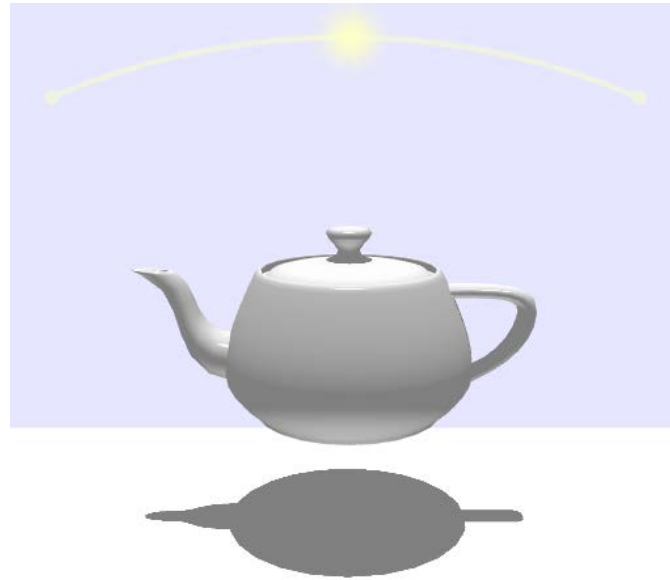




Universidad
Rey Juan Carlos

Escuela Técnica Superior
Ingeniería de Telecomunicación



Gráficos y visualización 3D

7. Iluminación con WebGL

JOSÉ MIGUEL GUERRERO HERNÁNDEZ

EMAIL: JOSEMIGUEL.GUERRERO@URJC.ES

Índice de contenidos

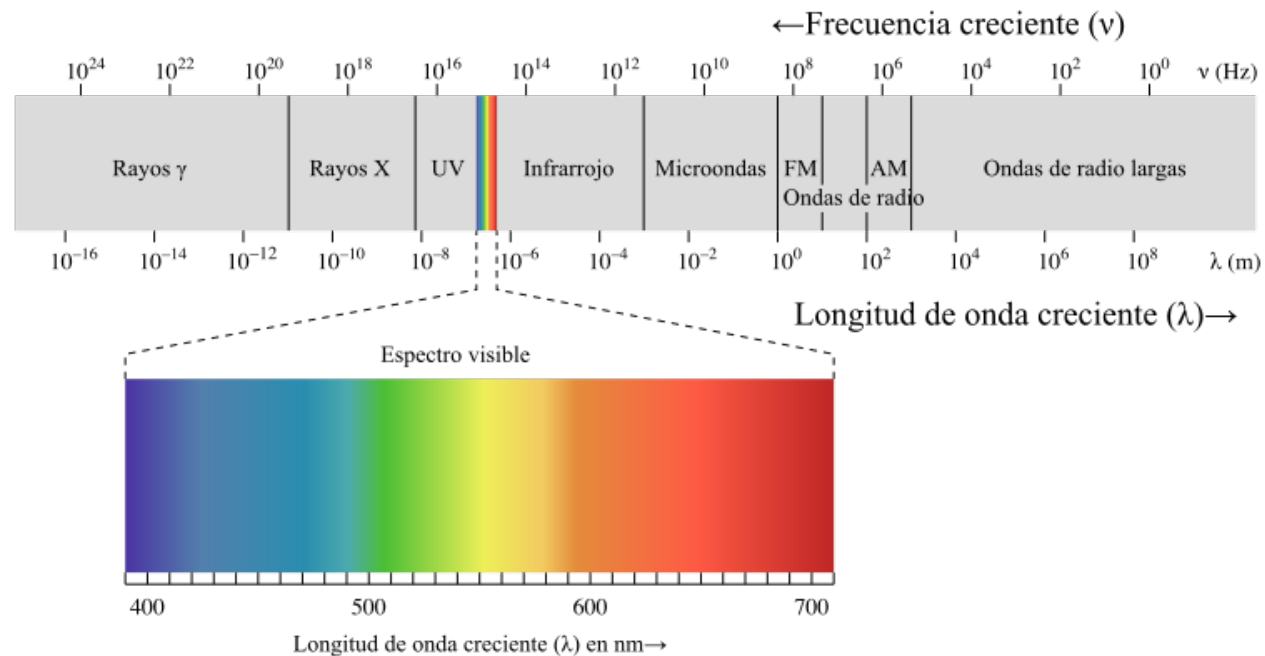
1. Introducción
2. Tipos de fuente de luz
3. Referencia API GLSL ES
4. Referencia glmatrix
5. Ejemplo: cubo con luz ambiental
6. Ejemplo: cubo con luz direccional
7. Ejemplo: cubo con luz ambiental y direccional
8. Matriz mágica: inversa y transpuesta
9. Ejemplo: luz direccional y matriz mágica
10. Ejemplo: cubo con luz puntual
11. Ejemplo: cubo con degradado realista

Índice de contenidos

1. Introducción
2. Tipos de fuente de luz
3. Referencia API GLSL ES
4. Referencia glmatrix
5. Ejemplo: cubo con luz ambiental
6. Ejemplo: cubo con luz direccional
7. Ejemplo: cubo con luz ambiental y direccional
8. Matriz mágica: inversa y transpuesta
9. Ejemplo: luz direccional y matriz mágica
10. Ejemplo: cubo con luz puntual
11. Ejemplo: cubo con degradado realista

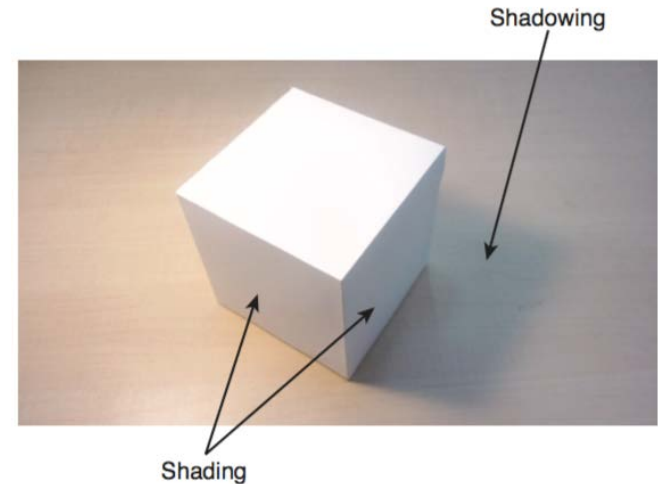
1. Introducción

- La **luz** es la parte de la radiación electromagnética que puede ser percibida por el ojo humano



1. Introducción

- Cuando una onda de luz alcanza un objeto en el mundo real, parte de la luz es reflejada por la superficie del objeto, y cuando esa luz llega a nuestros ojos nuestro cerebro interpreta el color
- Hay varios fenómenos que se producen cuando la luz alcanza un objeto:
 - **Degradación** (*shading*): diferencia de colores de una superficie
 - **Sombreado** (*shadowing*): región de oscuridad



