

# Modèles d'illumination

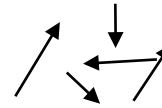


# Illumination

- L'intensité et la couleur d'une surface:
  - dépend des sources de lumière,
  - de l'orientation vis-à-vis de ces sources,
  - de la position de l'observateur, ainsi que du type de surfaces.
- Modèle de lumière : approxime l'ensemble de ces facteurs et donne une méthode pour les calculs requis.
- Sources lumineuses : réflecteurs ou émetteurs.
  - Réflecteurs : surfaces (les parois des objets) réfléchissant la lumière émise par des sources actives.
  - Émetteurs : sources de lumière.

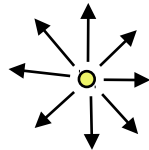
## Types de sources lumineuses

● Lumière non orientée (ambiante)

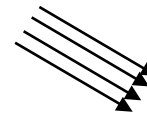


● Lumière orientée

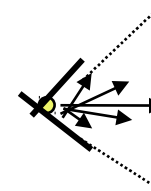
■ ponctuelle



■ à l'infini (directionnelle)



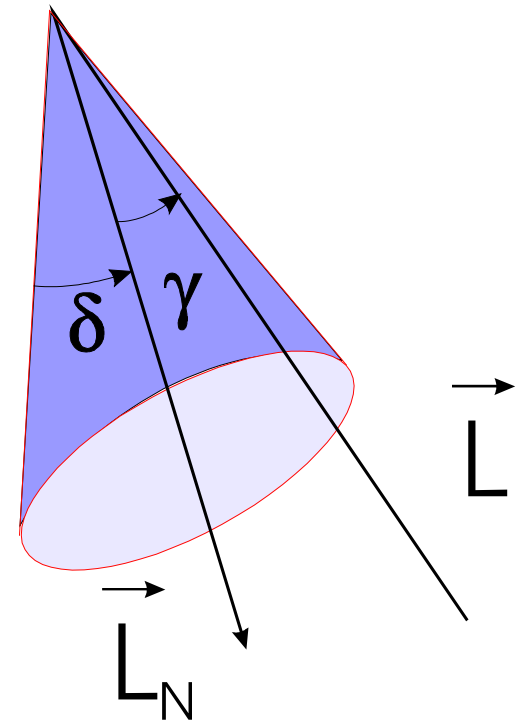
■ projecteur (spot)



## Types de sources lumineuses : projecteur (spot)

Un projecteur (spot) n'éclaire qu'à l'intérieur d'un cône de lumière.

- spécifier un angle d'ouverture  $\delta$  d'un cône dont le sommet correspond à la position de la source de lumière.
- l'intensité lumineuse est fonction de l'angle  $\gamma$  que fait la direction du rayon avec la direction principale de la source.



## Types de sources lumineuses

- Nous nous intéressons à l'intensité des sources lumineuses (monochromes, pour l'instant).
- **Ambiante**: intensité  $I_a$  uniforme
- **Ponctuelle**: intensité  $I_p$  uniforme
- **Projecteur**: intensité  $I_s$  variant selon la direction

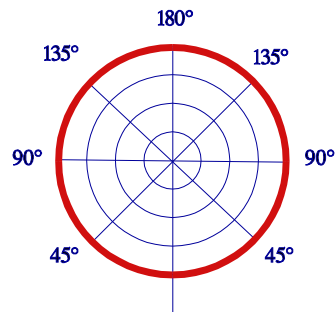
$$I_s = \begin{cases} I_p \cos^c \gamma & \text{si } \gamma < \delta \\ 0 & \text{sinon} \end{cases}$$

où  $c$  est l'exposant du réflecteur spéculaire ou exposant de concentration,  
 $\gamma$  l'angle entre  $\vec{L}$  et la normale à la direction principale  $\vec{L}_N$   
(le calcul du cosinus peut être remplacé par celui du produit scalaire de ces vecteurs)

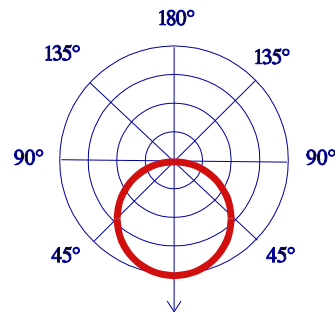
(modélisation inspirée de la réflexion spéculaire)

## Types de sources lumineuses : projecteur (spot)

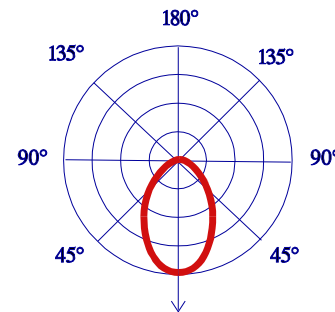
- Les diagrammes goniométriques illustrent l'intensité de la lumière en fonction de la direction angulaire autour d'un axe de lumière en coordonnées polaires.



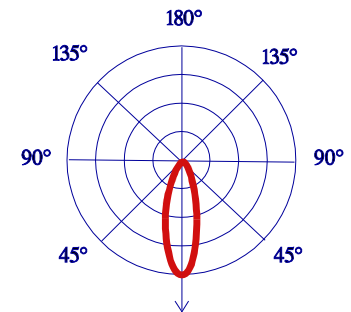
Source ponctuelle  
irradiant uniformément



$\cos \gamma$



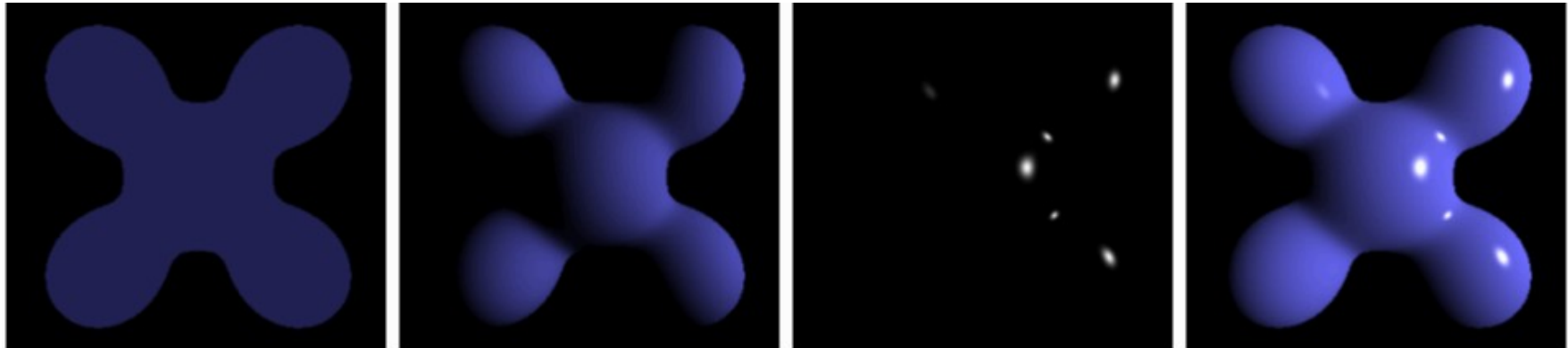
$\cos^4 \gamma$




$\cos^{32} \gamma$

# Le modèle de réflexion

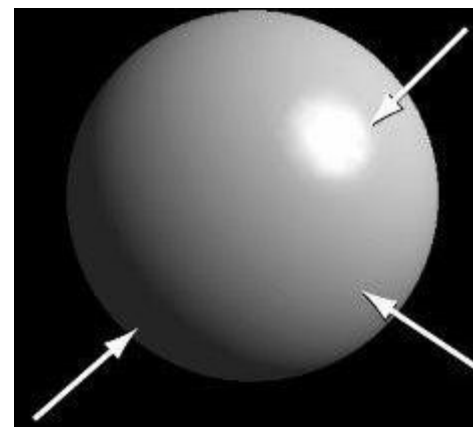
## Types de réflexion de la lumière



 **ambiante** + **diffuse** + **spéculaire** = **objet illuminé**

Cette décomposition en trois types de réflexion est le modèle de réflexion mis au point par **Bùi Tuong Phong** (1942-1975).

ambiante



spéculaire

diffuse

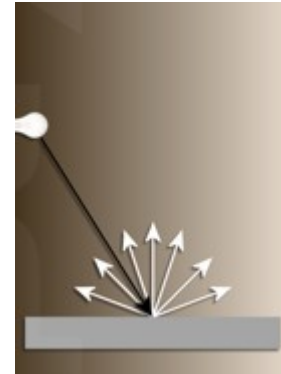




## Types de réflexion de la lumière

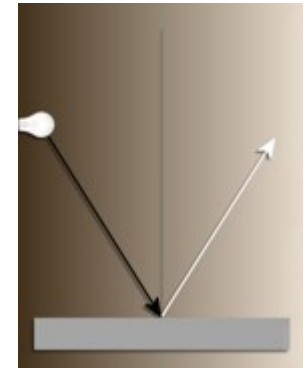
● **Réflexion diffuse:** lumière provenant d'une source lumineuse définie et réfléchi par la surface également dans toutes les directions. *Lumière reçue d'une direction et réfléchi dans toutes les directions.*

- Caractéristique: les rugosités de la surface produisent une intensité lumineuse réfléchi uniformément.



