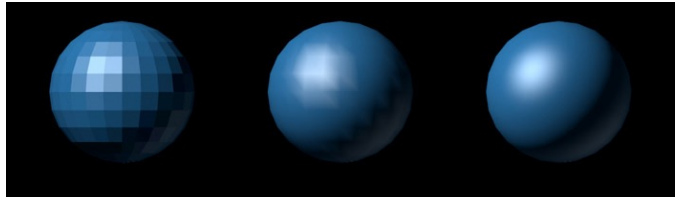
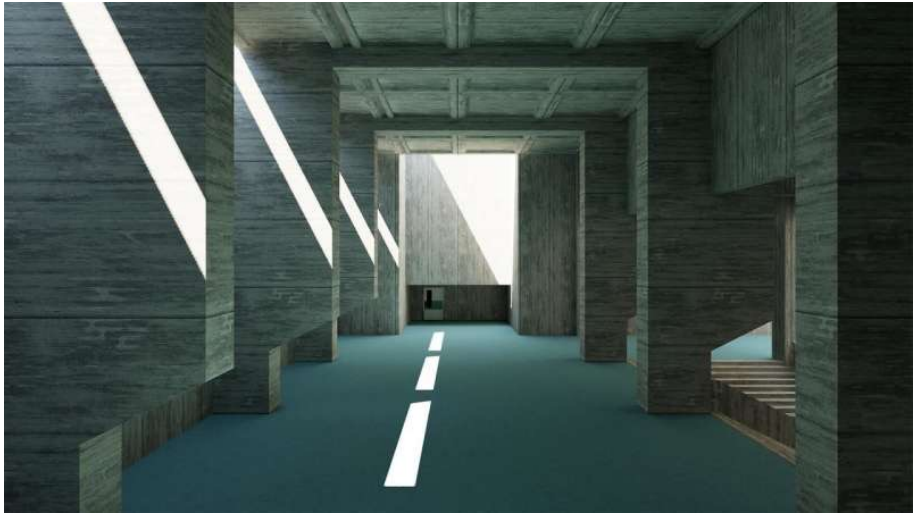
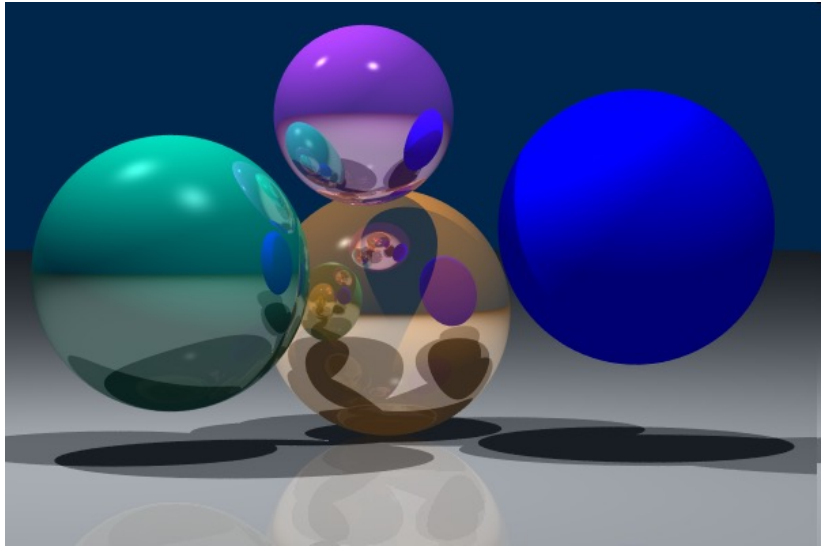
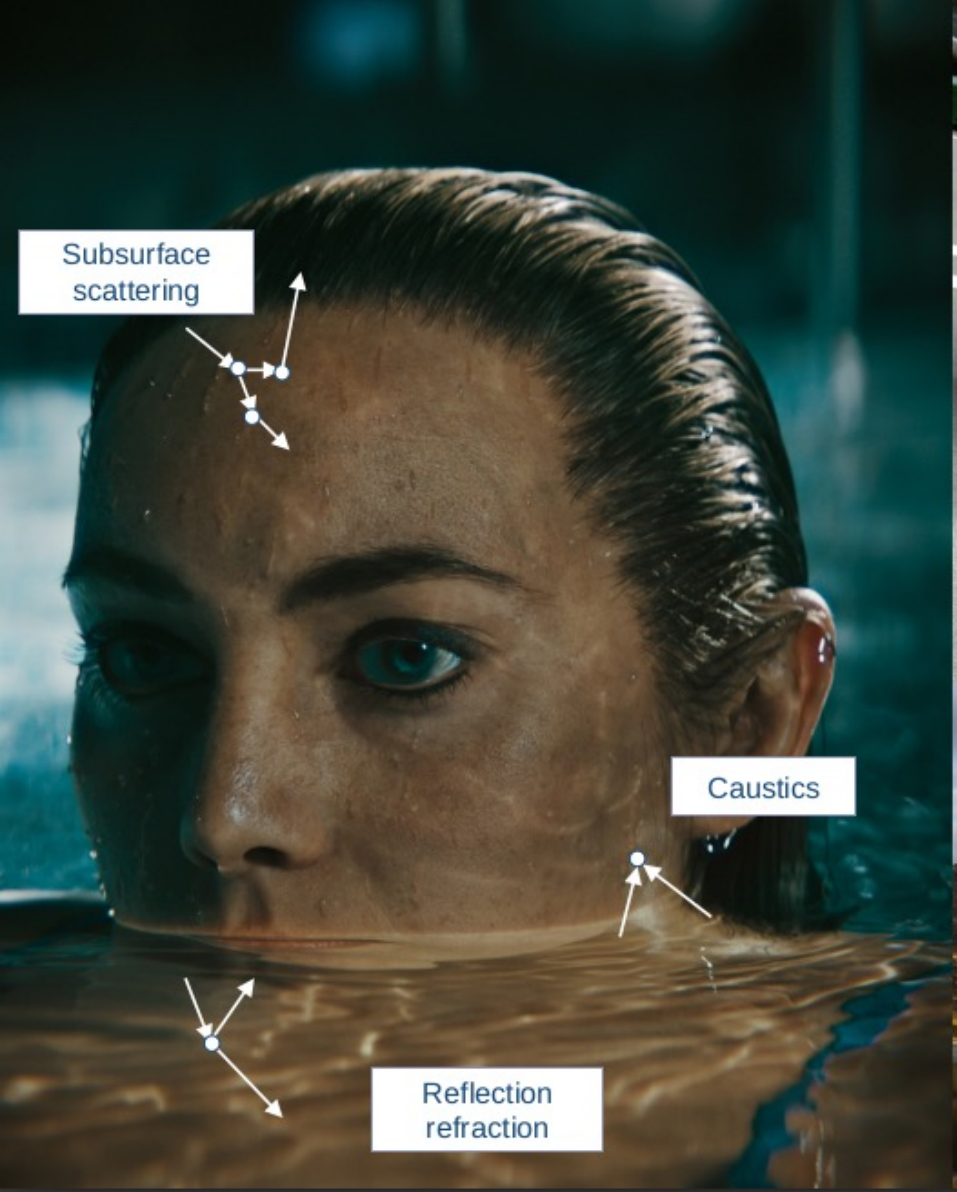
Illumination

