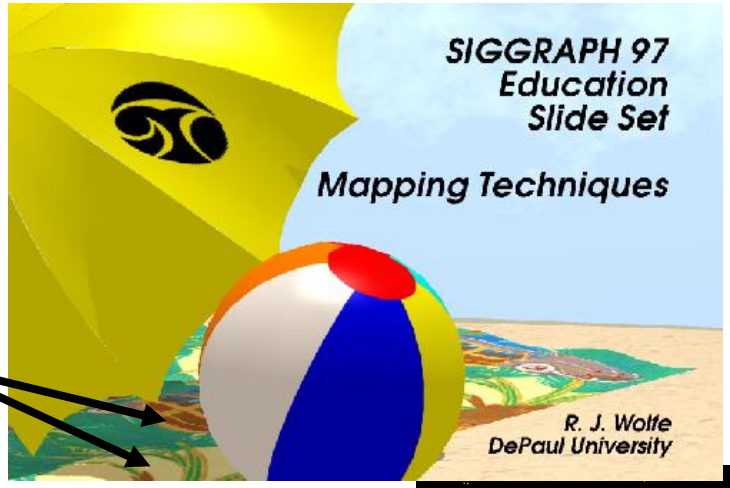
Textures

Modélisation de l'apparence

Ne pas tout modéliser à l'échelle de la géométrie !

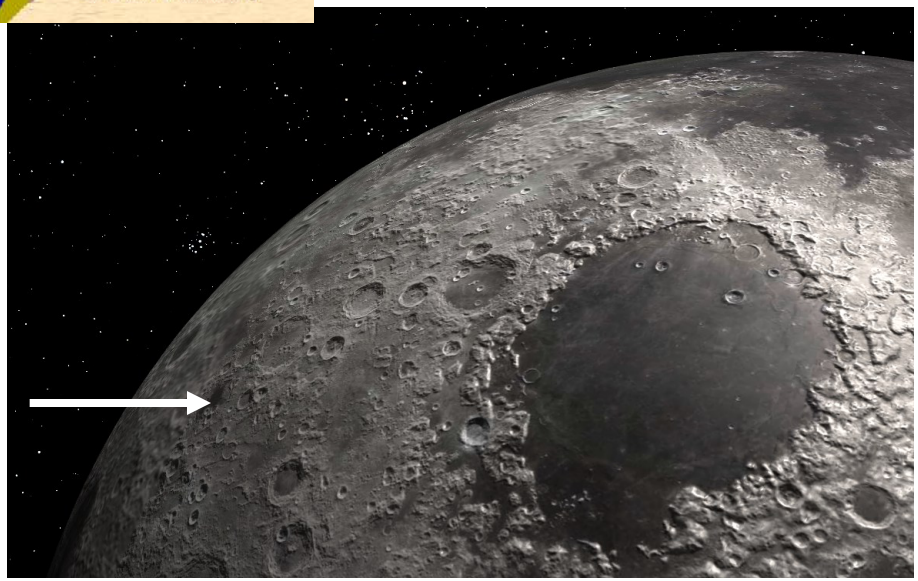
On veut garder une seule face, mais plusieurs couleurs



⟹ **Texture de couleurs**

Des micro-polygones seraient nécessaires

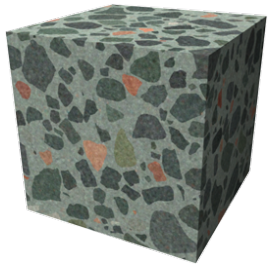
⟹ **Texture de normales**



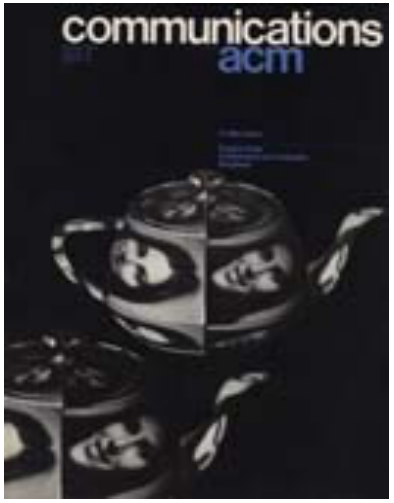
Les textures

Champ scalaire discret *(lookup table)*

- Défini sur un domaine
  - linéaire (1D)
  - rectangulaire (2D)
  - cubique (3D)



- Appliqué sur la surface par des **coordonnées de texture**
  - spécifiées en chaque sommet
  - interpolées par fragment



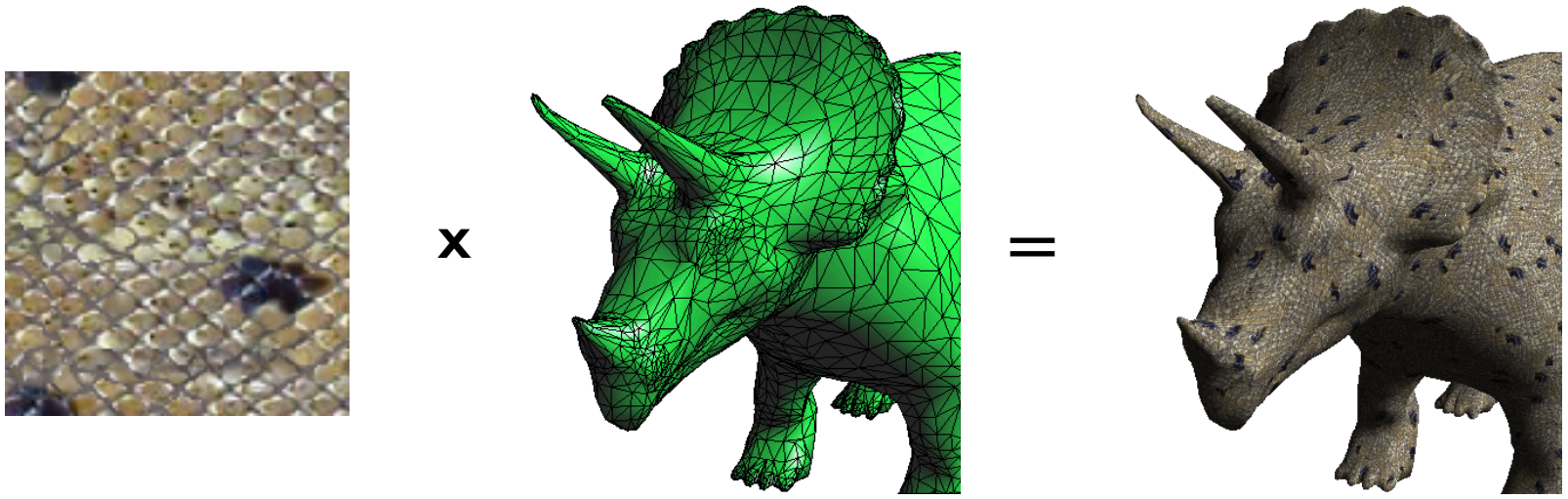
Texture and reflection in computer generated images.
J. F. Blinn & M.E. Newell. 1976

Les textures

Ajout d'information visuelle à **petit prix**

Support matériel

- interpolation des coordonnées de texture
- interpolation des valeurs de couleur
- filtrage multi-résolution (*mip-mapping*)



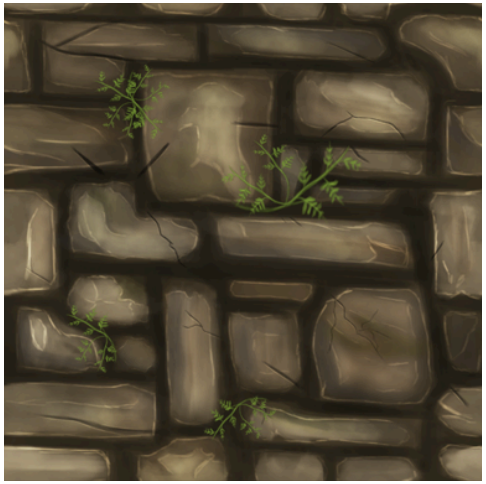
Les textures

Peuvent être utilisées pour spécifier :

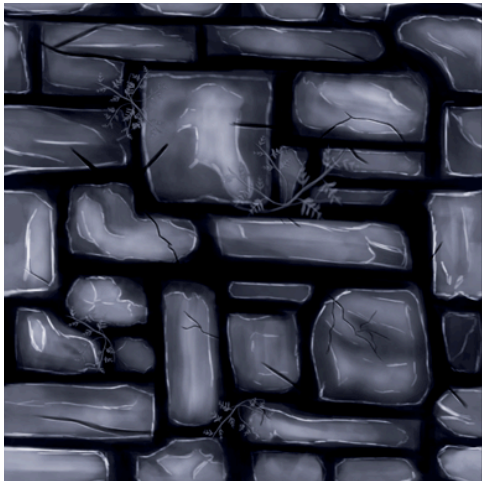
Les textures

Peuvent être utilisées pour spécifier :

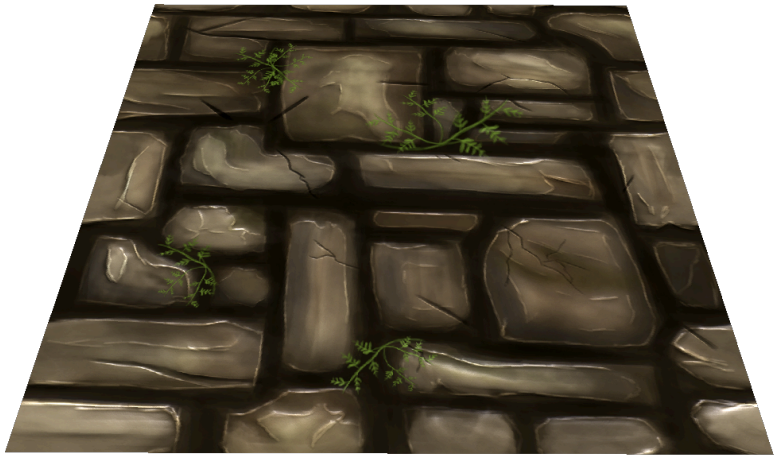
- **La couleur :** ambiante / diffuse, la brillance, la transparence



Diffuse Map



Specular Map

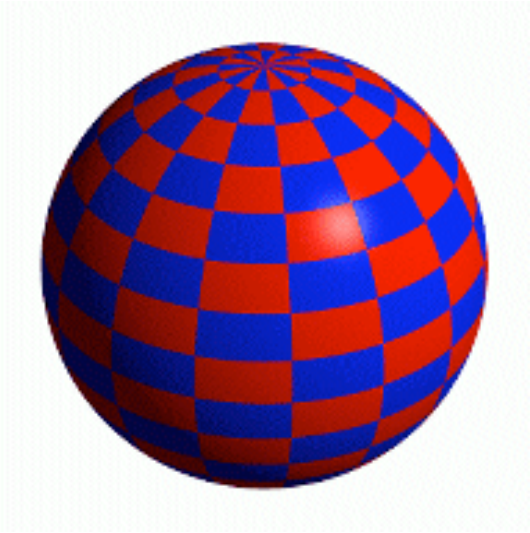


http://emiliozedan.blogspot.fr/2012/11/hand-painted-texture-stone.html

Les textures

Peuvent être utilisées pour spécifier :