1. Introducción

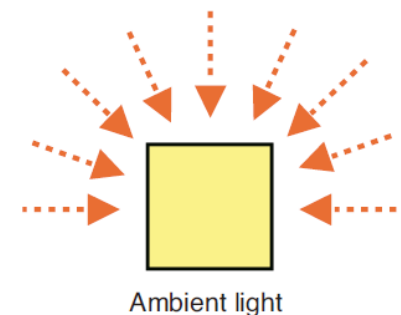
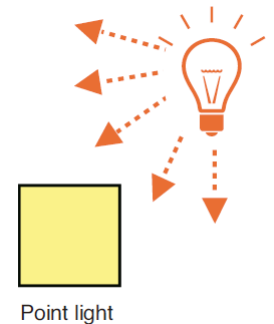
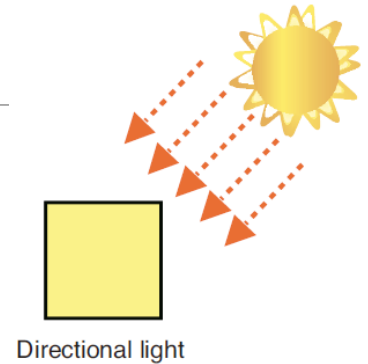
- En gráficos 3D consideramos ambos aspectos:
 - El término ***shading*** se usa para describir el proceso que recrea el fenómeno por el cual el color de las distintas caras de una figura difiere a causa de la luz
 - El término ***shadowing*** se usa para describir el proceso que recrea el fenómeno por el cual un objeto proyecta su sombra
- En la recreación de estos fenómenos, es importante tener en cuenta aspectos:
 - El tipo de **fuentes de luz** que está emitiendo
 - La luz es **reflejada** por las caras de un objeto

Índice de contenidos

1. Introducción
2. Tipos de fuente de luz
 - I. Tipos
 - II. Reflexión difusa
 - III. Reflexión ambiental
 - IV. Reflexión difusa y ambiental
3. Referencia API GLSL ES
4. Referencia glmatrix
5. Ejemplo: cubo con luz ambiental
6. Ejemplo: cubo con luz direccional
7. Ejemplo: cubo con luz ambiental y direccional
8. Matriz mágica: inversa y transpuesta
9. Ejemplo: luz direccional y matriz mágica
10. Ejemplo: cubo con luz puntual
11. Ejemplo: cubo con degradado realista

2. Fuente de luz - Tipos

1. **Direccional:** sus rayos de luz son paralelos, debido a la distancia infinita de la fuente de la que provienen, como el sol. Por lo tanto, sólo se puede especificar utilizando **dirección y color**
2. **Puntual:** emite luz en todas direcciones desde un único punto: bombillas, lámparas, llamas. La luz se atenúa con la distancia. Se especifica por medio de **posición y color**
3. **Ambiental** (o indirecta): es un modelo de luz emitida por las luces anteriores (direccional o puntal), reflejada por otros objetos y que alcanza los objetos de forma indirecta. No tiene ni dirección ni posición, se especifica únicamente por el **color**

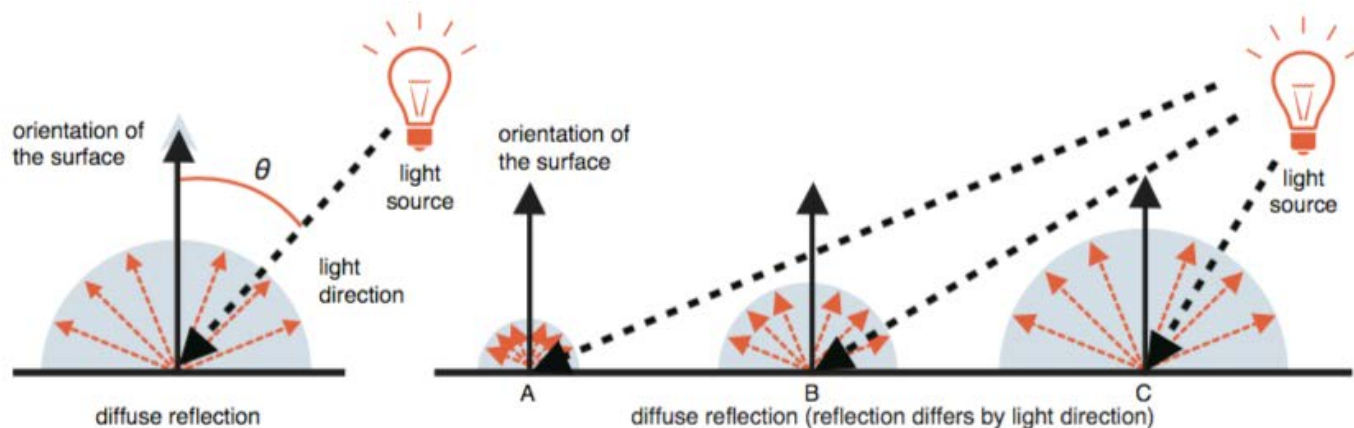


2. Fuente de luz - Tipos

- El tipo de fuente de luz también tendrá incidencia directa en el tipo de **luz reflejada**
- Hay dos tipos principales de reflexión (que serán modelados en un sistema 3D):
 - Reflexión **difusa**. Se trata de la luz reflejada procedente de una fuente de luz direccional o puntual
 - Reflexión **ambiental**. Se trata de la luz reflejada procedente de una fuente de luz ambiental

2. Fuente de luz – Reflexión difusa

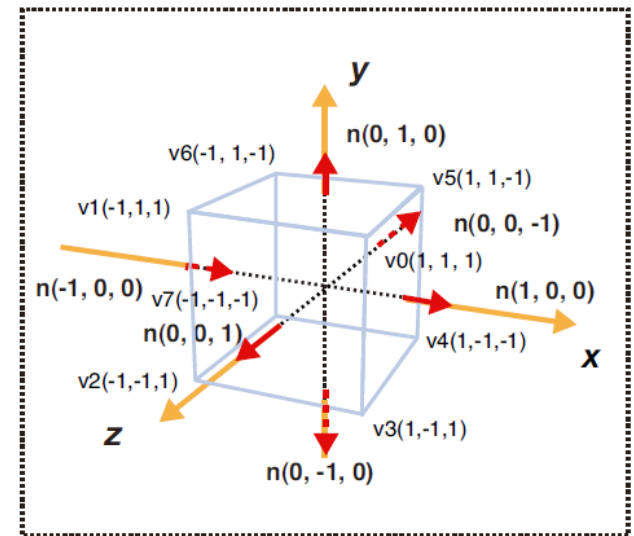
- En la reflexión difusa, la luz es reflejada equitativamente en todas direcciones
- Si la superficie es lisa, toda la luz que incide en la superficie es reflejada



$$(\text{color_superficie_reflexión_difusa}) = (\text{color_luz}) \times (\text{color_base_superficie}) \times \cos \theta$$

2. Fuente de luz – Reflexión difusa

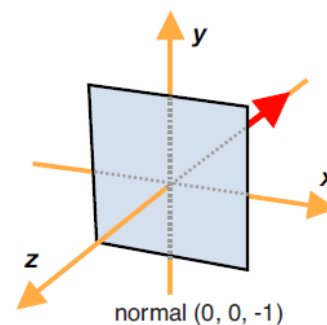
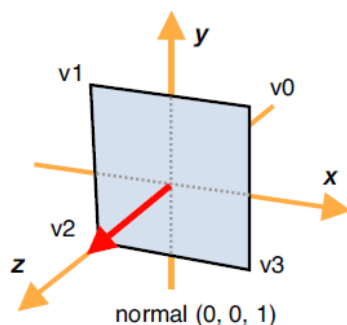
- Para realizar el cálculo de la reflexión difusa, es necesario conocer la dirección de la luz y la orientación de la superficie (para calcular $\cos \theta$)
- La dirección de la luz reflejada es la dirección opuesta a la fuente de luz
- Para determinar la orientación de la superficie, usaremos la **normal** (orientación de una superficie se especifica por la dirección perpendicular a la superficie)



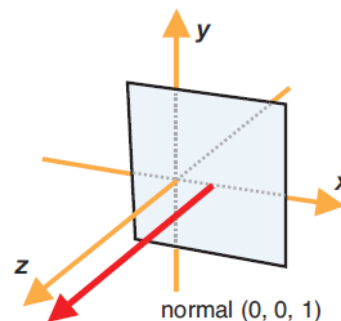
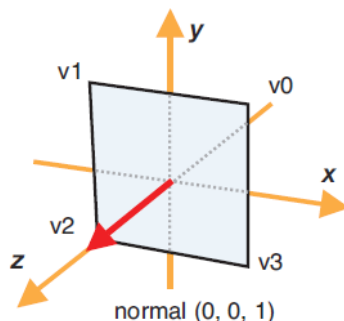
2. Fuente de luz – Reflexión difusa

