TP OpenGL – Illumination

Malek.Bengougam@gmail.com

Partie 1 – Ombrage et Illumination

Dans ce TP nous allons étudier l'équation d'illumination de Phong qui reste encore un standard de l'infographie.
Ce modèle est dit "phénoménologique" ou empirique, c'est à dire qu'il est non conforme aux modèles physiques mais émule / approxime le comportement des lumières et matériaux.

Ce modèle sépare l'illumination en une somme de quatre intensités chromatiques (des couleurs RGB) :

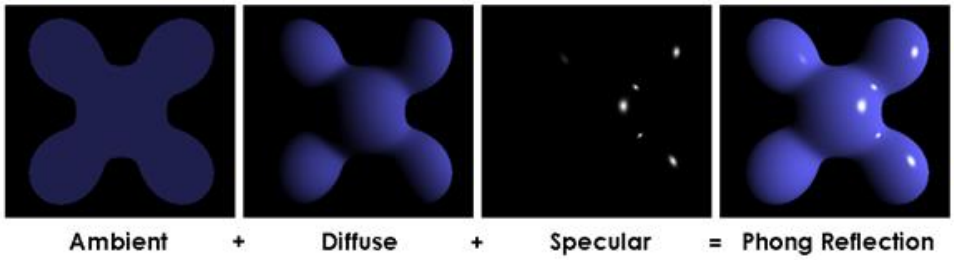
- **Emissive** : modélise une émission de lumière, essentiel pour visualiser un objet ou particule émettant de la lumière source luminescente (*self illumination*).

- **Ambient**: couleur globale environnante. Indique l'ambiance lumineuse d'une scène. Normalement cette composante dépend grandement des intéractions lumineuses (diffusions, absorptions, réflections, réfractions etc...), mais elle est ici approximée à une couleur.

- **Diffuse**: couleur propre de l'objet réfléchie par le matériau. Les matériaux diffus sont généralement non métallique, mais cette distinction n'existe pas dans le modèle de Phong.

- **Spéculaire** : couleur de la réflexion spéculaire. Cette couleur dépend techniquement du type de matériau, mais là encore cette distinction n'existe pas dans le modèle de Phong.

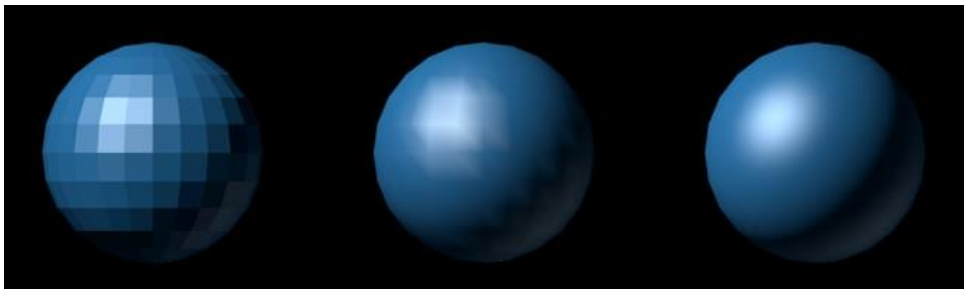
Dans le modèle de Phong la couleur finale est la somme de ces 4 composantes.



Les composantes emissive et ambiante sont essentiellement des constantes, de ce fait nous allons nous intéresser spécifiquement aux composantes diffuses et spéculaires dans les parties suivantes.

Avant cela, il faut comprendre la notion d'ombrage (shading) en particulier dans le cadre d'un GPU.
Comme leur nom l'indique les "shaders" ont historiquement un rapport étroit avec l'ombrage.

On distingue deux méthodes d'ombrage qui sont les méthodes d'ombrages de Gouraud (Gouraud Shading) et de Phong (Phong Shading).
Il existe une autre méthode d'ombrage par facette (Flat Shading) mais il s'agit d'un cas particulier.



Flat shading, Gouraud shading et Phong shading

<u>Note</u>: attention à ne pas confondre l'équation / modèle d'illumination de Phong ("Phong lighting") et la méthode de rasterization par interpolation ("Phong shading").

Le "Phong shading" se distingue du "Gouraud shading" ou méthode de Gouraud par le fait que la méthode d'Henri Gouraud interpole seulement des couleurs entre les sommets, tandis que la méthode de Phong est basée sur l'interpolation de tous les attributs (traditionnellement les normales et coordonnées de texture).

