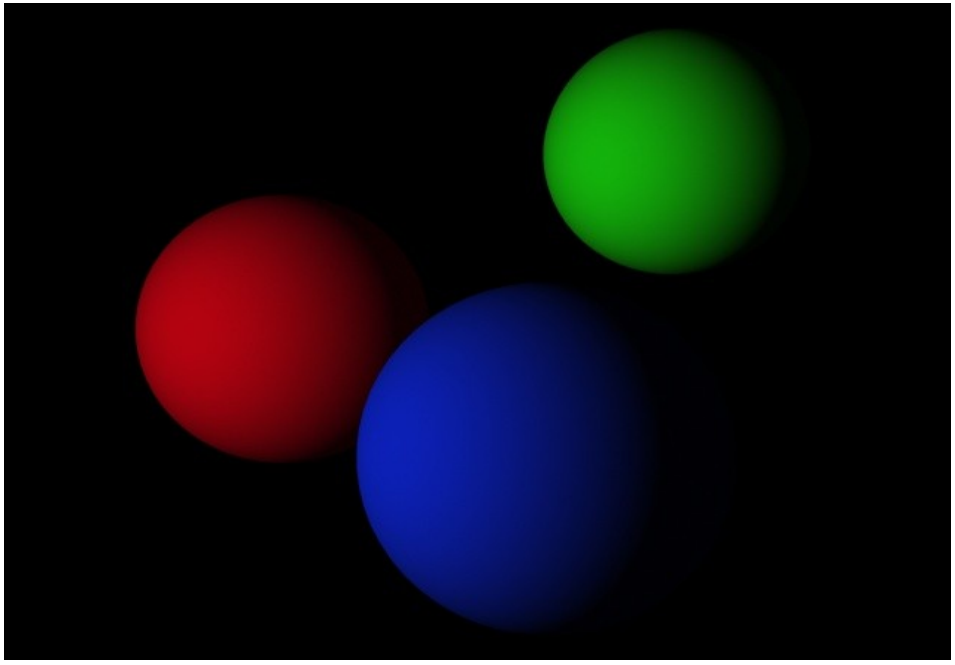
Illumination

Illumination (shading)

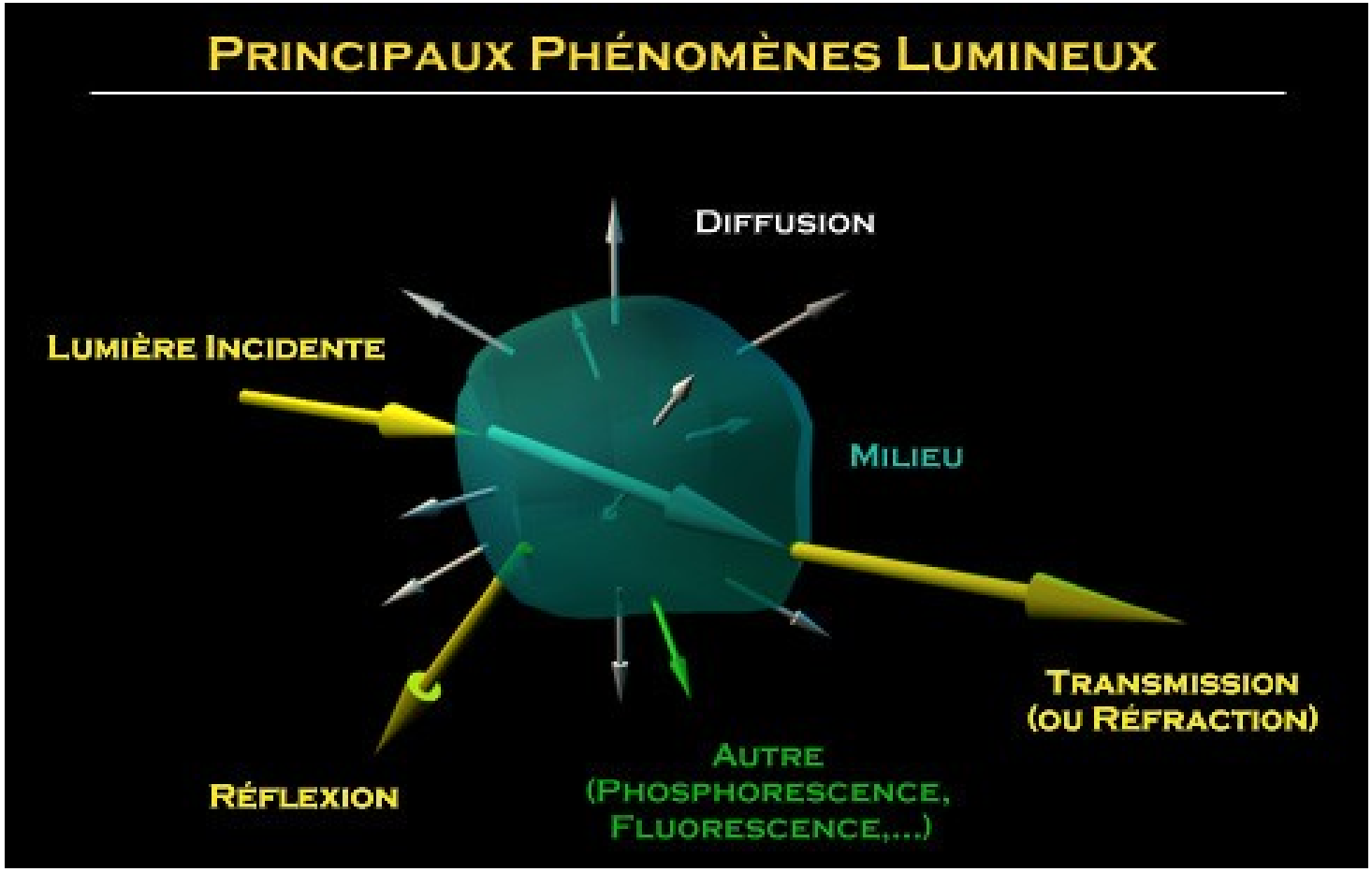
- Transformation du modèle
- **Illumination**
- Transformation de vue
- Découpage (Clipping)
- Projection (dans l'espace écran)
- Rastérisation
- Visibilité/Affichage

Les primitives sont éclairées selon leur matériau, leur surface, et les sources de lumières

Le calcul est local aux primitives: pas d'ombrage

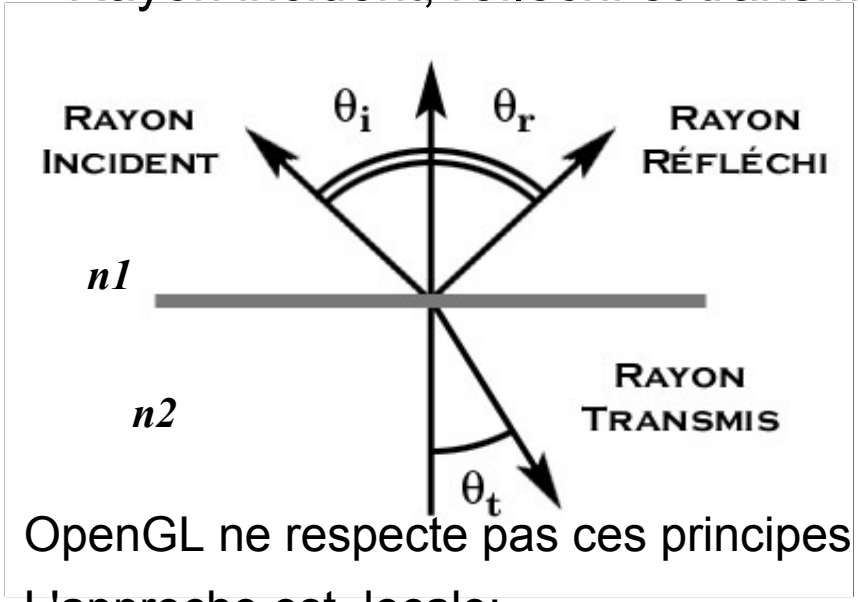


Introduction



Illumination (shading)

- Lois de l'optique géométrique (Descartes)
  - Modèle le plus simple (rayon lumineux)
  - Propagation rectiligne, principe du chemin inverse
  - Rayon incident, réfléchi et transmis dans le même plan

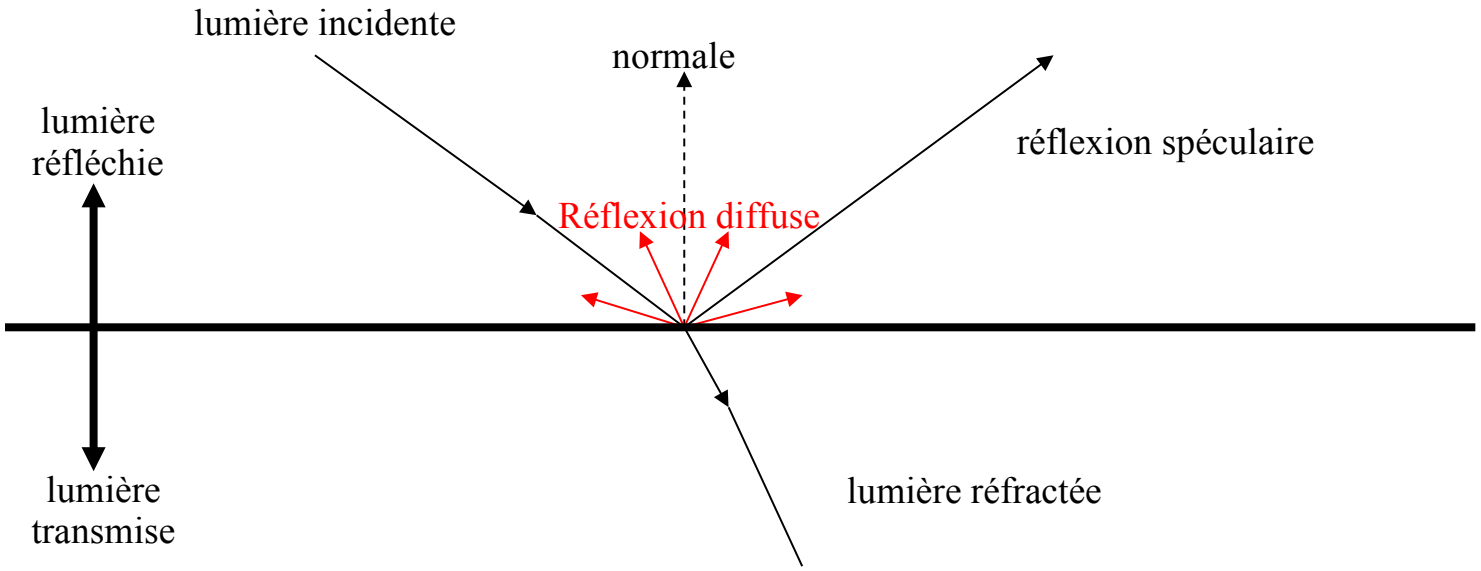


$$\theta_r = -\theta_i$$
$$n_1 \sin(\theta_i) = n_2 \sin(\theta_t)$$

- OpenGL ne respecte pas ces principes
- L'approche est locale:
  - Seuls les éclairages directs sont pris en considération
  - Permet d'expliquer les effets simples (ombre, pénombre, pleine lumière)
  - Les effets des éclairages indirects sont simulés (souvent de manière assez approximative).

Modèle de réflexion

- Un modèle de réflexion décrit l'interaction entre la lumière et une surface en fonction des propriétés du matériau constitutif de la surface ainsi que de la nature de la lumière et de son incidence.



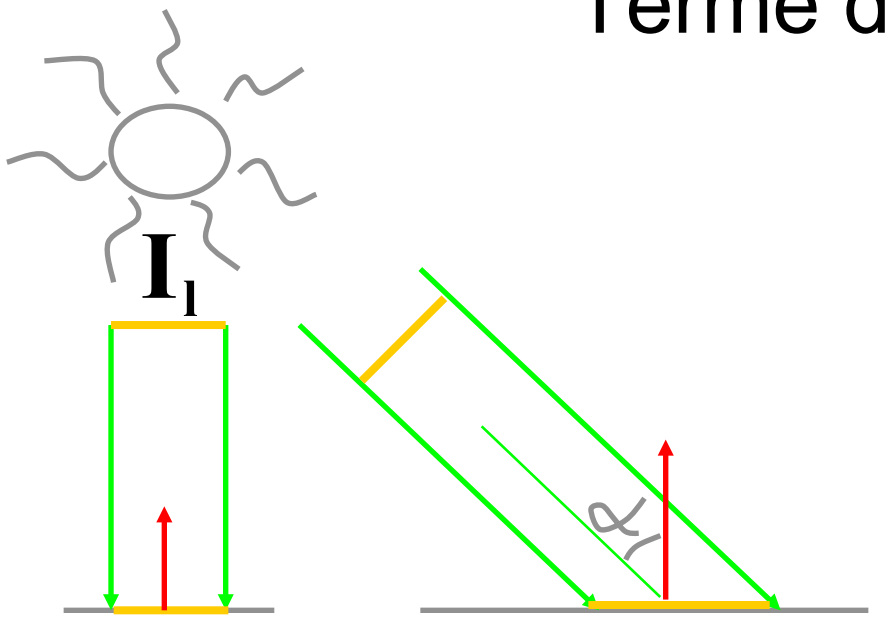
Une partie du flux lumineux est absorbée par le matériau.