● **Réflexion spéculaire:** lumière provenant d'une source lumineuse définie et réfléchi par la surface dans une direction (ou un cône) directement opposée à la source lumineuse. *Lumière reçue d'une direction et réfléchi dans une direction.*

- Caractéristique: réflexion directionnelle et reflets



● **Réflexion ambiante:** lumière provenant de sources indéterminées et réfléchi par la surface également dans toutes les directions. *Lumière reçue de toute part et réfléchi dans toutes les directions.*



## Types de réflexion de la lumière

● **Réflexion ambiante:** niveau de lumière provenant de sources indéterminées.

- L'intensité des rayons émis par réflexion ambiante est

$$I = k_a I_a$$

où

$k_a$  : la réflectivité du matériau de la surface à la lumière ambiante et varie entre 0 et 1,

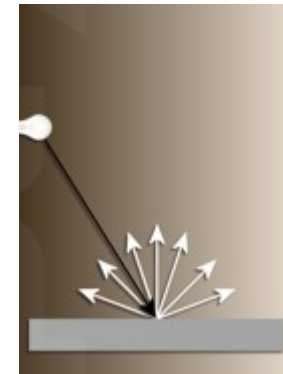
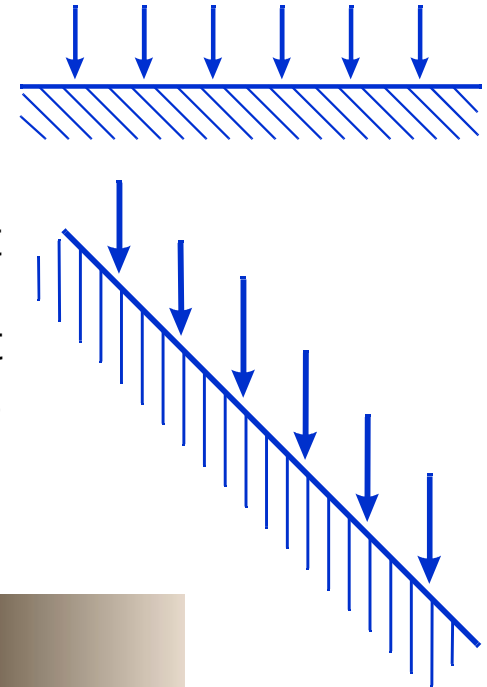
$I_a$  : intensité de la lumière arrivant sur la surface.



## Réflexion diffuse

### Réflexion diffuse de sources ponctuelles:

- La loi de Lambert en optique relie l'intensité avec l'angle entre la normale et la direction des rayons incidents.  
Ainsi une surface perpendiculaire à l'incidence émet plus intensément que lorsque placée dans le sens du rayonnement. Ceci suppose que les rayons sont parallèles, ce qui n'est pas le cas pour une source à une distance finie.
- Exprimée mathématiquement, la loi de Lambert donne  $I = k_d I_p \cos \theta$   
où  
 $I_p$  (ou  $I_s$ ): intensité de la source ponctuelle  
 $\theta$ : angle entre la normale et la direction de la source  
 $k_d$ : coefficient de réflectivité diffuse du matériau de la surface



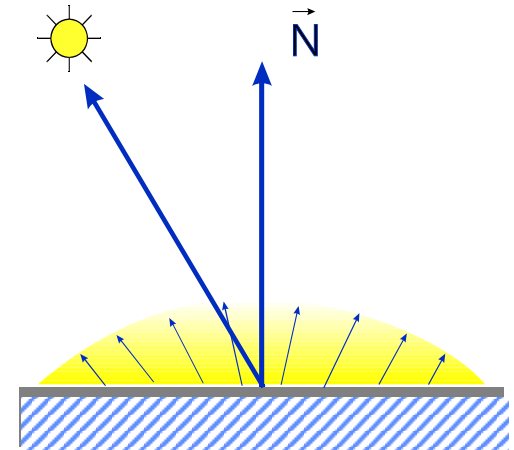
## Réflexion diffuse

- Le calcul de l'angle peut être effectué ainsi:

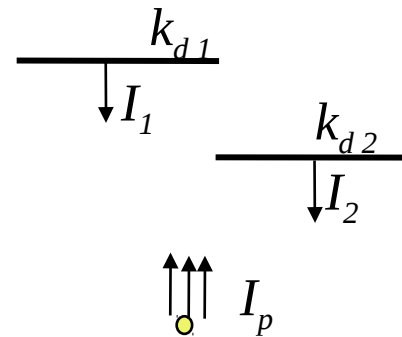
$$\cos \theta = (N * L)$$

où  $N$  : vecteur normal à la surface

$L$  : vecteur unitaire donnant la direction de la lumière.



- Mais si on a



- devrait-on obtenir  $I_1 = I_2$  ?

## Réflexion diffuse:

### Atténuation de la lumière en fonction de la distance

- Si la projection de deux surfaces parallèles d'un même matériau se superposent, l'équation de Lambert ne permet pas de distinguer où commence une surface et où se termine l'autre, quelque soit leur distance respective de la source de lumière. Pour remédier à ce problème, on introduit un facteur d'atténuation  $f_{att}$  de la lumière et l'équation devient:

$$I = f_{att} k_d I_p \cos \theta$$

- Un choix évident pour ce facteur d'atténuation  $f_{att}$  est basé sur le fait que la lumière s'atténue proportionnellement à l'inverse du carré de la distance  $d$  entre la source de lumière et la surface. Dans ce cas,

$$f_{att} = \frac{1}{d^2}$$

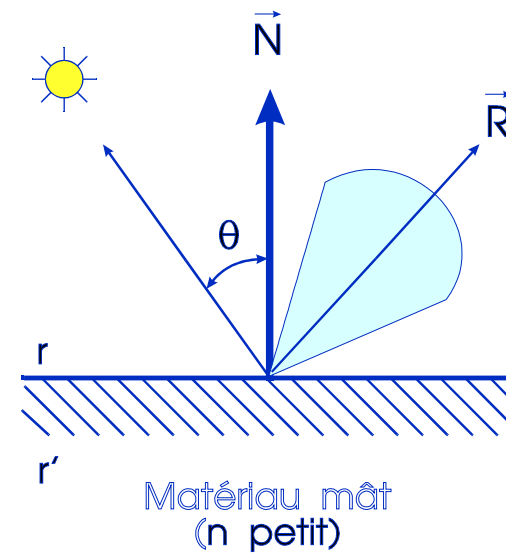
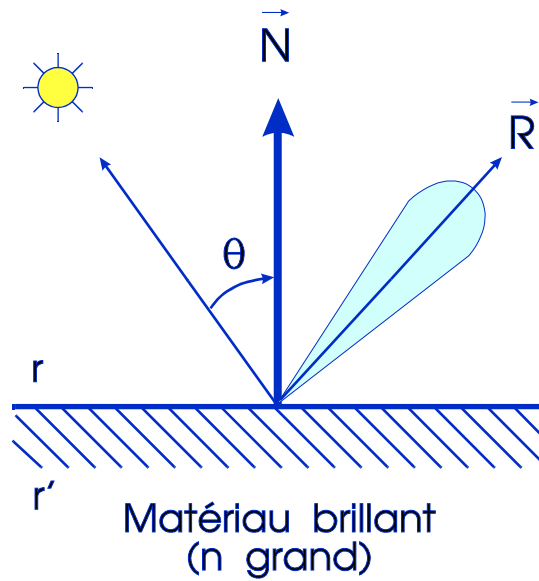
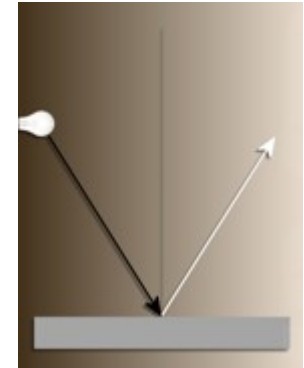
- Cependant, en pratique ce facteur ne donne pas de très bons résultats. En effet, si la source de lumière est très éloignée,  $1/d^2$  ne varie pas beaucoup. Un bon compromis est:

$$f_{att} = \min \left( \frac{1}{c_1 + c_2 d + c_3 d^2}, 1 \right) \quad \text{où } c_1, c_2, c_3 : \text{constantes} \geq 0$$

## Réflexion spéculaire

● La réflexion spéculaire est obtenue sur une plage d'angle ou un cône autour de cette valeur.

