Universidad Camilo José Cela

U-tad
CENTRO UNIVERSITARIO DE TECNOLOGÍA Y ARTE DIGITAL

MÁSTER EN COMPUTACIÓN GRÁFICA REALIDAD VIRTUAL Y LA SIMULACIÓN

MATERIALES PBR EN WEBGL 2.0

AUTOR: ANXO BABÍO RODRÍGUEZ
TUTOR: IVÁN ALDUÁN ÍÑIGUEZ

1 12345678910111213

[1] Michal Drobot. «Lighting of Killzone Shadow Fall». En: *Digital Dragons* (abr. de 2013), págs. 1-118. URL: http://www.klayge.org/material/4_6/Drobot_Lighting_of_Killzone_Shadow_Fall.pdf.

[2] Brian Karis. «Real Shading in Unreal Engine 4». En: *SIGGRAPH 2013* (jul. de 2013), págs. 1-59. URL: https://cdn2.unrealengine.com/Resources/files/2013SiggraphPresentationsNotes-26915738.pdf.

[3] Dimitar Lazarov. «Getting More Physical in Call of Duty: Black Ops II talk». En: *SIGGRAPH 2013* (jul. de 2013), págs. 1-35. URL: https://blog.selfshadow.com/publications/s2013-shading-course/\#course_content.

[4] Dimitar Lazarov. «Getting More Physical in Call of Duty: Black Ops II». En: *SIGGRAPH 2013* (jul. de 2013), págs. 1-18. URL: https://blog.selfshadow.com/publications/s2013-shading-course/lazarov/s2013_pbs_black_ops_2_notes.pdf.

[5] Matt Pettineo David Neubelt. «Crafting a Next-Gen Material Pipeline for The Order: 1886 talk». En: *SIGGRAPH 2013* (jul. de 2013), págs. 1-74. URL: https://blog.selfshadow.com/publications/s2013-shading-course/rad/s2013_pbs_rad_slides.pdf.

[6] Matt Pettineo David Neubelt. «Crafting a Next-Gen Material Pipeline for The Order: 1886». En: *SIGGRAPH 2013* (jul. de 2013), págs. 1-29. URL: https://blog.selfshadow.com/publications/s2013-shading-course/rad/s2013_pbs_rad_notes.pdf.

[7] Charles de Rousiers Sébastien Lagarde. «Moving Frostbite to Physically Based Rendering talk». En: *SIGGRAPH 2014* (ago. de 2014), págs. 1-91. URL: https://media.contentapi.ea.com/content/dam/eacom/frostbite/files/s2014-pbs-frostbite-slides.pdf.

[8] Charles de Rousiers Sébastien Lagarde. «Moving Frostbite to Physically Based Rendering». En: *SIGGRAPH 2014* (ago. de 2014), pág. 122. URL: https://media.contentapi.ea.com/content/dam/eacom/frostbite/files/course-notes-moving-frostbite-to-pbr-v32.pdf.

[9] Brent Burley. «Extending the Disney BRDF to a BSDF with Integrated Subsurface Scattering talk». En: *SIGGRAPH 2015* (ago. de 2015), págs. 1-62. URL: https://blog.selfshadow.com/publications/s2015-shading-course/burley/s2015_pbs_disney_bsdf_slides.pdf.

[10] Brent Burley. «Extending the Disney BRDF to a BSDF with Integrated Subsurface Scattering». En: *SIGGRAPH 2015* (ago. de 2015), págs. 1-19. URL: https://blog.selfshadow.com/publications/s2015-shading-course/burley/s2015_pbs_disney_bsdf_notes.pdf.

[11] Yibing Jiang. «The process of creating volumetric-based materials in Uncharted 4». En: *SIGGRAPH 2016* (jul. de 2016), págs. 1-70. URL: http://advances.realtimerendering.com/s2016/The%20Process%

Índice general

Capítulo 1

Introducción

Capítulo 2

Objetivo

Objetivos generales:

1. Conseguir materiales más realistas para la visualización tejidos sobre el navegador web.
2. Aumentar coherencia entre las imágenes del renderer online y el offline en Seddi.

Objetivos específicos

1. Analizar las técnicas empleadas en los motores de render de Seddi.
2. Identificar las diferencias en los algoritmos que más afectan al resultado visual.
3. Extender el motor de render del cliente para mostrar con mayor fidelidad ciertas propiedades de los tejidos.