Terme ambiant

- Une couleur ambiante de source lumineuse Il
  - représente l'ambiance lumineuse émise dans la scène
  - simulation bon marché d'une illumination globale
- Coefficient des matériaux ka
  - représente l'absorption de l'éclairage ambiant
- Très pauvre mais utile
  - pas de sens physique
  - pas d'indication sur la forme des objets

$$I_a = k_a I_l$$

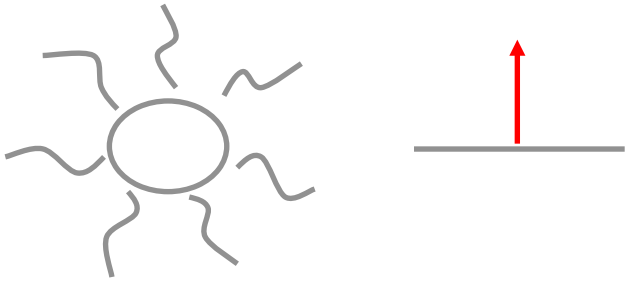
Le modèle Lambertien Terme diffus



$$I_d = k_d I_l (\cos \alpha)$$

énergie dispersée

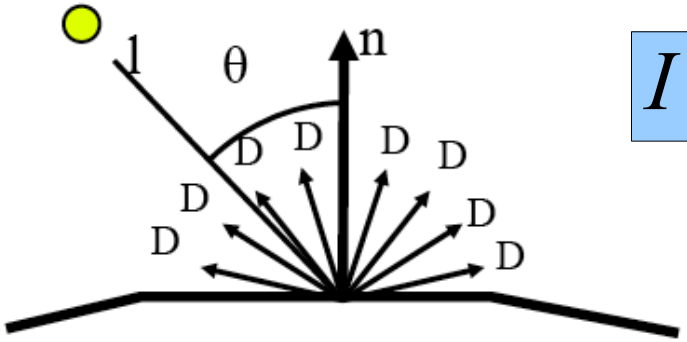
Incidence normale ; énergie conservée



Incidence nulle ; Pas d'énergie

Le modèle Lambertien
Terme diffus

- Suppose un matériau lambertien
  - la lumière se réfléchie de la même manière dans toutes les directions
  - La lumière réfléchie en un point dépend de
    - du coef d'absorption du matériau K_d et de la « couleur diffuse » de la source lumineuse I_d
    - de l'angle entre normale et lumière incidente

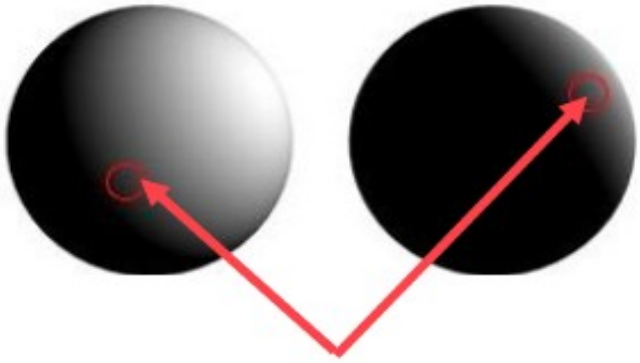


$$I_d = k_d I_l(\cos\theta)$$

Le modèle Lambertien
Terme diffus

Permet d'interpréter la forme des objets

Augmentation de K_d



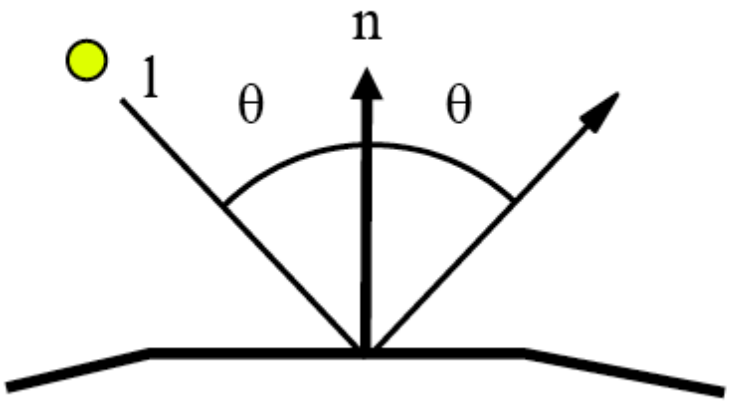
même couleur

La couleur est indépendante du point de vue:
La couleur/lumière est la même quel que soit le point de vue

Terme spéculaire

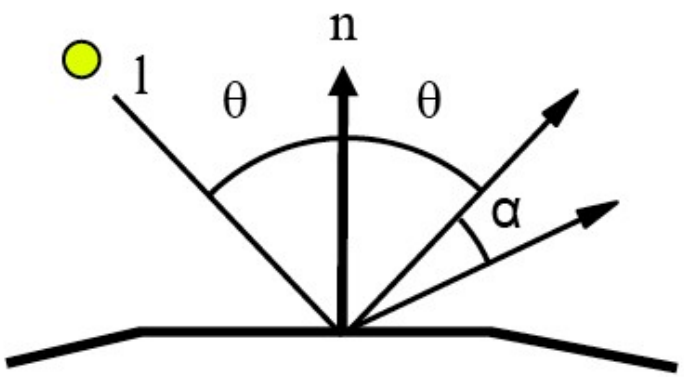
Cas idéal : un mirroir

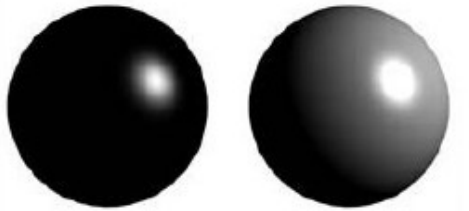
- loi de descartes
  - « la lumière est réfléchie selon un angle égal à l'angle incident »
- problème : la réflexion d'une source ponctuelle n'est visible qu'en peu de point sur une surface :



Terme spéculaire

- En réalité : objets légèrement dépolis
  - la lumière est réfléchie
    - « autour » du vecteur de réflexion
    - avec un coef de diminution exponentielle n (shininess)
    - suivant un coef d'absorption K_s et une couleur de source lumineuse I_s



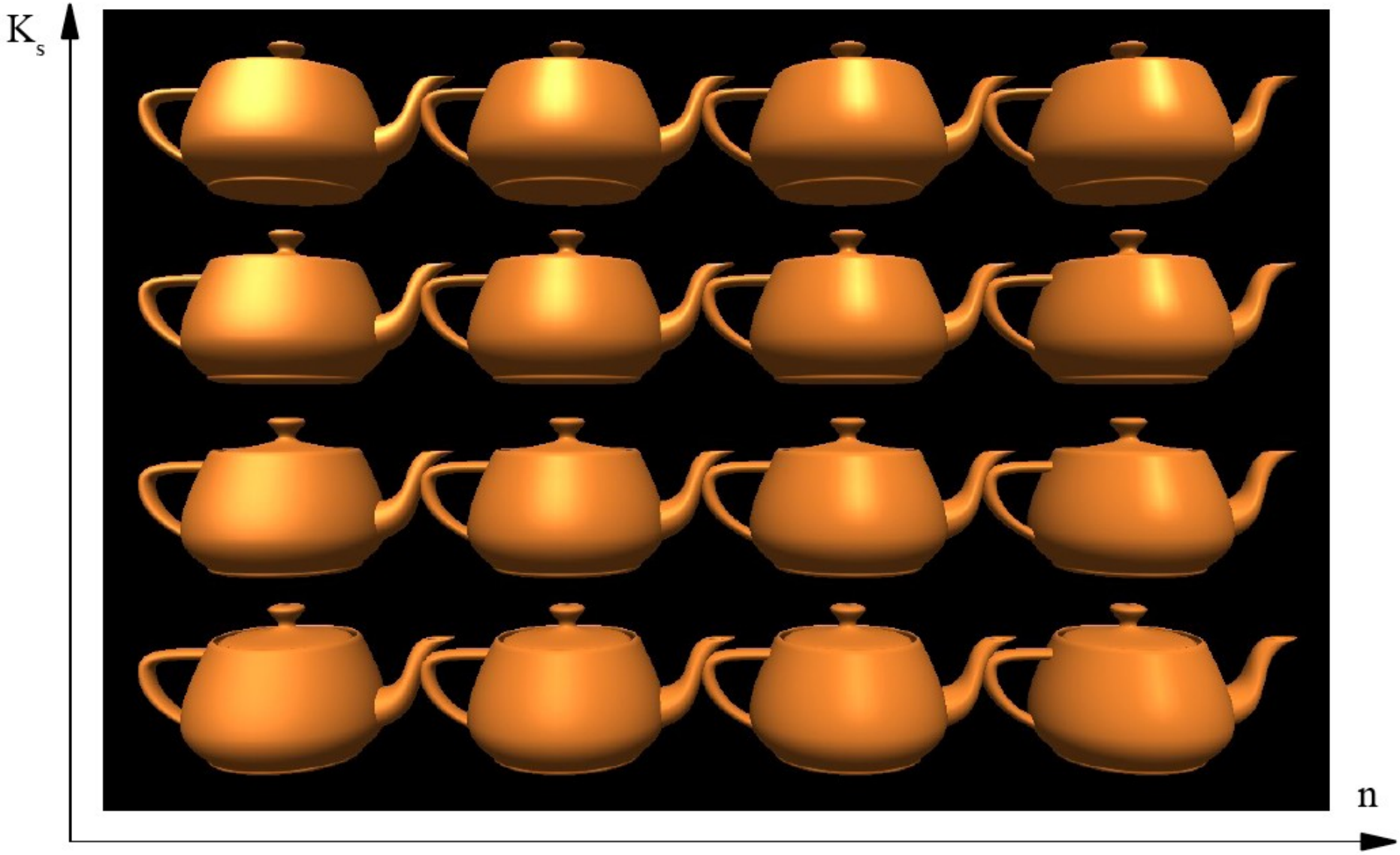


spéculaire | spéculaire + diffus

lobe spéculaire

Terme spéculaire : $S = K_s I_s (\cos \alpha)^n$

Terme spéculaire



Le modèle de Phong 1975

formule pour représenter la lumière spéculaire

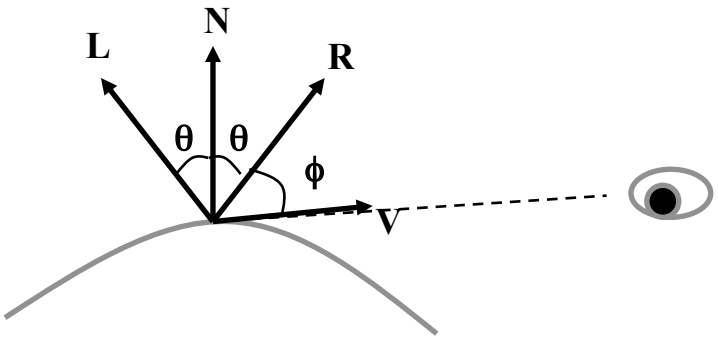
- Pas physiquement correct
- Le modèle d'illumination de *Phong* permet de calculer la quantité de lumière allant vers l'observateur en fonction du matériau.

$$I_s = k_s I_l (\cos\varphi)^{n_{shiny}}$$

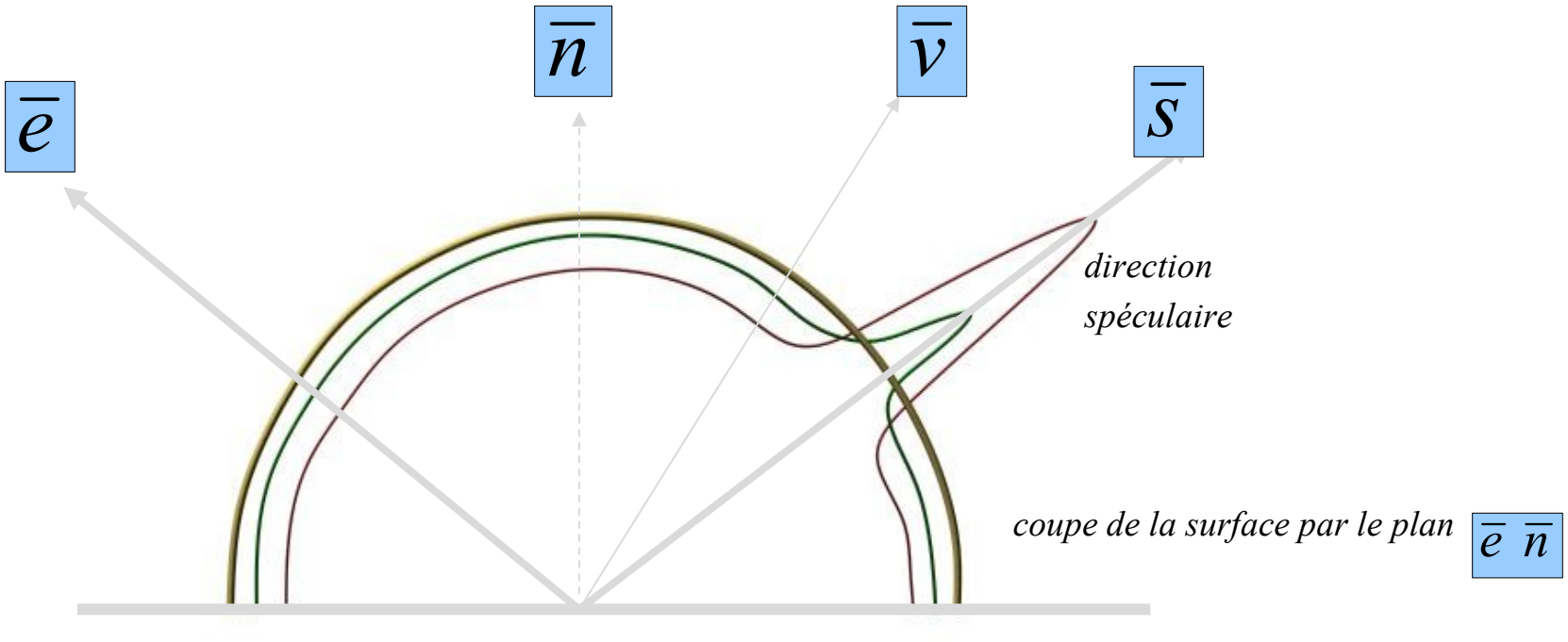
φ : Angle entre le rayon de lumière réfléchi **R** et l'observateur **V**

k : Luminance spéculaire

n_{shiny} : Brillance de l'objet



Le modèle de Phong

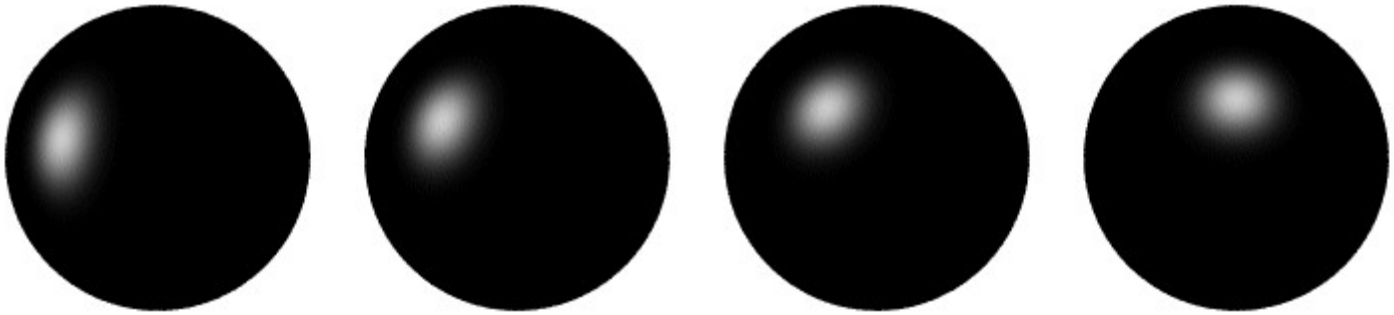


Réflectance pour un matériau lambertien (pas de spécularité)