- Pour des matériaux brillants, ce cône est petit tandis que pour des matériaux mats, il est plus grand.



## Réflexion spéculaire - Phong

- Dans le modèle de *Phong*, l'intensité de réflexion spéculaire est proportionnelle à  $\cos \phi$  où  $\phi$  est l'angle entre la direction d'observation ( $\vec{V}$ ) et celle du rayon réfléchi idéal ( $\vec{R}$ ). Le facteur  $n$  sert à caractériser le fini de la surface ou la **brillance**.

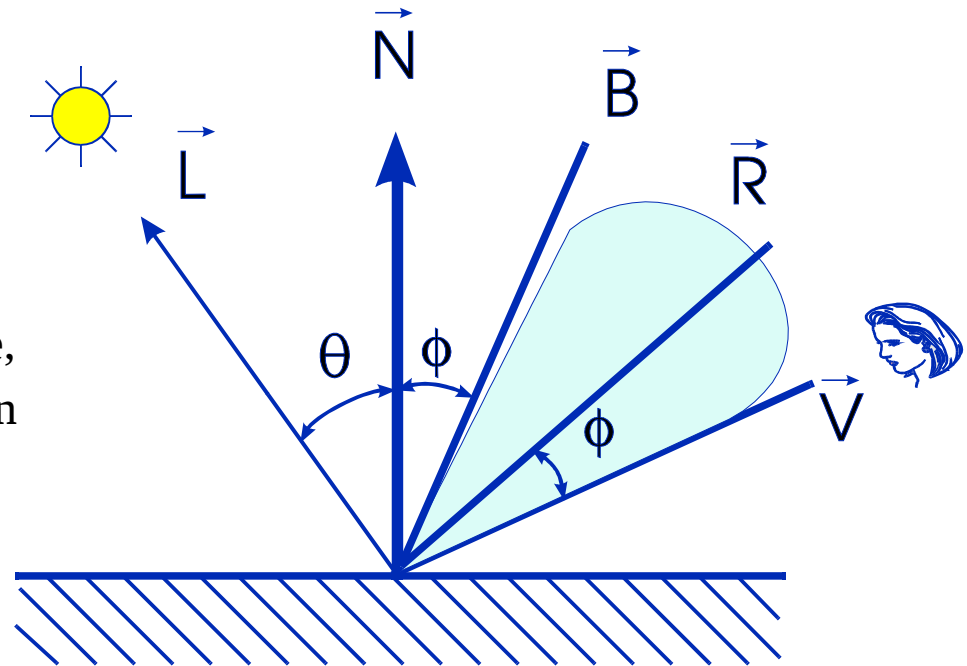
$> 100$  (pour des surfaces brillantes)

1 (pour des surfaces mates).

$$I = f_{att} k_s I_p (\cos \phi)^n$$

où  $I_p$  : intensité de la source ponctuelle,  
 $\phi$  : angle entre la direction d'observation  
 et celle du rayon réfléchi,

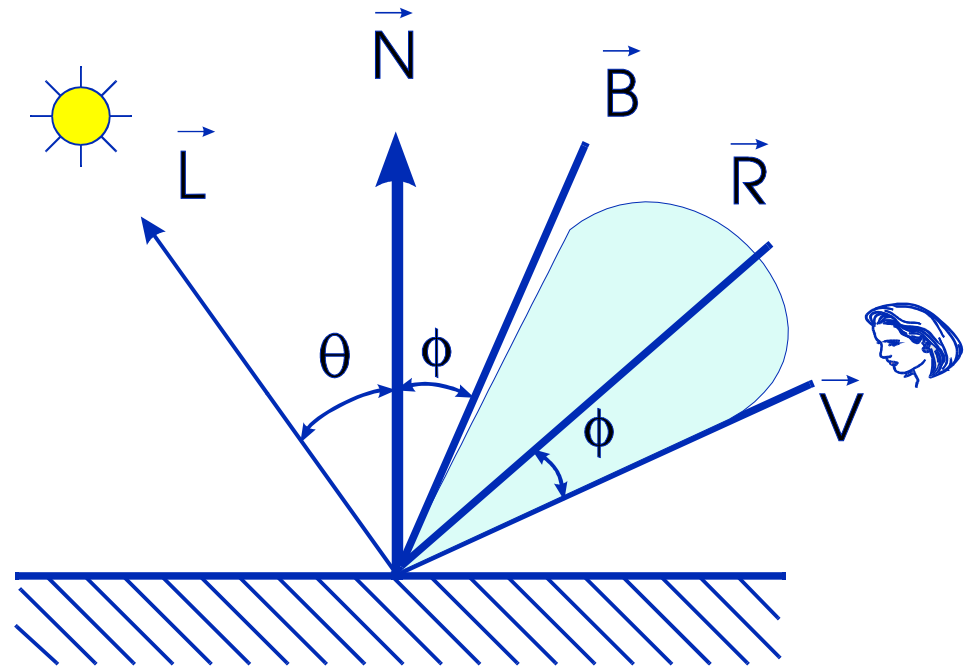
$k_s$  = coefficient de réflectivité spéculaire  
 du matériau de la surface.



## Réflexion spéculaire - Blinn

Plutôt que recalculer le produit scalaire  $\mathbf{R} \cdot \mathbf{V}$ , *Jim Blinn* définit le vecteur moyen,  $\mathbf{B}$ , entre  $\mathbf{V}$  et  $\mathbf{L}$  et utilise plutôt le produit scalaire  $\mathbf{B} \cdot \mathbf{N}$  comme valeur de  $\cos \phi$  dans l'équation. Le résultat est très semblable au modèle de Phong, mais moins coûteux à calculer.

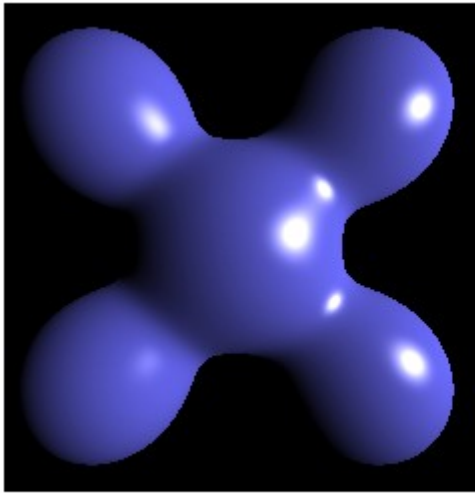
$$I = f_{att} k_s I_p (\cos \phi)^n$$



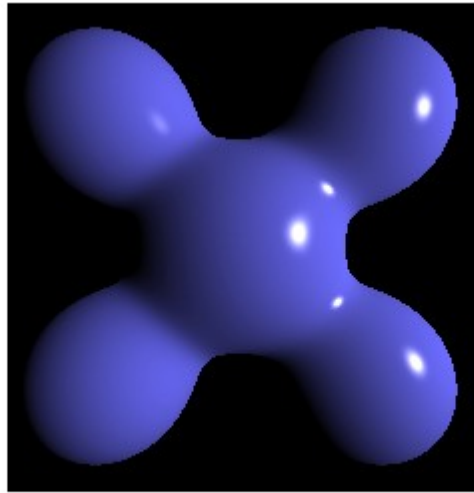


## Réflexion spéculaire

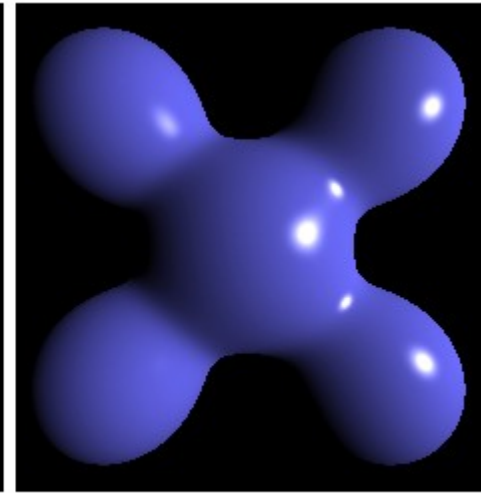
- On peut jouer avec l'exposant de brillance pour obtenir un effet presque identique entre les deux.



