

Informática Gráfica

Juan Carlos Torres

Curso 2024/25

Dpto. Lenguajes y Sistemas Informáticos
Universidad de Granada

Disclaimer

You can edit this page to suit your needs. For instance, here we have a no copyright statement, a colophon and some other information. This page is based on the corresponding page of Ken Arroyo Ohori's thesis, with minimal changes.

CC BY-NC-SA

 This book is released into the public domain using the CC BY-NC-SA. This license enables reusers to distribute, remix, adapt, and build upon the material in any medium or format for noncommercial purposes only, and only so long as attribution is given to the creator. If you remix, adapt, or build upon the material, you must license the modified material under identical terms.

To view a copy of the CC BY-NC-SA code, visit:

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

Colophon

This document was typeset with the help of KOMA-Script and L^AT_EX using the kaobook class.

5

Iluminación

La luz es la radiación electromagnética que puede ser percibida por el ojo humano. De todo el espectro electromagnético el ojo solo es capaz de ver el intervalo comprendido entre los 380 nm y los 780 nm (Figura 5.1). Físicamente la luz es compleja, su comportamiento no se puede explicar totalmente por su naturaleza ondulatoria. Aunque la mayor parte de sus propiedades se corresponden con el comportamiento de una onda:

1. Difracción: La capacidad de la luz de doblar alrededor de obstáculos o de dispersarse al pasar por una pequeña abertura. Esto es similar a lo que ocurre con las ondas en el agua.
2. Interferencia: Cuando dos o más ondas de luz se superponen, pueden reforzarse mutuamente (interferencia constructiva) o cancelarse (interferencia destructiva). Esto da lugar a patrones de franjas claras y oscuras, como los que se observan en las películas delgadas de jabón.
3. Polarización: La luz vibra en todas las direcciones perpendiculares a su dirección de propagación. Sin embargo, es posible filtrar la luz seleccionando sola la que vibra en una dirección específica, creando luz polarizada.

Pero hay comportamientos de la luz que solo se pueden explicar si la consideramos un flujo de fotones:

1. Efecto fotoeléctrico: Cuando la luz incide sobre ciertos materiales, puede liberar electrones. Este fenómeno no puede explicarse con la teoría ondulatoria clásica, pero se entiende perfectamente si consideramos que la luz está compuesta por fotones, partículas de energía que pueden ceder su energía a los electrones.
2. Cuantización: La energía de la luz está cuantizada, es decir, solo puede tomar ciertos valores discretos. Esto se explica si consideramos que la luz está compuesta por fotones, cada uno de los cuales tiene una energía determinada.

Nos interesan dos propiedades de la luz: su intensidad y su color. La intensidad está relacionada con la energía que transporta (con el flujo de fotones). El color está relacionado con la longitud de onda. Vemos el color violeta en un extremo del espectro (40nm) y el rojo en el otro (750 nm) (Figura 5.1).

En nuestras retinas hay dos tipos de células sensibles: conos y bastones. Los bastones no son sensibles a la longitud de onda de la luz, pero tienen una sensibilidad mayor en condiciones de baja iluminación, son los que nos permiten ver cuando hay poca luz. Los conos son sensibles a la longitud de onda, pero necesitan mas intensidad de luz para excitarse que los bastones. Hay tres tipos de conos, cada uno con una curva de sensibilidad distinta: centrada en el azul, en el verde y en el rojo (Figura 5.2).

Nuestro cerebro percibe el color componiendo las señales recibidas por los tres tipos de conos. Esto nos permite generar todas las sensaciones de color que podemos percibir combinando tres fuentes de luz monocromáticas

5.1	Interacción luz/materia	37
5.2	Modelo de iluminación local	38
5.2.1	Luces	38
5.2.2	Reflexión y propiedades de los materiales	38
5.3	Sombreado	40
5.4	Iluminación en OpenGL	41
5.5	Iluminación en Unity	42
5.6	GLSL	43
5.7	Sombras y transparencias	45
5.7.1	Materiales Transparentes	45
5.8	Ejercicios	46