- Cada superficie tiene dos normales:

- Cara frontal
- Cara posterior



- Si dos caras tienen la misma orientación, tienen las mismas normales



2. Fuente de luz – Reflexión difusa

- El $\cos \theta$ se calcula por medio del producto escalar de la dirección de la luz por la orientación de la superficie:

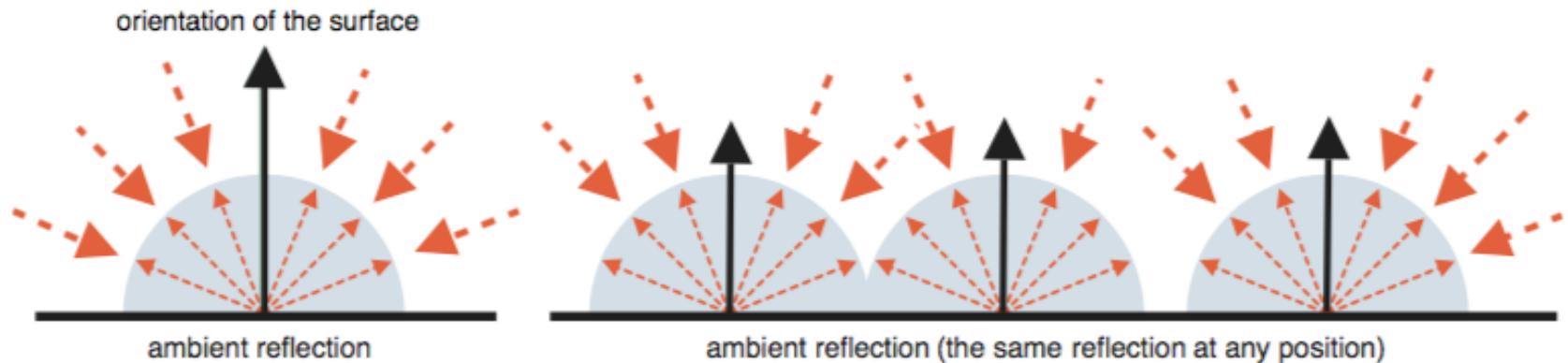
$$\cos \theta = (\text{dirección_luz}) \cdot (\text{orientación_superficie})$$

- De esta forma, la ecuación para el cálculo del color de la superficie reflejada por reflexión difusa:

$$(\text{color_superficie_reflexión_difusa}) = (\text{color_luz}) \times (\text{color_base_superficie}) \times ((\text{dirección_luz}) \cdot (\text{orientación_superficie}))$$

2. Fuente de luz – Reflexión ambiental

- En la reflexión ambiental, la luz es reflejada con el mismo ángulo con el que incide
- Como la luz ambiental ilumina un objeto por igual desde todas las direcciones con la misma intensidad, su brillo es el mismo en cualquier posición



$$(\text{color_superficie_reflexión_ambiental}) = (\text{color_luz}) \times (\text{color_base_superficie})$$

2. Fuente de luz – Reflexión difusa y ambiental

- Cuando ambas reflexiones (ambiental y difusa) están presentes, el color de la superficie se calcula sumando ambas reflexiones:

```
(color_superficie_reflexión_ambiental_y_difusa) =  
    (color_superficie_reflexión_difusa) +  
    (color_superficie_reflexión_ambiental)
```

Índice de contenidos

1. Introducción
2. Tipos de fuente de luz
- 3. Referencia API GLSL ES**
4. Referencia glmMatrix
5. Ejemplo: cubo con luz ambiental
6. Ejemplo: cubo con luz direccional
7. Ejemplo: cubo con luz ambiental y direccional
8. Matriz mágica: inversa y transpuesta
9. Ejemplo: luz direccional y matriz mágica
10. Ejemplo: cubo con luz puntual
11. Ejemplo: cubo con degradado realista

3. Referencia API GLSL ES

Name

`normalize` — calculates the unit vector in the same direction as the original vector

Declaration

```
genType normalize( genType v);  
  
genDType normalize( genDType v);
```

Parameters

`v`

Specifies the vector to normalize.

Description

normalize returns a vector with the same direction as its parameter, `v`, but with length 1.

<https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/normalize.xhtml>

3. Referencia API GLSL ES

Name

`max` — return the greater of two values

Declaration

```
genType max( genType x,  
             genType y);
```

Parameters

x

Specify the first value to compare.

y

Specify the second value to compare.

Description

`max` returns the maximum of the two parameters. It returns *y* if *y* is greater than *x*, otherwise it returns *x*.

<https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/max.xhtml>

3. Referencia API GLSL ES

Name

dot — calculate the dot product of two vectors

Declaration

```
float dot( genType x,
           genType y);

double dot( genDType x,
            genDType y);
```

Parameters

x

Specifies the first of two vectors

y

Specifies the second of two vectors

Description

dot returns the dot product of two vectors, *x* and *y*. i.e., $x[0] \cdot y[0] + x[1] \cdot y[1] + \dots$

<https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/dot.xhtml>

Índice de contenidos

1. Introducción
2. Tipos de fuente de luz
3. Referencia API GLSL ES
- 4. Referencia glmatrix**
5. Ejemplo: cubo con luz ambiental
6. Ejemplo: cubo con luz direccional
7. Ejemplo: cubo con luz ambiental y direccional
8. Matriz mágica: inversa y transpuesta
9. Ejemplo: luz direccional y matriz mágica
10. Ejemplo: cubo con luz puntual
11. Ejemplo: cubo con degradado realista

4. Referencia glmatrix

```
(static) normalize(out, a) → {vec3}
```

Normalize a vec3

Parameters:

Name	Type	Description
out	vec3	the receiving vector
a	vec3	vector to normalize

Source: [vec3.js, line 335](#)

Returns:

out

Type

vec3

<http://glmatrix.net/docs/module-vec3.html>

4. Referencia glmatrix

`(static) invert(out, a) → {mat4}`

Inverts a mat4

Parameters:

Name	Type	Description
out	mat4	the receiving matrix
a	mat4	the source matrix

Source: [mat4.js, line 293](#)

Returns:

out

Type

mat4

`(static) transpose(out, a) → {mat4}`

Transpose the values of a mat4

Parameters:

Name	Type	Description
out	mat4	the receiving matrix
a	mat4	the source matrix

Source: [mat4.js, line 242](#)

Returns:

out

Type

mat4

<http://glmatrix.net/docs/module-mat4.html>

Índice de contenidos

1. Introducción
2. Tipos de fuente de luz
3. Referencia API GLSL ES
4. Referencia glmatrix
- 5. Ejemplo: cubo con luz ambiental**
6. Ejemplo: cubo con luz direccional
7. Ejemplo: cubo con luz ambiental y direccional
8. Matriz mágica: inversa y transpuesta
9. Ejemplo: luz direccional y matriz mágica
10. Ejemplo: cubo con luz puntual
11. Ejemplo: cubo con degradado realista