**Blinn-Phong**

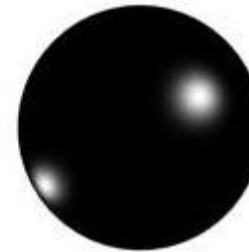


**Phong**



**Blinn-Phong**

*Enfin, presque...!*



## Réfraction spéculaire

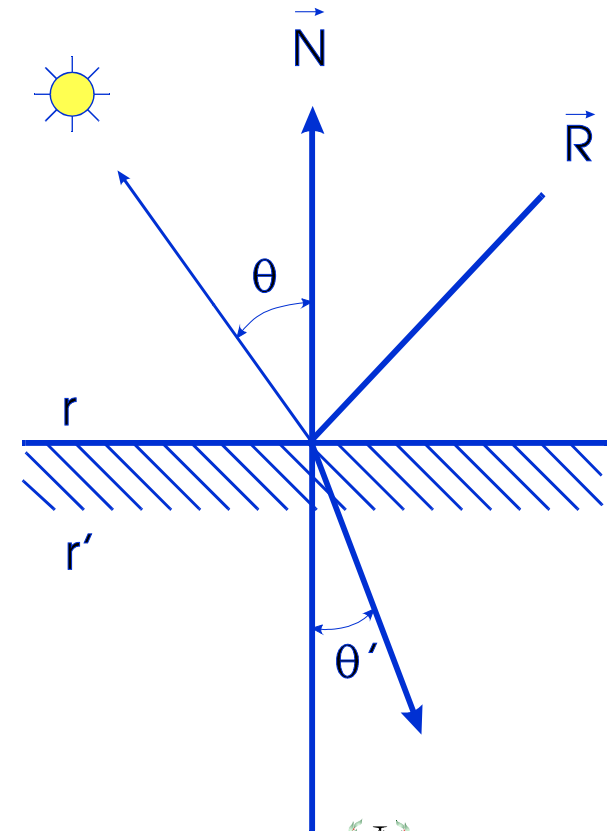
Objets transparents : inclure les rayons transmis en provenance de sources situées en arrière.

- La réfraction diffuse est difficile à modéliser à cause de la dispersion du rayonnement.
- La réfraction spéculaire est obtenue en calculant la variation de l'angle des rayons lumineux avec la relation

$$r \sin \theta = r' \sin \theta'$$

où  $r$  et  $r'$  : indices de réfraction dans le milieu ambiant et dans le matériau respectivement.

- Dans la réalité, des valeurs de  $r$  et  $r'$  varient avec la longueur d'onde mais on peut les supposer constants.



## Réfraction spéculaire

- Dans la pratique, ce type de calculs s'avèrent complexes et on utilise quelques simplifications. Par exemple, on projette l'intensité  $I_f$  des objets masqués sur la partie visible de l'objet transparent.
- L'intensité résultante: moyenne pondérée des deux valeurs

$$I = t I_t + (1 - t) I_f$$

où  $I_t$  est l'intensité de l'objet transparent telle que calculée par le modèle élaboré précédemment.

- Le coefficient  $t$  :
  - 0 pour des objets très transparents
  - 1 pour des objets opaques.
- Cette simplification, la fusion de couleurs, donne des résultats acceptables pour des objets non courbes et minces.



## Lumière (monochrome) réfléchiée en un point

- En combinant réflexions diffuses et spéculaires, de source ambiante et sources ponctuelles, on obtient l'équation du modèle de Phong:

$$I = k_a I_a + \sum_{1 \leq j \leq m} f_{att_j} I_{p_j} \left( k_d \cos \theta_j + k_s \cos^n \phi_j \right)$$

- ( La librairie OpenGL ajoute aussi un terme ambient pour chaque source lumineuse dans la sommation. )
- Généralement, ces termes varient d'un point à l'autre, mais certaines simplifications sont possibles afin de réduire le nombre des calculs. Par exemple,  $\phi$  est approximativement constant pour un observateur éloigné, ainsi que  $\theta$  pour une source éloignée.



## Lumière (couleur) réfléchie en un point

- Il suffit de réécrire cette équation pour chaque composante rouge, vert et bleu avec les coefficients propres à chaque couleur. Les trois valeurs sont alors utilisées pour contrôler les intensités des canons du moniteur. Une formulation (encore un peu incomplète) est donc:

$$\begin{bmatrix} I_r \\ I_v \\ I_b \end{bmatrix} = \begin{bmatrix} k_{ar} \\ k_{av} \\ k_{ab} \end{bmatrix} \begin{bmatrix} I_{ar} \\ I_{av} \\ I_{ab} \end{bmatrix} + \sum_{1 \leq j \leq m} f_{att_j} \begin{bmatrix} I_{pr} \\ I_{pv} \\ I_{pb} \end{bmatrix} \left( \begin{bmatrix} k_{dr} \\ k_{dv} \\ k_{db} \end{bmatrix} \cos(\theta_j) + \begin{bmatrix} k_{sr} \\ k_{sv} \\ k_{sb} \end{bmatrix} \cos^n(\phi_j) \right)$$

( la formule complète utilisée par OpenGL est sur la page suivante )

## Éclairage selon OpenGL

$$\begin{aligned}
 \text{Couleur}_{\text{Sommet}} = & \text{Émission}_{\text{Matériau}} + \text{Ambiante}_{\text{ModèleLumière}} * \text{Ambiante}_{\text{Matériau}} + \\
 & \sum_{i=0}^{n-1} \left( \frac{1}{k_c + k_l d + k_q d^2} \right) * (\text{EffetSpot}) * \\
 & [ \text{Ambiante}_{\text{Lumière}} * \text{Ambiante}_{\text{Matériau}} + \\
 & \max\{L \cdot n, 0\} * \text{Diffuse}_{\text{Lumière}} * \text{Diffuse}_{\text{Matériau}} + \\
 & \max\{S \cdot n, 0\}^n * \text{Spéculaire}_{\text{Lumière}} * \text{Spéculaire}_{\text{Matériau}} ]
 \end{aligned}$$

$$\text{EffetSpot} = \begin{cases} 1 & \text{si la lumière n'est pas un projecteur (GL\_CUT\_OFF = 180^\circ)} \\ 0 & \text{si la lumière est un projecteur, mais le sommet n'est} \\ & \text{pas à l'intérieur du cône} \\ (\max\{v \cdot d, 0\})^{\text{GL\_SPOT\_EXPONENT}} & \text{si le sommet est à l'intérieur du cône} \end{cases}$$

où  $v = (v_x, v_y, v_z)$  vecteur unitaire de la lumière (GL\_POSITION) au sommet

$d = (d_x, d_y, d_z)$  vecteur direction du projecteur

Un sommet est à l'intérieur du cône

$$\text{si } (\max\{v \cdot d, 0\}) < \cos(\text{GL\_SPOT\_CUTOFF})$$

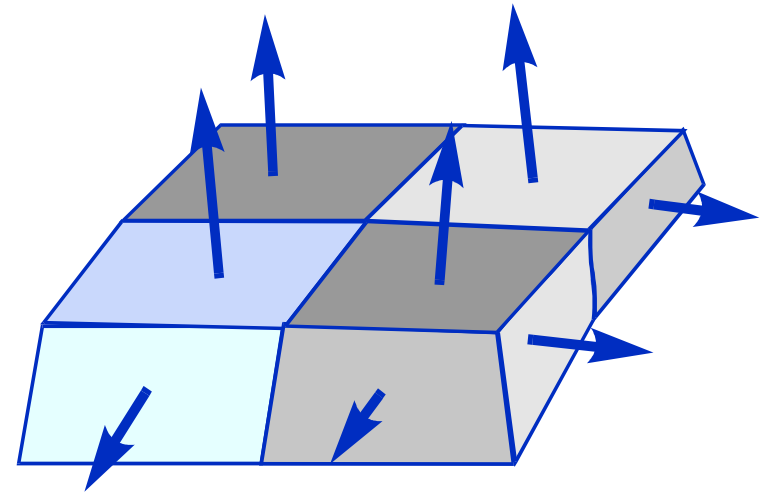


# Les modèles d'illumination



## Modèles d'illumination: Modèle de Lambert ("flat")

- Un objet constitué de faces planes: colorié avec une **intensité constante par face**. (*1 normale par face*)
- Une bonne représentation est obtenue si la source de lumière est éloignée ainsi que l'observateur.
- On peut utiliser cette approche pour des objets avec des surfaces gauches en approximant à l'aide de facettes ou polygones et en appliquant une intensité constante sur chaque facette.
- Si le nombre de facettes est grand, alors le résultat est bon avec, cependant, des discontinuités entre deux faces.



```
glShadeModel( GL_FLAT);
```

