Chapitre 5

Éclairement

5.1 Introduction

Les clefs d'un rendu réaliste de l'éclairement :

- Des sources lumineuses induisant des éclairements différents selon l'orientation des faces par rapport à ces sources
- Un modèle d'ombrage fournissant un rendu réaliste
- Des composantes réfléchies de la lumière pour donner à certains objets un aspect brillant ou réléchissant
- Un ajout de texture sur les objets pour leur donner une apparence telle que le bois ou la pierre

5.2 Modèle d'éclairement

• Les deux composantes de la lumière :

- Lumière ambiante non directionnelle
- Sources lumineuses directionnelles
  - à distance finie (*ex.* un spot)
  - à distance infinie (*ex.* le soleil)

Remarque 5.2.1 Le modèle que nous considérons ne prend pas en compte l'éclairage réciproque des objets

• Les trois modes d'intéraction entre les objets et la lumière :

- L'absorption de la lumière ($\rightarrow$ conversion en chaleur)
- La réflexion vers d'autres objets
- La transmission à travers l'objet

• On a besoin de trois données (Fig. 5.1) :

- Le vecteur $\overrightarrow{n}$, normale à la surface
- Le vecteur $\overrightarrow{v}$ de direction de visualisation
- Le vecteur $\overrightarrow{s}$ de direction de la source lumineuse

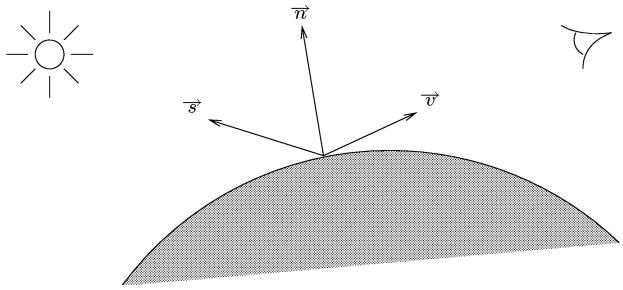

FIG. 5.1 – *Les vecteurs nécessaires au calcul de l'éclairement*

5.3 Les modes de réflexion

On considère deux types de réflexions, la diffusion et la réflexion spéculaire :

– Éparpillement diffus (surfaces poreuses)
  ↪ pas de direction privilégiée de réémission.
  ↪ forte intéraction avec l'objet
    → la lumière diffuse est fortement affectée par la couleur de la surface
– réflexion spéculaire (surfaces brillantes)
  ↪ réflexion selon une direction privilégiée, fonction de l'angle de la source avec la normale
  ↪ faible intéraction avec l'objet
    → la lumière réfléchie est proche de la lumière incidente avec une composante plus ou moins importante de l'objet

Les surfaces combinent en général les deux modes de réflexion de la lumière.

5.4 Lumière diffuse

Le réflexion diffuse est non directionnelle, donc indépendante de l'angle entre la normale et le vecteur de vue (Fig. 5.2).

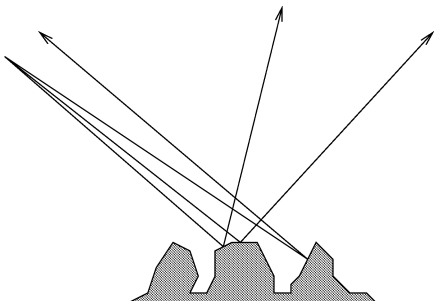

FIG. 5.2 – *Réflexion diffuse*

On utilise la loi de Lambert pour modéliser ce type de réflexion :

Loi de Lambert : La lumière diffuse dépend de l'angle entre la soure et la normale à la surface. On a

$$I_d = I_{sd}\rho_d \cos\alpha$$

où

I_d : Intensité de la lumière diffuse
I_{sd} : Intensité de la source lumineuse
ρ_d : Cœfficient de réflexion diffuse du matériau
α : angle de la source avec la normale à la surface

On a donc, pour un objet convexe,

$$I_d = I_{sd}\rho_d \max\left(\frac{\overrightarrow{s}.\overrightarrow{n}}{||\overrightarrow{s}||.||\overrightarrow{n}||}, 0\right)$$

(les parties cachées ne sont pas éclairées)

5.5 Lumière spéculaire

Il se produit un effet de faisceau pour les surfaces réfléchissantes qui ne sont pas des miroirs parfaits (Fig. 5.3) :

- La lumière est réfléchie maximalement dans la direction miroir parfait $\overrightarrow{r}$
- L'intensité de la lumière réfléchie diminue rapidement lorsqu'on s'éloigne de l'angle de réflexion

