MV52
Synthèse d'images

CM #9

Modèles d'illumination simples

Fabrice LAURI
fabrice.lauri@utbm.fr

Plan du cours

- Le système visuel humain
- Les modèles de couleurs
- Les modèles d'illumination
- Les modèles d'ombrage
- Programmation avec shaders

Plan du cours

- Le système visuel humain
- Les modèles de couleurs
- Les modèles d'illumination
- Les modèles d'ombrage
- Programmation avec shaders

Le système visuel humain

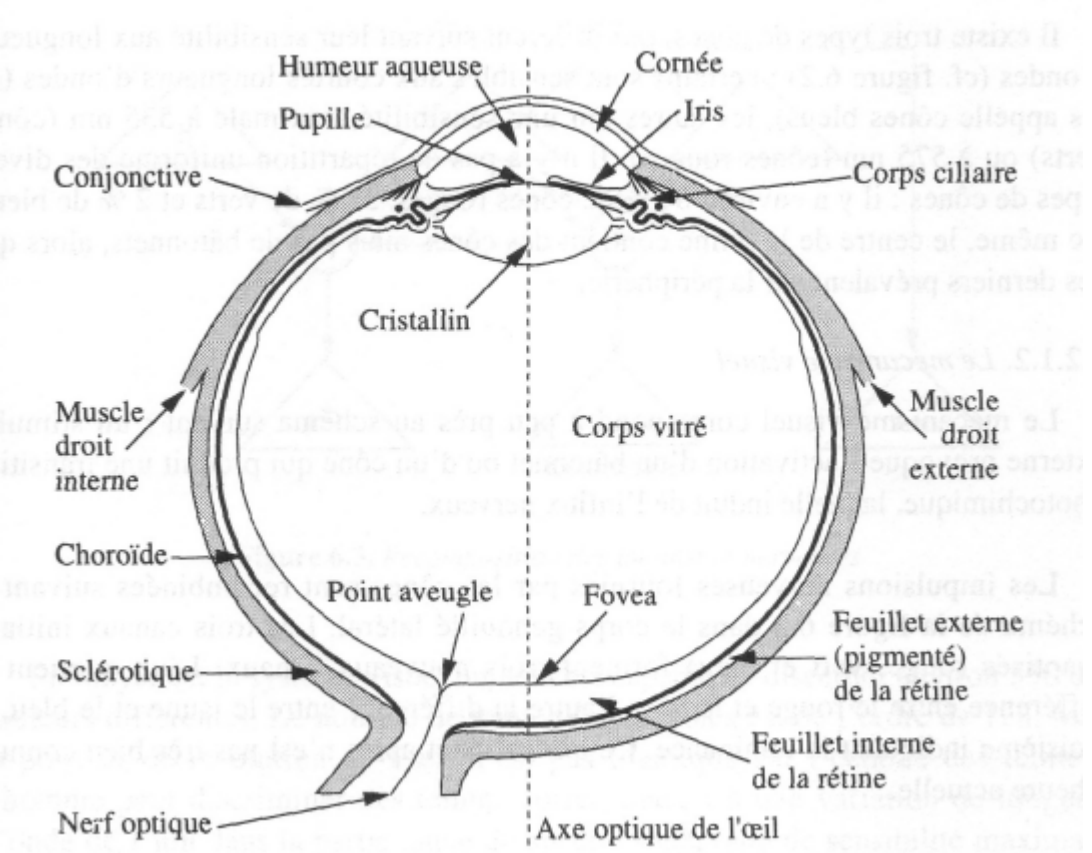
La physique de la lumière

- La lumière est un mélange d'ondes électromagnétiques se propageant à la vitesse de $c=3.10^8$ ms^{-1}.
- Chaque composante du mélange d'ondes, appelée monochromatique, a une période T, une fréquence f et une longueur d'onde λ liées par les relations $\lambda = c.T$ et $T = 1/f$.
- Le photon est le corpuscule associé à l'onde électromagnétique. Une source lumineuse est définie par une distribution spectrale d'énergie. Le système visuel humain interprète l'énergie électromagnétique comme de la lumière visible entre 380 et 780 nm environ. Mais il est important de noter que l'énergie électromagnétique n'a pas de couleur.

Le système visuel humain

L'œil

– L'iris joue le rôle du diaphragme de l'appareil photographique et la pupille celui de la lentille. L'image se forme sur la rétine qui est la surface de l'œil sensible à la lumière. Les muscles ciliaires influent sur la forme de la pupille, permettant la convergence et la focalisation des rayons lumineux sur la rétine.



Le système visuel humain

L'œil

– La lentille absorbe deux fois plus d'ondes dans les fréquences correspondant au bleu que dans celles correspondant au jaune ou au rouge.
Nous sommes beaucoup plus sensibles aux grandes longueurs d'ondes (jaunes et orange) qu'aux faibles (cyan et bleu).

– Avec l'âge, la pupille et le liquide du corps vitré jaunissent.
Ceci explique notamment qu'il existe une grande variation entre individus pour la sensation de couleur (qualia).

Le système visuel humain

Le système visuel humain

– Sur la rétine, il existe deux types de cellules photo-réceptrices qui transforment la lumière en impulsions nerveuses :
  - les cônes (environ 6 millions) et
  - les bâtonnets (environ 120 millions).

  Les bâtonnets sont plus sensibles à la lumière que les cônes et permettent la vision nocturne. Les cônes servent à la vision des couleurs et à la vision diurne.
– Il existe trois types de cônes, qui diffèrent suivant leur sensibilité aux longueurs d'ondes.
– Le centre de la rétine contient des cônes mais pas de bâtonnets, alors que ces derniers prévalent sur la périphérie.

