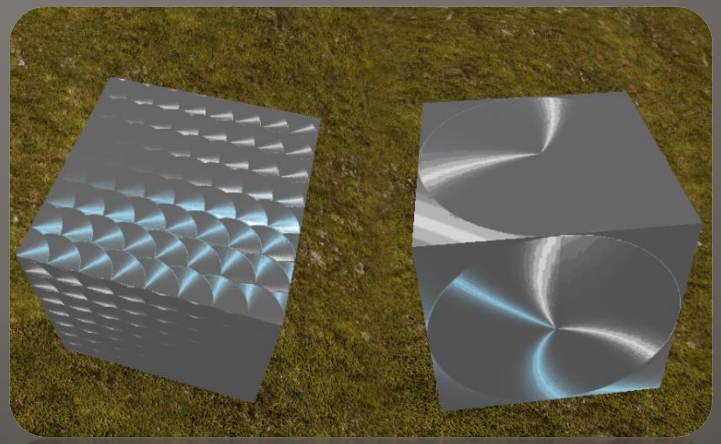
Modelo de iluminación anisotrópico para el renderizado de pelo.

PATG. Rosa Mª Sánchez Banderas

1. ¿Qué es la anisotropía?

- En el contexto de los gráficos por ordenador, se entiende por anisotropía a la variación de la iluminación de un material basada en una rotación sobre la superficie normal del mismo.
- Ejemplos de anisotropía sería el metal pulido o el pelo.



2. Modelos de iluminación para el pelo

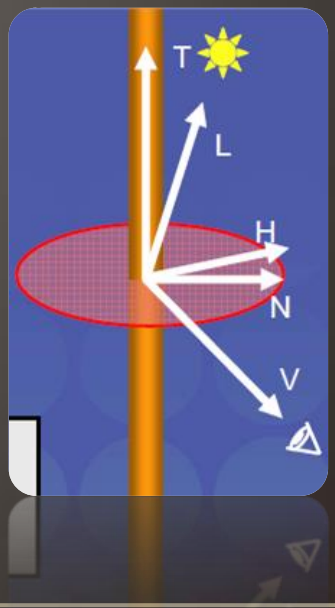
- Existen varios modelos de iluminación para el renderizado de pelo.
  - **Marchsner** – Modelo físico.
  - **Kayija-Kay** – Modelo fenomenológico.
  - **Scheuermann** – Modelo fenomenológico, combinación de Kayija-Kay con las observaciones de Marchsner.

2.1. Kayija-Kay

- Modelo que trata de simular fenomenológicamente la iluminación del pelo.
- Para ello emplea la tangente del cabello en vez de la normal para el cálculo de la iluminación especular.
- Asume que la normal se halla en el plano que forman la tangente y el vector de vista.

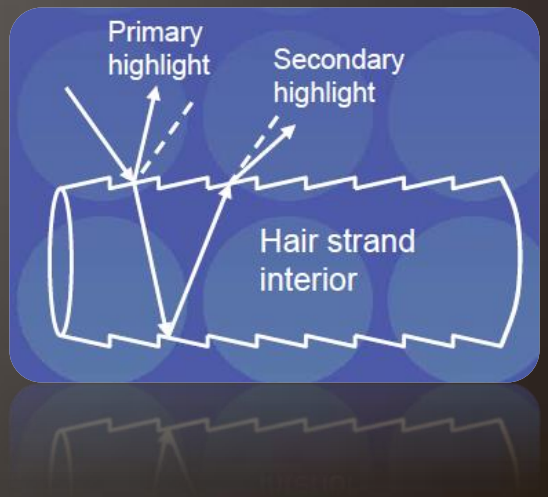
$$H = \frac{L + V}{|L + V|}$$

$$specular = \sin(T, H)^{shininess} = \sqrt{1 - \cos^2(T, H)}^{shininess}$$



2.2. Marchsner

- Modelo físico basado en el estudio de la refracción de la luz con el pelo.
  - Muy complejo de implementar.
- Realiza las siguientes observaciones:
  1. Existe un brillo especular primario orientado hacia la punta del cabello.
  2. Existe otro brillo especular secundario orientado hacia la raíz del cabello.
     1. Posee una apariencia “centelleante”.



2.3. Scheuermann

- Modelo de iluminación basado en Kayija-Kay que tiene en cuenta las observaciones de Marchsner.
  - Modelo también fenomenológico.
  - Intenta incluir las observaciones de los dos brillos especulares observados en el modelo de Marchsner.

3.1. Implementación

- Por simplicidad he decidido escoger el modelo de Scheuermann.
  - Fácil.
  - Rápido.
  - Resultados convincentes.
  - Ampliamente utilizado en videojuegos.
- Pasaré a explicar cada componente del modelo de iluminación.