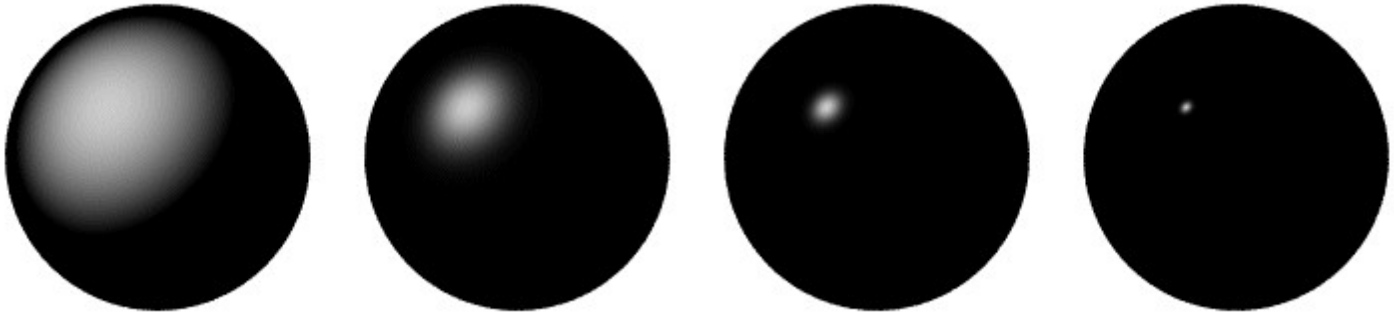
Réflectance pour un matériau légèrement spéculaire

Réflectance pour un matériau moyennement spéculaire

Le modèle de



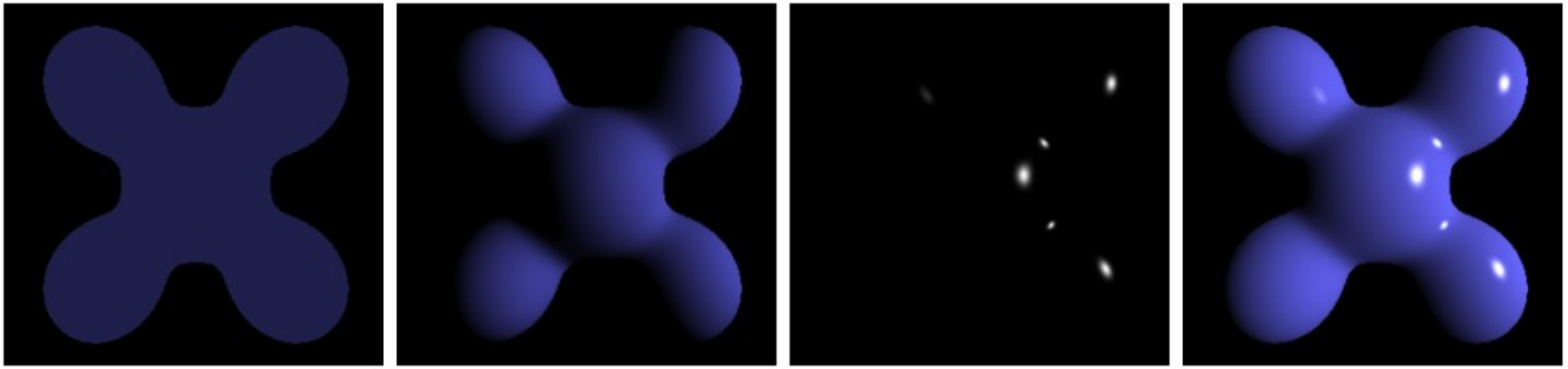
Si on déplace une source lumineuse



Si on change la brillance

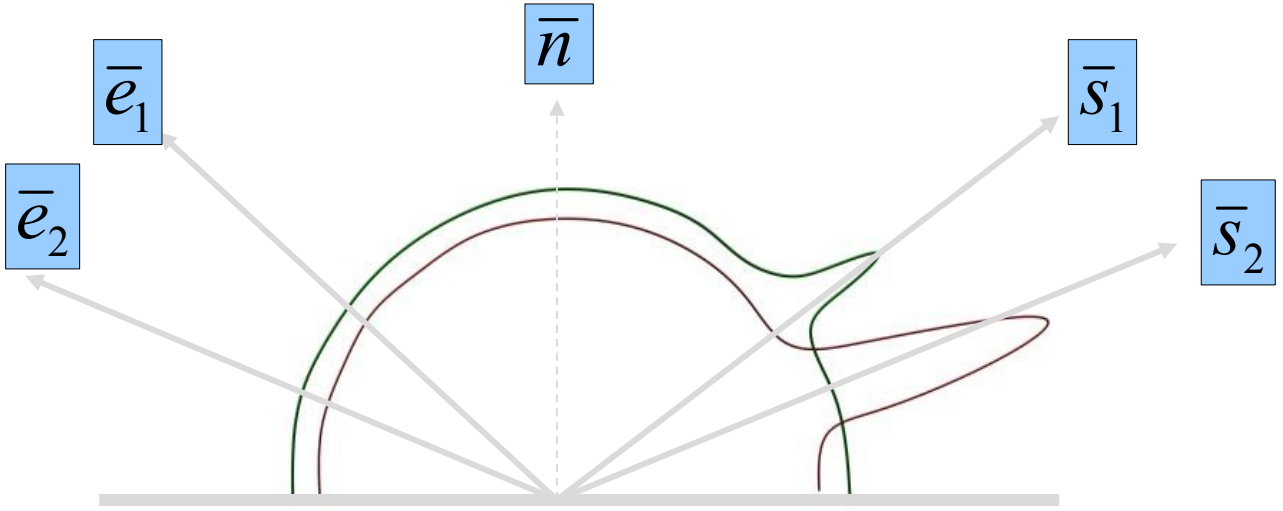
Éclairage Phong

Crédits: fr.wikipedia.org



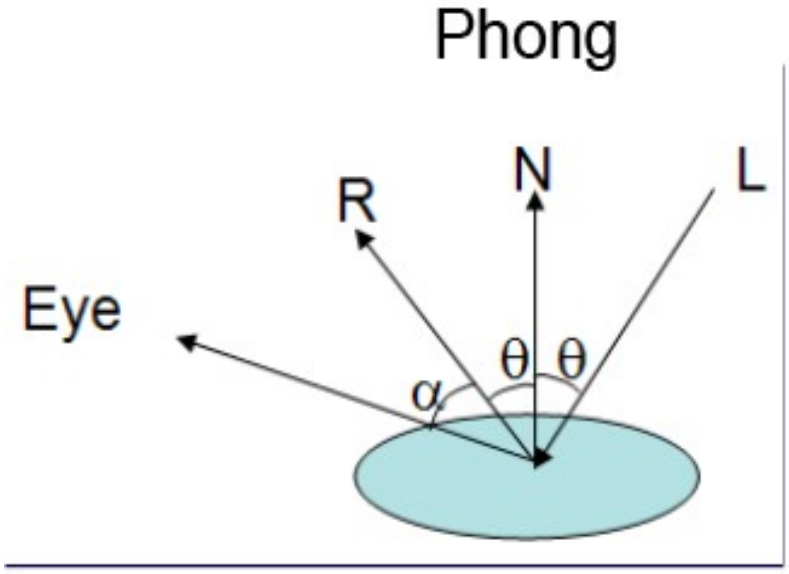
Plus fin que le modèle de Phong: le modèle de Cook-Torrance

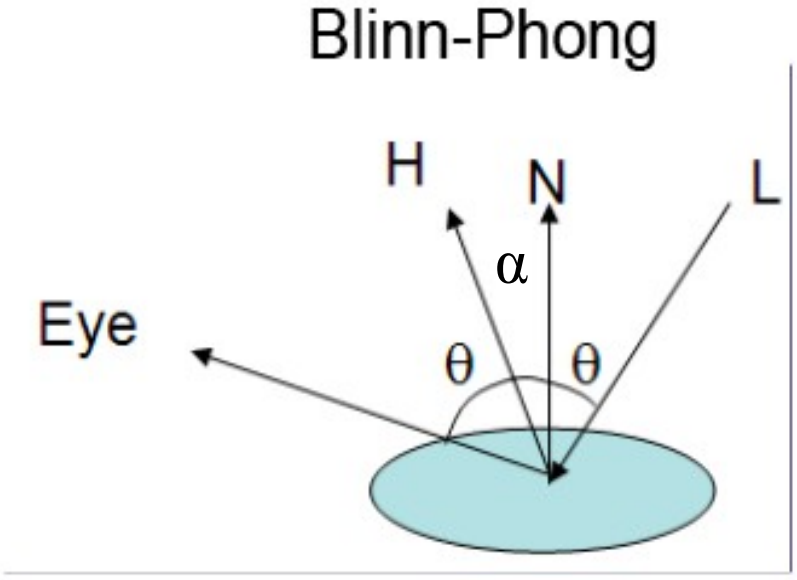
Ce modèle corrige une faiblesse du modèle de Phong en ce qui concerne la composante spéculaire ; en effet, pour beaucoup de matériaux (métalliques, entre autres), la protubérance spéculaire augmente avec l'angle d'incidence et s'écarte de la direction de spéculaire.



Moins fin que le modèle de Phong: le modèle de Blinn-Phong (1977)

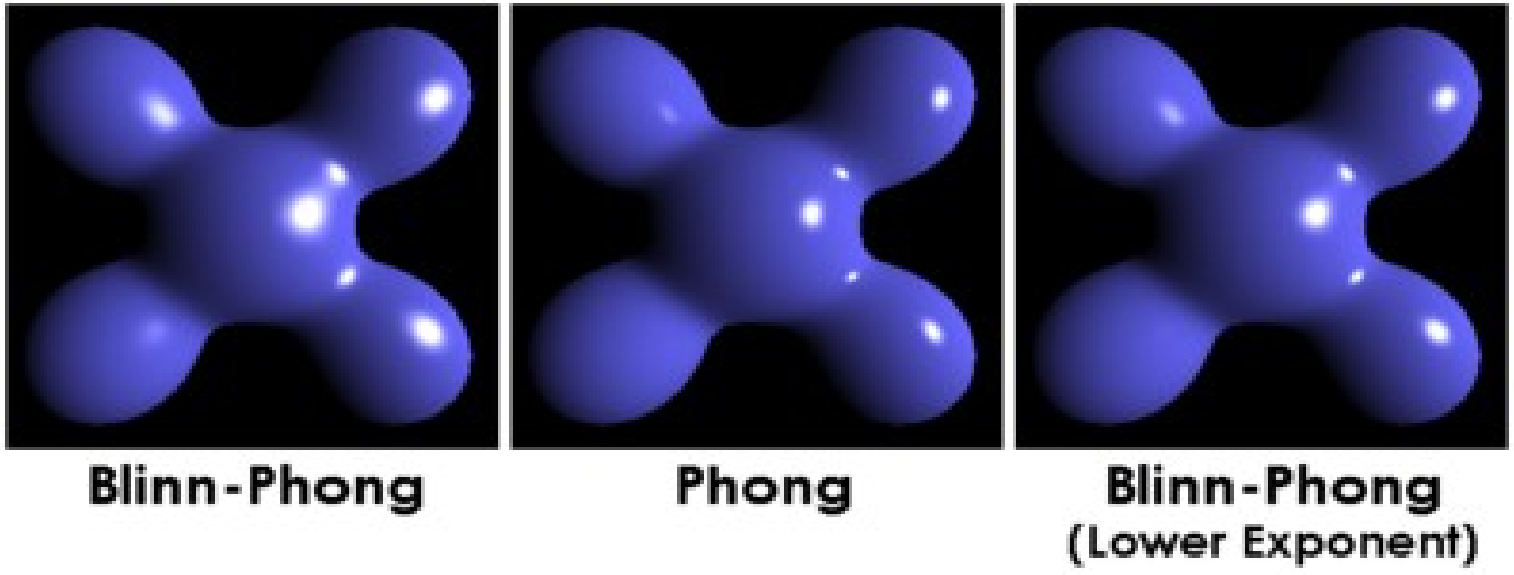
- Simplification:
  - un vecteur entre ceux de la caméra et de l'oeil (« demi-vecteur »)
  - pas d'évaluation du vecteur réfléchi R (couteux)
  - On remplace le terme R.Eye par N.H
  - H=(L+Eye) /2