Capítulo 3

Contexto

Seddi es una compañia de computación gráfica especializada en simulación de tejidos para la industria de la moda. Su objetivo es crear una servicio en la nube en el que las empresa dedicadas a la moda puedan digitalizar todo su flujo de trabajo para agilizar el prototipado y reducir costes tanto económicos como medioambientales. Para ello se ofrece una plataforma online de trabajo colaborativo con herramientas que permiten editar el diseño o las propiedades de una prenda a los diferentes perfiles implicados en las fases de producción: diseñadores textiles, diseñadores de moda, patronistas, product managers, etc.

3.1. Caracteristicas: especial atención a los tejidos.

El motor de render online de Seddi debe ser capaz de mostrar un amplio abanico de materiales, pero prestando una especial atención a los tejidos. Los materiales que pueden ser usados durante el proceso de diseño de una prenda pueden ir desde cuero, lana, toalla, terciopelo... cada uno con unas propiedades opticas y mecánicas diferentes y únicas que influyen completamente en las decisiones de diseño del producto.

3.2. Author: aplicación web donde el usuario diseña sus creaciones

El proceso empieza con la creación de un digital twin de un hilo, capturando las propiedades físicas y mecánicas de las fibras que lo componen y que sirven como entradas a los algoritmos de simulación y render. A partir de este hilo los usuarios pueden diseñar un tejido o escoger entre una gran cantidad de ellos, en función de la frecuencia o el tipo de entrelazado de los hilos. Utilizando estos tejidos se diseñan patrones en plano con las formas de las piezas que conforman una prenda y se permite al usuario previsualizar y editar la construcción del modelo en una vista 3D sobre un maniquí. Una vez completado el ciclo de diseño se generan automáticamente fichas técnicas con especificaciones sobre los elementos estéticos o constructivos que facilitan la comunicación entre todas las partes implicadas en el proceso de fabricación.

Capítulo 4

Conceptos previos

4.1. Render

El render es un area de la computacion gráfica que se encarga de traducir estructuras de datos y funciones a píxeles en la pantalla con el objetivo de formar una imagen bidimensional.

La representación digital de una imagen consiste en un grafo que contiene los elementos de la escena: cámara, luces y objetos visibles en la escena. Sobre esta escena virtual, se simulan las interacciones de la luz con las superficies para acabar estimando un color para cada superficie visible desde el punto de vista de la cámara.

Sus principales aplicaciones se encuentran en industrias como el cine de animación, los efectos especiales, videojuegos, arquitectura, simuladores del ámbito industrial, visualizaciones de producto o la imágen médica. Debido a las diferentes necesidades y caracteristicas de cada industria, podemos categorizar a los motores de render principalemente en base a dos criterios: su intención de imitar o no la realidad (motores fotorrealistas y no fotorrealistas) y la necesidad o no de interaccion de usuario (motores online y offline).

En este trabajo, la intencion es el renderizado de telas con el mayor grado de fidelidad posible a la realidad en una aplicación con interaccion de usuario, por lo que nos centraremos en el renderizado online fotorrealista.

Añadir informaciõn elementos y grafo de escena. A scene file contains objects in a strictly defined language or data structure; it would contain geometry, viewpoint, texture, lighting, and shading information as a description of the virtual scene. The data contained in the scene file is then passed to a rendering program to be processed and output to a digital image or raster graphics image file. If a scene is to look relatively realistic and predictable under virtual lighting, the rendering software should solve the rendering equation

4.2. Modelo físico de la luz

Para conseguir imágenes parecidas a la realidad, el render utiliza algoritmos que simulan el comportamiento físico de la luz. El comportamiento de los fotones se describe con el fenomeno cuantico conocido como dualidad onda-partícula. El comportamiento como particula establece que se mueve en linea recta, rebotando con los objetos del entorno a una velocidad de 300.000 km/s, la mas alta conocida y que en el modelo de simulación se considera infinita. Por otra parte, el comportamiento de de la luz como onda, permite trabajar sobre propiedades como frecuencia, amplitud y longitud de onda, que se utilizan para modelar la representación digital del color .

Inverse square law. Ley de Lambert. Ley de reflexion. Ley de Snell

4.2.1. Unidades de medida de la luz

Cuando hablamos de la cantidad de luz, no es la medicion sobre un foton de luz independiente, si no que se tratan de medidas en relacion al tiempo, la direccion o el área.