Nomenclature

Dans la suite du cours et des TP nous allons adopter la nomenclature suivante dans le corps du texte ainsi que dans les shaders.

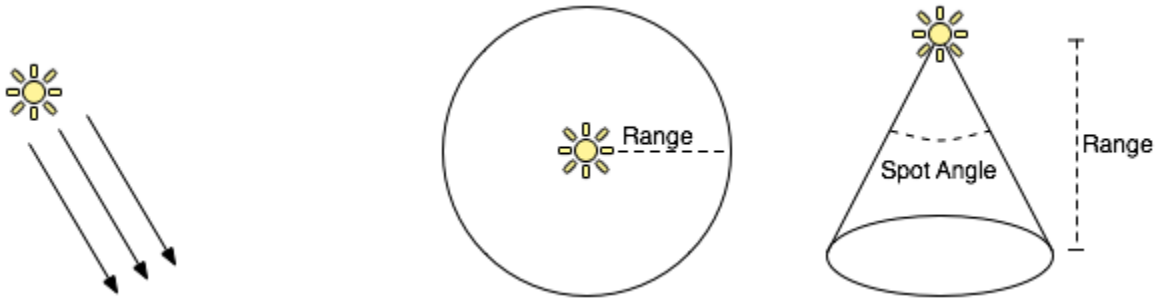
P = position d'un point (vertex ou fragment) dans l'espace
N = normale en un point (vertex ou fragment) dans l'espace
L = vecteur directeur normalisé allant d'un point VERS une lumière
E = position du point de vue (caméra) dans l'espace
V = vecteur directeur allant d'un point VERS la caméra
$\mathbf{i}_x$ = intensité lumineuse d'une composante (x pouvant être ambient, diffuse ou spéculaire)
$\mathbf{k}_x$ = couleur réfléchie par le matériau (x pouvant être emissive, ambient, diffuse ou spéculaire)

Partie 2 - Lumières

Sources lumineuses

Dans le modèle de Phong il est possible de spécifier une intensité lumineuse sous forme du tri-stimuli RGB par exemple, ou bien mono chromatique.
On a ainsi des intensités séparées par composantes, ia pour l'ambiance, id pour l'émission diffuse, et is pour l'émission spéculaire.
Là encore, rien de correct du point de vue physique, mais cela s'avère pratique afin que les artistes aient plus de controle sur le rendu.

On distingue 3 types de sources lumineuses principales: les lumières **directionnelles**, les lumières **omnidirectionnelles** et les **spots**, respectivement, de gauche à droite ci-dessous.



Il s'agit de lumières dites "ponctuelles" ce qui signifie qu'elles sont supposées ne pas avoir d'aire d'émission -elles sont réduites à un simple point infinitémisal-, ce qui ne veut pas dire que l'aire d'influence est nulle. On voit nettement que les omnidirectionnelles et les spots ont un rayon d'émission.

Réduire l'émission à une dimension infinitésimale permet de réduire la complexité des calculs en évitant d'avoir recours à des intégrales et autres outils mathématiques coûteux afin d'évaluer la quantité d'énergie émise.

Dans ce TP nous allons nous intéresser spécifiquement aux lumières **directionnelles**.
La particularité des lumières directionnelles est de ne pas avoir de position stricto-sensus.
On considère ainsi les lumières directionnelles comme étant des sources lumineuses distantes.

En pratique, tous les rayons lumineux qui atteignent une surface sont supposés être parallèles, ce qui est plus simple et moins coûteux en terme de calcul.

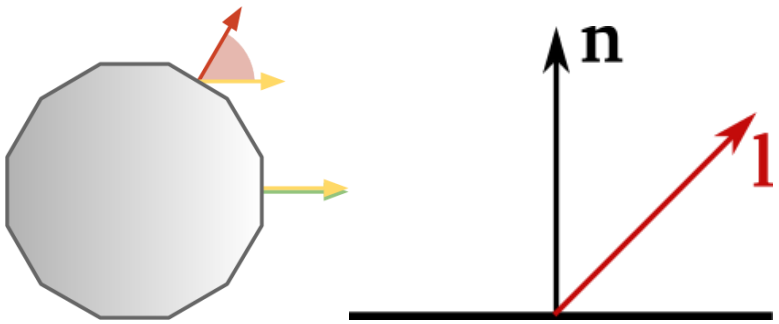
Note: un autre facteur important contribuant au réalisme de l'illumination est **l'atténuation** mais nous laisserons pour le moment de côté les phénomènes de déperdition et d'absorption pour le moment.