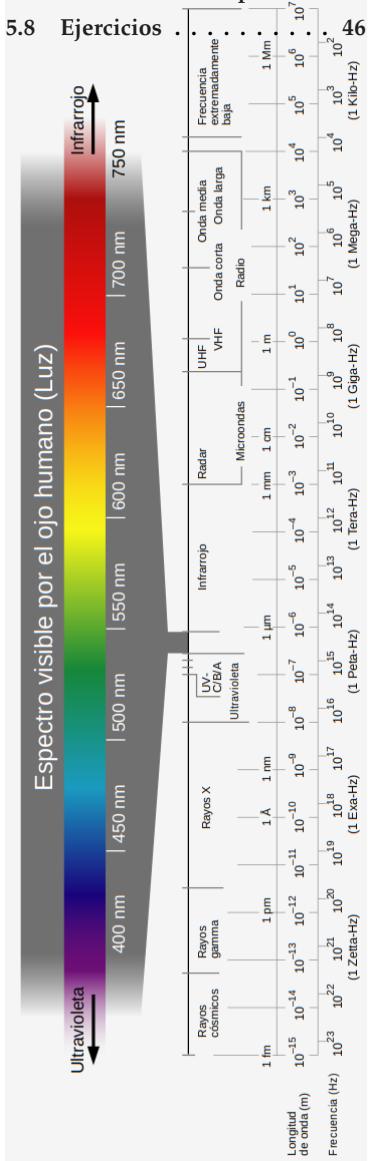


Figura 5.1: Espectro electromagnético <https://es.wikipedia.org/wiki/Luz>.

y nos permite representar los colores como una terna de intensidades de rojo, verde y azul. Estas intensidades se da en el intervalo (0,1) para que la representación sea independiente del dispositivo. Este modelo da lugar a un espacio para la descripción de los colores como un cubo en el espacio 3D.

Hay otros espacios de representación de colores. El espacio de color CIE fue creado por la Comisión Internacional de la Iluminación (CIE, por sus siglas en inglés) y se basa en una serie de experimentos que midieron cómo el ojo humano percibe los colores. Este espacio de color utiliza tres coordenadas (X, Y, Z) para representar cualquier color, y se ha convertido en un estándar internacional en la industria de la impresión, el diseño gráfico y la fotografía, ya que proporciona una forma precisa y objetiva de comunicar y gestionar el color. La coordenada Y representa la luminancia o brillo del color. Un valor de Y más alto indica un color más brillante, mientras que un valor más bajo indica un color más oscuro. Las coordenadas X y Z determinan la cromaticidad del color, es decir, el tono y la saturación. La combinación de X y Z es un diagrama de cromaticidad, que es una representación bidimensional de todos los colores visibles (figura 5.3). En este diagrama, los colores espectrales (arcoíris) se encuentran en el borde, mientras que el centro representa los colores acromáticos (blanco, gris y negro).

Los colores que se pueden generar a partir de tres primarios son los que se encuentran en el triángulo que delimitan en el diagrama cromático.

5.1. Interacción luz/materia

La luz nos permite ver nuestro entorno gracias a su interacción con los objetos. Cuando la luz incide en un objeto se puede producir los siguientes efectos:

1. Reflexión: La luz que incide en la superficie es reflejada por esta.
2. Absorción: La luz es transformada en calor en el objeto.
3. Transmisión: La luz penetra en el objeto. Esto ocurre cuando el objeto es total o parcialmente transparente.

Estos procesos no son excluyentes, de hecho lo normal es que se produzcan varios simultáneamente. Por ejemplo, cuando incide luz blanca sobre un objeto verde se refleja la radiación en la longitud de onda del verde y se absorbe el resto. En un objeto translúcido parte de la luz se transmite y parte se refleja.

La radiación que se transmite se refracta, esto es sufre un cambio de dirección que depende del índice de refracción de los dos medios. Además al propagarse dentro de un medio puede sufrir dispersión (scattering) si el medio es participativo, como el humo o la niebla.

Cuando la luz se refleja puede hacerlo en todas direcciones (reflexión difusa), o en la dirección simétrica a la dirección de incidencia (reflexión especular). Encontramos reflexión difusa en las superficies mate (como una pared) y reflexiones especulares en superficies metalizadas. En la figura 5.4 podemos apreciar ambos comportamientos además de transmisión con dispersión.

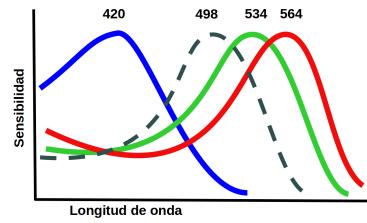


Figura 5.2: Curvas de sensibilidad espectral de los conos y los bastones (en gris discontinuo).

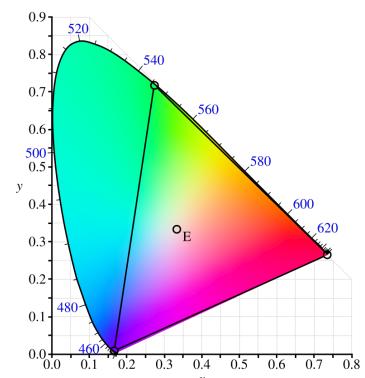


Figura 5.3: Diagrama de cromaticidad del espacio de color CIE (By BenRG, https://en.wikipedia.org/wiki/CIE_1931_color_space). Los colores que se pueden generar a partir de tres primarios son los que se encuentran dentro del triángulo formado por ellos.



