nueva clase *MeshClothMaterial*

El nuevo material tiene un BRDF diferente al *MeshPhysicalMaterial*, y diferentes parámetros, sin embargo soporta los mismos efectos de *normal mapping*, *displacement*, etc. y comparte gran cantidad de los parámetros con el material base, añadiendo *subsurface*, *sheen*. *MeshClothMaterial* utiliza sus propios *type*, para identificar el tipo de shader, en la clase *WebGLMaterials* y *defines*, que se utilizarán para añadir directivas de preprocesador al código GLSL generado en la clase *WebGLProgram*. Para que el nuevo material esté quede expuesto como un material nativo de ThreeJs, se añade a objeto *Materials*.

<i>MeshPhysicalMaterial</i>	<i>MeshClothMaterial</i>
color	color
roughness	roughness
clearCoat	NO
clearCoatRoughness	NO
sheen	sheen
NO	subsurface
metalness	NO
emissive	emissive
alpha	alpha
lightmap	lightmap
normalMap	normalMap

Figura 5.2: Comparativa *MeshPhysicalMaterial*, *MeshClothMaterial*

En la clase *WebGLMaterials* se incluye un nuevo método para actualizar los uniforms de éste nuevo material.

```

1 // WebGLMaterials.js
2
3 // ...
4 refreshUniformsCommon( uniforms, material );
5 // ...
6 function refreshUniformsCloth( uniforms, material, environment ) {
7
8     refreshUniformsStandard( uniforms, material, environment );
9
10    uniforms.reflectivity.value = material.reflectivity; // also part of
11        uniforms common
12
13    if ( material.sheen ) {
14
15        uniforms.sheen.value.copy( material.sheen );
16
17    } else {
18
19        uniforms.sheen.value.copy( material.color );
20        uniforms.sheen.value.r = Math.sqrt( uniforms.sheen.value.r );
21        uniforms.sheen.value.g = Math.sqrt( uniforms.sheen.value.g );
22        uniforms.sheen.value.b = Math.sqrt( uniforms.sheen.value.b );
23
24    }
25
26    if ( material.subsurface ) {
27        uniforms.subsurface.value.copy( material.subsurface );
28    }
29
30    if ( material.brdfCloth ) {
31        uniforms.brdfCloth.value = material.brdfCloth;
32    }
33 // ...

```

Listing 5.1: Cambios sobre la clase WebGLMaterials de ThreeJs

En la clase *WebGLProgram* se modifica el constructor para para añadir las directivas de precompilación para los parámetros *subsurface* y *sheen*, en caso de utilizarse estos parámetros.

```

1 // WebGLProgram.js
2 // ...
3 function WebGLProgram( renderer, cacheKey, parameters, bindingStates ) {
4     // ...
5     prefixFragment = [
6         // ...
7         ( parameters.sheen || parameters.shaderID === 'cloth' ) ? '#define
8             USE_SHEEN' : '',
9         parameters.subsurface ? '#define SUBSURFACE' : '',
10        // ...
11    ]
12 }
13 // ...

```

Listing 5.2: Cambios sobre la clase WebGLProgram de ThreeJs

En el closure *WebGLPrograms* se actualiza añade una clave para el nuevo tipo de

material, *MeshClothMaterial* y se añade la cadena de texto *subsurface* al array *parameterNames*, además de añadir nuevas claves para el *subsurface* y la información precomputada del BRDF, dentro del objeto *parameters* dentro de la función *getParameters*. De esta forma nos aseguramos que el sistema interpreta correctamente el nuevo material y puede acceder y cachear sus propiedades.

```

1 // WebGLPrograms.js
2 // ...
3 const shaderIDs = {
4   // ...
5   MeshClothMaterial: 'cloth',
6   // ...
7 };
8
9 const parameterNames = [
10   // ...
11   "subsurface", "brdfCloth",
12   // ...
13 ];
14 // ...
15 function getParameters( material, lights, shadows, scene, object ) {
16   // ...
17   const parameters = {
18     // ...
19     subsurface: !! material.subsurface,
20     brdfCloth: !! envMap && shaderID === 'MeshClothMaterial',
21     // ...
22   };
23   // ...
24 }
25 // ...