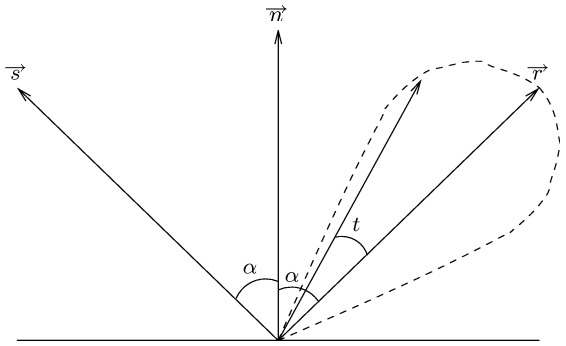

FIG. 5.3 – *Réflexion spéculaire*

Le vecteur de réflexion miroir est :

$$\overrightarrow{r} = -\overrightarrow{s} + 2\frac{\overrightarrow{s}.\overrightarrow{n}}{||\overrightarrow{n}||^2}\overrightarrow{n}.$$

On calcule ce type de réflexion selon le modèle de Phong.

Modèle de Phong : L'intensité de la lumière réfléchie est

$$I_s = I_{ss}\rho_s \cos^f t$$

où

I_s : Intensité de la lumière spéculaire
I_{ss} : Intensité de la source lumineuse
ρ_s : Cœfficient de réflexion spéculaire du matériau
t : angle entre le vecteur de vue et le vecteur de réflexion parfaite
f : paramètre de Phong

c'est-à-dire

$$I_s = I_{ss}\rho_s \max\left(\frac{\overrightarrow{r}.\overrightarrow{v}}{||\overrightarrow{r}||.||\overrightarrow{v}||}, 0\right)^f$$

Rôle de f : plus il est grand, plus on se rapproche du comportement d'un miroir parfait (Fig. 5.4).

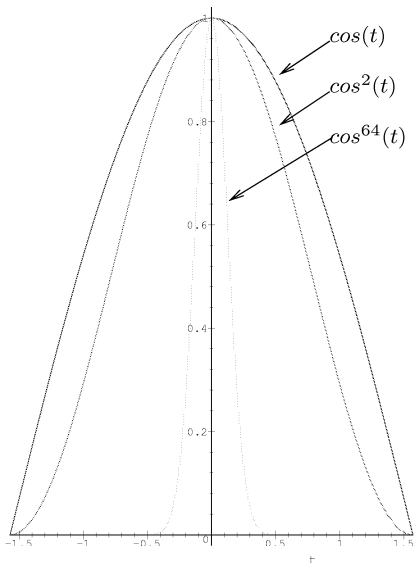

FIG. 5.4 – *Effet du cœfficient de Phong*

Accélération du calcul : on utilise le vecteur de mi-chemin : $\overrightarrow{h} = \overrightarrow{s} + \overrightarrow{v}$. Lorsque $\overrightarrow{s}$ et $\overrightarrow{v}$ sont de même norme, il s'agit de la normale à la surface qui fournirait le réflexion maximale (Fig. 5.5).

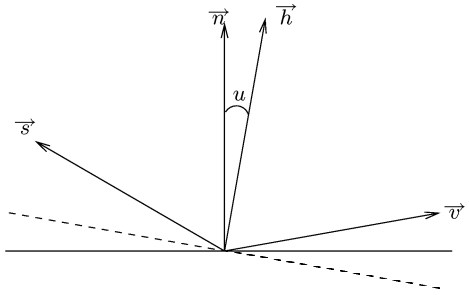

FIG. 5.5 – *Vecteur de mi-chemin*

L'angle u entre la normale et le vecteur de mi-chemin fournit alors une indication de la baisse d'intensité de la lumière spéculaire. La formule devient :

$$I'_s = I_{ss}\rho_s \max\left(\frac{\overrightarrow{h}.\overrightarrow{n}}{||\overrightarrow{h}||.||\overrightarrow{n}||}, 0\right)^f$$

L'accélération du calcul intervient si on suppose la source lumineuse et l'observateur à l'infini. En effet, dans ce cas, le vecteur $\overrightarrow{h}$ est constant sur toute la surface.

5.6 Lumière ambiante

On ajoute généralement un autre type de réflexion au modèle : celle de la lumière ambiante (provient de toutes parts de l'espace et émise dans toute les directions).