Figura 5.4: Escultura de la diosa Aurora realizada por Denys Puech en 1900. Museo Orsay.

5.2. Modelo de iluminación local

Para generar imágenes es tenemos que calcular el color con el que debe verse cada punto de la escena, para ello es necesario disponer de un modelo matemático que nos permita estimar la interacción de la luz con la superficie de los objetos. A este modelo se le llama **modelo de iluminación**.

Vamos a ver un modelo de iluminación simple. Comenzaremos viendo como representar las luces y los materiales.

5.2.1. Luces

Una luz es una fuente de radiación electromagnética en el rango visible. Se caracteriza por su geometría, posición, intensidad en cada longitud de onda (espectro de emisión) y la direcciones en las que emite.

En el modelo mas simple consideramos que las luces son puntuales y que pueden estar colocadas en un punto de la escena, en cuyo caso su posición estará dada por las coordenadas de ese punto, o en infinito, simulando una fuente de luz lejana (como el sol). Las luces posicionales emiten luz en todas direcciones, las que están en el infinito (direccionales) solo en una dirección, con haces de luz (ver figura 5.5). Las luces direccionales se representan con coordenadas homogéneas que tienen valor 0 en la cuarta coordenada.

La forma mas simple de representar el espectro de emisión es como un color RGB, aunque esto implica una simplificación tanto de la distribución de frecuencias como de la intensidad, las limitaciones prácticas no son importantes salvo que el objetivo sea calcular la energía recibida en lugar de generar una imagen. Utilizar colores para representar la intensidad de la luz implica que solo podamos calcular el color resultante y la intensidad relativa respecto a otros puntos de la escena.

5.2.2. Reflexión y propiedades de los materiales

En el modelo de iluminación simple vamos a considerar únicamente materiales opacos, por tanto solo formalizaremos la reflexión. Aunque la reflexión en una superficie real puede ser compleja¹ en el modelo mas simple se representa el comportamiento como una combinación de reflexión difusa y especular pura.

Reflexión difusa

La reflexión difusa ocurre cuando la luz incide sobre una superficie rugosa o mate y se dispersa en múltiples direcciones distribuyendo la luz de manera uniforme en todas las direcciones 5.6, un objeto iluminado por una fuente de luz parecerá tener la misma intensidad de brillo sin importar desde dónde se observe.

Lla reflexión difusa se usa para simular el comportamiento de materiales como la madera, el yeso o el papel, que no tienen brillos especulares. La cantidad de luz reflejada depende del color del material, de la intensidad

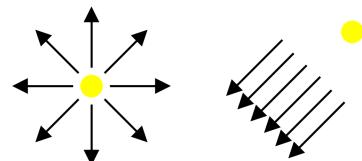


Figura 5.5: Luz posicional (izquierda) y direccional (derecha).

1: Para modelar la reflexión física en una superficie se usan funciones que indican para cada dirección de luz incidente la distribución de direcciones reflejadas (Bi-directional reflectance distribution function, BRDF).

de la luz incidente y de la orientación de la superficie respecto a la luz incidente.

Para modelar la reflexión difusa, utilizamos el modelo de Lambert, en el que la cantidad de luz reflejada es proporcional al coseno del ángulo entre la dirección de la luz incidente y la normal de la superficie. La intensidad de luz incidente por unidad de superficie depende de la orientación de esta respecto a la dirección de la luz incidente (ver figura 5.7).

En términos matemáticos, la intensidad de la luz reflejada (F_d) se puede expresar como:

$$F_d = I_i \cdot k_d \cdot \max((\mathbf{L} \cdot \mathbf{N}))$$

donde:

I_i es la intensidad de la luz incidente.

k_d es el coeficiente de reflexión difusa del material.

\mathbf{L} es el vector de la luz incidente.

\mathbf{N} es el vector normal de la superficie.

Este modelo simplificado permite calcular el color resultante en cada punto de la superficie, considerando únicamente la reflexión difusa. Es especialmente útil en aplicaciones donde se busca un equilibrio entre realismo y eficiencia computacional.

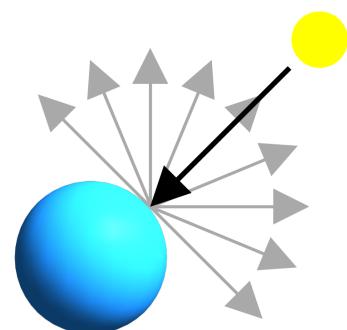


Figura 5.6: Reflexión difusa. Luz incidente en negro, reflejada en gris.