Florence Zara (semestre automne)
LIRIS-ORIGAMI, Université Lyon 1







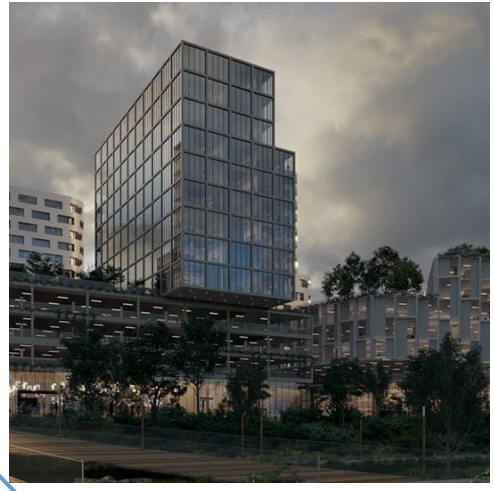

Subsurface scattering
Caustics
Reflection refraction


Halo in transparent coating


Scattering
Global illumination

Rendu

Rendu
Simulation de l'éclairement



Rendu réaliste
Simulation des phénomènes physiques complexes, interactions lumière-matière, équations globales

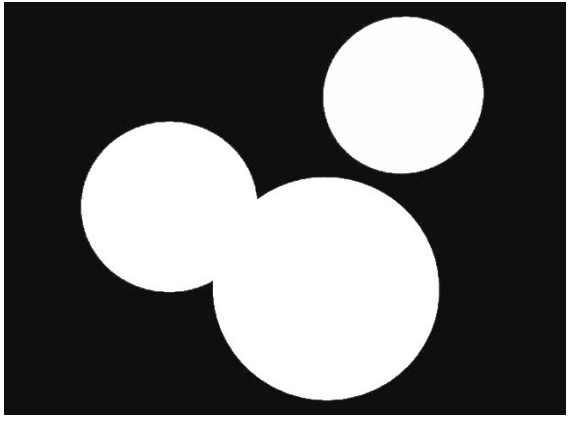
Films d'animation

Rendu temps réel
Simplifications extrêmes, équations locales

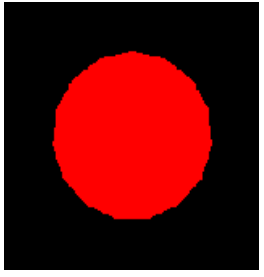
Jeux vidéos, simulateurs

Couleur / éclairage / illumination

Comment calculer la **couleur** et **l'éclairage** de chaque partie des objets ?



Objets sans couleur ni éclairage



Objet rouge sans éclairage

Objet rouge avec éclairage

Gestion de la couleur en OpenGL

- **Mode RGBA** (Rouge, Vert, Bleu, Alpha / transparence)
  - Valeur comprise entre 0.0 et 1.0
- **Tampon chromatique** :
  - Contient la dernière image dessinée
  - A vider en lui attribuant une couleur de fond / couleur de vidage spécifiée

`glClearColor()`: permet de spécifier la couleur de vidage
`glClear()` : vide le tampon spécifié en attribuant la couleur de vidage

`GL_COLOR_BUFFER_BIT` : tampon chromatique
`GL_DEPTH_BUFFER_BIT` : tampon de profondeur
`GL_STENCIL_BUFFER_BIT` : tampon stencil

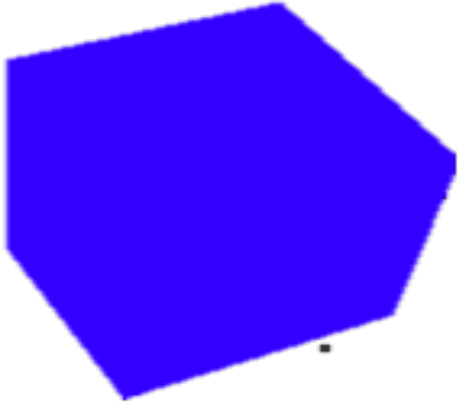
Ajout de la couleur - code gkit

m_cube.color (Color(0, 0, 1)) définit la couleur active

- Valeurs comprises entre 0.0 et 1.0

Exemple avec l'ajout de la couleur bleu à un cube

```
void Viewer::init_cube(){
      m_cube.color( Color(0.0, 0.0, 1.0) );
          …
      m_cube.vertex(...);
      m_cube.vertex(...);
      …}
```

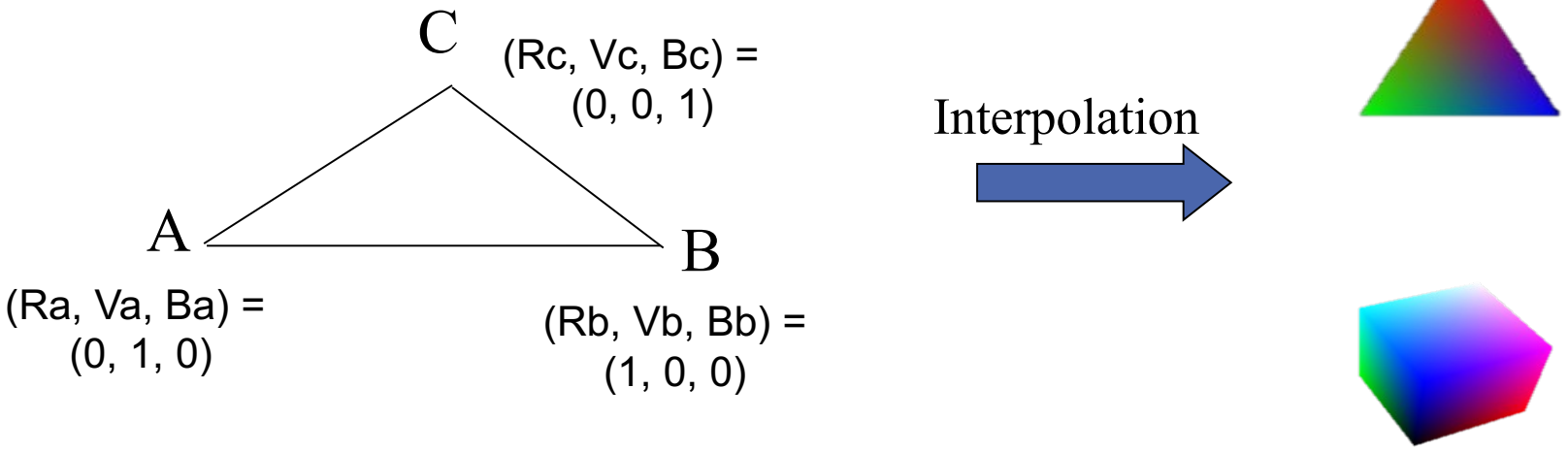


La **couleur est constante** pour l'ensemble du cube
(spécifiée une seule fois avant la création du cube)

1 couleur par sommet du maillage

La couleur peut être spécifiée pour chacun des sommets

Il faut le faire avant la création de chacun des sommets



Rappel - Interpolation linéaire – cas d'une droite

Soit une fonction f définie par une droite avec f : t → V(t)

On connait les valeurs de la fonction f en a et b ∈ R, avec f(a) = V_a et f(b) = V_b