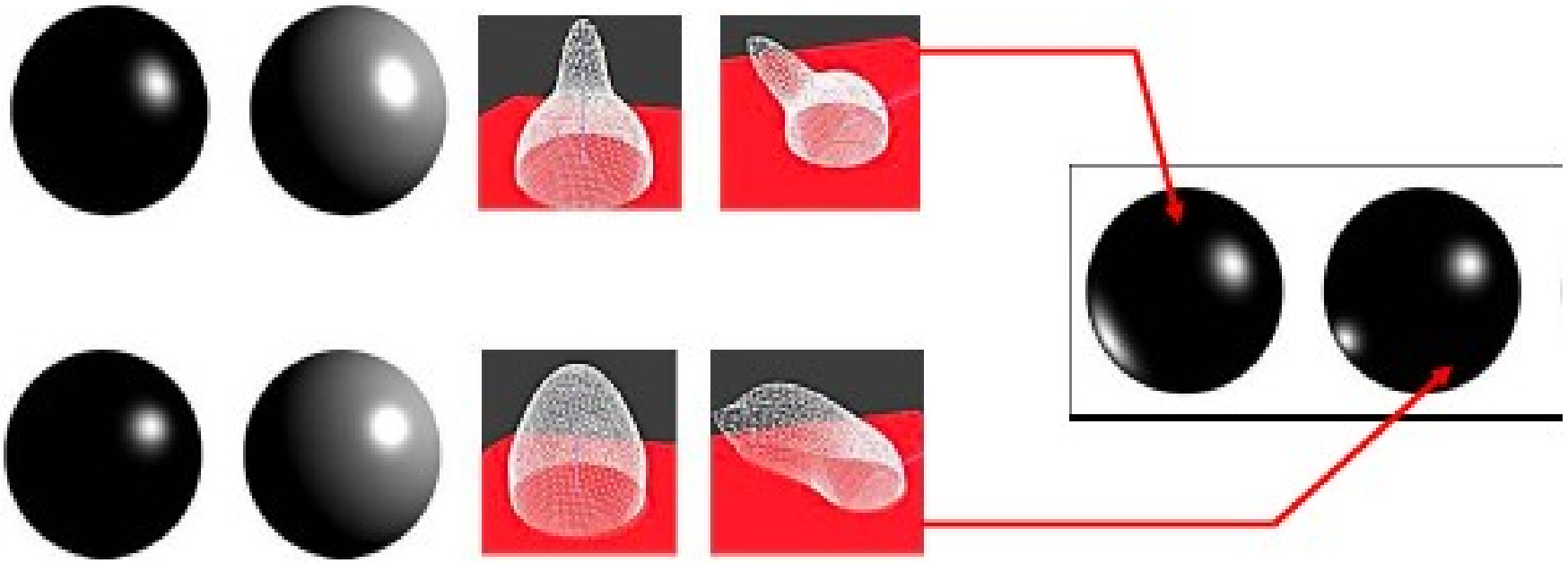
Moins fin que le modèle de Phong: le modèle de Blinn-Phong

- Évidemment l'angle n'est pas le même
  - Mais on remplace $(R.Eye)^n$ par $(N.H)^{n'}$
  - On peut compenser par le choix de l'exposant *n'*



Moins fin que le modèle de Phong: le modèle de Blinn-Phong

- Peu de différences
- C'est le modèle de OpenGL



Equation d'éclairement

•La couleur d'éclairement d'un pixel est égal à la somme des termes d'éclairement pour toutes les sources lumineuses

$$I = \sum_{Sources} \{ I_a + I_d + I_s \}$$

Equation d'éclairement

•La couleur d'éclairement d'un pixel est égal à la somme des termes d'éclairement pour toutes les sources lumineuses

$$I_a = k_a I_l$$

$$I = \sum_{Sources} \left\{ k_a I_l + I_d + I_s \right\}$$

Equation d'éclairement

•La couleur d'éclairement d'un pixel est égal à la somme des termes d'éclairement pour toutes les sources lumineuses

$$I_d = k_d I_l (\cos\theta)$$

$$I = \sum_{Sources} \{ k_a I_l + k_d I_l (\cos\theta) + I_s \}$$

Equation d'éclairement

•La couleur d'éclairement d'un pixel est égal à la somme des termes d'éclairement pour toutes les sources lumineuses

$$I_s = k_s I_l (\cos\alpha)^{n_{shiny}}$$

ou bien

$$I_s = k_s I_l (\vec{R}.\overrightarrow{Eye})^{n_{shiny}}$$

$$I = \sum_{Sources} \left\{ k_a I_l + k_d I_l (\cos\theta) + k_s I_l (\cos\alpha)^{n_{shiny}} \right\}$$

Equation d'éclairement

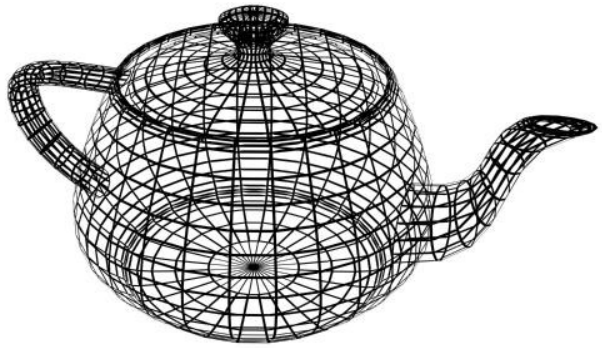
•La couleur d'éclairement d'un pixel est égal à la somme des termes d'éclairement pour toutes les sources lumineuses

$$I = \sum_{Sources} \left\{ k_a I_l + k_d I_l (\cos\theta) + k_s I_l (\cos\alpha)^{n_{shiny}} \right\}$$

Le rendu des polygones

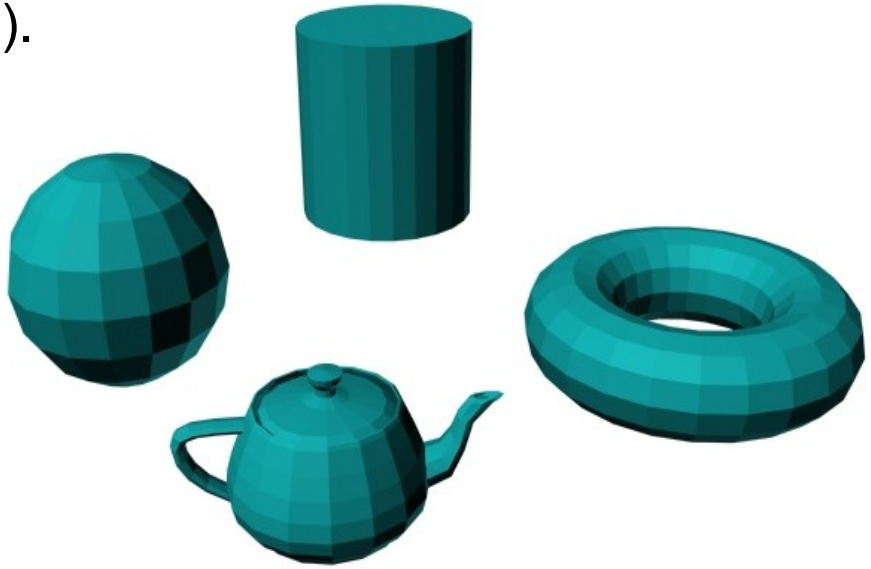
- Etant donné un modèle d'illumination local (ex. Phong), comment « rendre » en globalité les polygones de la scène ?
- Les algorithmes de rendu travaillent presque toujours avec des maillages de polygones... c 'est beaucoup plus simple et rapide !

- <u>Algorithmes de rendu</u>
  - fil de fer
  - rendu facette
  - Gouraud
  - Phong
  - Texturation



Rendu facette (flat shading)

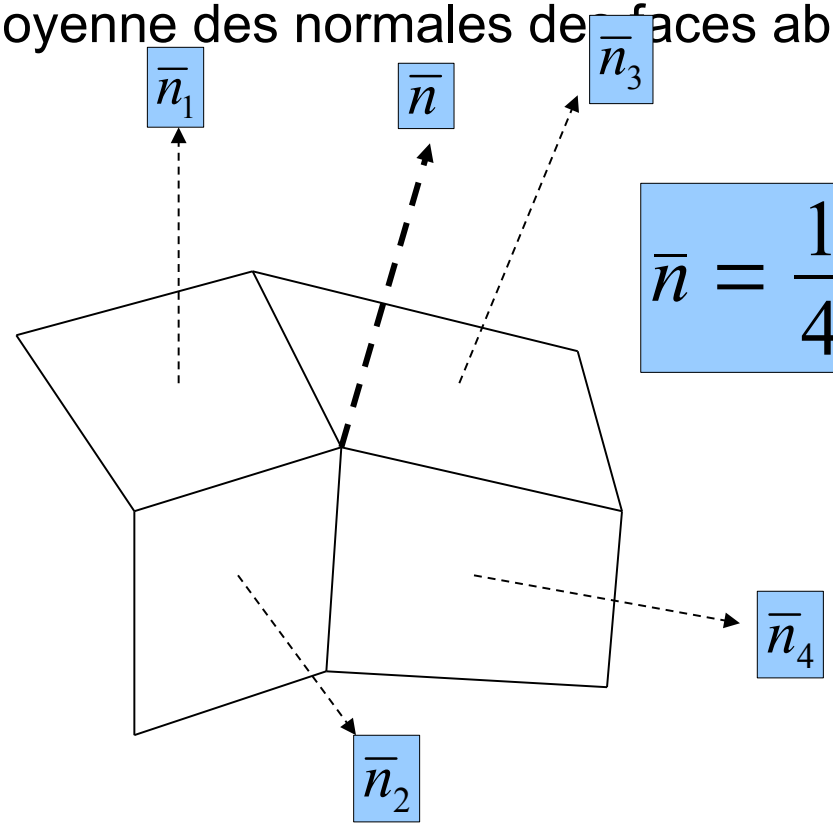
- C'est l'algorithme le plus simple dans la mesure où il n'y a pas d'interpolation entre les faces. Tous les points d'une face ont la même normale.
- Les points sont rendus en utilisant le modèle d'illumination Lambertien, par exemple.
- Les arêtes entre faces sont partout visibles (discontinuité des normales).



Les arêtes visibles séparent des faces appartenant à des groupes différents.

Gouraud

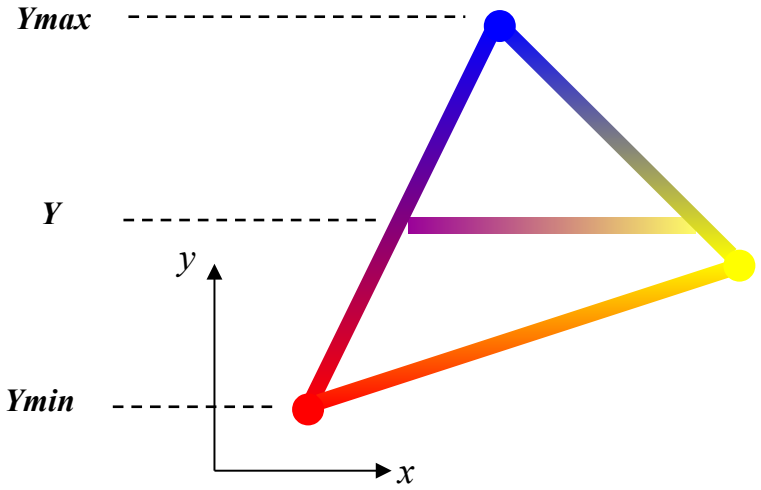
- Cet algorithme améliore le rendu en faisant une interpolation entre les faces appartenant à un même groupe (smoothing group).
- On attribue à chaque sommet une “normale” qui est la moyenne des normales des faces aboutissant au sommet.



$$\bar{n} = \frac{1}{4}\left(\bar{n}_1 + \bar{n}_2 + \bar{n}_3 + \bar{n}_4\right)$$

Gouraud (suite)

- Calcul de l'intensité réfléchie sur chaque sommet (avec le modèle d'illumination de Phong par exemple).
- L'interpolation des intensités réfléchies (ou plus simplement des couleurs) se fait alors le long des arêtes et ensuites dans les faces.



$$C_{Ymin},\ C_{Ymax} = \text{shading at Ymin, Ymax}$$

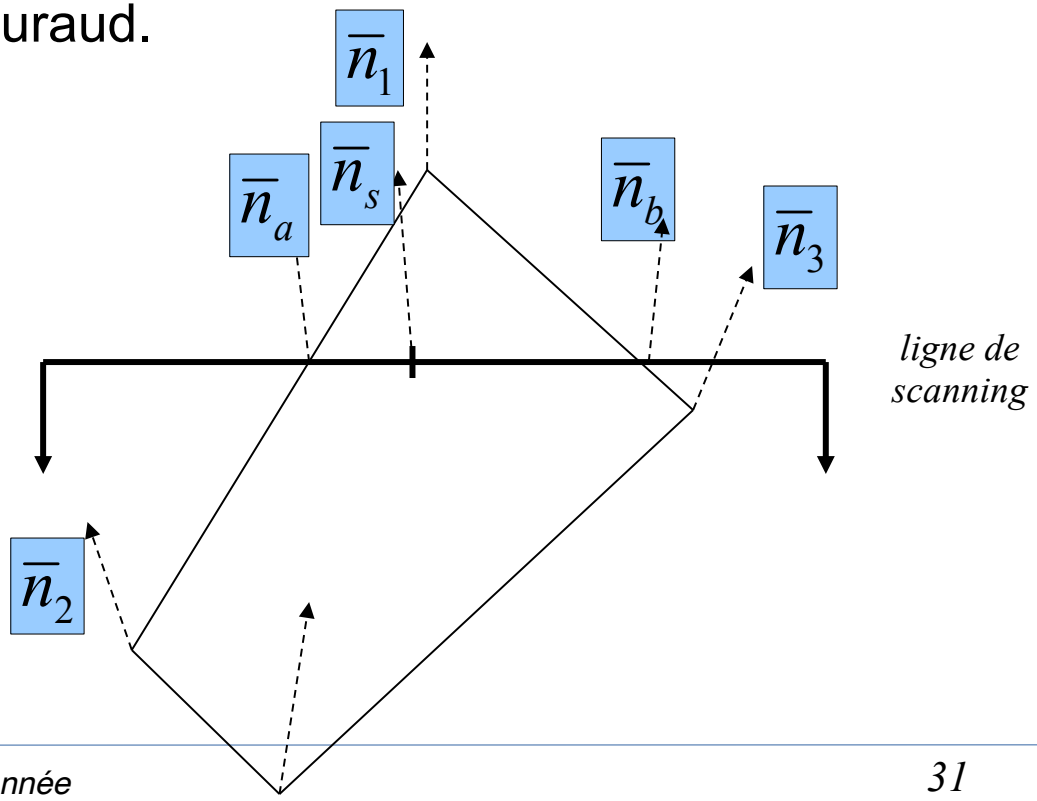
$$C_y = \frac{(Y - Y\min)C_{Y\max} + (Y\max - Y)C_{Y\min}}{Y\max - Y\min}$$

- Plus rapide que de calculer l'illumination pour chaque pixel.
- Le rendu spéculaire n'est pas de très bonne qualité; en effet, une réflexion spéculaire doit apparaître à un sommet pour être rendue.