Ángulo sólido

Flujo La unidad que representa la energia sobre la unidad de tiempo es el flujo, representado por Phi. Es la energía que transportan las ondas por unidad de tiempo, se mide en vatios. En los motores de render se utiliza para expresar la cantidad total de energia emitida por un a fuente de luz.

$$\Phi_e = \frac{dQ_e}{dt} \tag{4.1}$$

Intensidad

$$I = \frac{d\Phi}{d\omega} \tag{4.2}$$

Irradiancia La irradiancia, es la cantidad de energía por unidad de tiempo por unidad de superficie, o flujo por superficie. Se representa como E, su unidad son los vatios/m2 y se utiliza para medir la cantidad de luz que incide sobre una superficie.

$$E = \frac{d\Phi}{dA} \tag{4.3}$$

Cuando este flujo de radiancia se mide en dirección contraria, de salida, se llama emitancia (M)

Radiancia Simula luz tan lejana que sus rayos son paralelos entre si, como por ejemplo el sol.

$$L = \frac{d^2\Phi}{dA_{proj}d\omega} \tag{4.4}$$

4.3. Ecuación de render

El propósito de la ecuacion de render es conocer el valor de radiancia que llega a la camara en una dirección por cada pixel de la camara.

Para saber la radiancia sobre un punto en una dirección, utilizamos la ecuacion de reflectancia, que depende de la luz que llega al puntos, el coseno del angulo con el que incide la luz y el BRDF (bidirectional reflectance distribution function), que modela el comportamiento de la luz al rebotar sobre la superficie.

$$L(x \to \vec{v}) = L_i(x \leftarrow \vec{l}) f(x, \vec{v}, \vec{l})(\vec{l} \cdot \vec{n}) \tag{4.5}$$

Ademas de recibir la luz directamente de una fuente de luz, el punto también puede recibir luz rebotada por otras partes del entorno:

$$L(x \to \vec{v}) = L_e(x, \vec{v}) + L_i(x \leftarrow \vec{l}) f(x, \vec{v}, \vec{l})(\vec{l} \cdot \vec{n}) \tag{4.6}$$

Podríamos sumar todas las fuentes de luz para calcular la radiancia total debido a las fuentes de luz:

$$L(x \to \vec{v}) = L_e(x, \vec{v}) + \sum L_i(x \leftarrow \vec{l_i}) f(x, \vec{v}, \vec{l_i})(\vec{l_i} \cdot \vec{n}) \tag{4.7}$$

Sin embargo, para tener en cuenta el total de luz incidente, ademas de las fuentes de luz (luz directa), debemos de tener en cuenta toda la luz proviniente del entorno en todas direcciones (luz indirecta):

$$L(x \to \vec{v}) = L_e(x, \vec{v}) + \int_\Omega L_i(x' \to \vec{l_i}) f(x, \vec{v}, \omega_i) \cos\theta_i d\omega_i \tag{4.8}$$

Como podemos ver, la radiancia aparece a los dos lados de la ecuación, lo que quiere decir que para calcular la radiancia en un punto, necesitamos la radiancia en otros puntos por lo que se trata de un calculo recursivo.

Esta ecuación cumple dos propiedades: linearidad y descomposicion por partes, gracias a ellas, podemos transformar el problema en una serie de Neumann.

Agrupando los terminos conseguimos:

$$L_{out} = \Upsilon L_{in} L_{in} = \Lambda L_{out} \tag{4.9}$$

Y juntandolos:

$$L_{out} = \Upsilon \Lambda L_{out} + L_e \tag{4.10}$$

Al agruparlos:

$$L_{out} = K L_{out} + L_e \tag{4.11}$$

Reordenarlos:

$$(I - K) L_{out} = L_e L_{out} = (I - K)^{-1} L_e \tag{4.12}$$

Se puede separar la emisión en partes:

$$L_{out} = (I - K)^{-1} E_1 + (I - K)^{-1} E_2 \tag{4.13}$$

Y sustituyendo:

$$(I - K)^{-1} = \sum_{i=0}^{\infty} K^i = I + K + KK + KKK + K + \cdots \tag{4.14}$$

Obtenemos la serie de Neumann:

$$L_{out} = L_e + K L_e + KK L_e + KKK L_e + \cdots \tag{4.15}$$

Demostrando que la radiancia total puede ser calculada como la suma de luz emi-

tida, mas la suma de la reflejada una vez, mas la suma de la reflejada dos veces, etc.