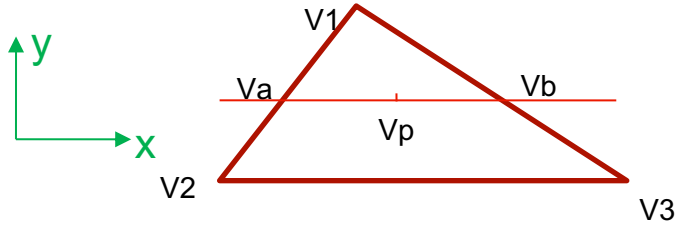
On recherche la valeur Vt en t avec t ∈ [a,b]

Pente de la droite : $\frac{V_b - V_a}{b - a} = m$ On a aussi : $m = \frac{Vt - Va}{t - a}$

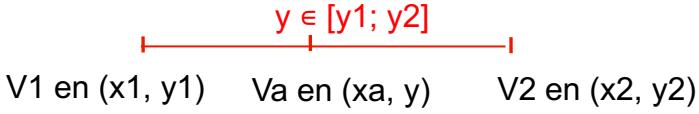
On en déduit : $Vt = m\,(t - a) + V_a = (V_b - V_a) + V_a$

Rappel - Interpolation bilinéaire sur un triangle

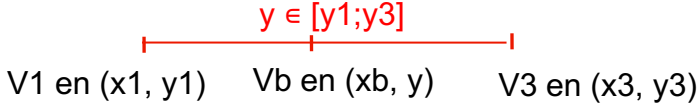
- V1 en (x1, y1)
- V2 en (x2, y2)
- V3 en (x3, y3)
- Vp en (x, y)
- Va en (xa, y)
- Vb en (xb, y)



- Interpolation linéaire pour xa selon valeurs des abscisses aux extrémités (V1 et V2) :
  - xa = x1 + (x2 - x1)

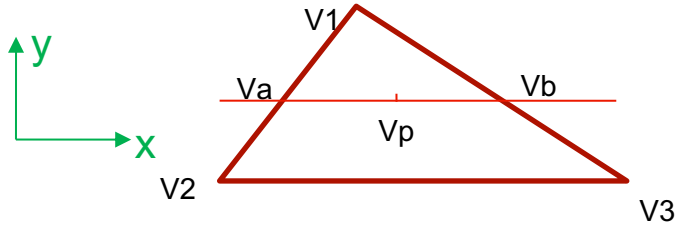


- Interpolation linéaire pour xb selon valeurs des abscisses aux extrémités (V1 et V3) :
  - xb = x1 + (x3 - x1)



Rappel - Interpolation bilinéaire sur un triangle

- V1 en (x1, y1)
- V2 en (x2, y2)
- V3 en (x3, y3)
- Vp en (x, y)
- Va en (xa, y)
- Vb en (xb, y)

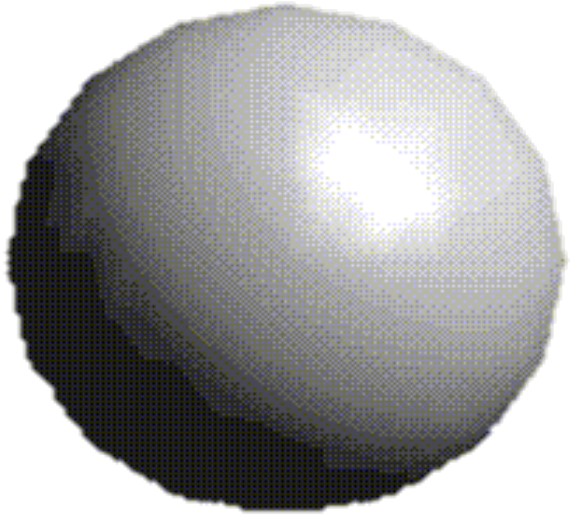


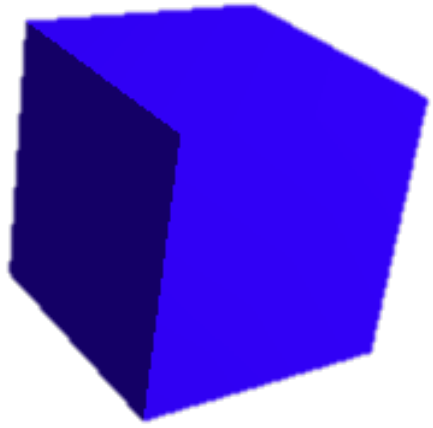
- Interpolation linéaire pour Va selon valeurs aux extrémités (V1 et V2):
  - Va = V1 + (V2 - V1) par rapport à l'ordonnée
- Interpolation linéaire pour Vb selon valeurs aux extrémités (V1 et V3) :
  - Vb = V1 + (V3 - V1) par rapport à l'ordonnée
- Interpolation linéaire pour Vp selon valeurs aux extrémités (Va et Vb) :
  - Vp = Va + (Vb - Va) par rapport à l'abscisse

1 couleur par sommet – code gkit

```
Mesh m_triangle; // Déclaré dans ViewerEtudiant.h

void ViewerEtudiant::init_triangle()
{
   m_triangle = Mesh(GL_TRIANGLES);

   m_triangle.color(Color(1, 0, 0));
   m_triangle.vertex(-0.5, -0.5, 0);

   m_triangle.color(Color(0, 1, 0));
   m_triangle.vertex(0.5, 0.5, 0);

   m_triangle.color(Color(0, 0, 1));
   m_triangle.vertex(-0.5, 0.5, 0);
}
```

Gestion de la lumière





On aimerait ajouter de l'éclairage (illumination)

Apport de la lumière

Selon la lumière, les objets réfléchissent ou n'absorbent pas la même couleur

- Ils apparaissent ainsi différents

(la mer n'a pas la même couleur qd il fait beau ou qd il pleut)

Couleur de la lumière déterminée par composantes rouge, vert, bleu émises

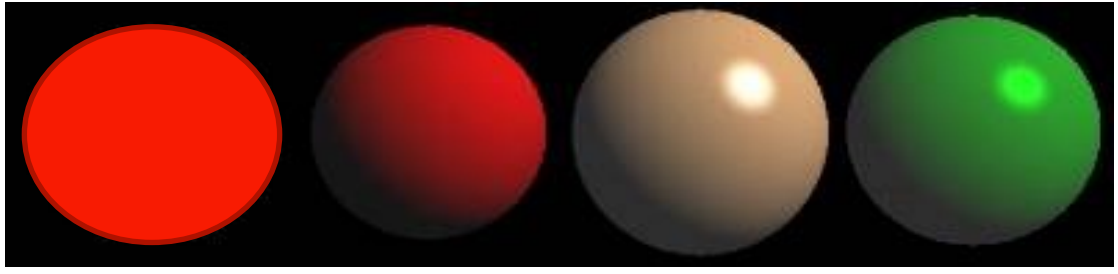
(3 longueurs d'ondes)

Propriétés du matériel permettent de déterminer les composantes réfléchies et ainsi la couleur de l'objet

- OpenGL : éclairage dépend des caractéristiques de la **lumière** et des **matériaux**

Composantes de la lumière

- Ambiante : colore les pixels d'un objet par la même couleur qq soit environnement lumineux
- Diffuse (différente pour chaque sommet) : donne effet 3D et lissé aux objets
- Spéculaire : correspond au reflet de la lumière sur les bords des objets
- Emissive : lumière émise par un objet

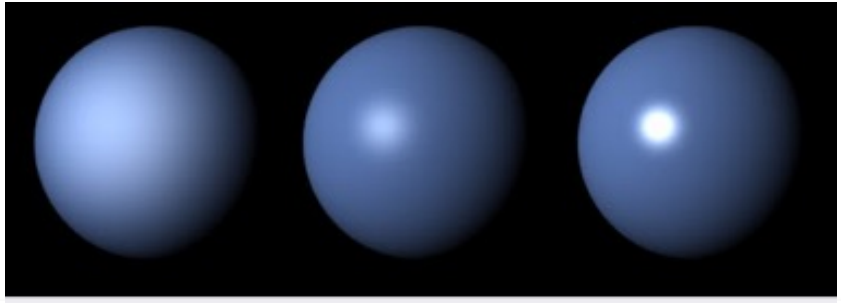


Composantes de la lumière

- Ambiante : représente les parasites provenant d'autres éléments que la source considérée (p. ex. lumière réfléchie par d'autres points).