- La couleur
- **Les normales (*bump / normal mapping*)**



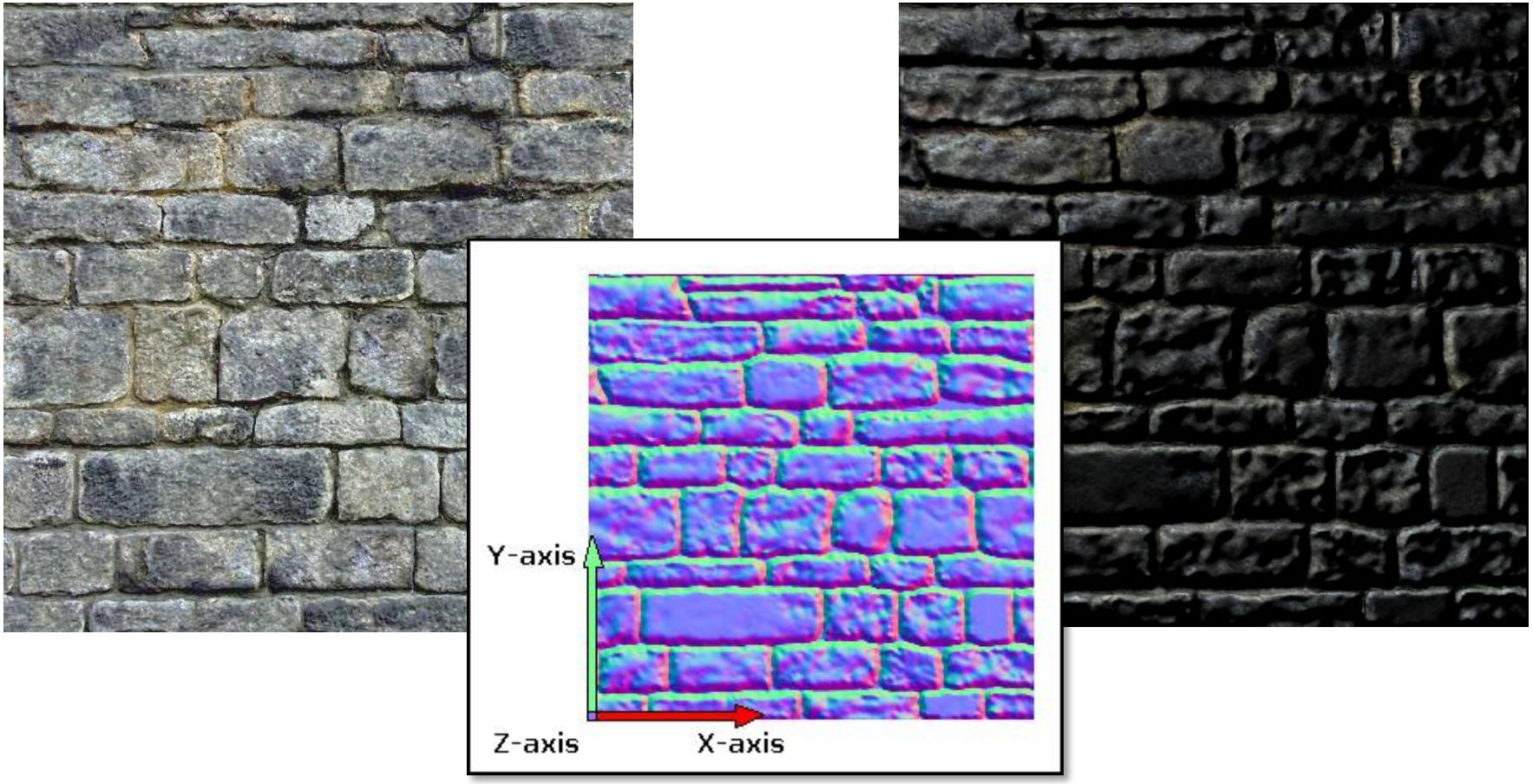
Texture Diffuse mappée sur la sphère

Bump Map

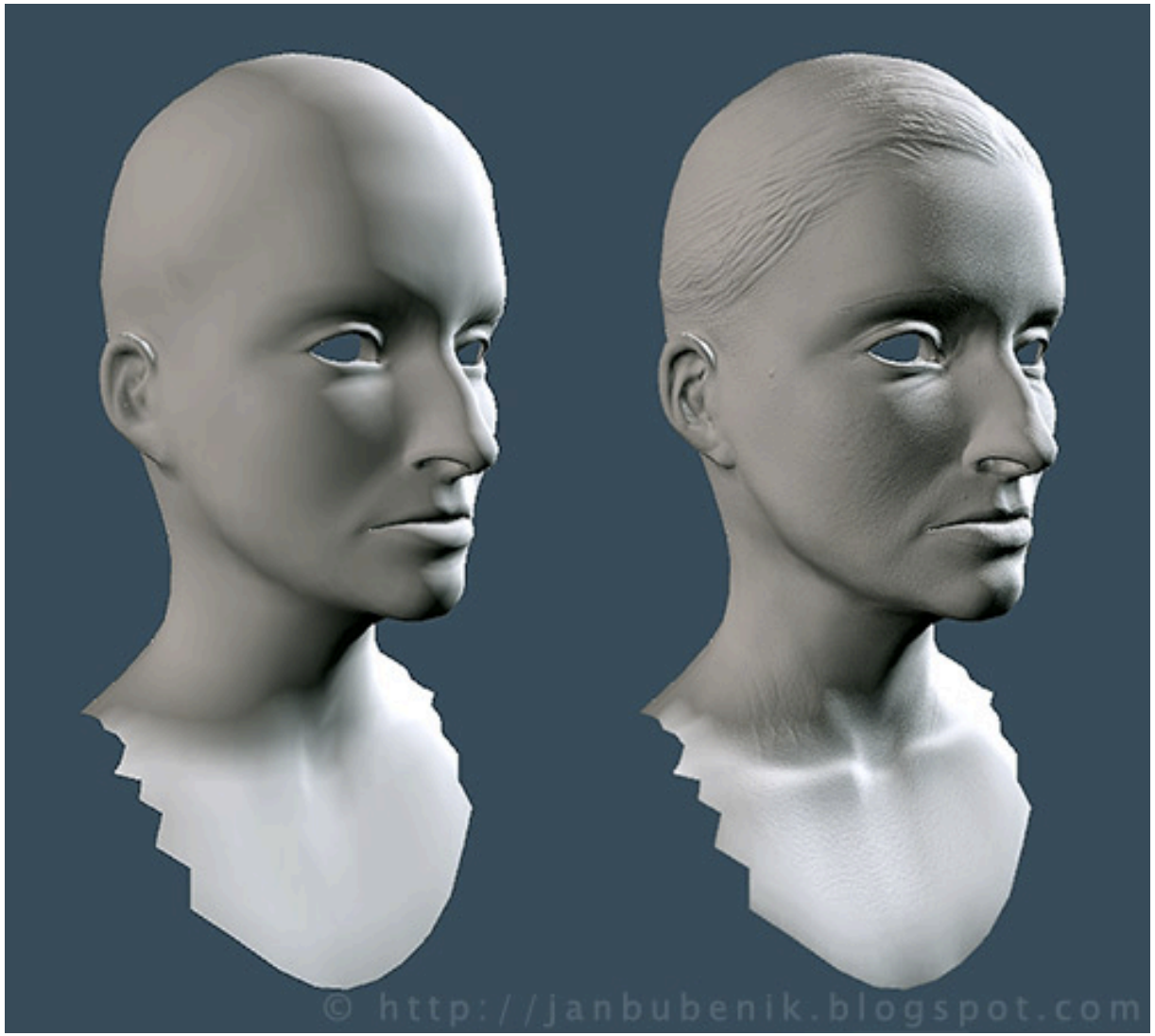
Textures combinées

Perturbation des normales

Donnée par une texture de normale *(normal map)*



Perturbation des normales

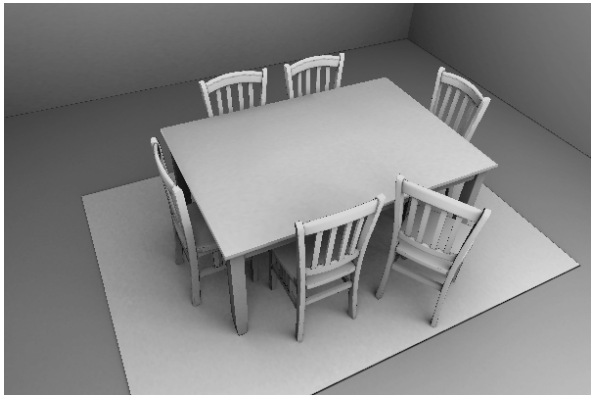


Les textures

Peuvent être utilisées pour spécifier :

- La couleur
- Les normales *(bump / normal mapping)*
- **Une illumination pré-calculée *(light mapping)***

Texture diffuse appliquée à la scène



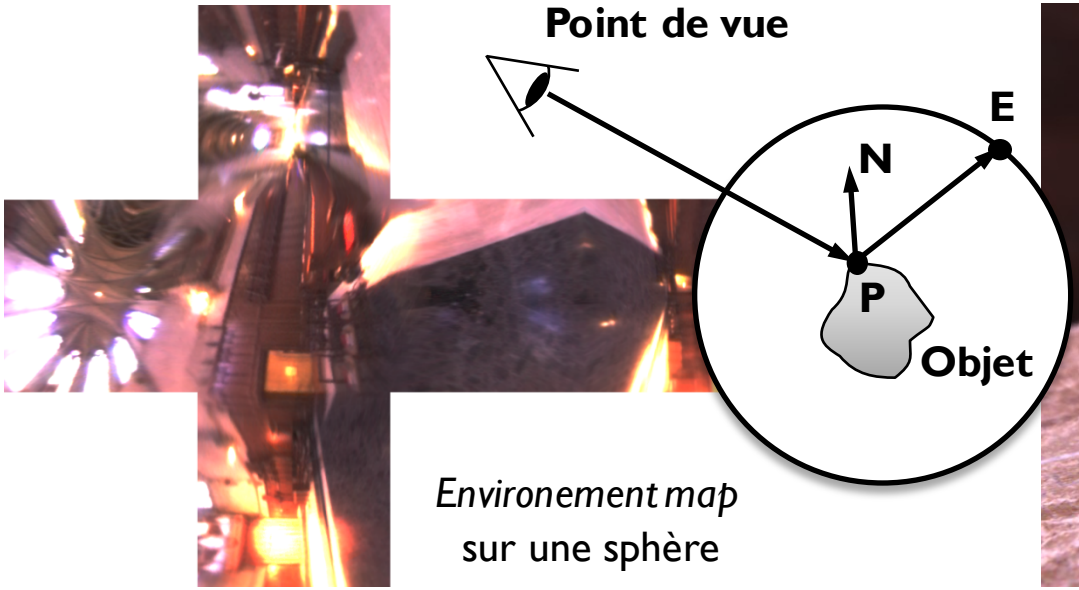
Light map appliquée à la scène

Textures combinées

Les textures

Peuvent être utilisées pour spécifier :

- La couleur
- Les normales *(bump / normal mapping)*
- Une illumination pré-calculée *(light mapping)*
- **Les réflexions *(environment mapping)***



Environement map
sur une sphère

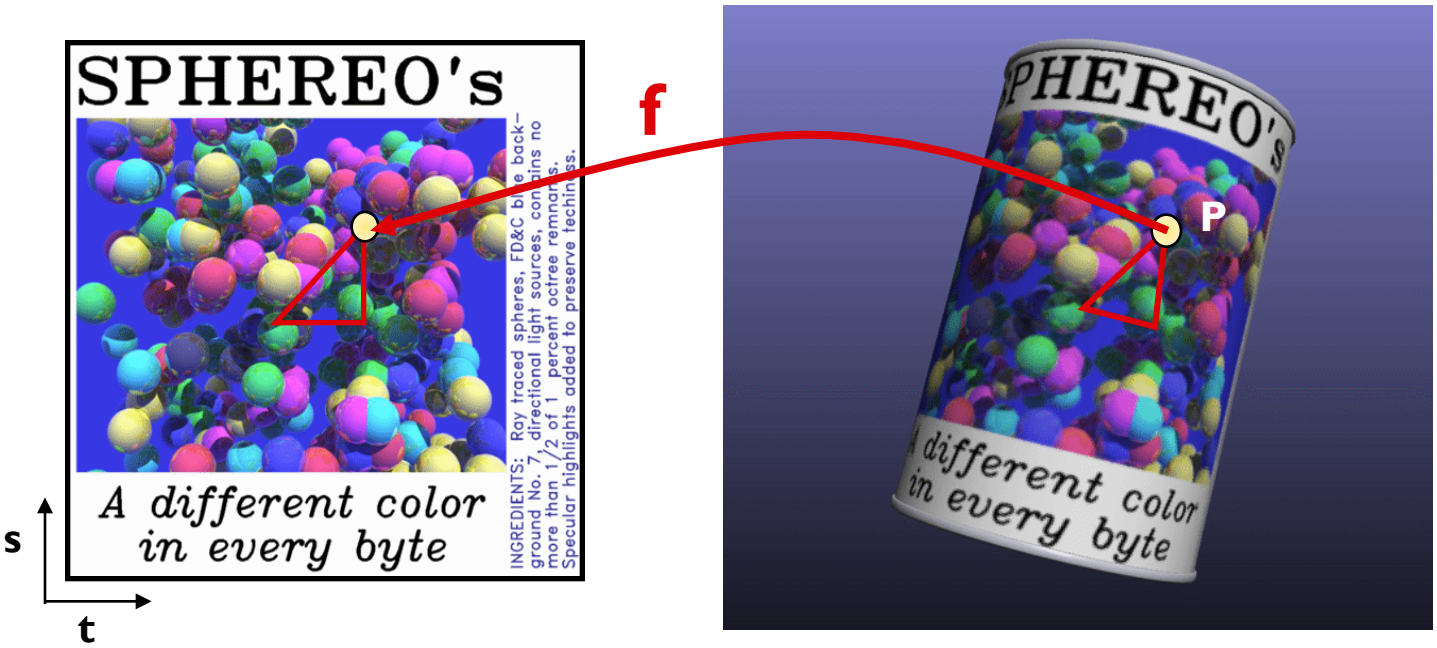
Image : Henrik Wann Jensen - Environment map : Paul Debevec

Les textures 2D

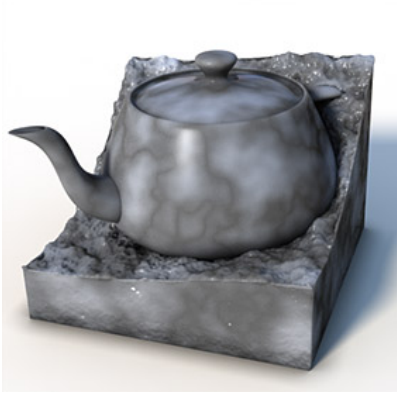
Image **I(s,t)**

Fonction de placage (*mapping*)

f : P(x,y,z) → (s,t) dans [0,1]



Les textures 3D

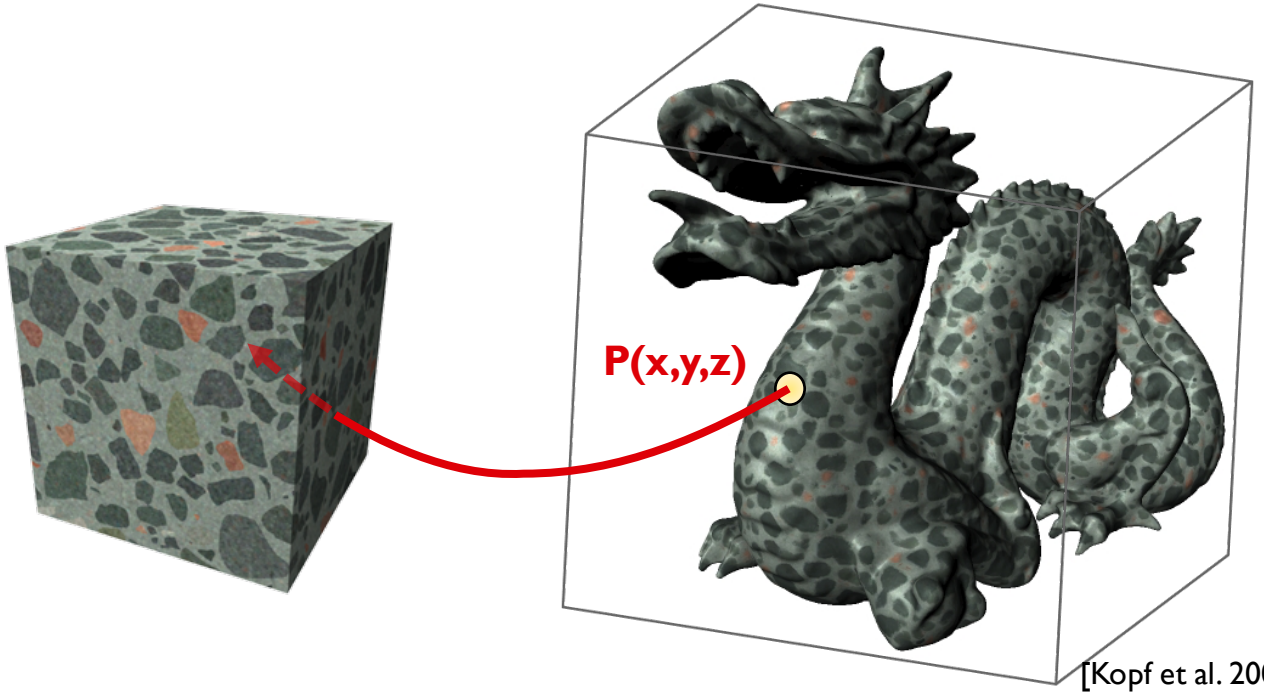


Volume **V(s,t,r)**

Fonction de placage (*mapping*) **triviale**

x = s, y = t, z = r

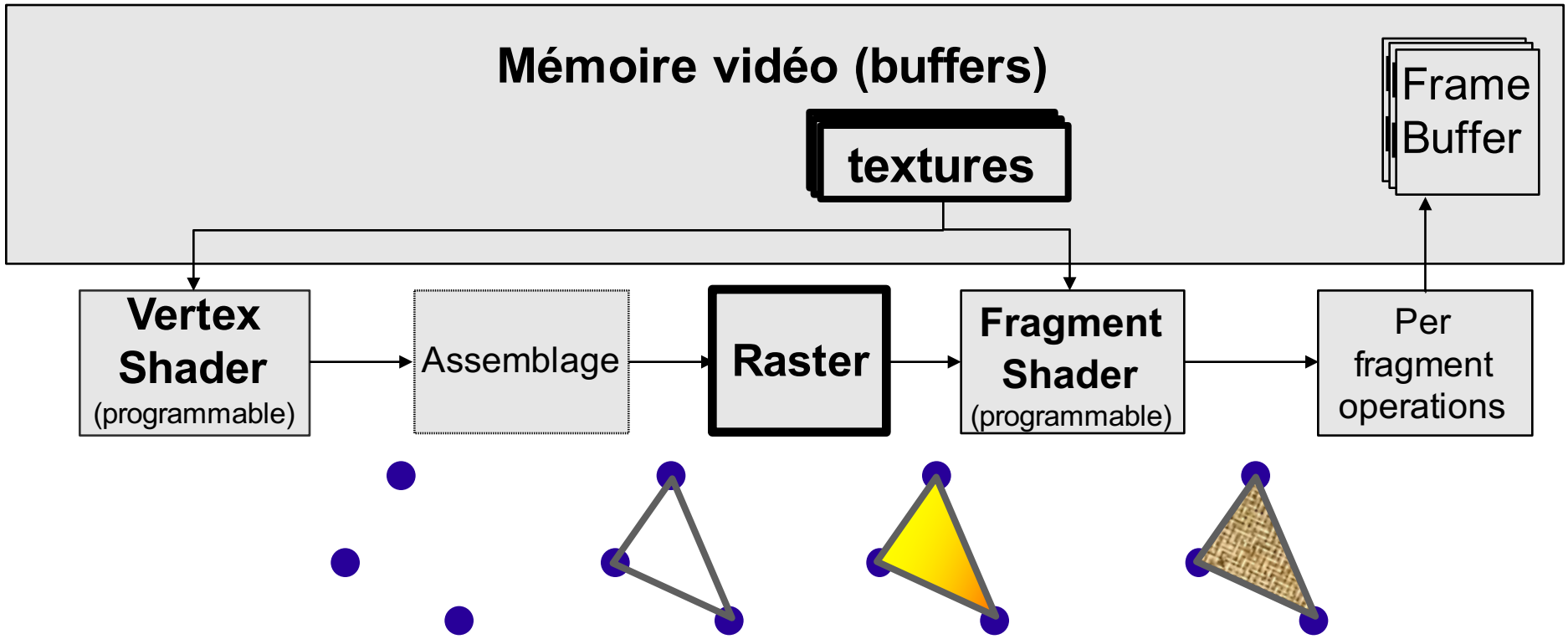
Attention : coût mémoire important



[Kopf et al. 2006]

