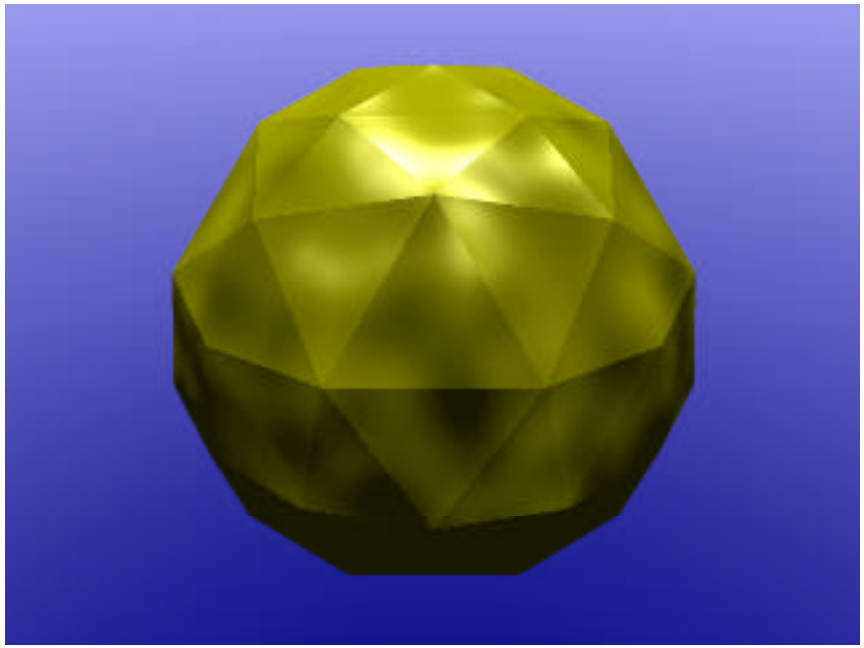
Synthèse d'images
(8) Coloriage réaliste



Plan de l'exposé

0. Un peu de physique

1. Coloriage à plat

2. Interpolation de Gouraud

3. Modèle de Phong

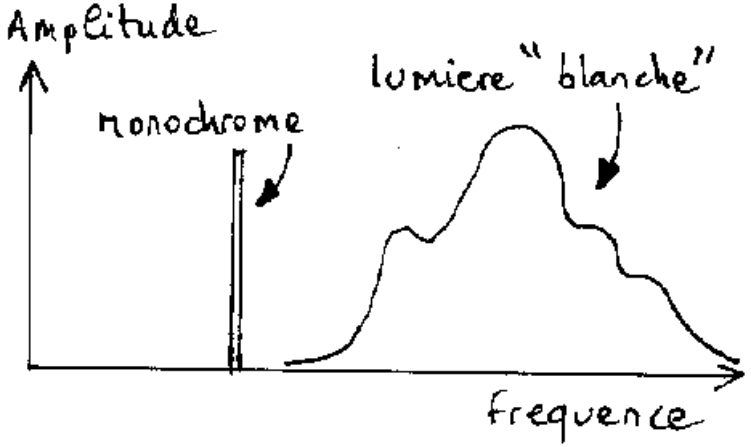
4. Microfacettes

5. Textures

v5 P. CUBAUD <cubaud@cnam.fr> Dept. informatique CNAM, Paris

Un peu de physique…

- Couleur des corps diffusifs : 3 paramètres
  - Le spectre de la lumière qu'il reçoit
  - Les propriétés d'absorption du corps
  - Les propriété de perception de l'oeil
- Source lumineuse
  - Spectre d'émission :



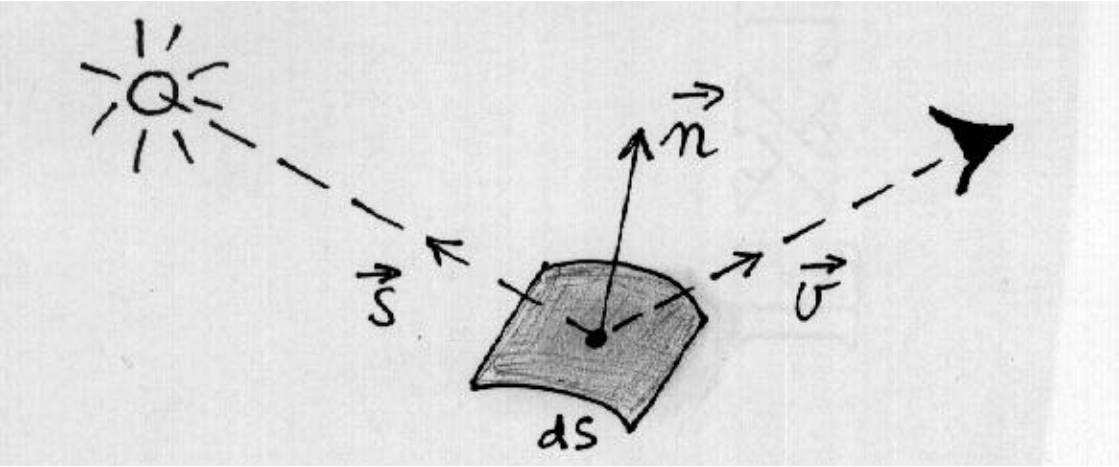
  - Source ponctuelle ou non => cohérence spatiale
- Interaction lumière / objet
  - une partie de la puissance lumineuse reçue est absorbée par l'objet et convertie en chaleur
  - une partie est réfléchie par la surface
  - le reste est transmis à l'intérieur de l'objet (réfraction)
- Lumière ambiante : produit des multiples réflections dans la scène

- Deux types de réflection :

  Diffuse : La surface de l'objet "réagit". La lumière est ré-émise dans toutes les directions. Sa couleur est affectée.

  Spéculaire : La surface de l'objet ne "réagit" pas. La lumière est ré-émise selon l'angle d'incidence. Sa couleur n'est pas affectée.

  Les 2 composantes sont toujours plus ou moins présentes (exemple : papier ≠ aluminium)



- Loi de Lambert pour la réflexion diffuse

  Idiff = Intensité réfléchie

  Ip = Intensité de la source ponctuelle

  Kd = coefficient de réflexion diffuse du matériau

$$I_{diff} = I_P K_d (\vec{n}.\vec{s}) = I_P K_d \cos\theta$$

  Si la source est infiniment loin, θ est constant pour toute la surface éclairée.

  Sinon : facteur (empirique) d'atténuation de Ip en fonction de la distance

D : $I_{p'} = \min(1, \frac{1}{a + bD + cD^2})$

Bibliographie

- P.Léna, A. Blanchard "Lumières, une introduction aux phénomènes optiques" InterEditions, 1990.

  Excellente introduction à la physique de la lumière. Niveau 1ère année DEUG ?

- M. Minnaert "The nature of light and color in the open air", Dover, 19??

  Moins mathématique que le précédent. Arcs en ciel, mirages, halos…

- B. Maitte "La lumière" collection "Points science", Seuil, 1981

  Introduction à l'histoire des théories concernant la lumière et plus largement au monde physique. Très clair.

- V. Ronchi "L'optique, science de la vision", Masson, 1966

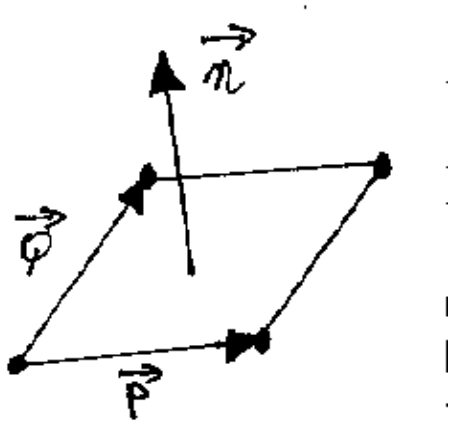
  Un classique de l'histoire de l'optique, mais un peu vieilli (par rapport aux travaux de G. Simon par ex.)

SURTOUT :

- P. Callet "Couleur-lumière, couleur-matière" Diderot 1998

  Interaction lumière-matière et synthèse d'images

1. Le coloriage à plat ("flat shading") Bouknight 1970



$$\vec{N} = \vec{P} \circ \vec{Q} \qquad N = P.Q.\sin\theta$$

$$\begin{cases} x_N = y_P.z_Q - z_P.y_Q \\ y_N = z_P.x_Q - x_P.z_Q \\ z_N = x_P.y_Q - y_P.x_Q \end{cases}$$

• On utilise la loi de lambert. L'intensité réfléchie est calculée pour chaque face et supposée constante pour toute la surface du polygone.

• L'approximation est valide si

- la source est infiniment loin

- l'observateur aussi, ou bien si la surface n'est pas réfléchissante

• Calcul de la normale à la face : On utilise le produit vectoriel (cf. chapitre 6)

• Les vecteurs N et S sont ensuite normalisés

Reprise de l'algorithme

- On ajoute des structures de données

```
double IN[MAXFACES];       /* intensite des faces */
double LUM[3] = {0.0,10.0,0.0};  /* position source
de lumiere */
```

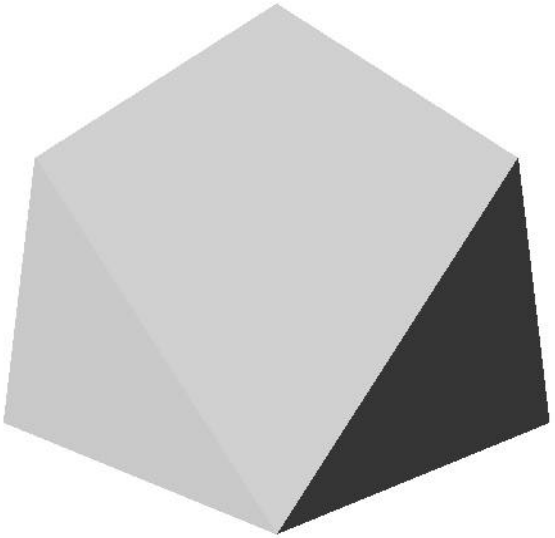
- Calcul de l'intensité recue par les faces (après le test de visibilité)

```
 for (i=0;i<NB;i++)
 {
  px=LUM[0]-OB[i][0][0];
  py=LUM[1]-OB[i][0][1];
  pz=LUM[2]-OB[i][0][2];
  nn=sqrt(px*px+py*py+pz*pz);
  px/=nn;py/=nn;pz/=nn;
  IN[i]=px*NO[i][0]+py*NO[i][1]+pz*NO[i][2];
  /* résultat entre -1 et +1 */
  IN[i]=(IN[i]+1)/2; /* valeur finale entre 0 et 1 */
 }
```

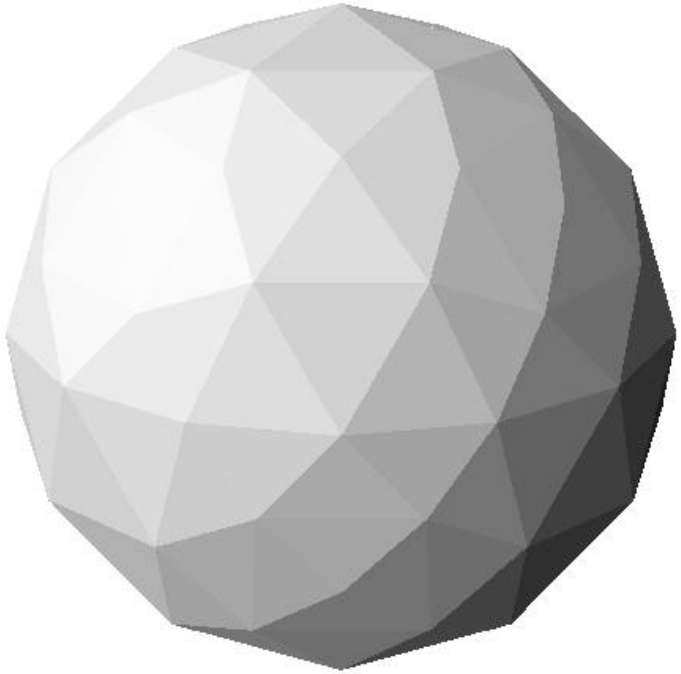
- On ne dessine que les faces visibles

```
 for (i=0;i<NB;i++)
 {
  if  (VI[i]>0) {
  printf("newpath\n");
  printf("%.5f setgray\n",IN[i]);
  printf("%.5f %.5f
moveto\n",O2[i][0][0],O2[i][0][1]);
  printf("%.5f %.5f
lineto\n",O2[i][1][0],O2[i][1][1]);
  printf("%.5f %.5f
lineto\n",O2[i][2][0],O2[i][2][1]);
  printf("%.5f %.5f
lineto\n",O2[i][0][0],O2[i][0][1]);
  printf("fill\n");
  printf("newpath\n");
  printf("0 setgray\n",IN[i]);
  printf("%.5f %.5f
moveto\n",O2[i][0][0],O2[i][0][1]);
  printf("%.5f %.5f
lineto\n",O2[i][1][0],O2[i][1][1]);
  printf("%.5f %.5f
lineto\n",O2[i][2][0],O2[i][2][1]);
  printf("%.5f %.5f
lineto\n",O2[i][0][0],O2[i][0][1]);
  printf("stroke\n");
  }
 }
```

Résultats



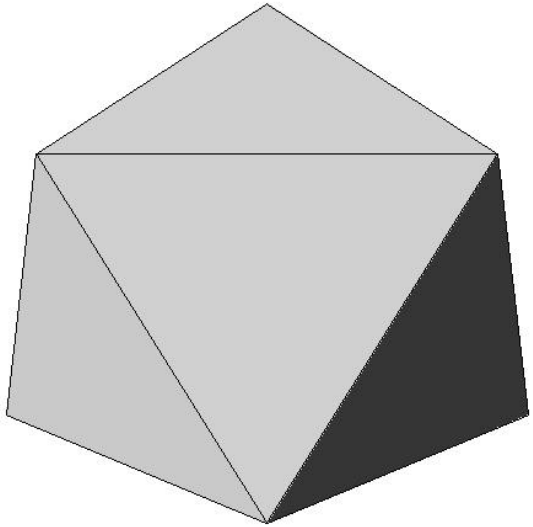
Niveau 1



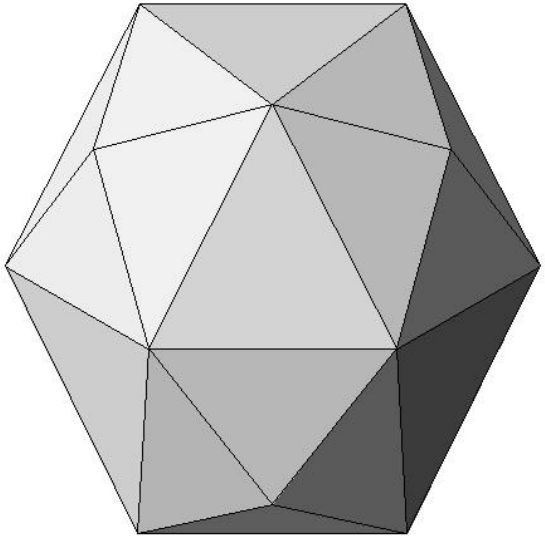
Niveau 3

niveau 2

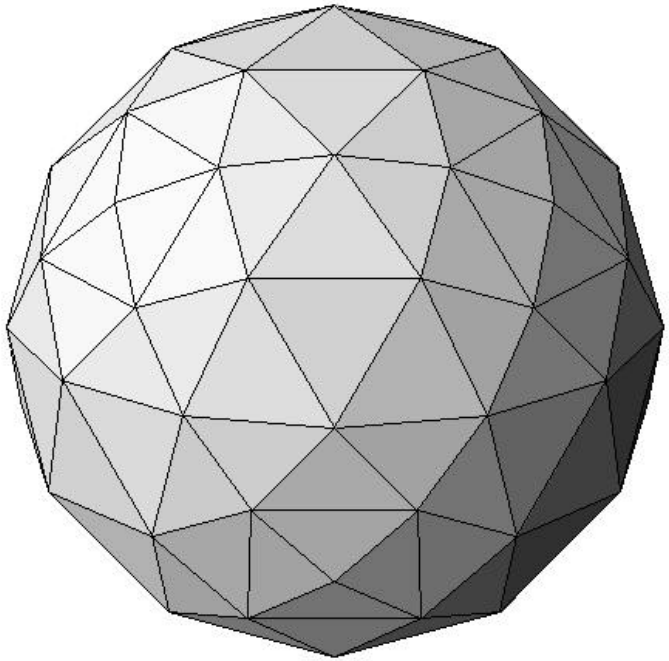
Renforcement des contours



Niveau 1



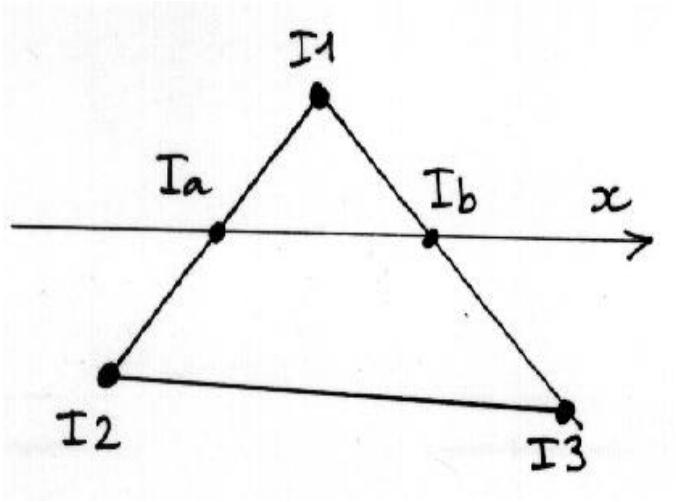
Niveau 2 et 3



2. Interpolation de Gouraud "Gouraud shading" (1971)

Principe : interpolation linéaire de l'intensité pour éliminer les discontinuités

Une technique similaire à l'interpolation de la profondeur pour le Z-Buffer :

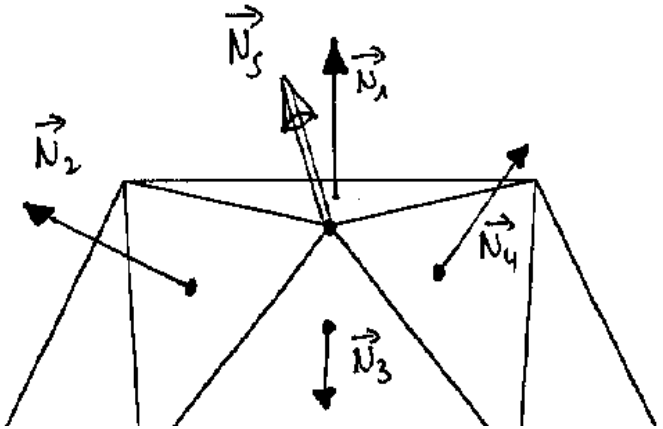


$$I_a = I_1 - (I_1 - I_2)\frac{y_1 - y_S}{y_1 - y_2}$$

$$I_b = I_1 - (I_1 - I_3)\frac{y_1 - y_S}{y_1 - y_3}$$

$$I_p = I_b - (I_b - I_a)\frac{x_b - x_p}{x_b - x_a}$$

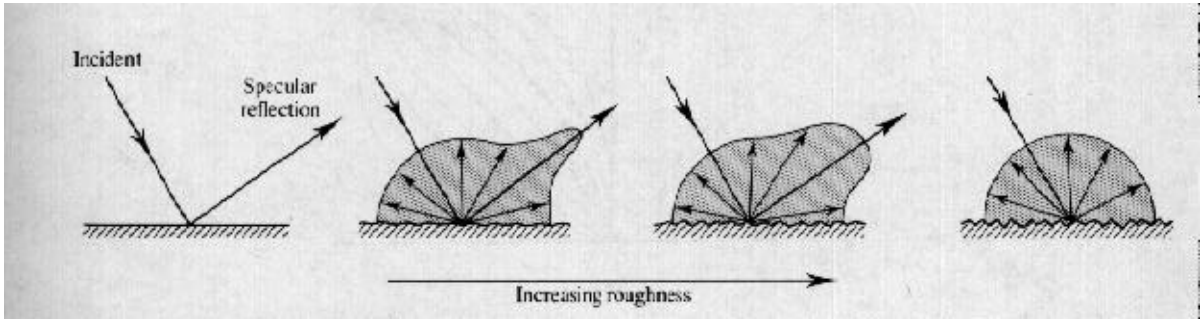
L'intensité aux sommets est obtenue par interpolation des normales des faces adjacentes.



$$\vec{N}_S = \frac{\vec{N}_1 + \vec{N}_2 + \vec{N}_3 + \vec{N}_4}{\left\|\vec{N}_1 + \vec{N}_2 + \vec{N}_3 + \vec{N}_4\right\|}$$

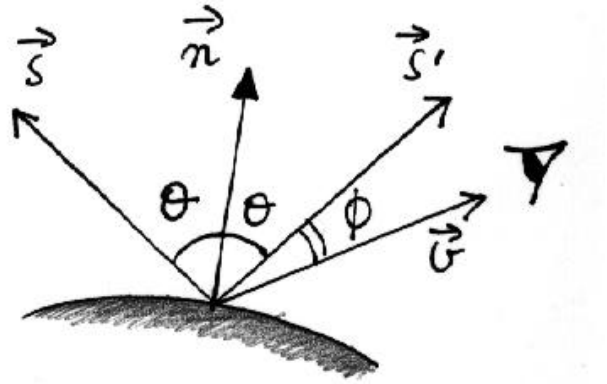
3. Modèle de Phong Bui-Tuong (1975)

- Prise en compte de la réflexion spéculaire, variable selon le matériau



[WATT] p. 53

- On doit ici tenir compte de la position de l'observateur :



- Modèle (empirique) retenu :

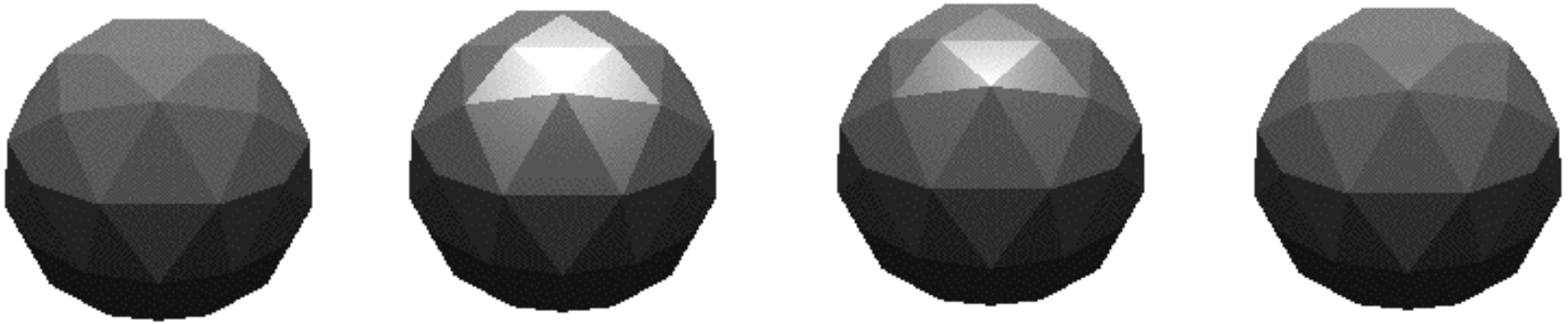
$$I_{spec} = I_P.K_S.\cos^n\phi \text{ donc } I_{total} = I_{diff} + I_{spec} = I_P(K_d\cos\theta + K_s\cos^n\phi)$$

Exemples avec POV

Option de rendu :

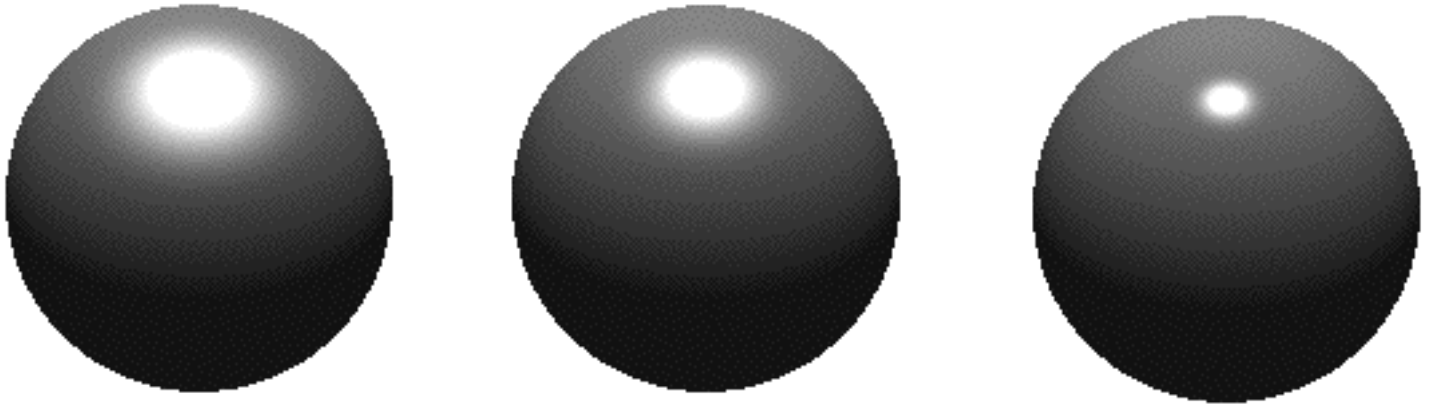
`finish { phong Ks phong_size n }` avec $0 \leq Ks \leq 1$ et $1 \leq n \leq 256$

Résultat avec la pseudo-sphère (niveau 3) :



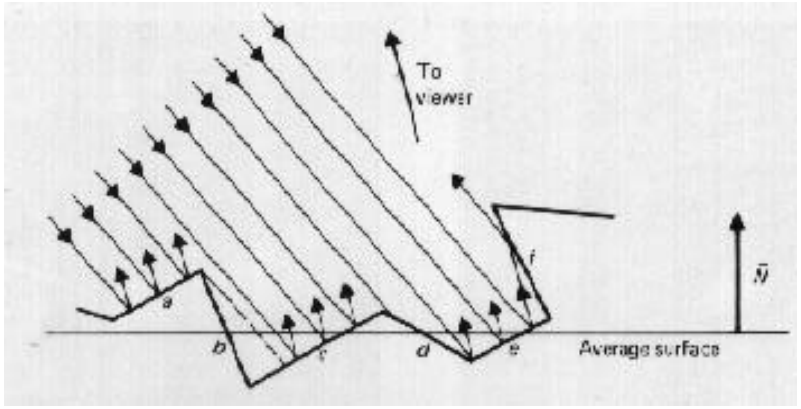
(sans) | Ks = 1 et N = 5 | Ks = 1 et N = 10 | Ks = 1 et N = 50

- Résultat avec une sphère



Ks = 1 et N = 5 N = 10 N = 50

4. Modèles de micro-facettes



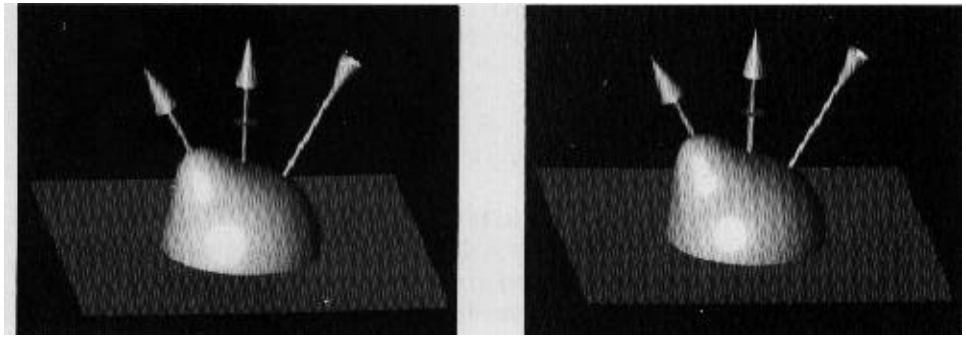
- Modèle de Davies (1954) :

  Pas d'interaction entre les micro-facettes + distribution des hauteurs selon une loi normale

- Modèle de Torrance & Sparrow (1966) :

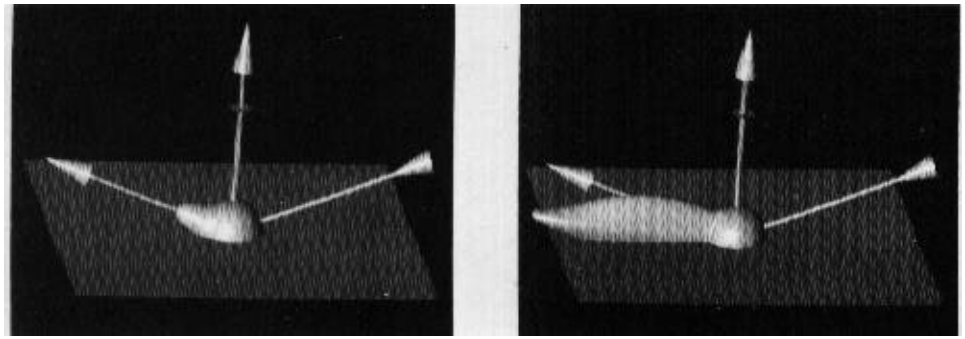
  Facettes "en V", réfléchissantes, orientées de manière aléatoire selon une loi de Beckmann

θ = 30°



Phong Torrance-Sparrow

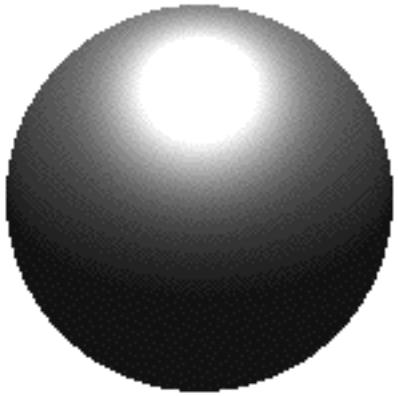
θ = 70°



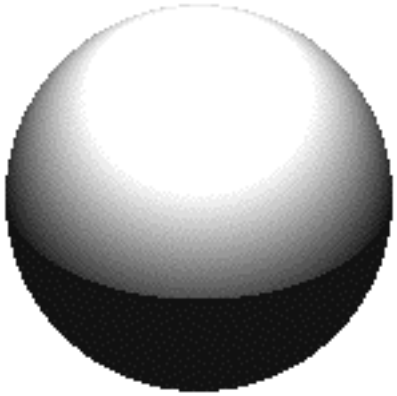
Phong Torrance-Sparrow

- Le modèle de POV

```
finish
{specular 1  roughness 0.1}
```



Ks = 1 et Rough = 0.1



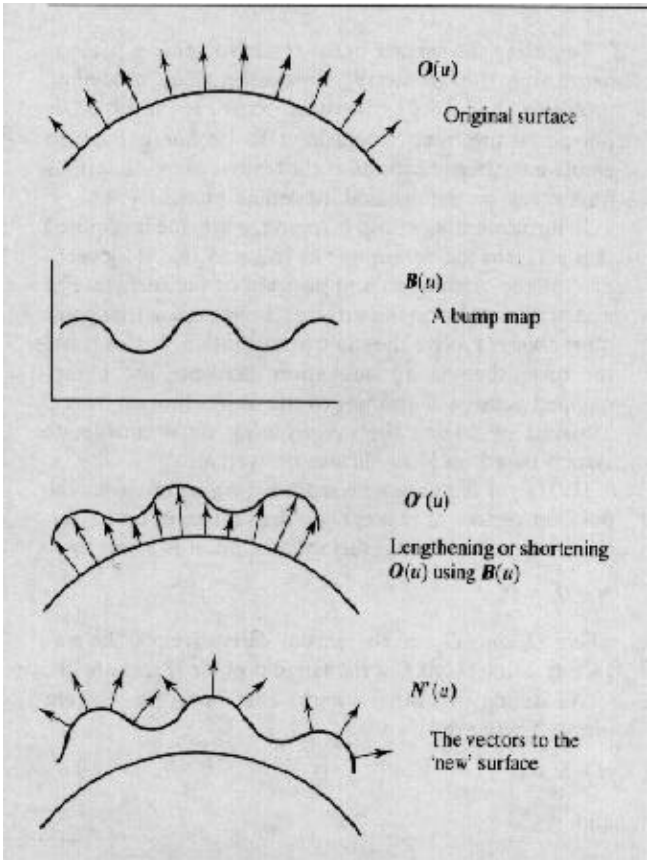
Ks = 1 et Rough = 0.5

Ks = 1 et Rough = 1.0

Le "bump mapping"

- Perturber la normale à la surface

=> Astuce pour donner un relief aux objets.



[WATT] p. 200

- Plusieurs techniques :

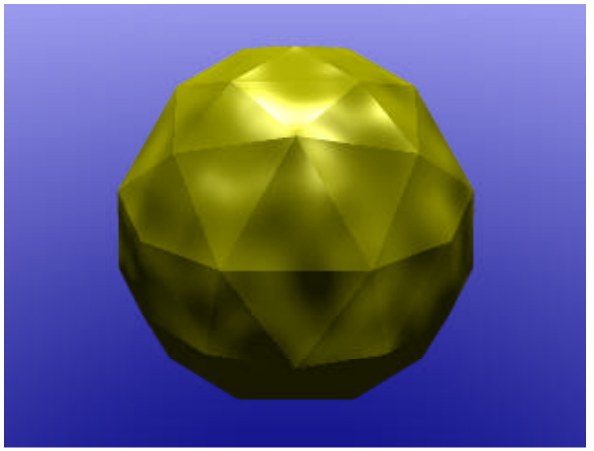
  - Le plus simple : perturbation aléatoire

  - Le plus général : un tableau indiquant pour chaque point de la surface comment se fait la perturbation

- Exemple en POV

  `texture { normal { bumps` *`profondeur`* `scale` *`largeur`* `}}`

  => nombreuses options très spectaculaires



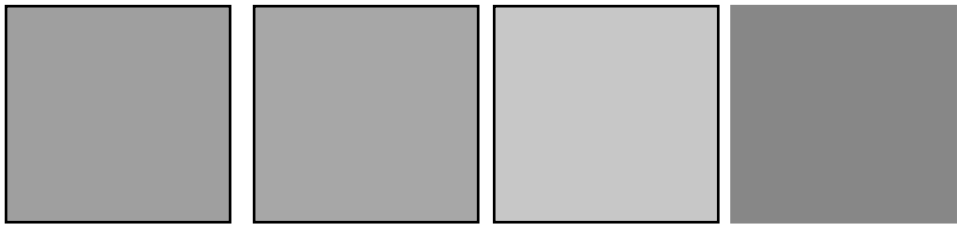
5. Textures

- Les algorithmes précédents ne retirent pas complètement le caractère "artificiel" du coloriage.

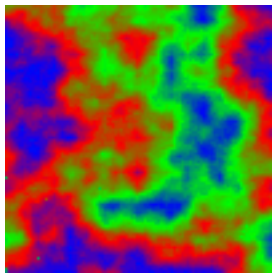
  Une solution souvent employée : le plaqué de textures sur les faces

- Divers types de textures :

  - "géométriques" => forme répétitive



  - "synthétiques" => génération aléatoire



  - "réalistes" => photographie numérisée

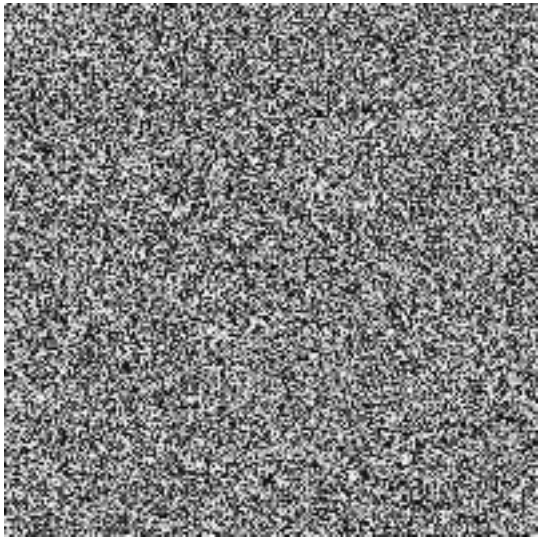
- On se ramène toujours à un tableau de points (de petite taille)

Un premier essai de synthèse

```
int Hasard()
{
 return (int)(255*Uniform());
}

main()
{
int i,j;

 s1=11;s2=13;
 /*--- remplissage du pixmap ---*/
 for (i=0;i<SIZE;i++)
  for (j=0;j<SIZE;j++)
   Pix[i][j]=Hasard();

 /*--- affichage ---*/
 printf("P2\n%d %d\n255\n",SIZE,SIZE);
 for (i=0;i<SIZE;i++)
  for (j=0;j<SIZE;j++)
  {
   printf("%d ",Pix[i][j]);
   if ((j%30)==0) printf("\n");
  }
}
```

Ce programme fabrique un fichier image à la norme PGM (fichier textuel, à visualiser avec xv par exemple - ou converti avec pnmtogif)

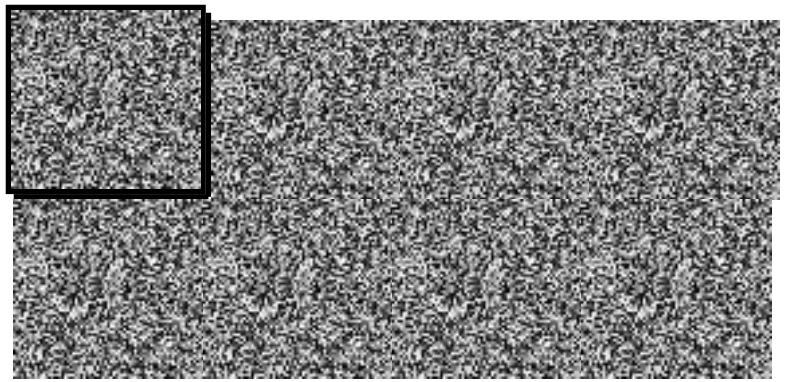
Résultat :

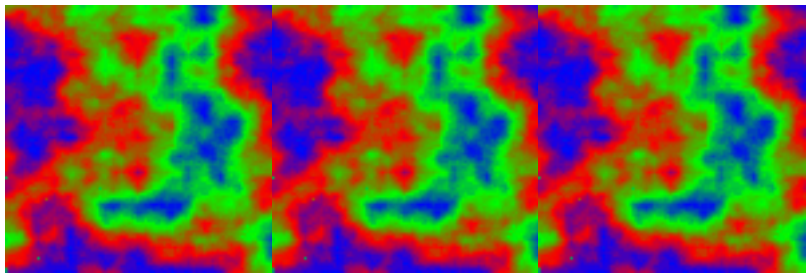


200x200 pixels

256 niveaux de gris

Problème : s'en servir ! La texture sera répétée sur toute la surface de l'objet 3D :





Bonnes propriétés du bruit à synthétiser:

- invariance statistique en cas de rotation et de translation

- peut être échantilloné à une fréquence basse (pour éviter l'aliasage)

Comment générer une texture "réaliste" ?

- texture minérale : marbre, granite - métallique

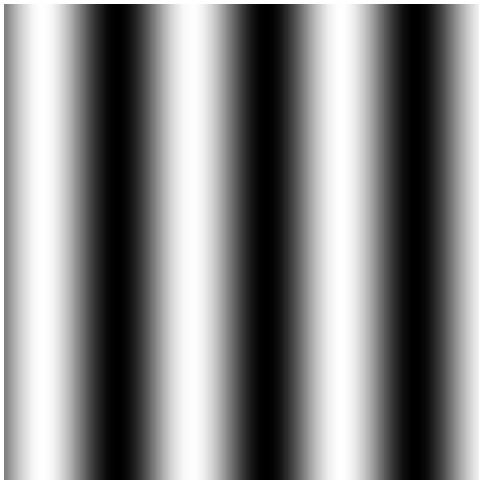
- textures pour les phénomènes turbulents (flammes)

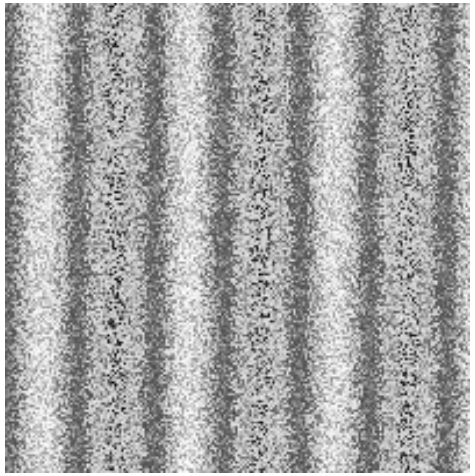
- texture pour la granulosité : peau d'une orange

- texture pour les couleurs animales : léopard, coquillage

Exemple du marbre

Idée : utiliser le bruit pour perturber une structure colorée simple :



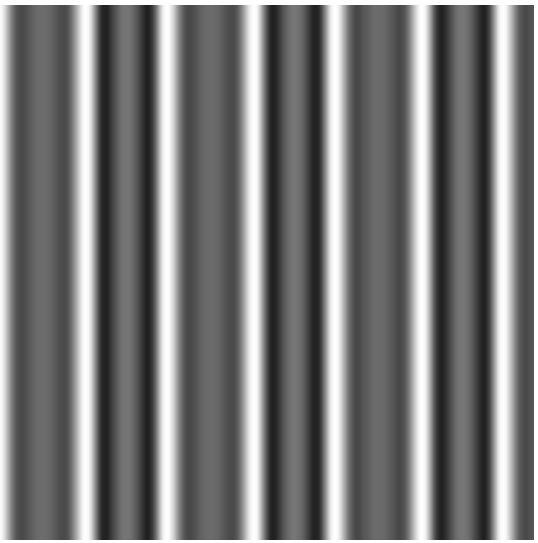


```
int marbre(x)
float x;
{
 x=2.0*(x+1)/3.0 -1.0;
 return (int)(127.5*(x+1.0));
}


main()
{
int i,j;

 s1=11;s2=13;
 /*--- remplissage du pixmap ---*/
 for (i=0;i<SIZE;i++)
  for (j=0;j<SIZE;j++)
   Pix[i][j]=marbre(sin(0.1*j)+Uniform());

 /*--- affichage ---*/
 printf("P2\n%d %d\n255\n",SIZE,SIZE);
 for (i=0;i<SIZE;i++)
  for (j=0;j<SIZE;j++)
  {
   printf("%d ",Pix[i][j]);
   if (j%80) printf("\n");
  }
}
```

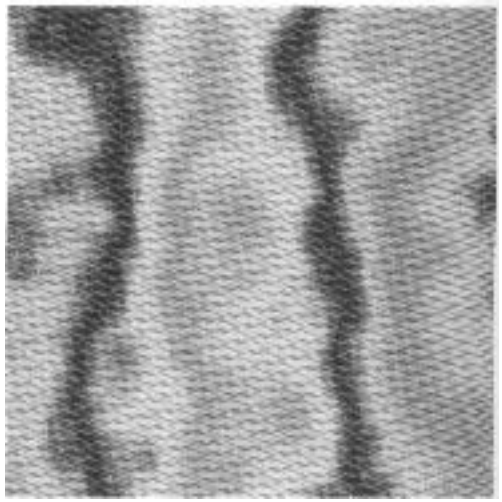
- Algorithme de Perlin :

  Voir Comp. Graph. 23(3) 253-62, 1989

  Utilise une fonction spline pour créer l'image initiale (ici, avec modif. sous xv du précédent) :



marbre(x) = couleur(sin(x+turbulence(x))



=> écrire l'algorithme complet d'après Watt & Watt.

Projection de la texture

- Textures "géométriques" :

  - la texture est décrite par une matrice P de taille 16x16

  - Pour chaque pixel (x,y) du polygone, la couleur de base est déterminée par P[x mod 16, y mod 16]

- les autres : problème de projection

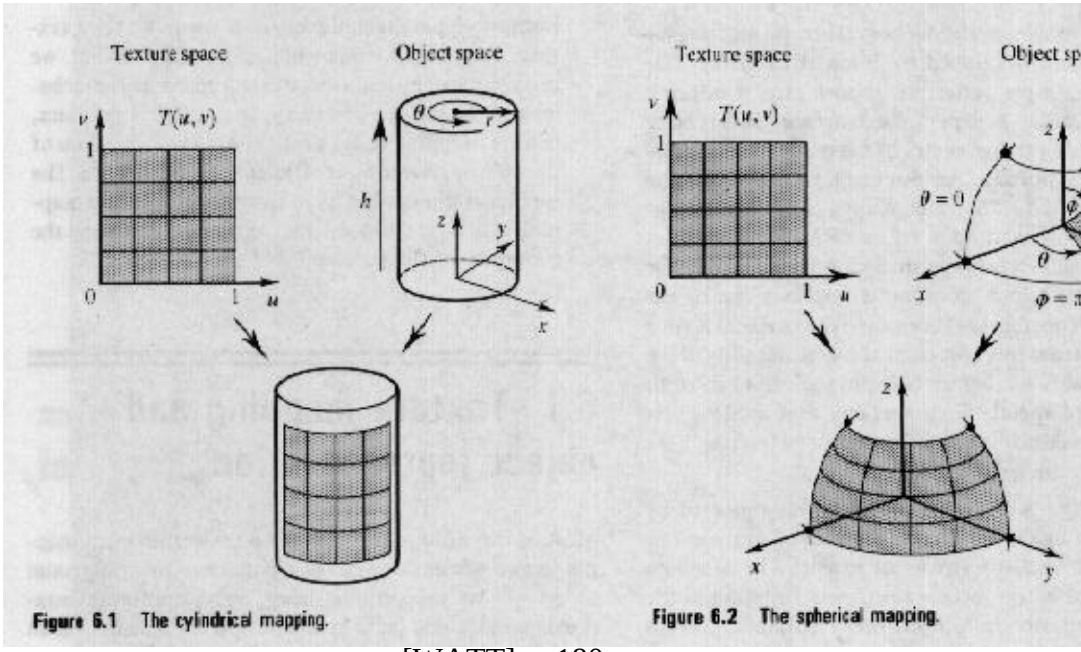
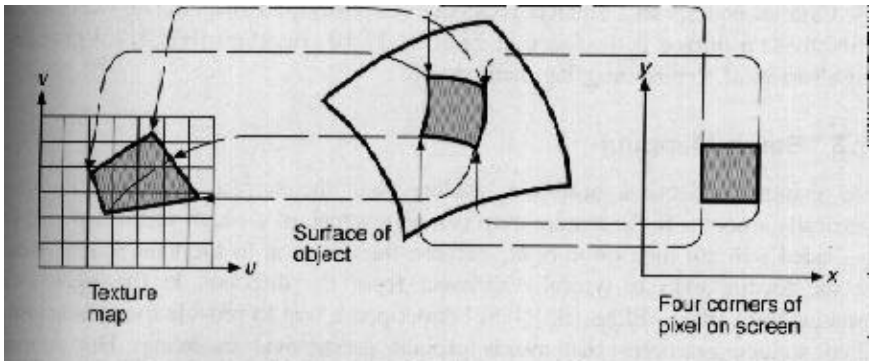


Figure 6.1 The cylindrical mapping.

Figure 6.2 The spherical mapping.

[WATT] p. 180



[FOLEY] p. 743

P.S. Heckbert "Survey of texture mapping" CG&A Novembre 1986, pp. 56-67.