- Diffuse : indique l'intensité qui repart. Tient compte de l'inclinaison avec laquelle la lumière incidente arrive sur la surface. Mais suppose que l'intensité est la même quelque soit la direction que prend le rayon réfléchi.

- Spéculaire : permet de prendre en compte le fait qu'il y a plus de lumière renvoyée dans la direction de réflexion.



Illumination

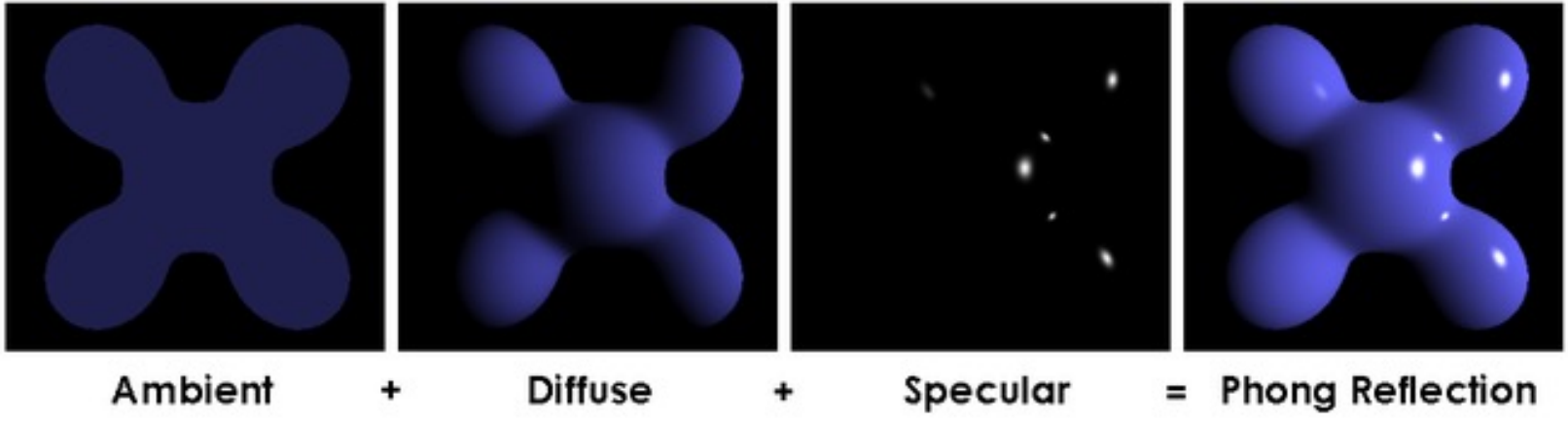
Objectif : déterminer la couleur d'un point de l'objet en tenant compte de la lumière, éclairage de la scène

Solution : calculer comment ce point est illuminé

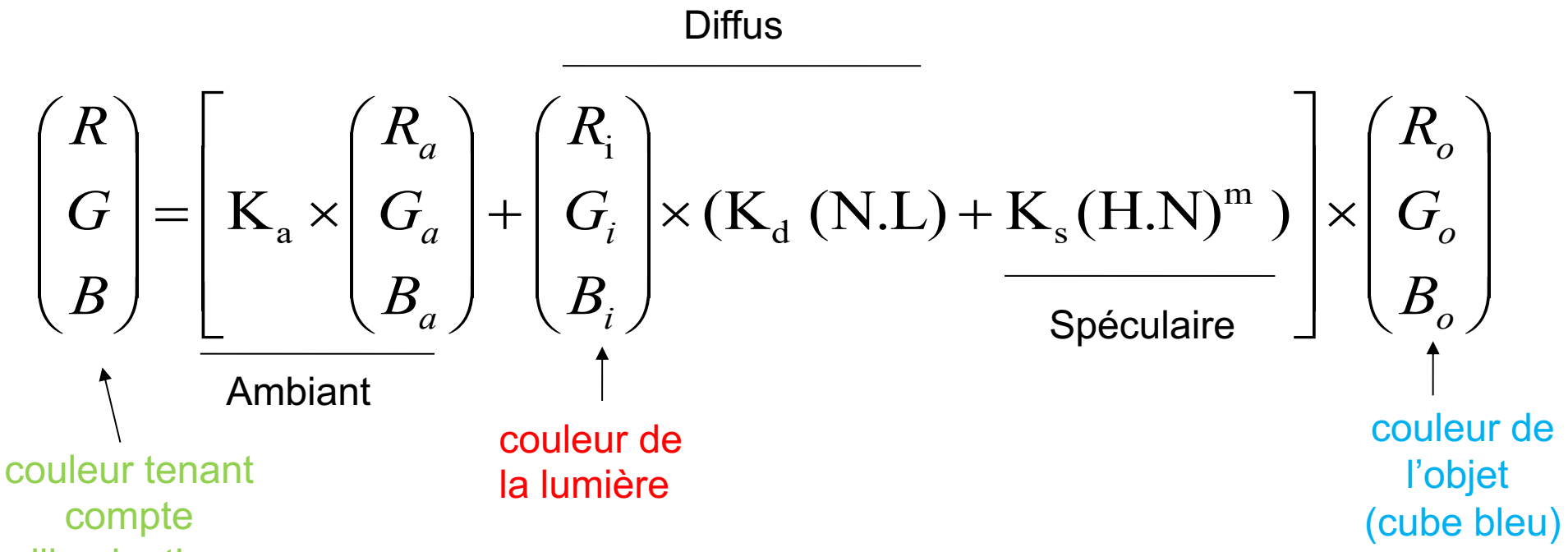
Modèle d'illumination de Phong

Modèle de Phong

- Modèle local : calcul effectué en chaque point
- Empirique : plausible mais loin d'une simulation physique
- Calcule la **lumière réfléchie** par le point étudié
- Combine 3 éléments : ambiant + diffus + spéculaire



Modèle d'illumination de Phong

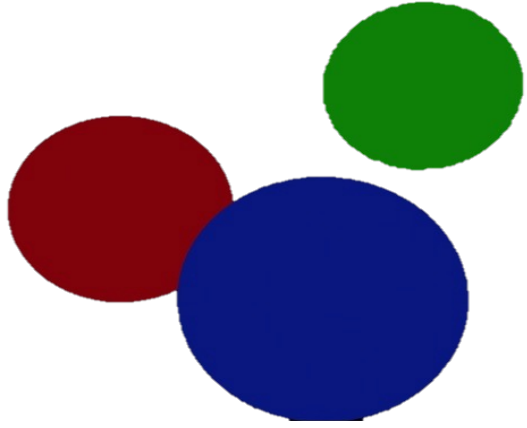


- K : proportion de lumière réfléchie *vs* absorbée par le matériel

Lumière ambiante (*Ambient Light*)