```

Listing 5.3: Cambios sobre la clase WebGLPrograms de ThreeJs

5.2. Integración del nuevo *shader* sobre ThreeJs

Para añadir un nuevo BRDF hemos de extender el sistema de *chunks* de ThreeJs. Para ello hemos de añadir el punto de entrada del nuevo shader al objeto *shaderChunk*. Siguiendo el sistema de nomenclatura de ThreeJs, llamaremos *meshcloth_frag.glsLjs* al nuevo shader, que utiliza chunks comunes con el *MeshStandardMaterial*, salvo para el cálculo de la iluminación directa e indirecta, que utiliza sus propios *chunks* para el cálculo de la ecuación de render: *lights_cloth_fragment.glsLjs*, donde se inicializa la estructura de datos relativa al material y *lights_cloth_pars_fragment.glsLjs*, donde se calculan las ecuaciones de render de la iluminación directa e indirecta del nuevo material.

5.2.1. Componente difusa

El color difuso y el color de *subsurface*, modelan en realidad el mismo fenómeno físico, la cantidad de reflexión interna que resulta en aportación sobre la reflexión en

dirección al vector de vista. Cuando la distancia de reflexión es despreciable en comparación con el tamaño del pixel, se aproxima como una reflexión difusa, sin embargo, cuando la distancia es considerable, el fenómeno se modela de forma diferente.

En tiempo real, es común utilizar la citada aproximación El modelo para tejidos de Filament utiliza el difuso de Lambert en caso de no utilizar el parámetro *subsurface*, mientras que si lo utiliza utiliza el factor de corrección de la técnica *wrapped diffuse lighting* [Fer04] que permite aproximar el efecto debido a la absorción y flexión interna de la luz incidente en la superficie.

$$f_d(v, h) = \frac{c_{diff}}{\pi} \left\langle \frac{(n \cdot l)w}{(1+w)^2} (c_{subsurface} + n \cdot l) \right\rangle \quad (5.1)$$

Ecuación 5.1: Componente difusa del modelo de *Cloth* en Filament

```

1 vec3 BRDF_Diffuse_Cloth(
2     const in IncidentLight incidentLight,
3     const in vec3 viewDir,
4     const in vec3 normal,
5     const in vec3 diffuseColor
6 ) {
7
8     float dotNL = dot( normal, incidentLight.direction );
9     float radiance = RECIPROCAL_PI; // Lambert BRDF
10
11    #if defined(SUBSURFACE)
12        // Fd_Wrap
13        radiance *= saturate((dotNL + 0.5) / 2.25);
14    #endif
15
16    vec3 Fd = radiance * diffuseColor;
17
18    return Fd;
19
20 }
21
22 void RE_Direct_Cloth(
23     const in IncidentLight directLight,
24     const in GeometricContext geometry,
25     const in PhysicalMaterial material,
26     inout ReflectedLight reflectedLight
27 ) {
28
29     // ...
30     reflectedLight.directDiffuse += irradiance * BRDF_Diffuse_Cloth(
31         directLight,
32         geometry.viewDir,
33         geometry.normal,
34         material.diffuseColor
35     );
36
37     #if defined(SUBSURFACE)
38         reflectedLight.directDiffuse *= saturate(subsurface + dotNL);
39     #endif
40     // ...

```

```

41
42 }
```

Listing 5.4: Implementación del BRDF para la componente difusa de *MeshClothMaterial*

5.2.2. Componente especular

El *MeshPhysicalMaterial* utiliza por defecto un BRDF GGX, [Wal07]. Sin embargo, si el parámetro de sheen está activo, utiliza Al contrario que el *MeshPhysicalMaterial*, que utiliza una una distribucion GGX, el material Cloth de Filament utiliza una version modificada del BRDF presentado por Ashikmin y Premoze. [Ash07]

$$D_{Velvet}(v, h, \alpha) = c_{norm}(1 + 4\exp\left(\frac{-\cot^2\theta_h}{\alpha^2}\right)) \quad (5.2)$$