Image : documentation « modo »

OpenGL



Différents types de base :

- `GL_TEXTURE_1D`
- `GL_TEXTURE_2D`
- `GL_TEXTURE_3D`
- `GL_TEXTURE_2D_ARRAY`
- `GL_CUBE_MAP`

Création, suppression, activation d'un objet de texture :

- `glGenTextures(...)`
- `glDeleteTextures(...)`
- `glBindTexture(...)`

GLSL

Accès aux textures :

```
vec4 texture(sampler, texcoord)
```

Exemple :

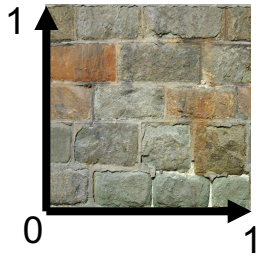
```
// vertex shader
uniform mat2 texcoord_mat;
uniform mat4 mvp;

in vec4 vtx_position;
in vec2 vtx_texcoord;

out vec2 texcoord;

void main(void) {
   gl_Position = mvp *
                 vtx_position;
   texcoord = texcoord_mat *
      vtx_texcoord;
}
```

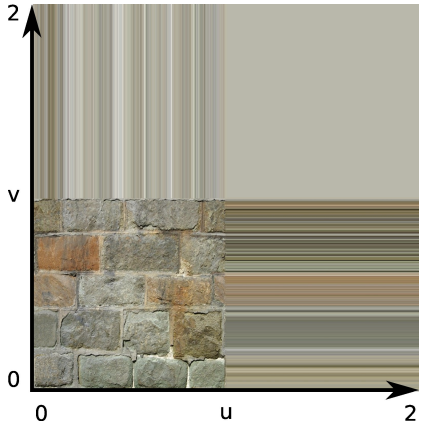
```
// fragment shader
uniform sampler2D image;

in vec2 texcoord;

out vec4 out_color;

void main(void) {
    out_color =
    texture(image,texcoord);
}
```

Contrôle du pavage

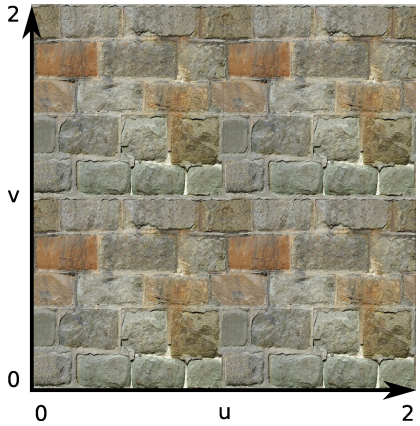
Gestion des coordonnées de texture **hors de [0,1]**

- `GL_CLAMP_TO_EDGE` : s = (s<0 ? 0 : (s>1 ? 1 : s))
- `GL_REPEAT` : utilisation de la partie fractionnaire
- `GL_MIRRORED_REPEAT`

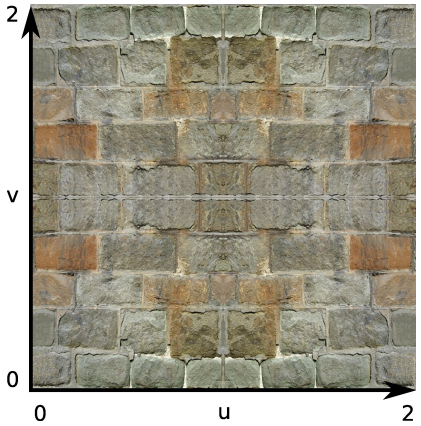




`GL_CLAMP_TO_EDGE`



`GL_REPEAT`



`GL_MIRRORED_REPEAT`

Possibilité d'appliquer une **matrice de transformation**

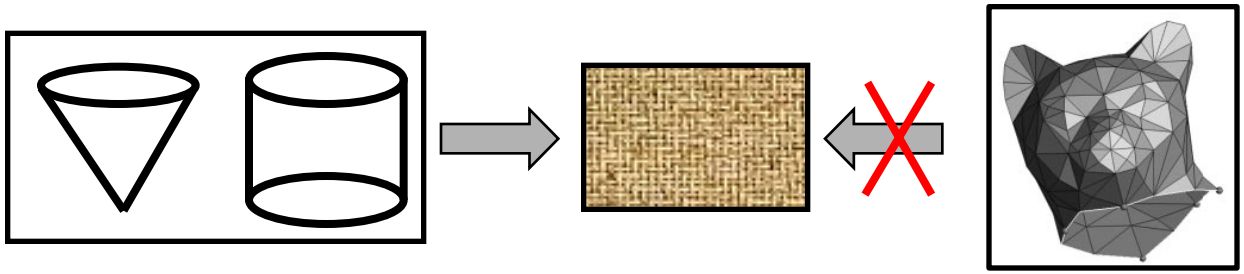
⟹ effet de changement d'échelle, glissement, etc.

Problèmes

1. **Plaquer une texture sans distorsion**
2. Réduire le Crénelage, l'a*liasing*
3. Synthétiser une texture

Fonction de placage

Équivalent à un **dépliage**

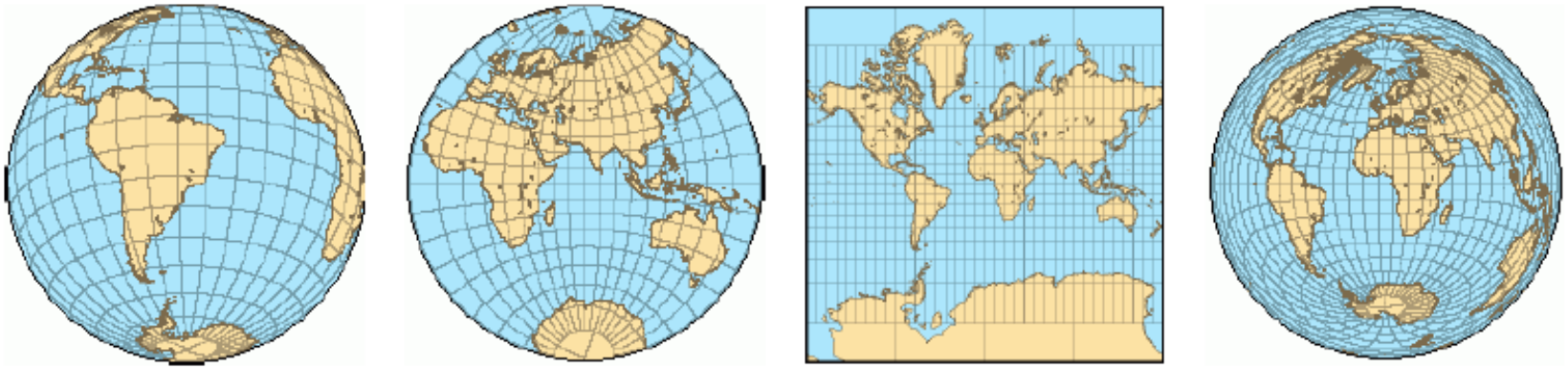


Pas toujours possible : **surface développable**

$\Longrightarrow$ rendre le problème local

Fonction de placage

Problème bien connu en cartographie



⟹ **Distorsion** globale ou locale

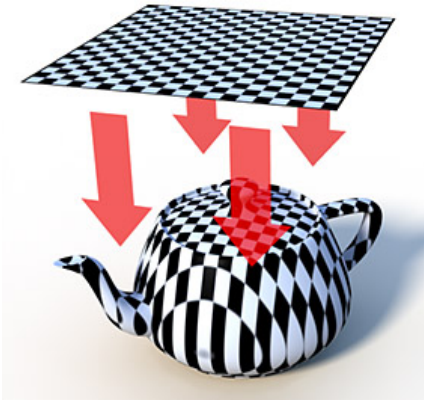
⟹ Choix de conserver les angles, distances, …

Solutions simples

f: (x,y,z) ⟶ [0,1] x [0,1]

Planaire : projection

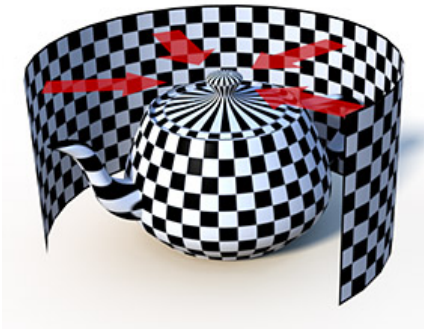
f (x,y,z) = (||x|| , ||y||)



Cylindrique

f (θ,y) = (θ/2π, y)

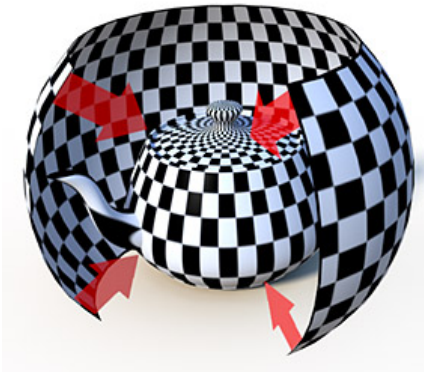
avec θ = atan(z/x)



Sphérique

f (θ,z) = (θ/2π , Φ/π)

avec θ = atan(z/x) et Φ=asin(y)

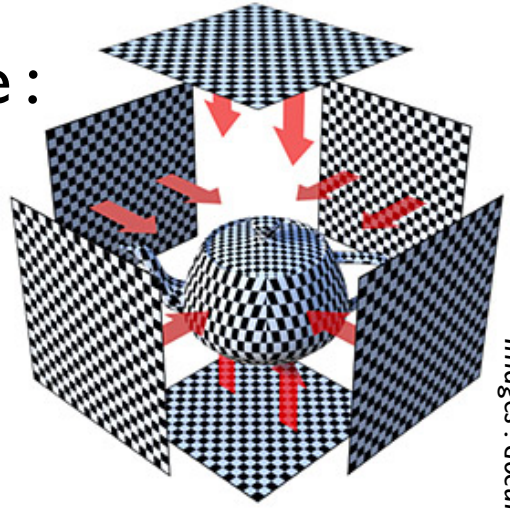


Images : documentation « modo »

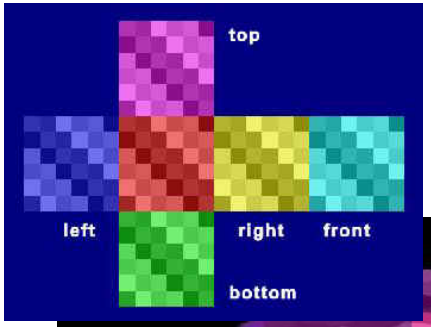
Solutions simples

Passer par un **objet simple** intermédiaire :

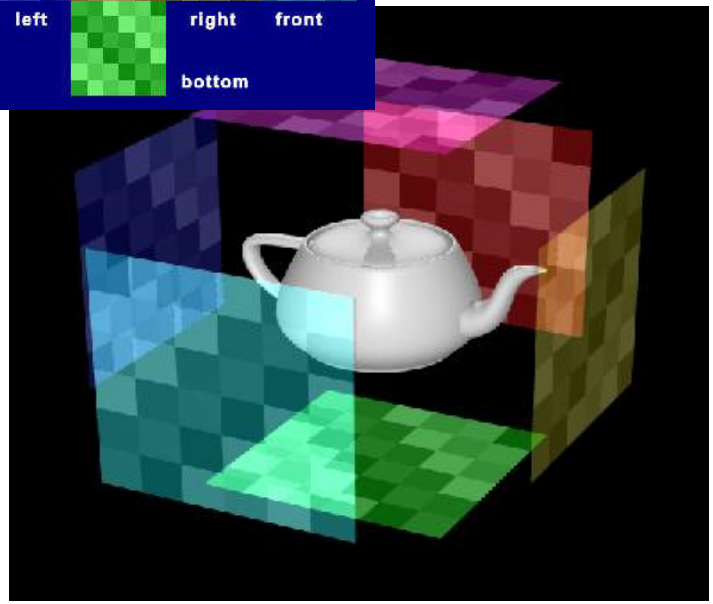
1. Dépliage
2. Projection

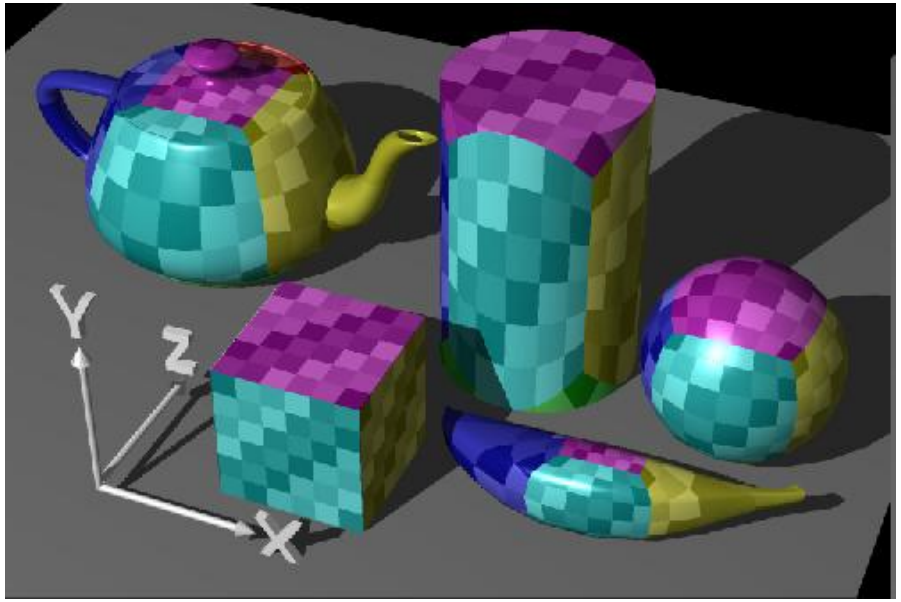


Images : documentation « modo »