Approximation d'illumination globale

- Même sans lumière directe, un objet est éclairé
- Correspond à la lumière réfléchie par d'autres points
- **Eclairage est constant sur tout l'objet**
- On ne voit que la silhouette des objets



Lumière ambiante (*Ambient Light*)

Lumière ambiante est définie pour toute la scène

- Intensité de la lumière ambiante : $\mathbf{I_a = (R_a, G_a, B_a)}$
- Par exemple la couleur du ciel

Chaque objet réfléchit seulement une portion de cette lumière

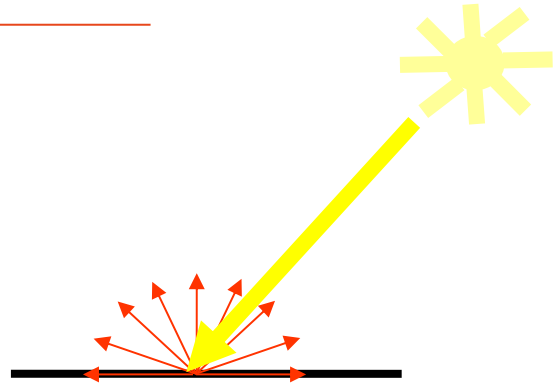
- K_a : proportion de la lumière ambiante réfléchie (valeur entre 0 et 1)
- Intensité de la lumière ambiante renvoyée par l'objet : $\mathbf{I = I_a K_a}$

Ambiant

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \left[\mathrm{K_a} \times \begin{pmatrix} R_a \\ G_a \\ B_a \end{pmatrix} + \begin{pmatrix} R_i \\ G_i \\ B_i \end{pmatrix} \times (\mathrm{K_d}\ (\mathrm{N.L}) + \mathrm{K_s}(\mathrm{H.N})^{\mathrm{m}}\ ) \right] \times \begin{pmatrix} R_o \\ G_o \\ B_o \end{pmatrix}$$

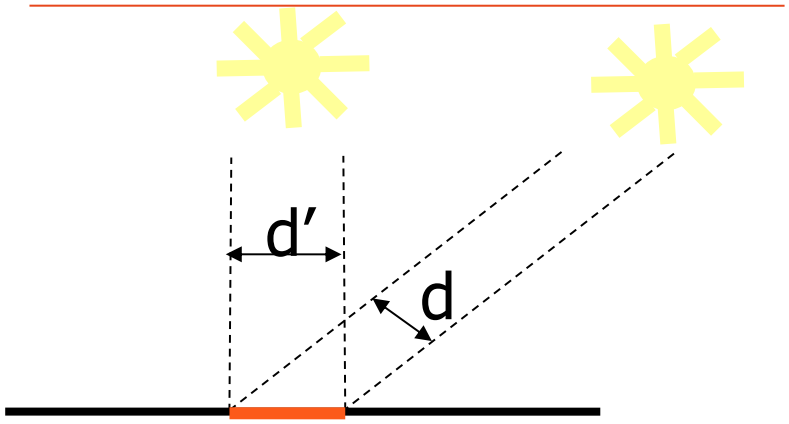
Diffusion de la lumière

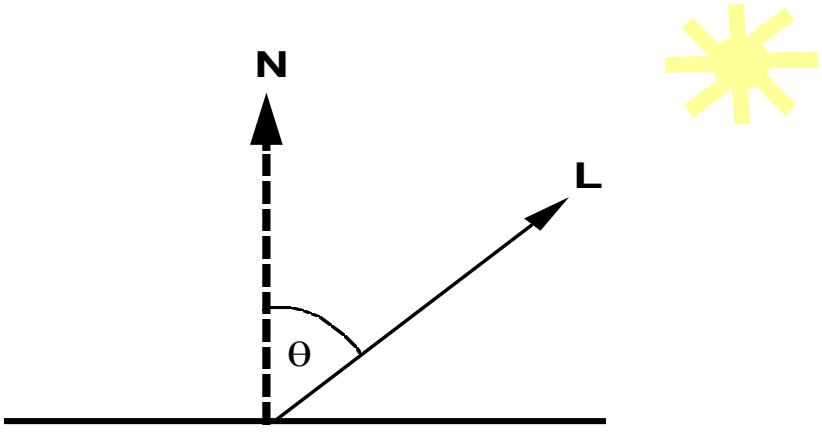
Loi de Lambert



- Lumière réfléchie de manière diffuse (réflexion diffuse)
- Diffusion uniforme dans toutes les directions
- **Diffusion dépend de l'angle d'arrivée de la lumière**
- Surface de ce type = surface Lambertienne

Loi de Lambert





- L = direction vers la lumière
- N = vecteur normal à la surface

  - Lumière incidente (arrivant sur la surface) est proportionnelle à d
  - d dépend de l'angle d'arrivée de la lumière
    - d est proportionnel à cos θ = N . L (produit scalaire)

  ⇨ Intensité réfléchie est proportionnelle à cos θ

Lumière diffuse (*Diffuse Light*)

La lumière est réfléchie selon la loi de Lambert

I_i : intensité de la lumière qui arrive

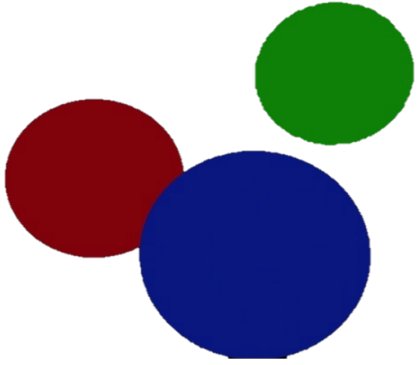
- énergie lumineuse par unité de surface

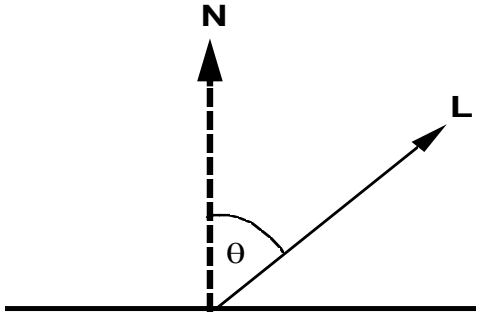
K_d : coefficient de diffusion de la surface

$\mathbf{I_d}$: intensité de la lumière diffusée (intensité qui repart) :

- Dépend de l'inclinaison de la lumière incidente sur la surface
- Suppose que l'intensité est la même qq soit direction rayon réfléchi

$$\mathbf{I_d = I_i\, K_d\ N \,.\, L = I_i\, K_d \cos \theta}$$





Lumière diffuse (*Diffuse Light*)

Diffus

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \left[\mathrm{K_a} \times \begin{pmatrix} R_a \\ G_a \\ B_a \end{pmatrix} + \begin{pmatrix} R_i \\ G_i \\ B_i \end{pmatrix} \times (\mathrm{K_d}\ (\mathrm{N.L}) + \mathrm{K_s}\,(\mathrm{H.N})^{\mathrm{m}}\ ) \right] \times \begin{pmatrix} R_o \\ G_o \\ B_o \end{pmatrix}$$

intensité de la lumière

Si on a plusieurs lumières ?