Ecuación 5.2: *Velvet distribution* [Ash07]

$$D_{Charlie}(\alpha) = \frac{(2 + \frac{1}{\alpha})\sin(\theta)^{\frac{1}{\alpha}}}{2\pi} \quad (5.3)$$

Ecuación 5.3: Componente difusa del modelo de *Cloth* en Filament

```

1 float D_Charlie(float roughness, float NoH) {
2     // Estevez and Kulla 2017, "Production Friendly Microfacet Sheen BRDF"
3     float invAlpha = 1.0 / roughness;
4     float cos2h = NoH * NoH;
5     float sin2h = max(1.0 - cos2h, 0.0078125); // 2^(-14/2), so sin2h^2 > 0
6                                         // in fp16
7     return (2.0 + invAlpha) * pow(sin2h, invAlpha * 0.5) / (2.0 * PI);
8 }
```

Listing 5.5: Implementación del factor de distribución de Charlie en ThreeJs

Mientras que para el factor de geometría o visibilidad se utiliza la version modificada de Neubelt [DN13] del denominador de microfacetas para suavizar el degradado de la sombra.

$$\frac{1}{4(n \cdot l + n \cdot v - (n \cdot l)(n \cdot v))} \quad (5.4)$$

Ecuación 5.4: Formulación del término de geometría de Neubelt [DN13]

```

1 float V_Neubelt(float NoV, float NoL) {
2     // Neubelt and Pettineo 2013, "Crafting a Next-gen Material Pipeline
3                                         for The Order: 1886"
4     return saturate(1.0 / (4.0 * (NoL + NoV - NoL * NoV)));
5 }
```

Listing 5.6: Implementación del término de de visibilidad de Neubelt [DN13]

```

1 vec3 BRDF_Specular_Sheen( const in float roughness, const in vec3 L,
  const in GeometricContext geometry, vec3 specularColor ) {
2
3   vec3 N = geometry.normal;
4   vec3 V = geometry.viewDir;
5
6   vec3 H = normalize( V + L );
7   float dotNH = saturate( dot( N, H ) );
8
9   return specularColor * D_Charlie( roughness, dotNH ) * V_Neubelt( dot(N
  , V), dot(N, L) );
10
11 }
```

Listing 5.7: Cambios sobre la clase WebGLPrograms de ThreeJs

5.2.3. Iluminación indirecta

De las técnicas de iluminación indirecta vistas en el capítulo 3, ThreeJs soporta tanto mapas de irradiancia como esféricos harmónicos para la componente difusa, mientras que para la componente especular utiliza la aproximación analítica presentada por Lazarov [Laz13].

5.2.3.1. Componente difusa

Para el cálculo del difuso, ha de estimarse la irradiancia debida al entorno sobre un punto y multiplicarlo por el BRDF difuso. ThreeJs soporta ambas técnicas vistas en el capítulo 3 para el cálculo de irradiancia: mapas de irradiancia y esféricos harmónicos.

$$L_o(p, w_o) = k_d \frac{c}{\pi} \int_{\Omega} L_i(p, w_i) n \cdot w_i dw_i \quad (5.5)$$

Ecuación 5.5: Cálculo de la luz indirecta difusa

La técnica de mapas de irradiancia, aunque muy eficiente durante la ejecución, son muy costosas como para ser generadas bajo demanda en un entorno interactivo. Por su parte, los esféricos harmónicos reducen el coste de generación de los mapas de entorno a costa de comprimir la información de irradiancia en un representación de frecuencias frente a espacio.