Cette réflexion produit une intensité constante qui ne dépend que du cœfficient de réflexion ambiante du matériau :

$$I_a = I_{sa}\rho_a$$

où

I_a	:	Intensité de la lumière ambiante
I_{sa}	:	Intensité de la source ambiante
ρ_a	:	Cœfficient de réflexion ambiante du matériau

5.7 Combinaison des réflexions et gestion des couleurs

Pour obtenir la luminosité totale on additionne les trois types de réflexions lumineuses vues précédemment : lumière ambiante, lumière diffuse et lumière spéculaire, on obtient donc :

$$I_{totale} = I_{sa}\rho_a + I_{sd}\rho_d \max\left(\frac{\overrightarrow{s}.\overrightarrow{n}}{||\overrightarrow{s}||.||\overrightarrow{n}||}, 0\right) + I_{ss}\rho_s \max\left(\frac{\overrightarrow{r}.\overrightarrow{v}}{||\overrightarrow{r}||.||\overrightarrow{v}||}, 0\right)^f.$$

Afin de prendre en compte la gestion des couleurs dans le calcul de la lumière on calcule séparemment l'intensité lumineuse de chacune des composantes de la couleur. Dans le système RVB, on ajoute ensuite les intensités rouge, verte et bleue. On définit donc les caractéristiques des sources lumineuses et des matériaux pour chacune de ces trois composantes :

$$I_{totale} = I_{totale,R} + I_{totale,V} + I_{totale,B}$$

et

$$\begin{cases} I_{totale,R} = I_{sa,R}\rho_{a,R} + I_{sd,R}\rho_{d,R}Lambert + I_{ss,R}\rho_{s,R}Phong^f \\ I_{totale,V} = I_{sa,V}\rho_{a,V} + I_{sd,V}\rho_{d,V}Lambert + I_{ss,V}\rho_{s,V}Phong^f \\ I_{totale,B} = I_{sa,B}\rho_{a,B} + I_{sd,B}\rho_{d,B}Lambert + I_{ss,B}\rho_{s,B}Phong^f \end{cases}$$

5.8 Ombrages

Afin de déterminer la couleur des points entre les sommets d'une primitive, on effectue une interpolation. Sur une arête on interpole les valeurs entre deux sommets puis, sur une ligne de remplissage, on interpole les valeurs entre deux arêtes.

Dans la suite nous considèrerons un triangle de sommets A, B, C et un point S dont on veut déterminer la couleur situé sur une ligne de remplissage intersectant les côtés $[AB]$ et $[BC]$ (Fig. 5.6). Notez qu'aucun des côtés n'est vertical, si c'était le cas, les équations s'en trouveraient modifiées (nous laissons au lecteur la réécriture des équations pour ce cas).

Deux méthodes sont généralement employées :

- ombrage de Gouraud
- ombrage de Phong

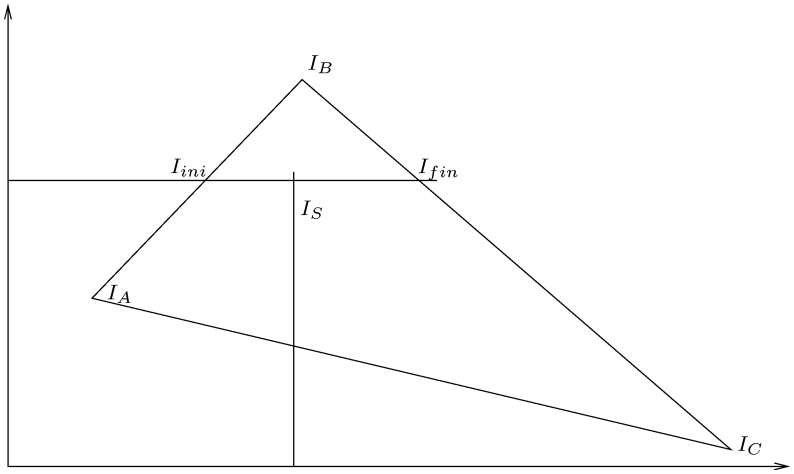

FIG. 5.6 – *Calcul de l'ombrage d'un triangle*

5.8.1 Ombrage de Gouraud

L'ombrage de Gouraud consiste à interpoler la couleur du pixel en fonction des couleurs des sommets (Fig. 5.7).