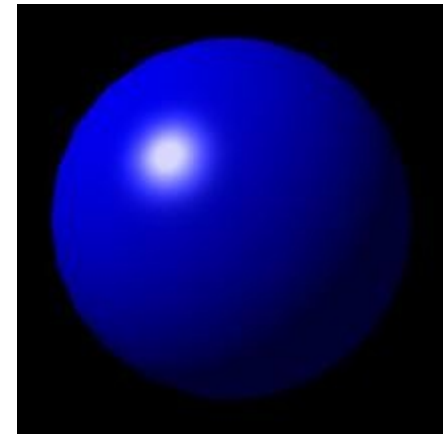


## Modèles d'illumination: Modèle de Gouraud ("smooth")

- **Interpolation de l'intensité (RGBA) linéairement sur chaque face** de manière à obtenir un raccord continu aux arêtes.
  - On part des normales définies aux sommets. (*1 normale par sommet*)
  - On calcule des intensités par les différents modèles de lumière en utilisant les normales calculées aux sommets.
  - L'intensité en un point d'une face est obtenue par interpolation linéaire des intensités des sommets.
  - Pour une arête, on interpole linéairement. Les calculs sont réalisés pour chaque pixel long d'une ligne de balayage.
  - L'équation d'un plan sous forme paramétrique

$$P = P_1 + \alpha (P_2 - P_1) + \beta (P_3 - P_1)$$

```
glShadeModel( GL_SMOOTH);
```



## Modèles d'illumination: Modèle de Gouraud

- Si  $0 \leq \alpha + \beta \leq 1$ ,  $\alpha \geq 0$  et  $\beta \geq 0$ , alors le lieu géométrique de  $P$  est la facette triangulaire avec les sommets (fragments)  $P_1$ ,  $P_2$  et  $P_3$ . Les coordonnées s'écrivent :

$$x = x_1 + \alpha(x_2 - x_1) + \beta(x_3 - x_1)$$

$$y = y_1 + \alpha(y_2 - y_1) + \beta(y_3 - y_1)$$

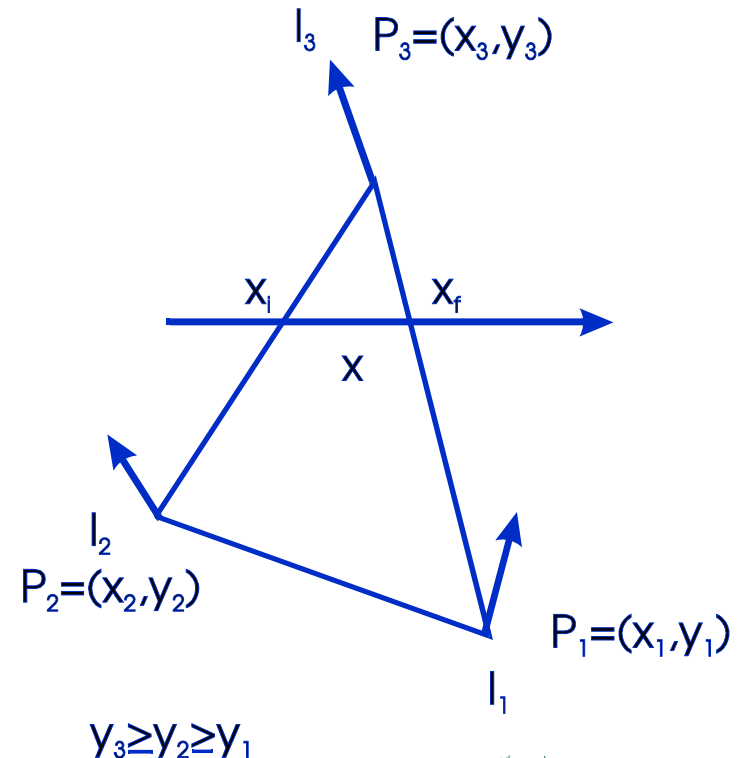
- En résolvant le système pour un point donné  $P = (x, y)$ , on obtient les valeurs de  $\alpha$  et  $\beta$

$$\alpha = \frac{(x - x_1)(y_3 - y_1) - (y - y_1)(x_3 - x_1)}{(x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1)}$$

$$\beta = \frac{(y - y_1)(x_2 - x_1) - (x - x_1)(y_2 - y_1)}{(y_3 - y_1)(x_2 - x_1) - (x_3 - x_1)(y_2 - y_1)}$$

- On peut alors interpoler les intensités aux faces

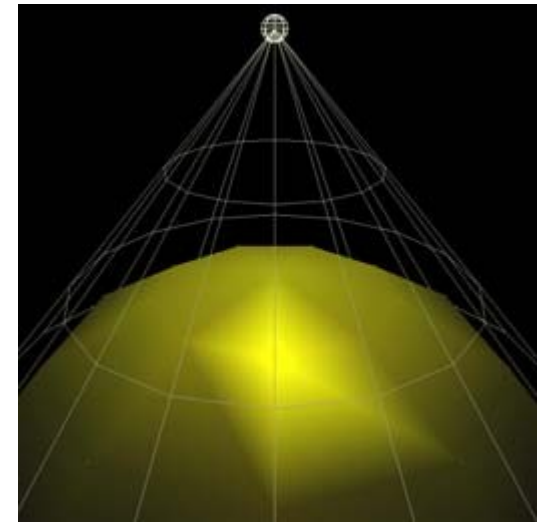
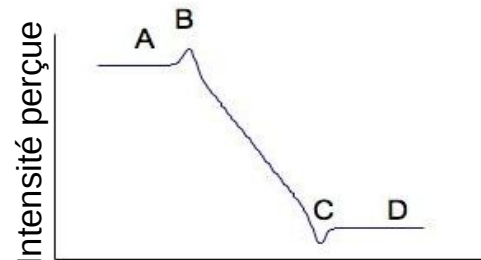
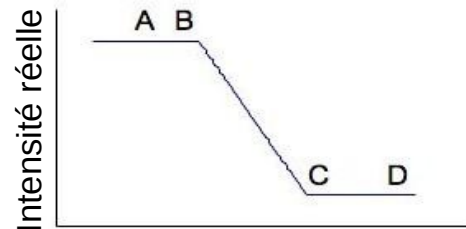
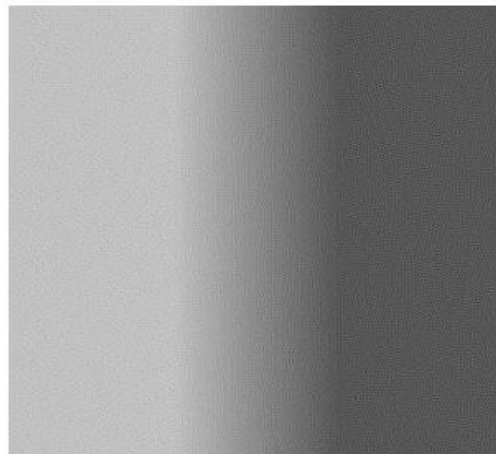
$$I(\alpha, \beta) = I_1 + \alpha(I_2 - I_1) + \beta(I_3 - I_1)$$



# Modèles d'illumination: Modèle de Gouraud

Une faiblesse: effet de Mach

- Exagération des changements d'intensité aux frontières où sont présentes une discontinuité en intensité ou une discontinuité des normales.
- Certaines arêtes apparaissent alors trop ou pas assez brillantes.
- Phénomène découvert par le physicien autrichien [Ernst Mach](#) (1838 – 1916).
- On peut corriger avec une discrétisation plus fine, mais ... plus de facettes! :(