Los *chunks* de ThreeJs separan el cálculo de la luz indirecta en una función para cada componente, difuso y especular. El cálculo de la componente difusa se realiza en la función *RE_IndirectDiffuse_Cloth*.

```

1 void RE_IndirectDiffuse_Cloth( const in vec3 irradiance, const in
  GeometricContext geometry, const in PhysicalMaterial material, inout
  ReflectedLight reflectedLight ) {
2
3   reflectedLight.indirectDiffuse += irradiance * BRDF_Diffuse_Lambert
  ( material.diffuseColor );
4
5 }
```

Listing 5.8: Cálculo de la ecuación de render para la componente difusa de *MeshClothMaterial*

Sin embargo, para tener en cuenta el principio de conservación de la energía, $k_d + k_s = 1$, tal y como se presenta en el modelo de Cook-Torrance [Coo81], parte del cálculo se hace en la función *RE_IndirectSpecular_Cloth*, que evalúa la componente especular. *RE_IndirectSpecular_Cloth* utiliza un mapa prefiltrado de entorno, explicado en el capítulo 3, para calcular la irradiancia, que se le resta al total de energía (1), para obtener el peso de la irradiancia difusa. Finalmente, en caso de utilizar *subsurface*, se aplica el factor de corrección.

```

1 void RE_IndirectSpecular_Cloth(
2     const in vec3 radiance,
3     const in vec3 irradiance,
4     const in vec3 clearcoatRadiance,
5     const in GeometricContext geometry,
6     const in PhysicalMaterial material,
7     inout ReflectedLight reflectedLight
8 ) {
9     // ...
10
11    float diffuseWrapFactor = 1.0;
12    #if defined(SUBSURFACE)
13        diffuseWrapFactor *= saturate(
14            dotNV + subsurfaceIntensity)
15        / ((1.0 + subsurfaceIntensity) * (1.0 + subsurfaceIntensity))
16    );
17    #endif
18
19    // E = specular irradiance
20    vec3 FdSH = reflectedLight.indirectDiffuse * ( 1.0 - E )
21        * diffuseWrapFactor;
22    vec3 FdIM = irradiance * BRDF_Diffuse_Lambert( material.diffuseColor )
23        * ( 1.0 - E ) * diffuseWrapFactor;
24
25    vec3 Fd = Fd_SH + Fd_Lod;
26
27    #if defined(SUBSURFACE)
28
29        Fd *= saturate(subsurface + dotNV);
30
31    #endif
32    // ...
33
34    reflectedLight.indirectDiffuse = Fd;
35
36 }
```

5.2.3.2. Componente especular

Tanto ThreeJs utilizan la técnica *split-sum approximation* [Kar14] en la componente especular. Sin embargo, mientras que Filament utiliza un *BRDF integration map*, ThreeJs utiliza la aproximación presentada por Epic [Kar14], ambas presentadas en el capítulo 3.

La aproximación de ThreeJs del BRDF especular funciona especialmente bien para dispositivos con un rendimiento limitado. Esta aproximación de la integral del BRDF es una solución analítica para el BRDF GGX [Wal07], por lo que no es correcta para el

$$L_o(p, w_o) = \int_{\Omega} k_s \frac{DFG}{4(w_o \cdot n)(w_i \cdot n)} L_i(p, w_i) n \cdot w_i dw_i = \int_{\Omega} k_s \frac{DFG}{4(w_o \cdot n)(w_i \cdot n)} * \int_{\Omega} L_i(p, w_i) n \cdot w_i dw_i \quad (5.6)$$

Ecuación 5.6: Separación de la integral de la componente especular en la integral del BRDF y la integral de la radiancia

nuevo BRDF del modelo de telas.

```

1  vec2 integrateSpecularBRDF( const in float dotNV, const in float
2    roughness ) {
3    const vec4 c0 = vec4( - 1, - 0.0275, - 0.572, 0.022 );
4    const vec4 c1 = vec4( 1, 0.0425, 1.04, - 0.04 );
5    vec4 r = roughness * c0 + c1;
6    float a004 = min( r.x * r.x, exp2( - 9.28 * dotNV ) ) * r.x + r
7      .y;
8    return vec2( -1.04, 1.04 ) * a004 + r.zw;
9 }
```

Listing 5.9: Apromixación analítica a la integral del BRDF en ThreeJs

El nuevo modelo de telas, de la misma forma que Filament, utiliza un *BRDF integration map* para almacenar en los canales rojo y azul la escala y offset del Fresnel para el modelo GGX [Wal07]. El resultado del BRDF de tejidos, que no utiliza Fresnel, se almacena en el canal azul, al que se accede de igual forma que al *BRDF integration map* para el modelo GGX [Wal07], utilizando $n \cdot l$ para el eje x y el valor de rugosidad del material para el eje y .