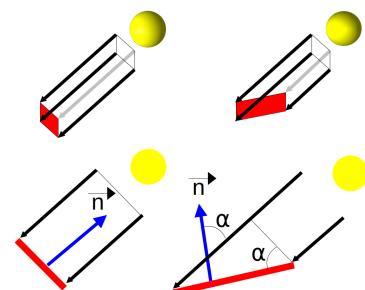


Figura 5.7: La intensidad de luz que incide en una superficie depende de su orientación respecto a la luz. El flujo en la configuración de derecha e izquierda es el mismo, pero la intensidad en cada unidad de superficie es máxima cuando es perpendicular a la dirección de incidencia.

Reflexión especular

La reflexión especular se produce cuando la luz incide sobre una superficie lisa y brillante, como una superficie metálica, reflejándose en una dirección específica. La reflexión especular mantiene la dirección de la luz reflejada de manera coherente (figura 5.8).

Para modelar la reflexión especular utilizamos el modelo de Phong, que simula el brillo y el reflejo de las superficies pulidas. Este modelo considera que la intensidad de la luz reflejada depende del ángulo entre la dirección de la luz reflejada y la dirección del observador.

En términos matemáticos, la intensidad de la luz reflejada F_s se puede expresar como:

$$F_s = I_i \cdot k_s \cdot \max((\mathbf{R} \cdot \mathbf{V}))^e$$

donde:

I_i es la intensidad de la luz incidente.

k_s es el coeficiente de reflexión especular del material.

\mathbf{R} es el vector de la luz reflejada.

\mathbf{V} es el vector de la dirección del observador.

e es el exponente de brillo, que determina la dispersión de la luz reflejada, o lo que es lo mismo la amplitud de la zona de brillo.

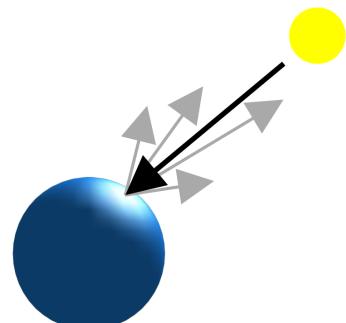


Figura 5.8: Reflexión Especular. La luz reflejada (en gris) se concentra en la dirección simétrica a la de incidencia.

Iluminación del ambiente

La componente ambiente es una parte fundamental del modelo de iluminación, representa la luz que llega a la superficie de forma indirecta (después de reflejarse en otros objetos de la escena). En el modelo de iluminación simple asumimos que ilumina todos los objetos de manera uniforme, sin importar su orientación o posición.

En términos matemáticos, la intensidad de la luz ambiente F_a se puede expresar como:

$$F_a = I_a \cdot k_a$$

donde:

I_a es la intensidad de la luz ambiente global en la escena.

k_a es el coeficiente de reflexión ambiente del material.

La componente ambiente es crucial para evitar que las áreas en sombra de los objetos aparezcan completamente negras, proporcionando así un nivel básico de iluminación que mejora el realismo de la escena.

Modelo de iluminación

En el modelo de iluminación de Phong, la luz total que incide sobre un punto de la superficie se calcula sumando las componentes ambiente, difusa y especular. La componente ambiente asegura que incluso las áreas no directamente iluminadas por una fuente de luz específica reciban algo de iluminación, contribuyendo a una representación más natural de los objetos en la escena.

Si hay mas de una fuente de luz se suman las contribuciones de todas ellas.

5.3. Sombreado

El calculo del color de los objetos de la escena (sombreado o shading) implica la aplicación del modelo de iluminación y puede realizarse a diferente nivel: caras, vértices o píxeles la figura 5.10 muestra el resultado de dibujar el mismo objeto usando cada uno de ellos.

El sombreado de caras (Flat Shading) asigna un solo color a cada cara. Este color se calcula utilizando la normal de la cara, sus reflectividades, un punto cualquiera de la cara y la fuente de luz. Este método es rápido y sencillo de implementar, pero produce bordes visibles entre las caras y color constante en las caras. Es ideal para modelos formados por objetos poliédricos donde la velocidad es crucial.

El sombreado de vértices (Gouraud Shading) calcula el color en cada vértice del polígono utilizando las normales y posiciones de cada vértice y luego interpola estos colores a lo largo de las caras. Esto suaviza las transiciones entre caras, eliminando los bordes visibles. Sin embargo, puede no capturar correctamente los reflejos especulares, ya que estos pueden quedar suavizados. Este método es comúnmente utilizado en

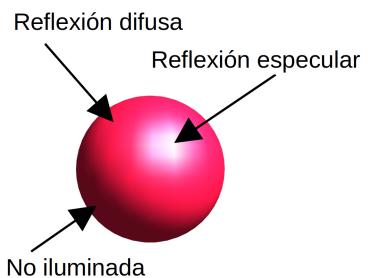


Figura 5.9: La zona no iluminada de la esfera no se ve negra por la iluminación ambiente.