Dans le modèle de Phong les intensités lumineuses peuvent être des scalaires ou bien un tristimulus RGB.
On peut donc, au choix, modéliser les lumières comme une lumière blanche (1.0,1.0,10) ou bien colorée.

Loi du cosinus de Lambert

C'est l'occasion d'aborder une première loi d'optique géométrique qui est la loi du cosinus de Lambert.

La formulation de la loi du cosinus de Lambert stipule qu'une surface perçoit un pourcentage d'intensité lumineuse qui dépend de son orientation par rapport à la source lumineuse.
Ainsi, une surface reçoit 100% d'intensité lorsqu'elle est parfaitement alignée (colinéaire, ou parallèle) à la direction de la lumière, et 0% d'intensité lorsqu'elle est parfaitement perpendiculaire à la direction de la lumière.



En marron, le vecteur L, vecteur allant du fragment (point) VERS la source lumineuse (directionnelle ici)

La normale d'un plan, ou normale en un point, est un élément qui nous permet d'évaluer facilement cette équation.

En effet, mathématiquement on sait que le cosinus d'un angle entre deux vecteurs peut être obtenu par le **produit scalaire** de deux vecteurs <u>normalisés</u>.

$$\begin{bmatrix} A_x & A_y & A_z \end{bmatrix} \begin{bmatrix} B_x \\ B_y \\ B_z \end{bmatrix} = A_x B_x + A_y B_y + A_z B_z = \vec{A} \cdot \vec{B}$$

$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \|\vec{b}\| \cos\theta$$

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$$

Pour rappel, le résultat de la fonction cosinus est une valeur réelle (scalaire) bornée par [-1;+1].

Le produit scalaire est très utile en infographie car il permet d'indiquer l'orientation d'un vecteur par rapport à un autre. Dans notre cas, il nous permet de savoir si deux vecteurs sont dans la même hémisphère en fonction du signe du produit scalaire.
De plus, on obtient une information angulaire qui permet d'indiquer le degré de similitude.

Pour implémenter la loi du cosinus de Lambert on va utiliser le résultat du produit scalaire entre la normale (N) et la direction VERS la lumière (L) comme facteur d'intensité.

Eq1. Lambert = max(N.L, 0) * I_d ,
où I_d est l'intensité diffuse de la lumière et N.L = cos**θ**

Illumination d'une sphère avec ambience seule (gauche) et diffuse (droite)

Ce facteur, qu'on appelera **NdotL**, ou facteur diffus, s'applique à la composante "diffuse" de l'équation d'illumination de Phong.

- Exercice 1 : implémentez la fonction vec3 **diffuse**(vec3 N, vec3 L) dans un Fragment Shader. Cette fonction prend comme paramètre la normale N en un point et la direction VERS la lumière L.
Notez que la fonction doit renvoyer un vec3 RGB, il faut donc également tenir compte de l'intensité lumineuse de la lumière qu'on suppose RGB également.

La fonction **dot**(u, v) du GLSL retourne le produit scalaire entre les vecteurs u et v.
La valeur du produit scalaire pouvant être négative, ce qui n'a pas de sens en infographie car les intensité lumineuse ne peuvent être négative.
Utilisez la fonction **max**() du GLSL afin de borner la fonction entre [0;+1].

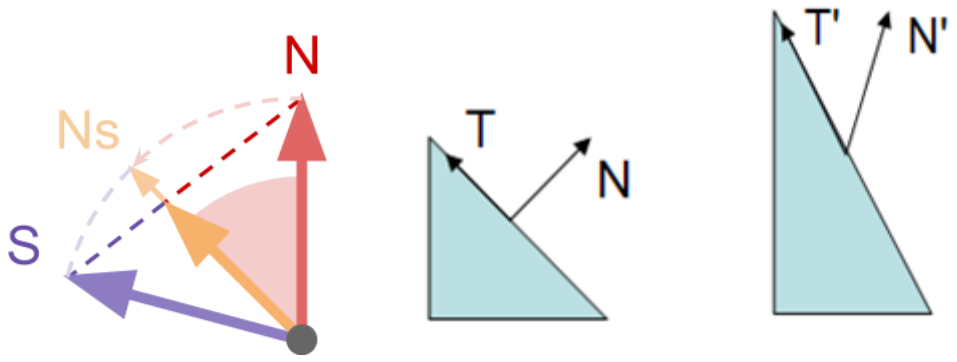
Étape 1 : il vous faut transmettre la direction de la lumière (ou bien la direction VERS la lumière) sous forme d'une variable **uniform** (ou une constante en dur dans le shader pour tester).