Additionne les termes diffus avec :

$$I_r = K_a I_a + \sum_{j=1}^{m} I_j K_d (N \cdot L_j)$$

- m : nombre de sources lumineuses
- I_j : intensité de la lumière incidente j
- L_j : vecteur indiquant la direction vers la lumière j

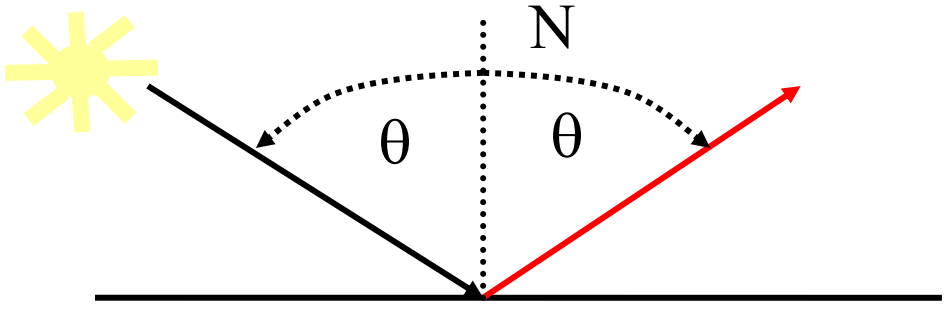
Réflexion spéculaire

Prise en compte du fait qu'il y a **plus de lumière renvoyée dans la direction de réflexion**

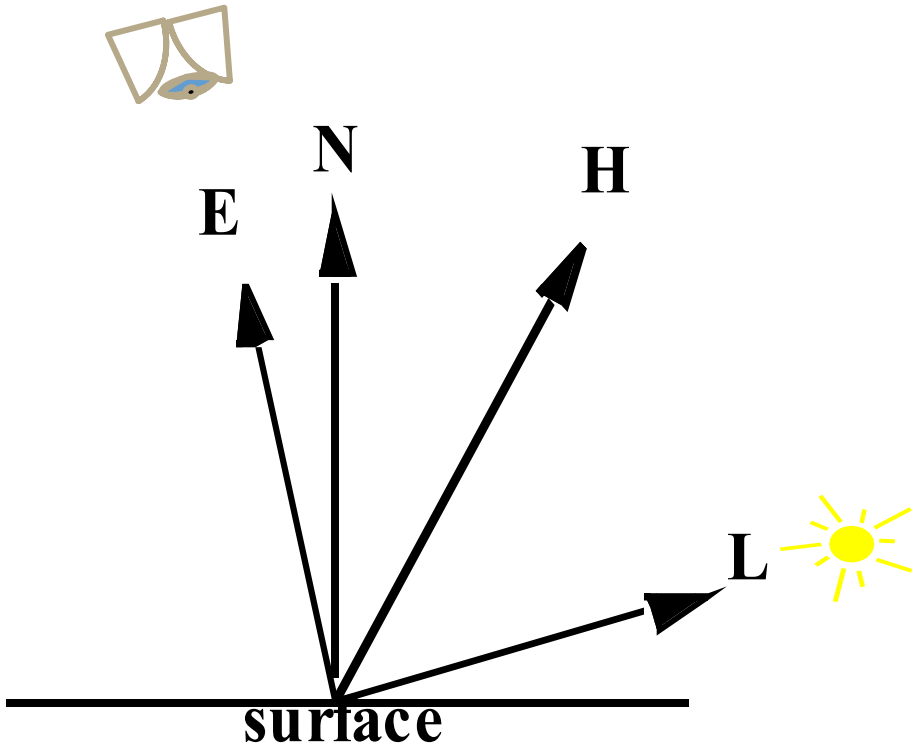
Apparaît sur les surfaces brillantes

Cas réflexion parfaite

- On ne verrait pratiquement jamais le reflet spéculaire !



Spéculaire « imparfait » (Phong)



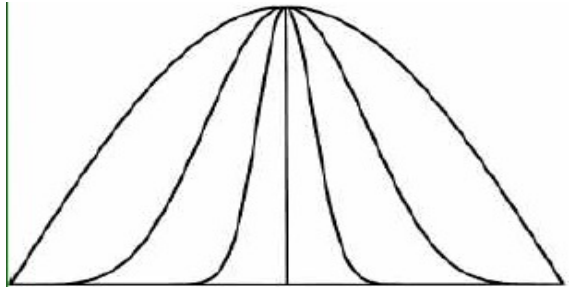
- E = direction de l'œil (Eye)
- N = normal
- L = direction de la lumière

- **H = (E+L) / 2**

Lumière spéculaire

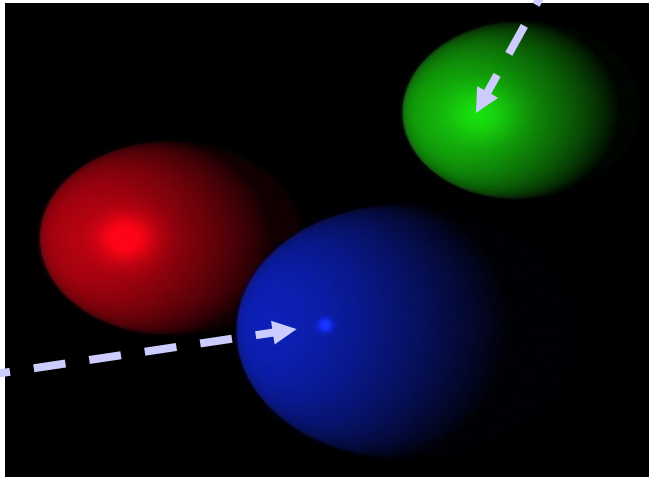
m = brillance (*shininess*)

- m grand : petit reflet spéculaire
- m petit : reflet spéculaire plus étalé



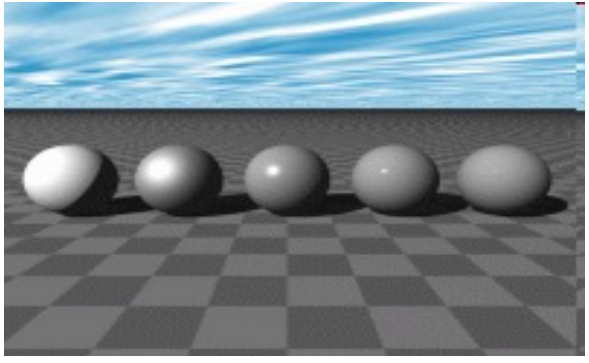
m petit

m grand



Intensité de la lumière spéculaire :

$$I_s = I_i \, K_s \, (H \,.\, N)^m$$



Modèle de Phong : ambient, spéculaire et diffus

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \Big[\underbrace{\mathrm{K_a} \times \begin{pmatrix} R_a \\ G_a \\ B_a \end{pmatrix}}_{\text{Ambiant}} + \overbrace{\begin{pmatrix} R_i \\ G_i \\ B_i \end{pmatrix} \times (\mathrm{K_d} \times (\mathrm{N.L}) + \underbrace{\mathrm{K_s} \times (\mathrm{H.N})^{\mathrm{m}}}_{\text{Spéculaire}})}^{\mathrm{I}}}^{\text{Diffus}} \Big] \times \underbrace{\begin{pmatrix} R_o \\ G_o \\ B_o \end{pmatrix}}_{\text{Couleur de l'objet}}$$

Si on a plusieurs sources lumineuses :

- somme les termes diffus et spéculaires

Comment on fait en pratique avec des polygones ?