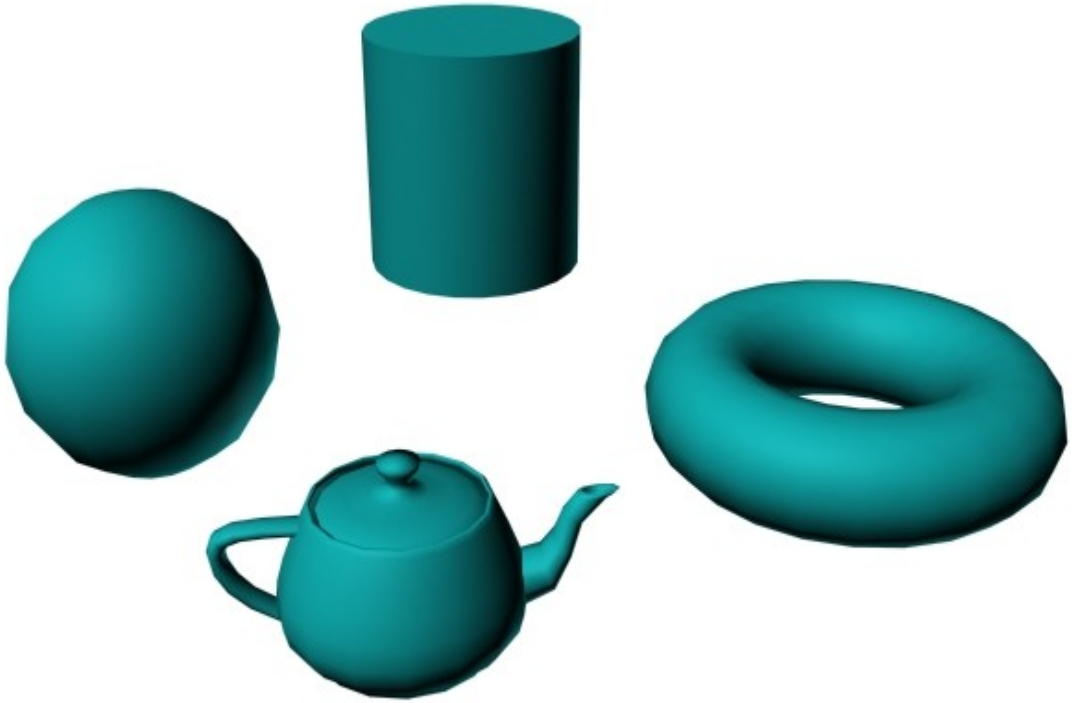
Rendu facette Gouraud Phong

Phong

- Interpolation des normales plutôt que des couleurs
- Méthode d'interpolation identique
- L'intensité réfléchie n'est calculée qu'après interpolation des normales.
- Le rendu spéculaire est de bien meilleure qualité que dans le cas de l'algorithme de Gouraud.



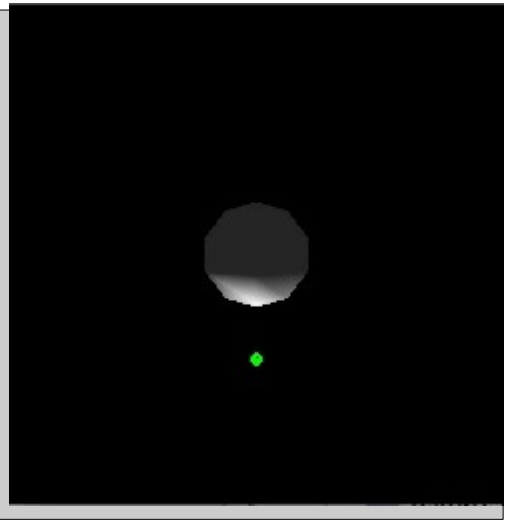
Phong



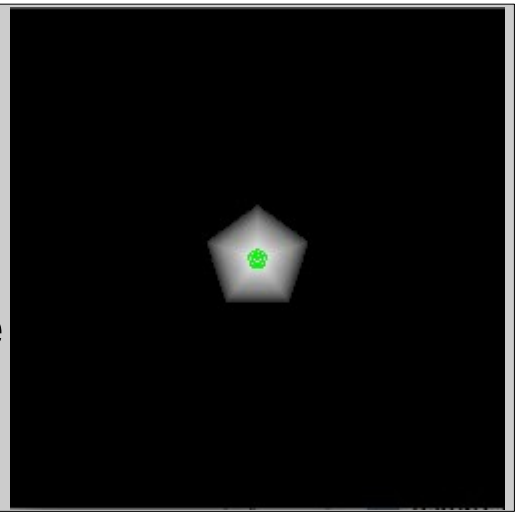
Contrôler la position de la lumière

- La position de la source et sa direction sont transformés par la matrice de modélisation et de visualisation.

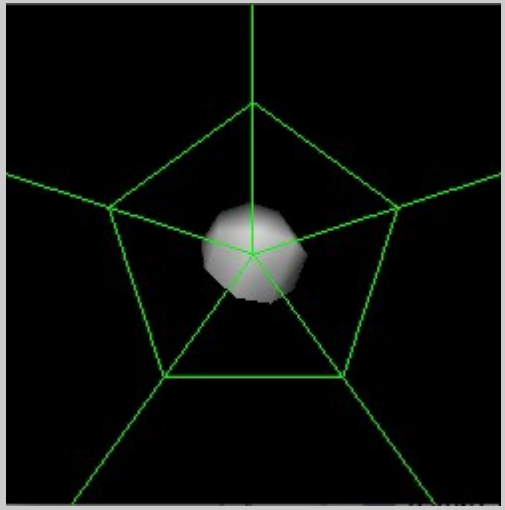
Lumière fixe



Lumière qui se déplace indépendamment de l'objet



Lumière qui ce déplace avec le point de vue



Contrôler la position de la lumière

- La position de la source et sa direction sont transformés par la matrice de modélisation et de visualisation.
- Lumière fixe: placer la source après initialisation de la matrice et la mise en place de la matrice de vision.
- Lumière qui se déplace indépendamment de l'objet:
  - Ajouter des transformations entre la mise en place de la matrice de visualisation et la source
  - Placer plutôt la lumière à la fin du dessin
  - ou bien placer la lumière entre glPushMatrix et glPopMatrix
- Lumière qui ce déplace avec le point de vue:
  - Placer la lumière avant la transformation de visualisation. Ainsi elle reste au point fixé initialement mais par rapport aux coordonnées de l'oeil. La source bouge en même temps que l'oeil car les transformations de visualisations affectent la lumière.

Assemblage Final

- Calcul pour chaque composant RVB séparés:

$$
\begin{aligned}
Couleur\ sommet = {} & émission_{matiere} \\
& + ambiant_{modèle\ d'éclairage} \times ambiant_{matière} \\
& + \sum_{i=0}^{n-1} \left\{ \left(\frac{1}{k_c + k_l \times d + k_q \times d^2} \right)_i \right. \\
& \quad \times effet\ projecteur_i \\
& \quad \times \left[ambiant_{lumière} \times ambiant_{matière} \right. \\
& \quad\quad + max\left\{ \vec{L} . \vec{N} , 0 \right\} \times diffus_{lumière} \times diffus_{matière} \\
& \quad\quad \left. \left. + max\left\{ \vec{H} . \vec{N} , 0 \right\}^{brillance} \times spéculaire_{lumière} \times spéculaire_{matière} \right]_i \right\}
\end{aligned}
$$

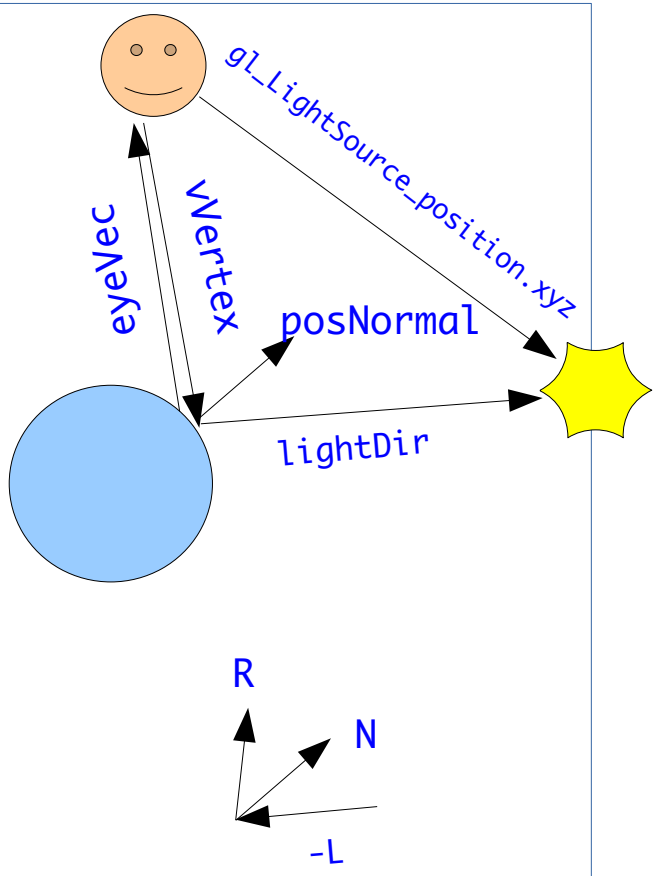
$\vec{L} = vecteur\ du\ sommet\ pointant\ vers\ la\ lumière$

$\vec{N} = vecteur\ normal$

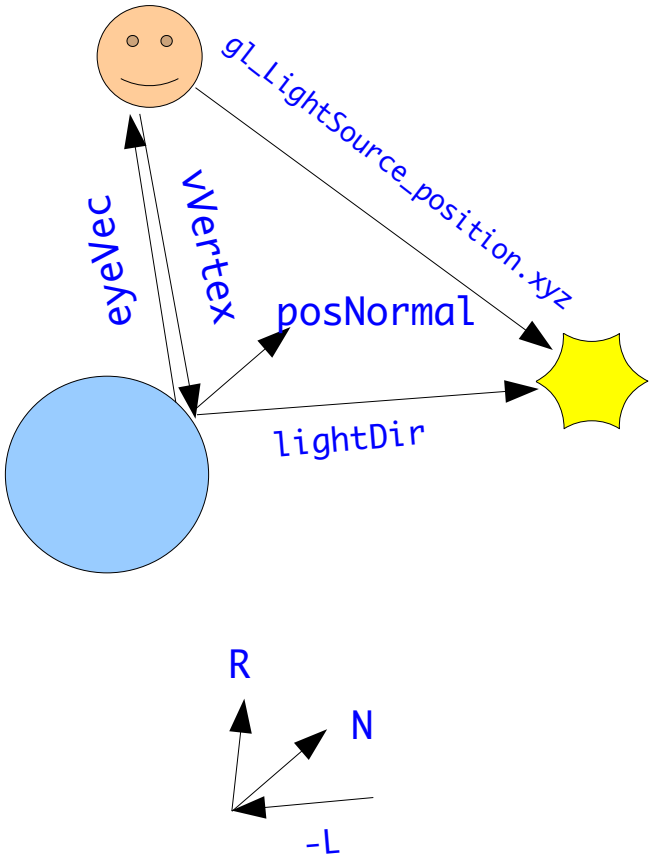
$\vec{H} = somme\ normalisée\ de\ \vec{L}\ et\ du\ vecteur\ \overrightarrow{EYE}\ pointant\ vers\ le\ point\ de\ vue$

Exemple: phong

```
#version 150 core

in vec3 pos;

in vec3   posNormal;

uniform mat4 MV;

uniform mat4 P;

uniform vec4 LightSource_position;

out vec3 lightDir;

out vec3 normalEyeSpace, eyeVec;

void main()

{    gl_Position = P*MV * vec4(pos, 1.0);

    normalEyeSpace = vec3(MV * vec4(posNormal,0.0));

    vec4 vVertex = MV * vec4(pos, 1.0);

        eyeVec = -vVertex.xyz;

    lightDir=vec3(LightSource_position.xyz - vVertex.xyz);

}
```



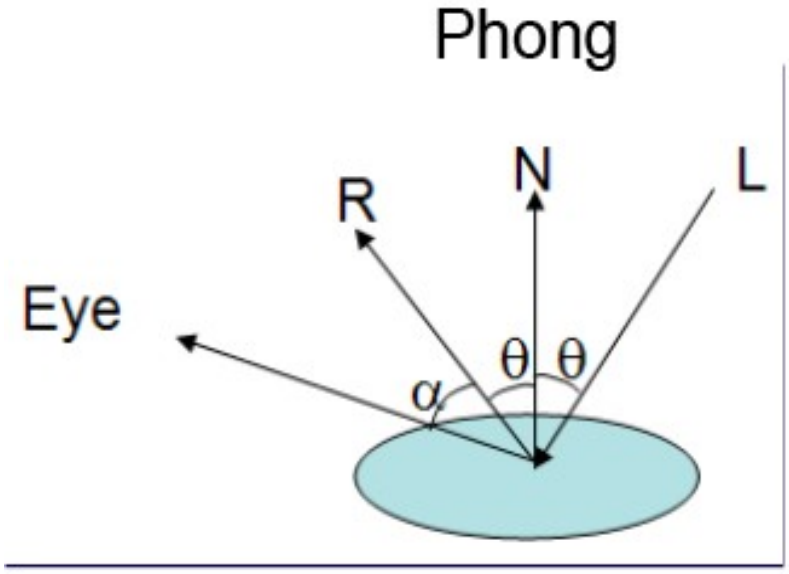
Exemple: phong

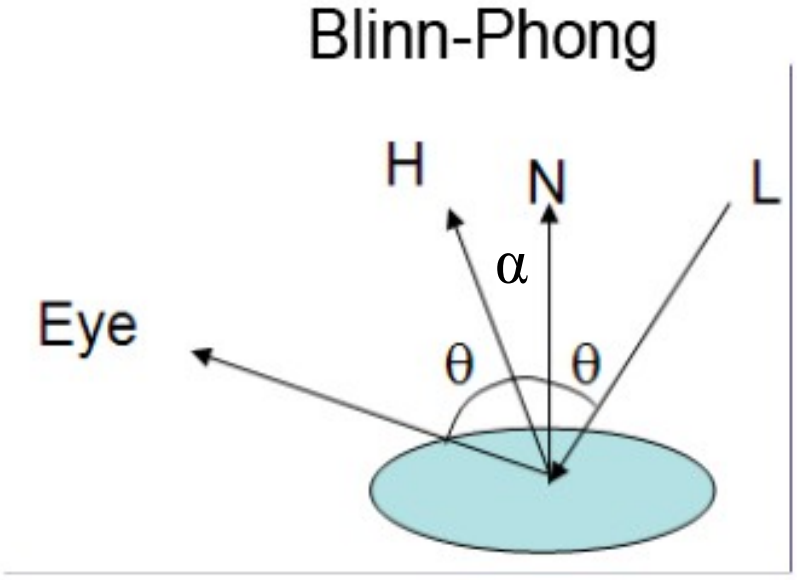



```
#version 150 core

in vec3 lightDir;

in vec3 normalEyeSpace, eyeVec;

void main()

{  vec4 final_color =vec4(0.5,0.5,0.5, 1.0);

     vec3 N = normalize(normalEyeSpace);

       vec3 L = normalize(lightDir);

      float lambertTerm = dot(N,L);

      if (lambertTerm > 0.0)

            {

                  final_color +=0.5*lambertTerm;

        vec3 E = normalize(eyeVec);

        vec3 R = reflect(-L, N);

        float specular = pow(max(dot(R, E), 0.0),0.5);

                  final_color +=0.1*specular;

            }

      outColor = final_color;

}
```