4.4. PBR

PBR o *physically based rendering* es el termino que se utiliza para nombrar los algoritmos de render basados en un modelo físico. La intención es conseguir aproximaciones rápidas y plausibles de la interacción de un flujo de luz con una superficie. Las principales ventajas de este tipo de algoritmos son la consistencia bajo diferentes condiciones de luz y su manejo intuitivo por parte de los artistas.

Un motor PBR ha de tener en cuenta las leyes de las física para el tratamiento de todos los elementos de la escena: luces, camaras y superficies de los objetos visibles.

4.4.1. Cámara

La camara se utiliza para determinar la parte de la escena visible en la imagen 2D final.

Una camara de un motor de render PBR utiliza la matriz de proyeccion perspectiva para simular la vista del ojo humano y su modelo básico consiste en la simulación de una camara estenopeica, un modelo basico de cámara fotográfica que consiste en un compartimento oscuro, con un fondo sobre el que se coloca el negativo fotográfico, mientras que en la parte delantera cuenta con una perforación, por la que entra la luz antes de golpear con el material fotosensible.

La cámara permite manipular aspectos esenciales de la imagen final como el campo de visión, la distancia mínima y máxima de la escena con respecto al ojo. Además, en función de los objetivos del motor de render, puede simular efectos de las camaras fotográficas, como la profunidad de campo, la velocidad de obturación o el tiempo de apertura, asi como su posición y orientación, que son, junto a la posición de la luz, factores clave para obtener la radiancia de una superficie.

4.4.2. Iluminación

Como hemos visto en la ecuación de render, para calcular el total de radiancia proviniente de una superficie, en función de su origen, podemos descomponer la suma de su iluminación en la suma de iluminación directa e iluminación indirecta. Gran parte de la radiancia total que se ve sobre un punto suele venir dado por las fuentes de iluminación directa. En una escena digital, las luces son aproximaciones a las fuentes de luz que tenemos en la naturaleza.

4.4.2.1. Emisores de luz

Podemos distinguir como los tipos de luz prinpicales:

Luz de ambiente Luz que simula la componente de iluminación global en caso de que esta no se calcule. Suma un valor fijo a la iluminacion de cada punto.

Luz puntual Luz emitida desde un punto en todas direcciones.

Luz de foco Emite desde un punto en un cono orientado.

Luz direccional Simula luz tan lejana que sus rayos son paralelos entre si, como por ejemplo el sol.

Luz de área Luz emitida desde una superficie, normalmente un plano o un disco.

Luz de entorno Usa un mapa sobre una esfera para simular las condiciones de iluminacion de la imagen.

4.4.2.2. Luz directa

Es la que proviene directamente de una fuente de luz y que contribuye en gran parte al aspecto final de una escena. Tendremos en cuenta la luz que viaja de una fuente de luz a una superficie y de la superficie al ojo.

4.4.2.3. Luz indirecta

4.5. Materiales

Los objetos de la escena están compuestos por su geometría y su material.

La geometría es una secuencia de vértices en el espacio que conforman polígonos que describen la forma de un objeto. La información de la geometría se almacenan en estructuras de datos que ademas de la información de vértices, puede contener información sobre las caras de los polígonos, las normales, tangentes, etc.

El material, por otra parte, describe las características visuales de la superficie del objeto. En la naturaleza, además del color de un objeto, podemos percibir propiedades de la textura, su conductividad, dielectrico o metalico, su rugosidad o su reflectividad a través de la interacción de la luz con la superficie de un objeto. Esta interacción a nivel microscopico entre la luz y la superficie de un material, se describe con una ecuacion conocida como BSDF.

4.5.1. Teoría de microfacetas

A physically-based BRDF is based on the microfacet theory, which supposes that a surface is composed of small-scaled planar detail surfaces of varying orientation called microfacets. Each of these small planes reflects light in a single direction based on its normal (Figure 07). Micro-facets whose surface normal is oriented exactly halfway between the light direction and view direction will reflect visible light. However, in cases where the microsurface normal and the half normal are equal, not all microfacets will contribute as some will be blocked by shadowing (light direction) or masking (view direction) as illustrated in Figure 07. The surface irregularities at a microscopic level cause light diffusion. For example, blurred reflections are caused by scattered light rays. The rays are not reflected in parallel, so we perceive the specular reflection as blurred (Figure 08). Son funciones que permiten describir el comportamiento de la luz al golpear una superficie. Para ello toma en cuenta el ángulo de incidencia del rayo de luz sobre la superficie y el rayo de salida, reflejado o refractado y su resultado es el ratio entre el ángulo del rayo y la superficie y la intesidad de salida de la luz.