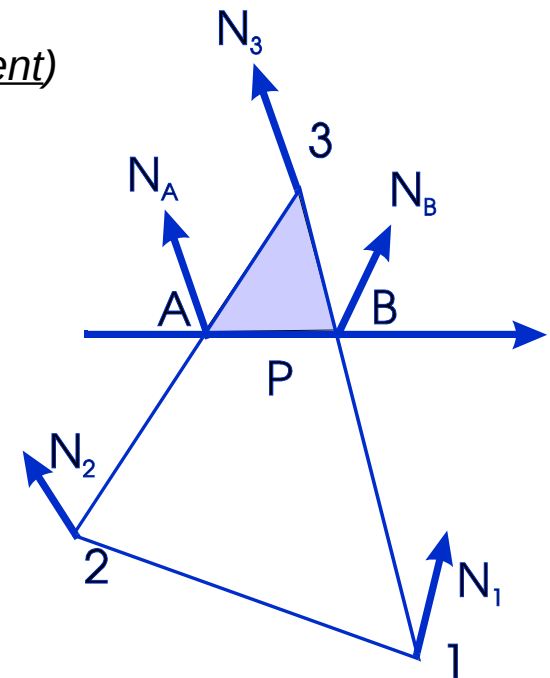
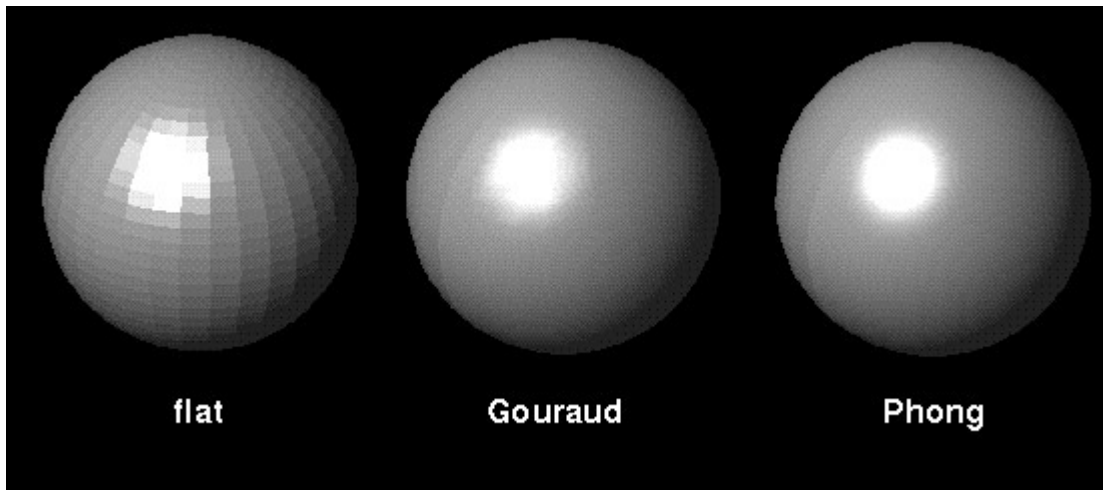


## Modèles d'illumination: Modèle de Phong

● Modèle de Phong: **interpolation des normales** plutôt que des intensités.

- A cause des relations non-linéaires entre les normales et l'intensité dans les modèles de lumière, on obtient un résultat nettement meilleur.
- L'illumination en un point recalculée à chaque pixel à partir de la normale interpolée. (*1 normale par fragment*)
- Méthode plus coûteuse.

•*Nuanceurs de fragments*



# Illumination en OpenGL

## Illumination en OpenGL

- Créer, positionner, activer une ou plusieurs sources de lumière (au moins 8 sources sont possibles)

```
void glLight{if}( GLenum light, GLenum pname, TYPE param );  
void glLight{if}v( GLenum light, GLenum pname, TYPE *param );  
void glEnable( GL_LIGHTx );  
void glDisable( GL_LIGHTx );
```

- Définir la lumière ambiante globale (et choisir modèle d'illumination)

```
glLightModel{if}( GLenum pname, TYPE param );  
glLightModel{if}v( GLenum pname, TYPE *param );  
glShadeModel( GLenum mode );
```

- Définir les propriétés des matériaux

```
glMaterial{if}( GLenum face, GLenum pname, TYPE param );  
glMaterial{if}v( GLenum face, GLenum pname, TYPE *param );
```

- Définir les vecteurs normaux pour les sommets

- définis automatiquement pour certaines surfaces (**glut** & **glu**)

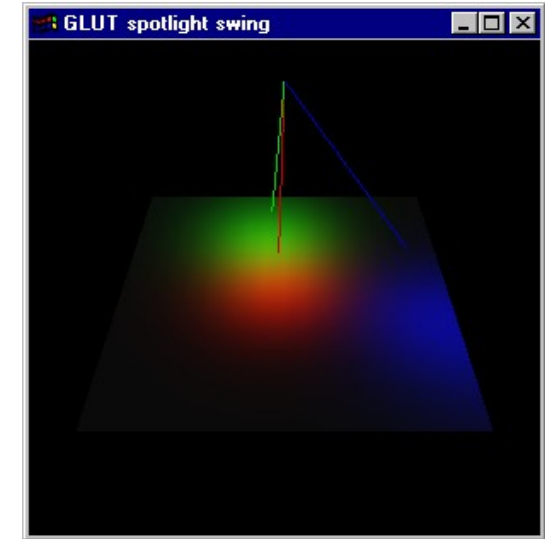
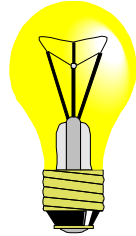
```
void glNormal3{bsidf}( TYPE nx, TYPE ny, TYPE nz );  
void glNormal3{bsidf}v( const TYPE *v );
```



# CRÉER LES SOURCES DE LUMIÈRE

## ● Propriétés

- type ( ambiante, diffuse, spéculaire, projecteur )
- couleur
- position
- direction
- etc.....



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

light     GL\_LIGHT0, ..... GL\_LIGHT7  
pname    identification du paramètre  
param    paramètre



# CRÉER LES SOURCES DE LUMIÈRE

Identification du paramètre	Valeur par défaut	Signification
GL_AMBIENT	(0.0, 0.0, 0.0, 1.0)	Intensité des composantes RGBA de la lumière ambiante
GL_DIFFUSE	(1.0, 1.0, 1.0, 1.0)	Intensité des composantes RGBA de la lumière diffuse
GL_SPECULAR	(1.0, 1.0, 1.0, 1.0)	Intensité des composantes RGBA de la lumière spéculaire
GL_POSITION	(0.0, 0.0, 1.0, 1.0) (0.0, 0.0, 1.0, 0.0) pointe le long des z négatifs	Position de la lumière (x, y, z, h) Si h=0 → source directionnelle, dir. = (x, y, z) Si h<>0 → source positionnelle, pos. = (x, y, z)
GL_SPOT_DIRECTION	( 0.0, 0.0, -1.0)	Direction du spot (x, y, z)
GL_SPOT_EXPONENT	( 0.0 )	Exposant pour le calcul du spot
GL_SPOT_CUTOFF	180°	Angle maximum pour un spot
GL_CONSTANT_ATTENUATION	1.0	Facteur constant pour l'atténuation ( $k_c$ )
GL_LINEAR_ATTENUATION	0.0	Facteur linéaire pour l'atténuation ( $k_l d$ )
GL_QUADRATIC_ATTENUATION	0.0	Facteur quadrique pour l'atténuation ( $k_q d^2$ )



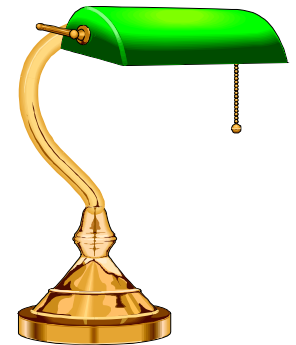
# CRÉER LES SOURCES DE LUMIÈRE

## ● Exemple

```
GLfloat lum_ambiante[ ] = { 0.0, 0.0, 0.0, 1.0 };  
GLfloat lum_diffuse[ ] = { 1.0, 1.0, 1.0, 1.0 };  
GLfloat lum_speculaire[ ] = { 1.0, 1.0, 1.0, 1.0 };  
GLfloat lum_position[ ] = { 0.1, -2.0, 2.5, 1.0 };
```

```
glLightfv( GL_LIGHT0, GL_AMBIENT, lum_ambiante );  
glLightfv( GL_LIGHT0, GL_DIFFUSE, lum_diffuse );  
glLightfv( GL_LIGHT0, GL_SPECULAR, lum_speculaire );  
glLightfv( GL_LIGHT0, GL_POSITION, lum_position );
```

```
glEnable( GL_LIGHT0 ); // Allumer la lumière  
glDisable( GL_LIGHT0 ); // Éteindre la lumière
```



# CRÉER LES SOURCES DE LUMIÈRE

## ● Spécifier la position

```
GLfloat lum_position[ ] = { 1.0, 1.0, 1.0, 0.0 };  
glLightfv( GL_LIGHT0, GL_POSITION, lum_position );
```

$$\text{Facteur d'atténuation} = \frac{1}{k_c + k_l d + k_q d^2}$$

$d$  = distance entre la lumière et le sommet

$k_c$  = GL\_CONSTANT\_ATTENUATION (1.0)

$k_l$  = GL\_LINEAR\_ATTENUATION (0.0)

$k_q$  = GL\_QUADRATIC\_ATTENUATION (0.0)

## ● Spécifier le facteur d'atténuation

```
glLightf( GL_LIGHT0, GL_CONSTANT_ATTENUATION, 2.0 );  
glLightf( GL_LIGHT0, GL_LINEAR_ATTENUATION, 1.0 );  
glLightf( GL_LIGHT0, GL_QUADRATIC_ATTENUATION, 0.5 );
```