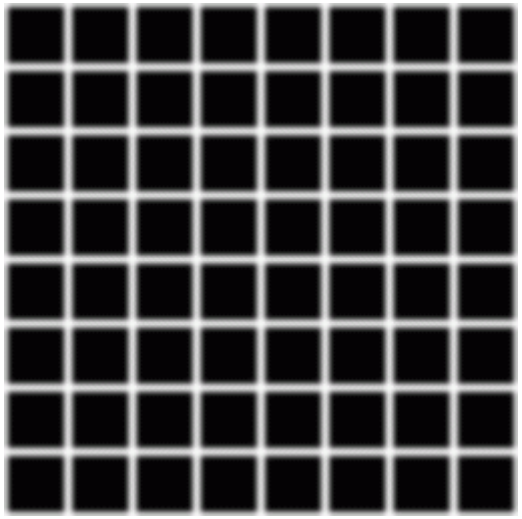
Moins fin que le modèle de Phong: le modèle de Blinn-Phong (1977)

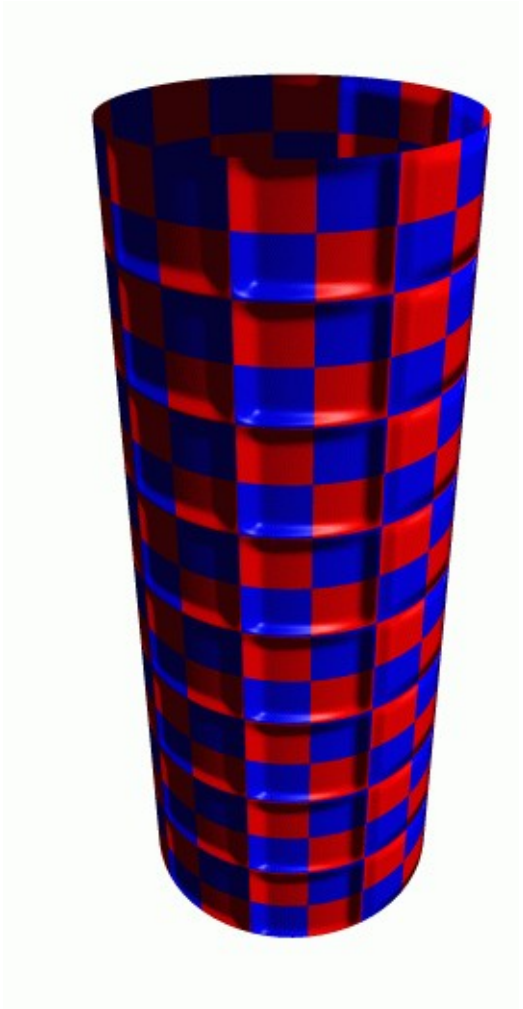
- Simplification:
  - un vecteur entre ceux de la caméra et de l'oeil (« demi-vecteur »)
  - pas d'évaluation du vecteur réfléchi R (couteux)
  - On remplace le terme R.Eye par N.H
  - H=(L+Eye) /2