4.5.2. BxDF

A continuación se detallan los nombres de las funciones en funcion del fenómeno físico que modelan.

BRDF Es la función que modela el comportamiento de la luz al golpear una superficie opaca, la reflexión. Fue definido por primera vez en 1965 por Fred Nicodemus y su definición es el ratio entre radiancia reflectada e irraciancia incidente. Para determinar el ángulo de salida del rayo se utiliza la ley de la reflexión y las tres características que ha de cumplir un BRDF basado en física es que sea positivo, que cumpla con la reciprocidad de Helmholtz y que cumpla con la ley de conservación de la energía.

BTDF Describe el comportamiento del rayo de luz al atravesar una superficie, la refracción. Para calcular el angulo de salida del rayo se utiliza la ley de Snell, y al contrario que el BRDF, no sumple el principio de reciprocidad de Helmholtz.

BSSRDF y BSSTF Son ampliaciones del modelo de reflexión y refracción, respectivamente, teniendo en cuenta las reflexiones internas del rayo a traves de la superficie del objeto.

BSDF Se utiliza comunmente para hablar de cualquier forma de BxDF. En un sentido mas estricto, se refiere al conjunto de un BSSRDF y un BSSTDF.

4.5.3. Términos del BRDF

4.6. Disney Principled

The BRDF used by Substance's PBR shaders are based on Disney's principled reflectance model. This model is based on the GGX microfacet distribution. GGX provides one of the better solutions in terms of specular distribution: with a shorter peak in the highlight and a longer tail in the falloff, it looks more realistic (Figure 10). Disney's physically-based reflectance model was designed to be a principled approach. That is, it was geared more towards art direction rather than a strictly physical model.

Capítulo 5

Estado del arte

Capítulo 6

Análisis

6.1. Comparación ThreeJS - Disney 2012

6.1.1. Esquema BRDF ThreeJS

En el modelo de Disney 2012, se utilizan dos BRDFs distintos, uno objetos metálicos y otro para dieléctricos y se hace blending entre ellos:

```
1  // main() {
2  // ...
3
4  float metalnessFactor = metalness;
5  #ifdef USE_METALNESSMAP
6      vec4 texelMetalness = texture2D( metalnessMap, vUv );
7      metalnessFactor *= texelMetalness.b;
8  #endif
9
10 // ...
11
12 material.diffuseColor = diffuseColor.rgb * (1.0 - metalnessFactor);
13 material.specularRoughness = clamp(roughnessFactor, 0.04, 1.0);
14
15 #ifdef REFLECTIVITY
16     // MAXIMUM_SPECULAR_COEFFICIENT = 0.16
17     material.specularColor = mix(vec3(MAXIMUM_SPECULAR_COEFFICIENT * pow2
           (reflectivity), diffuseColor.rgb, metalnessFactor);
18 #else
19     // DEFAULT_SPECULAR_COEFFICIENT = 0.04
20     material.specularColor = mix(vec3(DEFAULT_SPECULAR_COEFFICIENT),
           diffuseColor.rgb, metalnessFactor);
21
22 // ...
23 // calculate direct illumination...
24 // calculate indirect illumination...
25 // ...
26
27 // }
```

Listing 6.1: My Javascript Example

6.1.2. Iluminación directa

El valor del difuso se calcula directamente con el BRDF de Lambert.

Para calcular el especular, se acumulan las contribuciones de los tres lóbulos. Si se utiliza clearcoat, se estima el *clearcoatDHR* (clearcoat directional hemispherical reflectivity) y se calcula la contribución al especular del clearcoat. Al siguiente lóbulo, *sheen* en caso de existir, o si no directamente al BRDF del material base, se le aplica un factor de 1.0 - clearcoatDHR y se acumula su valor.

```
// main() {
// ...

float metalnessFactor = metalness;
#ifdef USE_METALNESSMAP
    vec4 texelMetalness = texture2D( metalnessMap, vUv );
    metalnessFactor *= texelMetalness.b;
#endif

// ...

material.diffuseColor = diffuseColor.rgb * ( 1.0 - metalnessFactor );
material.specularRoughness = clamp( roughnessFactor, 0.04, 1.0 );

#ifdef REFLECTIVITY
    // MAXIMUM_SPECULAR_COEFFICIENT = 0.16
    material.specularColor = mix(vec3( MAXIMUM_SPECULAR_COEFFICIENT *
        pow2(reflectivity),                                diffuseColor.rgb,
         metalnessFactor );
#else
    // DEFAULT_SPECULAR_COEFFICIENT = 0.04
    material.specularColor = mix(vec3(DEFAULT_SPECULAR_COEFFICIENT),
        diffuseColor.rgb,                                  metalnessFactor )
        ;

// ...
// calculate direct illumination...
// calculate indirect illumination...
// ...

// }
```