# MODÈLE D'ILLUMINATION

## Composantes du modèle d'éclairage

### ■ Choix du modèle d'illumination

```
glShadeModel( GLenum mode );
```

où *mode* est GL\_FLAT ou GL\_SMOOTH (par défaut)

- GL\_FLAT : Modèle de Lambert (intensité constante par face, 1 normale par face)
- GL\_SMOOTH : Modèle de Gouraud (intensité interpolée linéairement sur chaque face, 1 normale par sommet)
- doit être en nuanceur : Modèle de Phong (interpolation des normales plutôt que des intensités, 1 normale par fragment)

# MODÈLE D'ILLUMINATION

## Composantes du modèle d'éclairage

- Intensité de la lumière ambiante globale
- Spécifier si le point de vue est local à la scène (calculs plus complexes) ou à l'infini (calculs moins complexes) car  $\cos\phi = \text{cte}$
- Éclairage des faces avant et arrière

```
glLightModel{if} ( GLenum pname, TYPE param );
```

```
glLightModel{if}v( GLenum pname, TYPE *param );
```

Paramètre	Valeur par défaut	Signification
GL_LIGHT_MODEL_AMBIENT	(0.2, 0.2, 0.2, 1.0)	Intensité de la lumière ambiante pour toute la scène
GL_LIGHT_MODEL_LOCAL_VIEWER	0.0 ou GL_FALSE	Spécifie comment les angles pour la réflexion spéculaire seront calculés.
GL_LIGHT_MODEL_TWO_SIDE	0.0 ou GL_FALSE	Choisir entre l'éclairage d'un seul côté ou des deux côtés

# SPÉCIFIER LES CARACTÉRISTIQUES DU MATÉRIAU

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

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

**face** spécifie que les propriétés de matériau s'applique à la face avant, arrière ou les deux: GL\_FRONT, GL\_BACK, ou GL\_FRONT\_AND\_BACK.

Identification du paramètre	Valeur par défaut	Signification
GL_AMBIENT	(0.2, 0.2, 0.2, 1.0)	Intensité de la lumière ambiante du matériau
GL_DIFFUSE	(1.0, 1.0, 1.0, 1.0)	Intensité de la lumière diffuse du matériau
GL_AMBIENT_AND_DIFFUSE		Intensité des lumières ambiante et diffuse du matériau
GL_SPECULAR	(0.0, 0.0, 0.0, 1.0)	Intensité de la lumière spéculaire du matériau
GL_SHININESS	0.0	Exposant pour le calcul de la lumière spéculaire
GL_EMISSION	( 0.0, 0.0, 0.0, 1.0)	Couleur émise par le matériau
GL_COLOR_INDEXES	(0, 1, 1)	Index des couleurs ambiante, diffuse et spéculaire

## SPÉCIFIER LES CARACTÉRISTIQUES DU MATÉRIAU

Que se passe-t-il avec les glColor(...)?

- On peut continuer à utiliser glColor(...) si on active cette possibilité:

```
void glColorMaterial( GLenum face, GLenum mode );  
glEnable( GL_COLOR_MATERIAL );
```

*face* spécifie que les propriétés de matériau de la face avant, arrière ou les deux, seront contrôlées par la couleur courante:

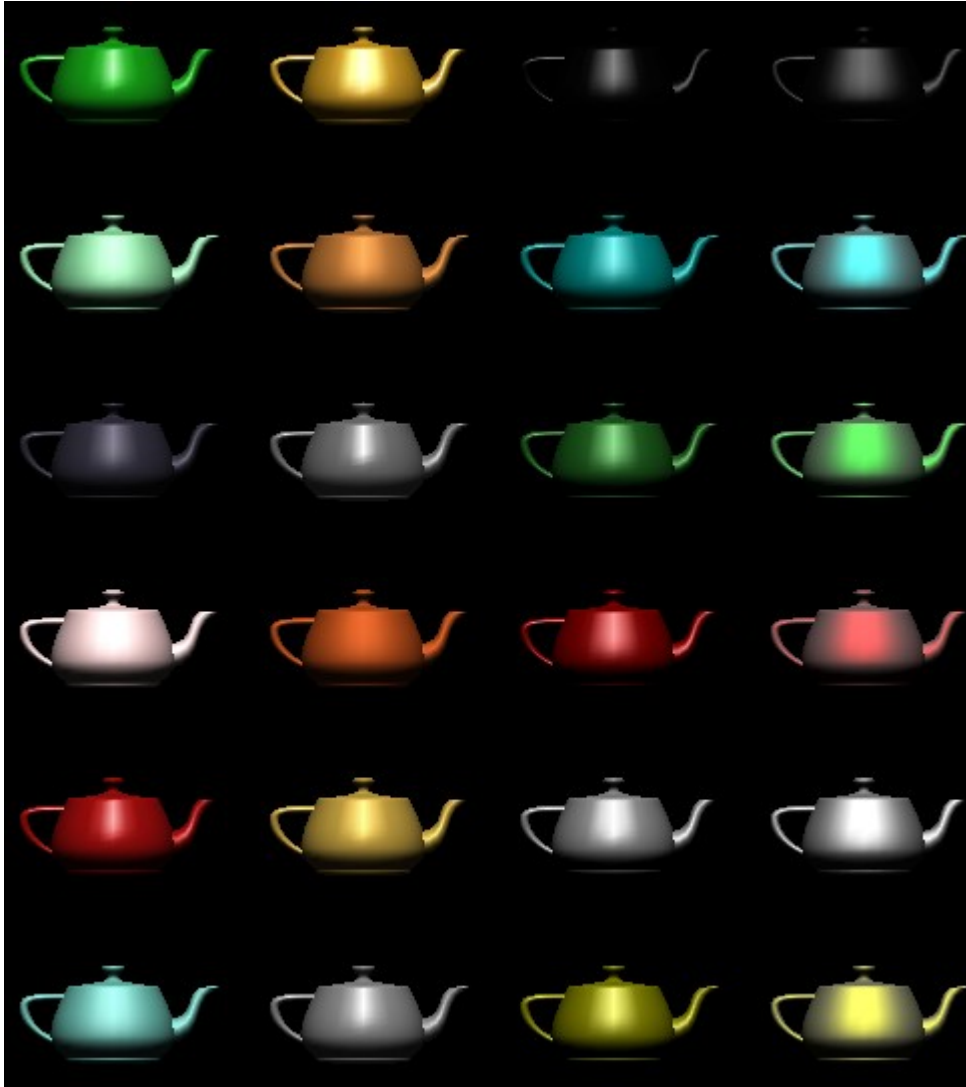
GL\_FRONT, GL\_BACK, GL\_FRONT\_AND\_BACK.

*mode* spécifie quelle partie du matériau utilise la couleur courante:

GL\_EMISSION, GL\_AMBIENT, GL\_DIFFUSE, GL\_SPECULAR,  
GL\_AMBIENT\_AND\_DIFFUSE.



## DIFFÉRENTES CARACTÉRISTIQUES DU MATÉRIAU



- Ces théières éclairées et dégradées accueillent différentes propriétés de matière qui suggèrent des matières existant dans la réalité.
- La première colonne simule (de haut en bas) l'émeraude, le jade, l'obsidienne, la perle, le rubis et le turquoise.
- Dans la deuxième colonne, ce sont le laiton, le bronze, le chrome, le cuivre, l'or et l'argent qui sont imités.
- La troisième produit un effet de plastique coloré : noir, cyan, vert, rouge, blanc et jaune.
- La quatrième colonne reprend les mêmes couleurs pour un effet caoutchouc.