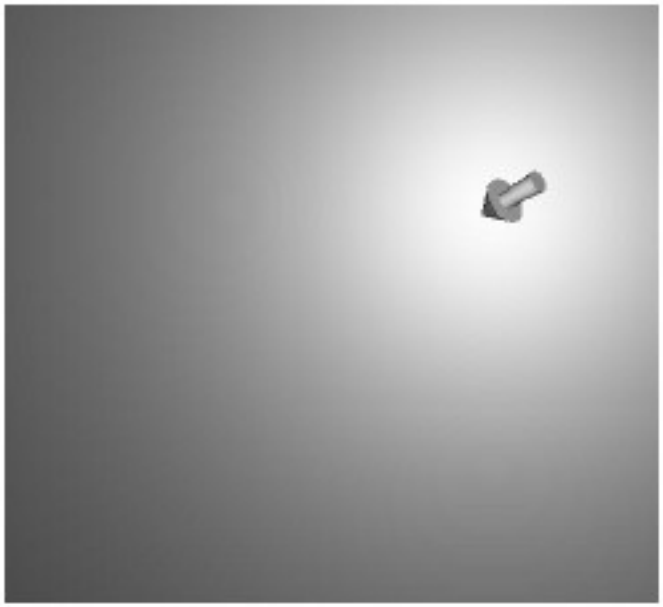


Bump mapping





Bump mapping

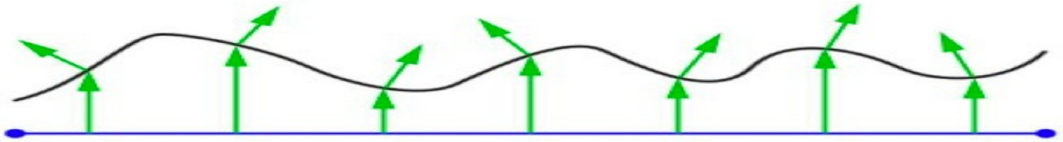


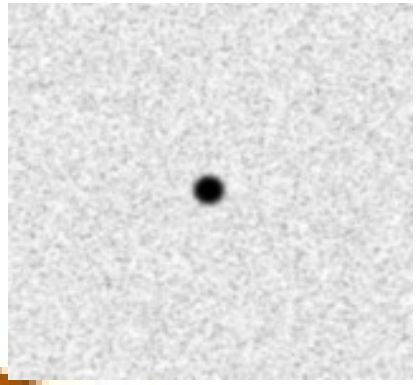
Éclairage diffus sans bump

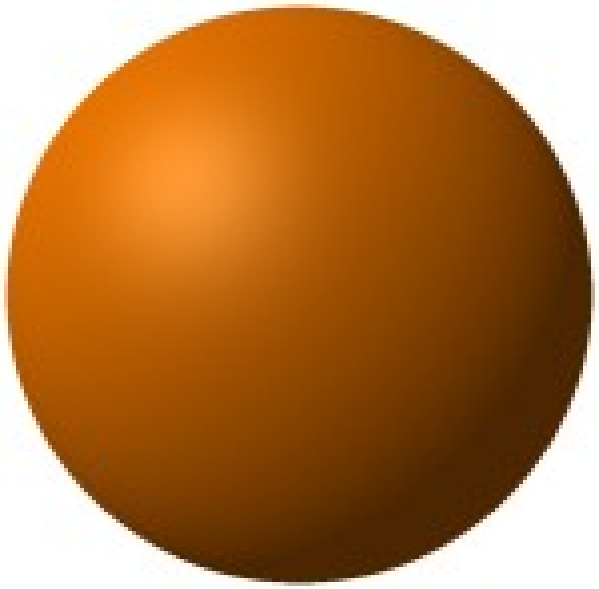
Normal map

Éclairage diffus avec bumps

Bump mapping



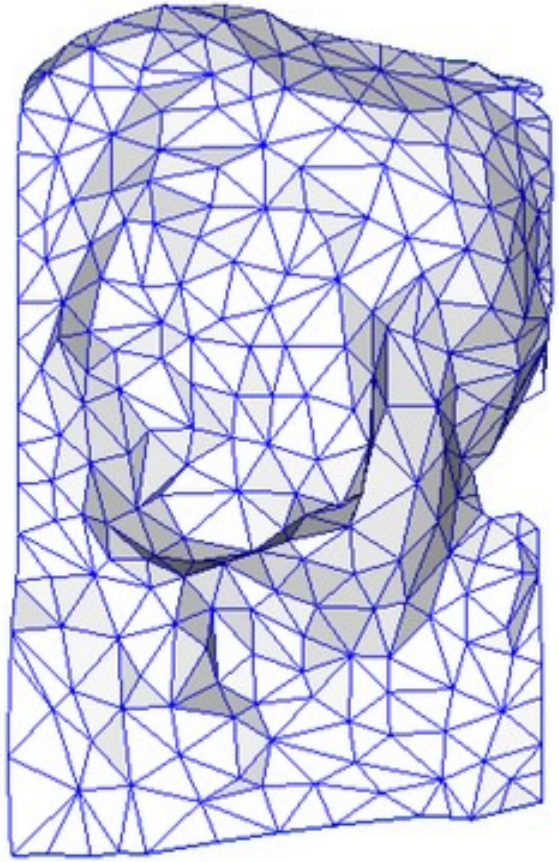




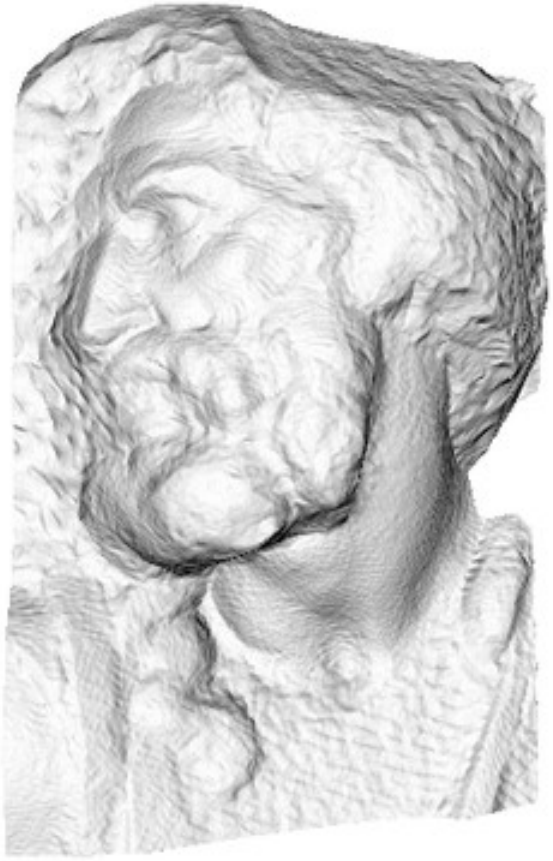
Application: Bump Mapping

Application: Normal Mapping

original mesh
4M triangles

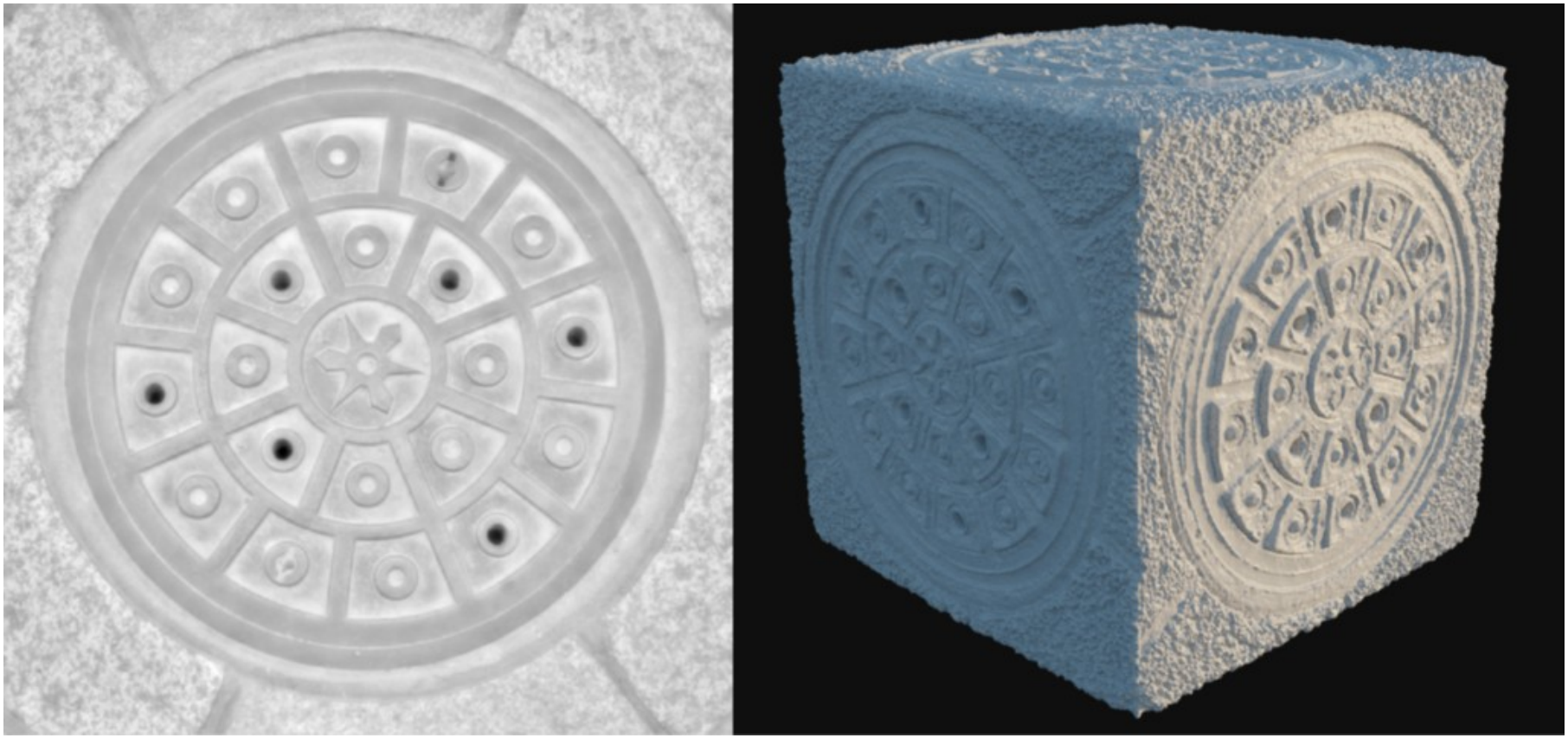


simplified mesh
500 triangles

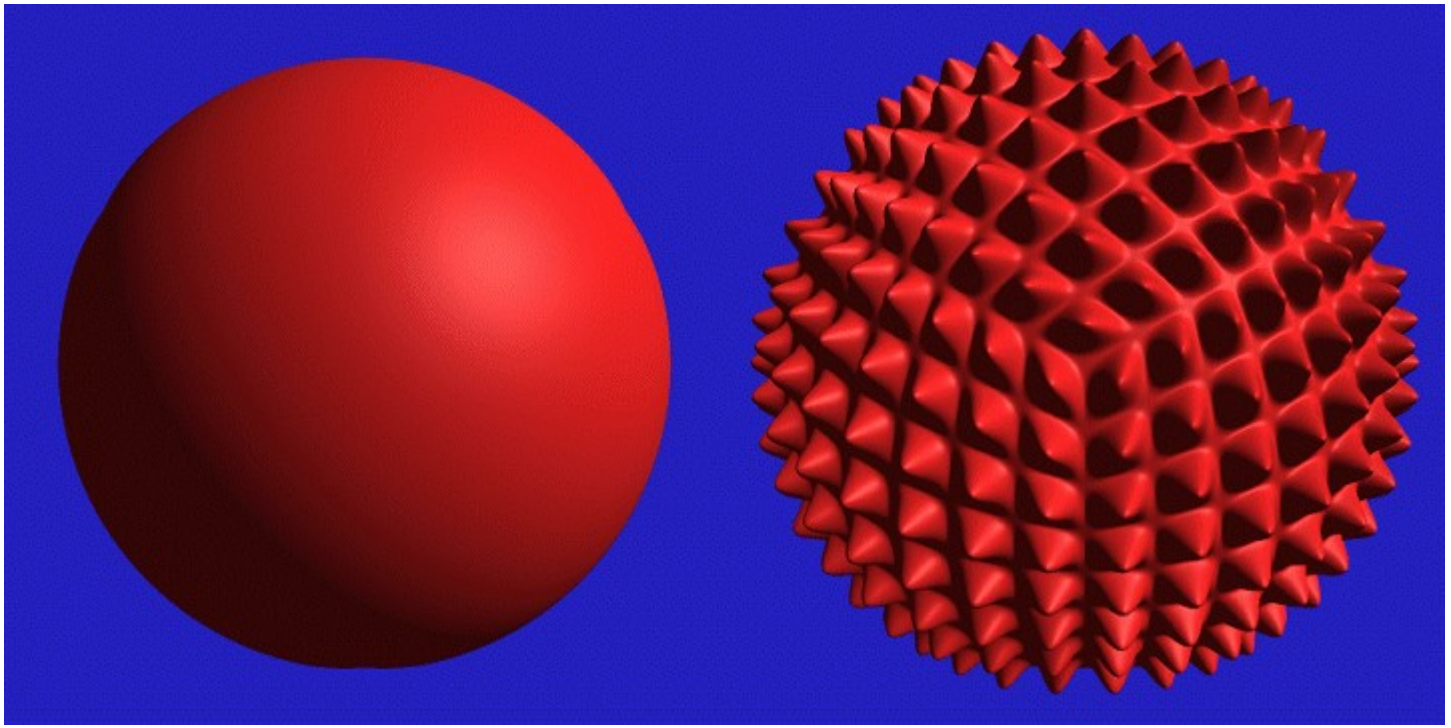


simplified mesh
and normal mapping
500 triangles

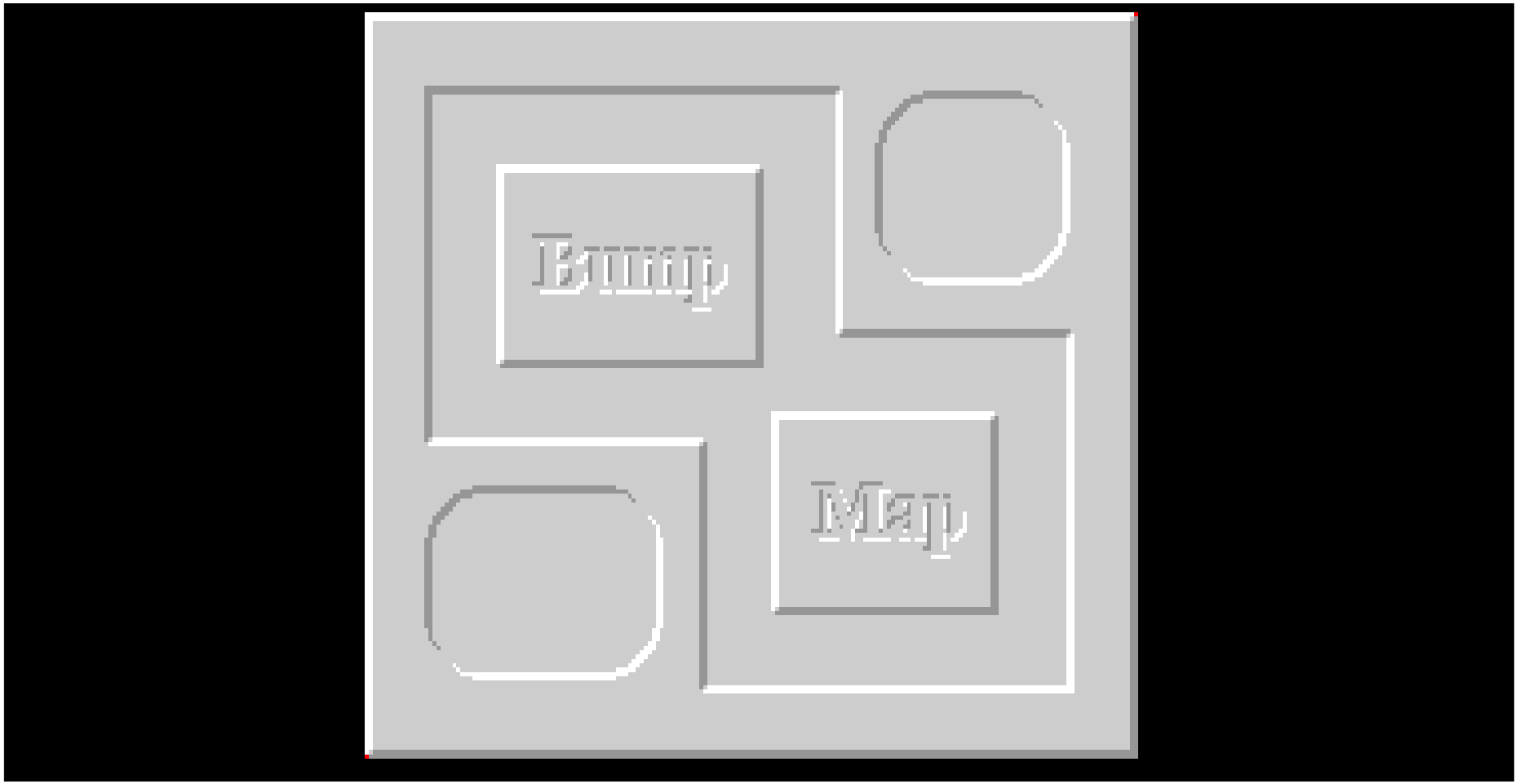
Application: Normal Mapping



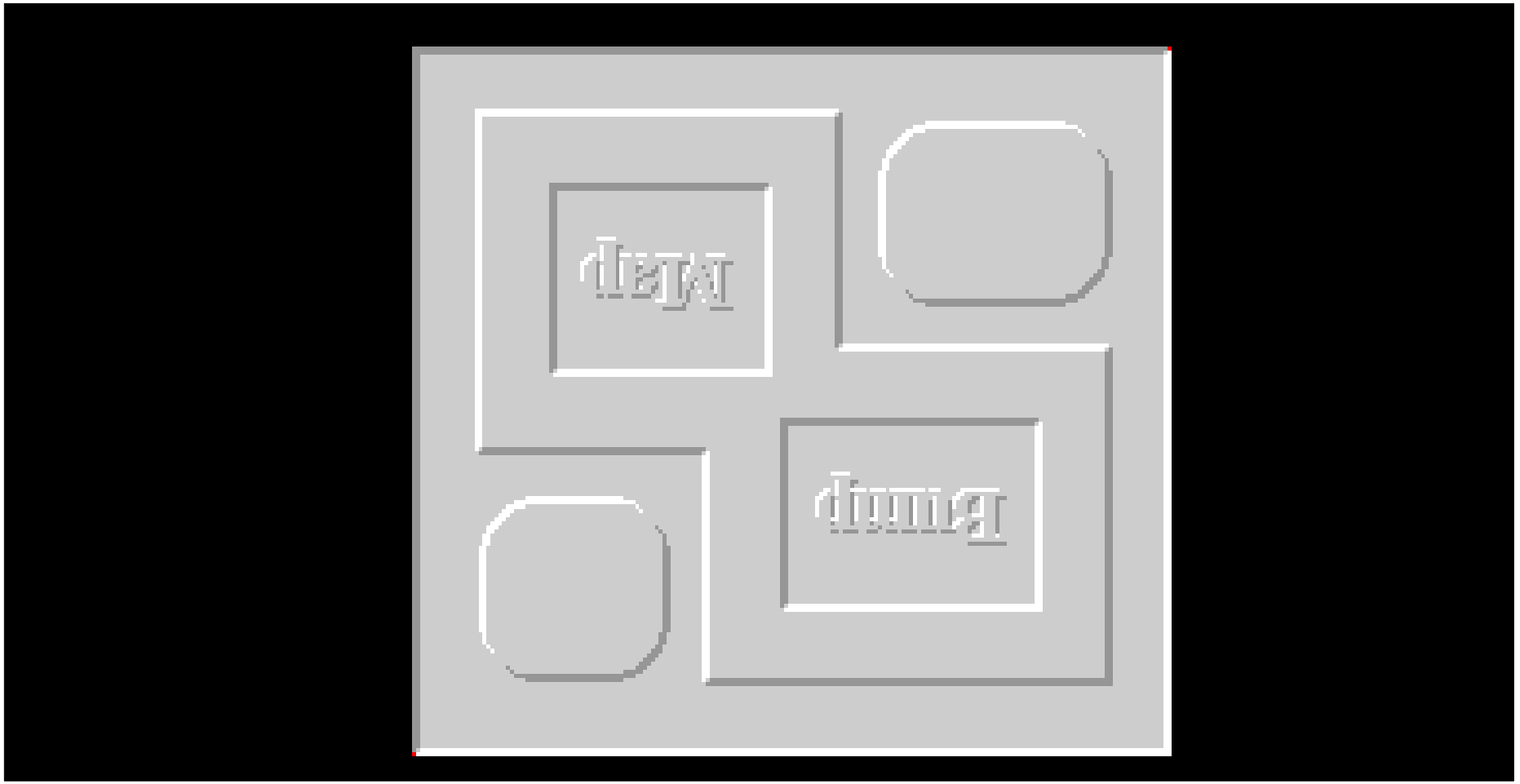
Displacement mapping



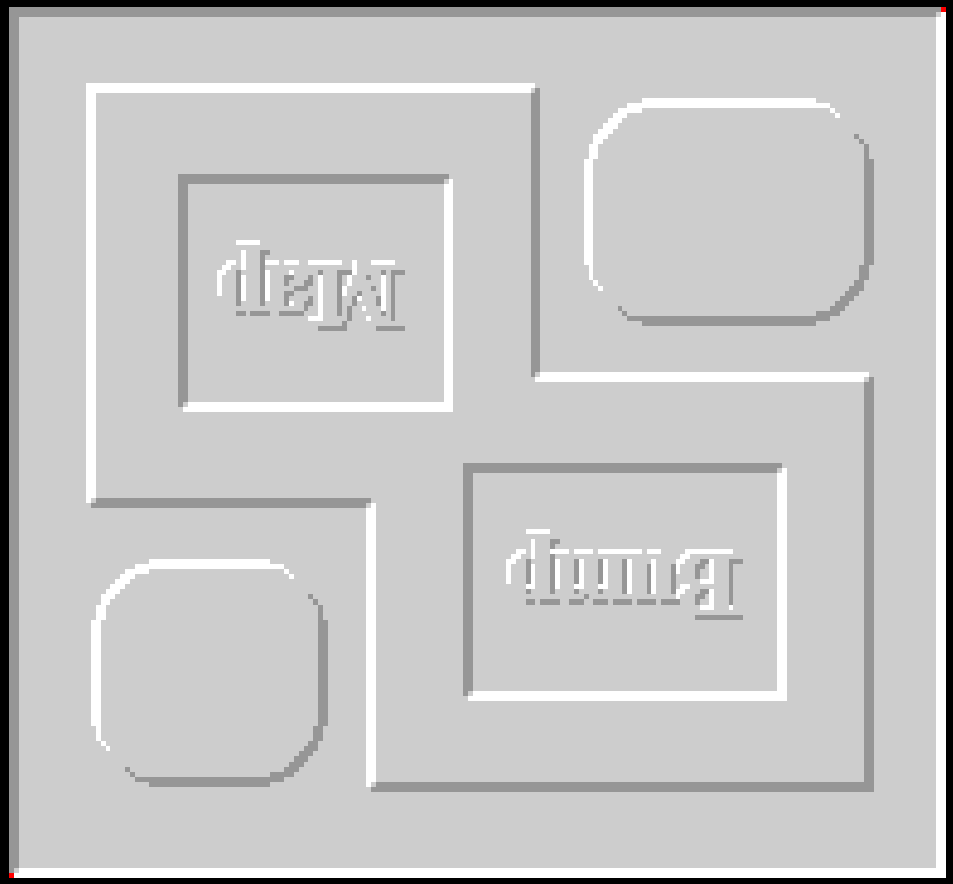
Où sont les creux?