Listing 6.2: My Javascript Example

6.1.2.1. Difuso

El difuso de Disney hace un blending entre el modelo de difuso de base y el BRDF para subsurfaces de HanrahanKrueger.

$$f_d = \frac{baseColor}{\pi}(1 + (F_{D90} - 1)(1 - cos\theta_t)^5)(1 + F_{D90} - 1)(1 - cos\theta_v)^5)$$

La implementación de ThreeJS utiliza el BRDF para el difuso de Lambert:

```
vec3 BRDF_Diffuse_Lambert( const in vec3 diffuseColor ) {
    return RECIPROCAL_PI * diffuseColor;
}
```

Listing 6.3: My Javascript Example

6.1.2.2. Lóbulo primario (material base)

Representa el material base y puede ser anisotrópico y/o metálico.
Implementación en ThreeJS:

```
vec3 BRDF_Specular_GGX(
    const in IncidentLight incidentLight,
    const in vec3 viewDir,
    const in vec3 normal,
    const in vec3 specularColor,
    const in float,
    roughness
) {
    float alpha = pow2(roughness);
    vec3 halfDir = normalize(incidentLight.direction + viewDir);
    float dotNL = saturate(dot(normal, incidentLight.direction));
    float dotNV = saturate(dot(normal, viewDir));
    float dotNH = saturate(dot(normal, halfDir));
    float dotLH = saturate(dot(incidentLight.direction, halfDir));
    vec3 F = F_Schlick(specularColor, dotLH);
    float G = G_GGX_SmithCorrelated(alpha, dotNL, dotNV);
    float D = D_GGX(alpha, dotNH);
    return F * (G * D);
}
```

Listing 6.4: My Javascript Example

El BRDF de ThreeJS es isotrópico y no metálico.

TÉRMINOS DEL BRDF:

1. D El mismo que Disney 2012[1]

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

```
float D_GGX( const in float alpha, const in float dotNH ) {
    float a2 = pow2( alpha );
    float denom = pow2( dotNH ) * ( a2 - 1.0 ) + 1.0;
    return RECIPROCAL_PI * a2 / pow2( denom );
}
```

Listing 6.5: My Javascript Example

2. F Versión optimizada de Fresnel-Schlick, la misma que la presentadapor Epic en el SIGGRAPH de 2013[2]

```
vec3 F_Schlick( const in vec3 specularColor, const in float dotLH )
    {
    float fresnel = exp2( ( -5.55473 * dotLH - 6.98316 ) * dotLH );
```

[1] Brent Burley. «Physically-Based Shading at Disney». En: *SIGGRAPH 2012* (jul. de 2012), pág. 26. URL: https://disney-animation.s3.amazonaws.com/library/s2012_pbs_disney_brdf_notes_v2.pdf.

[2] Karis, «Real Shading in Unreal Engine 4», óp.cit.

```
3       return ( 1.0 - specularColor ) * fresnel + specularColor;
4   }
```

Listing 6.6: My Javascript Example

3. G Utiliza que la misma presentada para Frostbite en el SIGGRAPH de 2014[3]

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

Listing 6.7: My Javascript Example

6.1.2.3. Lóbulo secundario (clearcoat)

El lóbulo secundario representa una capa de clearcoat sobre el material y siempre es isotrópica y no metálica. Se utiliza un IOR fijo de 1.5, que representa la refracción del poliuretano (F0 = 0.04) ThreeJS utiliza el mismo BRDF que para el material base, con un F0 = 0.04

```
1   float clearcoatDHR = material.clearcoat * clearcoatDHRApprox(material.
        clearcoatRoughness, ccDotNL);
2   // IOR = 1.5 -> F0 = DEFAULT_SPECULAR_COEFFICIENT = 0.04
3   reflectedLight.directSpecular += ccIrradiance * material.clearcoat *
        BRDF_Specular_GGX(
4       directLight,
5       geometry.viewDir,
6       geometry.clearcoatNormal,
7       vec3(DEFAULT_SPECULAR_COEFFICIENT),
8       material.clearcoatRoughness
9   );
```

