Documentación de la tercera fase

- Visualización y Entornos Virtuales -
- Curso 17/18 -

Andrea López Rodríguez

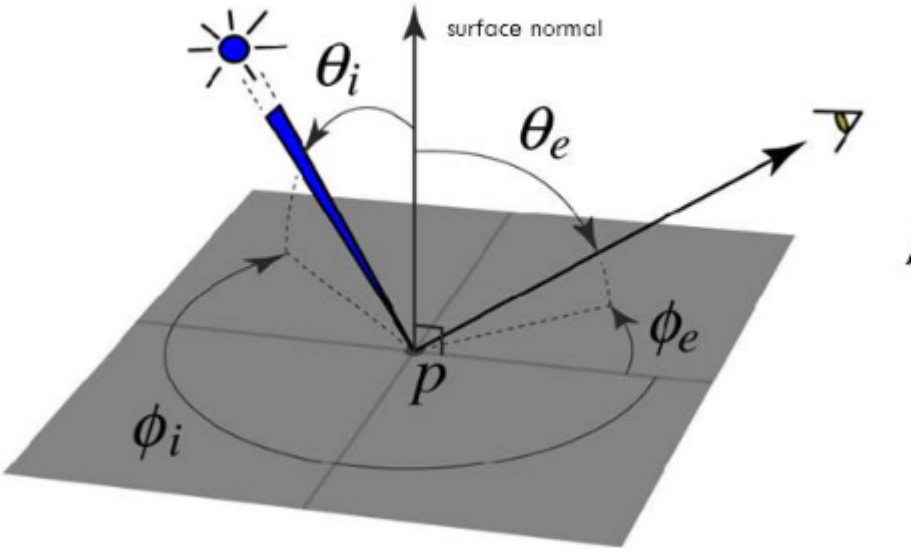
Contenidos

Introducción

Este documento trata el desarrollo de la práctica elegida para tercera fase del trabajo práctico de la asignatura Visualización y Entornos Virtuales. En él, se explica cómo se han implementado las funciones necesarias y su base teórica.

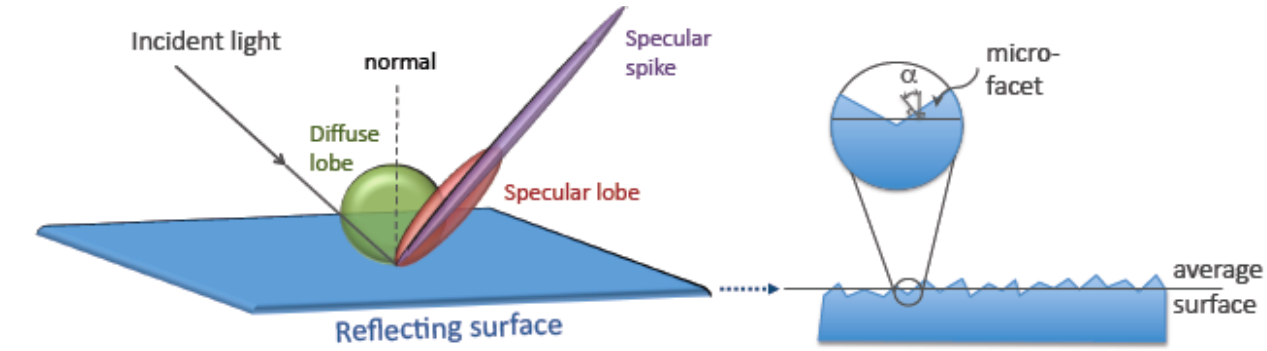
BRDF (Cook Torrance)

BDRF (bidirectional reflectance distribution function) es una función que define cómo se refleja la luz al iluminar una superficie opaca, y calcula la luz reflejada utilizando dos parámetros: la dirección de la **luz entrante** y la dirección de la **vista de observación**. Los modelos que utilizan esta función son los modelos BDRF, y se conocen varios modelos: el modelo *Lambertiano*, el modelo de reflectancia de *Phong* (seguido por el modelo de *Blinn-Phong)…* entre los que se incluye el modelo tratado en esta práctica: Modelo de ***Cook Torrance***.