3.1. Implementación – Término difuso

- El término difuso es el clásico lambertiano aplicando un bias y una escala para “simular” la refracción interna del pelo.

$$diffuse = \max(0, scale * N \cdot L + bias)$$

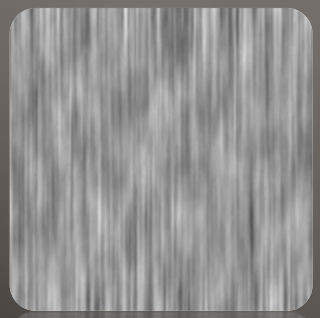
Término difuso

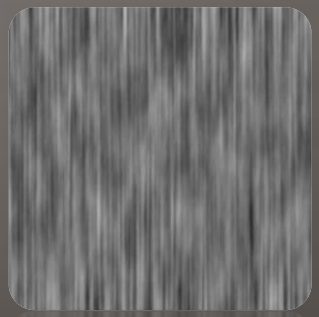
3.2. Implementación – Términos especulares

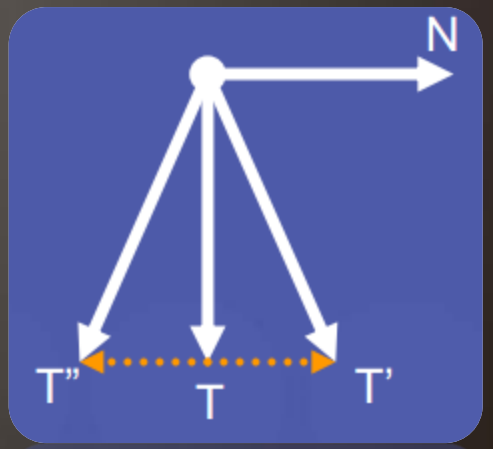
- Los término especulares se calculan con Kayija-Kay.
- Para simular el fenómeno de los dos brillos especulares, se calculan dos tangentes distintas desplazadas en la dirección de la normal, uno hacia la raíz y otro hacia la punta.

$$T' = T + N * Shift_1$$
$$T'' = T + N * Shift_2$$

- Los valores de Shift se obtienen desde una textura, transformados al rango [-1 1].







3.2. Implementación – Términos especulares

- Con estas tangentes se calculan los términos especulares 1 y 2. Cada uno de los brillos tiene su propio "shininess" y color.

$$Specular_1 = \max\left(0, L \cdot N * \sqrt{1 - (H \cdot T')^2}^{shininess_1}\right)$$

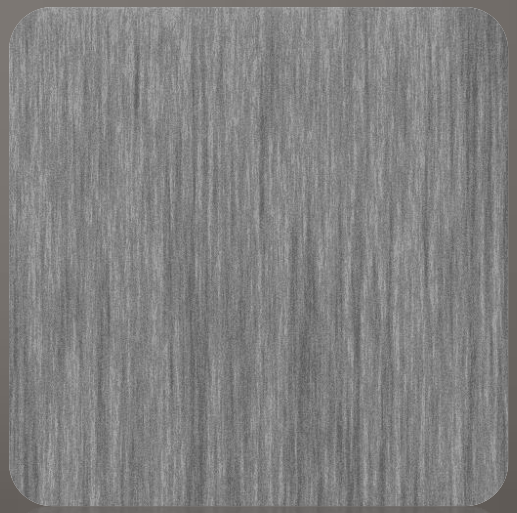
$$Specular_2 = \max\left(0, L \cdot N * \sqrt{1 - (H \cdot T'')^2}^{shininess_2}\right)$$

Término especular 1

Término especular 2

3.2. Implementación – Términos especulares

- Para dar esa sensación “centelleante” al brillo secundario, se modula el término con un mapa de ruido.
- Finalmente, la contribución de todos los brillos especulares se modulan con un Gloss-Map para simular la auto-oclusión.



Mapa de ruido

Gloss-Map

3.3. Implementación – Composición

- Finalmente, todos los términos se modulan con la textura base.
- El color final se calcula como:

$$C_{difusse} = base * diffuse * M_{diffuse}$$
$$C_{specular_1} = base * gloss * specular_1 * M_{specular_1}$$
$$C_{specular_2} = base * gloss * noise * specular_2 * M_{specular_2}$$
$$C = I_{light} * \left(C_{diffuse} + C_{specular_1} + C_{specular_2}\right)$$

Base

Difuso

Especular 1

Especular 2

Final

3.4. Implementación – Texturas

- Los mapas de ruido, shift 1, shift 2 y gloss están todos contenidos en la misma textura.