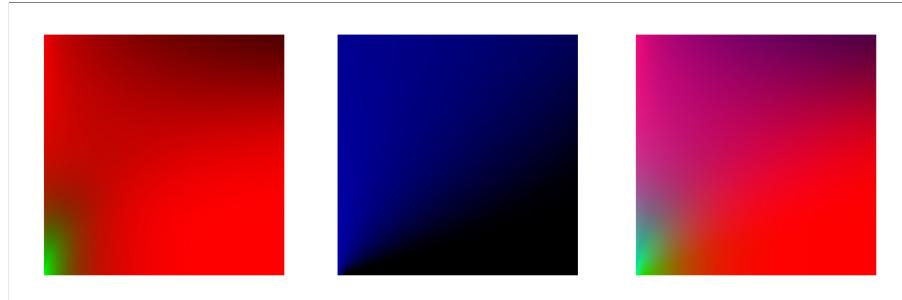


Figura 5.3: Canales RG (BRDF GGX [Wal07]), B (BRDF para telas [Guy13]) y la suma de ellos, RGB, del *BRDF integration map*

```

1 float prefilteredDG = textureLod(
2   brdfCloth,
3   vec2( dotNV, material.specularRoughness * material.specularRoughness ),
4   0.0
5 ).b;
6
7 vec3 E = material.sheenColor * prefilteredDG;
8 reflectedLight.indirectSpecular += E * radiance;
```

Listing 5.10: Cálculo de la irradiancia del entorno para el modelo de telas utilizando el nuevo *BRDF integration map*

```

1 void RE_IndirectSpecular_Cloth(
2     const in vec3 radiance,
3     const in vec3 irradiance,
4     const in vec3 clearcoatRadiance,
5     const in GeometricContext geometry,
6     const in PhysicalMaterial material,
7     inout ReflectedLight reflectedLight
8 ) {
9
10    float dotNV = dot( geometry.normal, geometry.viewDir );
11
12    float prefilteredDG = textureLod(
13        brdfCloth,
14        vec2(
15            dotNV,
16            material.specularRoughness * material.specularRoughness
17        ),
18        0.0
19    ).b;
20
21    vec3 E = material.sheenColor * prefilteredDG;
22    reflectedLight.indirectSpecular += E * radiance;
23
24    // ...
25
26 }

```

Listing 5.11: Cálculo de iluminación indirecta de *MeshClothMaterial*

Como resumen, la implementación completa de la luz indirecta para el *MeshClothMaterial*:

```

1 void RE_IndirectDiffuse_Cloth( const in vec3 irradiance, const in
2     GeometricContext geometry, const in PhysicalMaterial material, inout
3     ReflectedLight reflectedLight ) {
4
5     reflectedLight.indirectDiffuse += irradiance * BRDF_Diffuse_Lambert(
6         material.diffuseColor );
7
8     void RE_IndirectSpecular_Cloth( const in vec3 radiance, const in vec3
9         irradiance, const in vec3 clearcoatRadiance, const in GeometricContext
10        geometry, const in PhysicalMaterial material, inout ReflectedLight
11        reflectedLight) {
12
13        float dotNV = dot( geometry.normal, geometry.viewDir );
14
15        float prefilteredDG = textureLod(brdfCloth, vec2( dotNV, material.
16            specularRoughness * material.specularRoughness ), 0.0).b;
17        vec3 E = material.sheenColor * prefilteredDG;
18        reflectedLight.indirectSpecular += E * radiance;
19
20        float diffuseWrapFactor = 1.0;
21        #if defined(SUBSURFACE)
22            diffuseWrapFactor *= saturate((dotNV + subsurfaceIntensity) / ((1.0 +
23                subsurfaceIntensity) * (1.0 + subsurfaceIntensity)));
24        #endif
25
26 }