5. Ejemplo: cubo con luz ambiental

```
<!DOCTYPE html>
<html>
<head>
<title>Lighting: cube lit by ambient light</title>
<script src="https://cdn.jsdelivr.net/npm/gl-matrix@2.8.1/gl-matrix.min.js"></script>
<script id="shaderVs" type="x-shader/x-vertex">
    attribute vec4 a_Position;
    attribute vec3 a_Color;

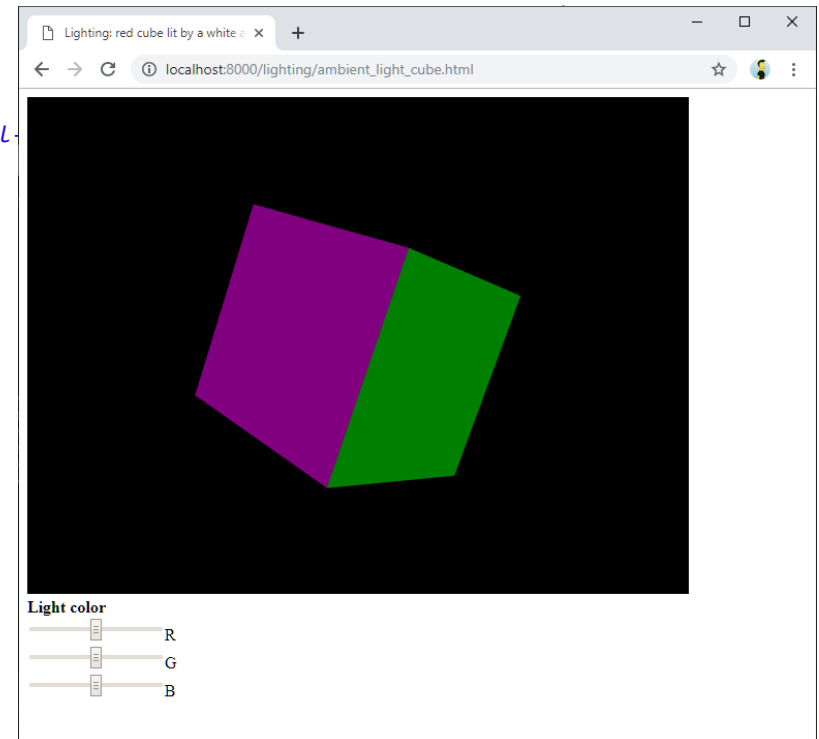
    uniform vec3 u_AmbientLight;
    uniform mat4 u_pMatrix;
    uniform mat4 u_vMatrix;
    uniform mat4 u_mvMatrix;

    varying highp vec4 v_Color;

    void main(void) {
        vec4 vertexPosition = u_mvMatrix * a_Position;
        gl_Position = u_pMatrix * u_vMatrix * vertexPosition;

        vec3 ambient = u_AmbientLight * a_Color;
        v_Color = vec4(ambient, 1.0);
    }
</script>
<script id="shaderFs" type="x-shader/x-fragment">
    varying highp vec4 v_Color;

    void main(void) {
        gl_FragColor = v_Color;
    }
</script>
...
</html>
```



Para el cálculo de la reflexión difusa debido a luz ambiental sólo influye el color de la luz

5. Ejemplo: cubo con luz ambiental

```
<script>
function initProjection(gl, canvas) {
    // Write u_pMatrix
    var ratio = canvas.width / canvas.height;
    var pMatrix = mat4.perspective(mat4.create(), 0.9, ratio, 0.1, 100);
    var pMatrixUniform = gl.getUniformLocation(gl.program, "u_pMatrix");
    gl.uniformMatrix4fv(pMatrixUniform, false, pMatrix);

    // Write u_vMatrix
    var vMatrix = mat4.lookAt(mat4.create(), [ 0, 0, 3 ], [ 0, 0, 0 ], [ 0, 1, 0 ]);
    var vMatrixUniform = gl.getUniformLocation(gl.program, "u_vMatrix");
    gl.uniformMatrix4fv(vMatrixUniform, false, vMatrix);

    // Write u_AmbientLight
    var r = document.getElementById("r").value;
    var g = document.getElementById("g").value;
    var b = document.getElementById("b").value;
    var ambientLight = [ r, g, b ];
    var ambientLightUniform = gl.getUniformLocation(gl.program, "u_AmbientLight");
    gl.uniform3fv(ambientLightUniform, ambientLight);
}
</script>
```

El color de la luz se obtiene desde la interfaz de usuario
y se escribe en la variable del vertex shader
u_AmbientLight

Índice de contenidos

1. Introducción
2. Tipos de fuente de luz
3. Referencia API GLSL ES
4. Referencia glmatrix
5. Ejemplo: cubo con luz ambiental
- 6. Ejemplo: cubo con luz direccional**
7. Ejemplo: cubo con luz ambiental y direccional
8. Matriz mágica: inversa y transpuesta
9. Ejemplo: luz direccional y matriz mágica
10. Ejemplo: cubo con luz puntual
11. Ejemplo: cubo con degradado realista

6. Ejemplo: cubo con luz direccional

```
<!DOCTYPE html>
<html>
<head>
<title>Lighting: cube lit by directional light</title>
<script src="https://cdn.jsdelivr.net/npm/gl-matrix@2.8.1/gl-matrix.min.js"></script>

<script id="shaderVs" type="x-shader/x-vertex">
    attribute vec3 a_Color;
    attribute vec3 a_Normal;
    attribute vec4 a_Position;

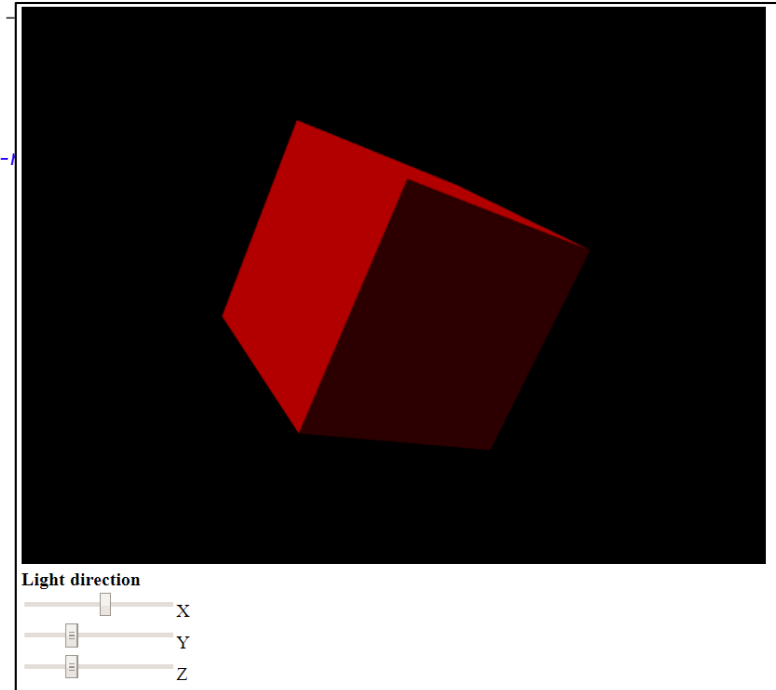
    uniform vec3 u_LightColor;
    uniform vec3 u_LightDirection;
    uniform mat4 u_pMatrix;
    uniform mat4 u_vMatrix;
    uniform mat4 u_mvMatrix;

    varying highp vec4 v_Color;

    void main(void) {
        gl_Position = u_pMatrix * u_vMatrix * u_mvMatrix * a_Position;

        vec3 normal = normalize(a_Normal);
        float nDotL = max(dot(u_LightDirection, normal), 0.0);

        vec3 diffuse = u_LightColor * a_Color.rgb * nDotL;
        v_Color = vec4(diffuse, 1.0);
    }
</script>
...
</html>
```



Para el cálculo de la reflexión difusa debido a luz direccional influye el color y dirección de la luz