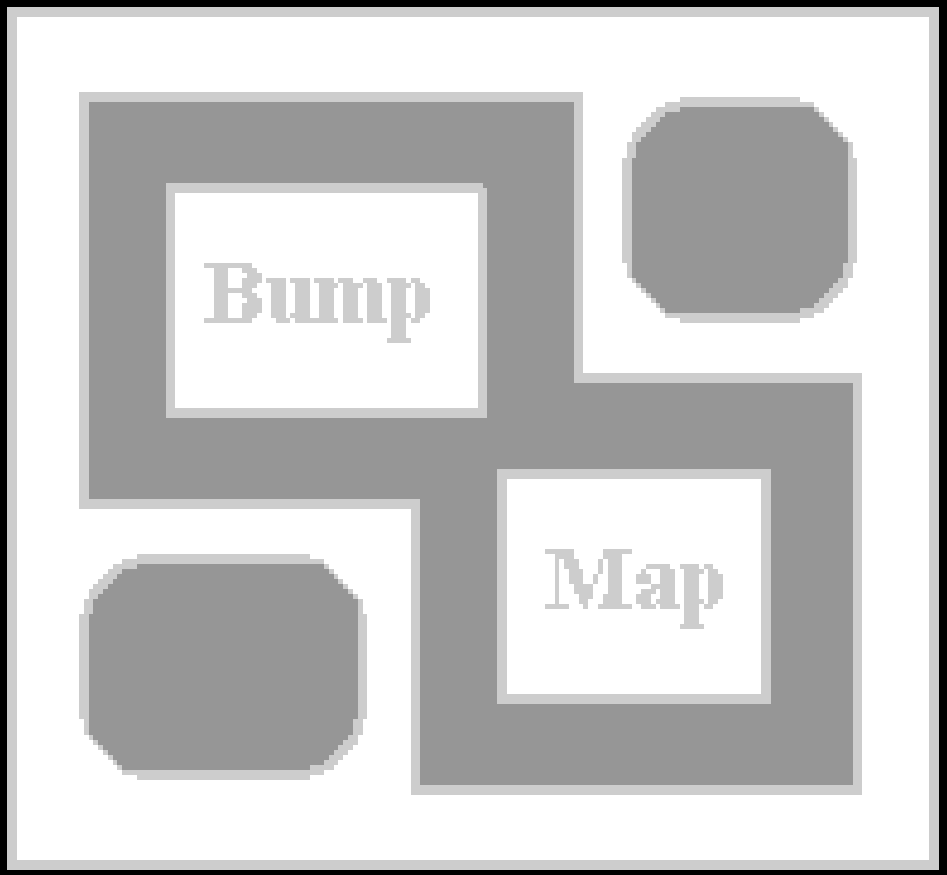
Où sont les ceux?



...et si la lumière vient du bas?

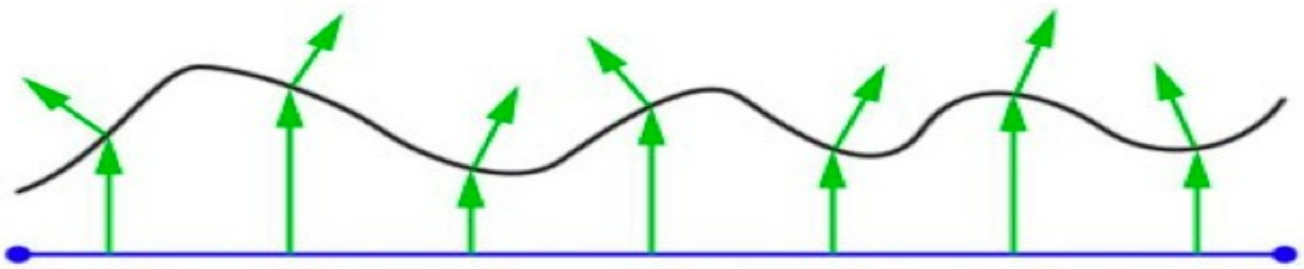


bump map = height field

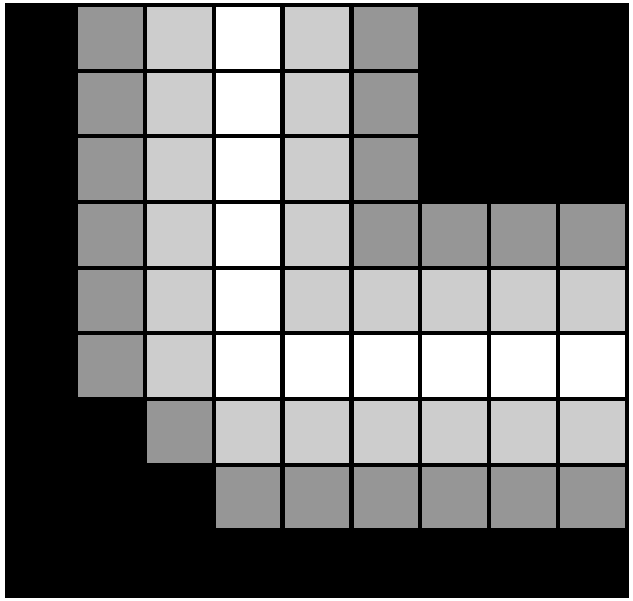


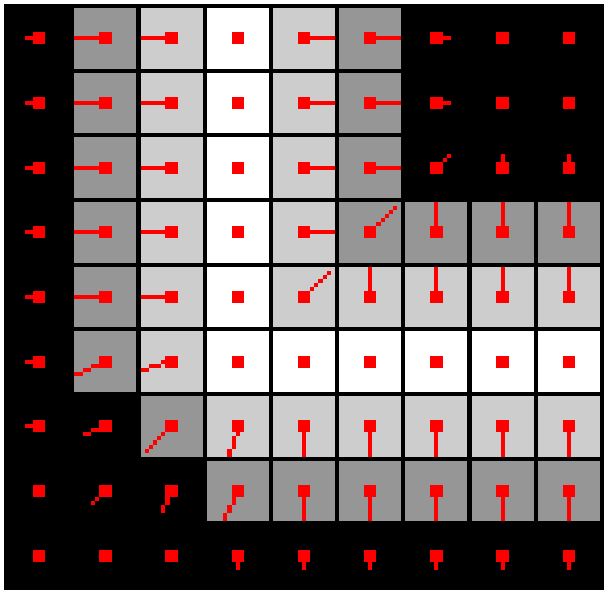
Bump mapping

- Bump Mapping = perturbation de la normale en direction et amplitude
- Usage de texture pour savoir comment déplacer les normales.
- Les déplacements sont calculés par rapport au dérivées des textures
- Présupposés:
  - Les déplacements sont faibles (ils ne faudrait pas avoir à re-normaliser)
  - Les déplacements sont dans le plan tangent à la surface
  - La texture est plaquée sur les plans tangents à la surface



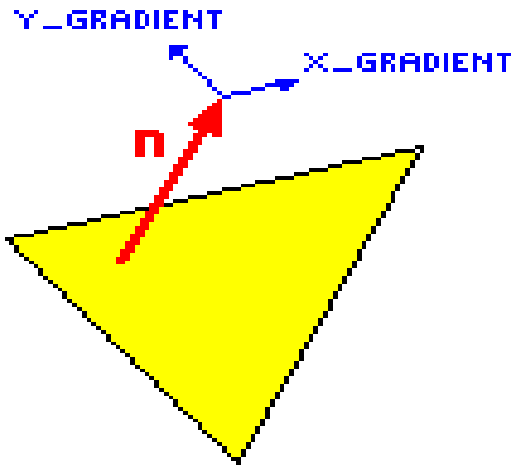
Exploitation de la bump map

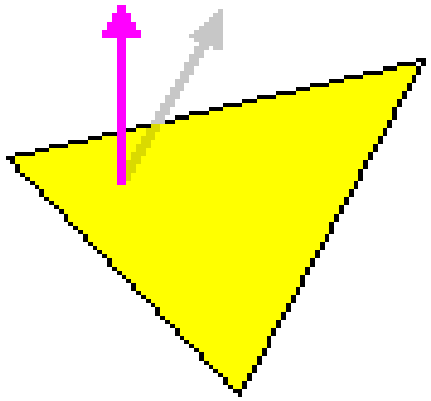




```
x_gradient = pixel(x-1, y) - pixel(x+1, y)
y_gradient = pixel(x, y-1) - pixel(x, y+1)
```

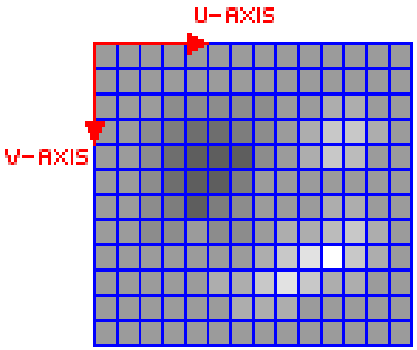
Exploitation de la bump map

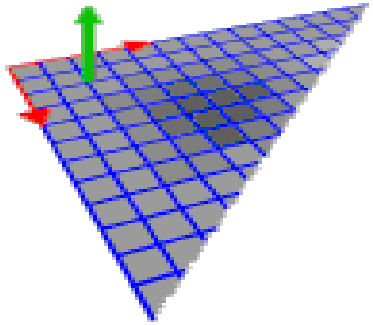




```
New_Normal = Normal + (U * x_gradient) + (V * y_gradient)
```

U et V sont les vecteurs du système des coordonnées de textures

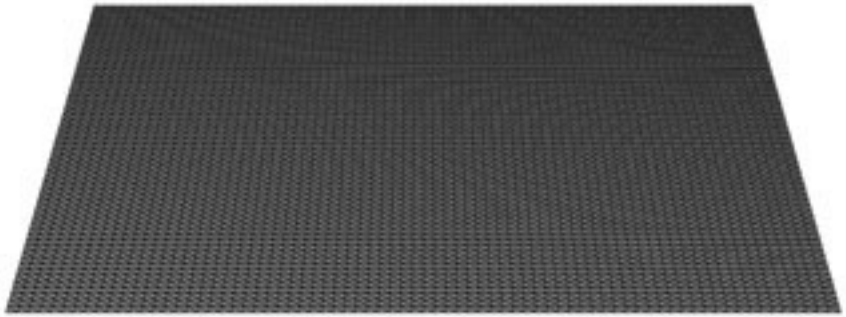




Displacement mapping

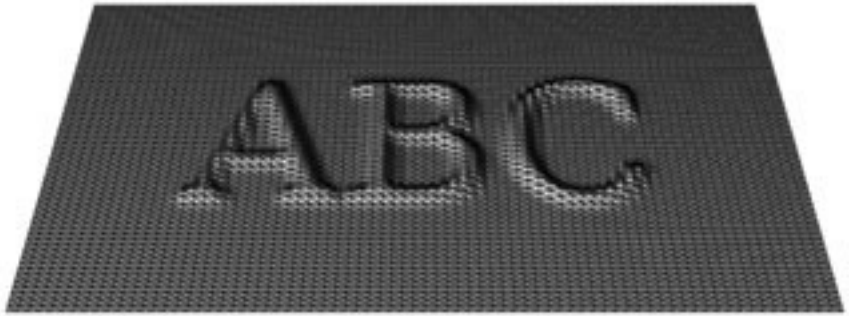
Displacement mapping :
- Changement de la géométrie en fonction d'une texture

Supporté par les architectures qui supportent le shade model 3.0



ORIGINAL MESH

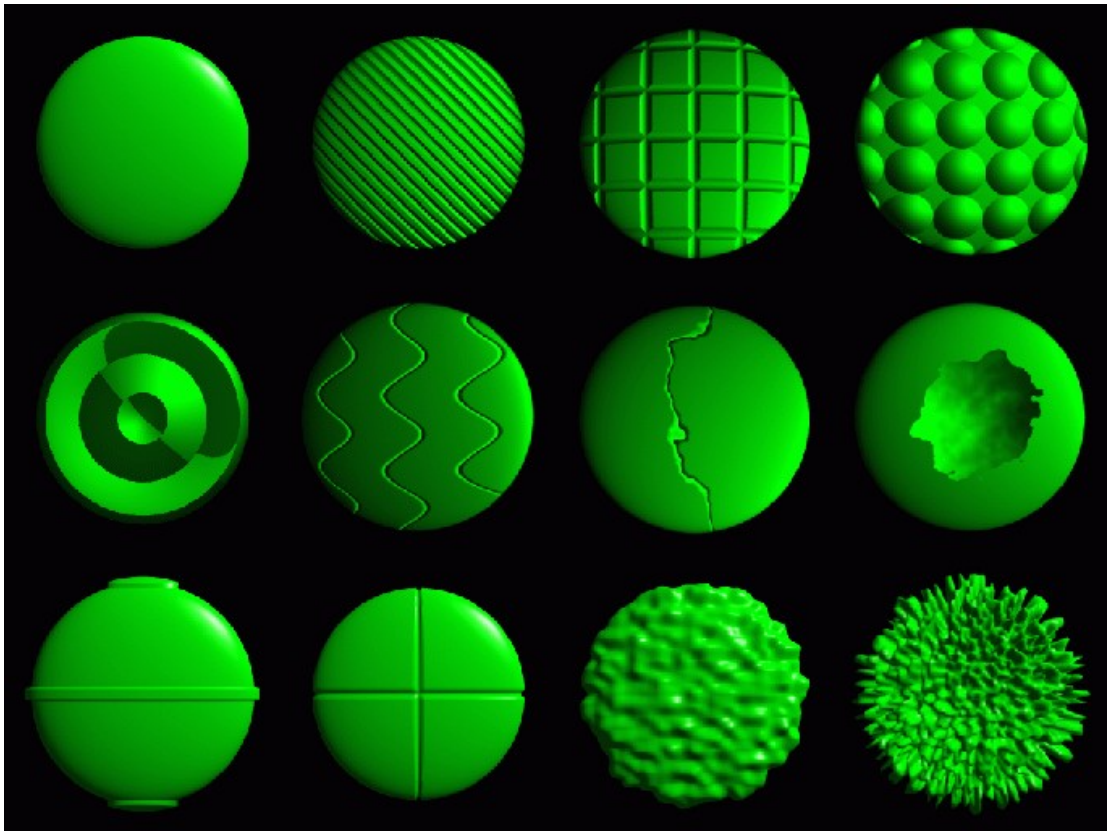
DISPLACEMENT MAP

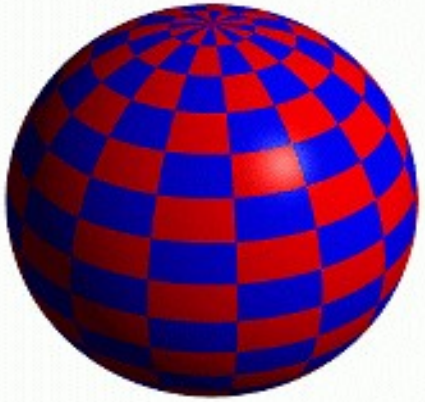


MESH WITH DISPLACEMENT

Effets de surface

- Apparence bosselée, rugueuse …
- Displacement Mapping = bruitage de la surface originale





Effets de surface

bump mapping ***displacement mapping***