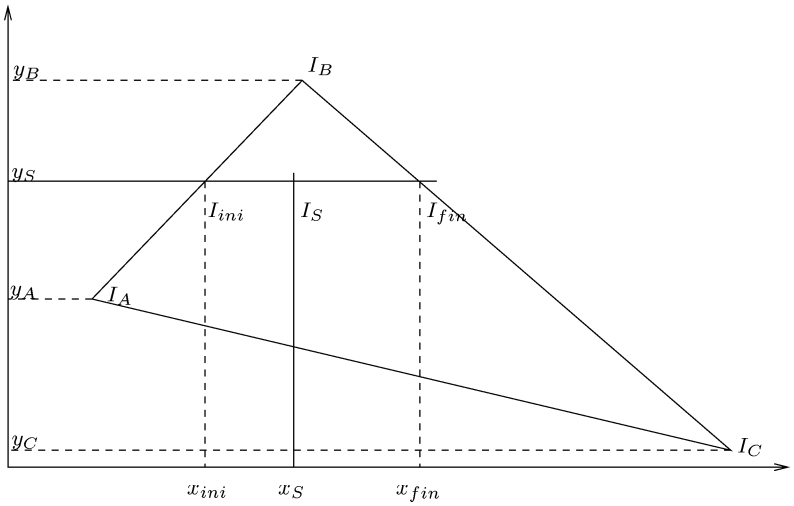

FIG. 5.7 – *Interpolation de Gouraud*

Le point de couleur I_{ini} se trouve sur la droite d'équation $P = tB + (1-t)A$. On en déduit que $y_{ini} = ty_B + (1-t)y_A$ et donc que

$$t = \frac{y_{ini} - y_A}{y_B - y_A}.$$

On en déduit donc la couleur I_{ini} :

$$I_{ini} = \frac{y_{ini} - y_A}{y_B - y_A} I_B + \left(1 - \frac{y_{ini} - y_A}{y_B - y_A}\right) I_A$$

soit, comme $y_{ini} = y_S$,

$$I_{ini} = I_A + \frac{y_S - y_A}{y_B - y_A} (I_B - I_A).$$

De même

$$I_{fin} = I_C + \frac{y_S - y_C}{y_B - y_C}\,(I_B - I_C)\,.$$

Enfin, le point S se trouve sur la droite d'équation $P = t.fin + (1-t)ini$. On en déduit que $x_S = tx_{fin} + (1-t)x_{ini}$ et donc

$$I_S = I_{ini} + \frac{x_S - x_{ini}}{x_{fin} - x_{ini}}\,(I_{fin} - I_{ini})\,.$$

5.8.2 Ombrage de Phong

Afin d'obtenir un ombrage prenant en compte la courbure des surfaces on peut utiliser l'ombrage de Phong. Celui-ci interpole les normales en chaque point au lieu d'interpoler directement les intensités lumineuses (Fig. 5.8). Il est donc plus coûteux que l'ombrage de Gouraud.

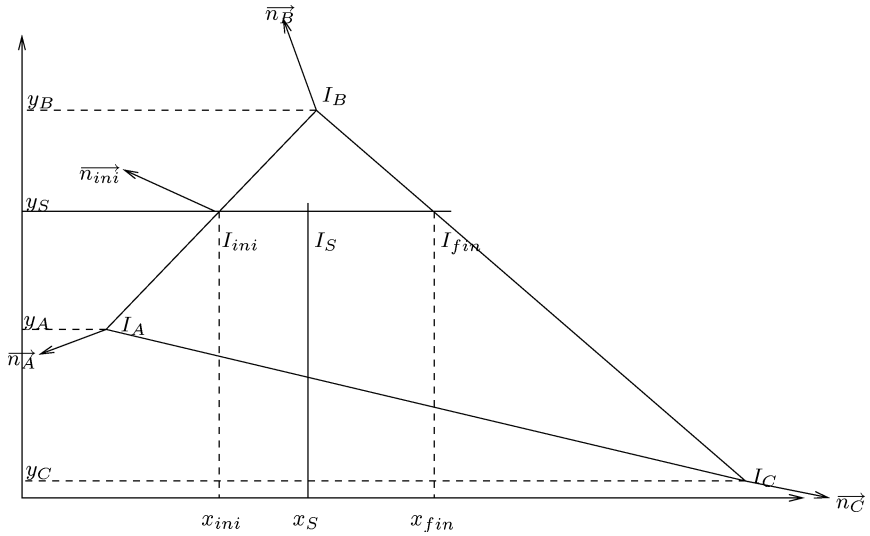

FIG. 5.8 – *Interpolation de Phong*

Par un raisonnement similaire à celui de la section précédente, on obtient :

$$\overrightarrow{n_{ini}} = \overrightarrow{n_A} + \frac{y_S - y_A}{y_B - y_A}\,(\overrightarrow{n_B} - \overrightarrow{n_A})\,,$$

$$\overrightarrow{n_{fin}} = \overrightarrow{n_C} + \frac{y_S - y_C}{y_B - y_C}\,(\overrightarrow{n_B} - \overrightarrow{n_C})\,,$$

$$\overrightarrow{n_S} = \overrightarrow{n_{ini}} + \frac{x_S - x_{ini}}{x_{fin} - x_{ini}}\,(\overrightarrow{n_{fin}} - \overrightarrow{n_{ini}})\,.$$