6. Ejemplo: cubo con luz direccional

```
<!DOCTYPE html>
<html>
<head>
<title>Lighting: cube lit by directional light</title>
<script src="https://cdnjs.cloudflare.com/ajax/libs/gl-matrix/2.8.1/gl-matrix-min.js"></script>

<script id="shaderVs" type="x-shader/x-vertex">
    attribute vec3 a_Color;
    attribute vec3 a_Normal;
    attribute vec4 a_Position;

    uniform vec3 u_LightColor;
    uniform vec3 u_LightDirection;
    uniform mat4 u_pMatrix;
    uniform mat4 u_vMatrix;
    uniform mat4 u_mvMatrix;

    varying highp vec4 v_Color;

    void main(void) {
        gl_Position = u_pMatrix * u_vMatrix * u_mvMatrix * a_Position;

        vec3 normal = normalize(a_Normal);
        float nDotL = max(dot(u_LightDirection, normal), 0.0);

        vec3 diffuse = u_LightColor * a_Color.rgb * nDotL;
        v_Color = vec4(diffuse, 1.0);
    }
</script>
</html>
```

Se normaliza cada vector, aunque no sería necesario ya que utilizamos longitud 1, pero es buena práctica

Solo `u_LightDirection` se normaliza en el código JavaScript para evitar la sobrecarga de normalizarlo por cada vértice

Valor máximo entre 0 y el producto escalar del vector normal y la dirección de la luz

Para el cálculo de la reflexión difusa debido a luz direccional influye el color de la superficie y el color y dirección de la luz

6. Ejemplo: cubo con luz direccional

```
<!DOCTYPE html>
<html>
<head>
<title>Lighting: cube lit by directional light</title>
<script src="https://cdnjs.cloudflare.com/ajax/libs/gl-matrix/2.8.1/gl-matrix-min.js"></script>

<script id="shaderVs" type="x-shader/x-vertex">
  attribute vec3 a_Color;
  attribute vec3 a_Normal;
  attribute vec4 a_Position;

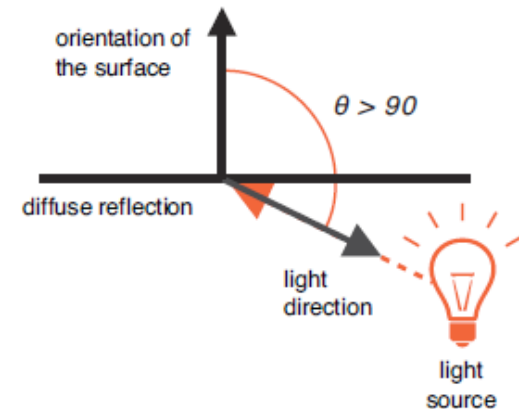
  uniform vec3 u_LightColor;
  uniform vec3 u_LightDirection;
  uniform mat4 u_pMatrix;
  uniform mat4 u_vMatrix;
  uniform mat4 u_mvMatrix;

  varying highp vec4 v_Color;

  void main(void) {
    gl_Position = u_pMatrix * u_vMatrix * u_mvMatrix * a_Position;

    vec3 normal = normalize(a_Normal);
    float nDotL = max(dot(u_LightDirection, normal), 0.0);

    vec3 diffuse = u_LightColor * a_Color.rgb * nDotL;
    v_Color = vec4(diffuse, 1.0);
  }
</script>
...
</html>
```



El producto escalar negativo significa que θ en $\cos \theta$ es más de 90 grados, es decir, la luz golpea la superficie en su cara posterior.

6. Ejemplo: cubo con luz direccional

```
<script>
function initVertexShader(gl) {
    var vertices = new Float32Array([ // Coordinates ]);
    var colors = new Float32Array([ // Colors   ]);

    var normals = new Float32Array([ // Normal
        0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, // v0-v1-v2-v3 front
        1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, // v0-v3-v4-v5 right
        0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, // v0-v5-v6-v1 up
        -1.0, 0.0, 0.0, -1.0, 0.0, 0.0, -1.0, 0.0, 0.0, -1.0, 0.0, 0.0, // v1-v6-v7-v2 left
        0.0, -1.0, 0.0, 0.0, -1.0, 0.0, 0.0, -1.0, 0.0, 0.0, -1.0, 0.0, // v7-v4-v3-v2 down
        0.0, 0.0, -1.0, 0.0, 0.0, -1.0, 0.0, 0.0, -1.0, 0.0, 0.0, -1.0 // v4-v7-v6-v5 back
    ]);

    // ...

    var vertexNormalAttribute = gl.getAttribLocation(gl.program, "a_Normal");
    gl.enableVertexAttribArray(vertexNormalAttribute);
    gl.vertexAttribPointer(vertexNormalAttribute, 3, gl.FLOAT, false, 0, 0);

    var indices = new Uint8Array([ // indices  ]);

    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, gl.createBuffer());
    gl.bufferData(gl.ELEMENT_ARRAY_BUFFER, indices, gl.STATIC_DRAW);
}
</script>
```

Es necesario especificar los
vectores normales cada uno
de los vértices

6. Ejemplo: cubo con luz direccional

```
<script>
function initProjection(gl, canvas) {
    // Write u_pMatrix
    var ratio = canvas.width / canvas.height;
    var pMatrix = mat4.perspective(mat4.create(), 0.9, ratio, 0.1, 100);
    var pMatrixUniform = gl.getUniformLocation(gl.program, "u_pMatrix");
    gl.uniformMatrix4fv(pMatrixUniform, false, pMatrix);

    // Write u_vMatrix
    var vMatrix = mat4.lookAt(mat4.create(), [ 0, 0, 3 ], [ 0, 0, 0 ], [ 0, 1, 0 ]);
    var vMatrixUniform = gl.getUniformLocation(gl.program, "u_vMatrix");
    gl.uniformMatrix4fv(vMatrixUniform, false, vMatrix);

    // Write u_LightColor
    var lightColor = [ 1.0, 1.0, 1.0 ]; // white
    var lightColorUniform = gl.getUniformLocation(gl.program, "u_LightColor");
    gl.uniform3fv(lightColorUniform, lightColor);

    // Write u_LightDirection
    var x = document.getElementById("x").value;
    var y = document.getElementById("y").value;
    var z = document.getElementById("z").value;
    var lightDirection = [ x, y, z ];
    vec3.normalize(lightDirection, lightDirection);
    var lightDirectionUniform = gl.getUniformLocation(gl.program, "u_LightDirection");
    gl.uniform3fv(lightDirectionUniform, lightDirection);
}
</script>
```

Estos valores indican **hacia dónde es reflejada la luz**, no de donde proviene

La dirección de la luz se obtiene desde la interfaz de usuario y se escribe en la variable del vertex shader **u_LightDirection**

6. Ejemplo: cubo con luz direccional

- Pero al rotar el cubo, vemos que existe un problema:
 - La cara opuesta a la fuente de luz es casi negra y no es claramente visible
- Para solucionarlo, se puede añadir luz ambiental



Índice de contenidos

1. Introducción
2. Tipos de fuente de luz
3. Referencia API GLSL ES
4. Referencia glmatrix
5. Ejemplo: cubo con luz ambiental
6. Ejemplo: cubo con luz direccional
- 7. Ejemplo: cubo con luz ambiental y direccional**
8. Matriz mágica: inversa y transpuesta
9. Ejemplo: luz direccional y matriz mágica
10. Ejemplo: cubo con luz puntual
11. Ejemplo: cubo con degradado realista