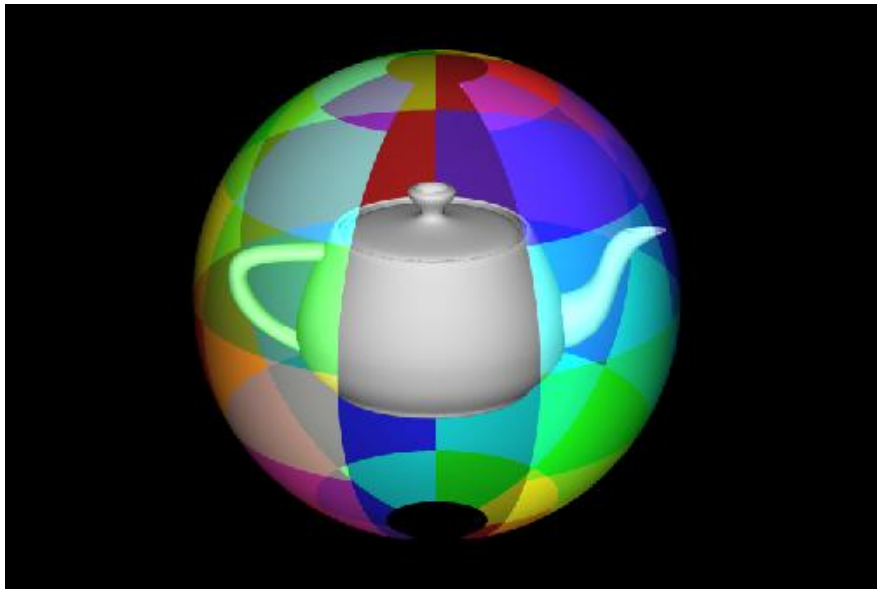
Cube mapping

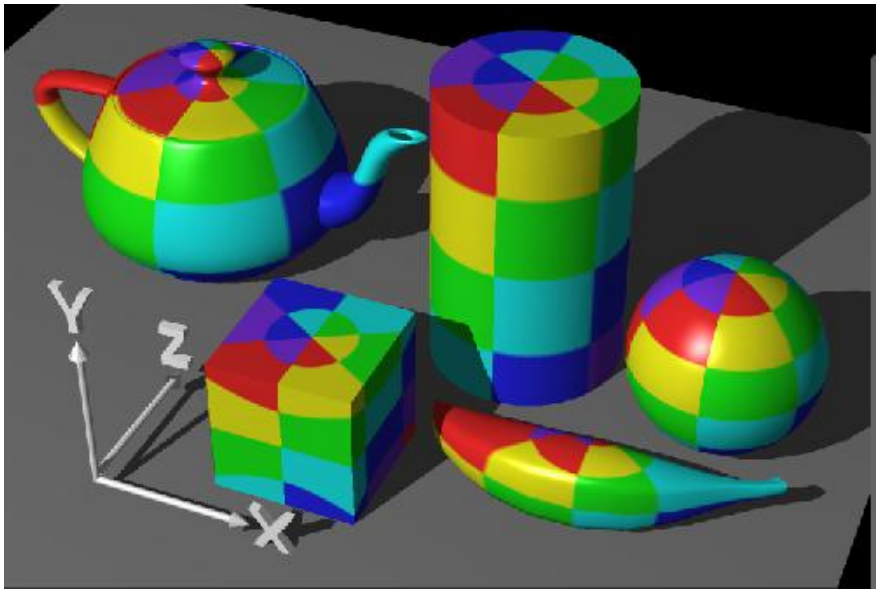




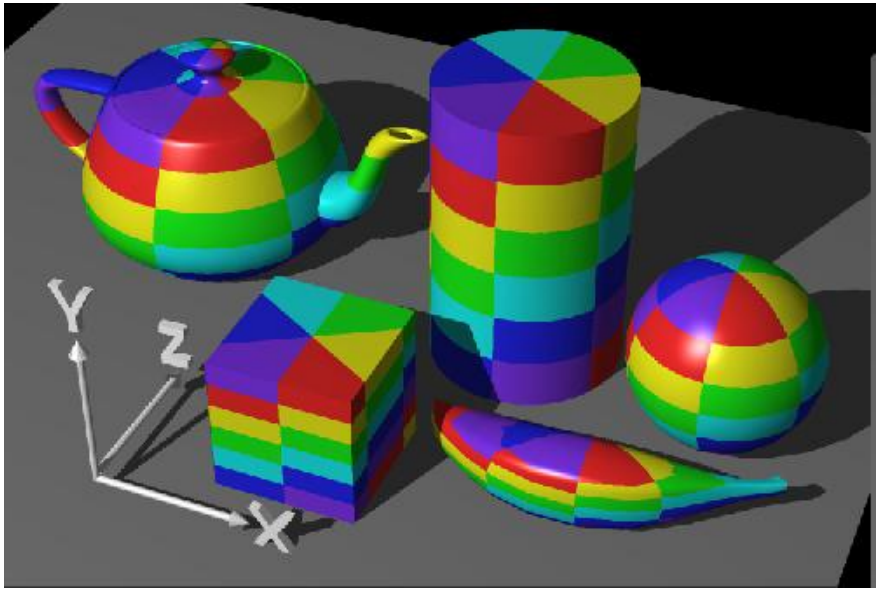
Images : Rosalee Wolfe

Spherical mapping





Cylindrical mapping



Images : Rosalee Wolfe

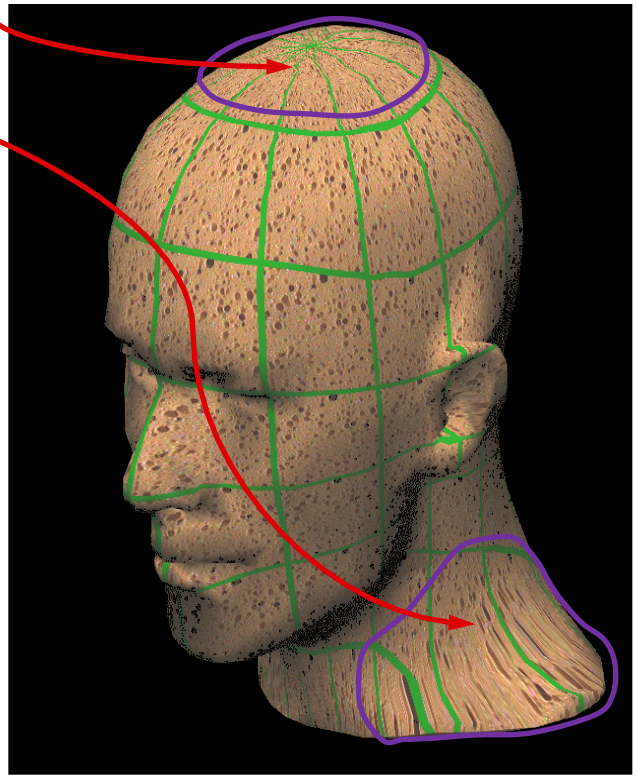
Cas général

Problème d'optimisation de la paramétrisation pour minimiser

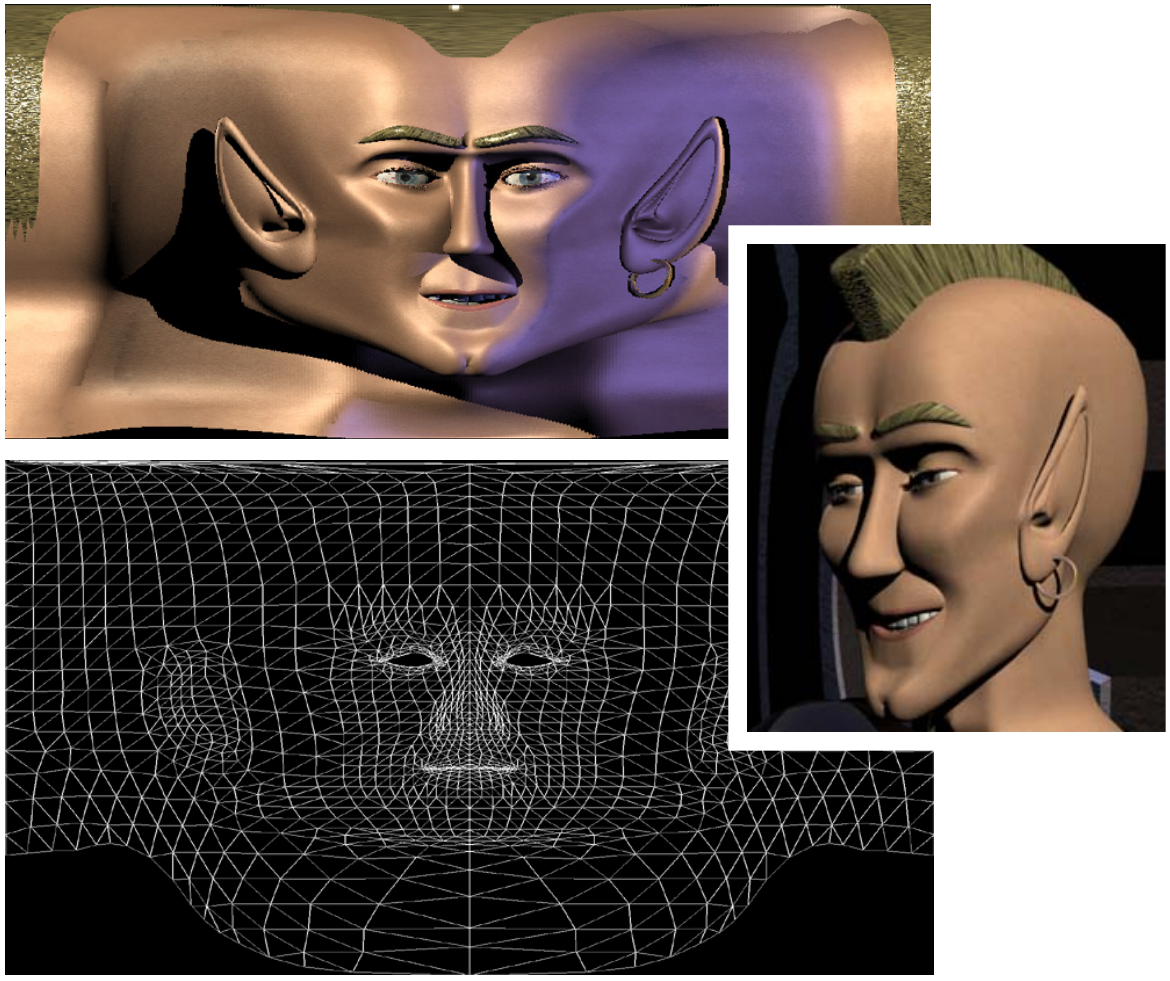
- les **singularités** (pôles)
- les **distorsions**

$\Longrightarrow$ Réalisée à la main par les artistes

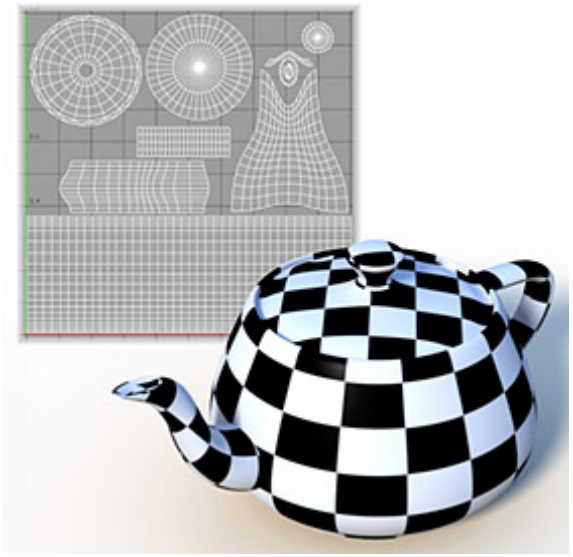
$\Longrightarrow$ Techniques automatiques



Paramétrisation manuelle

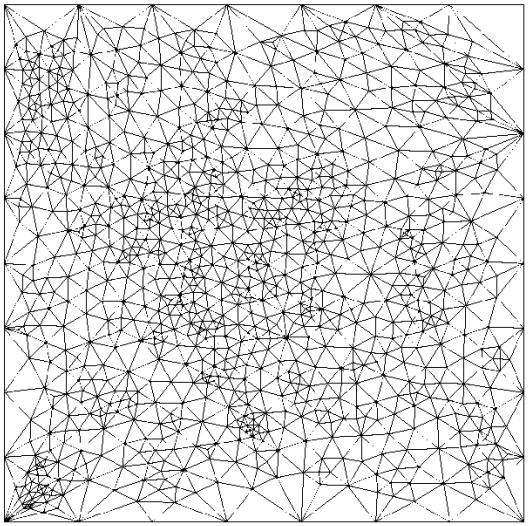


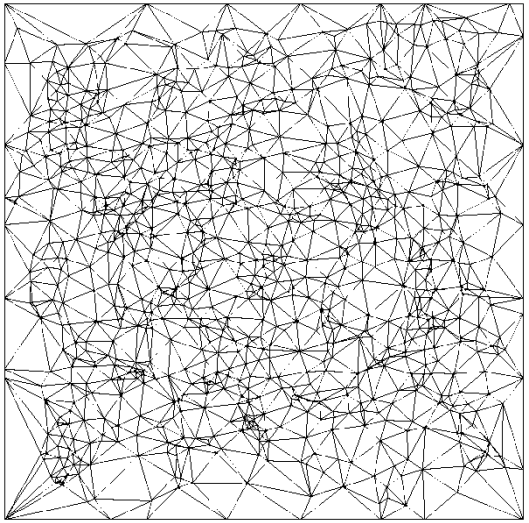
http://www.elfworks.com/Articles/skin-o-matic.html

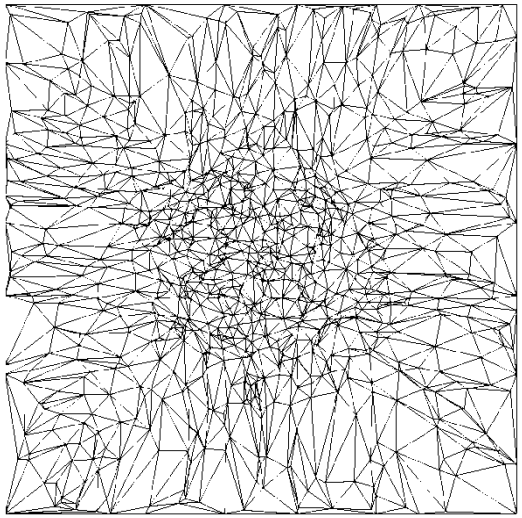


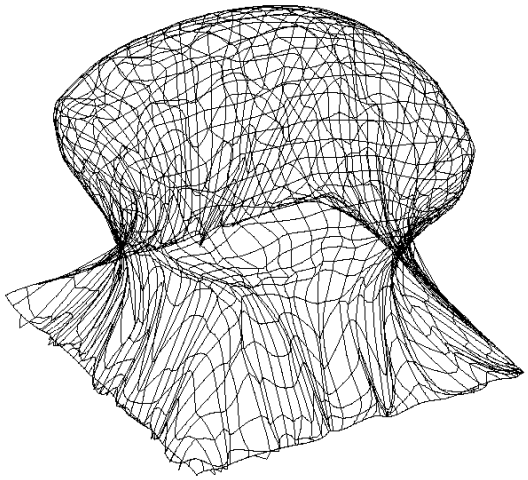
documentation « modo »

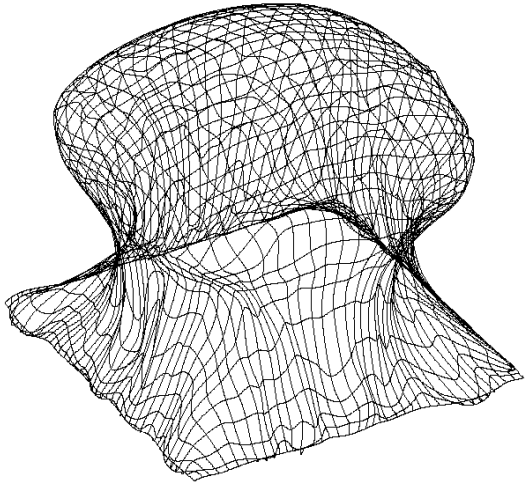
Exemples d'optimisations automatiques

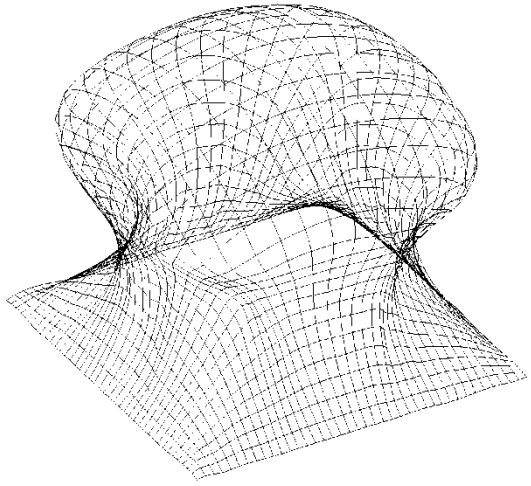












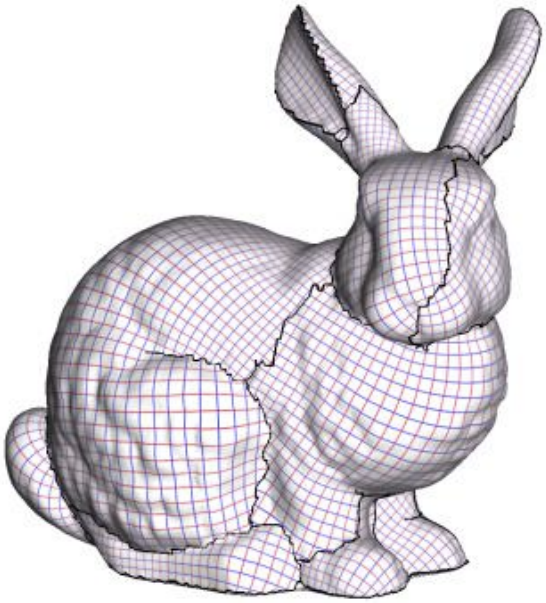
Iso-barycentre

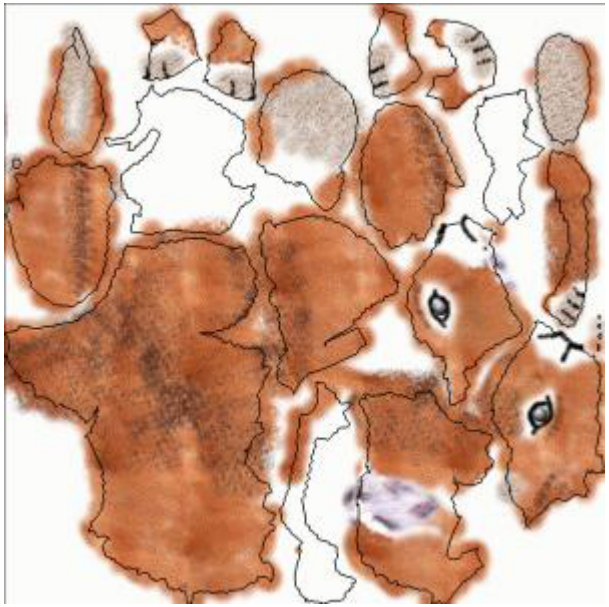
Conservation des distances

Barycentre (3 pts)

