# Tarea10

*Phong Lighting*

**Instrucciones**:

- Leer:

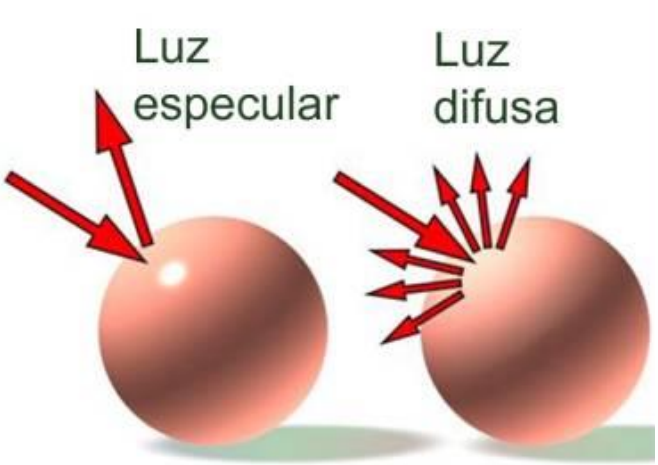
  https://docs.gameloft.org/3d-training/#Lighting

- Poner en escena:
  - Una luz direccional
  - Dos luces en movimiento

## Clasificación de los componentes de las luces

Desde el punto de vista de la física, la luz reflejada por un objeto puede ser clasificado en:

- ***Difusa***: Cuando la luz golpea un objeto, el componente difuso de la luz se dirige en todas direcciones (por las asperezas microscópicas del objeto). Esta componente no es dependiente del observador (puede ser vista sin importar la posición del observador).
- ***Especular***: Solo aplica para objetos brillosos/con alta reflexión, cuando la luz golpea un objeto la componente especular se refleja claramente en una dirección. Esta componente es dependiente del observador, si el observador no está lo suficientemente cerca dela dirección de reflejo, no podrá ver el especular.



En el modelo Phong las luces se definen un poco diferentes:

- **Ambiental:** Es solo una aproximación de la luz indirecta (la luz que se refleja de otros objetos en la escena, por eso se nos permite ver hasta los objetos que están en áreas con sombra) y es dada en el modelo como constante (por ejemplo, un ambiente de [1.0, 1.0, 1.0] significa una luz ambiental blanca). Tiene un comportamiento similar a la luz difusa, por lo que necesitaremos agregar estos componentes usando weights(w para la componente ambiente y **1-w** para la componente difusa; **usualmente w<1-w**) para evitar que nuestro objeto se torne blanco.
- **Difusa**
- **Especular**

  La luz ambiental y difusa se multiplicaran con la textura del objeto para obtener el color final del objeto, pero la especular no se multiplicara.

# Clasificación de los objetos en el mundo real

Los objetos en el mundo real pueden ser clasificados en:

- Muy reflejantes:
  - **Objetos opacos** (ejemplo: un espejo). Para estos objetos, podemos omitir la componente difusa por el hecho de que tiene una muy baja influencia (nosotros no estamos interesados en la textura del espejo, de hecho estamos interesados en el reflejo del ambiente en ese espejo). Por lo tanto, el color final para estos objetos será dado por el especular que es calculado por la reflexión del cielo (recuerda la reflexión de tu shader sky, en ese lugar usaste un espejo en la esfera).
  - **Objetos transparentes** (agua, vidrio, etc.). Para estos objetos combinaremos el especular calculado como una reflexión del cielo con el cual podemos ver atreves del objeto (refracción). Por ejemplo, cuando vamos a renderizar el agua (en una práctica más adelante) combinaremos:
    - El fondo del agua (superficie plana, afectada por la refracción), iluminado usando luz ambiental y difusa con el modelo Phong.
    - La superficie (muy reflejante), iluminada solamente con el especular calculado como la reflexión del cielo.

- **Objetos reflejantes**:
    - **Objetos con mucho reflejo** ( como el metal pintado de los automóviles) se iluminara por combinación (usando weights):
        - Luz difusa y ambiental (como el modelo Phong)
        - Luz especular es calculada como la reflexión del cielo.
    - **Objetos con menos reflejo** (como una pelota plastica) se iluminara completamente con el modelo Phong:
        - Luz difusa y ambiental (como en el modelo Phong)
        - Luz especular es calculada con Phong (**Nota:** El modelo Phong solo da una aproximación al especular, la aproximación funciona bien para los objetos menos reflejantes, para los objetos más reflejantes esta aproximación no es suficiente)
- **Objetos opacos (como el terreno)**:

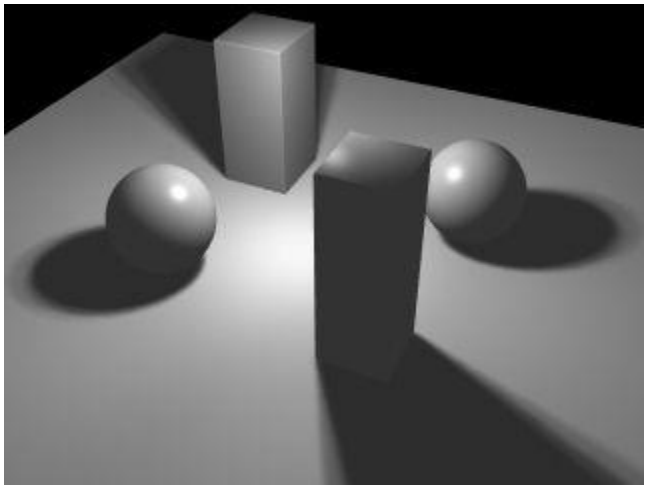
  Será iluminado solamente por la luz ambiental y la difusa (calculado por el modelo Phong)

# Tipos de fuentes de luz

- **Luz direccional**: Esta fuente de luz es muy muy distante, como el Sol. Esta luz se caracterizara solamente por su dirección (piensa en los rayos de luz del Sol, que son paralelos entre ellos) y esta luz no tendrá posición.

  **En SM.txt, tú tendrás que describir el tipo fuente de luz**:
    - En type:DIRECTIONAL
    - En POSITION/DIRECTION → Lo que describas en tu con los valores (x,y,z) será la dirección.
- **Point light** (como por ejemplo una lámpara) y la **spot light** (como una lámpara de mano). Para ambas especificaremos la posición.

**Point light**



**Spot Light**

# Fórmulas para Modelo Phong

**ObjectFinalColor** =vec4((( (AmbientColor * AmbientWeight + **DiffuseTotal** * (1 – AmbientWeight)) * texture + **SpecularTotal** ).xyz, texture.w);

**DiffuseTotal** = sum of (DiffuseColor * DiffuseIntensity for each light source)

**SpecularTotal** = sum of (SpecularColor * SpecularIntensity for each light source)

***AmbientColor, AmbientWeight – the values which you put in SM.txt. Ambient is unique per scene, a light source has only diffuse and specular components, not an ambient component.***

***DiffuseColor, SpecularColor – the values you put in SM.txt for that light source color***

**DiffuseIntensity** = max(dot(LightDirection, normal of the fragment), 0.0)

**SpecularIntensity** :

-> first you need to calculate the reflectDirection:

**reflectDirection** = reflect(LightDirection, normal of the fragment)

-> using the above calculation:

**SpecularIntensity** = pow(max(dot(reflectDirection , normal of the fragment ), 0.0), specularPower)

***LightDirection must be established so that to enter in the object's fragment (negative value for y component). The origin [0,0,0] of the light vector will be in that fragment. So:***

-> for directional light, just establish in SM. txt proper values txt for that light source

-> for point light (the same for spot light), you need to calculate the direction as:

**LightDirection** = posW – lightPos (i.e position of the fragment in world coordinates minus light source position; light source position is always expressed in world coordinates from the beginning, so in SM.txt, as you had for camera)

luis.martinez@gameloft.com

# Guía:

- **Cada objeto tendrá de cero a varias luces**
- **Las luces no son por escena, se aplicara objeto por objeto, será un atributo más del objeto.**
- **Para las luces no se necesita hacer una nueva clase para dibujarlas, las luces las generamos por medio del nuevo shader de iluminación que harás.**

- Tu SM quedaría algo parecido a:

#Objects: 2
ID 0
MODEL 0
NUM_TEXTURES 1
TEXTURE 0
NUM_CUBE_TEXTURES 0
SHADER 5
**NUM_LIGTHS 2**
**LIGTH 0**
**LIGTH 1**
NUM_SPECIFIC_PARAMETER 0
POSITION 0.0, 0.0, 0.0
ROTATION 0.0, 0.0, 0.0
SCALE 1.0, 1.0, 1.0
TILING_FACTOR 0.0

ID 1
MODEL 2
NUM_TEXTURES 4
TEXTURE 2
TEXTURE 3
TEXTURE 4
TEXTURE 5
NUM_CUBE_TEXTURES 0
SHADER 1
**NUM_LIGTHS 0**
POSITION 0.0, 0.0, 0.0
ROTATION 0.0, 0.0, 0.0
SCALE 1.0, 1.0, 1.0
TILING_FACTOR 24.0
.
.