5.9 OpenGL

Création des sources lumineuses :

```
void glLight{if}[v](GLenum light, GLenum pname, TYPE param);
```

où

– `light` : `GL_LIGHT0`, `GL_LIGHT1`, ..., `GL_LIGHT7` (au moins huit sources lumineuses sont gérées par OpenGL).

- `pname` : caractéristique de la lumière qui est définie par `param`.
  - `GL_AMBIENT` : intensité de la lumière ambiante émise pas la source (par défaut : $(0, 0, 0, 1)$).
  - `GL_DIFFUSE` : intensité de la lumière diffuse (par défaut : $(1, 1, 1, 1)$ pour `LIGHT0` et $(0, 0, 0, 1)$ pour les autres).
  - `GL_SPECULAR` : intensité de la lumière spéculaire (par défaut : $(1, 1, 1, 1)$ pour `LIGHT0` et $(0, 0, 0, 1)$ pour les autres).
  - `GL_POSITION` : position de la lumière (par défaut : $(0, 0, 1, 0)$).
    - $w = 0$: la source est à l'infini, le vecteur indique le vecteur $\overrightarrow{s}$ devant être utilisé dans les calculs.
    - $w \neq 0$: la source est ponctuelle.

    Les coordonnées transmises sont multipliées par la matrice modèle-vue. Ce sont les coordonnées transformées qui seront utilisées dans les calculs.
  - `GL_SPOT_CUTOFF` : angle d'extinction de la lumière du spot (par défaut : 180, seuls les valeurs comprises entre 0 et 90 ainsi que la valeur spéciale 180 sont acceptées ; 180 signifie que la source lumineuse n'est pas un spot).
  - `GL_SPOT_DIRECTION` : direction des rayons du spot (par défaut : $(0, 0, -1)$, intervient seulement si `GL_SPOT_CUTOFF` est différent de 180). Le vecteur transmis est transformé par la matrice 3×3 supérieure de la matrice modèle-vue.
  - `GL_SPOT_EXPONENT` : atténuation radiale, voir plus loin (par défaut : 0).
  - `GL_CONSTANT_ATTENUATION` : facteur d'atténuation constante (par défaut : 1).
  - `GL_LINEAR_ATTENUATION` : facteur d'atténuation linéaire (par défaut : 0).
  - `GL_QUADRATIC_ATTENUATION` : facteur d'atténuation quadratique (par défaut : 0).

La forme "sans v" ne peut-être utilisée que lorsque le paramètre n'est composé que d'une valeur.

On active la gestion des lumières avec

```
void glEnable(GL_LIGHTING);
```

puis

```
void glEnable(GL_LIGHTi);
```

où `GL_LIGHTi` est le numéro de la source qu'on veut activer.

Les spots :

L'intensité lumineuse des spots n'est pas constante dans tout l'espace.

Il s'agit d'une atténuation radiale, celle-ci dépend de l'angle a formé par le vecteur spot-objet (correspondant à `GL_POSITION`) et le vecteur direction d'émission (`GL_SPOT_DIRECTION`) :

- angle d'extinction : si l'objet se trouve à l'extérieur du cône d'éclairage du spot, alors il n'est pas éclairé (angle supérieur à `GL_SPOT_CUTOFF`).
- atténuation : sinon un facteur d'atténuation est calculé (=$\cos^{GL\_SPOT\_EXPONENT}(a)$)

L'intensité lumineuse subit aussi une atténuation en fonction de la distance d de l'objet à la source lumineuse (si celle-ci est ponctuelle), cette atténuation est formée par trois paramètres, l'atténuation constante (k_c ou GL_CONSTANT_ATTENUATION), l'atténuation linéaire (k_l ou

GL_LINEAR_ATTENUATION) et l'atténuation quadratique (k_q ou GL_QUADRATIC_ATTENUATION). Le facteur d'atténuation est alors

$$\frac{1}{k_c + k_l.d + k_q.d^2}$$

Les différents modèles de lumières

On définit le modèle de lumière à utiliser avec

```
void glLightModel{if}[v](GLenum pname, TYPE param);
```

Valeurs de `pname` :