Atlas de texture

Décomposition en différents patches





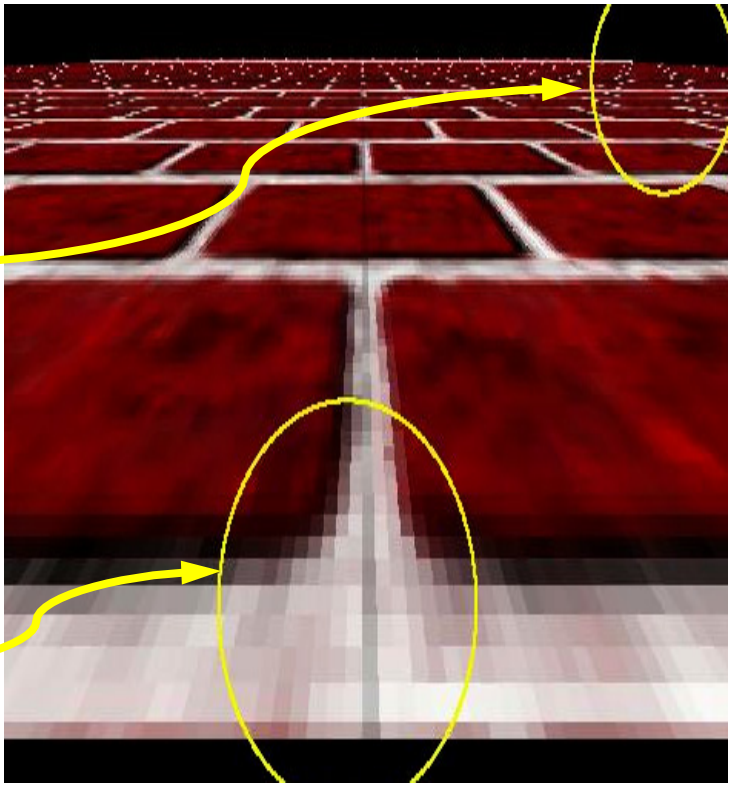
[Lévy et al. 2002]

Problèmes

1. Plaquer une texture sans distorsion
2. **Réduire le Crénelage, l'*aliasing***
3. Synthétiser une texture

Plusieurs couleurs pour un pixels

Pixels visibles

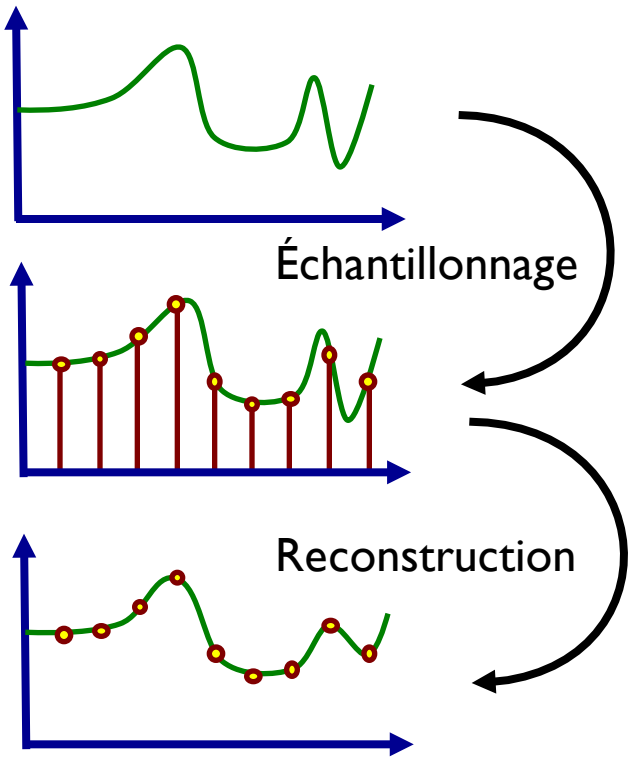


Échantillonnage

La majorité des signaux réels sont **continus**,
mais tout est **discret** en informatique

Échantillonnage et **quantification**
permettent de passer du continu
au discret

La **reconstruction** essaie
de reconstruire un signal continu
à partir des échantillons quantifiés



Exemple

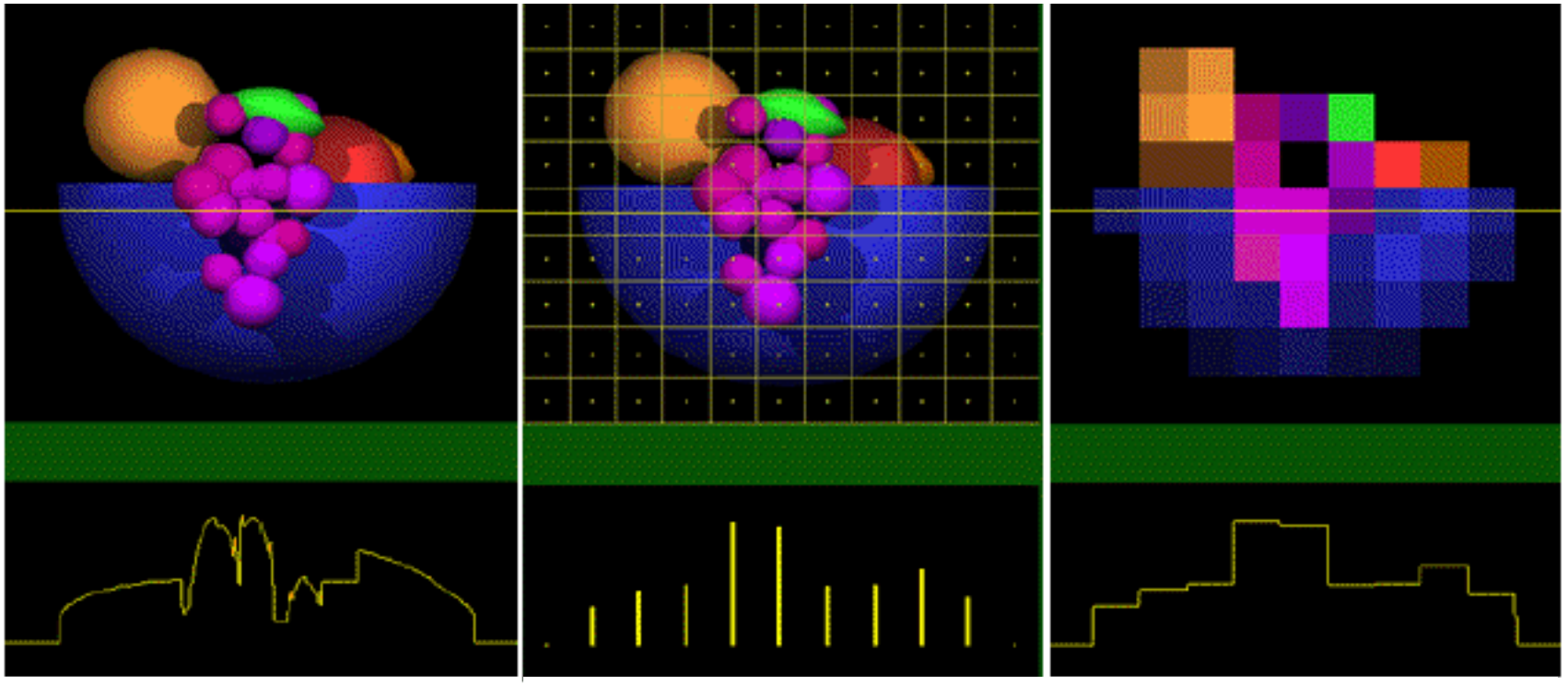
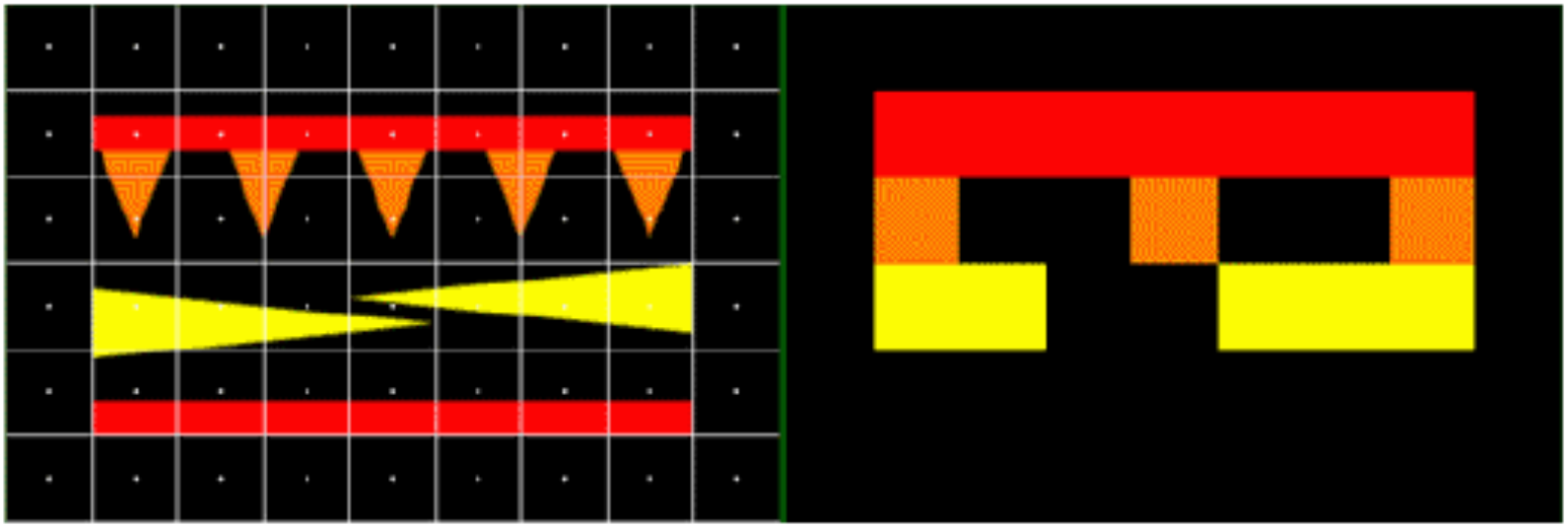