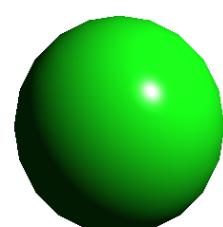


Figura 5.10: Métodos de sombreado. Arriba Flat, Gouraud en el centro, Phong Abajo.

videojuegos y aplicaciones interactivas donde se requiere un balance entre calidad y rendimiento.

El sombreado de píxeles (Phong Shading) calcula el color en cada píxel de la superficie. Determina la normal en el pixel interpolando las normales de los vértices y aplica el modelo de iluminación en cada píxel. Este método produce resultados muy realistas, con reflejos especulares precisos, pero es más costoso computacionalmente. Se utiliza en renderizados de alta calidad donde el realismo es prioritario, como en películas y visualizaciones arquitectónicas, implementándose en GPU.

5.4. Iluminación en OpenGL

OpenGL como parte de la funcionalidad fija (pre-programada), incluye una implementación de un modelo de iluminación simple, que tiene en cuenta la iluminación directa de las fuentes de luz, calculando: reflexión difusa, reflexión especular, iluminación ambiente y emisividad (luz emitida por la superficie del objeto). Para que se calcule la iluminación es necesario activarlo²:

```
glEnable(GL_LIGHTING);
```

Los colores, materiales e intensidades se representan en *rgb* o *rgba* con valores normalizados entre 0 y 1. Cuando hay mas de una fuente de luz o reflectividades altas es posible que el color resultante sea mayor que uno, en este caso las componentes mayores que uno se truncan generando una imagen sobre-expuesta (la imagen inferior de la figura 5.11 esta sobre-expuesta).

Las implementaciones de OpenGL están obligadas a gestionar un número mínimo de 8 fuentes de luz. Cada una de ellas se referencia con un identificador **GL_LIGHT0**, **GL_LIGHT1**, ..., **GL_LIGHT7**. Cada una de estas fuentes de luz puede encenderse y apagarse de forma individual, con:

```
glEnable(GL_LIGHTn) ; // enciende la enesima fuente de luz
glDisable(GL_LIGHTn) ; // apaga la enesima fuente de luz
```

De cada fuente de luz se puede especificar su posición y su intensidad. La posición se da con la función **glLightfv**:

```
GLfloat pos[4] = { 20.0, 10.0, 20.0, 0.0 };
glLightfv(GL_LIGHTn, GL_POSITION, pos);
```

a la que se le indica la luz que se está modificando y un vector 4D con su posición en coordenadas homogéneas. Si la coordenada homogénea (*w*) es cero, la fuente está en el infinito, y la luz incide en la dirección del vector (*x*, *y*, *z*). En caso contrario es una luz posicional.

Las luces están integradas en el grafo de escena. Es decir, les afectan las transformaciones geométricas activas cuando se ubican en la escena usando la función **glLightfv**.

A cada fuente de luz OpenGL le asocia tres intensidades que indican como contribuye en el cálculo de la iluminación difusa, especular y ambiente³. Cada una de estas intensidades se representa como un color:

2: Con la iluminación desactivada, el color de las primitivas dibujadas depende de una terna RGB del estado interno, que se modifica con **glColor**.

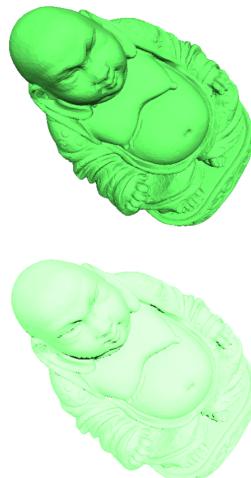


Figura 5.11: Dos imágenes del mismo modelo. La inferior está sobre-expuesta.

3: También se pueden especificar focos, que son luces posicionales que iluminan en una dirección.

```
float color[4] = { 0.8, 0.0, 1, 1 };
glLightfv(GL_LIGHTn, GL_AMBIENT, color);
glLightfv(GL_LIGHTn, GL_DIFFUSE, color);
glLightfv(GL_LIGHTn, GL_SPECULAR, color);
```

Al iniciar OpenGL solo está inicializada y encendida la luz **GL_LIGHT0**.