- `GL_LIGHT_MODEL_AMBIENT` : permet de définir la lumière ambiante de la scène (en plus de celles des sources lumineuses), par défaut : $(0.2, 0.2, 0.2, 1.0)$.
- `GL_LIGHT_MODEL_LOCAl_VIEWER` : indique comment les angles de réflexions sont calculés (peut-on considérer que l'observateur est à l'infini). Par défaut : `GL_FALSE` ; dans ce cas, lors du calcul du vecteur de mi-chemin (h), on utilise le vecteur $(0, 0, 1)$ comme vecteur v.
- `GL_LIGHT_MODEL_TWO_SIDE` : calcule-t-on l'éclairage des faces arrières, par défaut : `GL_FALSE`. Lorsque le paramètre vaut `GL_TRUE`, OpenGL inverse la normale à la face avant de calculer l'éclairement.
- `GL_LIGHT_MODEL_COLOR_CONTROL` : calcule-t-on séparément la couleur spéculaire (dans le cas de faces texturées), par défaut : `GL_SINGLE_COLOR`, l'autre possibilité étant `GL_SEPARATE_SPECULAR_COLOR`, elle n'intervient alors qu'après détermination de la couleur du pixel résultant de la texture.

Propriétés des matériaux

Les propriétés des matériaux sont définies à travers la fonction :

```
void glMaterial{if}[v](GLenum face, GLenum pname, TYPE param);
```

- `face` : définit quelles sont les faces affectées par la propriété, valeurs possibles : `GL_FRONT`, `GL_BACK` ou `GL_FRONT_AND_BACK`.
- `pname` : indique la propriété affectée par `param`.
  - `GL_AMBIENT` : valeur par défaut $(0.2, 0.2, 0.2, 1.0)$.
  - `GL_DIFFUSE` : valeur par défaut $(0.8, 0.8, 0.8, 1.0)$.
  - `GL_AMBIENT_AND_DIFFUSE` : permet de fixer les deux propriétés avec les mêmes valeurs.
  - `GL_SPECULAR` : valeur par défaut $(0, 0, 0, 1)$.
  - `GL_SHININESS` : cœfficient de Phong du matériau ($\in [0, 128]$), valeur par défaut 0.
  - `GL_EMISSION` : intensité émise par le matériau en plus de la lumière réfléchie, valeur par défaut $(0, 0, 0, 1)$

Il existe également une possibilité pour utiliser la fonction `glColor`, nous n'en dirons pas plus dans ce cours.

Calculs en OpenGL

Avec ce qui précède la compréhension du calcul de la couleur d'un pixel ne devrait pas poser de problèmes.

La couleur d'un pixel est l'addition de

- l'émission du matériau
- la lumière ambiante globale modulée par le matériau
- les lumières des sources modulées par le matériau

Pour l'émission il s'agit tout simplement des valeur RVB de `GL_EMISSION`.

Pour la lumière ambiante globale il s'agit des compsantes de la lumière ambiante définies dans le modèle de lumière multipliées par le cœfficient de réflexion ambiante du matériau.

L'intensité résultant des sources lumineuses est calculée par

$$f_a * e_{spot} * (t_a + t_d + t_s)$$

où f_a est le facteur d'atténuation par rapport à la distance vu précédemment ($= 1$ s'il s'agit d'une source à distance infinie).

e_{spot} est l'effet d'atténuation radiale du spot, il vaut 1 si la source n'est pas un spot, 0 si l'objet est extérieur au spot et sinon il vaut $(\max(\overrightarrow{-s}.\overrightarrow{d}, 0))^{GL\_SPOT\_EXPONENT}$ (ce que nous avons vu sous la forme $\cos^{GL\_SPOT\_EXPONENT}(a)$). $\overrightarrow{d}$ est le vecteur direction du spot (`GL_SPOT_DIRECTION`).

t_a est le terme ambiant : intensité ambiante émise par la source $\times$ le cœfficient ambiant du matériau.

t_d est le terme diffus : $\max(\overrightarrow{L}.\overrightarrow{n}, 0)$ $\times$ intensité diffuse émise par la source $\times$ le cœfficient diffus du matériau.

t_s est le terme spéculaire : $(\max(\overrightarrow{s}.\overrightarrow{n}, 0))^{shininess}$ $\times$ intensité spéculaire émise par la source $\times$ le cœfficient spéculaire du matériau.