Listing 6.8: My Javascript Example

Mismo BRDF que el lóbulo primario:

```
1   vec3 BRDF_Specular_GGX(const in IncidentLight incidentLight, const in
        vec3 viewDir,                                   const in vec3 normal, const
         in vec3 specularColor, const in float
        roughness ) {
2       float alpha = pow2( roughness );
3       vec3 halfDir = normalize( incidentLight.direction + viewDir );
4       float dotNL = saturate( dot( normal, incidentLight.direction ) );
5       float dotNV = saturate( dot( normal, viewDir ) );
6       float dotNH = saturate( dot( normal, halfDir ) );
7       float dotLH = saturate( dot( incidentLight.direction, halfDir ) );
8       vec3 F = F_Schlick( specularColor, dotLH );
9       float G = G_GGX_SmithCorrelated( alpha, dotNL, dotNV );
10      float D = D_GGX( alpha, dotNH );
```

[3]Sébastien Lagarde, «Moving Frostbite to Physically Based Rendering», óp.cit.

```
11      return F * ( G * D );
12  }
```

Listing 6.9: My Javascript Example

Calcula el factor de reflectividad del clearcoat para aplicarle 1 - clearcoatDHR al siguiente lóbulo

6.1.2.4. Lóbulo secundario (clearcoat)

El lóbulo secundario representa una capa de clearcoat sobre el material y siempre es isotrópica y no metálica. Se utiliza un IOR fijo de 1.5, que representa la refracción del poliuretano (F0 = 0.04)

ThreeJS utiliza el mismo BRDF que para el material base, con un F0 = 0.04

```
1  float clearcoatDHR = material.clearcoat * clearcoatDHRApprox(material.
       clearcoatRoughness,                    ccDotNL );
2  // IOR = 1.5 -> F0 = DEFAULT_SPECULAR_COEFFICIENT = 0.04
3  reflectedLight.directSpecular += ccIrradiance * material.clearcoat *
       BRDF_Specular_GGX(
4          directLight,
5          geometry.viewDir,
6          geometry.clearcoatNormal,
7          vec3(DEFAULT_SPECULAR_COEFFICIENT
8      ),
9      material.clearcoatRoughness
10 );
```

Listing 6.10: My Javascript Example

Mismo BRDF que el lóbulo primario

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

Listing 6.11: My Javascript Example

Calcula el factor de reflectividad del clearcoat para aplicarle *1 - clearcoatDHR* al siguiente lóbulo

```
// Clearcoat directional hemispherical reflectivity
float clearcoatDHRApprox( const in float roughness, const in float dotNL ) {
    // DEFAULT_SPECULAR_COEFFICIENT = 0.04
    return DEFAULT_SPECULAR_COEFFICIENT + ( 1.0 - DEFAULT_SPECULAR_COEFFICIENT ) *
        (pow( 1.0 - dotNL, 5.0 ) * pow( 1.0 - roughness, 2.0 ) );
}
```

Listing 6.12: My Javascript Example

6.1.2.5. Lóbulo terciario (sheen)

El lóbulo terciario se utiliza para representar el sheen, que aporta un extra de reflectancia en los ángulos muy agudos.

Este componente se parece mucho al factor de Fresnel y para modelar su BRDF se utiliza Schlick-Fresnel: $sheen * (1 - cos\zeta d \exp 5)$, se puede teñir su color con uno diferente al del material base con el parámetro sheenTint.

ThreeJS implementation

```
// Estevez and Kulla 2017, "Production Friendly Microfacet Sheen BRDF"
float D_Charlie(float roughness, float NoH) {
    float invAlpha = 1.0 / roughness;
    float cos2h = NoH * NoH;
    float sin2h = max(1.0 - cos2h, 0.0078125);
    return (2.0 + invAlpha) * pow(sin2h, invAlpha * 0.5) / (2.0 * PI);
}

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

vec3 BRDF_Specular_Sheen( const in float roughness, const in vec3 L, const in GeometricContext geometry, vec3 specularColor ) {
    vec3 N = geometry.normal;
    vec3 V = geometry.viewDir;
    vec3 H = normalize( V + L );
    float dotNH = saturate( dot( N, H ) );
    return specularColor * D_Charlie(roughness, dotNH) * V_Neubelt(dot(N, V), dot(N, L));
}
```

Listing 6.13: My Javascript Example

6.1.3. Iluminación indirecta

Capítulo 7

Resultados

Capítulo 8

Conclusión

Bibliografía

Alejandro Conty Estevez, Christopher Kulla. «Production Friendly Microfacet Sheen BRDF». En: *SIGGRAPH 2017* (jul. de 2017), págs. 1-2. URL: http://www.aconty.com/pdf/s2017_pbs_imageworks_sheen.pdf.