7. Ejemplo: cubo con luz ambiental y direccional

```
<!DOCTYPE html>
<html>
<head>
<title>Lighting: cube lit by point light</title>
<script src="https://cdn.jsdelivr.net/npm/gl-matrix@2.8.1/gl-matrix.min.js"></script>
<script id="shaderVs" type="x-shader/x-vertex">
    attribute vec3 a_Color;
    attribute vec3 a_Normal;
    attribute vec4 a_Position;

    uniform vec3 u_LightColor;
    uniform vec3 u_LightDirection;
    uniform vec3 u_AmbientLight;

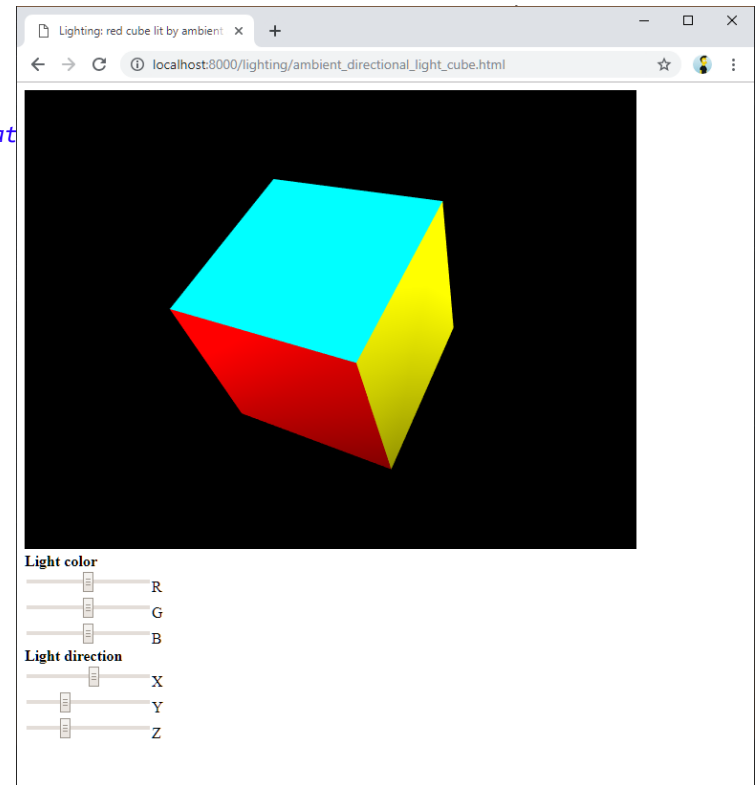
    uniform mat4 u_pMatrix;
    uniform mat4 u_vMatrix;
    uniform mat4 u_mvMatrix;

    varying highp vec4 v_Color;

    void main(void) {
        gl_Position = u_pMatrix * u_vMatrix * u_mvMatrix * a_Position;

        vec3 normal = normalize(a_Normal);
        float nDotL = max(dot(u_LightDirection, normal), 0.0);

        vec3 diffuse = u_LightColor * a_Color.rgb * nDotL;
        vec3 ambient = u_AmbientLight * a_Color.rgb;
        v_Color = vec4(diffuse + ambient, 1.0);
    }
</script>
...
</html>
```



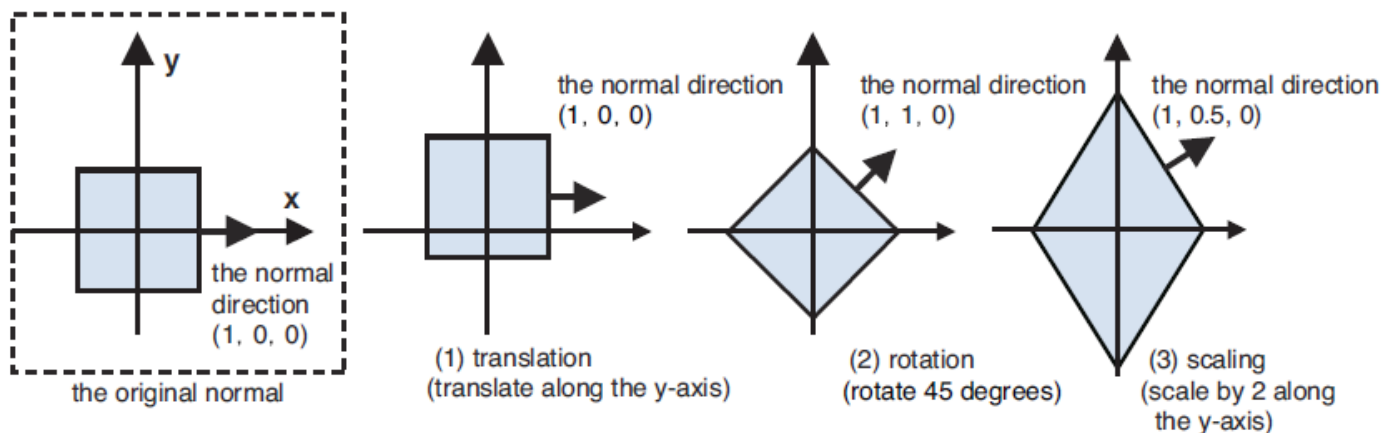
La reflexión ahora es la suma de
reflexión difusa y ambiental

Índice de contenidos

1. Introducción
2. Tipos de fuente de luz
3. Referencia API GLSL ES
4. Referencia glmatrix
5. Ejemplo: cubo con luz ambiental
6. Ejemplo: cubo con luz direccional
7. Ejemplo: cubo con luz ambiental y direccional
- 8. Matriz mágica: inversa y transpuesta**
9. Ejemplo: luz direccional y matriz mágica
10. Ejemplo: cubo con luz puntual
11. Ejemplo: cubo con degradado realista

8. Matriz mágica: inversa y transpuesta

- Hay que tener en cuenta, que como vimos en temas anteriores, un movimiento de los objetos, además de modificar las posiciones de los vértices, modifican los vectores normales
- Tras varias modificaciones, estos vectores no son siempre fáciles de calcular



8. Matriz mágica: inversa y transpuesta

- Para calcular la dirección que tienen los vectores tras varios cambios, se debe:

- Invertir la matriz original del modelo (mvMatrix)

```
(static) invert(out, a) → {mat4}
```

- Obtener la matriz traspuesta de matriz inversa

```
(static) transpose(out, a) → {mat4}
```

Índice de contenidos

1. Introducción
2. Tipos de fuente de luz
3. Referencia API GLSL ES
4. Referencia glmatrix
5. Ejemplo: cubo con luz ambiental
6. Ejemplo: cubo con luz direccional
7. Ejemplo: cubo con luz ambiental y direccional
8. Matriz mágica: inversa y transpuesta
- 9. Ejemplo: luz direccional y matriz mágica**
10. Ejemplo: cubo con luz puntual
11. Ejemplo: cubo con degradado realista

9. Ejemplo: luz direccional y matriz mágica

```
<!DOCTYPE html><html><head>
<title>Lighting: cube lit by directional light with magic matrix</title>
<script src="https://cdnjs.cloudflare.com/ajax/libs/gl-matrix/2.8.1/gl-matrix-min.js">
<script id="shaderVs" type="x-shader/x-vertex">
    attribute vec3 a_Color;
    attribute vec3 a_Normal;
    attribute vec4 a_Position;

    uniform vec3 u_LightColor;
    uniform vec3 u_LightDirection;
    uniform vec3 u_AmbientLight;
    uniform mat4 u_pMatrix;
    uniform mat4 u_vMatrix;
    uniform mat4 u_mvMatrix;
    uniform mat4 u_NormalMatrix;

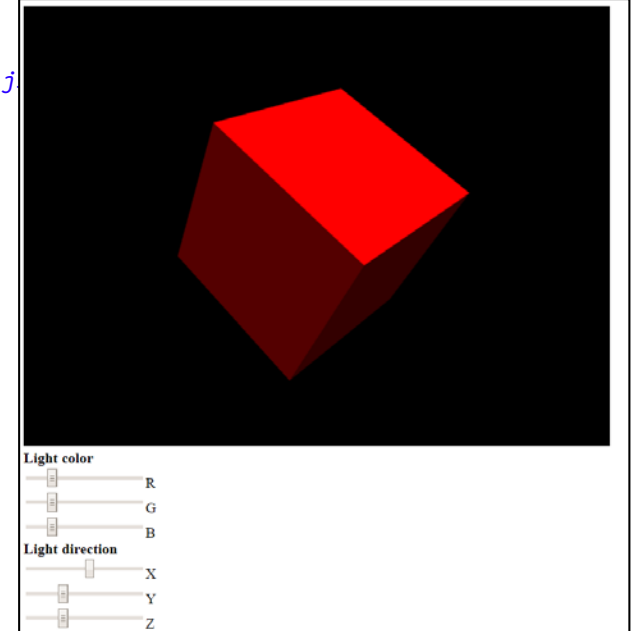
    varying highp vec4 v_Color;

    void main(void) {
        gl_Position = u_pMatrix * u_vMatrix * u_mvMatrix * a_Position;

        vec4 transformedNormal = u_NormalMatrix * vec4(a_Normal, 1.0);
        float nDotL = max(dot(u_LightDirection, normalize(transformedNormal.xyz)), 0.0);

        vec3 diffuse = u_LightColor * a_Color.rgb * nDotL;
        vec3 ambient = u_AmbientLight * a_Color.rgb;
        v_Color = vec4(diffuse + ambient, 1.0);
    }
</script>
...
</html>
```