Étape 2 : Votre Vertex Shader doit accepter un **attribut** de Vertex Normal en entrée et le transmettre au Fragment Shader (en varying donc).
Notez que la normale sera alors interpolée par le rasterizer ce qui correspond à l'ombrage de Phong.

--- Important ----------

Il est fondamental de re-normaliser les normales dans le Fragment Shader. En effet, le passage Vertex Shader vers Fragment Shader se fait par le biais du rasterizer. Or ce dernier procède par interpolation linéaire. Comme vous pouvez le voir ci-dessous dans la figure de gauche, le vecteur interpolé (lerp) entre S et N voit sa magnitude (longueur) changer et donc être dénormalisé. Il faudrait théoriquement effectuer une interpolation sphérique (slerp). Il n'est pas possible de modifier l'algorithme d'interpolation du rasterizer, cependant, comme l'interpolation est graduelle, il est suffisant de normaliser les normales en entrée du fragment shader à l'aide de la fonction GLSL

vec3 **normalize**(vec3) .



En général, on traite les vecteurs comme les sommets, les normales subissent donc les mêmes transformations et changements de repère à l'exception des translations - un vecteur directeur n'est pas influencée par une translation.

On peut représenter un vecteur directeur en GLSL de 2 façons, soit sous la forme d'un **vec3**, dans ce cas il ne pourra être transformé que par une **matrice 3x3**, soit sous la forme d'un **vec4** avec **zéro** comme dernière composante (**w**).

Une normale est une forme contrainte d'un vecteur directeur (généralement normalisé), elle doit toujours être perpendiculaire au plan tangent la surface.
Toutefois, dans certains cas de figure, la normale peut être perturbée. C'est le cas si la matrice de transformation n'est pas **orthogonale**.
Une matrice n'est pas orthogonale lorsque la transformation qui en découle produit une base (des axes) qui ne sont plus mutuellement perpendiculaires (par exemple à la suite d'une déformation) par exemple.

Dans la figure de droite (ci-dessus) le polygone subit une transformation de type scale non-uniforme (vers le haut). Appliquer la matrice de scale non-uniforme au vecteur N produit alors un vecteur N' qui modifie les propriétés de la normale : longueur différente (ce qui peut se corriger par une normalisation) et une normale qui n'est plus perpendiculaire au plan.

Il faut donc que la matrice de transformation préserve l'orientation des normales afin qu'on puisse l'utiliser également avec des normales.

Une telle matrice, souvent appelée « normal matrix », est la transposée de l'inverse.
Cf. http://www.lighthouse3d.com/tutorials/glsl-12-tutorial/the-normal-matrix/

En résumé : la transposée de l'inverse de **M** est notée $(\mathbf{M}^{-1})^{T}$.
C'est la matrice correcte à utiliser pour transformer les normales.
Toutefois, lorsque **M** est une matrice orthogonale, on sait que $\mathbf{M}^{-1} = \mathbf{M}^{T}$.
Donc $(\mathbf{M}^{-1})^{T} = (\mathbf{M}^{-1})^{-1} = (\mathbf{M}^{T})^{T} = \mathbf{M}$.

Partie 3 - Matériaux

a. propriétés

Dans la réalité, la perception d'un objet dépend grandement de l'intéraction de ses propriétés intrinsèques avec la lumière. On parle alors de **matériau** (*material*) pour qualifier l'apparence d'un objet.

Au niveau macroscopique, la couleur apparente d'un matériau dépend de sa nature réfléxive (pour simplifier, métallique ou non-métallique, rugueux ou poli) mais également de la lumière reçue.

La couleur apparente d'un matériau (en RGB) est d'abord composée de sa couleur propre (parfois appelée albedo) qui indique la couleur réfléchie après absorption.

En clair, un matériau absorde (ou transforme) une partie de l'énergie (lumière) perçue et restitue (réfléchi) une partie de l'énergie.