```

```

20 // Combine envmap and SH diffuse irradiance
21 vec3 Fd_SH = reflectedLight.indirectDiffuse * ( 1.0 - E ) *
22     diffuseWrapFactor;
23 vec3 Fd_Lod = irradiance * BRDF_Diffuse_Lambert( material.diffuseColor
24     ) * ( 1.0 - E ) * diffuseWrapFactor;
25 vec3 Fd = Fd_SH + Fd_Lod;
26
27 #if defined(SUBSURFACE)
28
29     Fd *= saturate(subsurface + dotNV);
30
31 #endif
32
33 }

```

5.3. Resultados

A continuación se muestran y comentan las imágenes como resultado de la implementación del *MeshPhysicalMaterial*. Los resultados se centran en el análisis de las citadas propiedades *sheen* y *subsurface*.



Figura 5.4: Comparativa *MeshPhysicalMaterial* (fila superior) contra *MeshClothMaterial* (fila inferior) incrementando progresivamente la intensidad de luz global

En la figura 5.4 se muestra una escena en la que la intensidad del entorno incrementa de izquierda a derecha frente a una luz direccional muy tenue. El *sheen* de *MeshPhysicalMaterial* solo está soportado en la iluminación directa, mientras que para la iluminación indirecta utiliza la aproximación analítica del lóbulo primario. En la comparativa se muestra como el material de ThreeJs pierde el efecto de *sheen* a medida que aumenta la intensidad del mapa de entorno. Además, al utilizar un bajo de rugosidad se hace patente la inconsistencia entre los dos modelos de BRDF utilizados para la reflexión especular en el *MeshPhysicalMaterial* generando un efecto parecido al de *clearcoat*.



Figura 5.5: Interpolación de un color de *sheen* aumentando su luminosidad frente a un color base constante.

El nuevo material no ofrece un control directo sobre la intensidad del *sheen*, sin embargo, un efecto similar se puede alcanzar utilizando un el mismo que el color base aumentando la luminosidad como se muestra en la figura 5.5.



Figura 5.6: Interpolación de un color de *sheen* aumentando su saturación frente a un color base constante.

Como se muestra en la figura 5.6, utilizar un color de *sheen* diferente al color base permite conseguir reflexiones especulares en dos tonos como resultado de la combinación del color base y el *sheen*.



Figura 5.7: Interpolación de *subsurfaceIntensity* utilizando un color base y color de *subsurface* blancos.

En la figura 5.8, se utiliza el blanco como color base y *subsurface*. La interpolación del parámetro de *subsurfaceIntensity* muestra una transición de sombras más intensas (*subsurfaceIntensity* = 0) a sombras más suaves, que toman el color base del *subsurface* (*subsurfaceIntensity* = 1)



Figura 5.8: Interpolación de un color de *subsurface* frente a un color base constante.

La figura 5.7 muestra interpolaciones entre un color base constante (blanco) y un color de *subsurface*, rojo en la fila superior y azul en la fila inferior. Subiendo el valor de *subsurfaceIntensity* se puede apreciar como el tejido se tiñe progresivamente del color de *subsurface*.



Figura 5.9: Interpolación del color de sheen en el eje *x* frente a interpolación del color base en el eje *y*.

La matriz bidimensional de la figura 5.9 utiliza un color de sheen que se interpola entre el color base del tejido, gris oscuro, hasta un rojo intenso en el eje *x*, mientras que en el eje *y*, el color base se interpola hacia el mismo tono rojo que el *sheen*.

A continuación se muestran comparativas entre el *MeshPhysicalMaterial* de ThreeJs, el nuevo *MeshClothMaterial* y las imágenes generadas por el motor de trazado de rayos.