Materiales en OpenGL

OpenGL asocia materiales a los objetos. Cada material se caracteriza por un conjunto de reflectividades: ambiente, difusa y especular⁴. Los objetos pueden tener dos materiales: uno para las caras delanteras y otro para las caras traseras⁵. Para asignar las reflectividades se usa la función **glMaterialfv**, a la que se le indica las caras a las que se va aplicar (**GL_FRONT**, **GL_BACK** o **GL_FRONT_AND_BACK**), el parámetro a fijar (**GL_AMBIENT**, **GL_DIFFUSE**, **GL_SPECULAR**) y un color. Para indicar el exponente de brillo se usa la función **glMaterialf**, indicando la cara, el parámetro **GL_SHININESS** y el valor del exponente). El siguiente código asigna un material verde con brillos blancos:

```
1 GLfloat rd[4]={0.2,1.0,0.2,1.0}, re ={0.6,0.6,0.6,1.0},
2      ra ={0.0,0.2,0.0,1.0};
3 glMaterialfv( GL_FRONT, GL_AMBIENT, ra ) ;
4 glMaterialfv( GL_FRONT, GL_DIFFUSE, rd ) ;
5 glMaterialfv( GL_FRONT, GL_SPECULAR, re ) ;
6 glMaterialf( GL_FRONT, GL_SHININESS, 5 ) ;
```

Los materiales se almacenan en el estado de OpenGL y se aplica a todos los objetos que se dibujen después de asignarlos.

4: También es posible asignar emisividad a los materiales

5: Este solo se usa si no se están eliminando las caras traseras

Sombreado en OpenGL

OpenGL permite, en la funcionalidad fija, realizar el sombreado plano y de Gouraud (por vértices). La función **glShadeModel** permite seleccionar el método de sombreado activo en cada momento (que afectará a todas las primitivas que se dibujen posteriormente). Admite dos valores como argumento **GL_FLAT** y **GL_SMOOTH**.

5.5. Iluminación en Unity

Unity3D utiliza un sistema de iluminación flexible y potente que permite simular diferentes tipos de luces y materiales en una escena. A continuación, se describen los conceptos básicos de luces y materiales en Unity3D.

Luces en Unity3D

Unity3D soporta varios tipos de luces, añadiendo a los tipos existentes en OpenGL luces de área, que emiten luz desde una superficie rectangular. Las luces se crean como objetos en el grafo de escena y están afectadas por las transformaciones de sus nodos padre. Se pueden crear desde la interfaz o desde el código. En la interfaz añadir como *GameObject > Light*.

Fijar parámetros en *Inspector*. La intensidad es independiente del color y se da normalizada.

Para crear y configurar una luz en Unity3D, se puede utilizar el siguiente código en C#:

```
Light myLight = gameObject.AddComponent<Light>();
myLight.type = LightType.Point;
myLight.color = Color.white;
myLight.intensity = 1.0f;
myLight.range = 10.0f;
```

Materiales en Unity3D

Los materiales en Unity3D se crean y configuran utilizando el sistema de shaders de Unity. El color difuso del material es su *Albedo*, la reflectividad especular se controla con dos parámetros *Metalicidad*, que controla cuán metálico es el material y *Suavidad*, que controla si el objeto es pulido o rugoso. Los materiales se crean como *Assets* y sus parámetros se editan en Inspector.

También se puede asignar un material a un objeto desde el código del programa:

```
Renderer renderer = gameObject.GetComponent<Renderer>();
Material material = new Material(Shader.Find("Standard"));
material.color = Color.green;
material.SetFloat("_Metallic", 0.5f);
material.SetFloat("_Glossiness", 0.5f);
renderer.material = material;
```

Sombreado en Unity3D

Para cambiar el método de sombreado en Unity3D se debe cambiar el shader (el programa en GPU que dibuja la superficie). Cada material tiene asociado un shader. Si se quiere forzar que en una malla se aparezcan discontinuidades en determinadas aristas independientemente del shader con el que se dibuja se deben duplicar los vértices.

5.6. GLSL

Las versiones recientes de OpenGL no implementa un modelo de iluminación. El cálculo de iluminación debe programarse en shaders. Los shaders son programas que se ejecutan en la GPU para generar las imágenes. Un shader está compuesto de dos programas: un programa para procesar vértices (*vertex shader*) y un programa para procesar la imagen a nivel de pixel (*fragment shader*). Las primitivas llegan al *vertex shader*. La salida de este se rasteriza y se procesa en el *fragment shader*.