Flat shading

Eclairage d'une face proportionnel à l'angle entre la normale et la direction de la lumière

- 1 normale par face
- Valeur de l'intensité uniforme pour toute la face

- Discontinuité d'ombrage aux bords de chaque face

Flat

Problème :
on voit les facettes !!

Lissage de Gouraud (*Gouraud shading*)

Gouraud calcule la couleur :

- I = ambient + diffus (sans terme spéculaire)

- **Pour chaque sommet du polygone**

- Au moment du remplissage,

interpolation bilinéaire des couleurs :

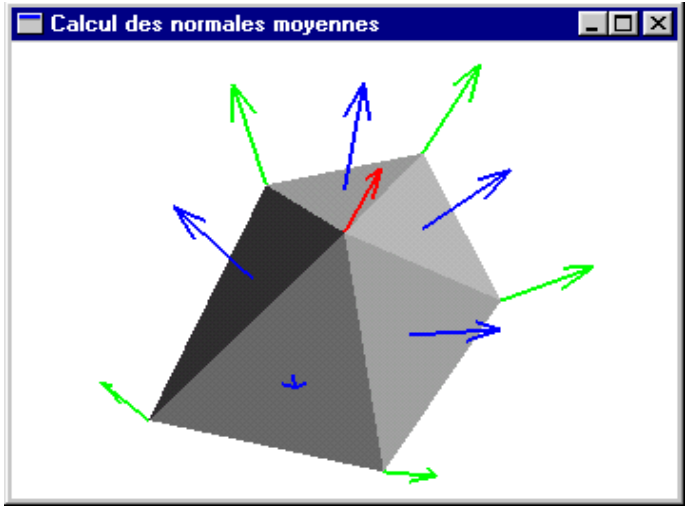
- fait une interpolation des couleur le long des arêtes du polygones
- fait une seconde interpolation lors du remplissage du polygone

Coûte moins cher que de calculer I pour chaque pixel du polygone

Lissage de Gouraud

On calcule une normale par sommet

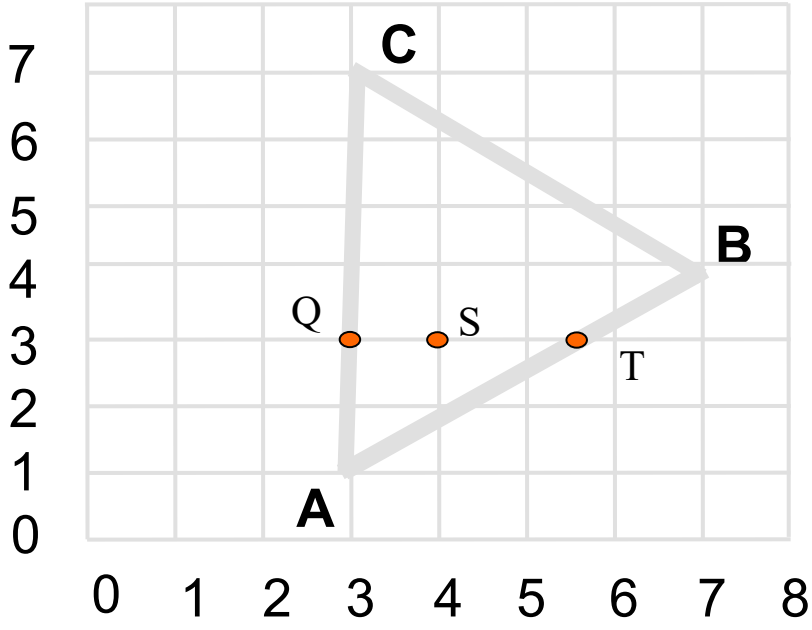
- Moyenne des normales des faces adjacentes



Normale rouge = moyenne des normales bleues

Exemple du lissage de Gouraud

- Gouraud calcule la couleur des sommets A, B, C
- Puis la couleur du pixel S est calculée par interpolation bilinéaire
- Pour composante rouge R_S :



$yQ \in]yA, yB[$

$$VQ = VA + (VC - VA)\frac{yQ-yA}{yC-yA}$$

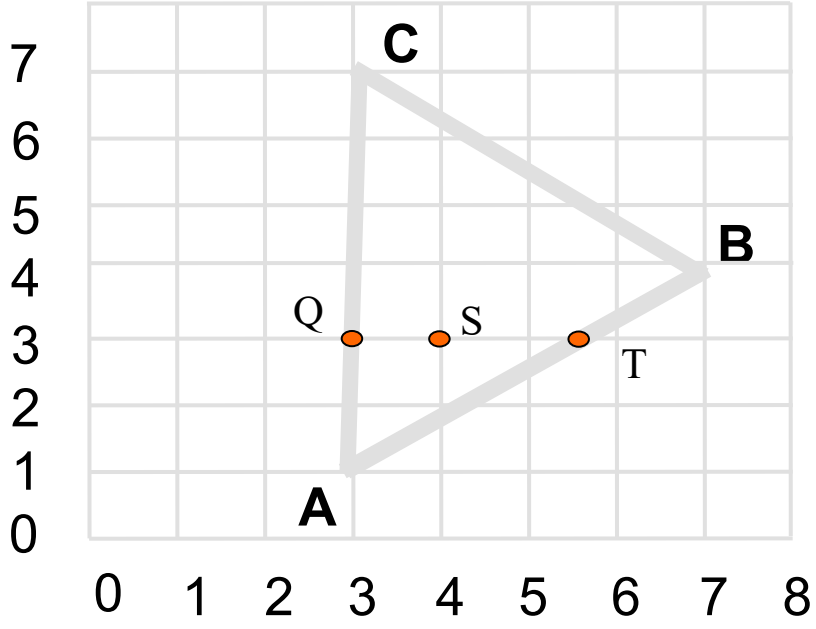
$$\rightarrow VQ = VA + VC\frac{yQ-yA}{yC-yA} - VA\frac{yQ-yA}{yC-yA}$$

$$VQ = VA\left(\frac{yQ-yA}{yC-yA} - \frac{yQ-yA}{yC-yA}\right) + VC\frac{yQ-yA}{yC-yA}$$

$$VQ = VA\frac{yC-yA}{yC-yA} + VC\frac{yQ-yA}{yC-yA}$$

Exemple du lissage de Gouraud

- Gouraud calcule la couleur des sommets A, B, C
- Puis la couleur du pixel S est calculée par interpolation bilinéaire
- Pour composante rouge R_S :



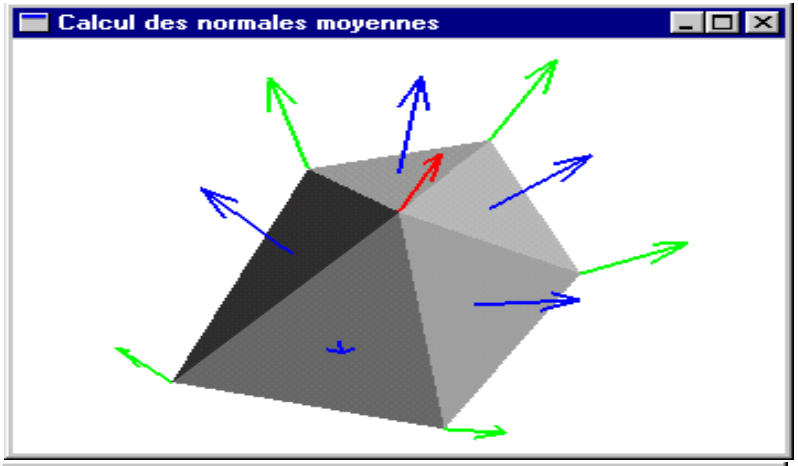
$$R_Q = \frac{y_Q - y_A}{y_C - y_A} R_C + \frac{y_C - y_Q}{y_C - y_A} R_A = \frac{2}{6} R_C + \frac{4}{6} R_A$$