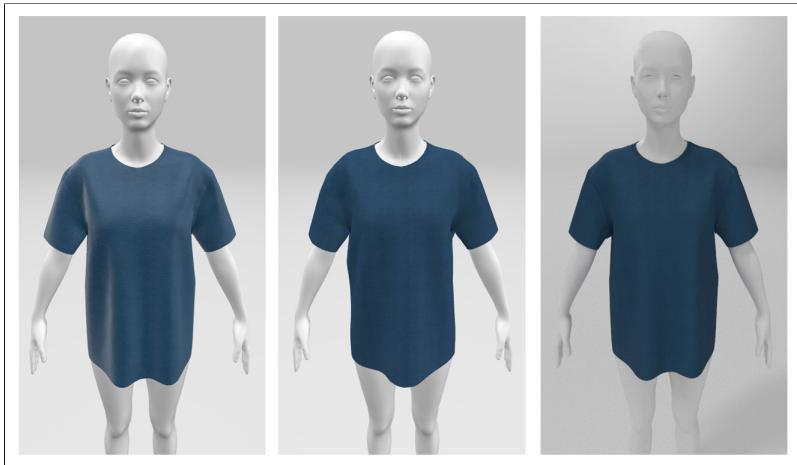


Figura 5.10: Tejido de canvas renderizado con *MeshPhysicalMaterial*, *MeshClothMaterial* y el servicio de renderizado de trazado de rayos de Seddi

Mientras que el brillo especular de *MeshPhysicalMaterial* da al tejido un aspecto de plástico, el mayor control sobre el especular de *MeshClothMaterial* permite atenuar el efecto consiguiendo una mayor coherencia con los resultados obtenidos por el servicio de renderizado en la nube de Seddi.



Figura 5.11: Tejido de sarga renderizado con *MeshPhysicalMaterial*, *MeshClothMaterial* y el servicio de renderizado de trazado de rayos de Seddi

De igual forma, en la figura 5.11, se aprecia como el *MeshPhysicalMaterial* falla al capturar un tejido de sarga, cuyas reflexiones son muy definidas el ángulo crítico. Por su parte, el *MeshPhysicalMaterial* utiliza un color marrón con muy baja luminosidad que atenúa el efecto consiguiendo un resultado más parecido al obtenido por el motor de trazado de rayos.

Capítulo 6

Conclusiones y trabajo futuro

Durante el desarrollo de este TFM se ha integrado efectivamente un modelo de datos que soporte representaciones de diferentes modelos de BRDF dentro de la arquitectura de SEDDI. Además se ha creado un *fork* que extiende la librería de ThreeJs para representar de forma más realista los tejidos y que permite futuras adaptaciones sobre los sistemas de iluminación y materiales de ThreeJs.

La integración del nuevo material sobre la plataforma de Author ha permitido trabajar sobre las réplicas digitales de tejidos y validar que este modelo de BRDF y los fenómenos físicos que modela, aumentan el realismo y coherencia con el motor de trazado de rayos sobre una amplia variedad de tejidos.

No obstante, este trabajo supone el punto de partida en el incremento de realismo de tejidos sobre el motor de renderizado de ThreeJs y la discusión de los resultados ha constatado las siguientes posibles líneas de trabajo:

1. Investigar modelos anisotrópicos que permitan representar la direccionalidad del brillo especular en ciertos tipos de tejidos en función de la dirección de peinado de sus fibras.
2. Invesitar alternativas a la aproximación del efecto de *subsurface*. Mientras que para la luz directa, la aproximación del efecto funciona correctamente, dependiendo de la dirección de la luz, para la luz indirecta, al no conocer la dirección de la luz, se utiliza la dirección de la vista. Esto provoca que la aproximación funcione bien sobre imágenes estáticas, pero no funciona correctamente durante los movimientos de cámara.