GLSL (OpenGL Shading Language) es el lenguaje de programación utilizado para escribir shaders en OpenGL. Los shaders se cargan y compilan en tiempo de ejecución del programa. Abordaremos su estudio más adelante, de momento para clarificar el concepto se muestra el código

de un programa simple que dibuja una esfera usando sombreado de Phong:

```

1 // Variables para los materiales
2 GLfloat mat_diffuse[] = { 0.6f, 0.6f, 0.6f, 1.0f }, GLfloat
   mat_specular[] = { 1.0f, 1.0f, 1.0f, 1.0f }, GLfloat
   mat_shininess[] = { 50.0f };
3 GLuint phongShaderProgram; // Shader program para Phong (con GLSL)
4
5 void loadShaders() {      // Carga los shaders Phong
6   const char* vertexShaderSource = "#version 120\n"
7   "varying vec3 normal, lightDir, viewDir;" 
8   "void main() {"
9     "    vec4 vertexPos = gl_ModelViewMatrix * gl_Vertex;" 
10    "    lightDir = vec3(gl_LightSource[0].position - vertexPos);"
11    "    viewDir = vec3(-vertexPos.xyz);"
12    "    normal = gl_NormalMatrix * gl_Normal;" 
13    "    gl_Position = ftransform();"
14  "}";
15
16  const char* fragmentShaderSource = "#version 120\n"
17  "varying vec3 normal, lightDir, viewDir;" 
18  "void main() {"
19    "    vec3 N = normalize(normal);"
20    "    vec3 L = normalize(lightDir);"
21    "    vec3 V = normalize(viewDir);"
22    "    vec3 R = reflect(-L, N);"
23    "    float lambertTerm = max(dot(N, L), 0.0);"
24    "    vec4 diffuse = lambertTerm * gl_FrontMaterial.diffuse;" 
25    "    vec4 ambient = 0.1*gl_FrontMaterial.ambient;" 
26    "    vec4 specular = vec4(0.0);"
27    "    if (lambertTerm > 0.0) {"
28      "        float spec = pow(max(dot(R, V), 0.0),
29      gl_FrontMaterial.shininess);"
30      "        specular = spec * gl_FrontMaterial.specular;" 
31      "    }"
32    "    gl_FragColor = ambient + diffuse + specular;" 
33  "}";
34
35 GLuint vertexShader = glCreateShader(GL_VERTEX_SHADER);
36 GLuint fragmentShader = glCreateShader(GL_FRAGMENT_SHADER);
37 glShaderSource(vertexShader, 1, &vertexShaderSource, NULL);
38 glShaderSource(fragmentShader, 1, &fragmentShaderSource, NULL);
39
40 glCompileShader(vertexShader);
41 glCompileShader(fragmentShader);
42
43 phongShaderProgram = glCreateProgram();
44 glAttachShader(phongShaderProgram, vertexShader);
45 glAttachShader(phongShaderProgram, fragmentShader);
46
47 glLinkProgram(phongShaderProgram);
48 }
49
50 // Dibuja una esfera con sombreado Phong usando shaders GLSL
51 void drawPhongShadedSphere() {
52   glUseProgram(phongShaderProgram);
53   glutSolidSphere(3.0, 20, 20);
54   glUseProgram(0); // Desactiva los shaders

```

54 | }

5.7. Sombras y transparencias

Las sombras y la representación de materiales transparentes permiten aumentar el realismo visual en las escenas renderizadas.

Cálculo de Sombras

El *shadow mapping* es una técnica de generación de sombras simple basada en renderizar la escena desde la perspectiva de la luz, crea una textura de profundidad que almacena las distancias desde la luz a los objetos. A continuación renderiza la escena desde la perspectiva de la cámara. Para cada fragmento calcula su distancia a la luz y la compara con el valor almacenado en la textura de profundidad para determinar si el fragmento está en sombra (figura 5.13). Para usar este algoritmo en OpenGL se debe implementar en el shader. Unity 3D incorpora la generación de sombras utilizando este método.

5.7.1. Materiales Transparentes

La representación de materiales transparentes se realiza mediante el uso de la cuarta componente del color (α). Un valor 1 de α indica que el material es opaco, un valor cero corresponde a una superficie totalmente transparente. La figura ?? muestra una escena con dos superficies transparentes.

OpenGL procesa la transparencia usando el modo de mezcla (*blending*) que permite combinar el color del fragmento que se va a dibujar con el color ya dibujado en el pixel. Por tanto, es necesario dibujar antes los objetos opacos que los transparentes. El siguiente código muestra un esquema del proceso.

```

1 ...
2 glEnable(GL_BLEND);
3 ...
4 void dibuja()
5 ...
6 //Dibujar objetos opacos
7 ...
8 ...
9 ...
10 // Dibujar objetos transparentes
11 ...
12 ...
13 ...
14 ...
15 ...
16

```

La función `glBlendFunc` indica que ecuación de combinación se debe utilizar. En este caso se indica:

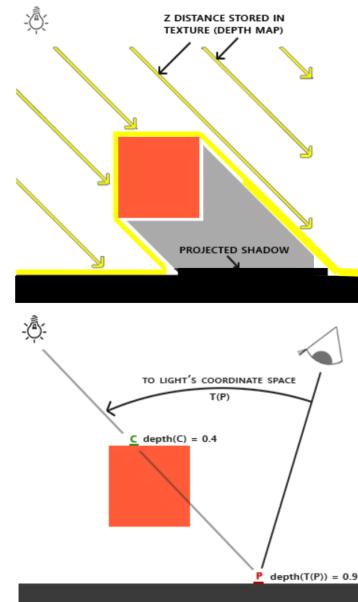


Figura 5.12: Métodos de sombreado. Arriba Flat, Gouraud en el centro, Phong Abajo.