Le système visuel humain

Le mécanisme visuel

– Un stimulus externe provoque l'activation d'un bâtonnet ou d'un cône, ce qui produit une transition photochimique, laquelle induit de l'influx nerveux.

– Les impulsions nerveuses fournies par les cônes sont recombinées dans le corps genouillé latéral. Ces trois canaux initiaux (baptisés rouge, vert et bleu) forment trois nouveaux canaux.

L'un contient la différence entre le rouge et le vert, l'autre la différence entre le jaune et le bleu, le troisième indiquant la luminance.

Ce qui survient après n'est pas très bien connu à l'heure actuelle.

Le système visuel humain

La perception de la couleur

C'est un phénomène subjectif et personnel. La sensation de couleur dépend des paramètres suivants :

- La luminance (pour une source) ou la clarté (pour un objet) : c'est l'énergie d'onde rayonnant jusqu'à l'œil.
- La teinte, qui est la composante de base de la couleur : c'est la longueur d'onde dominante.
- La saturation : une couleur saturée est une couleur proche de la couleur monochromatique de même teinte. Cette notion correspond à la largeur du spectre : plus la bande du spectre est étroite et plus la couleur est saturée.

Le système visuel humain

La perception de la couleur

– On peut percevoir la lumière de deux manières différentes :

- soit directement à partir d'une source lumineuse,
- soit indirectement par réflexion sur la surface d'un objet ou par réfraction à travers un objet.

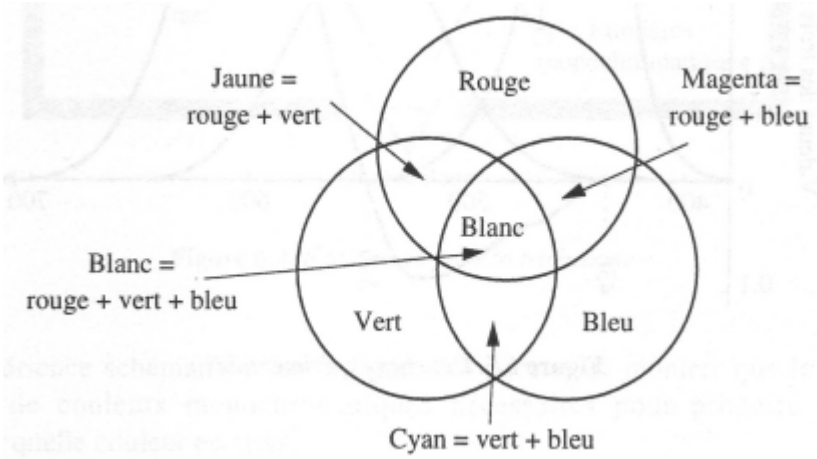
– Ces deux types de perception conduisent à deux méthodes de combinaison de couleurs :

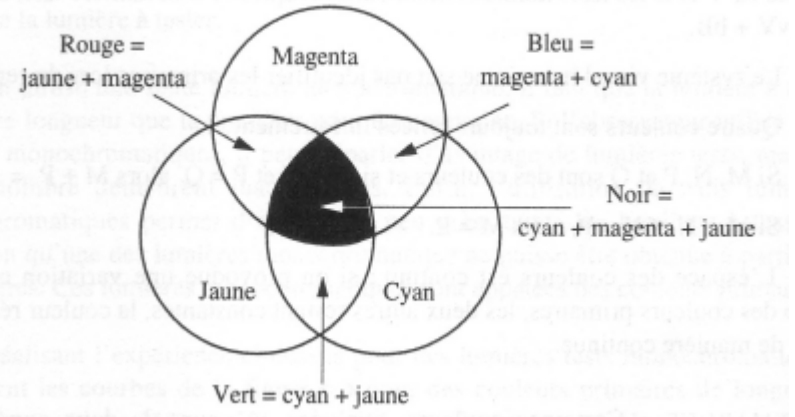
- sur un moniteur vidéo (qui est une source lumineuse), une couleur est obtenue par synthèse additive, par juxtaposition de luminophores.
- Par contre, sur une imprimante, une couleur est obtenue par synthèse soustractive.

Le système visuel humain

La perception des couleurs

- La colorimétrie (ou science de la couleur) définit trois teintes fondamentales, appelées primaires additives : le rouge, le vert et le bleu.
- En mélangeant deux de ces trois teintes, on obtient la couleur complémentaire de la troisième primitive.
- Ceci peut se traduire aussi en soustrayant une couleur d'une autre pour en obtenir une troisième.
  C'est la raison pour laquelle les trois couleurs cyan, jaune et magenta sont appelées les trois primaires soustractives.





Plan du cours

- Le système visuel humain
- Les modèles de couleurs
- Les modèles d'illumination
- Les modèles d'ombrage
- Programmation avec shaders

Les modèles de couleurs

- Contrairement aux idées reçues, le noir et le blanc ne sont pas des couleurs, mais des « compositions » de couleurs.
Le noir est l'absence de couleurs.
Le blanc est un mélange de l'ensemble des couleurs.

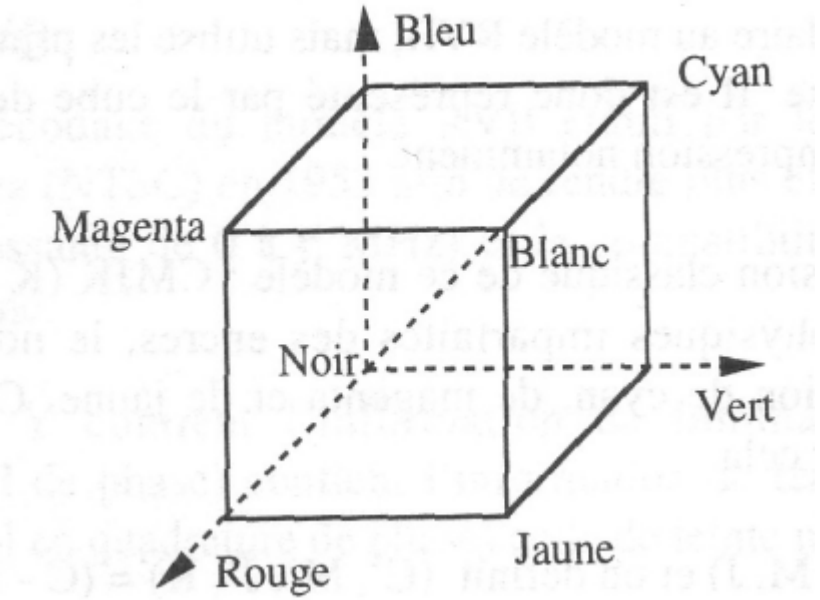
- En informatique, différents modèles ont dû être développés afin de répondre à tous les cas de figure.
En effet, les supports d'affichage diffèrent du tout au tout : blanc pour une feuille de papier, noir pour un écran…

Les modèles de couleurs

Le modèle RGB

– C'est un modèle additif utilisé principalement sur les écrans informatiques (support noir).

Il utilise trois couleurs primaires : le rouge (red), le vert (green) et le bleu (bleu) pour définir un repère cartésien.

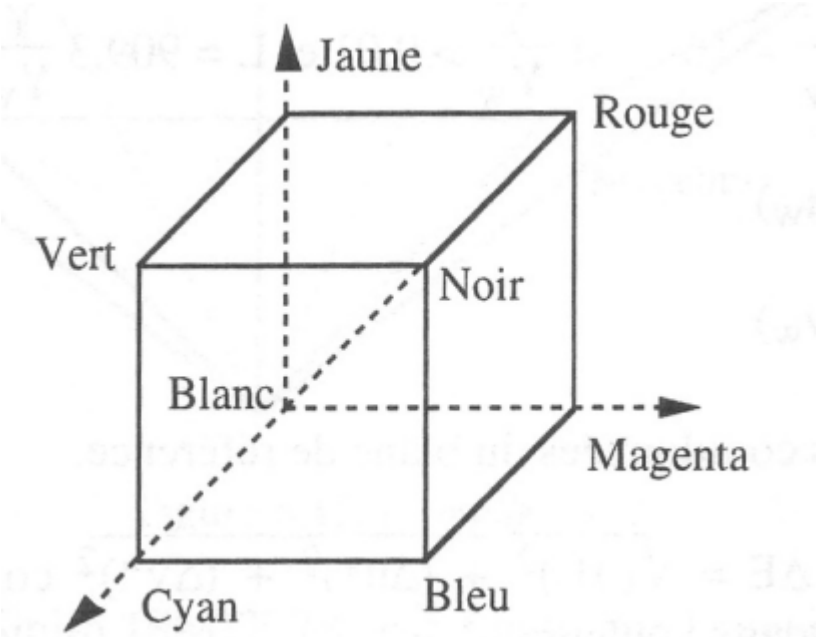


Les modèles de couleurs

Le modèle CMY

– Le modèle Cyan, Magenta, Yellow est le complémentaire du RGB.

Il est donc soustractif et est principalement utilisé pour les impressions (les feuilles de papier étant généralement blanches).

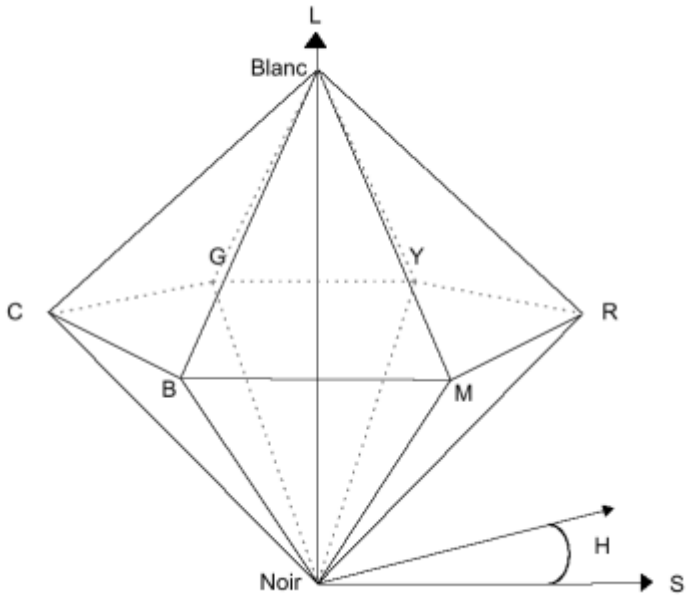


Les modèles de couleurs

Le modèle HLS

– Ce modèle est basé sur la teinte (Hue), la brillance (Lightness) et la saturation (Saturation) de la couleur.

Ainsi, la teinte définit la couleur de base, la brillance correspond à l'intensité de la couleur et la saturation à son degré de pureté.



Plan du cours

- Le système visuel humain
- Les modèles de couleurs
- Les modèles d'illumination
- Les modèles d'ombrage
- Programmation avec shaders

Les modèles d'illumination

Préambule

En infographie, deux modèles sont utilisés conjointement :

- Un **modèle d'illumination**, qui permet de déterminer la couleur d'un point de la surface.
- Un **modèle d'ombrage**, qui détermine la manière d'appliquer le modèle d'illumination sur les surfaces (par fragment, ou par interpolation de certaines valeurs calculées au niveau des vertices).

Les modèles d'illumination

Les modèles d’illumination utilisés dans le domaine du graphisme sont inspirés de modèles physiques plus ou moins simplifiés et adaptés à l’usage informatique.

La validation de ces modèles découlent autant de l’usage pratique que d’un raisonnement mathématique.

Nous allons tout d’abord voir des modèles d’illumination simples ne prenant pas en compte les interactions entre objets mais seulement les interactions entre une surface et les sources lumineuses.

Les modèles d'illumination

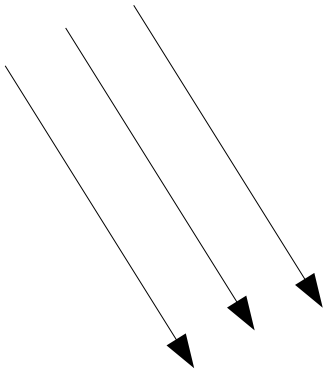
Plusieurs modèles d'illumination peuvent être combinés pour prendre en compte les interactions existant entre :

- différents **types de lumière** et
- différents **types de surfaces**.

Les modèles d'illumination (types de lumières)

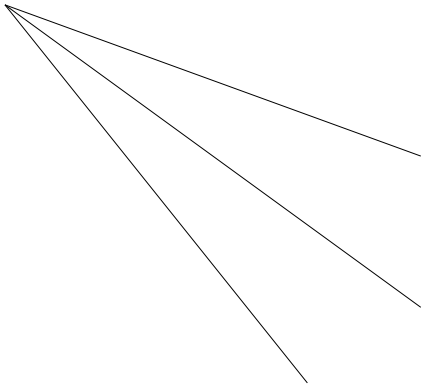
Ces types de lumière peuvent être :

Lumière ambiante



Lumière directionnelle

Lumière localisée



Spot

Les modèles d'illumination (types de lumières)

Lumière ambiante (*ambiant light*)

Source lumineuse non ponctuelle qui émet une lumière d'intensité constante, dans toutes les directions et sur toutes les surfaces.

Lumière directionnelle

Source lumineuse non ponctuelle qui émet une lumière selon une direction donnée.

Les modèles d'illumination (types de lumières)

Lumière localisée (*point light*)

Source lumineuse ponctuelle qui émet une lumière dans toutes les directions.

L'intensité lumineuse reçue par la surface d'un objet dépend de sa distance à la source de lumière.

Spot (*spot light*)

Source lumineuse ponctuelle qui émet une lumière dans toutes les directions à l'intérieur d'un cône.

L'intensité lumineuse reçue par la surface d'un objet dépend de sa distance à la source de lumière.

Les modèles d'illumination (types de surfaces)

Les types de surfaces définissent les différentes propriétés lumineuses que peut posséder le matériau avec lequel est fabriqué un objet ou une partie d'un objet.

On peut distinguer au moins deux types de surfaces :
– les surfaces diffuses (aussi appelées surfaces Lambertiennes)
– les surfaces spéculaires (ou réfléchissantes)

Les surfaces Lambertiennes
– Surfaces mates (neige, papier) présentant une réflexion diffuse (ou réflexion Lambertienne).
– Surfaces apparaissant comme uniformément brillantes quelle que soit la direction d'observation.
Ceci est dû au fait que l'intensité en un point de la surface dépend uniquement de l'angle entre la normale à la surface et la direction du point à la source de lumière.

Les modèles d'illumination (types de surfaces)

Les surfaces spéculaires

– Surfaces présentant des tâches plus claires se déplaçant en fonction du point d'observation.
– Ces tâches reflètent plus la lumière provenant de la source que ne le fait le reste de la surface.

Calculs d'illumination

L'illumination d'un point d'une surface avec OpenGL prend en compte des :

- Composantes ambiante, diffuse et spéculaire des sources lumineuses
- Composantes ambiante, diffuse et spéculaire du matériel appliqué sur la surface

Ces différentes composantes permettent à une source (respectivement au matériau d'une surface) de générer ou non ces types de réflexion (ambiante, diffuse ou spéculaire).

Une composante = vecteur 4D de couleur

Les vecteurs indispensables lors du calcul d'illumination

Intensité lumineuse en un point P d'une surface :

$$I_P = M_{amb} \otimes I_{amb} + M_{diff} \otimes I_{diff} + M_{spec} \otimes I_{spec}$$

avec :

I_{amb} , I_{dif} , I_{spe} : contributions ambiante, diffuse et spéculaire des sources lumineuses sur la surface.

M_X : Composante ambiante, diffuse et spéculaire du matériau

$\otimes$: multiplication vectorielle composante par composante.

Les vecteurs indispensables lors du calcul d'illumination

Tels que :

$$I_{amb} = \sum_i S^i_{amb}$$

$$I_{diff} = \sum_i S^i_{diff}\, max\left(\left(N.L_i\right), 0\right)$$

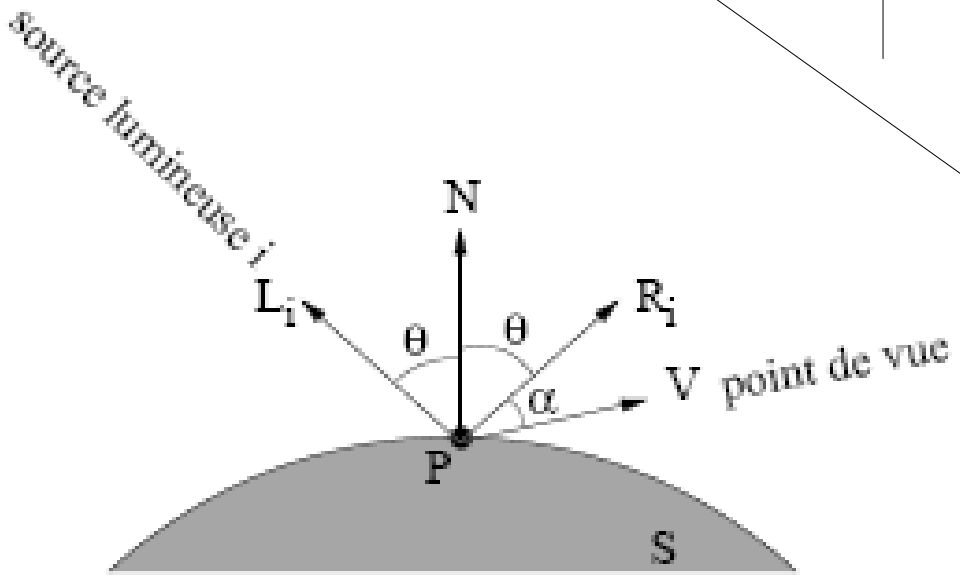
$$I_{spec} = \sum_i S^i_{spec}\, max\left(\left(V.R_i\right)^{M_{shi}}, 0\right)$$

ou

$$I_{spec} \approx \sum_i S^i_{spec}\, max\left(\left(N.H_i\right)^{4M_{shi}}, 0\right)$$

S^i_X : Composante ambiante, diffuse ou spéculaire de la source i

M_{shi} : Brillance du matériau (*shininess*)



Halfway Vector
$(V+L_i)/||V+L_i||$

Calculs d'illumination pour les lumières localisées et les spots

Etant donné que pour ces types de lumières l'intensité lumineuse reçue par une surface dépend de sa distance à la source, un facteur d'atténuation doit être calculé pour pondérer les contributions diffuses et spéculaires.

Facteur d'atténuation :

$$att_i = \frac{1}{S_c^i + S_l^i d\left(S_{pos}^i, P\right) + S_q^i d\left(S_{pos}^i, P\right)^2}$$

P : position du point
S^i_{pos} : position de la source

Pour des résultats proches de la réalité :

$$S_c^i = 0 \qquad S_l^i = 0 \qquad S_q^i = 1$$

Calculs d'illumination pour les lumières localisées et les spots

Intensité lumineuse en P en tenant compte de l'atténuation :

$$I_{diff} = \sum_i S^i_{diff} \, max\left(\left(N.L_i\right), 0\right) att_i$$

$$I_{spec} = \sum_i S^i_{spec} \, max\left(\left(V.R_i\right)^{M_{shi}}, 0\right) att_i$$

Calculs d'illumination pour les spots

Si P est dans le cône de lumière du spot i :

$$c^i_{spot} = max\left(-L_i . S^i_{dir}, 0\right)^{S^i_{\exp}}$$

sinon :

$$c^i_{spot} = 0$$

Intensité lumineuse en P pour les spots i :

$$I_{amb} = \sum_i c^i_{spot} S^i_{amb}$$

$$I_{diff} = \sum_i c^i_{spot} S^i_{diff} max\left(\left(N.L_i\right), 0\right) att^i$$

$$I_{spec} = \sum_i c^i_{spot} S^i_{spec} max\left(\left(V.R_i\right)^{M_{shi}}, 0\right) att^i$$

Plan du cours

- Le système visuel humain
- Les modèles de couleurs
- Les modèles d'illumination
- Les modèles d'ombrage
- Programmation avec shaders

Les modèles d'ombrage

La méthode venant tout de suite à l'esprit pour effectuer le rendu d’une surface consiste à calculer l’illumination en chaque point visible de la surface.

Cette méthode est extrêmement coûteuse en temps de calcul.

Nous allons voir différents modèles d'ombrage permettant de diminuer le coût en terme de temps, en n’effectuant le calcul d’illumination qu’en un nombre limité de points.

Le modèle d'ombrage plat

Ombrage plat

Méthode d'ombrage la plus simple pour les facettes polygonales.

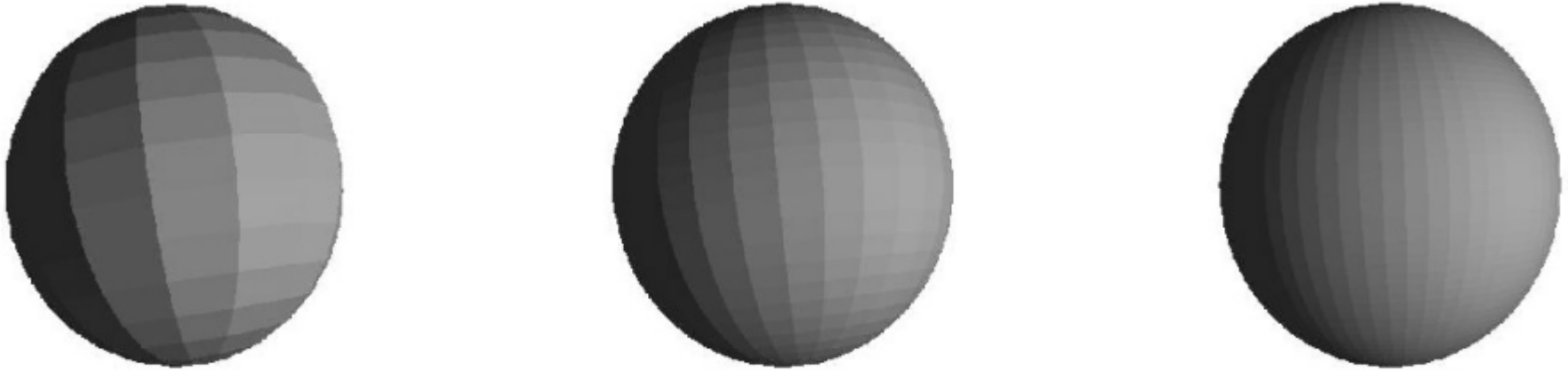
L'idée est de calculer une seule valeur d'illumination pour l'ensemble de la facette. Par exemple au point milieu de la facette en prenant pour normale à la surface celle du plan contenant la facette.

Cette approche est valide par rapport aux modèles d'illumination vus précédemment lorsque :

- La source lumineuse est à l'infini
- La projection est orthographique
- La surface est composée de facettes polygonales uniquement

Le modèle d'ombrage plat

Le principal inconvénient de ce modèle est la forte visibilité des limites des facettes due aux changements brusques d'intensité des couleurs.



Ombrage plat de la sphère pour : 16×16 *facettes,* 32×32 *et* 64×64

Le modèle d'ombrage de *Gouraud*

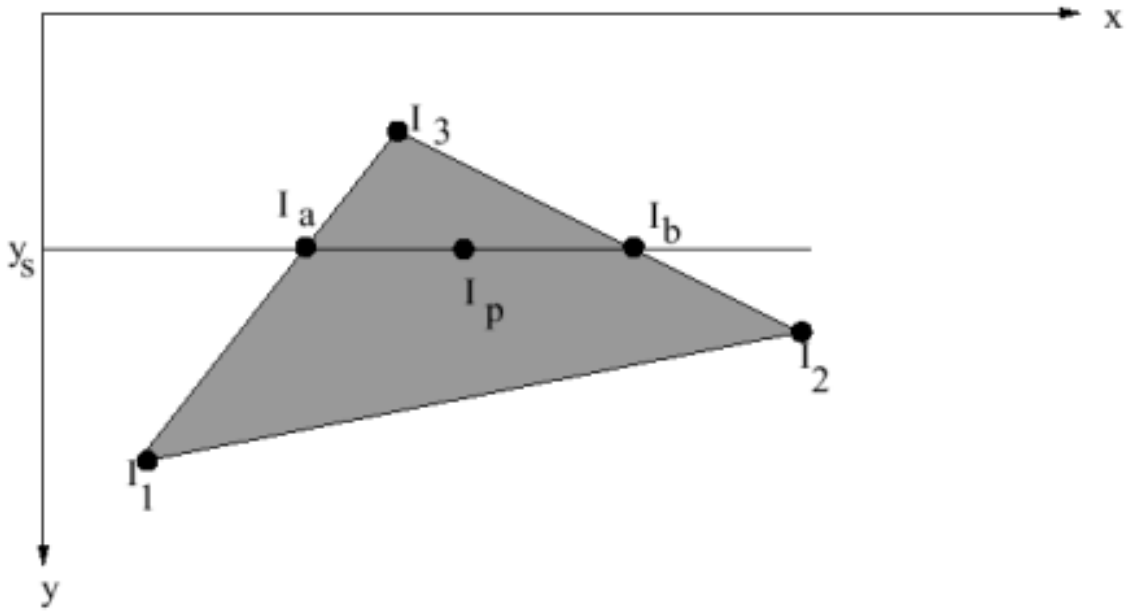
Ombrage de *Gouraud*

La méthode développée par Gouraud [1971] élimine les discontinuités d'intensité sur une facette polygonale par interpolation des valeurs d'intensité aux sommets de la facette. Cette méthode est largement utilisée et se retrouve dans la majorité des matériels graphiques existants (librairies, cartes graphiques…).

Le modèle d'ombrage de *Gouraud*

Ombrage de *Gouraud*

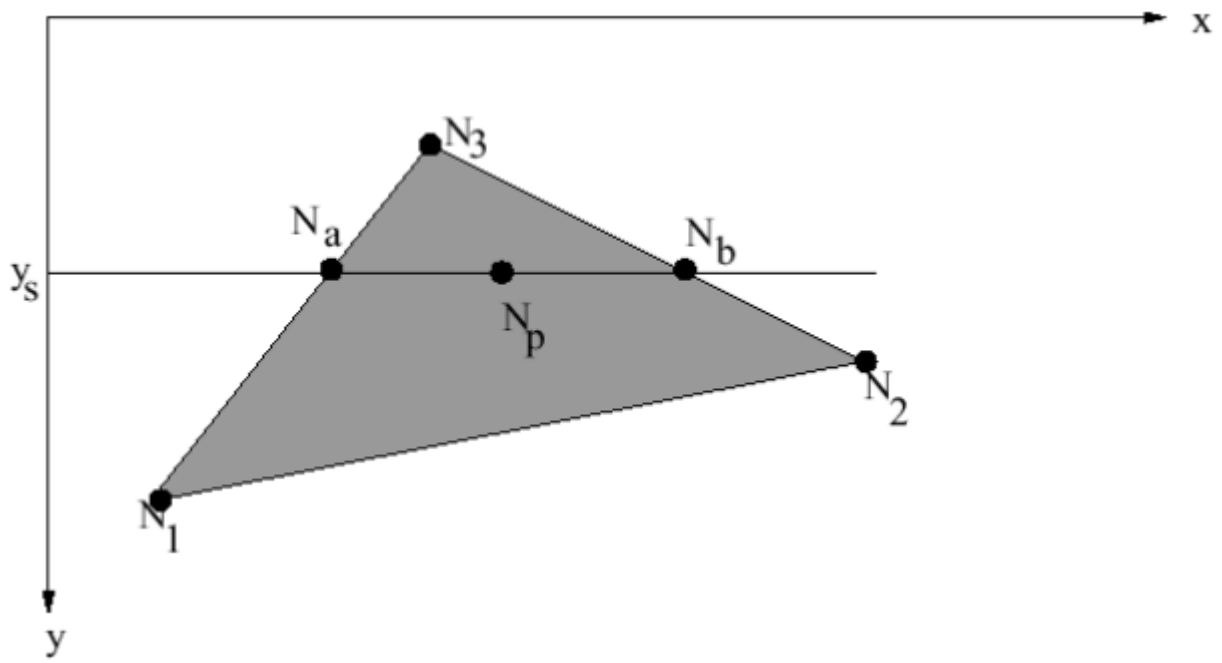
Lorsque les normales sont connues, les intensités aux sommets des facettes polygonales sont calculées.
L'interpolation peut ensuite être effectuée à l'aide de l'algorithme de balayage de lignes utilisé pour le remplissage de polygone et le z-buffer.



Le modèle d'ombrage de *Phong*

Ombrage de *Phong*

L'ombrage de Phong [1975] consiste à déterminer la normale en un point d'une facette polygonale par interpolation des normales aux sommets de cette facette.

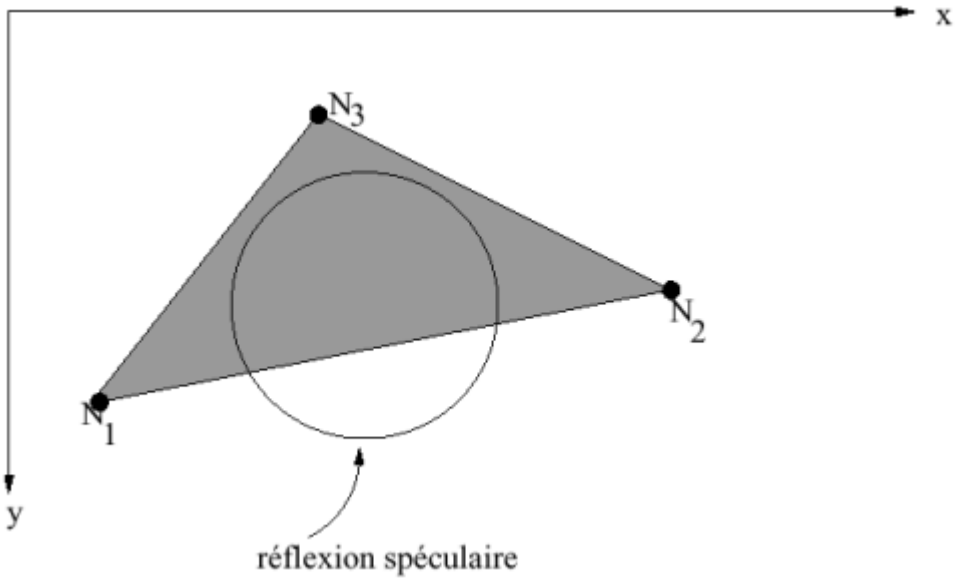


Le modèle d'ombrage de *Phong*

Ombrage de *Phong*

L'intérêt de cette approche par rapport à l'ombrage de *Gouraud* réside principalement dans sa capacité à traiter de manière plus précise les réflexions spéculaires.

Gouraud ne permet pas, en effet, de prendre en compte les réflexions spéculaires lorsque celles-ci sont localisées au centre d'une facette.



Les modèles d'ombrage

Bilan

– L'ombrage de *Phong* est d'une manière générale meilleur que celui de *Gouraud* car l'interpolation est effectuée sur les normales et non les intensités. Cela même pour un modèle d'illumination sans réflexion spéculaire.
– Toutefois, ce modèle augmente le coût d'un rendu.
– Ce modèle est aujourd'hui très simple à programmer avec des shaders.

plat Gouraud Phong

Plan du cours

- Le système visuel humain
- Les modèles de couleurs
- Les modèles d'illumination
- Les modèles d'ombrage
- Programmation avec shaders

Modèle d'ombrage de *Gouraud* par *Vertex Shader*

Variables d'entrées du *Vertex Shader*

- Attributs de vertex
  - Position
  - Normale
- Variables uniformes
  - Matrice MVP Modèle-Vue-Projection
  - Matrice MV Modèle-Vue
  - Matrice $(M^{-1})^T$ transposée inverse de la matrice Modèle
  - Structure de données contenant les paramètres des sources (nombre de sources, positions ou directions de chaque source, composantes de chaque source...)
  - Structure de données contenant les propriétés du matériau (composantes, brillance)

Modèle d'ombrage de *Gouraud* par *Vertex Shader*

Variables d'entrées du *Vertex Shader*

```
in vec4 pos;
in vec4 normal;

uniform mat4 mvp;
uniform mat4 mv;
uniform mat3 normal_matrix;
uniform LightSources lights;
uniform Material material;
```

Variable de sortie du *Vertex Shader*

```
out vec4 fColor;
```

Modèle d'ombrage de *Gouraud* par *Vertex Shader*

Structures de données

```
const int MNL = 10 ;

struct LightSources
{
    int nbrLights;
    LightSource source[MNL];
};

struct Material
{
    vec4 ambient;
    vec4 diffuse;
    vec4 specular;
    float shininess;
};
```

```
struct LightSource
{
    vec4 posDir;
    vec4 ambient;
    vec4 diffuse;
    vec4 specular;
    float constantAttenuation;
    float linearAttenuation;
    float quadraticAttenuation;
    vec3 spotDirection;
    float spotCutoff;
    float spotExponent;
};
```

Position si posDir.w = 1
Direction si posDir.w = 0

Attention, les positions des sources lumineuses sont supposées être exprimées dans l'espace Caméra.

Modèle d'ombrage de *Gouraud* par *Vertex Shader*

```
void main()
{
  vec3 sp = vec3( mv*pos );
  vec3 V = -normalize( sp );
  vec3 unit_normal = normalize( normal_matrix*normal.xyz );

  vec4 amb = vec4(0.0);  vec4 diff = vec4(0.0);  vec4 spec = vec4(0.0);

  for( int i = 0; i < lights.nbrLights; i++ )
  {
    if (lights.source[i].posDir.w == 0.0)
      DirectionalLight(i, V, unit_normal, amb, diff, spec);
    else if (lights.source[i].spotCutoff == 180.0)
      PointLight(i, V, sp, unit_normal, amb, diff, spec);
    else
      SpotLight(i, V, sp, unit_normal, amb, diff, spec);
  }

  fColor = material.ambient * amb + material.diffuse * diff + material.specular * spec;
  gl_Position=mvp*pos;
}
```

Modèle d'ombrage de *Gouraud* par *Vertex Shader*

```
void DirectionalLight( in int i, in vec3 V, in vec3 normal,
                       inout vec4 ambient, inout vec4 diffuse, inout vec4 specular)
{
  float nDotLi;    // normal . light direction
  float nDotH;     // normal . light half vector
  float pf;        // power factor
  vec3 halfway_vector = normalize(V+lights.source[i].posDir.xyz);

  nDotLi = max(0.0, dot(normal, normalize(vec3(lights.source[i].posDir))));
  if (nDotLi == 0.0)
    pf = 0.0;
  else
  {
    nDotH = max(0.0, dot(normal, halfway_vector));
    pf = pow(nDotH, 4*material.shininess);
  }
  ambient += lights.source[i].ambient;
  diffuse += lights.source[i].diffuse * nDotLi;
  specular += lights.source[i].specular * pf;
}
```

Modèle d'ombrage de *Gouraud* par *Vertex Shader*

```
void PointLight( in int i, in vec3 V, in vec3 sp, in vec3 normal,
                 inout vec4 ambient, inout vec4 diffuse, inout vec4 specular )
{
  float nDotLi;              // normal . light direction
  float nDotH;               // normal . light half vector
  float pf;                  // power factor
  float attenuation;         // computed attenuation factor
  float d;                   // distance from surface to light source
  vec3 L;                    // direction from surface to light position
  vec3 halfway_vector;       // direction of maximum highlights

  L = lights.source[i].position.xyz - sp;
  d = length( L );
  L = normalize( L );

  attenuation = 1.0 / (lights.source[i].constantAttenuation +
                       lights.source[i].linearAttenuation * d +
                       lights.source[i].quadraticAttenuation * d * d);

  halfway_vector = normalize(L + V);
```

Modèle d'ombrage de *Gouraud* par *Vertex Shader*

```
  nDotLi = max(0.0, dot(normal, L));
  nDotH = max(0.0, dot(normal, halfway_vector));

  if (nDotLi == 0.0)
    pf = 0.0;
  else
    pf = pow(nDotH, material.shininess);

  ambient += lights.source[i].ambient;
  diffuse += lights.source[i].diffuse * nDotLi * attenuation;
  specular += lights.source[i].specular * pf * attenuation;
}
```

Modèle d'ombrage de *Gouraud* par *Vertex Shader*

```
void SpotLight( in int i, in vec3 V, in vec3 sp, in vec3 normal,
                inout vec4 ambient, inout vec4 diffuse, inout vec4 specular )
{
  float nDotLi;                 // normal . light direction
  float nDotH;                  // normal . light half vector
  float pf;                     // power factor
  float spotDot;                // cosine of angle between spotlight
  float spotAttenuation;        // spotlight attenuation factor
  float attenuation;            // computed attenuation factor
  float d;                      // distance from surface to light source
  vec3 L;                       // direction from surface to light position
  vec3 halfway_vector;          // direction of maximum highlights

  L = lights.source[i].position.xyz – sp;
  d = length( L );
  L = normalize( L );
```

Modèle d'ombrage de *Gouraud* par *Vertex Shader*

```
attenuation = 1.0 / (lights.source[i].constantAttenuation +
                     lights.source[i].linearAttenuation * d +
                     lights.source[i].quadraticAttenuation * d * d);

float cos_cutoff;

// See if point on surface is inside the cone of illumination
spotDot = dot( -L, normalize(lights.source[i].spotDirection) );
cos_cutoff = cos( lights.source[i].spotCutoff ) ;
if (spotDot < cos_cutoff)
  spotAttenuation = 0.0;
else
  spotAttenuation = pow(spotDot, lights.source[i].spotExponent);

attenuation *= spotAttenuation;
```

Modèle d'ombrage de *Gouraud* par *Vertex Shader*

```
  halfway_vector = normalize( L + V );

  nDotLi = max( 0.0, dot(normal, L) );
  nDotH = max( 0.0, dot(normal, halfway_vector) );

  if (nDotLi == 0.0)
    pf = 0.0;
  else
    pf = pow(nDotH, material.shininess);

  ambient += lights.source[i].ambient;
  diffuse += lights.source[i].diffuse * nDotLi * attenuation;
  specular += lights.source[i].specular * pf * attenuation;
}
```