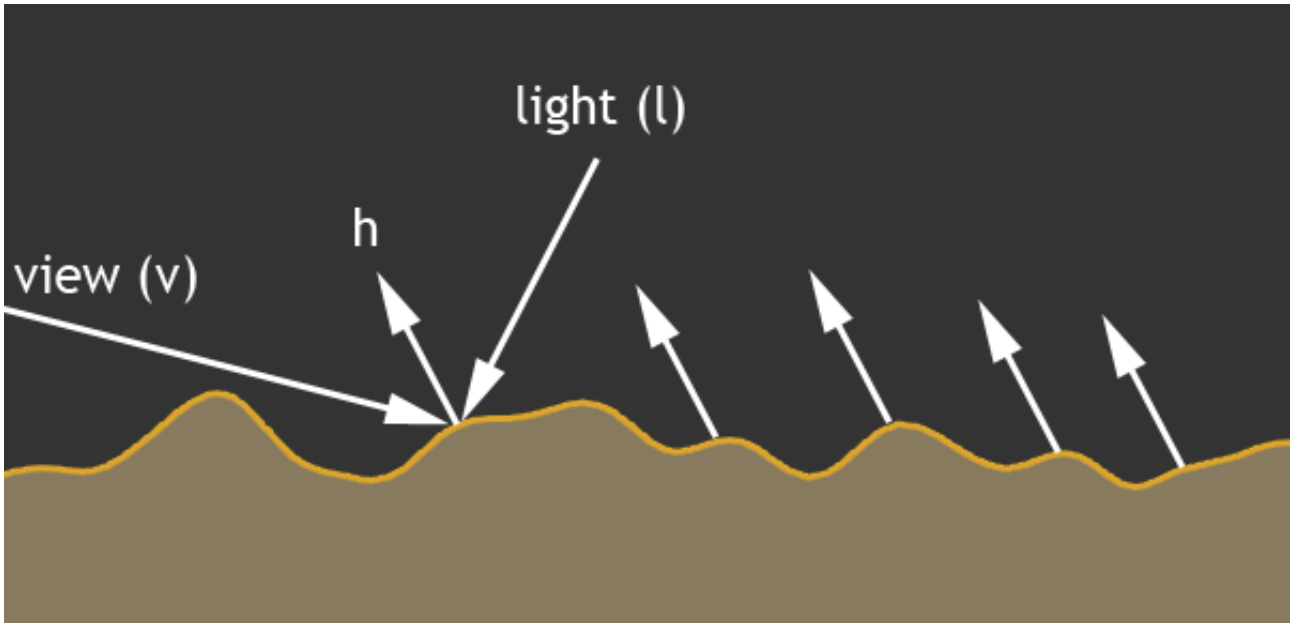


Modelo Cook Torrance

Es el modelo que presentaron Robert Cook y Ken Torrance conjuntamente en 1982. Es un modelo matemático que calcula la reflexión especular de la luz en una superficie, y busca una simulación real basada en **micro-facetas** que la componen. Las micro-facetas componen el conjunto de irregularidades que se pueden observar en una superficie a nivel microscópico. Estas micro-facetas pueden ser de varias formas, tales como depresiones, llanos… Y en estos, la luz se refracta en una dirección distinta. El conjunto de dichas direcciones da como resultado la apariencia final de la superficie. Este modelo se dice que es



Dibujo 1: Esquema de micro-faceta



Dibujo 2: Reflexión de la luz en las micro-facetas

Dependiendo de la orientación de las micro-facetas, el material puede parecer **difuso**, **especular** o **brillante** (acabado *glossy*). La composición del material también afecta a cómo la luz se comporta respecto a éste; por ejemplo, los metales absorben al momento la luz refractada, mientras que los no metales la dispersan y no la absorben totalmente.

Implementación

Para implementar este modelo de iluminación son necesarios dos archivos: el perfragment.vert, que ya se ha completado en prácticas anteriores (implementación del *fragment shader*), y el fichero cookTorrance.frag, que contiene todas las funciones referentes al cálculo de la reflexión, las luces y etc. Estas funciones son las siguientes:

- Modelo BDRF *Cook-Torrance*

$$f_r(\mathbf{l},\mathbf{v}) = f_{\text{diffuse}}(\mathbf{l},\mathbf{v}) \cdot k_d + f_{\text{specular}}(\mathbf{l},\mathbf{v}) \cdot (1 - k_d)$$

Siendo k_d el porcentaje de la reflexión difusa, y su complementario, 1 – k_d, el porcentaje de reflexión especular. Su rango de valores va desde 0 a 1, [0, 1]. Esta función se llama cook_torrance, y tiene como entrada el vector con los valores de la componente difusa y los vectores L, N y V. Devuelve un vector de tres posiciones con los valores del color.