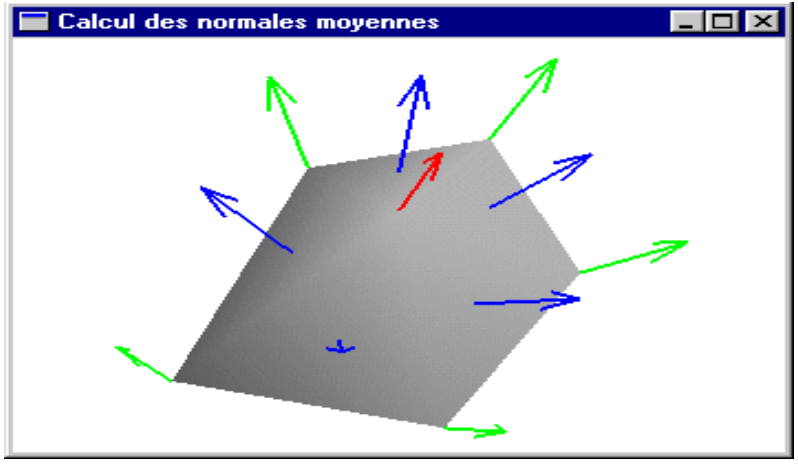
$$R_T = \frac{y_T - y_A}{y_B - y_A} R_B + \frac{y_B - y_T}{y_B - y_A} R_A$$

$$R_S = \frac{x_S - x_Q}{x_T - x_Q} R_T + \frac{x_T - x_S}{x_T - x_Q} R_Q$$

Idem pour G_S et B_S

Flat vs Gouraud shading





Flat shading :

- 1 normale par face
- Couleur uniforme sur la face
- Discontinuité aux bords des faces

Lissage de Gouraud :

- 1 normale par sommet
- Calcul de la couleur aux sommets
- Interpolation de la couleur pendant le remplissage de la face

Lissage de Gouraud

Problème

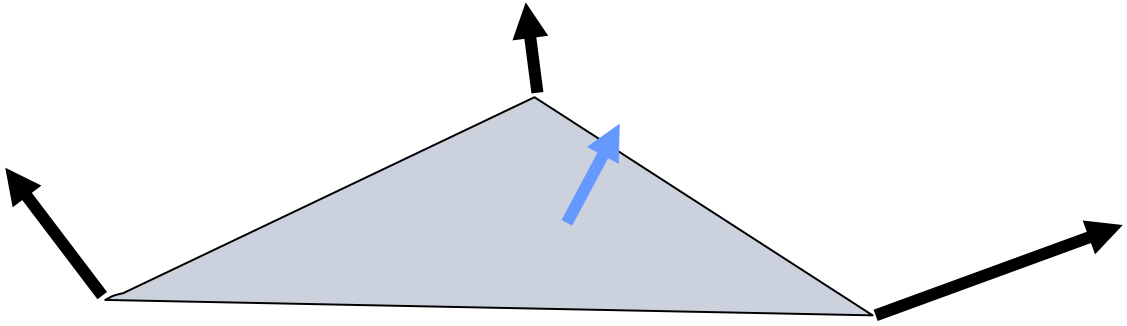
- Ne tient pas compte de la réflexion spéculaire
  - Pas de tâche spéculaire

- Couleur non constante avec la rotation des points

Phong Shading

Phong calcule la couleur en chaque pixel

- I = ambiant + diffus + spéculaire
- Pour trouver la normale en chaque pixel, **interpolation des normales** au moment du remplissage



La normale bleue est une interpolation des normales noires

- Recapture la tâche spéculaire au centre du polygone
- Inclus la composante spéculaire
- Possible en hard avec les Pixel shader (depuis ~2000)

Flat *vs* Gouraud *vs* Phong *Shading*

Gauche : Flat **=** pas d'interpolation (1 normale par surface)

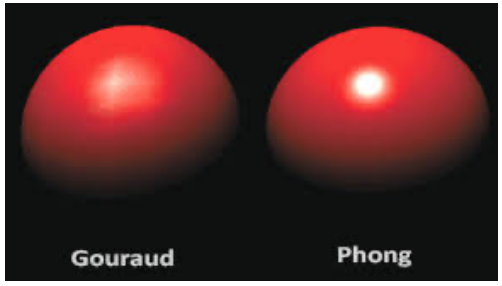
Milieu : Gouraud (Interpolation des couleurs)

Droite : Phong (interpolation des normales)

En résumé

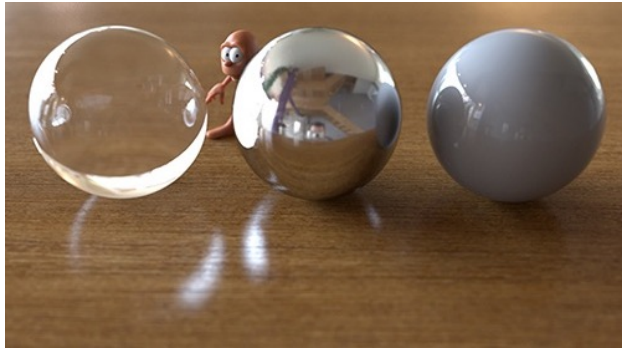
Couleur d'un pixel

- Ambiant
- Diffuse
- Spéculaire
- Il peut y avoir plusieurs sources de lumière



Modèles plus complexes existent

- Ombres
- Modèle plus réaliste de réflexion (BRDF)
- Illumination globale



Eclairage en OpenGL

Définir une source de lumière

```
glEnable(GL_LIGHTING); // Active éclairage
glEnable(GL_LIGHT0); // 8 lumières possibles : 0, ..., 8

// Définition de la position de la lumière 0
GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
```

Rq : La couleur de la source vaut 1,1,1 (blanc) par défaut

Eclairage – code gkit

```
DrawParam gl; // Contexte OpenGL
// Position de la lumiere + couleur de la lumière
 gl.light( Point(0, 20, 20), White() );

// Active la lumière
 gl.lighting(true);

// Désactive la lumière
gl.lighting(false);
```

Eclairage en OpenGL

Donner une couleur diffuse et spéculaire à un objet

```
// Définition des propriétés du matériel
GLfloat mat_diffus[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat mat_shininess[] = { 50.0 };

glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUS, mat_diffus);
glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, mat_specular);
glMaterialfv(GL_FRONT_AND_BACK, GL_SHININESS, mat_shininess);

// GL_FRONT_AND_BACK : indique faces où la matière doit être appliquée
draw_cube(T); // avec T une transformation géométrique
```

Ce qu'il faut retenir du cours

Ajout de la couleur dans Gkit aux sommets du maillage

Notions de lumière ambiante, diffuse, spéculaire

Notions d'illumination en Informatique Graphique

Modèle d'éclairage de Phong

Notion de lissage