Práctica 2 – Trazador de rayos

Julio Martín Saez

Raquel Peces Muñoz

# Funcionalidades implementadas

## Características obligatorias

### Algoritmo de ray casting e iluminación directa.

Hemos implementado un algoritmo básico de traza de rayos, en el cual se lanzan rayos por cada pixel de la pantalla a renderizar, y se calcula el aporte de cada una de las luces de la escena. El cálculo del color se ha realizado mediante la fórmula de iluminación de Phong.

### Incorporación de sombras.

En el momento del cálculo del color final del pixel, en el método de iluminación de Phong, se lanza un nuevo rayo desde el punto a cada una de las luces, de este modo podemos comprobar si el punto está oculto por otro objeto y si está en sombra o no.

El método “*collisionShadow”* es el encargado de esta comprobación, y devuelve un Vector, el cual puede tomar valores desde (1,1,1) en cuyo caso no hay sombra, hasta (0,0,0) en cuyo caso la sombra es completa. Si dicho Vector toma valores intermedios, es porque la sombra puede ser parcial al tratarse de un objeto “transparente”

### Algoritmo de traza de rayos recursivo con 4 rebotes.

El algoritmo envía rayos reflejados con respecto a la normal. Para ello, una vez calculado el punto de intersección se calcula la normal en dicho punto y se envía un rayo en la dirección reflejada. Cada rayo tienen una variable “*numRebounds*” la cual almacena el número de rebotes restantes que le queda a cada rayo, por tanto, al enviar un rayo en la dirección reflejada dicha variable decrementa, y de este modo queda un valor menor de rebotes restantes.

Si el rayo no tiene más rebotes pendientes, devolverá el color correspondiente, mientras que en las iteraciones anteriores devolverán su color sumado, de forma ponderada por su índice de refracción.

El número de rebotes máximos por rayo se pueden variar, cambiando la constante definida *REBOUNDS* en la clase “*RayTrace*”

### Geometría: esferas y triángulos individuales

Se ha implementado la colisión con esferas y triángulos individuales, utilizando las fórmulas de intersección de rayo con ambas primitivas.

También se han variado las clases “*SceneTriangle”* y “*SceneSphere”* para tener en cuenta las posibles transformaciones que se le aplican a dichos objetos como pueden ser translaciones, rotaciones y escalados. En dichas clases se han añadido métodos para obtener los valores correctos directamente.

### Materiales: lambertianos, con reflexión tipo Phong y mapeado de texturas

Se ha implementado el mapeado de texturas, para ello se hace uso de las coordenadas u, v y el método del material que obtiene el color de la textura. Las texturas se aplican en la componente difusa.

### Luces puntuales

A la hora de calcular la iluminación de Phong, se recorren todas las luces que hay en la escena, y se calcula el aporte de cada uno de ellas.

## Características opcionales

### Modelos externos descritos como mallas de triángulos

Se ha implementado la funcionalidad de poder añadir modelos definidos como mallas de triángulos. Dichos modelos mantienen todas las características implementadas obligatorias.

### Refracción