Bibliografía

- [ACE17] C. K. Alejandro Conty Estevez. «Production Friendly Microfacet Sheen BRDF». En: *SIGGRAPH 2017* (jul. de 2017), págs. 1-2.
- [Ash07] M. Ashikhmin. «Distribution-based BRDFs». En: (2007), pág. 9.
- [Ash+00] M. Ashikhmin y col. «A microfacet-based BRDF generator». En: (ene. de 2000), págs. 65-74.
- [Bec+63] P. Beckmann y A. Spizzichino. *The Scattering of Electromagnetic Waves from Rough Surfaces*. Artech House, Inc., 1963.
- [Bli77] J. F. Blinn. «Models of Light Reflection for Computer Synthesized Pictures». En: *SIGGRAPH 1977: Proceedings of the 4th annual conference on Computer graphics and interactive techniques* (jul. de 1977), págs. 192-198.
- [Bur12] B. Burley. «Physically-Based Shading at Disney». En: *SIGGRAPH 2012* (jul. de 2012), pág. 26.
- [Bur15] B. Burley. «Extending the Disney BRDF to a BSDF with Integrated Subsurface Scattering». En: *SIGGRAPH 2015* (ago. de 2015), págs. 1-19.
- [Coo81] R. Cook. «A Reflectance Model for Computer Graphics». En: *SIGGRAPH Comput. Graph.* 15.3 (ago. de 1981), págs. 307-316.
- [DN13] M. P. David Neubelt. «Crafting a Next-Gen Material Pipeline for The Order: 1886». En: *SIGGRAPH 2013* (jul. de 2013), págs. 1-29.
- [Deb86] P. Debevec. «Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-based Graphics with Global Illumination and High Dynamic Range Photography». En: *SIGGRAPH '86* (1986), págs. 143-150.
- [Fer04] R. Fernando. *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*. 2004.
- [Got12] Y. Gotanda. «Beyond a Simple Physically Based Blinn-Phong Model in Real-Time». En: (2012).
- [Guy13] R. Guy. *Physically Based Rendering in Filament*. <https://google.github.io/filament/Filament.html>. 2013.
- [Jor] M. Jordan. *Entities, components and systems*. <https://medium.com/ingeniouslysimple/entities-components-and-systems-89c31464240d>.
- [Kaj86] J. T. Kajiya. «The Rendering Equation». En: *SIGGRAPH '86* (1986), págs. 143-150.
- [Kar13] B. Karis. «Real Shading in Unreal Engine 4». En: *SIGGRAPH 2013* (jul. de 2013), págs. 1-59.
- [Kar14] B. Karis. *Physically Based Shading on Mobile*. Inf. téc. 2014.
- [Lam60] J. H. Lambert. *Photometria*. 1760.

- [Laz13] D. Lazarov. «Getting More Physical in Call of Duty: Black Ops II». En: *SIGGRAPH 2013* (jul. de 2013), págs. 1-18.
- [Mat+03] W. Matusik y col. «A Data-Driven Reflectance Model». En: *ACM Transactions on Graphics* 22.3 (2003), págs. 759-769.
- [Mon] *MongoDB official website*. <https://www.mongodb.com>.
- [Nod] *Node official website*. <https://nodejs.org>.
- [Ore+94] M. Oren y S. K. Nayar. *Generalization of Lambert's Reflectance Model*. ACM, 1994, págs. 239-246.
- [Pho73] B. T. Phong. «Illumination of Computer-Generated Images». En: *Communications of the ACM* 18.6 (jul. de 1973), págs. 311-317.
- [Ram+01] R. Ramamoorthi y P. Hanrahan. «On the relationship between Radiance and Irradiance: Determining the illumination from images of a convex Lambertian object». En: *Journal of the Optical Society of America. A, Optics, image science, and vision* 18 (2001), págs. 2448-59.
- [Sch94] C. Schlick. «An Inexpensive BRDF Model for Physically-based Rendering». En: *Computer Graphics Forum* 13.3 (1994), págs. 233-246.
- [SL14] C. d. R. Sébastien Lagarde. «Moving Frostbite to Physically Based Rendering». En: *SIGGRAPH 2014* (ago. de 2014), pág. 122.
- [Smi67] B. Smith. «Geometrical shadowing of a random rough surface». En: *IEEE Transactions on Antennas and Propagation* 15.5 (1967), págs. 668-671.
- [Swa] *Swagger official website*. <https://swagger.io/>.
- [Tor+67] K. Torrance y E. Sparrow. «Theory for Off-Specular Reflection from Roughened Surfaces». En: *Journal of the Optical Society of America (JOSA)* 57 (1967), págs. 1105-1114.
- [Wal07] B. Walter. «Microfacet Models for Refraction through Rough Surfaces». En: *EGSR'07: Proceedings of the 18th Eurographics conference on Rendering Techniques* 18.12 (jul. de 2007), págs. 1-12.