## ACTIVATION DE L'ÉCLAIRAGE

- Allumer/éteindre les sources de lumières

```
glEnable( GL_LIGHTx );  
glDisable( GL_LIGHTx );
```

- Activer/désactiver l'éclairage

```
glEnable( GL_LIGHTING );  
glDisable( GL_LIGHTING );
```





# CONTRÔLE DE LA POSITION ET DIRECTION DE LA LUMIÈRE

- On utilise *glLight\*(...)* spécifier la source et la direction (position) de la lumière. La position ou direction est transformée par la matrice de modélisation courante.
- Position de la lumière
  - La position de la lumière demeure fixe
  - La lumière se déplace autour d'un objet stationnaire
  - La lumière se déplace avec le point de vue

## POSITION DE LA LUMIÈRE FIXE

- Il suffit de positionner la lumière et de **ne pas changer** la position
- Exemple:

```
// dans reshape
glViewport( 0, 0, w, h );
glMatrixMode( GL_PROJECTION );
glLoadIdentity();
if ( w <= h )
    glOrtho( -1.5, 1.5, -1.5 * h / w, 1.5 * h / w, -10.0, 10.0 );
else
    glOrtho( -1.5 * h / w, 1.5 * h / w, -1.5, 1.5, -10.0, 10.0 );
glMatrixMode( GL_MODELVIEW );
glLoadIdentity();
...
// Dans init()
GLfloat position[] = { 1.0, 1.0, 1.0, 1.0 }; // lumière positionn.
glLightfv( GL_LIGHT0, GL_POSITION, position );
```



# LA LUMIÈRE SE DÉPLACE AUTOUR D'UN OBJET STATIONNAIRE

- Déplacer et/ou tourner la lumière **après** la transformation de modélisation (probablement dans la fonction d'affichage)

- Exemple:

```
void Afficher ( void )
{
    GLfloat lum_position[] = {0.0, 0.0, 1.5, 1.0}; // lumière
    positionnelle
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
    glPushMatrix();
    GluLookAt( 0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 10.0, 1.0, 0.0 );
    glPushMatrix ();
    glRotatef( spin, AXE_X);
    glLightfv( GL_LIGHT0, GL_POSITION, lum_position );
    glPopMatrix();
    glutSolidTorus( 0.275, 0.85, 8, 15 );
    glPopMatrix();
}
```



## LA LUMIÈRE SE DÉPLACE SELON LE POINT DE VUE ( OBSERVATEUR)

- Il faut spécifier la position de la lumière **avant** la transformation de visualisation. La position de la lumière est en coordonnées de visualisation ( par exemple, (coordonnées de l'œil) (0, 0, 0) → lumière émanant de la lentille de la caméra ).

- Exemple:

```
void Initialisation ( void )
{
    GLfloat lum_position[] = {0.0, 0.0, 0.0, 1.0};

    glViewport( 0, 0, w, h );
    glMatrixMode( GL_PROJECTION );
    glLoadIdentity();
    glPerspective( 40.0, (Gldouble) w / (Gldouble) h, 1.0, 100.0);
    glMatrixMode( GL_MODELVIEW );
    glLoadIdentity();
    glLightfv( GL_LIGHT0, GL_POSITION, lum_position );
}
```



## LA LUMIÈRE SE DÉPLACE SELON LE POINT DE VUE ( OBSERVATEUR)

- Si la caméra est déplacée (point de vue), la lumière se déplace avec la caméra

- Exemple:

```
Camera_T Camera = { {0.0, 1.0, 0.0},  
                    {0.0, 0.0, 0.0},  
                    {0.0, 1.0, 0.0} };  
  
void Afficher ( void )  
{  
    glClear ( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );  
    glPushMatrix( );  
    gluLookAt( Camera.Observateur.x, Camera.Observateur.y,  
              Camera.Observateur.z,  
              Camera.PtVise.x, Camera.PtVise.y, Camera.PtVise.z,  
              Camera.VUP.x, Camera.VUP.y, Camera.VUP.z);  
    glutSolidTorus ( 0.275, 0.85, 8, 15);  
    glPopMatrix( );  
}
```



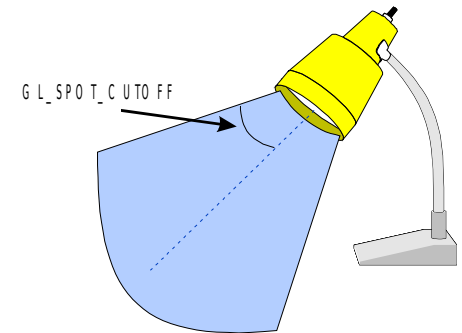
# CRÉER LES SOURCES DE LUMIÈRE

## Exemple

```
typedef struct
{
    GLfloat Constant;           // GL_CONSTANT_ATTENUATION
    GLfloat Linear;             // GL_LINEAR_ATTENUATION
    GLfloat Quadratic;          // GL_QUADRATIC_ATTENUATION
} glgAttenuation;

typedef struct
{
    Point3D Direction;          // GL_SPOT_DIRECTION
    GLfloat Exponent;           // GL_SPOT_EXPONENT
    GLfloat CutOff;             // GL_SPOT_CUTOFF
} glgSpot;

typedef struct
{
    glgRGBA Ambient;            // GL_AMBIENT
    glgRGBA Diffuse;            // GL_DIFFUSE
    glgRGBA Specular;           // GL_SPECULAR
    Point4D Position;           // GL_POSITION
    glgSpot Spot;               // GL_SPOT_DIRECTION,
                                // GL_SPOT_EXPONENT,
                                // GL_SPOT_CUTOFF
    glgAttenuation Attenuation;  // GL_CONSTANT_ATTENUATION
                                // GL_LINEAR_ATTENUATION,
                                // GL_QUADRATIC_ATTENUATION
} glgLightSource;
```



# CRÉER LES SOURCES DE LUMIÈRE

## Exemple

```
void ConstruireEclairage( void )
{
    glLightModel Model = { { .1, .1, .1, 1.0 }, GL_FALSE, GL_FALSE };
    glLightSource Lumiere = { { 0.0, 0.0, 0.0, 1.0 },           // Ambiante
                              { 1.0, 1.0, 1.0, 1.0 },         // Diffuse
                              { 1.0, 1.0, 1.0, 1.0 },         // Spéculaire
                              { 0.0, 0.0, 0.0, 0.0 },         // Direction
                              { {0.0, 0.0, 0.0}, 0.0, 0.0 },   // Spot
                              { 1.0, 0.0, 0.0 } };             // Atténuation
    glMaterialProperties Matériau = { { .2, .2, .2, 1.0 },      // Ambiante
                                      { 1.0, 1.0, 1.0, 1.0 }, // Diffuse
                                      { 0.2, 0.2, 0.2, 0.2, 1.0 }, // Spéculaire
                                      { 1000 },                 // Brilliance
                                      { 0., 0., 0., 0.0 },      // Emission
                                      { 0, 0, 0 } };             // Index
    glLightModelfv( GL_LIGHT_MODEL_AMBIENT, Model.Ambient );
    glLightfv( GL_LIGHT0, GL_AMBIENT, Lumiere.Ambient );
    glLightfv( GL_LIGHT0, GL_DIFFUSE, Lumiere.Diffuse );
    glLightfv( GL_LIGHT0, GL_SPECULAR, Lumiere.Specular );
    glEnable( GL_LIGHT0 );
    glMaterialfv( GL_FRONT_AND_BACK, GL_AMBIENT, Matériau.Ambient );
    glMaterialfv( GL_FRONT_AND_BACK, GL_DIFFUSE, Matériau.Diffuse );
    glMaterialfv( GL_FRONT_AND_BACK, GL_SPECULAR, Matériau.Specular );
    glMaterialfv( GL_FRONT_AND_BACK, GL_SHININESS, Matériau.Shininess );
    glEnable( GL_COLOR_MATERIAL );
}
```



## Éclairage selon OpenGL

$$\begin{aligned}
 \text{Couleur}_{\text{Sommet}} = & \text{Émission}_{\text{Matériau}} + \text{Ambiante}_{\text{ModèleLumière}} * \text{Ambiante}_{\text{Matériau}} + \\
 & \sum_{i=0}^{n-1} \left( \frac{1}{k_c + k_l d + k_q d^2} \right) * (\text{EffetSpot}) * \\
 & [ \text{Ambiante}_{\text{Lumière}} * \text{Ambiante}_{\text{Matériau}} + \\
 & \max\{L \cdot n, 0\} * \text{Diffuse}_{\text{Lumière}} * \text{Diffuse}_{\text{Matériau}} + \\
 & \max\{S \cdot n, 0\}^n * \text{Spéculaire}_{\text{Lumière}} * \text{Spéculaire}_{\text{Matériau}} ]
 \end{aligned}$$

$$\text{EffetSpot} = \begin{cases} 1 & \text{si la lumière n'est pas un projecteur (GL\_CUT\_OFF = 180^\circ)} \\ 0 & \text{si la lumière est un projecteur, mais le sommet n'est} \\ & \text{pas à l'intérieur du cône} \\ (\max\{v \cdot d, 0\})^{\text{GL\_SPOT\_EXPONENT}} & \text{si le sommet est à l'intérieur du cône} \end{cases}$$

où  $v = (v_x, v_y, v_z)$  vecteur unitaire de la lumière (GL\_POSITION) au sommet

$d = (d_x, d_y, d_z)$  vecteur direction du projecteur

Un sommet est à l'intérieur du cône

$$\text{si } (\max\{v \cdot d, 0\}) < \cos(\text{GL\_SPOT\_CUTOFF})$$