Image d'origine | Échantillons | Reconstruction

Alisasing – perte de détails

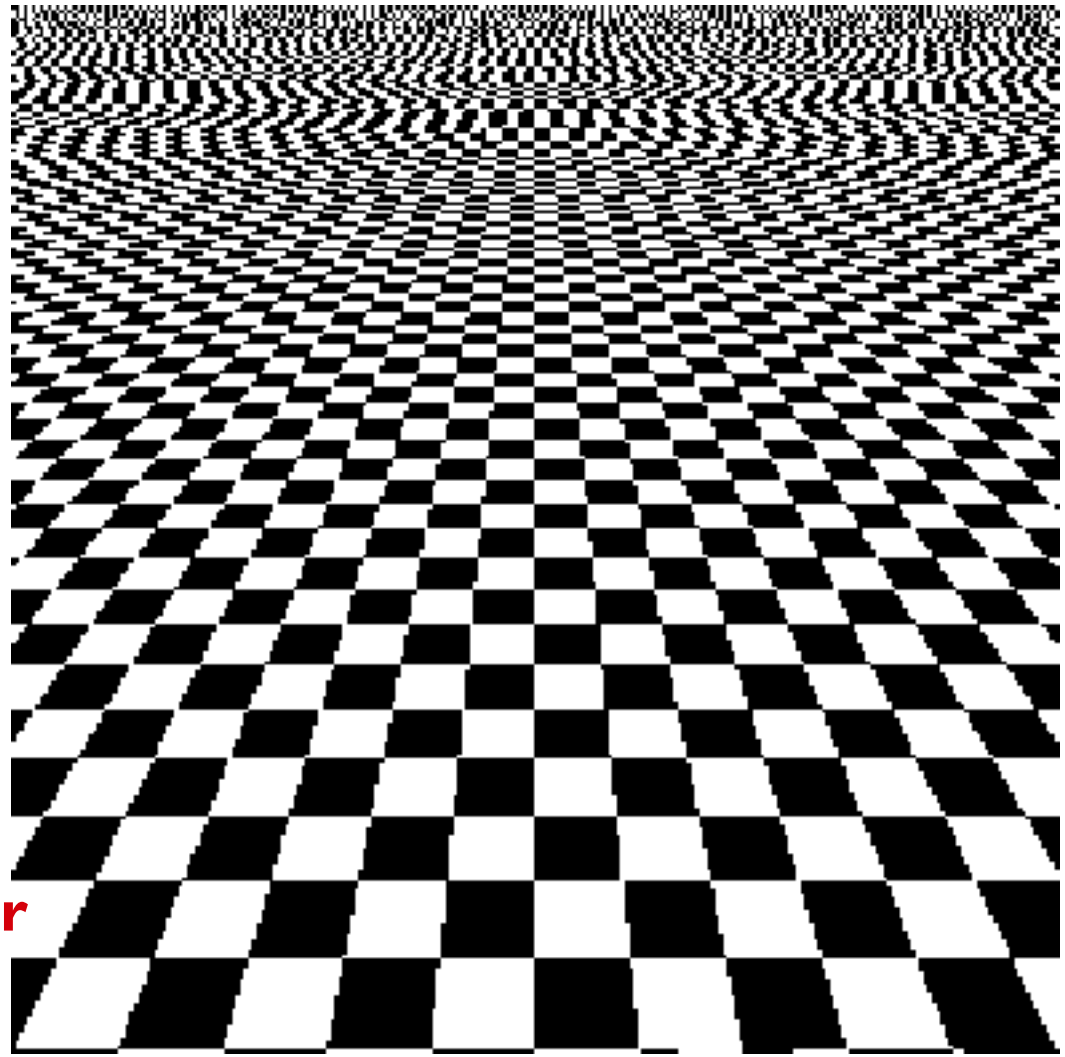


Échantillons

Reconstruction

Alisasing – effet d'escalier / moiré

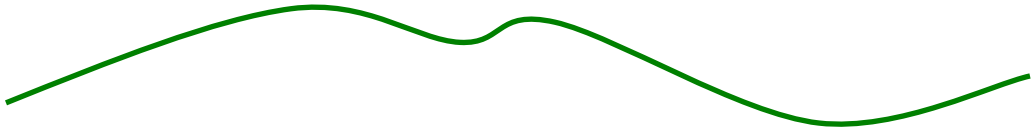
Moiré



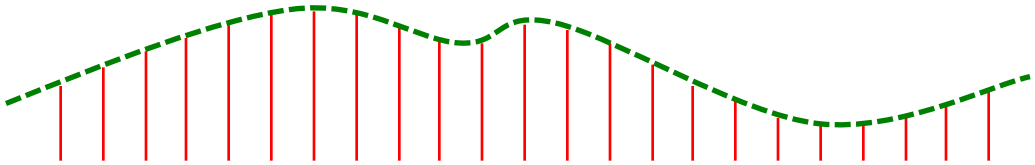
Effet d'escalier

Signal basse fréquence

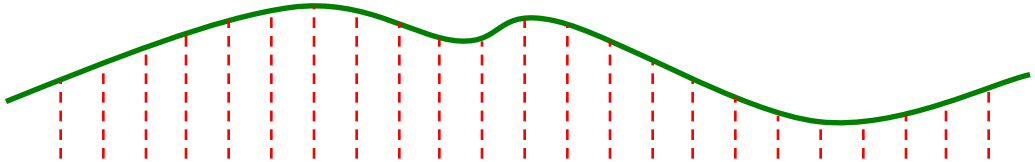
Signal original



Échantillonnage
haute fréquence

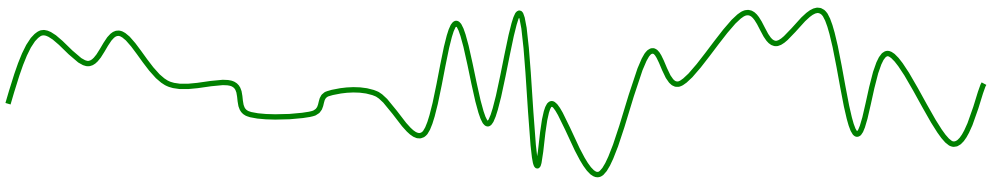


Signal reconstruit

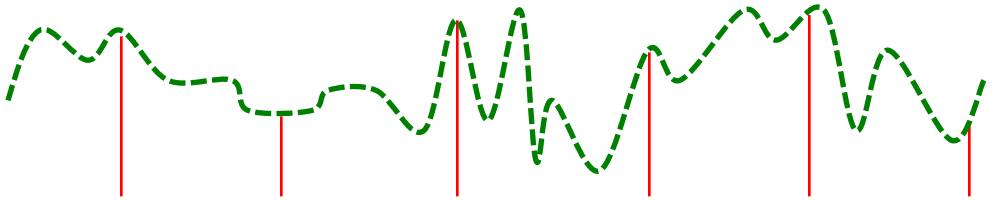


Signal haute fréquence

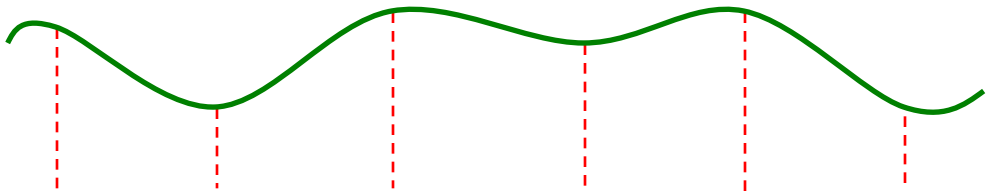
Signal original



Échantillonnage
basse fréquence

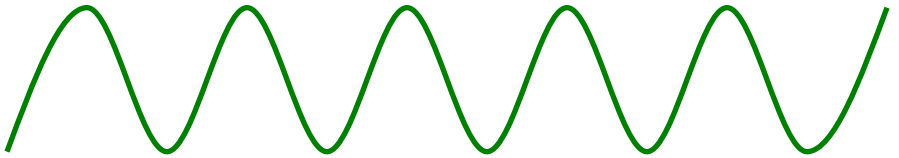


Signal reconstruit

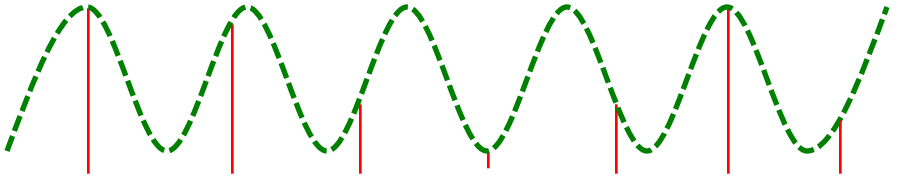


Signaux périodiques

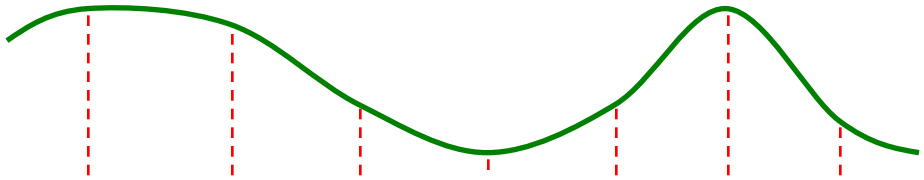
Signal original



Échantillonnage
basse fréquence



Signal reconstruit



Théorème de Nyquist-Shannon

En théorie, pour reconstruire un signal de fréquence *f*, il faut l'échantillonner à la fréquence *2f*

$\Longrightarrow$ **fréquence de Nyquist**

Valide pour les signaux à support fréquentiel fini.

Problème :

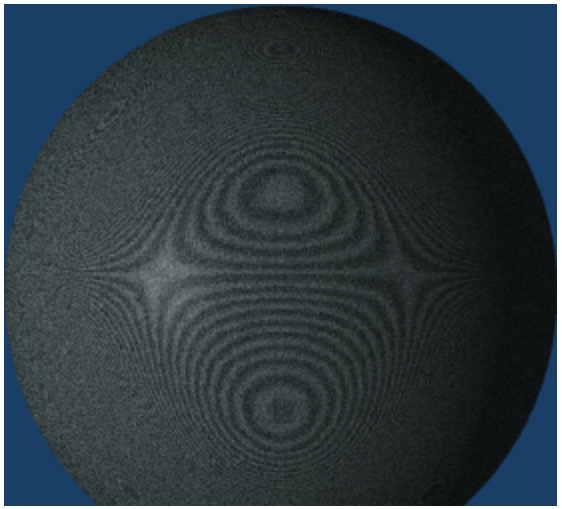
la plupart des images comportent des **discontinuités** (lignes, ombres dures, textures, taches spéculaires, etc.)

$\Longrightarrow$ **fréquences infinies !**

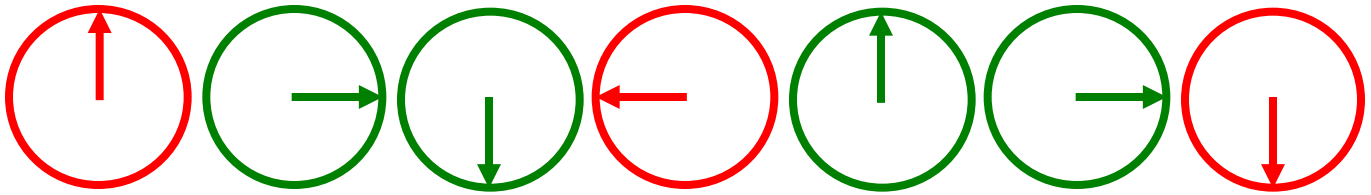
Aliasing

Spatial

- Effets d'escalier, Moiré, etc.



Temporel



Antialiasing

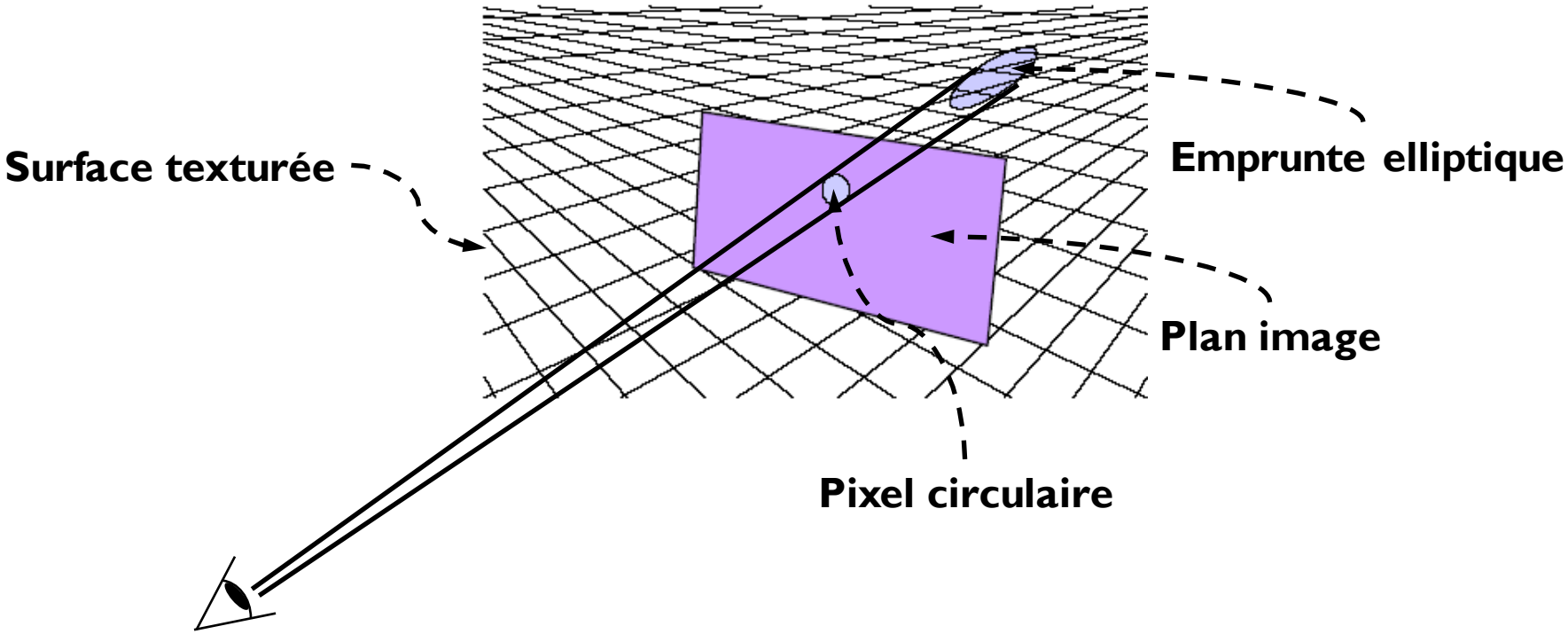
Antialiasing

Théorie :

Projection du pixel sur la texture et **intégration de la couleur sur l'aire du pixel projeté**

$\Longrightarrow$ Très couteux

En pratique : approximations !

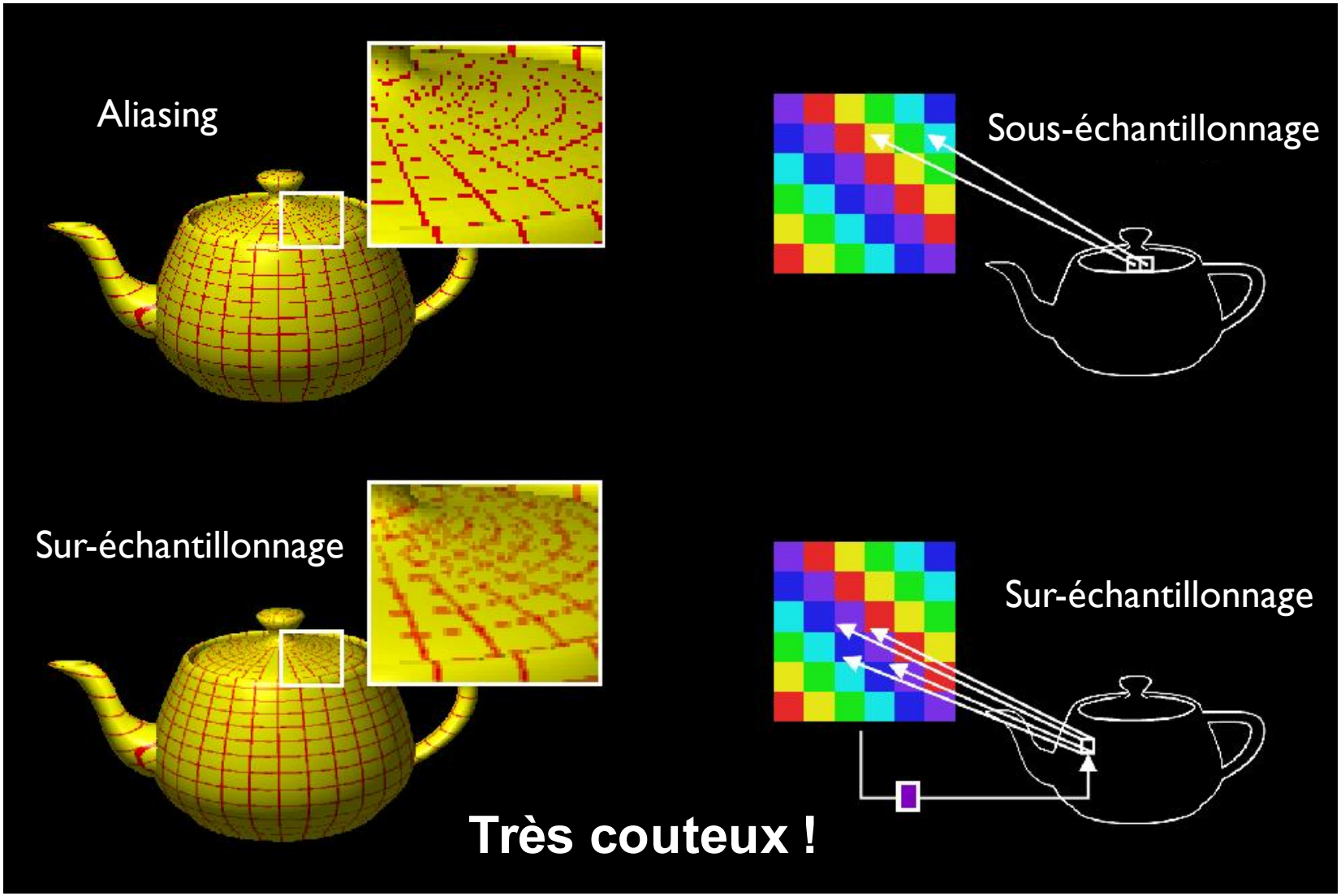


Antialiasing

Échantillonner à plus haute fréquence

- Pas toujours possible (coût, contraintes matériels)
- Signaux réels ont des fréquences infinis