Remarque 5.9.1 Les vecteurs sont considérés normalisés (OpenGL ne divise pas par la norme des vecteurs contrairement à ce qui a été vu plus haut), il faudra donc penser à entrer des vecteurs de norme 1, en particulier lorsqu'on spécifie des normales. Lorsque ce n'est pas possible, on peut spécifier à OpenGL qu'il aura à normaliser les normales avant le calcul de l'éclairement : `GL_NORMALIZE`. Lorsqu'on spécifie des «normales normées» et que les seules transformations suceptibles de les «dénormaliser» sont des mises à l'échelles uniformes, on peut utiliser `GL_RESCALE_NORMAL` (la norme est alors calculée à partir de la diagonale de la matrice `GL_MODELVIEW`).

5.10 Quelques propriétés de matériau

Les propriétés suivantes sont tirées de [McR].

Materiau	GL_AMBIENT	GL_DIFFUSE	GL_SPECULAR	GL_SHININESS
Laiton	0.329412 0.223529 0.027451 1.0	0.780392 0.568627 0.113725 1.0	0.992157 0.941176 0.807843 1.0	27.8974
Bronze	0.2125 0.1275 0.054 1.0	0.714 0.4284 0.18144 1.0	0.393548 0.271906 0.166721 1.0	25.6
Bronze poli	0.25 0.148 0.06475 1.0	0.4 0.2368 0.1036 1.0	0.774597 0.458561 0.200621 1.0	76.8
Chrome	0.25 0.25 0.25 1.0	0.4 0.4 0.4 1.0	0.774597 0.774597 0.774597 1.0	76.8
Cuivre	0.19125 0.0735 0.0225 1.0	0.7038 0.27048 0.0828 1.0	0.256777 0.137622 0.086014 1.0	12.8
Cuivre poli	0.2295 0.08825 0.0275 1.0	0.5508 0.2118 0.066 1.0	0.580594 0.223257 0.0695701 1.0	51.2
Or	0.24725 0.1995 0.0745 1.0	0.75164 0.60648 0.22648 1.0	0.628281 0.555802 0.366065 1.0	51.2
Or poli	0.24725 0.2245 0.06745 1.0	0.34615 0.3143 0.0903 1.0	0.797357 0.723991 0.208006 1.0	83.2
Étain	0.105882 0.058824 0.113725 1.0	0.427451 0.470588 0.541176 1.0	0.333333 0.333333 0.521569 1.0	9.84615
Argent	0.19225 0.19225 0.19225	0.50754 0.50754 0.50754	0.508273 0.508273 0.508273	51.2

Materiau	GL_AMBIENT	GL_DIFFUSE	GL_SPECULAR	GL_SHININESS
	1.0	1.0	1.0	
Argent poli	0.23125 0.23125 0.23125 1.0	0.2775 0.2775 0.2775 1.0	0.773911 0.773911 0.773911 1.0	89.6
Émeraude	0.0215 0.1745 0.0215 0.55	0.07568 0.61424 0.07568 0.55	0.633 0.727811 0.633 0.55	76.8
Jade	0.135 0.2225 0.1575 0.95	0.54 0.89 0.63 0.95	0.316228 0.316228 0.316228 0.95	12.8
Obsidienne	0.05375 0.05 0.06625 0.82	0.18275 0.17 0.22525 0.82	0.332741 0.328634 0.346435 0.82	38.4
Perle	0.25 0.20725 0.20725 0.922	1.0 0.829 0.829 0.922	0.296648 0.296648 0.296648 0.922	11.264
Rubis	0.1745 0.01175 0.01175 0.55	0.61424 0.04136 0.04136 0.55	0.727811 0.626959 0.626959 0.55	76.8
Turquoise	0.1 0.18725 0.1745 0.8	0.396 0.74151 0.69102 0.8	0.297254 0.30829 0.306678 0.8	12.8
Plastique noir	0.0 0.0 0.0 1.0	0.01 0.01 0.01 1.0	0.50 0.50 0.50 1.0	32
Caoutchouc noir	0.02 0.02 0.02 1.0	0.01 0.01 0.01 1.0	0.4 0.4 0.4 1.0	10