Dans le modèle d'illumination de Phong, les composantes "ambient", "diffuse" et "specular" du matériau se mélangent individuellement avec les composantes similaires de la lumière.

Dans le cas d'un matériau diffus on calcul la valeur finale de la composante diffuse ainsi :

Eq2. Diffuse = N.L * I_d * K_d

Plutôt que d'utiliser une couleur uniforme K_d pour l'ensemble d'un matériau il est souvent préférable d'utiliser une texture afin d'introduire plus de réalisme et de variation.

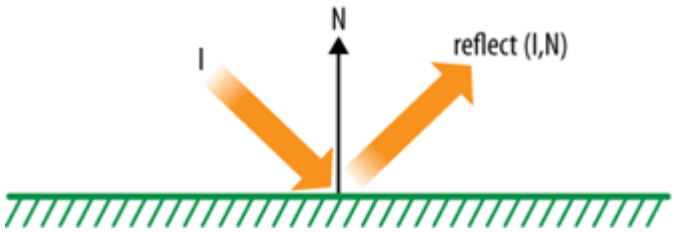
- Exercice 2 : Affichez votre modèle en combinant texture et illumination

b. Spécularité

Le modèle de Phong introduit une méthode simple et efficace permettant de simuler les reflets spéculaires. Les reflets spéculaires sont dépendant du point de vue (oeil, caméra) c'est-à-dire qu'ils n'apparaissent que dans la direction de l'observateur.

Il nous faut donc une autre variable qui est la direction de l'observateur. Cette direction –que l'on nommera V- peut se calculer comme le vecteur allant du point à illuminer (P) à la position de la caméra (E).

Le reflet spéculaire se produit selon la loi de réflexion de Ibn Sahl, Snell, Descartes qui stipule que le réfléchi R d'un vecteur incident I par rapport à la normale N d'une surface est réfléchi par le même angle que l'angle séparant le vecteur incident est la normale. Autrement dit, N.L == R.N, où L = -I.

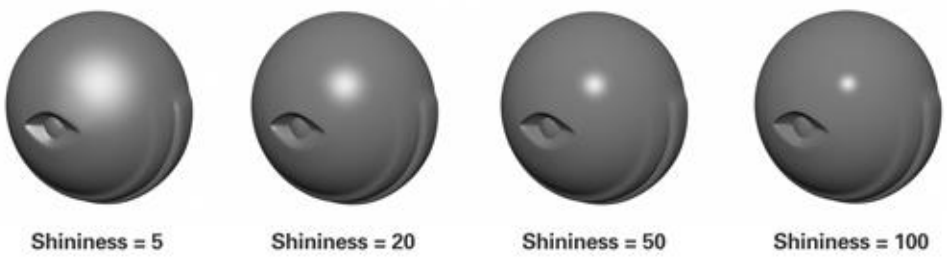


La fonction vec3 **reflect**(vec3 incident, vec3 normal) du GLSL permet de calculer ce vecteur R.

A l'aide du vecteur R, on peut implémenter la composante spéculaire du modèle de Phong :

Eq3. Specular = max((R.V)^{shininess}, 0) * I_s * K_s

"*shininess*" est un facteur de puissance spéculaire qui permet de contrôler la netteté du spéculaire. Concrètement, plus une surface est lisse plus son coefficient spéculaire est élevé. A contrario, plus une surface est rugueuse, moins le coefficient spéculaire est élevé.



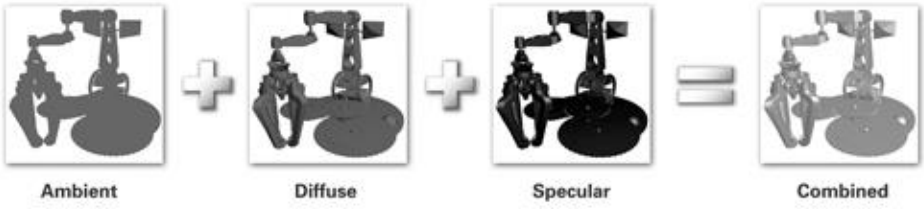
Is est la couleur de la composante spéculaire émise par la source lumineuse, tandis que K_s est la couleur de la composante spéculaire du matériau. Théoriquement cette couleur spéculaire dépend du type de matériau (métallique ou non). Les objets non métalliques (diélectriques) reflètent directement l'intensité de la source tandis que les objets métalliques reflètent leur couleur propre (albedo). Plus de détail dans un cours ultérieur.