Canal	Rojo	Verde	Azul	Alfa
Significado	Ruido	Shift 1	Shift 2	Gloss

- El mapa de la base contiene además un canal alfa que no está siendo usado en estos momentos.

4. Simulación

- A modo de trabajo complementario a la implementación de este modelo, he implementado una simulación simple para animar los cabellos.
  - Basado en el paper "Fast Simulation of Inextensible Hair and Fur" [Muller et al. 2012].
- Para ello considero cada cabello del pelo como una pequeña cuerda inextensible.
  - La restricción se resuelve geométricamente para asegurar que estén a la distancia correcta.
- Sólo he hecho la implementación básica, no existe interacción entre pelos.

4.1. Simulación – Esquema general

- Inicialización:
  - Generar pelos sobre parte de la superficie de una esfera con una longitud determinadas.
- Simulación.
  - Por cada pelo
    - Predecir próxima posición (integrar).
    - Resolver restricciones.
    - Resolver colisiones (punto vs esfera).
    - Calcular velocidad.
    - Calcular posición.
    - Calcular tangente.
- Los detalles clave del algoritmo vienen descritos en el paper.

4.2. Simulación – Renderizado

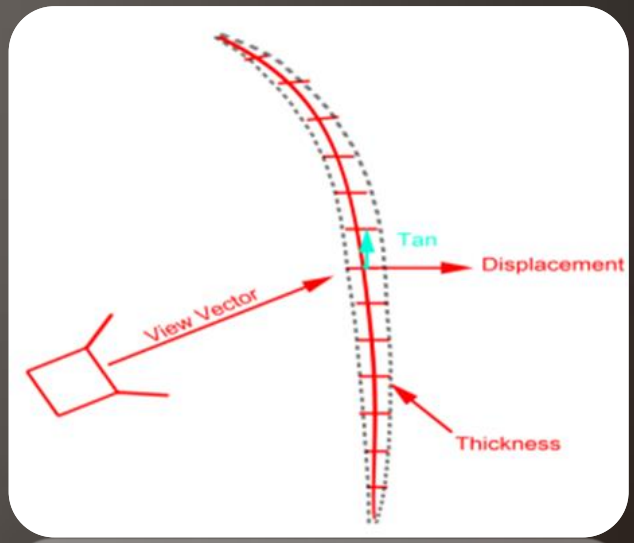
- Para poder dibujar los resultados, se deben volcar los datos de las partículas a tres buffers: posición, tangentes e índices.
- Esto nos dará información básicamente en forma de líneas.
  - Para poder visualizarlo correctamente, necesitamos generar geometría a partir de estas líneas.
- Para ello he usado un Geometry Shader, al cual le llega la información de cada uno de los segmentos de los cabellos y genera la geometría adecuada.

4.2. Simulación – Renderizado

- Por cada segmento del cabello que llega al Geometry Shader, se genera un quad "estirandolo" en la dirección perpendicular a la vista.
  - Pero sólo contamos con la información de la tangente y de la vista. Necesitamos averiguar la normal y la bitangente para poder hacerlo.
- Esto se calcula usando la técnica de ortonormalización de Gram-Schmidt:

$$N = V - T(V \cdot T)$$
$$B = T \times N$$

- Conociendo la bitangente se calculan los cuatro vértices del *quad* y se emite la geometría.
- Las coordenadas de textura se calculan en función de la distancia hasta la raíz y el ID del cabello.



5. Resultados

5. Resultados

5. Resultados

6. Trabajo futuro

- En cuanto a la iluminación:
  - La iluminación a veces es inconsistente debido a cómo se calculan los vectores de la geometría. Sería interesante buscar mejores métodos.
  - Buscar métodos para atenuar la iluminación cuando un cabello esté ocluido. ¿Mapas de sombras?
- En cuanto a la simulación:
  - Sin interacción entre cabellos no hay volumen. Esto provoca la aparición de artefactos al estar los cabellos solapados.
  - La implementación en CPU es algo lenta, implementarlo en GPU con Compute Shaders sería ideal.

7. ¿Preguntas?

8. Bibliografía

- **James T Kajiya and Timothy L Kay,** Rendering Fur With Three Dimensional Textures.
- **Marschner et. al.,** Light Scattering from Human Hair Fibers.
- **Thorsten Scheuermann,** Practical Real-Time Hair Rendering and Shading.
- **Muller et. al.,** Fast Simulation of Inextensible Hair and Fur.
- **Stephan Hodes and Wolfgang Engel,** Hair Rendering in Tomb Raider.

Gracias

Se puede ver el vídeo en https://www.youtube.com/watch?v=JP4TP-TVC0A