Sur-échantillonnage

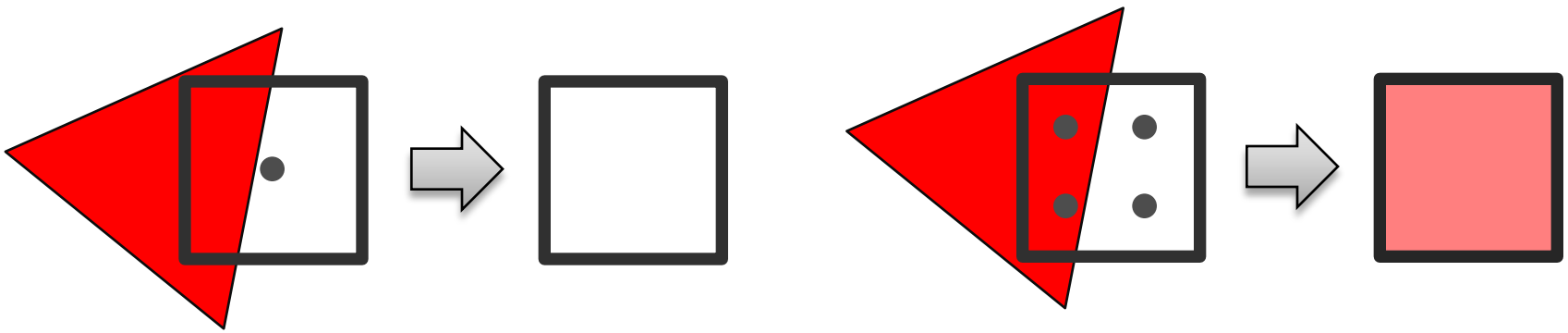


Sur-échantillonnage

Stratégie générale

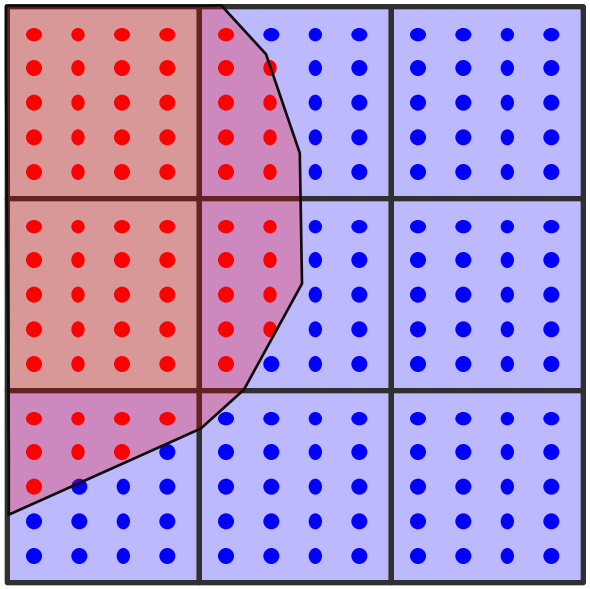
- Utiliser plusieurs échantillons par pixel
- Moyenner ces échantillons pour obtenir la couleur finale

$$\text{couleur du pixel } (x, y) = \sum_{i=1}^{n} w_i c(i, x, y)$$

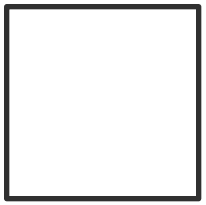


Sur-échantillonnage

Full Screen Anti-Aliasing (FSAA)

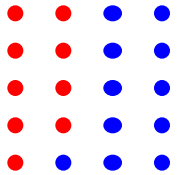


1. **Sur-échantillonner** chaque pixel



1. **Moyenner** les couleurs de chaque échantillon

$$w_i = \frac{1}{n}$$



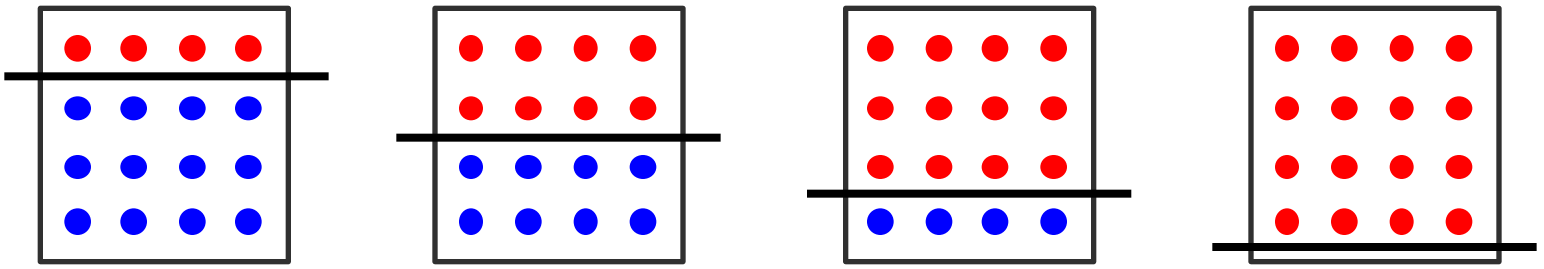
⇔ **à générer l'image à une plus grande résolution puis à réduire sa taille en moyennant**

Motif d'échantillonnage

Grille régulière

- Efficace pour les diagonales
- Pas efficace pour les lignes proches de l'horizontal / vertical

⟹ **avec 4x4 échantillons, seulement 4 tons de rouge !**

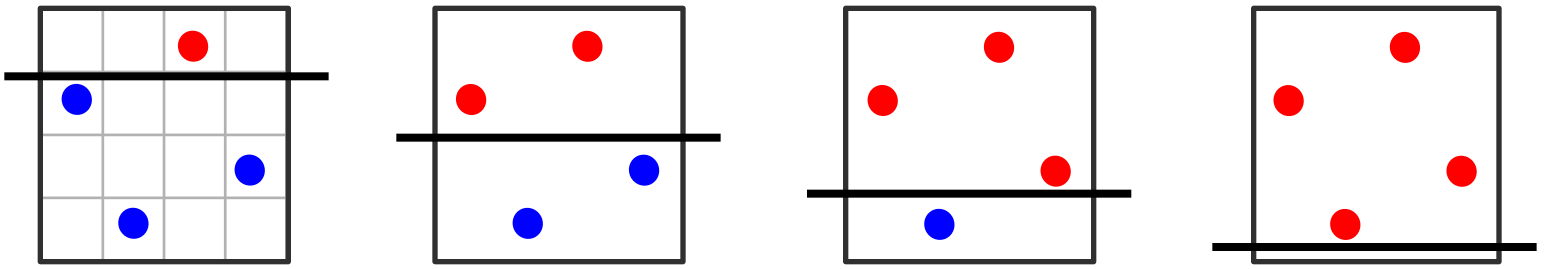


Motif d'échantillonnage

Rotated Grid Super sampling (RGSS)

- Rotation d'une grille régulière
- Diagonales moins efficaces, mais mieux en moyenne

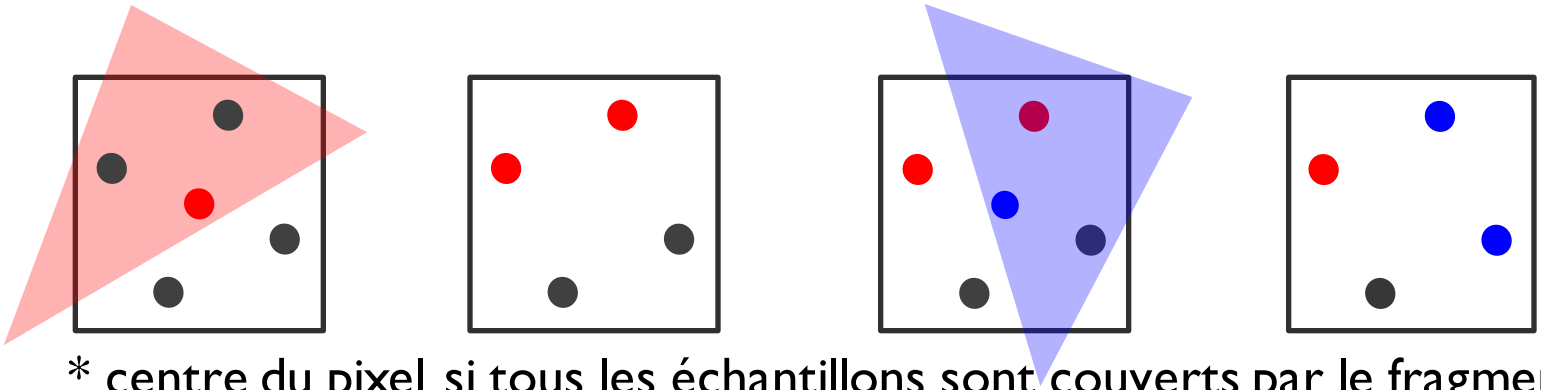
$\Longrightarrow$ **4 tons de rouge avec seulement 4 échantillons !**



Multisample AA (MSAA)

Calculer la couleur pour chaque échantillon **couteux**

- Calculer la **couleur pour un seul échantillon***,
  mais calculer la **visibilité de tous les échantillons**
- Écrire la **même couleur** pour tous les échantillons visibles



* centre du pixel si tous les échantillons sont couverts par le fragment, sinon peut être déplacé (*centroid interpolation*)

En OpenGL : `glEnable( GL_MULTISAMPLE )`

http://www.opengl.org/wiki/Multisampling

Antialiasing

Échantillonner à plus haute fréquence

- Pas toujours possible (coût, contraintes matériels)
- Signaux réels ont des fréquences infinis

Pré-filtrage pour limiter les hautes fréquences

- Filtre passe-bas spatial (mip-map) / flou cinétique
- Compromis entre flou et aliasing

Pré-filtrage

Supprimer les hautes fréquences qui crées les artefacts lors de la minification

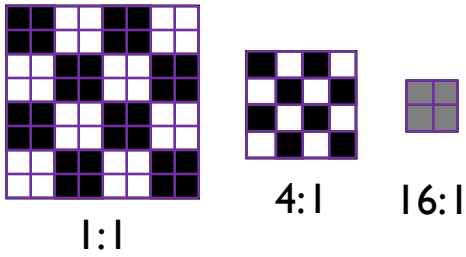
Intégration (convolution) **variant spatialement** en chaque pixel

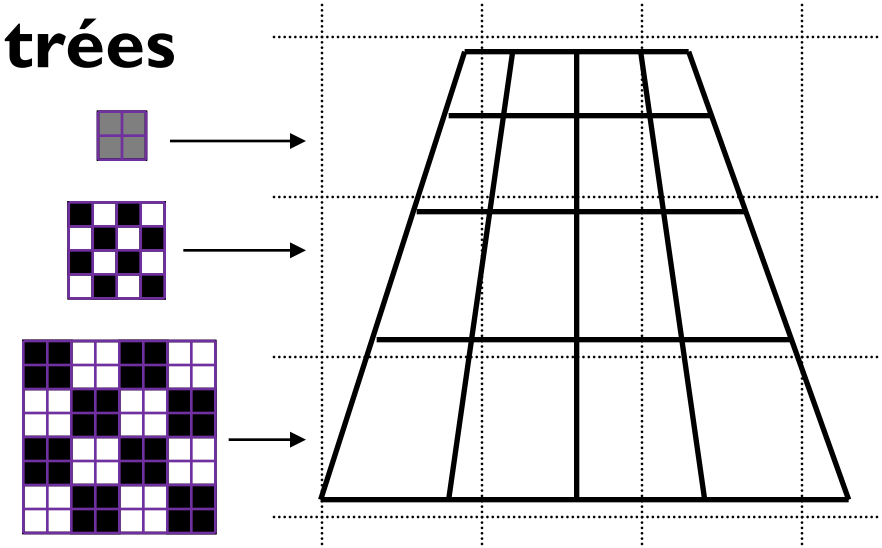
$\Longrightarrow$ **trop couteux**

$\Longrightarrow$ en pré-calculer une **approximation**

MIP Mapping

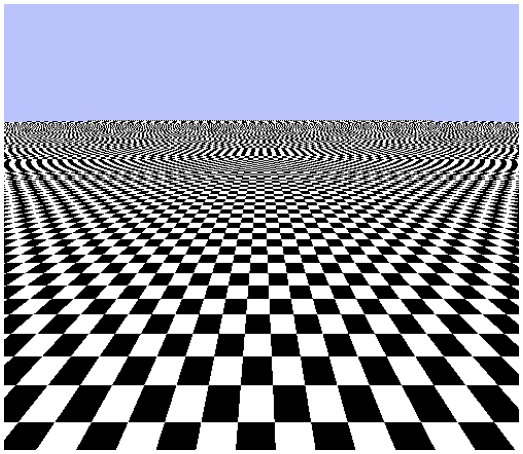
Pyramide d'images pré-filtrées





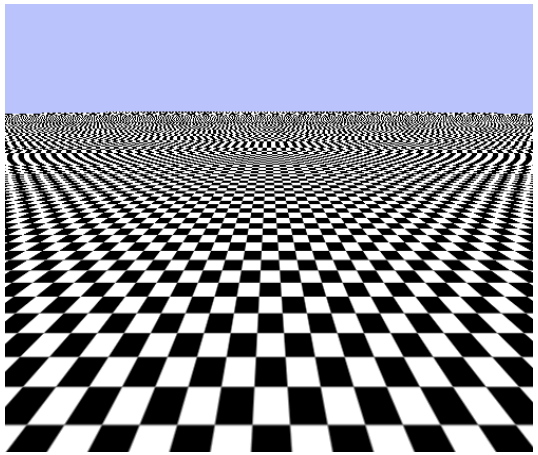
Génération automatique

`GL_GENERATE_MIPMAP, glGenerateMipmap()`



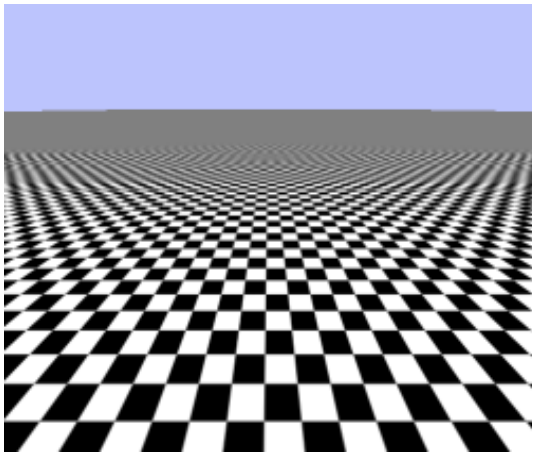
Plus proche voisin

`GL_NEAREST`



Filtrage linéaire

`GL_LINEAR`



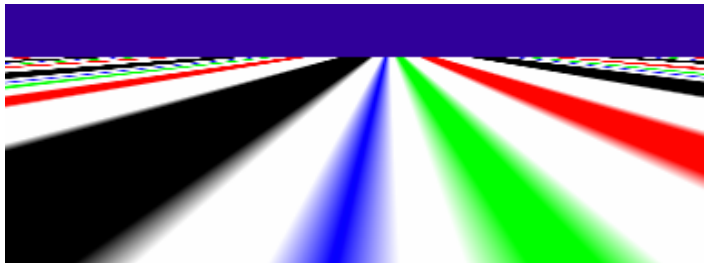
MIP mapping

`GL_LINEAR_MIPMAP_LINEAR`

MIP Mapping

Problème : ne tient pas compte de l'**anisotropie**

L'approximation carrée de la *MIP-map* devient mauvaise…

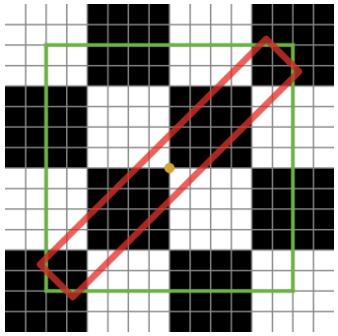


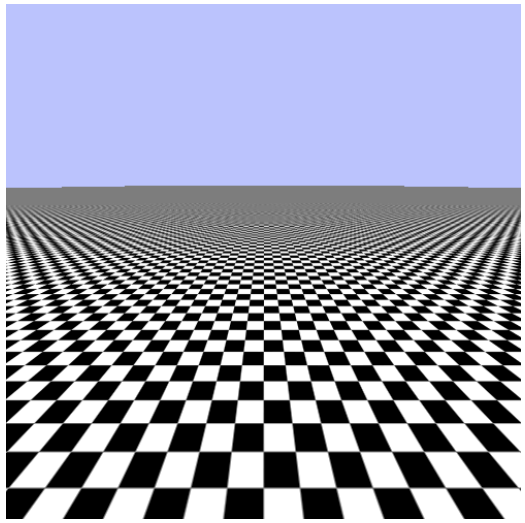
Plus proche voisin

MIP Mapping

Solutions (plus coûteuses)

- Filtrage anisotrope
  matériel, mais « brute-force »
- *Elliptical weighted average*
  calcul de emprunte elliptique du pixel





Filtrage anisotrope

Antialiasing

Échantillonner à plus haute fréquence

- Pas toujours possible (coût, contraintes matériels)
- Signaux réels ont des fréquences infinis