Note: parfois K_s est remplacé par une texture de contrôle appelée "specular map" permettant à un artiste de spécifier précisement l'intensité de la réflexion spéculaire d'un texel (de 0% à 100%).

- Exercice 3 - implémentez la fonction vec3 **specular**(…)

Il est important de remarquer que seule les polygones faisant face à la lumière montrent une réflexion spéculaire. On pourra donc utiliser le signe de N.L pour vérifier si le fragment d'une face est spéculaire.

- Exercice 4 – implémentez le modèle d'illumination de Phong au complet.



Il est recommandé d'utiliser les structures suivantes dans le shader

```
struct Light {
    vec3 direction; // I ou L
    vec3 diffuseColor;  // Id
    etc…
};
uniform Light u_light;
```

```
struct Material {
      vec3 diffuseColor;  // Kd
      vec3 specularColor; // Ks
      etc…
};
uniform Material u_material;
```

En GLSL les champs d'une structure se référencent depuis le C/C++ en spécifiant le nom complet. Par exemple, si vous souhaitez affecter une valeur au champ "direction" de u_light il faut récupérer le location de "u_light.direction" :

```
int lightDir_location = glGetUniformLocation(program, "u_light.direction");
```

--- Très important -------

Assurez-vous que **tous vos calculs** se font dans le **même espace** (généralement l'espace Monde/World c'est-à-dire celui de la scène).
Veillez donc à ce que tous les vecteurs et positions soient bien dans le repère de la scène en les multipliant par les matrices de transformation local vers monde (la Tranform de l'objet donc).

Partie 4 – TP à rendre: modèle d'illumination de Blinn-Phong

Jim Blinn, autre personnage illustre de l'infographie, a modifié la composante spéculaire du modèle de Phong de sorte à le rendre un peu plus optimal pour les machines de l'époque.
Le modèle de Blinn-Phong a d'autres propriétés intéressantes dont on reparlera plus tard, le changement principal étant l'abandon du vecteur R issue de la fonction reflect() qui est remplacé par un vecteur H bisecteur (half-vector, semi-vecteur) entre L et V :

H = normalize((L + V) / 2) = normalize(L + V)

Eq4. Specular = $\max((N.H)^{shininess}, 0) * I_s * Ks$

Notez que l'équation utilise maintenant la normale de façon explicite.
Le paramètre "shininess" fonctionne de la même manière que pour l'équation de Phong mais les valeurs de puissance ne sont pas équivalentes : Shininess_Blinn ~= ¼ Shininess_Phong.

Paramètrage par fichier MTL

Le format OBJ décrit le format MTL permettant de spécifier les paramètres des matériaux associés à une surface ou un objet.
A l'aide Tiny Obj Loader, récupérer les informations de matériau telles que les couleurs diffuse, spéculaire ainsi que le paramètre de brillance spéculaire (shininess, specular highlight)

Bonus – attenuation

La lumière -ici, la somme des composantes diffuse et spéculaire - subit une atténuation dûe à son passage dans plusieurs médias absorbant son énergie. Les formules usuelles sont les suivantes :

$1/\text{distance}$ ou $1/\text{distance}^2$ ou encore $1/(1+\text{distance}^2)$

La forme la plus répandue dans les API graphique est celle-ci: $1/(k_c+kld+kqd^2)$

avec d = distance, k_c = constante, k_l = linéaire, k_q = quadratique

Bonus - Lumières omnidirectionnelles

Simuler une source lumineuse omnidirectionnelle (émettant dans toutes les directions) nécessite de prendre en compte la distance entre la source et un point sur une surface.
Il nous faut donc remplacer la direction de la lumière par un vecteur directeur (normalisé) L calculé comme le différenciel entre un point P et la position de Pl de la lumière.

Références

Modèle d'illumination de Phong
http://morpheo.inrialpes.fr/~franco/3dgraphics/practical4.html

Sources lumineuses
https://learnopengl.com/Lighting/Light-casters

Composante diffuse
https://www.alanzucconi.com/2019/10/08/journey-sand-shader-2/
https://www.tomdalling.com/blog/modern-opengl/06-diffuse-point-lighting/

Composante spéculaire

Texture spéculaire (Specular Map)