Nueva matriz
u_NormalMatrix
que será la inversa
y transpuesta de
u_mvMatrix



La nueva normal es la multiplicación
de ambas matrices

9. Ejemplo: luz direccional y matriz mágica

```
<script>
function drawScene(gl, canvas) {
    // Clear
    gl.clear(gl.COLOR_BUFFER_BIT);

    // Init projection
    initProjection(gl, canvas);

    // Rotate
    var mvMatrix = mat4.fromRotation(mat4.create(), count, [ 0.0, 1.0, 0.5 ]);
    var uMvMatrix = gl.getUniformLocation(gl.program, "u_mvMatrix");
    gl.uniformMatrix4fv(uMvMatrix, false, mvMatrix);

    // Get the normal direction using the model Matrix
    var normalMatrixInvert = mat4.invert(mat4.create(), mvMatrix);
    var normalMatrixInvertTranspose = mat4.transpose(mat4.create(), normalMatrixInvert);
    var normalUniformMatrix = gl.getUniformLocation(gl.program, "u_NormalMatrix");
    gl.uniformMatrix4fv(normalUniformMatrix, false, normalMatrixInvertTranspose);

    // Draw
    gl.drawElements(gl.TRIANGLES, 6 * 2 * 3, gl.UNSIGNED_BYTE, 0);

    // Call drawScene again in the next browser repaint
    count += 0.01;
    requestAnimationFrame(function() {
        drawScene(gl, canvas)
    });
}
</script>
```

Para el cálculo de la matriz mágica:

1. se invierte la original
2. se realiza su traspuesta

Índice de contenidos

1. Introducción
2. Tipos de fuente de luz
3. Referencia API GLSL ES
4. Referencia glmatrix
5. Ejemplo: cubo con luz ambiental
6. Ejemplo: cubo con luz direccional
7. Ejemplo: cubo con luz ambiental y direccional
8. Matriz mágica: inversa y transpuesta
9. Ejemplo: luz direccional y matriz mágica
- 10. Ejemplo: cubo con luz puntual**
11. Ejemplo: cubo con degradado realista

10. Ejemplo: cubo con luz puntual

```
<!DOCTYPE html><html><head>
<title>Lighting: cube lit by point light</title>
<script src="https://cdnjs.cloudflare.com/ajax/libs/gl-matrix/2.8.1/gl-matrix-min.js"></script>
<script id="shaderVs" type="x-shader/x-vertex">
    attribute vec3 a_Color;
    attribute vec3 a_Normal;
    attribute vec4 a_Position;

    uniform vec3 u_LightColor;
    uniform vec3 u_LightPosition;
    uniform vec3 u_AmbientLight;

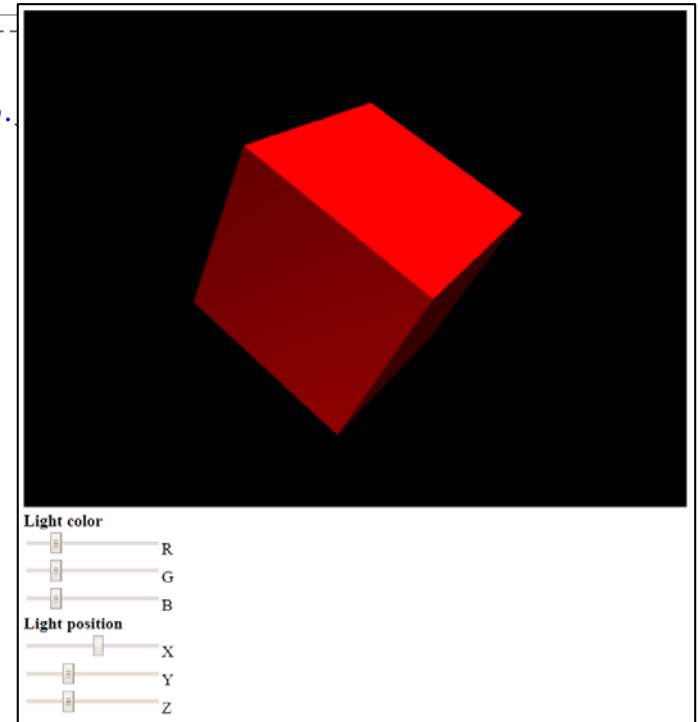
    uniform mat4 u_pMatrix;
    uniform mat4 u_vMatrix;
    uniform mat4 u_mvMatrix;
    uniform mat4 u_NormalMatrix;

    varying highp vec4 v_Color;

    void main(void) {
        gl_Position = u_pMatrix * u_vMatrix * u_mvMatrix * a_Position;

        vec3 lightDirection = normalize(u_lightPosition - vec3(gl_Position));
        vec4 transformedNormal = u_NormalMatrix * vec4(a_Normal, 1.0);
        float nDotL = max(dot(lightDirection, normalize(transformedNormal)), 0.0);

        vec3 diffuse = u_LightColor * a_Color.rgb * nDotL;
        vec3 ambient = u_AmbientLight * a_Color.rgb;
        v_Color = vec4(diffuse + ambient, 1.0);
    }
</script> ... </html>
```



Ahora se calcula la dirección de la luz en función de dónde se encuentra la fuente de emisión y los vértices del objeto

10. Ejemplo: cubo con luz puntual

```
<!DOCTYPE html><html><head>
<title>Lighting: cube lit by point light</title>
<script src="https://cdn.jsdelivr.net/npm/gl-matrix@2.8.1/gl-matrix-min.js"></script>
<script id="shaderVs" type="x-shader/x-vertex">
    attribute vec3 a_Color;
    attribute vec3 a_Normal;
    attribute vec4 a_Position;

    uniform vec3 u_LightColor;
    uniform vec3 u_LightPosition;
    uniform vec3 u_AmbientLight;

    uniform mat4 u_pMatrix;
    uniform mat4 u_vMatrix;
    uniform mat4 u_mvMatrix;
    uniform mat4 u_NormalMatrix;

    varying highp vec4 v_Color;

    void main(void) {
        gl_Position = u_pMatrix * u_vMatrix * u_mvMatrix * a_Position;

        vec3 lightDirection = normalize(u_lightPosition - vec3(gl_Position));
        vec4 transformedNormal = u_NormalMatrix * vec4(a_Normal, 1.0);
        float nDotL = max(dot(lightDirection, normalize(transformedNormal.xyz)), 0.0);

        vec3 diffuse = u_LightColor * a_Color.rgb * nDotL;
        vec3 ambient = u_AmbientLight * a_Color.rgb;
        v_Color = vec4(diffuse + ambient, 1.0);
    }
</script> ... </html>
```

Cálculo de la dirección de la luz

Valor máximo entre 0 y el producto escalar del vector normal transformado y la dirección de la luz

Para el cálculo de la reflexión difusa debido a luz direccional influye el color de la superficie y el color y dirección de la luz