Burley, Brent. «Extending the Disney BRDF to a BSDF with Integrated Subsurface Scattering». En: *SIGGRAPH 2015* (ago. de 2015), págs. 1-19. URL: https://blog.selfshadow.com/publications/s2015-shading-course/burley/s2015_pbs_disney_bsdf_notes.pdf.

— «Extending the Disney BRDF to a BSDF with Integrated Subsurface Scattering talk». En: *SIGGRAPH 2015* (ago. de 2015), págs. 1-62. URL: https://blog.selfshadow.com/publications/s2015-shading-course/burley/s2015_pbs_disney_bsdf_slides.pdf.

— «Physically-Based Shading at Disney». En: *SIGGRAPH 2012* (jul. de 2012), pág. 26. URL: https://disney-animation.s3.amazonaws.com/library/s2012_pbs_disney_brdf_notes_v2.pdf.

David Neubelt, Matt Pettineo. «Crafting a Next-Gen Material Pipeline for The Order: 1886». En: *SIGGRAPH 2013* (jul. de 2013), págs. 1-29. URL: https://blog.selfshadow.com/publications/s2013-shading-course/rad/s2013_pbs_rad_notes.pdf.

— «Crafting a Next-Gen Material Pipeline for The Order: 1886 talk». En: *SIGGRAPH 2013* (jul. de 2013), págs. 1-74. URL: https://blog.selfshadow.com/publications/s2013-shading-course/rad/s2013_pbs_rad_slides.pdf.

Drobot, Michal. «Lighting of Killzone Shadow Fall». En: *Digital Dragons* (abr. de 2013), págs. 1-118. URL: http://www.klayge.org/material/4_6/Drobot_Lighting_of_Killzone_Shadow_Fall.pdf.

Jiang, Yibing. «The process of creating volumetric-based materials in Uncharted 4». En: *SIGGRAPH 2016* (jul. de 2016), págs. 1-70. URL: http://advances.realtimerendering.com/s2016/The%20Process%20of%20Creating%20Volumetric-based%20Materials%20in%20Uncharted%204.pptx.

Karis, Brian. «Real Shading in Unreal Engine 4». En: *SIGGRAPH 2013* (jul. de 2013), págs. 1-59. URL: https://cdn2.unrealengine.com/Resources/files/2013SiggraphPresentationsNotes-26915738.pdf.

Lagarde, Sébastian. «Physically-Based Material, where are we». En: *SIGGRAPH 2017* (jul. de 2017), págs. 1-119. URL: http://openproblems.realtimerendering.com/s2017/02-PhysicallyBasedMaterialWhereAreWe.pdf.

Lazarov, Dimitar. «Getting More Physical in Call of Duty: Black Ops II». En: *SIGGRAPH 2013* (jul. de 2013), págs. 1-18. URL: https://blog.selfshadow.com/publications/s2013-shading-course/lazarov/s2013_pbs_black_ops_2_notes.pdf.

— «Getting More Physical in Call of Duty: Black Ops II talk». En: *SIGGRAPH 2013* (jul. de 2013), págs. 1-35. URL: https://blog.selfshadow.com/publications/s2013-shading-course/\#course_content.

Sébastien Lagarde, Charles de Rousiers. «Moving Frostbite to Physically Based Rendering». En: *SIGGRAPH 2014* (ago. de 2014), pág. 122. URL: https://media.contentapi.ea.com/content/dam/eacom/frostbite/files/course-notes-moving-frostbite-to-pbr-v32.pdf.

— «Moving Frostbite to Physically Based Rendering talk». En: *SIGGRAPH 2014* (ago. de 2014), págs. 1-91. URL: https://media.contentapi.ea.com/content/dam/eacom/frostbite/files/s2014-pbs-frostbite-slides.pdf.