5.11 Un exemple

On va dessiner un triangle de couleur verte dans le plan (O, x, z), centré autour de l'origine. L'observateur est placé en $(0, 0, 4)$. On fait tourner le triangle autour de l'axe (Ox) lorsqu'on appuie sur la touche «expace». On ajoute à la scène un source lumineuse fixe par rapport à l'observateur à la position $(0, 4, -4)$.

```
#include<stdlib.h>
#include<GL/glut.h>

/* Angle de rotation de la flêche par rapport à l'axe (Ox) */
float angle=0;
/* Booléen indiquant si la flêche est en traine de tourner */
int tourne=0;

/* Propiétés de la lumière (blanche et ponctuelle) */
/* La position de la lumière sera affectée par la matrice modèle-vue ;
   pour la positionner à la position (0,4,-4) par rapport à l'observateur,
   il faut prendre en compte le fait que la matrice modèle-vue contiendra
   lors de la définition de la position de la lumière la matrice
   correspondant à un gluLookAt positionnant l'observateur en (0,0,4)
 */
static GLfloat composante_diffuse[]={.9,.9,.9,1.};
static GLfloat composante_ambiante[]={1.,1.,1.,1};
static GLfloat composante_speculaire[]={1,1,1,1.};
static GLfloat position_lumiere[]={0.,4.,0.,1.};

/* Propiétés du matériau de la flêche */
static GLfloat fleche_speculaire[]={.508273,.508273,.508273,1.};
static GLfloat fleche_diffuse[]={0,.50754,0,1.};
static GLfloat fleche_ambiente[]={.19225,.19225,.19225,1.};
static GLfloat fleche_shininess=51.2;

/* Initialisation du contexte OpenGL */
void myInit(void) {
  /* Couleur du fond */
  glClearColor(0.0,0.0,0.0,0.0);

  /* Matrice de projection : une perspective */
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glFrustum(-1,1,-1,1,1,20.0);

  /* Matrice modèle-vue : l'observateur est en (0,0,4) */
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  gluLookAt(0,0,4,0,0,0,0,1,0);
```

```
  /* Définition de l'éclairage */
  /* On active la gestion de l'éclairement */
  glEnable(GL_LIGHTING);
  /* On active la source lumineuse 0 */
  glEnable(GL_LIGHT0);
  /* La flêche n'est pas convexe, il faut donc gérer correctement
    l'éclairement des faces arrières */
  glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE);
  /* Définition des propriétés de la lumière */
  glLightfv(GL_LIGHT0, GL_DIFFUSE,composante_diffuse);
  glLightfv(GL_LIGHT0, GL_AMBIENT,composante_ambiante);
  glLightfv(GL_LIGHT0, GL_SPECULAR,composante_speculaire);
  glLightfv(GL_LIGHT0, GL_POSITION,position_lumiere);

  /* On active la gestion des faces cachées */
  glEnable(GL_DEPTH_TEST);
}

/* Affichage de la scène */
void display(void) {
  /* On fait le vide dans nos tampons */
  glClear(GL_DEPTH_BUFFER_BIT|GL_COLOR_BUFFER_BIT);

  /* On sauvegarde la matrice modèle-vue contenant la position de
     l'observateur
   */
  glPushMatrix();

  /* On fait tourner la flêche */
  glRotatef(angle,1,0,0);

  /* Propriétés du matériau de la flêche
     (ici, comme on n'a qu'un objet, on aurait pu les mettre dans
     myInit) */
  glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, fleche_ambiente);
  glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, fleche_diffuse);
  glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, fleche_speculaire);
  glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, fleche_shininess);
  /* On dessine la flêche (un triangle) */
  glBegin(GL_TRIANGLES);
  glNormal3f(0,0,-1);
  glVertex3f(-1,0,1);
  glVertex3f(1,0,0);
  glVertex3f(-1,0,-1);
  glEnd();

  /* On restaure la matrice modèle-vue */
```

```
  glPopMatrix();

  /* Et c'est fini */
  glutSwapBuffers();
}

/* Gestion du clavier */
void clavier(unsigned char key, int x, int y) {
  switch (key) {
  case 27: exit(0);
           break;
  case 32: tourne^=1;
  }
}

/* Fonction appelée quand la GLUT a le temps */
void idle(void) {
  if (tourne) {
    /* Si la flêche est en train de tourner, on modifie
       l'angle de rotation */
    angle-=.1;
    /* Il faut alors réafficher la scène */
    glutPostRedisplay();
  }
}

int main(int argc, char *argv[])
{
  /* Initialisation de la GLUT */
  glutInit(&argc, argv);
  /* Création des tampons */
  glutInitDisplayMode(GLUT_DEPTH|GLUT_DOUBLE|GLUT_RGBA);
  /* Création de la fenêtre et du contexte OpenGL */
  glutCreateWindow("Fleche");
  /* Initialisation de contexte OpenGL */
  myInit();
  /* Evènements de la GLUT */
  glutDisplayFunc(display);
  glutKeyboardFunc(clavier);
  glutIdleFunc(idle);
  /* Boucle principale */
  glutMainLoop();
  return 0;
}
```