.

**#LIGHTS: 3**
**AMBIENTCOLOR 1.0, 1.0, 1.0**
**AMBIENTWEIGTH 0.8**

**ID 0**
**POS/DIR 4.0, -100.0, 0.0**
**TYPE DIRECTIONAL**
**COLOR 1.0, 0.0, 0.0**

**ID 1**
**POS/DIR 0.0, 2.0, 0.0**
**TYPE POINT**
**COLOR 0.0, 0.0, 1.0**
**MOVING CIRCLE**
**RADIUS 100.0**
**SPEED 25.0**

**ID 2**
**POS/DIR 0.0, 2.0, 0.0**
**TYPE POINT**
**COLOR 0.0, 1.0, 0.0**
**MOVING CIRCLE**
**RADIUS 5.0**
**SPEED 40.0**

- Por lo tanto necesitaras guardar la información de tus luces que abra disponibles para tus objetos, al hacer una estructura puedes hacer un array de tipo LIGHT con toda la cantidad de luces disponibles:

```
struct  LIGHT
      {
            int         ID;
            int         light_type;
            int         move_type;
            float radius;
            float speed;
            Vector3 pos_dir;
            Vector3 color;
      };
```

- En tu clase Object tendrás que definir los atributos de iluminación, para mantener los atributos de determinada estructura en el Array de luces:

```
Vector3 * light_color;
Vector3 * light_PosDir;
int * light_Type;
int * light_move_type;
float * light_radius;
float * light_speed;
```

*Usamos punteros debido a que haremos arrays de light_color, lightPosDir, etc. Ya que cada objeto tiene de 0 a muchas luces.

- La información de las luces será enviada al shader desde el Draw () de cada objeto como se ha estado haciendo anteriormente.

- En tu Vertex Shader necesitaremos calcular lo siguiente:

gl_Position = WVP * posL;
v_normW //mandar la normal en world space como varying al fragment shader
v_posW //mandar la posición en World Space
u_uv //Mandar las uv para pintar la textura

- En tu fragment shader usaras las uniforms de la siguiente manera:

```
uniform vec3  u_ambient_color;
uniform float u_ambient_weight;
uniform float u_specular_power;

uniform int  u_numLights;
uniform int  u_light_type[10];
uniform vec3 u_pos_dir[10];
uniform vec3 u_color[10];
```

Para tomar la localización en el Shader.cpp se hace igual como lo hacíamos anteriormente:

```
sharderInfo.uLight_color=glGetUniformLocation(sharderInfo.program,
"u_color");
```

Para mandar la informacion desde C para los arrays usaremos algo como:

```
if(shaderObject->getShaderInfo().uLight_color != -1)
      glUniform3fv( shaderObject->getShaderInfo().uLight_color,
object_info.num_lights, &light_color[0].x);
```

*light_color es attributo de la clase Object, e indicamos el numero de Vector3 que inviaremos

- Para calcular la iluminación en el Fragment Shader lo ideal sería hacer funciones para calcular si es posición o dirección, la difusa y el especular, asi poder llamar estas funciones en la función main() y hacer iteración con un ciclo for(i=0; i<`u_numLights` ; i++) para calcular la acumulación de la difusa y del especular

```
vec3 Normal = normalize(v_normW);
vec3 totalDifusse = vec3(0.0,0.0,0.0);
vec3 totalSpecular = vec3(0.0,0.0,0.0);
vec3 lightDirection[10];

for(int i = 0; i<u_numLights; i++)
{
	lightDirection[i] = getlightDirection(i);
	totalDifusse+=diffuseCI(Normal, lightDirection[i], i);
	totalSpecular+=specular(lightDirection[i], Normal, i);
}

vec4 ObjColor = texture2D(u_s_texture0, v_uv);
gl_FragColor = vec4(((u_ambient_color * u_ambient_weight +
                                        totalDifusse  *
                                            (1.0 - u_ambient_weight)) *
ObjColor.xyz +
                                        totalSpecular ).xyz, ObjColor.w);
```

- En la función getLightDirection retornaremos la dirección o posición ya normalizada, para saber si es posición o dirección usa el array `u_light_type, si fuera 0 es direccional y si es 1 es de posicion por ejemplo.`

    - Si es dirección solo retórnanos la dirección ya normalizada (se puede mandar normalizada desde C)

    - Si es posición tenemos hacer la siguiente operación:
      `= normalize(v_posW - u_pos_dir[pos]);`

- Para calcular la difusa y especular sigue las formulas explicadas anteriormente en este documento (hay que poner un poco de reto ☺)

**Resultado**: