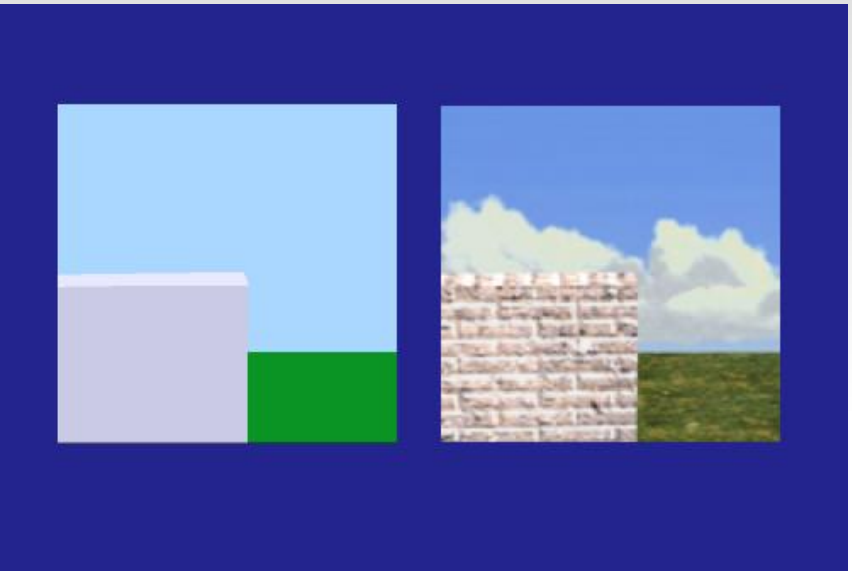
TEXTURES

M. NEVEU

à partir notamment du cours Siggraph 97 et d'autres ….

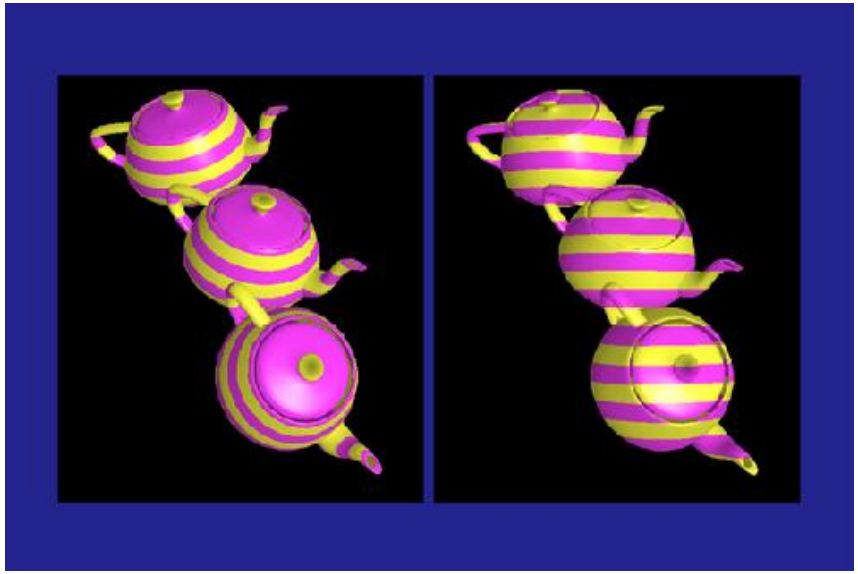
TEXTURES



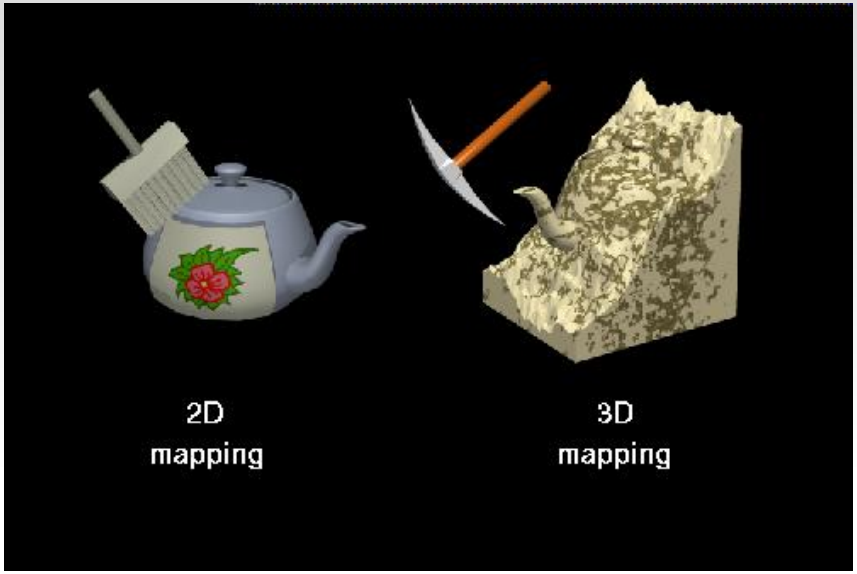
Gain de réalisme

Pas « cher »

Attacher la texture à l'objet ⇔ travailler dans l'espace de l'objet



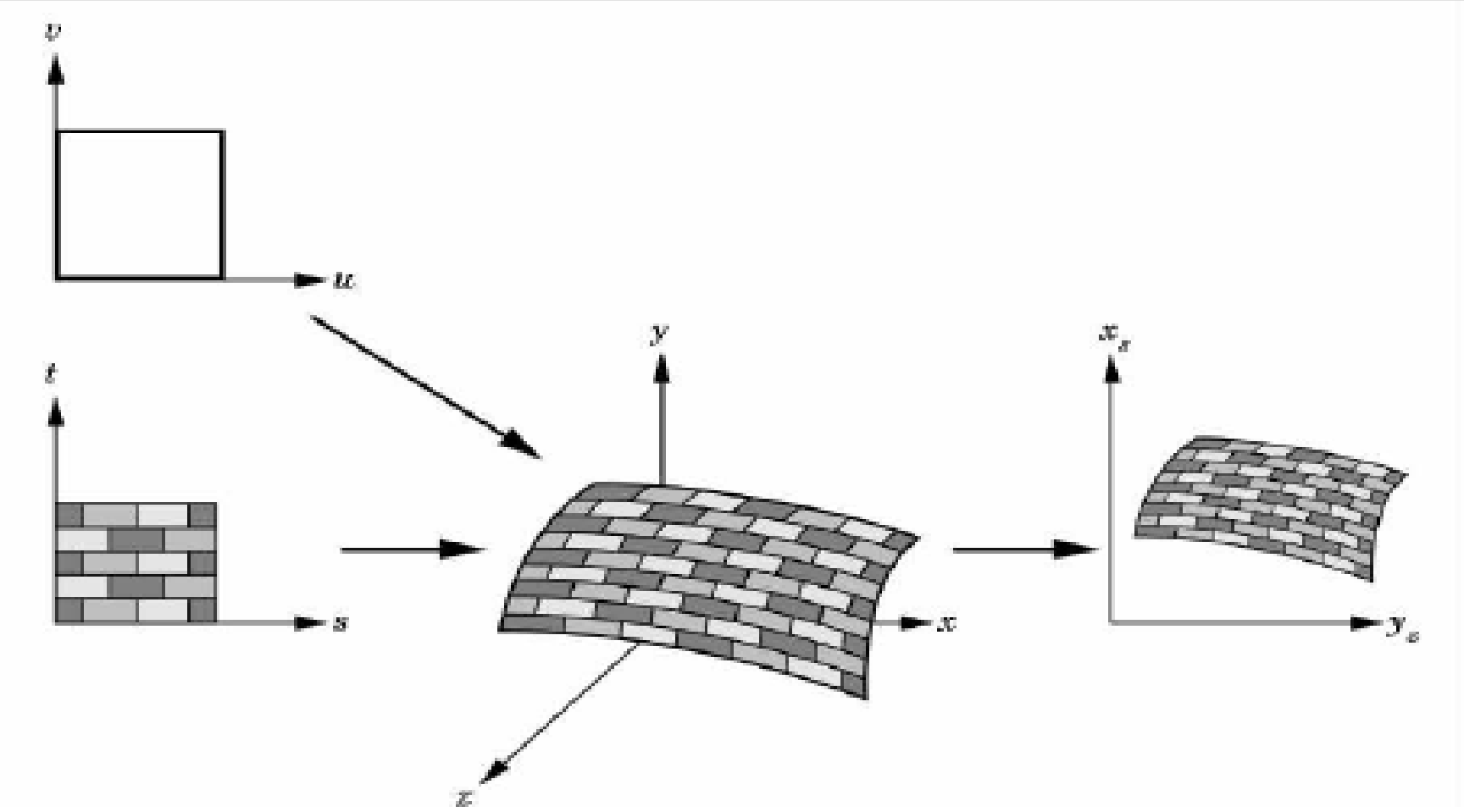
Textures 2D et 3D



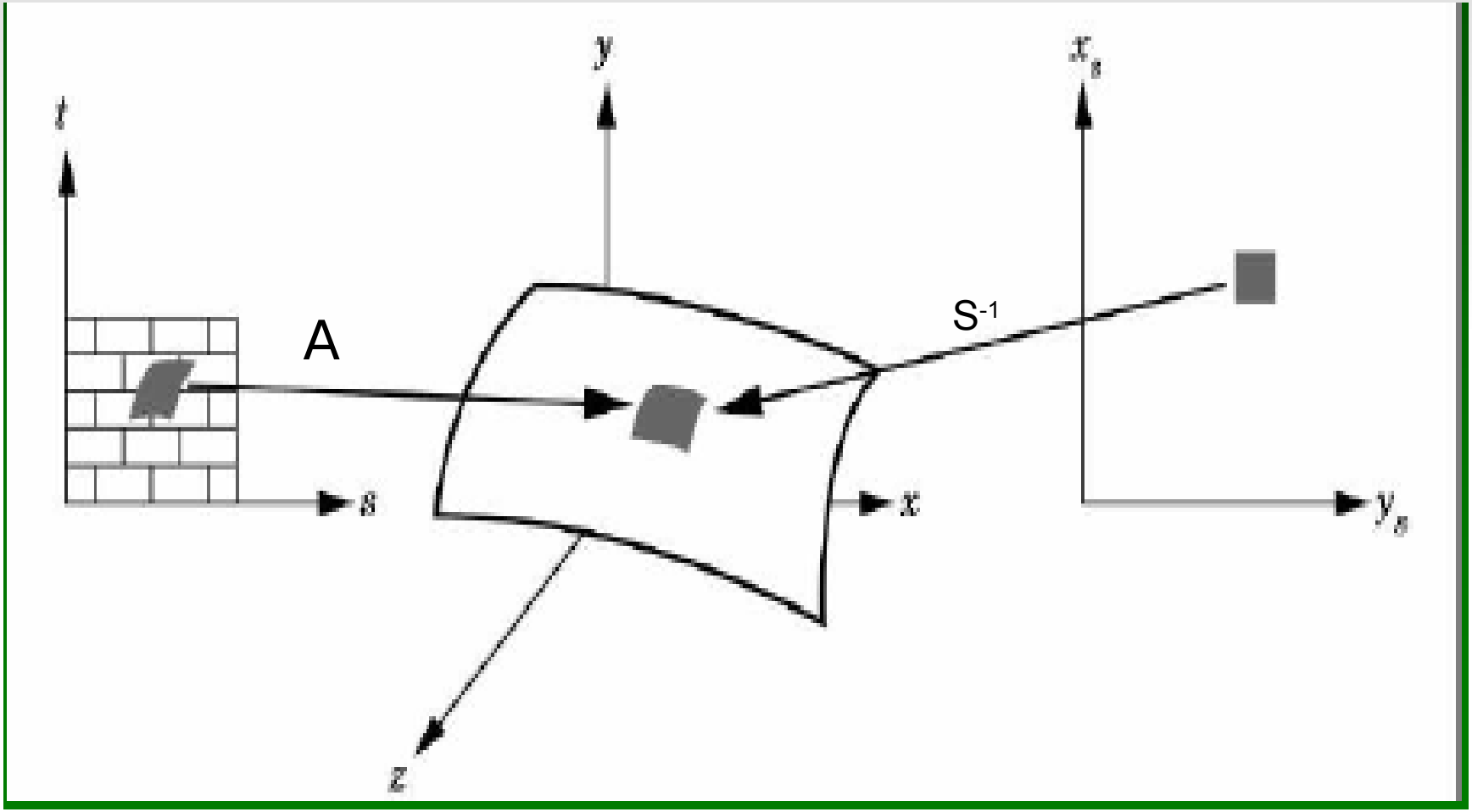
Analogies : papier peint (2D) , sculpture (3D)

Images réelles ou virtuelles

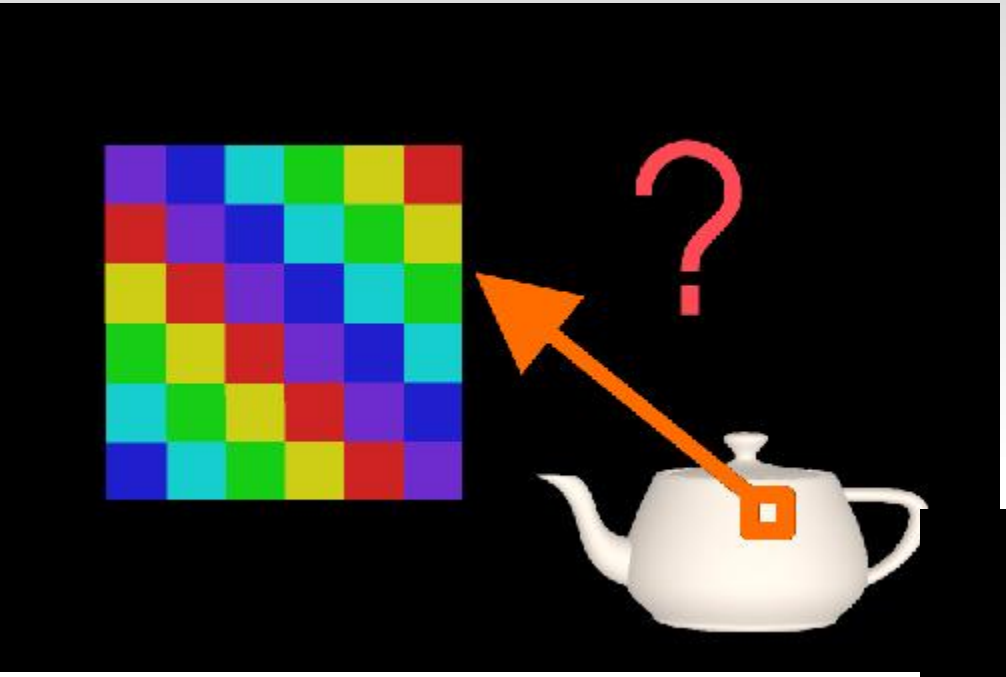
PRINCIPE



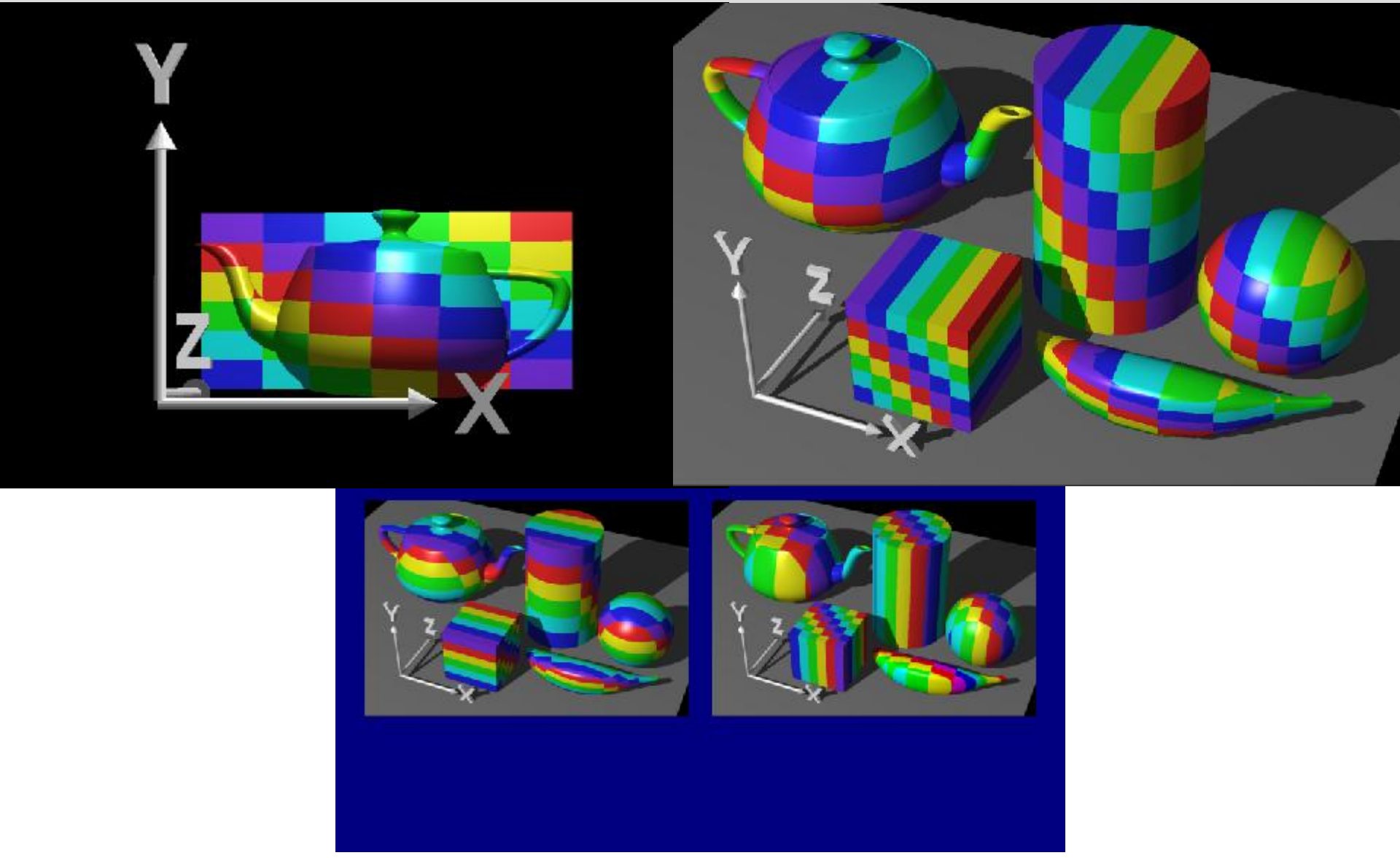
PRINCIPE



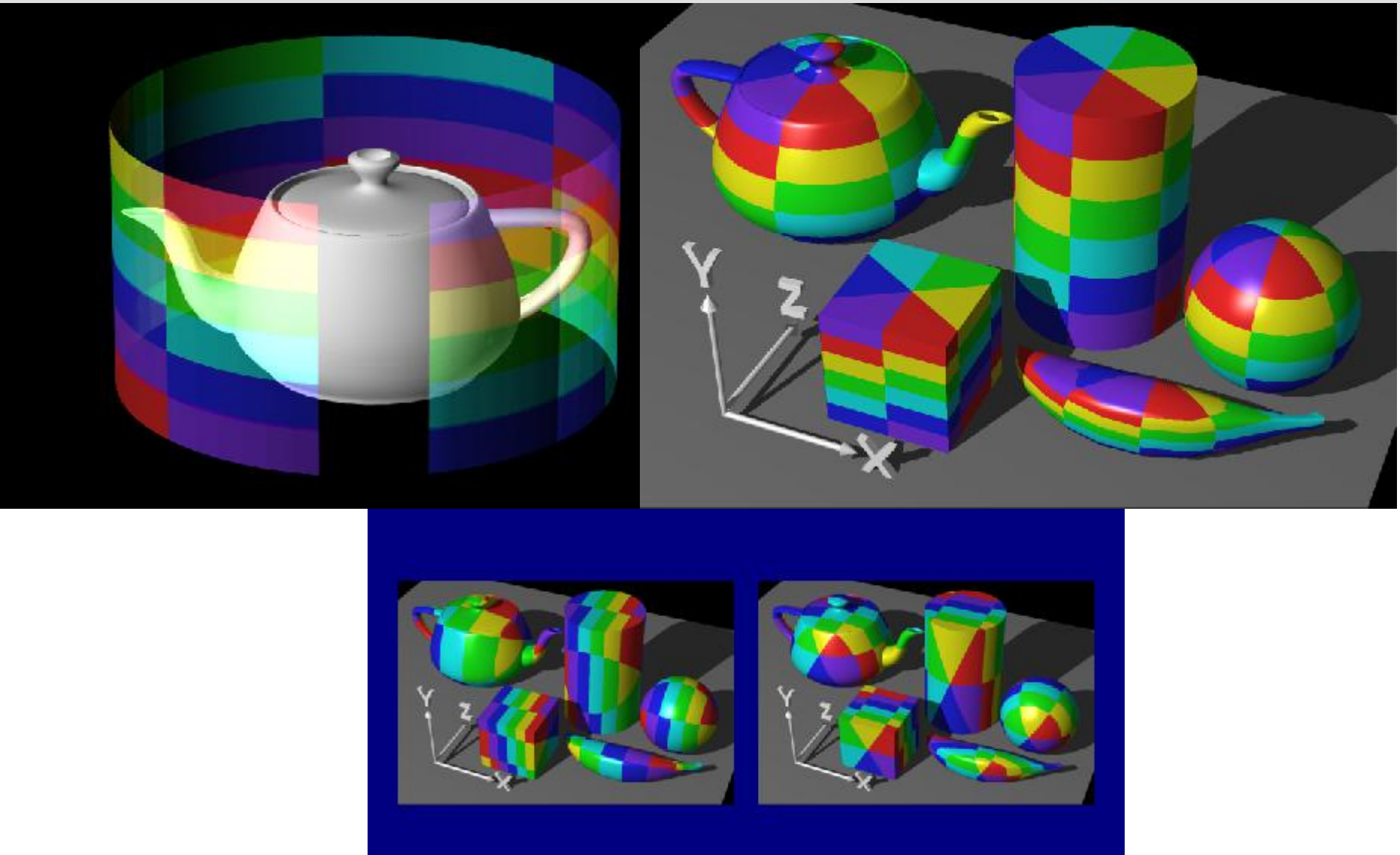
Application A



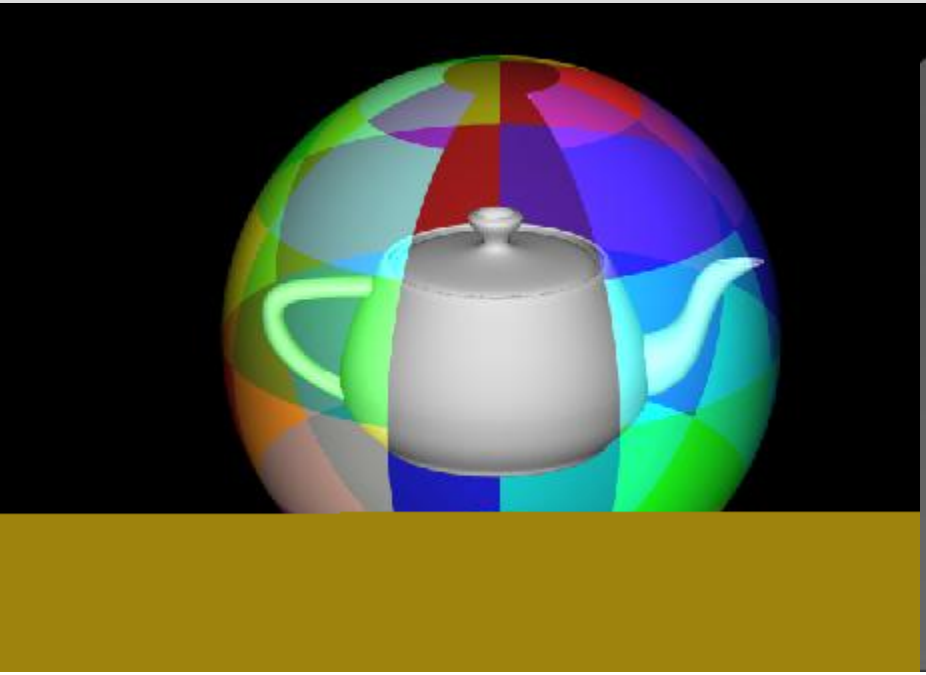
Projection selon un axe

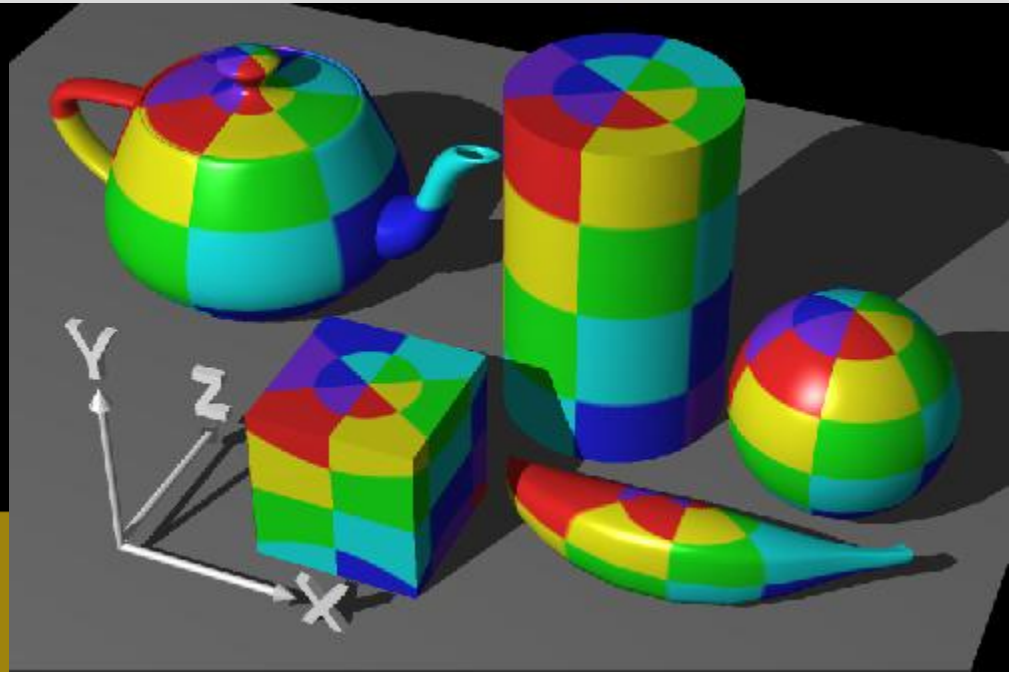


Projection cylindrique



Projection sphérique





Sous échantillonnage (aliasing)

Mip-Mapping : texture à plusieurs échelles

On suppose l'image de la texture carrée et de taille $2N \times 2N$.
Changement de résolution le plus simple : construire une image deux fois plus petite.
A chaque pixel de l'image deux fois plus petite, on associe un bloc de 2 ×2 pixels sur l'image originale de valeur = moyenne des 4 pixels du bloc.

$$T_{i,j}^{k-1} = \frac{1}{4}(T_{2i,2j}^{k} + T_{2i+1,2j}^{k} + T_{2i,2j+1}^{k} + T_{2i+1,2j+1}^{k})$$

La taille de l'image $T_{i,j}^{k}$ est $2k \times 2k$.

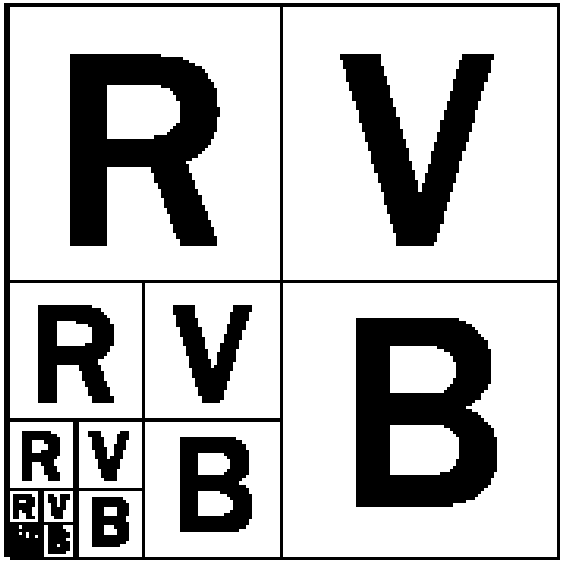
A partir de l'image T^N on construit un ensemble de $N + 1$ images T^k, k de 0 à N

Facteur de compression au niveau k : $d_k = 2^{-k}$

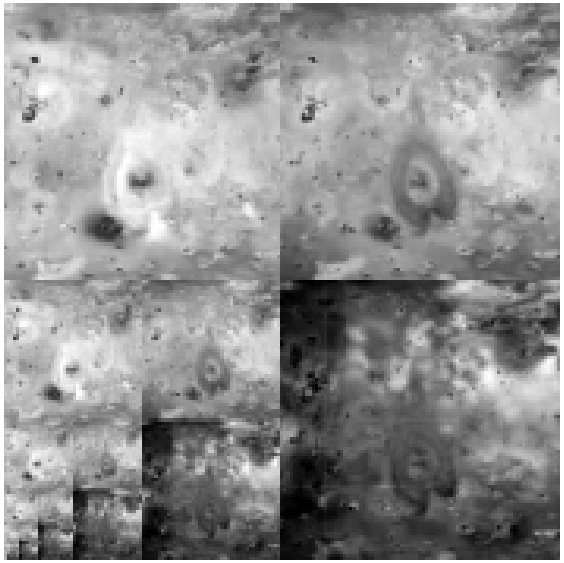
Mip-Mapping

Ensemble des images calculées pour le mip-mapping.

Stockage optimal du mipmap d'une texture couleur (de la figure précédente) en mémoire.

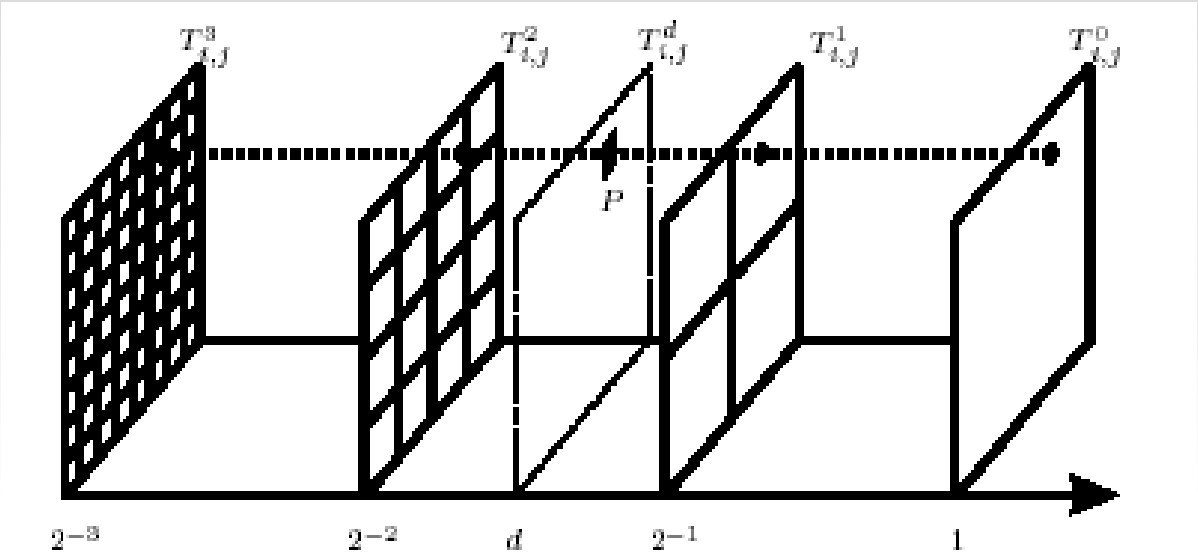


(a) Principe



(b) Exemple

Mip mapping : couleur d'un pixel



Pixel courant (u_c,v_c,d)

$k_c = -\log_2(d)$

Voisin le plus proche

$$C(u_c,v_c)=T^{k'}_{i',j'} \quad où \quad k' =[k_c] \quad et \quad (i',j')=\left([u_c.2^{k'}],[v_c.2^{k'}]\right).$$

Interpolation bilinéaire

$$C(u_c,v_c)=C_k(u',v') \quad avec \quad k'=[k_c], \quad (u',v')=(u_c-\lfloor u_c\rfloor,v_c-\lfloor v_c\rfloor)$$

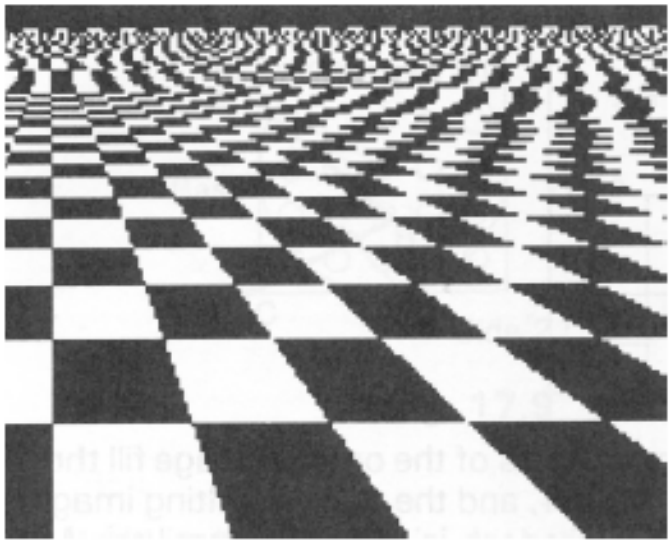
$$C_{k'}(u,v)=\begin{bmatrix}(1-u) & u\end{bmatrix}\begin{bmatrix}T_{i,j} & T_{i,j}\\ T_{i+1,j} & T_{i+1,j+1}\end{bmatrix}\begin{bmatrix}(1-v)\\ v\end{bmatrix} où \quad (i',j')=(u_c.2^{k'},v_c.2^{k'}).$$

Interpolation trilinéaire

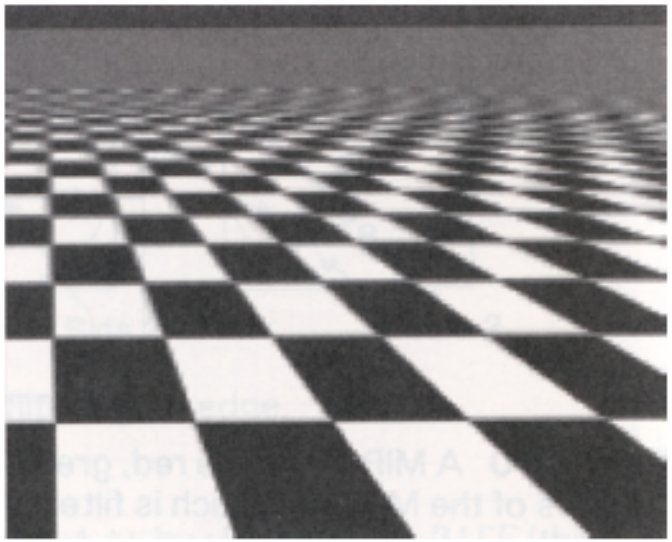
$$C(u_c,v_c)=(1-\lambda).C_{k'}(u',v')+\lambda.C_{k'+1}(u',v')$$

$$k_c=(1-\lambda).k'+\lambda.(k'+1)$$

$$avec \quad k'=\lfloor k_c\rfloor, \quad (u',v')=(u_c-\lfloor u_c\rfloor,v_c-\lfloor v_c\rfloor)$$

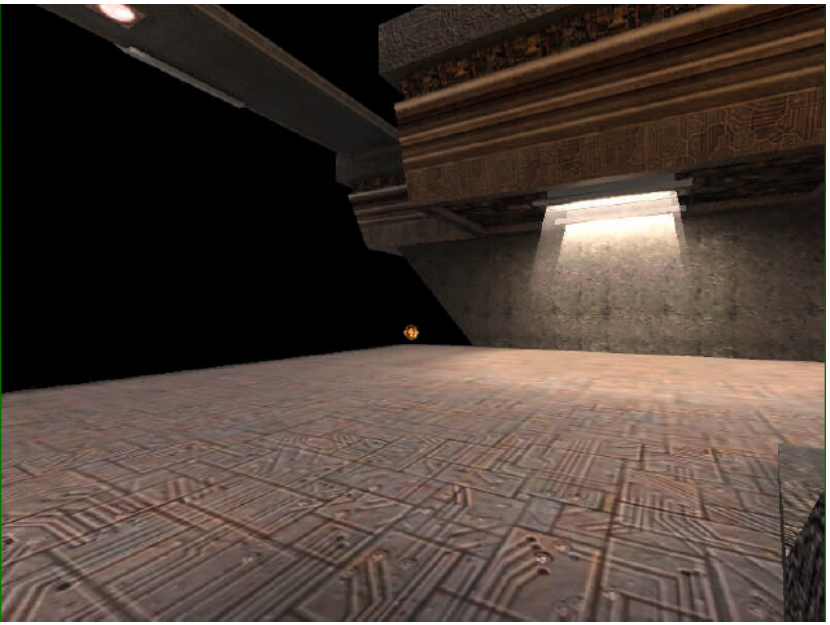


(a) Sans traitements

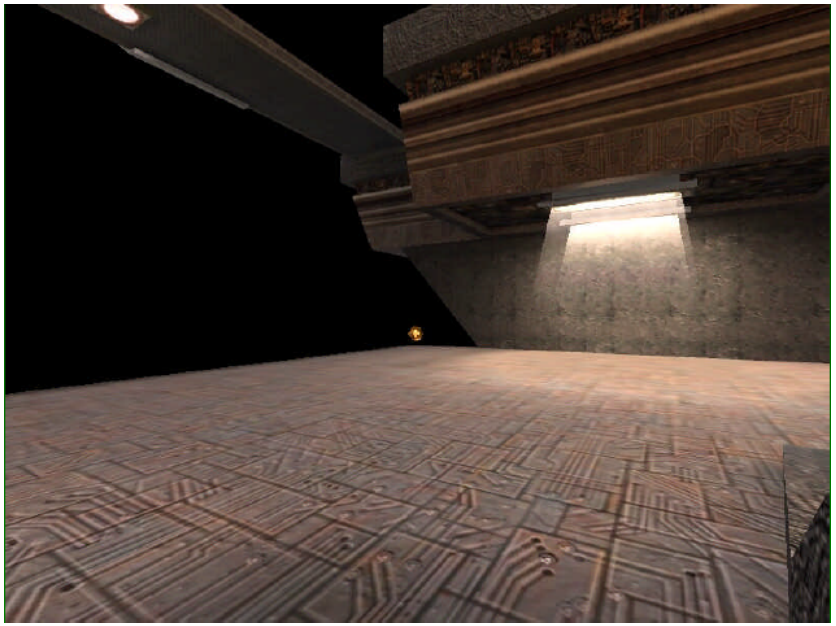


(b) Avec mip-mapping

Technique du mip-mapping appliqué à l'échiquier

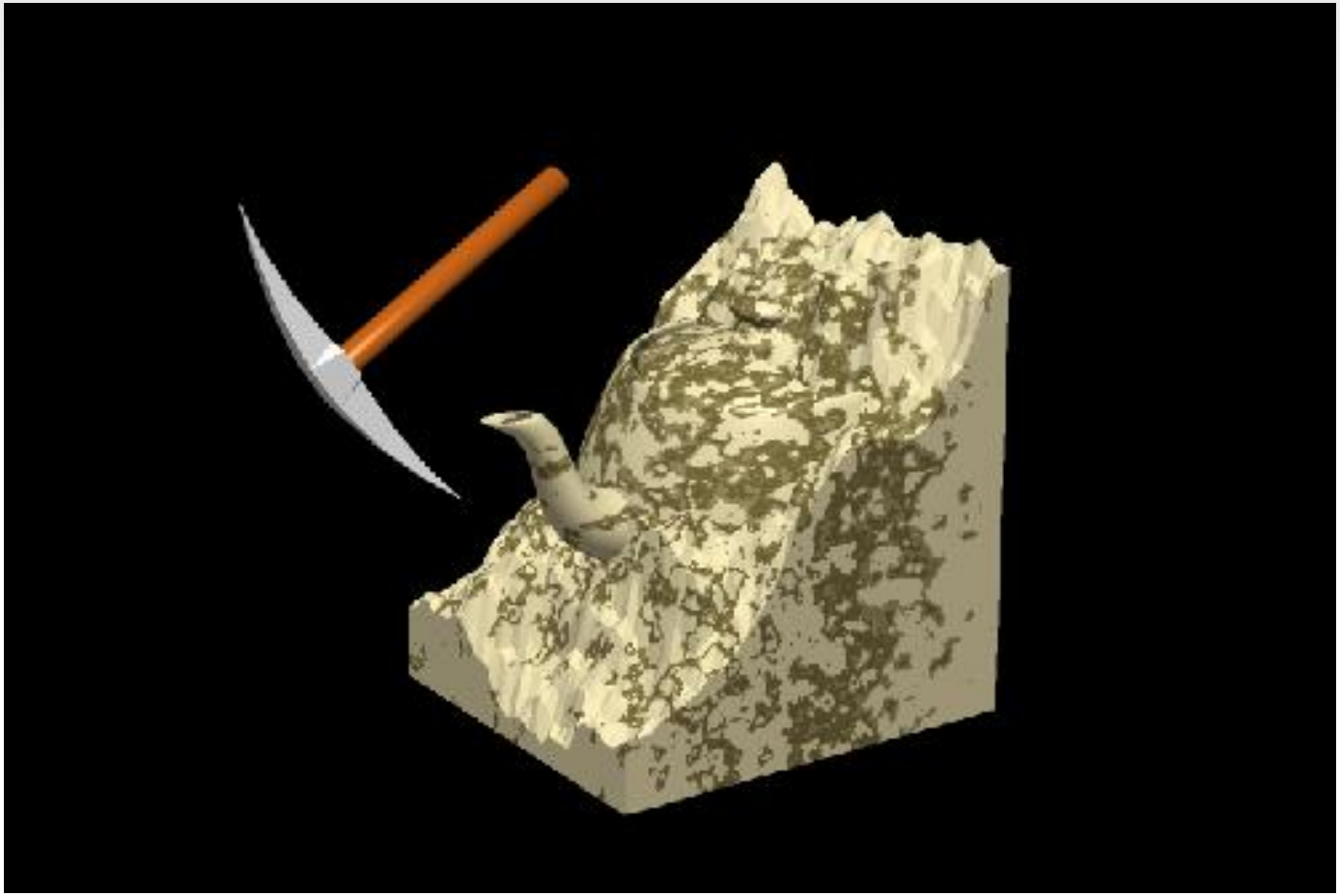


bilinéaire

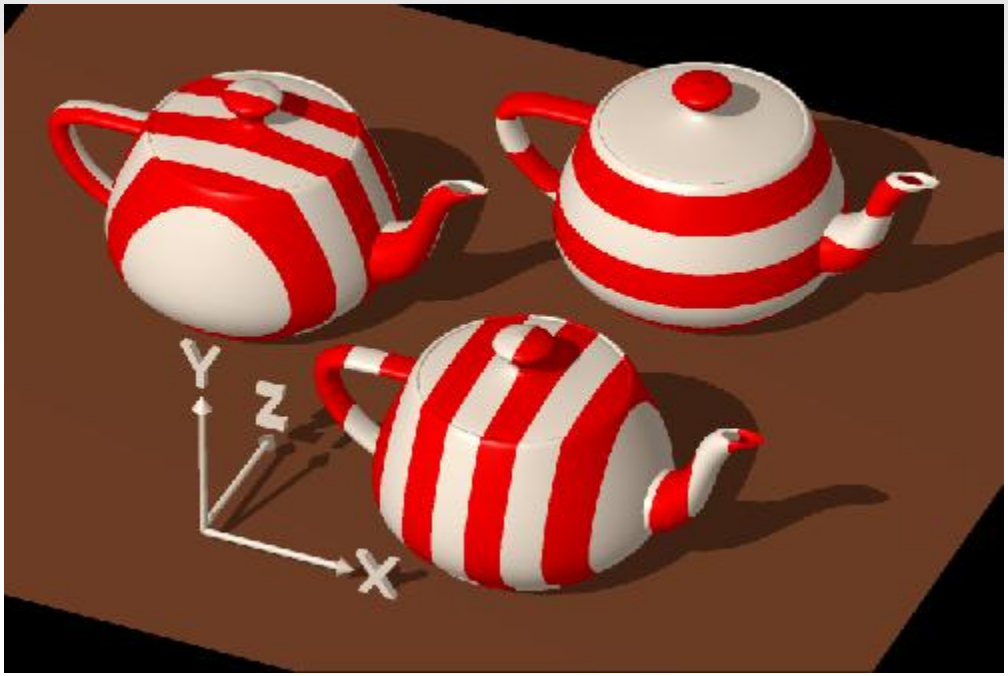


trilinéaire

En 3D

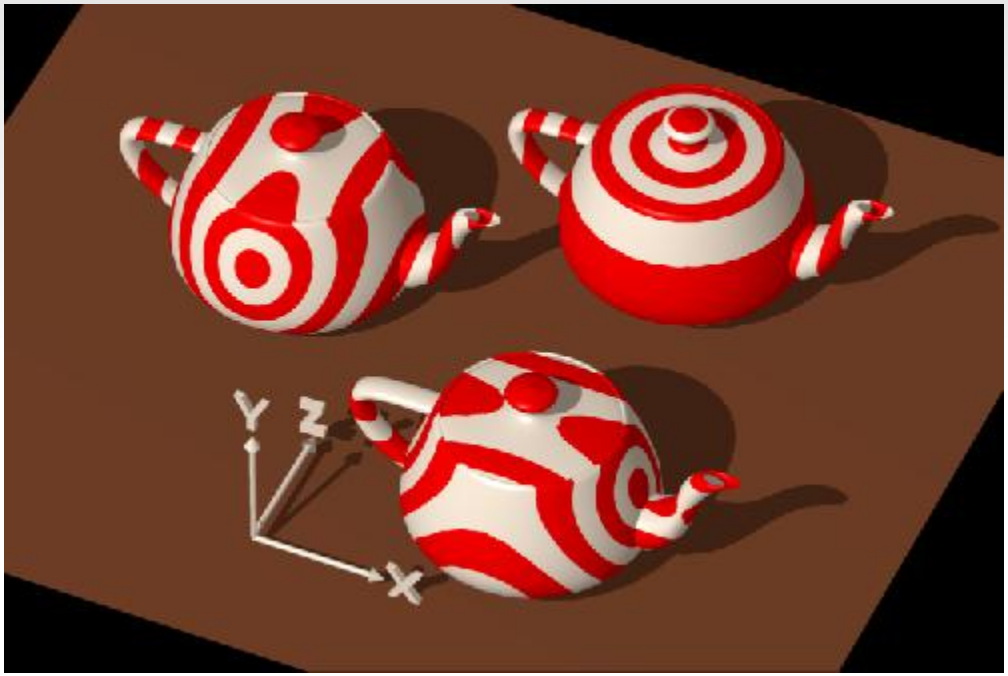


3D



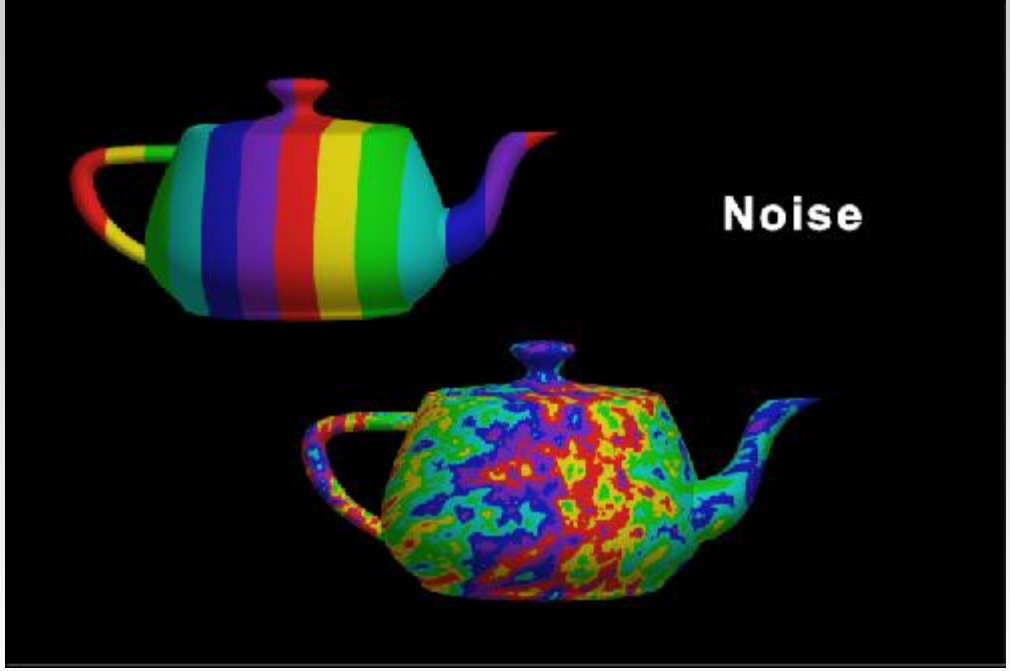
Pour faire des bandes sur la théière de gauche partie entière des x(resp y ou *z)* de chaque point. Si impair rouge; Si pair blanc .

3D



- Pour faire les anneaux, on prend les coord en *x* et *y* (resp x et z, y et z) pour calculer la distance d'un point au centre de l'objet + troncature.
- Si impair rouge; Si pair blanc

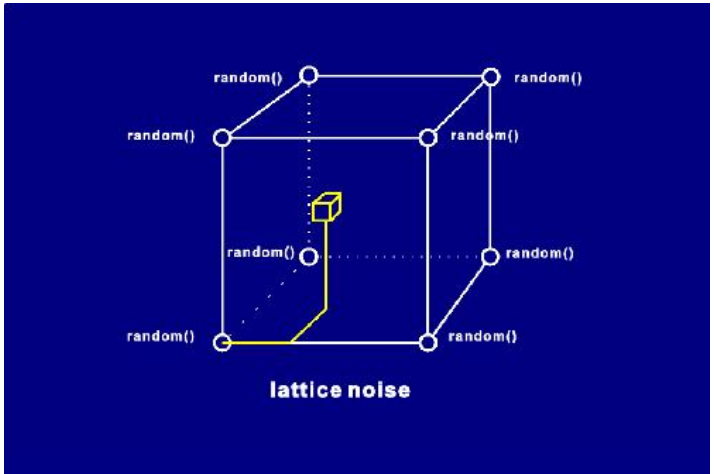
Bruit

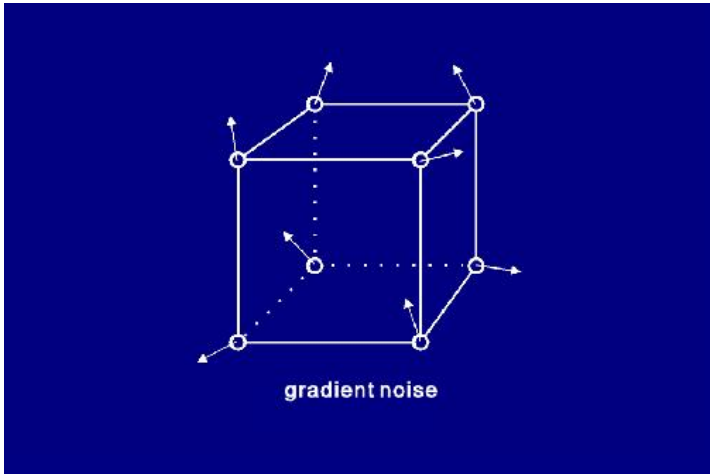


“Lattice noise” : grille de bruit
Valeur aléatoire pour chaque coord entière
Si coord entière : lookup table
Si coord réelle : interpolation trilinéaire

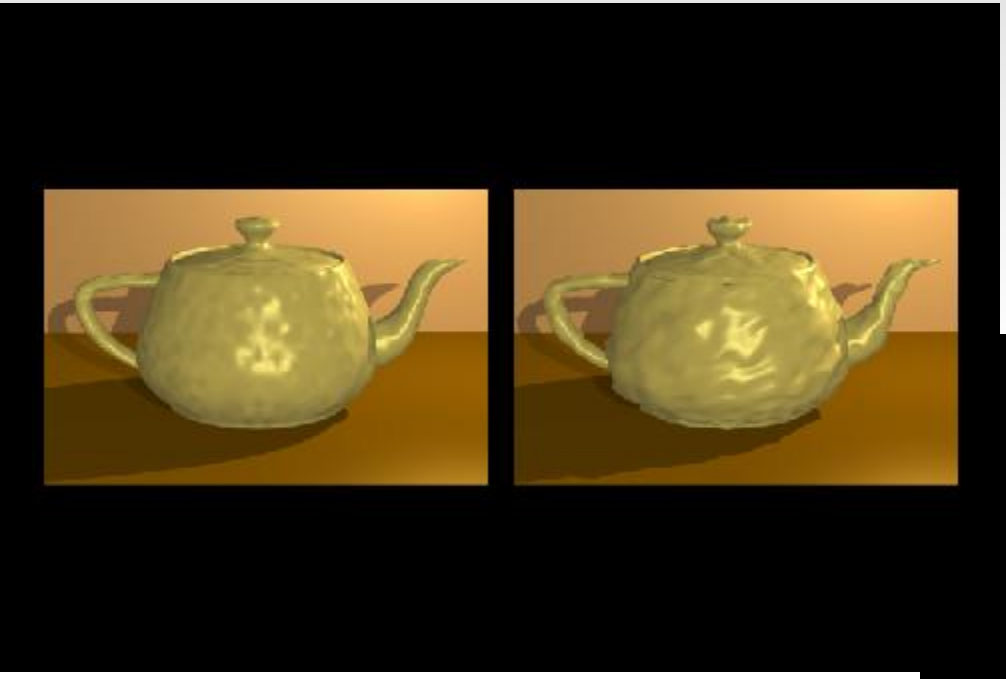
« Gradient noise » : vecteurs de bruit générés pour chaque coord entière

Interpolation trilinéaire





Bump Mapping/ Displacement Mapping



Carte de hauteurs

Modification des normales

En Opengl

placer une image (la texture) sur un objet

image rectangulaire → face rectangulaire facile !

échelle ? Répétition de motif?

objets à faces non rectangulaires ?

Recette (rapide)

Générer les noms de texture (entiers)

Lier (<=>créer) les objets-texture aux données de texture (matrice image et propriétés de texture).

Lier les objets-texture, pour l'utilisation courante des données pour le rendu.

Les Objets-Texture

Objet =

- image [+ images mipmap] + largeur, hauteur, bord, format interne, résolution
- propriétés de texture (filtres, répétition, couleur de bord, priorité de texture).

Les Objets-Textures

déclarer, nommer et rappeler des Objets-Textures.

- **glGenTextures**(GLsizei ***n***, GLuint ****textureNames***) renvoie ***n*** numéros de textures dans ***textureNames[]***

- **glBindTexture**(GL_TEXTURE_2D, GLuint ***textureNames***) :
  - la première fois qu'il est appelé avec une valeur ***textureNames*** non nulle, un nouvel Objet-Texture est crée, et le nom ***textureNames*** lui est attribué.
  - avec une valeur ***textureNames*** déjà liée à une objet-Texture, cet objet devient actif
  - avec la valeur zéro, OpenGL arrête d'utiliser des objets-textures et retourne à la texture par défaut, qui n'a pas de nom.

- **glDeleteTextures**(GLsizei ***n***,const GLuint ****textureNames***)efface ***n*** objets-textures du tableau ***textureNames***

Recette

créer un Objet-Texture et spécifier la texture à utiliser.

choisir le mode de placage de la texture avec **glTexEnv**[f,i,fv,iv]()

- Le mode ***replace*** remplace la couleur de l'objet par celle de la texture.
- Le mode ***modulate*** utilise la valeur de la texture en modulant la couleur précédente de l'objet.
- Le mode ***blend*** mélange la couleur de la texture et la couleur précédente de l'objet.
- Le mode ***decal*** avec une texture RGBA utilise la composante alpha de la texture pour combiner la couleur de la texture et la couleur précédente de l'objet.

autoriser le placage de textures : **glEnable**(GL_TEXTURE_2D) (ou GL_TEXTURE_1D).

spécifier les coordonnées de texture en plus des coordonnées géométriques pour les objets à texturer.

Exemple

```
static GLuint texName[2];
....
glGenTextures(2, texName);                                  // 2 noms créés
glBindTexture(GL_TEXTURE_2D, texName[0]); // création objet 1
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, checkImageWidth,          checkImageHeight,
     0, GL_RGBA, GL_UNSIGNED_BYTE, checkImage);

glBindTexture(GL_TEXTURE_2D, texName[1]);  // création objet 2
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, checkImageWidth,
     checkImageHeight, 0, GL_RGBA, GL_UNSIGNED_BYTE, otherImage);

glEnable(GL_TEXTURE_2D);
```

Répétition de la texture

en dehors de l'intervalle [0.0, 1.0] ?

glTexParameter()

GL_TEXTURE_WRAP_S	« horizontale »
GL_TEXTURE_WRAP_T	« verticale »
GL_CLAMP	sans répétition
GL_REPEAT	avec répétition

Filtrage

si la partie de l'image de texture appliquée sur un pixel est plus petite qu'un texel : agrandissement.

si la partie de l'image de texture appliquée sur un pixel contient plus d'un texel : réduction.

glTexParameter() spécifie les méthodes d'agrandissement (GL_TEXTURE_MAG_FILTER) et de réduction (GL_TEXTURE_MIN_FILTER) utilisées :

- GL_NEAREST choisit le texel le plus proche
- GL_LINEAR calcule une moyenne sur les 2x2 texels les plus proches.

L'interpolation produit des images plus lisses, mais prend plus de temps à calculer.

Mode

créer un Objet-Texture et spécifier la texture à utiliser.

choisir le mode de placage de la texture avec **glTexEnv**[f,i,fv,iv]()

- Le mode ***replace*** remplace la couleur de l'objet par celle de la texture.
- Le mode ***modulate*** utilise la valeur de la texture en modulant la couleur précédente de l'objet.
- Le mode ***blend*** mélange la couleur de la texture et la couleur précédente de l'objet.
- Le mode ***decal*** avec une texture RGBA utilise la composante alpha de la texture pour combiner la couleur de la texture et la couleur précédente de l'objet.

Placage

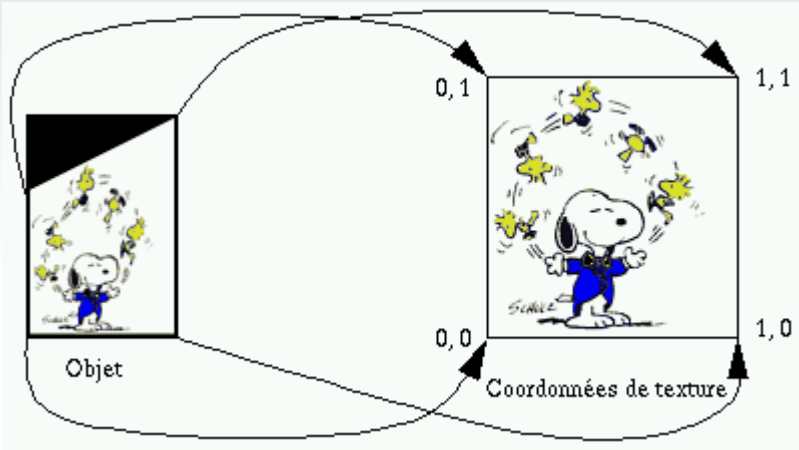
autoriser le placage de textures : **glEnable**(GL_TEXTURE_2D) (ou GL_TEXTURE_1D).

spécifier les coordonnées de texture en plus des coordonnées géométriques pour les objets à texturer.

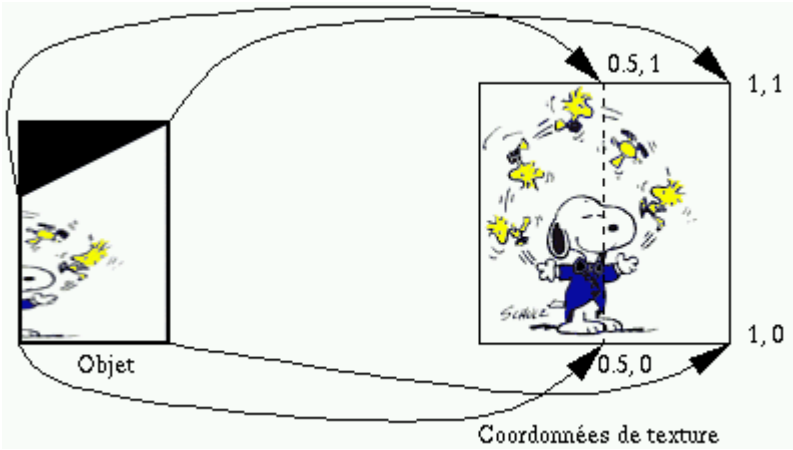
coordonnées de texture

de 0.0 à 1.0 pour chaque dimension.

glTexCoord*(): pour chaque sommet un couple de coordonnées de texture



Tout ou partie de la texture



Exemple

```
glBindTexture(GL_TEXTURE_2D, texName[0]);          // objet texture 1
glBegin(GL_QUADS);                                 // objet géométrique 1
   glTexCoord2f(0.0, 0.0); glVertex3f(-2.0, -1.0, 0.0);
   glTexCoord2f(0.0, 1.0); glVertex3f(-2.0, 1.0, 0.0);
   glTexCoord2f(1.0, 1.0); glVertex3f(0.0, 1.0, 0.0);
   glTexCoord2f(1.0, 0.0); glVertex3f(0.0, -1.0, 0.0);
glEnd();

glBindTexture(GL_TEXTURE_2D, texName[1]);          // objet texture 2
glBegin(GL_QUADS);                                 // objet géométrique 2
   glTexCoord2f(0.0, 0.0); glVertex3f(1.0, -1.0, 0.0);
   glTexCoord2f(0.0, 1.0); glVertex3f(1.0, 1.0, 0.0);
   glTexCoord2f(1.0, 1.0); glVertex3f(2.41421, 1.0, -1.41421);
   glTexCoord2f(1.0, 0.0); glVertex3f(2.41421, -1.0, -1.41421);
 glEnd();
```

Niveaux de détail

Selon la distance à la caméra taille + ou - importante de l'image de texture utilisée (mipmapping)

Soit fournir les images de texture aux différentes tailles (**glTexImage2D**() à différentes résolutions),

ou bien **gluBuild2DMipmaps**(), qui calcule les images réduites.

Et le **Filtrage *?***

- Avec les mipmaps, 4 autres filtres :
  - GL_NEAREST_MIPMAP_NEAREST et GL_LINEAR_MIPMAP_NEAREST utilisent l'image mipmap la plus proche, et dans cette image, le premier choisit le texel le plus proche, et le deuxième réalise une interpolation sur les quatre texels les plus proches.
  - GL_NEAREST_MIPMAP_LINEAR et GL_LINEAR_MIPMAP_LINEAR travaillent à partir de l'interpolation de 2 mipmaps.

Coordonnées de Texture

Par *glTexGen*(coordinate, parameter, value)*

- Coordinate : GL_R, GL_S, GL_T or GL_Q.
- Parameter : GL_TEXTURE_GEN_MODE :
  - GL_OBJECT_LINEAR : fixe la texture sur un objet. L'objet sert de référence => coordonnées générées sur l'objet indépendamment de sa position/rotation (plan GL_OBJECT_PLANE pour définir la direction de projection de la texture).
  - GL_EYE_LINEAR : texture « mouvante » selon la position de caméra (eye) = référence => coordonnées générées en fonction de la position de la caméra (plan GL_EYE_PLANE pour définir la direction de projection de la texture).
- Direction de projection:
  - Coordinate : GL_R, GL_S, GL_T or GL_Q.
  - Parameter : GL_OBJECT_PLANE ou GL_EYE_PLANE
  - Value : équation du plan

Coordonnées de Texture

GL_SPHERE_MAP environment mapping.

```
glBindTexture(GL_TEXTURE_2D, texture);
glEnable(GL_TEXTURE_2D);

//Sphere mapping and enable s & t texture generation
glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP);
glTexGeni(GL_T, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP);
glEnable(GL_TEXTURE_GEN_S);
glEnable(GL_TEXTURE_GEN_T);

//Draw the shapes to apply the texture
 ...
glDisable(GL_TEXTURE_GEN_S);
glDisable(GL_TEXTURE_GEN_T);
glDisable(GL_TEXTURE_2D);
```

Coordonnées de Texture

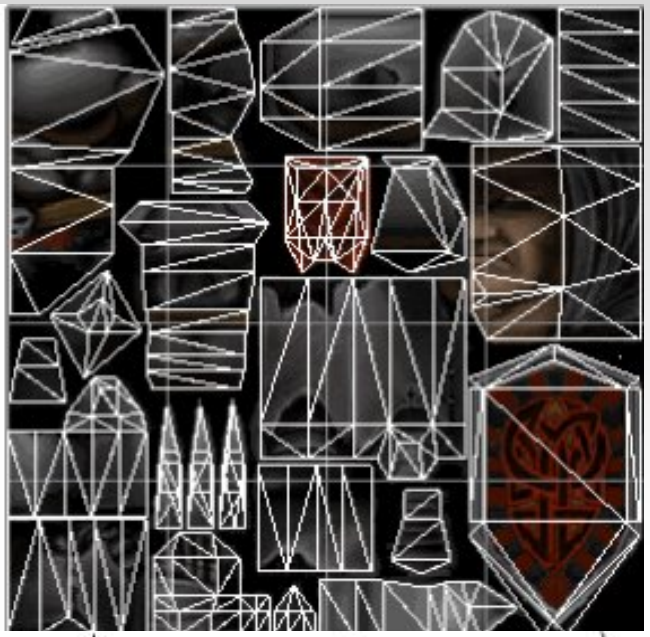
GL_REFLECTION_MAP reflection mapping.

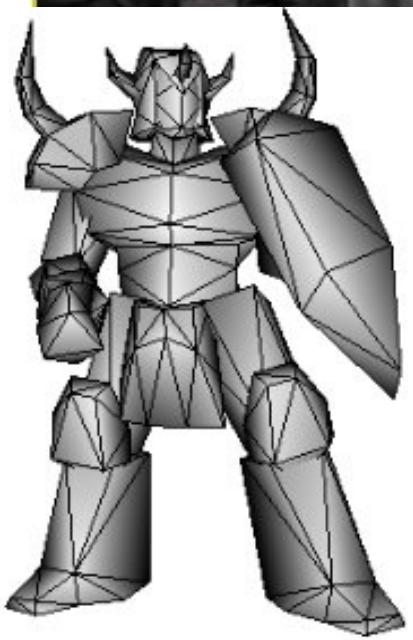
```
 glBindTexture( GL_TEXTURE_CUBE_MAP_POSITIVE_X, cubeMap[0]);
 glBindTexture( GL_TEXTURE_CUBE_MAP_NEGATIVE_X, cubeMap[1]);
 glBindTexture( GL_TEXTURE_CUBE_MAP_POSITIVE_Y, cubeMap[2]);
 glBindTexture( GL_TEXTURE_CUBE_MAP_NEGATIVE_Y, cubeMap[3]);
 glBindTexture( GL_TEXTURE_CUBE_MAP_POSITIVE_Z, cubeMap[4]);
 glBindTexture( GL_TEXTURE_CUBE_MAP_NEGATIVE_Z, cubeMap[5]);
 glEnable( GL_TEXTURE_CUBE_MAP);

//GL_REFLECTION_MAP for s,t,r texture coordinates
glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_REFLECTION_MAP);
glTexGeni(GL_T, GL_TEXTURE_GEN_MODE, GL_REFLECTION_MAP);
glTexGeni(GL_R, GL_TEXTURE_GEN_MODE, GL_REFLECTION_MAP);
glEnable(GL_TEXTURE_GEN_S);
glEnable(GL_TEXTURE_GEN_T);
glEnable(GL_TEXTURE_GEN_R);

//Draw shape to apply reflection on it
...
glDisable(GL_TEXTURE_GEN_S);
glDisable(GL_TEXTURE_GEN_T);
glDisable(GL_TEXTURE_GEN_R);
...
 glDisable( GL_TEXTURE_CUBE_MAP);
```

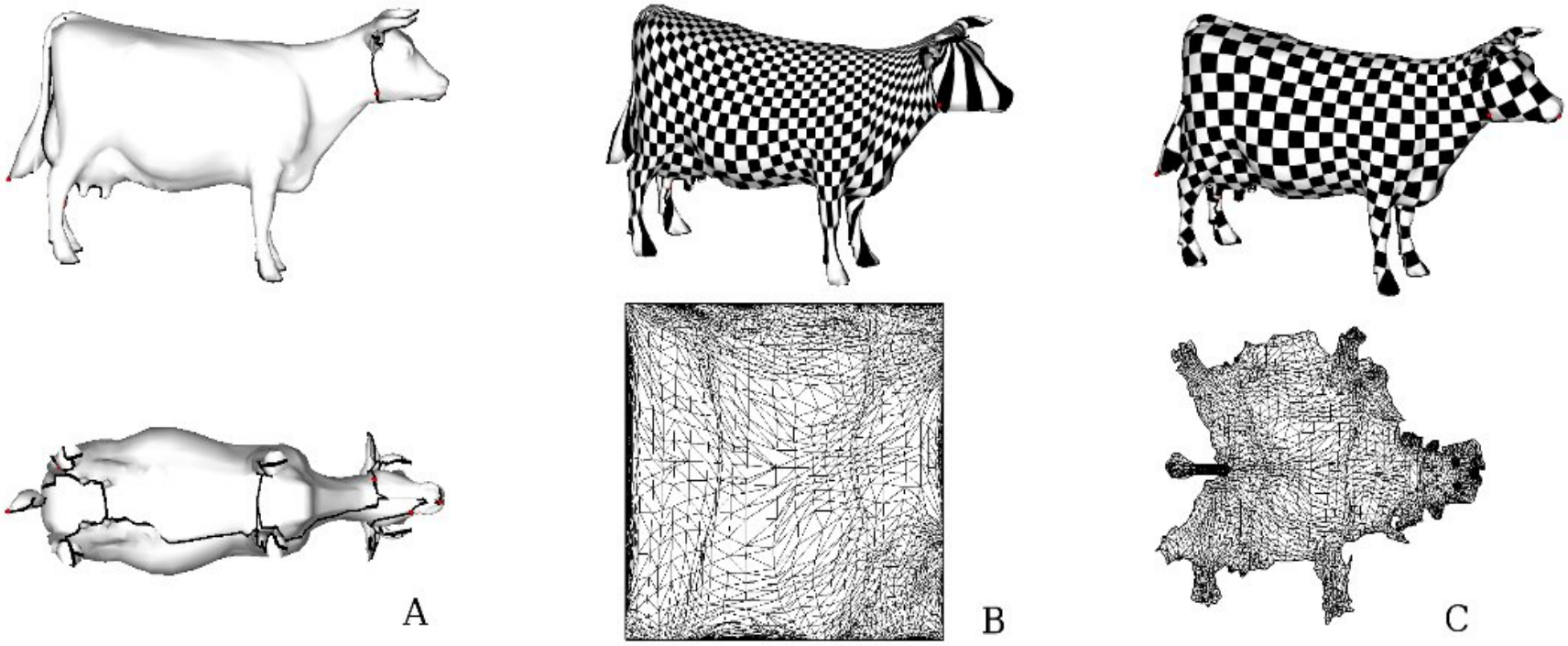
Automatisation?



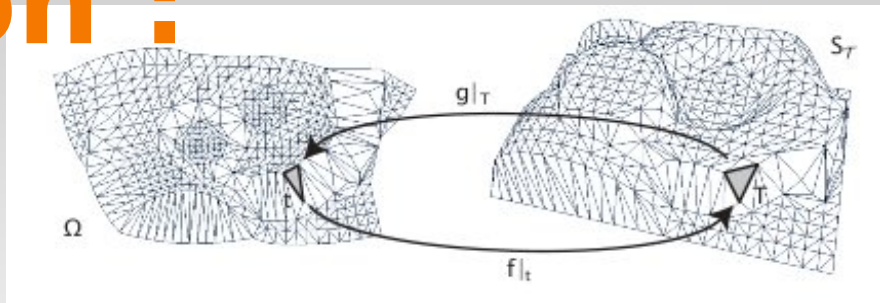


Application de Textures

Découper le maillage en parties homéomorphes à un disque puis paramétrisation



Paramétrisation : Principe



Arête ~ressort : $u_i=(u_i,v_i)$ paramètres des sommets p_i du maillage

M_T (i=1,..,n = intérieur; i=n+1,..n+b=bord)

Equilibre global : $$E=\frac{1}{2}\sum_{i=1}^{n}\sum_{j\in N_i}\frac{1}{2}D_{ij}\|u_i-u_j\|^2$$

Comme $$\frac{\partial E}{\partial u_i}=\sum_{j\in N_i}D_{ij}\left(u_i-u_j\right)$$ le minimum est atteint pour

$$\sum_{j\in N_i}D_{ij}u_i=\sum_{j\in N_i}D_{ij}u_j \quad \forall i=1,\cdots,n$$

C'est à dire pour $$u_i=\sum_{j\in N_i}\lambda_{ij}u_j \quad avec \quad \lambda_{ij}=\frac{D_{ij}}{\sum_{k\in N_i}D_{ik}}$$

Principe

On sépare intérieur et bord

$$u_i - \sum_{j \in N_i, j \leq n} \lambda_{ij} u_j = \sum_{j \in N_i, j > n} \lambda_{ij} u_j$$

On résoud

$$\tilde{U}$$

$$avec \quad U = \left(u_{1,} \cdots, u_n\right)^T \quad et \quad \tilde{U} = \left(\tilde{u}_{1,} \cdots, \tilde{u}_n\right)^T \quad les \quad inconnues$$

$$AU =$$

Tutte

Quel choix pour les D_{ij} ? (<=> quels λ_{ij} ?)

ppté de reproduction linéaire d'une paramétrisation :

supposons M_T / $\mathbf{p}_i=(x_i,y_i,0)$. Paramétrisation :

$$u_i = x_i = (x_i, y_i) \quad \forall i \in \{1, n+b\}$$
$$x_i = \sum_{j \in N_i} \lambda_{ij} x_j \quad \wedge \quad \sum_{j \in N_i} \lambda_{ij} = 1 \quad i \in \{1, n\}$$

$D_{ij} = 1$: algorithme barycentrique de Tutte pour les graphes

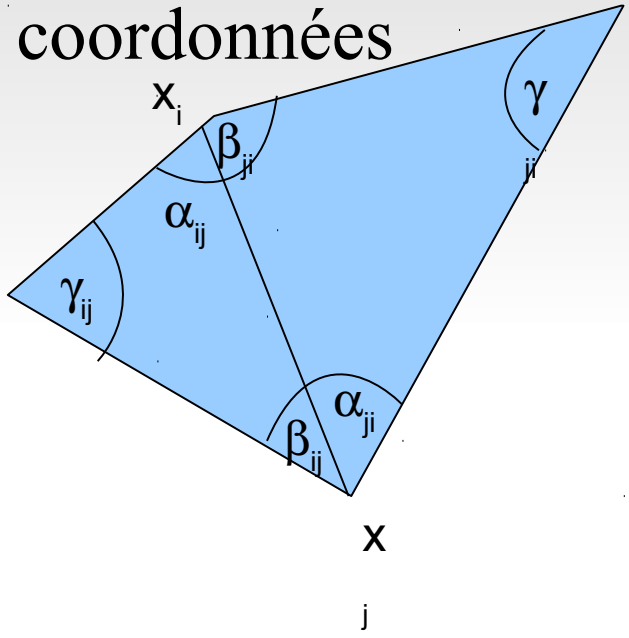
$D_{ij} = k. \|\mathbf{x}_i - \mathbf{x}_j\|$ (longueur des arêtes) : Greiner& Hormann

Ne satisfont pas la ppté précédente

$$\lambda_{ij} = \frac{w_{ij}}{\sum_{k \in N_i} w_{ik}}$$

Application barycentrique

=> polygone avec #sommets > 3 : plusieurs solutions pour barycentre : généralisation des coordonnées barycentriques



- Coord de Wachspress

$$w_{ij} = \frac{\cot \alpha_{ji} + \cot \beta_{ij}}{r_{ij}^2}$$

- Coord Harmoniques discrètes

$$w_{ij} = \cot \gamma_{ji} + \cot \gamma_{ij}$$

- Coord moyennes

$$w_{ij} = \frac{\tan \alpha_{ij} + \tan \beta_{ji}}{r_{ij}}$$

Problèmes

w_{ij} ← angles et distances sur triangles de Ni

$\mathbf{x}_i$ pas nécessairement coplanaires

Pbs si non coplanarité:

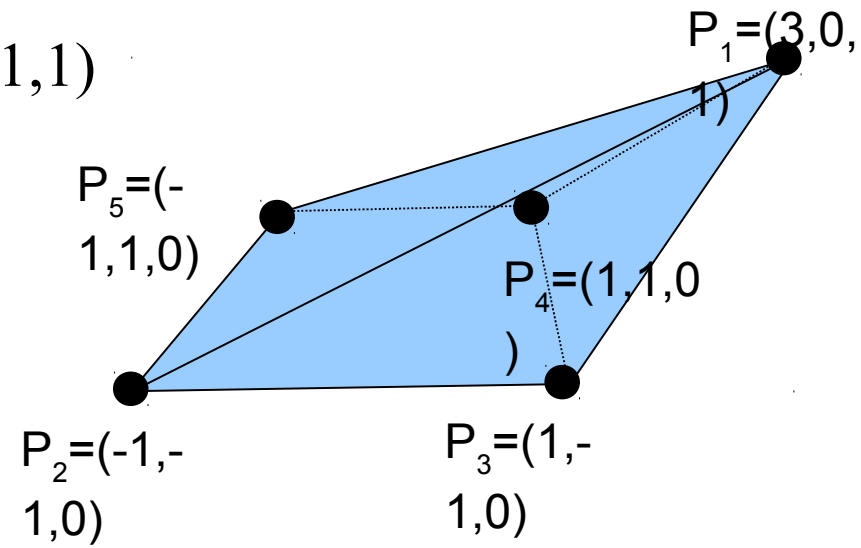
$u_2=(-1,-1)$, $u_3=(1,-1)$, $u_4=(1,1)$, $u_5=(-1,1)$

on calcule $\lambda_{12},\lambda_{13},\lambda_{14},\lambda_{15}$

$u_{1wachspress} = (-35.1369,0)$

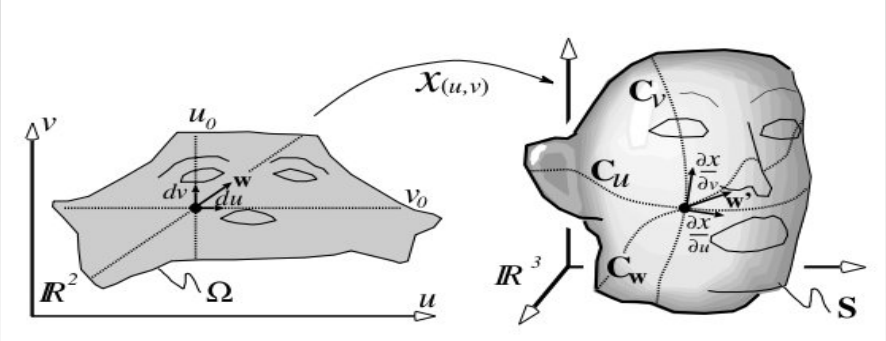
$u_{1harmonique}=(2.1138,0)$

$u_{1moyenne}=(0.4538,0)$ Seulement coord moyennes → bijection



Transformation conforme

Déplacement élémentaire de (u,v) selon les axes u, v transformés en vecteurs tangents aux courbes iso-u et iso-v passant par le point f(u,v)



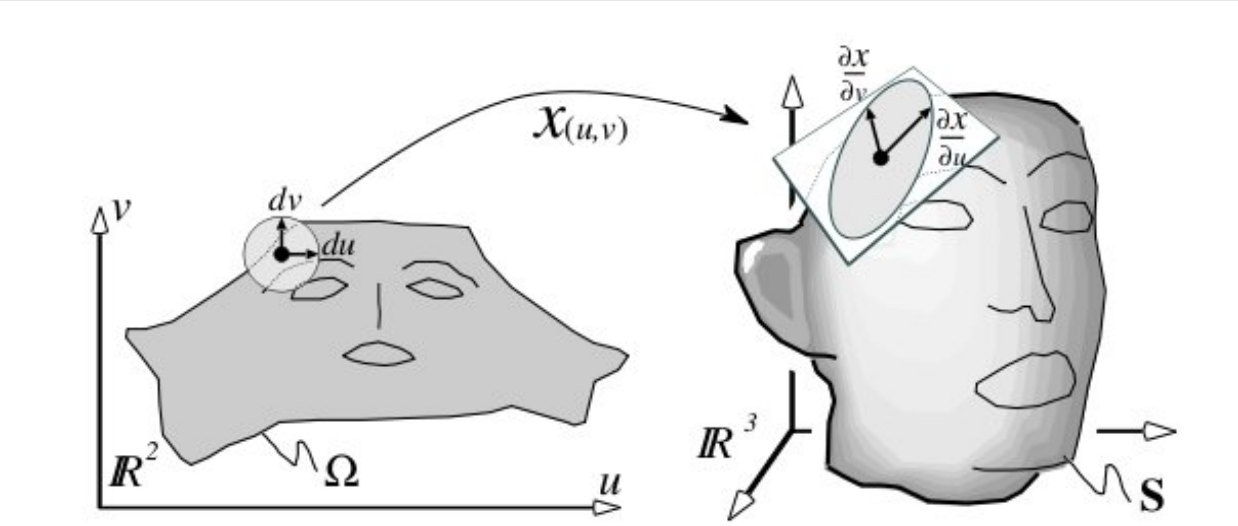
$$w' = J(u_0, v_0)\, w \quad avec \quad J(u_0, v_0) = \begin{pmatrix} \frac{\partial x}{\partial u}(u_0, v_0) & \frac{\partial x}{\partial v}(u_0, v_0) \\ \frac{\partial y}{\partial u}(u_0, v_0) & \frac{\partial y}{\partial v}(u_0, v_0) \\ \frac{\partial z}{\partial u}(u_0, v_0) & \frac{\partial z}{\partial v}(u_0, v_0) \end{pmatrix} = \begin{pmatrix} f_u(u_0, v_0) & \vdots & f_v(u_0, v_0) \end{pmatrix}$$

$$I(u_0, v_0) = J^T J = \begin{pmatrix} f_u \cdot f_u & f_u \cdot f_v \\ f_v \cdot f_u & f_v \cdot f_v \end{pmatrix}$$

1e forme fondamentale : tenseur métrique ~ mesure les déformations:

$$en \ \ distance \ : \ \|w'\|^2 = w^T I w$$
$$et \ \ angle \ : \ w'^T_1 w'_2 = w^T_1 I w_2$$

Transformation conforme

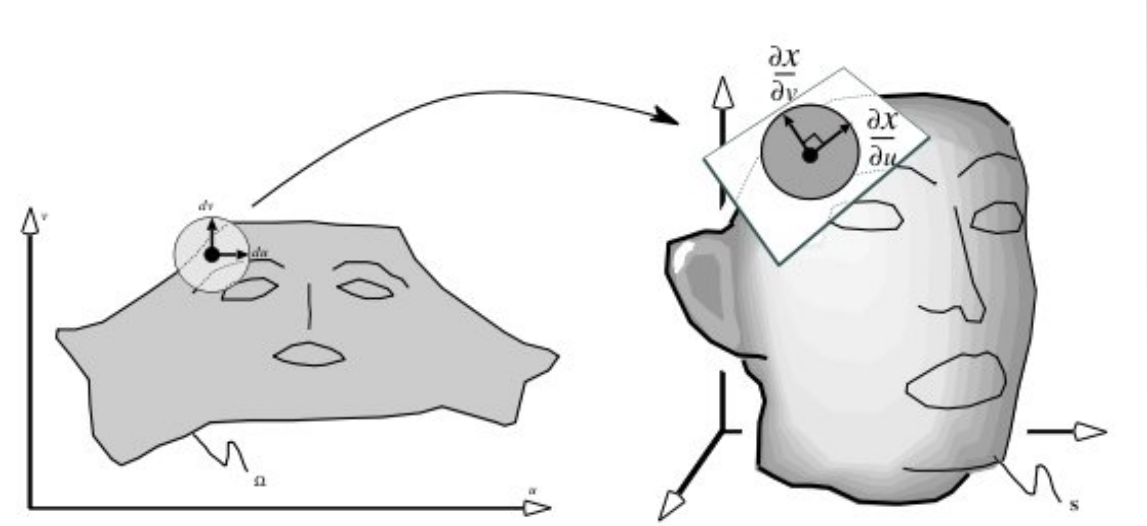


axes : Jw_1 *et* Jw_2

longueurs : $\sqrt{(\lambda_1)}$ *et* $\sqrt{(\lambda_2)}$ *valeurs propres de* I

=>Analyse de la déformation

Transformation conforme



Conformalité: $f_u \perp f_v \ \wedge \ \|f_u\| = \|f_v\|$

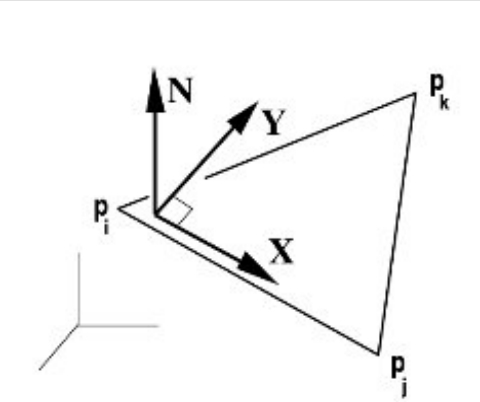
$$f_u = n \times f_v$$

$$J = R.S \quad \textit{similitude} \quad \textit{locale}$$

$$\lambda_1 = \lambda_2$$

Riemann : toute surface admet une paramétrisation conforme. Mais dans le cas discret?

Cas des Triangulations



repère local :

$$X = \frac{p_j - p_i}{\|p_j - p_i\|}$$

$$N = \frac{X \times (p_k - p_i)}{X \times (p_k - p_i)}$$

$$Y = N \times X$$

$$(X,Y) \rightarrow (u,v)$$

$$\begin{cases} u(X,Y) = \lambda_1 u_i + \lambda_2 u_j + \lambda_3 u_k \\ v(X,Y) = \lambda_1 v_i + \lambda_2 v_j + \lambda_3 v_k \end{cases}$$

$$\begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{pmatrix} = \frac{1}{2|T|_{X,Y}} \begin{pmatrix} Y_j - Y_k & X_k - X_j & X_j Y_k - X_k Y_j \\ Y_k - Y_i & X_i - X_k & X_k Y_i - X_i Y_k \\ Y_i - Y_j & X_j - X_i & X_i Y_j - X_j Y_i \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

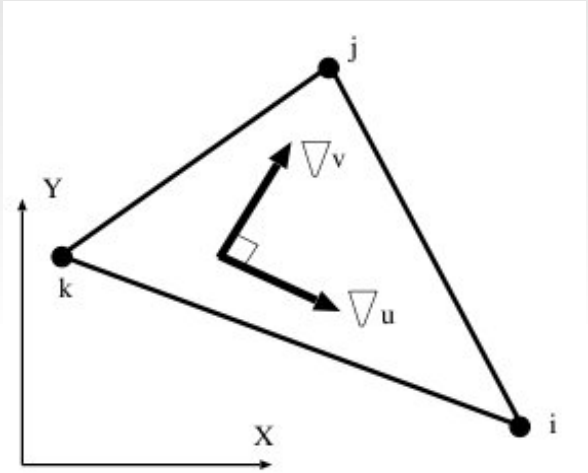
avec $2|T|_{X,Y} = (X_i Y_j - Y_i X_j) + (X_j Y_k - Y_j X_k) + (X_k Y_i - Y_k X_i)$

avec $2|T|_{X,Y} = 2.\ aire\ \ triangle$

$$\begin{pmatrix} \frac{\partial u}{\partial X} \\ \frac{\partial u}{\partial Y} \end{pmatrix} = M_T \begin{pmatrix} u_i \\ u_j \\ u_k \end{pmatrix} = \frac{1}{2|T|_{X,Y}} \begin{pmatrix} Y_j - Y_k & Y_k - Y_i & Y_i - Y_j \\ X_k - X_j & X_i - X_k & X_j - X_i \end{pmatrix} \begin{pmatrix} u_i \\ u_j \\ u_k \end{pmatrix}$$

M_T dépend uniquement de la géométrie de T

Application Conforme (moindres carrés)



$$\nabla v = rot_{90}(\nabla u) = \begin{pmatrix} 0 & -1 \\ 1 & 1 \end{pmatrix} \nabla u$$

$$M_T \begin{pmatrix} v_i \\ v_j \\ v_k \end{pmatrix} - \begin{pmatrix} 0 & -1 \\ 1 & 1 \end{pmatrix} M_t \begin{pmatrix} u_i \\ u_j \\ u_k \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Toute surface continue admet une paramétrisation conforme.
Mais dans le cas discret seulement les surfaces développables
Cas général : minimiser la ”non-conformalité”

$$E_{LSCM} = \sum_{T=(i,j,k)} |T| \left\| M_T \begin{pmatrix} v_i \\ v_j \\ v_k \end{pmatrix} - \begin{pmatrix} 0 & -1 \\ 1 & 1 \end{pmatrix} M_t \begin{pmatrix} u_i \\ u_j \\ u_k \end{pmatrix} \right\|^2$$

Segmentation

- Germes
- Méthodes gloutonnes
- Voronoi

.....

=> ensemble de ”disques”

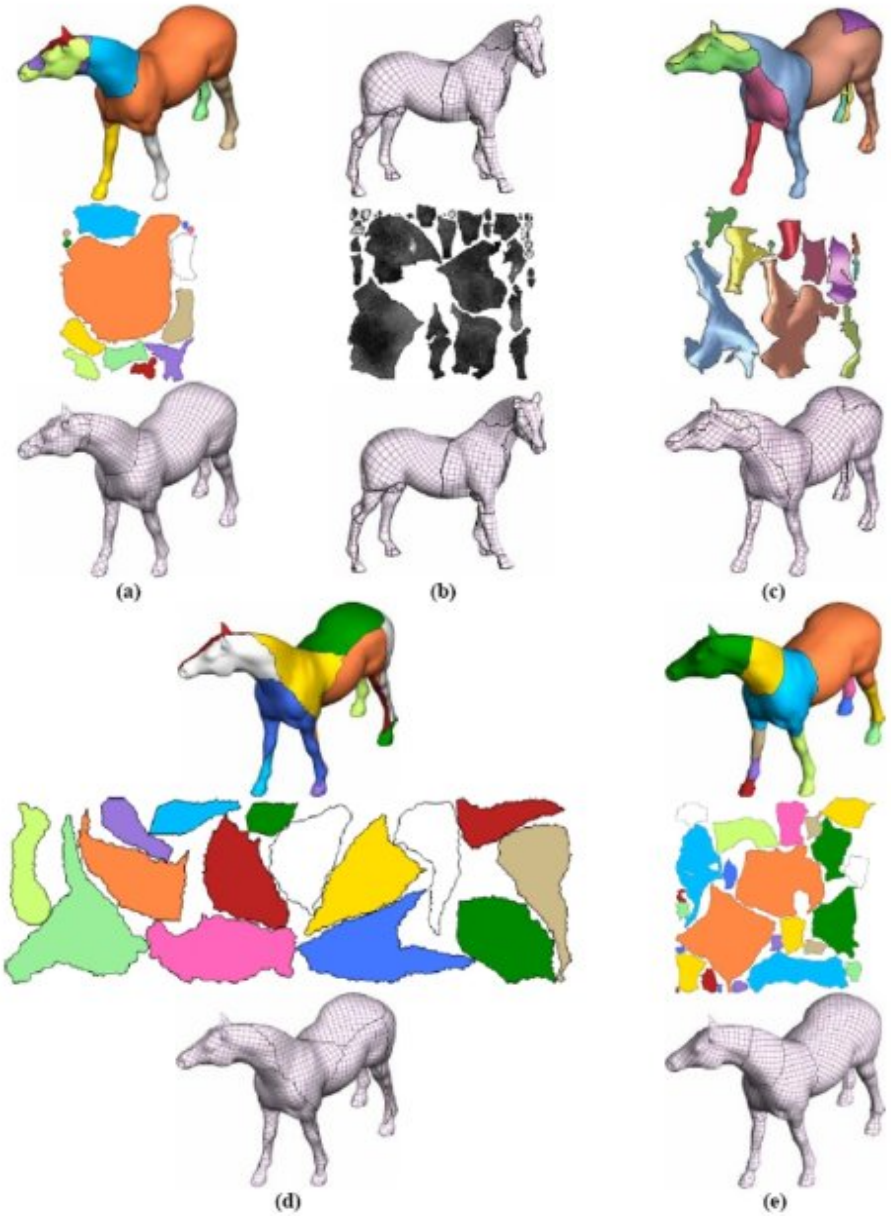
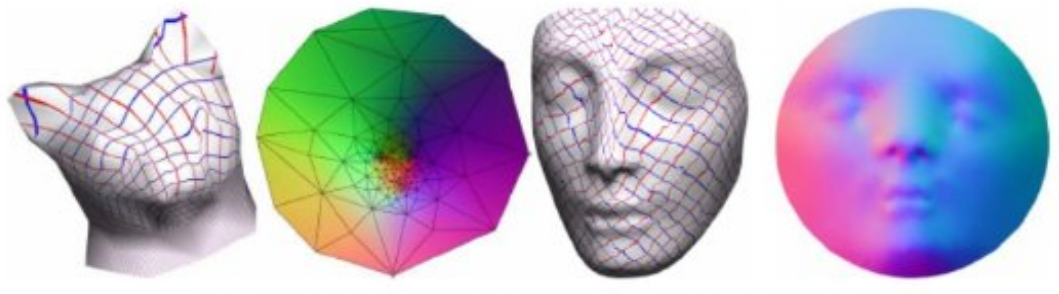
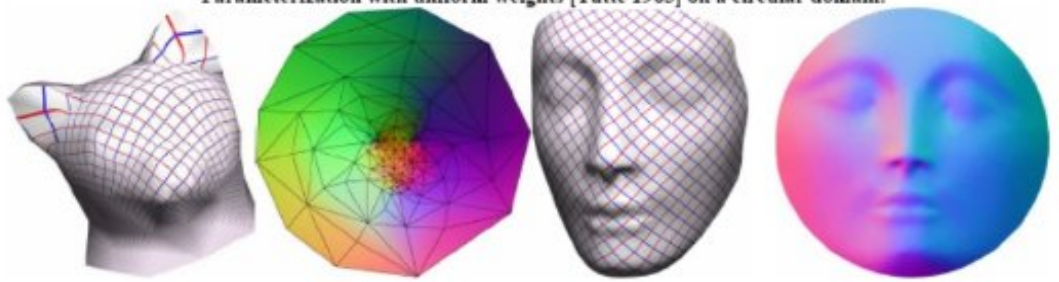


Figure 6.1: Multi-chart segmentations (a) Isocharts [Zhou et al., 2004] – 13 charts; (b) [Lévy et al., 2002] – 44 charts; (c) D-charts [Julius et al., 2005] – 12 charts; (d) multi-chart geometry images [Sander et al., 2003] – 15 charts; and (e) [Zhang et al., 2005] – 24 charts.

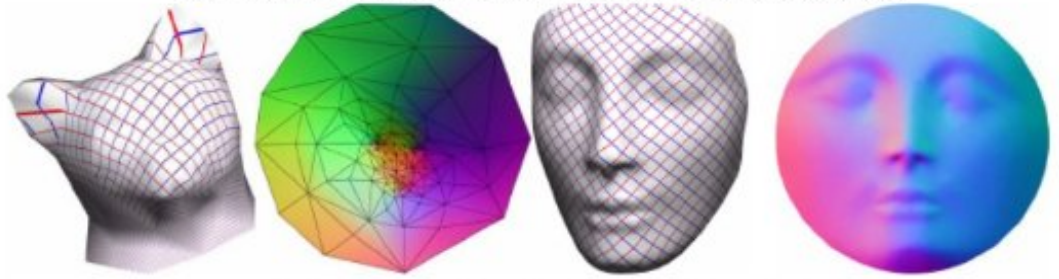
Comparaison de paramé-trisations



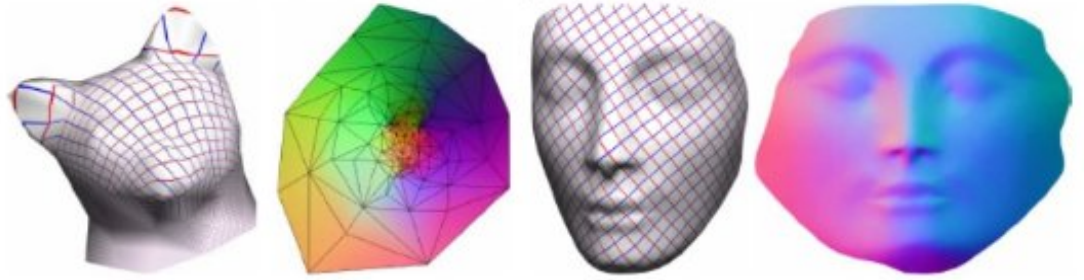
Parameterization with uniform weights [Tutte 1963] on a circular domain.



Parameterization with harmonic weights [Eck et al. 1995] on a circular domain.



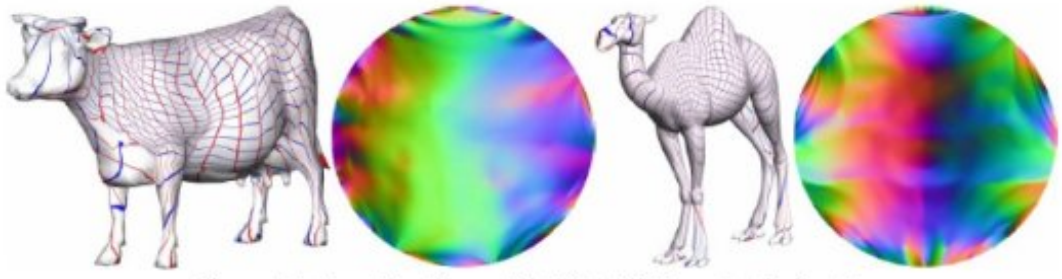
Parameterization with mean value weights [Floater 2003] on a circular domain.



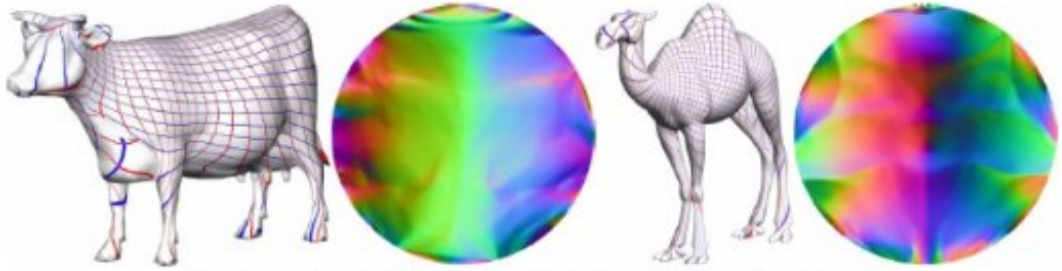
Parameterization with LSCM [Lévy et al. 2002].

Figure 7.1: Surfaces with nearly convex boundaries parameterized with linear methods (images made with Graphite, http://alice.loria.fr/software).

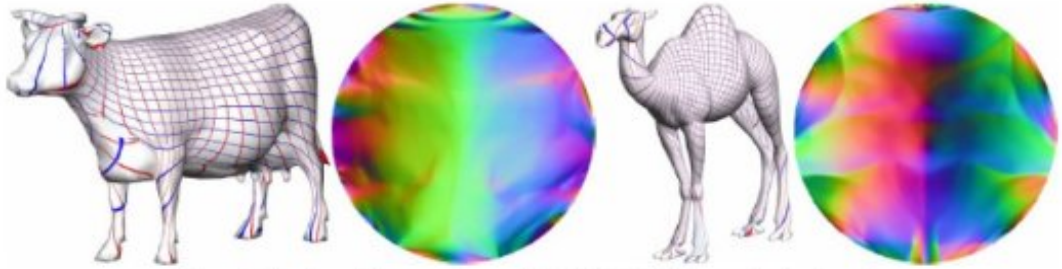
Comparaison de paramé-trisations



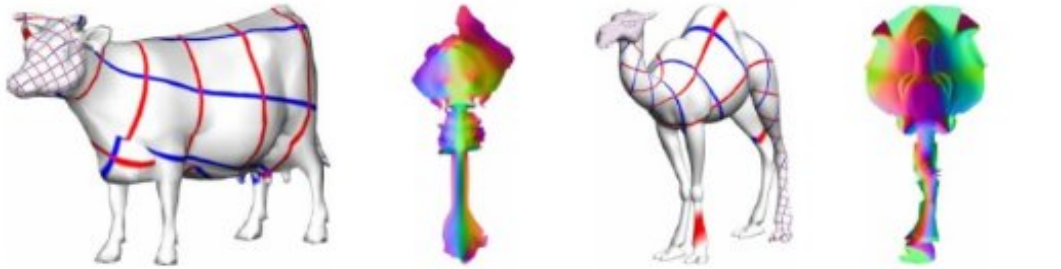
Parameterization with uniform weights [Tutte 1963] on a circular domain.



Parameterization with harmonic weights [Eck et al. 1995] on a circular domain.



Parameterization with mean value weights [Floater 2003] on a circular domain.



Parameterization with LSCM [Lévy et al. 2002].

Figure 7.2: Surfaces with non-convex boundaries parameterized with linear methods (images made with Graphite, http://alice.loria.fr/software).