- Cálculo del término difuso
  - ➔ En este caso, tomaremos la componente difusa como la componente difusa del material:

$$diffuse \approx theMaterial.diffuse$$

- Cálculo del término especular

$$f_{specular}(\mathbf{l},\mathbf{v}) = \frac{F(\mathbf{l},\mathbf{h}) \cdot G(\mathbf{l},\mathbf{v},\mathbf{h}) \cdot D(\mathbf{h})}{4(\mathbf{n}\cdot\mathbf{l})(\mathbf{n}\cdot\mathbf{v})}$$

Donde ***D(h)*** es la función de distribución de Beckman, ***G(l, v, h)*** el término geométrico, ***F(l, h)*** es el término de *Fresnel* y h es el vector medio normalizado, esto es, $\frac{l+v}{|l+v|}$. En el fichero, esta funcion se llama funcion_especular, devuelve un float como resultado y toma como parámetros el vector L (luz), el vector V (*viewDirection*) y el vector N (la normal).

- Distribución de *Beckman*

$$D(\mathbf{h}) = \frac{1}{\pi r^2 (\mathbf{h}\cdot\mathbf{n})^4} \exp\left(\frac{(\mathbf{h}\cdot\mathbf{n})^2 - 1}{r^2 (\mathbf{h}\cdot\mathbf{n})^2}\right)$$

Siendo r la el factor de rugosidad, cuyo rango va de 0 (suave) a 1 (rugoso). En la práctica, esta función se llama distribucion_beckman, y toma como parámetros el vector h (vector medio normalizado) y el vector normal. Devuelve un float como resultado.

- Función geométrica

$$G(\mathbf{l},\mathbf{v},\mathbf{h}) = min\left(1, \underbrace{\frac{2(\mathbf{n}\cdot\mathbf{h})(\mathbf{n}\cdot\mathbf{v})}{\mathbf{v}\cdot\mathbf{h}}}_{\text{masking}}, \underbrace{\frac{2(\mathbf{n}\cdot\mathbf{h})(\mathbf{n}\cdot\mathbf{l})}{\mathbf{v}\cdot\mathbf{h}}}_{\text{shadowing}}\right)$$

Teniendo como el enmascaramiento de una faceta por otra la componente *masking*, y como microfaceta no iluminada la componente *shadowing*. En la implementación, esta función se llama funcion_geometrica, y toma como parámetros el vector L (vector de luz), el vector V (vector de vista *viewDirection*) y los vectores h y la normal. Devuelve un float como resultado.

- Reflectancia de *Fresnel*

$$F(\mathbf{l},\mathbf{h}) \simeq F_{Schlick}(\lambda,\mathbf{l},\mathbf{h}) = \lambda + (1-\lambda)(1-(\mathbf{l}\cdot\mathbf{h}))^5$$

Donde la variable lambda (λ) el factor de reflectancia de *Fresnel*, cuyo rango va desde 0.01 hasta 0.95, [0.01, 0.95]. En el programa esta función se puede encontrar como fresnel, y toma como parámetros el vector de luz L y el vector medio normalizado h. Devuelve un float como resultado.

- Color del fragmento

$$\text{color del fragmento} = \text{color de la luz} \cdot (\mathbf{n}\cdot\mathbf{l}) \cdot f_r(\mathbf{l},\mathbf{v})$$

Esta fórmula se aplica en la iteración del programa principal main() (cuando iteramos entre los tipos de luces), donde el color del fragmento se guarda en un vector de tres posiciones color_sum(). Finalmente, este vector se le pasa a la variable gl_FragColor.

Bibliografía

- Apuntes de la profesora en la plataforma de eGela.
- Artículo de la Wikipedia sobre los modelos de iluminación y función de distribución de reflectancia bidireccional:
  - https://en.wikipedia.org/wiki/Bidirectional_reflectance_distribution_function
- Artículo de Coding::Labs sobre el modelo Cook-Torrance
  - http://www.codinglabs.net/article_physically_based_rendering_cook_torrance.aspx