7. Ejemplo: cubo con luz puntual

```
<script>
function initProjection(gl, canvas) {
    // Write u_pMatrix
    // ...
    // Write u_vMatrix
    // ...
    // Write u_LightColor
    // ...
    // Write u_LightPosition
    var x = document.getElementById("x").value;
    var y = document.getElementById("y").value;
    var z = document.getElementById("z").value;
    var lightPosition = [ x, y, z ];
    var lightPositionUniform = gl.getUniformLocation(gl.program, "u_LightPosition");
    gl.uniform3fv(lightPositionUniform, lightPosition);

    // Write u_AmbientLight
    var r = document.getElementById("r").value;
    var g = document.getElementById("g").value;
    var b = document.getElementById("b").value;
    var ambientLight = [ r, g, b ];
    var ambientLightUniform = gl.getUniformLocation(gl.program, "u_AmbientLight");
    gl.uniform3fv(ambientLightUniform, ambientLight);
}
</script>
```

La posición de la luz se obtiene desde la interfaz de usuario y se escribe en la variable del vertex shader **u_LightPosition**

El color de la luz ambiente se obtiene desde la interfaz de usuario y se escribe en la variable del vertex shader **u_AmbientLight**

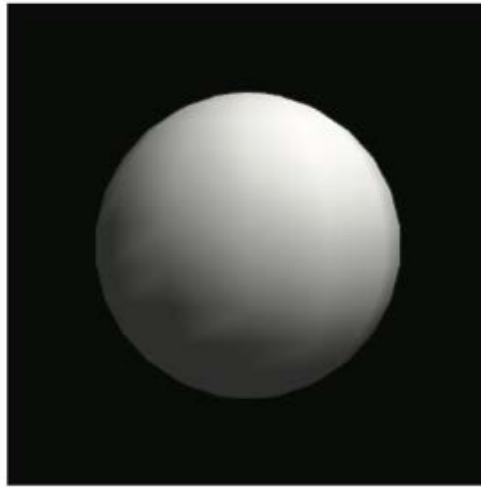
Índice de contenidos

1. Introducción
2. Tipos de fuente de luz
3. Referencia API GLSL ES
4. Referencia glmMatrix
5. Ejemplo: cubo con luz ambiental
6. Ejemplo: cubo con luz direccional
7. Ejemplo: cubo con luz ambiental y direccional
8. Matriz mágica: inversa y transpuesta
9. Ejemplo: luz direccional y matriz mágica
10. Ejemplo: cubo con luz puntual
- 11. Ejemplo: cubo con degradado realista**

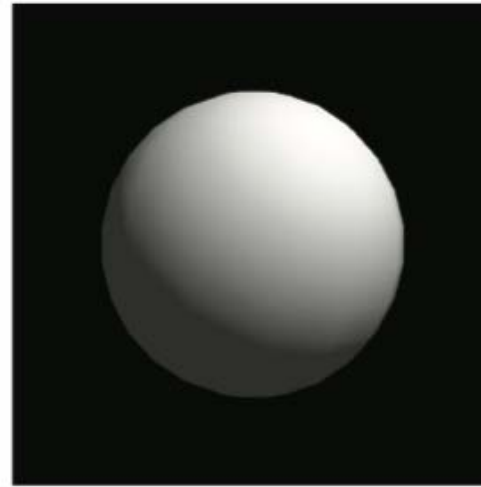
11. Ejemplo: cubo con degradado realista

- En los ejemplos anteriores hemos calculado el color de la reflexión (difusa, ambiental) **por vértice**
- Este color se pasa del vertex al fragment shader mediante una variable de tipo `varying` (los colores de los fragmentos se calcula por interpolación de esos valores)
- Se puede conseguir un degradado (shading) más realista si el color de reflexión se calcula **por fragmento**, esto es, en el fragment shader
- Es un proceso más costoso a nivel de procesamiento en la GPU, ya que normalmente hay más fragmentos que vértices en una escena

11. Ejemplo: cubo con degradado realista



Cálculo en el
vertex shader



Cálculo en el
fragment shader

```
<script id="shaderVs" type="x-shader/x-vertex">
    attribute vec3 a_Color;
    attribute vec3 a_Normal;
    attribute vec4 a_Position;

    uniform vec3 u_lightPosition;

    uniform mat4 u_pMatrix;
    uniform mat4 u_vMatrix;
    uniform mat4 u_mvMatrix;
    uniform mat4 u_NormalMatrix;

    varying highp vec4 v_Color;
    varying highp vec4 v_Normal;
    varying highp vec3 v_lightDirection;

    void main(void) {
        gl_Position = u_pMatrix * u_vMatrix * u_mvMatrix * a_Position;

        v_lightDirection = normalize(u_lightPosition - vec3(gl_Position));
        v_Normal = u_NormalMatrix * vec4(a_Normal, 1.0);

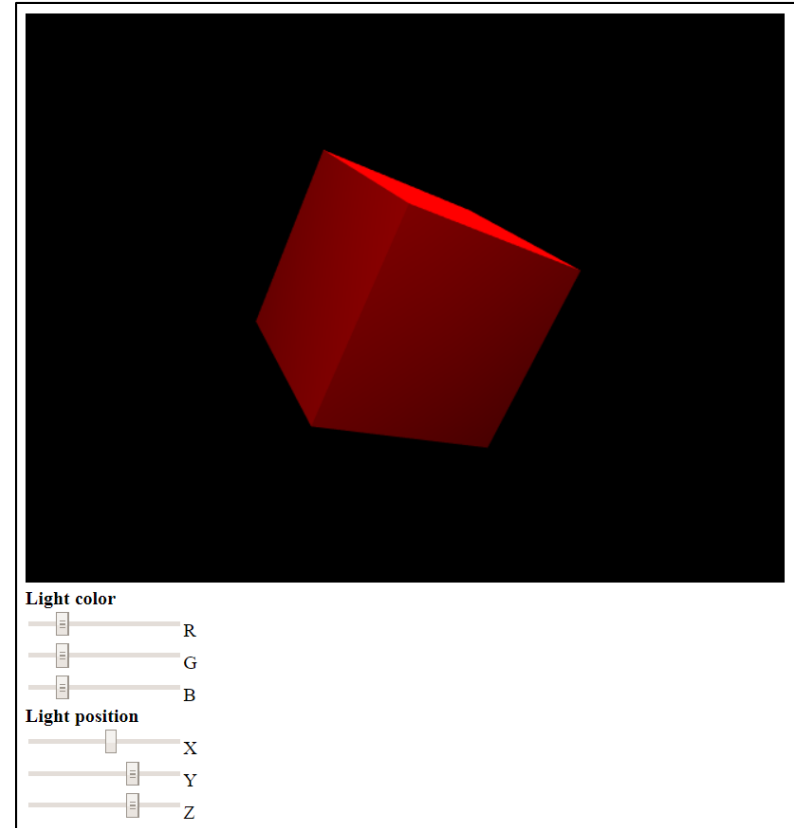
        v_Color = vec4(a_Color, 1.0);
    }
</script>
<script id="shaderFs" type="x-shader/x-fragment">
    precision mediump float;

    uniform vec3 u_AmbientLight;
    uniform vec3 u_LightColor;

    varying highp vec4 v_Color;
    varying highp vec4 v_Normal;
    varying highp vec3 v_lightDirection;

    void main(void) {
        float nDotL = max(dot(v_lightDirection, normalize(v_Normal.xyz)), 0.0);

        vec3 diffuse = u_LightColor * v_Color.rgb * nDotL;
        vec3 ambient = u_AmbientLight * v_Color.rgb;
        gl_FragColor = vec4(diffuse + ambient, v_Color.a);
    }
</script>
...
```



Movemos el cálculo del color de la reflexión difusa al fragment shader