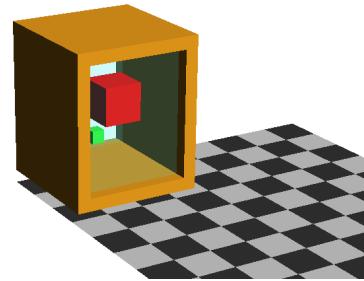


Figura 5.13: Escena con dos cristales semitransparentes.

$$color = Source_{\alpha}Source_{Color} + (1 - Source_{\alpha})Destination_{Color}$$

Donde *Source* es el fragmento a dibujar, y *Destination* es el contenido actual de Frame buffer

Para crear materiales transparentes en Unity3D se debe cambiar el *renderingmode* del material a *transparente* y asignar α en el Albedo.

5.8. Ejercicios

1. ¿Todos los monitores reproducen el mismo color para cada terna RGB?
2. ¿Se puede hacer que reproduzcan los mismos colores?
3. ¿Porque no vemos colores cuando hay poca luz?
4. ¿Que implicaciones tiene que se represente la intensidad de las fuentes de luz con valores RGB normalizados?
5. ¿Que pasa si se añaden muchas fuentes de luz a una escena?
6. ¿Como se puede colocar una luz posicional en el extremo del modelo articulado del ejercicio 5.10 de forma que se mueva con el modelo.
7. En el ejercicio anterior se quiere añadir una superficie plana debajo del modelo de forma que se aprecie el cambio de iluminación en el suelo al mover objeto. ¿Como se debería hacer?.
8. La imagen de la figura 5.14 corresponde a la visualización de un poliedro regular de radio 1, centrado en el origen, que aproxima una esfera.¿Que fuentes de luz y que propiedades de material se han usado para generarla?
9. ¿Que cambios se deben hacer en el código del ejercicio anterior para que la visualización se parezca a una esfera?
10. Identifica donde está colocada la fuente de luz en el código base usado en prácticas.

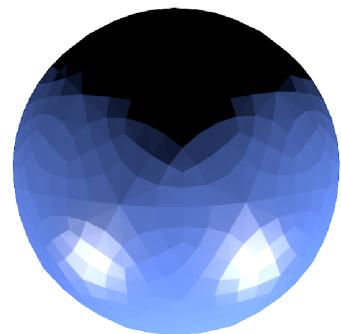


Figura 5.14: Imagen del ejercicio 10.

Bibliografía

- [Bot+10] Mario Botsch et al. *Polygon mesh processing*. CRC press, 2010.
- [Cig+08] Paolo Cignoni et al. «MeshLab: an Open-Source Mesh Processing Tool». En: *Eurographics Italian Chapter Conference*. Ed. por Vittorio Scarano, Rosario De Chiara y Ugo Erra. The Eurographics Association, 2008. ISBN: 978-3-905673-68-5. doi: [10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136](https://doi.org/10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136).
- [Kil96] Mark J Kilgard. *The OpenGL utility toolkit (GLUT) programming interface API version 3*. [accedido 26-Abril-2024]. 1996. URL: <https://www.opengl.org/resources/libraries/glut/glut-3.spec.pdf>.
- [KSS16] John Kessenich, Graham Sellers y Dave Shreiner. *OpenGL Programming Guide: The official guide to learning OpenGL, version 4.5 with SPIR-V*. [accedido 26-Abril-2024]. Addison-Wesley Professional, 2016. URL: <https://www.cs.utexas.edu/users/fussell/courses/cs354/handouts/Addison.Wesley.OpenGL.Programming.Guide.8th.Edition.Mar.2013.ISBN.0321773039.pdf>.
- [Shr+07] Dave Shreiner et al. *OpenGL (R) programming guide: The official guide to learning OpenGL (R), version 2.1*. [accedido 26-Abril-2024]. Addison-Wesley Professional, 2007. URL: <https://theswissbay.ch/pdf/Gentoomen%20Library/Game%20Development/Programming/The%20Official%20Guide%20to%20Learning%20OpenGL%20Version%202.1.pdf>.