Pré-filtrage pour limiter les hautes fréquences

- Filtre passe-bas spatial (mip-map) / flou cinétique
- Compromis entre flou et aliasing

Filtrage en post-traitement

- En général incorrect (en particulier temporellement)
- Utilisable dans des cas particuliers (effets d'escalier)

Filtrage en post-traitement

Cours Siggraph 2011 : http://www.iryoku.com/aacourse/

Filtrage en post-traitement

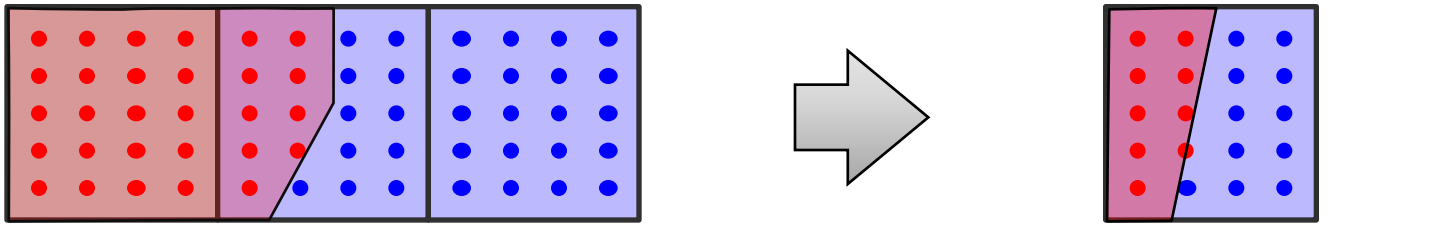
Filtrage en post-traitement

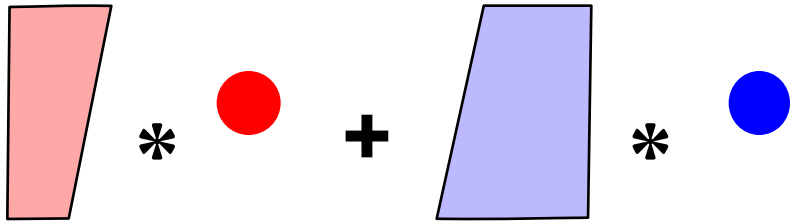
1. **Détecter les silhouettes**
2. **Filtrer les couleurs près des silhouettes**

$\Longrightarrow$ **MorphoLogical AntiAliasing (MLAA)**

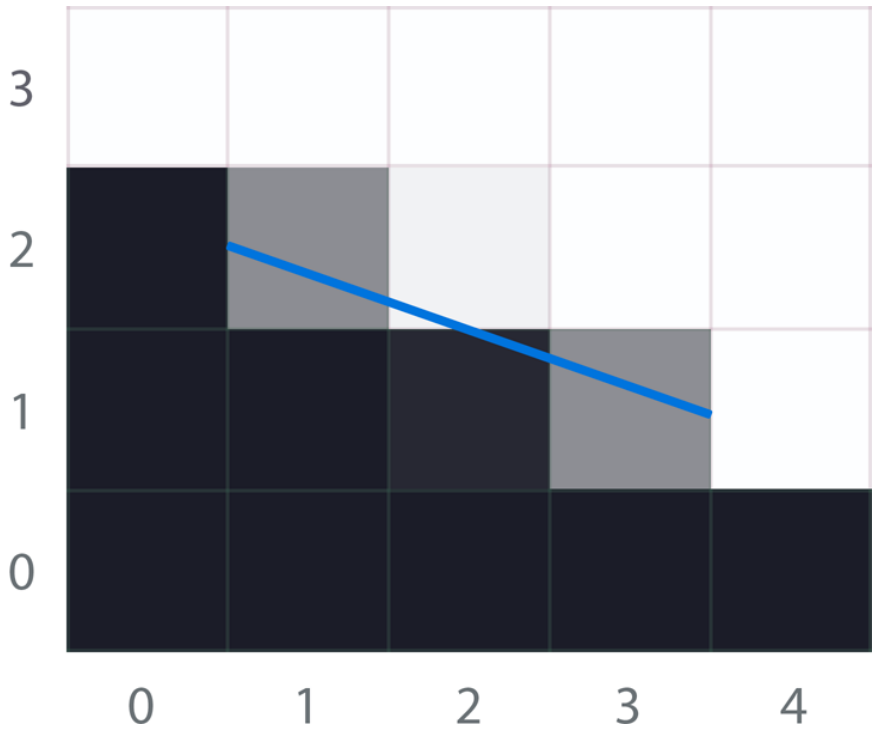
Pourquoi ça devrait marcher ?

Pour les pixels avec 2 couleurs distinctes, l'intégrale peut être approximée par le calcul de l'aire :



pixel du milieu = 

MLAA [Reshetov 2009]

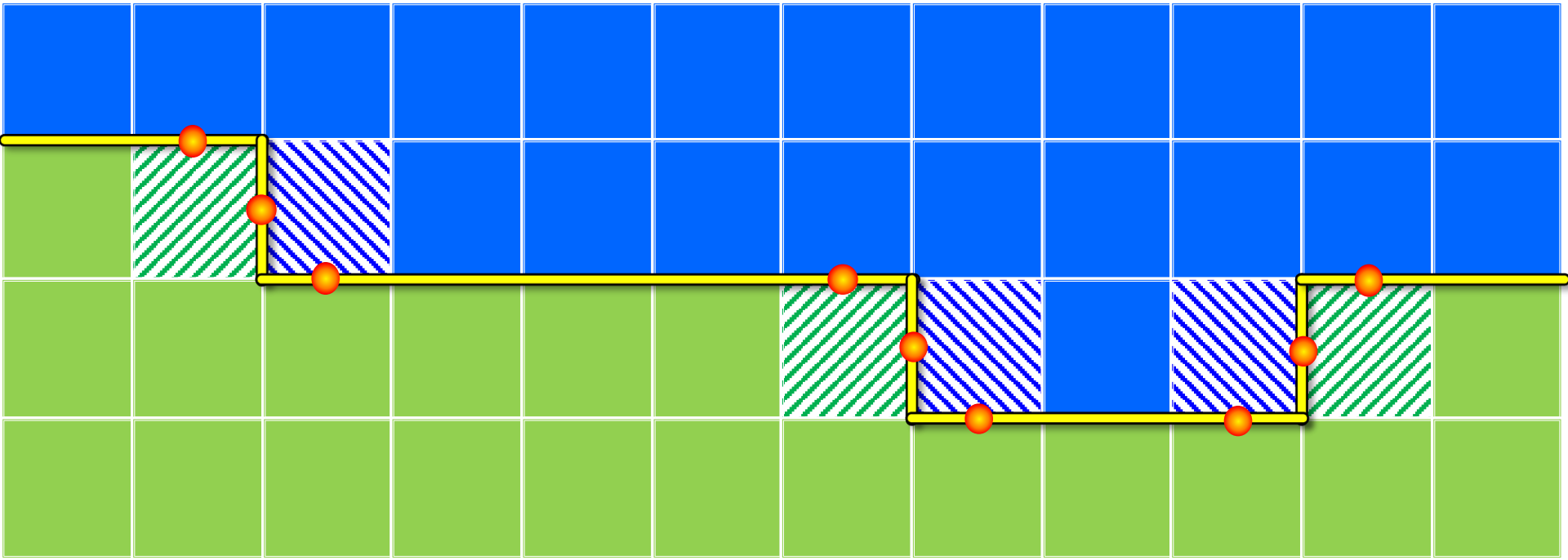


MLAA

MLAA – règle 1/2

Silhouettes débutent / s'achèvent à un « coin » :

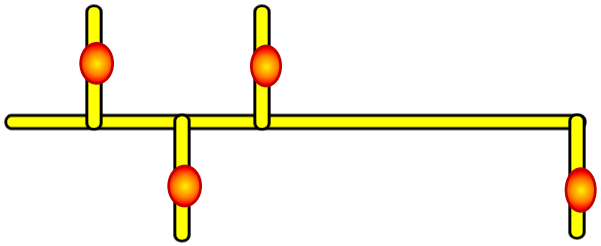
discontinuité verticale et horizontale



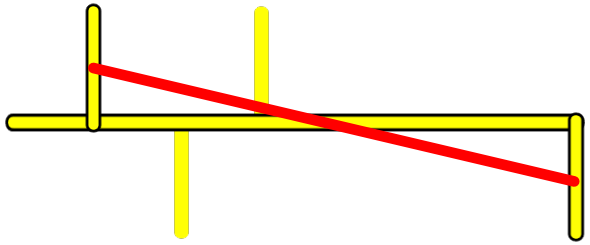
MLAA – règle 2/2

Pour chaque ligne de séparation :

Considérer toutes les lignes orthogonales adjacentes

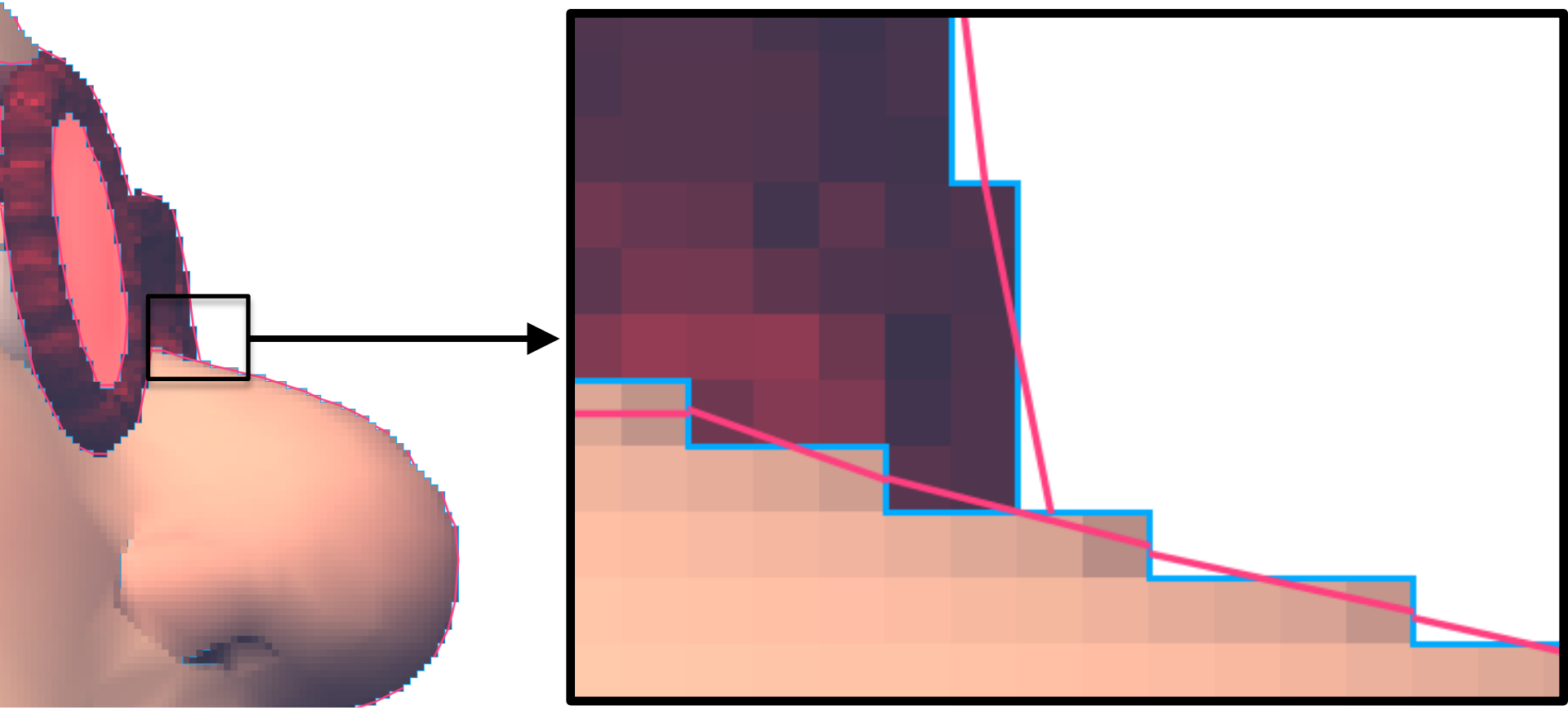


Choisir le plus long segment

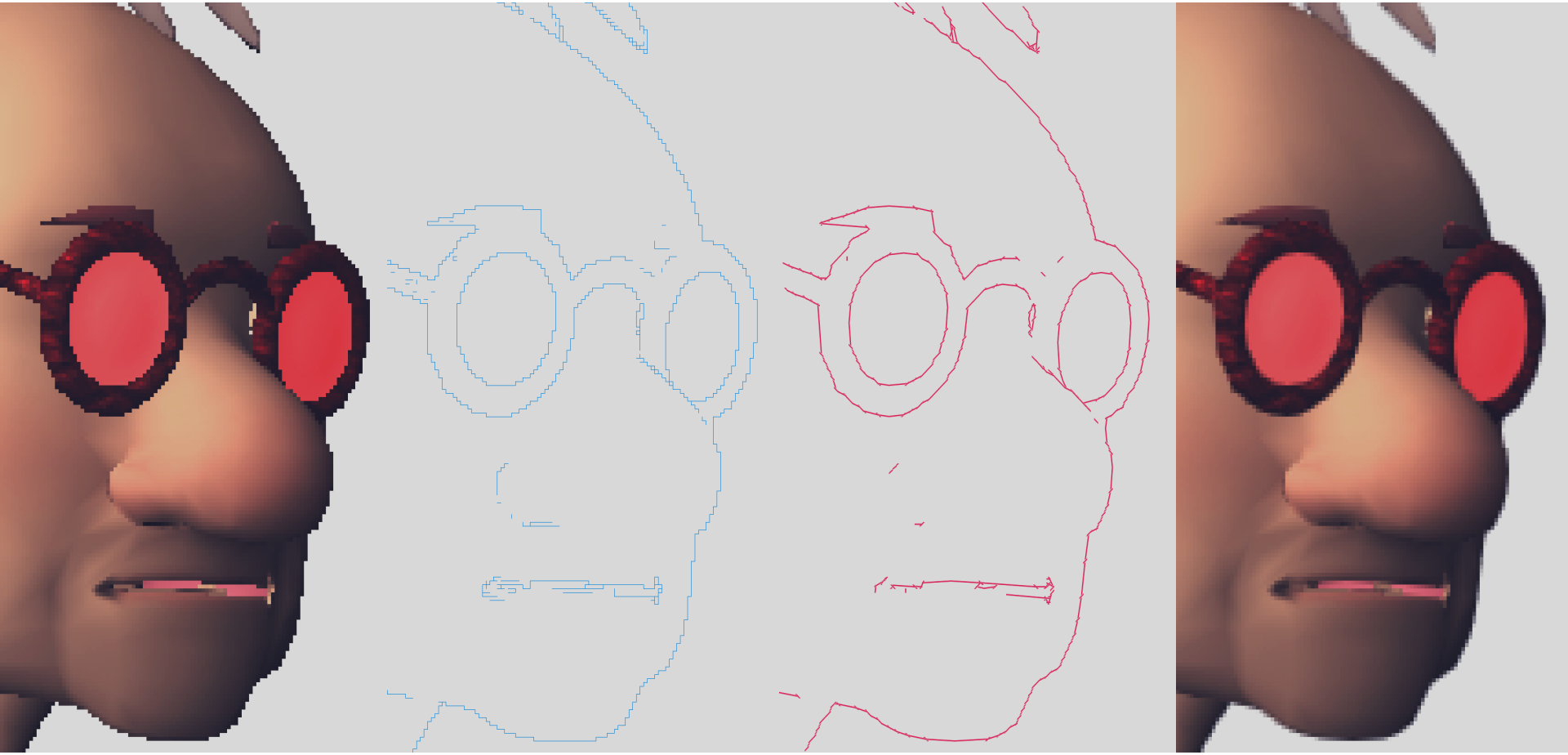


MLAA – règle 2/2

$\Longrightarrow$ gérer correctement les intersections d'objets

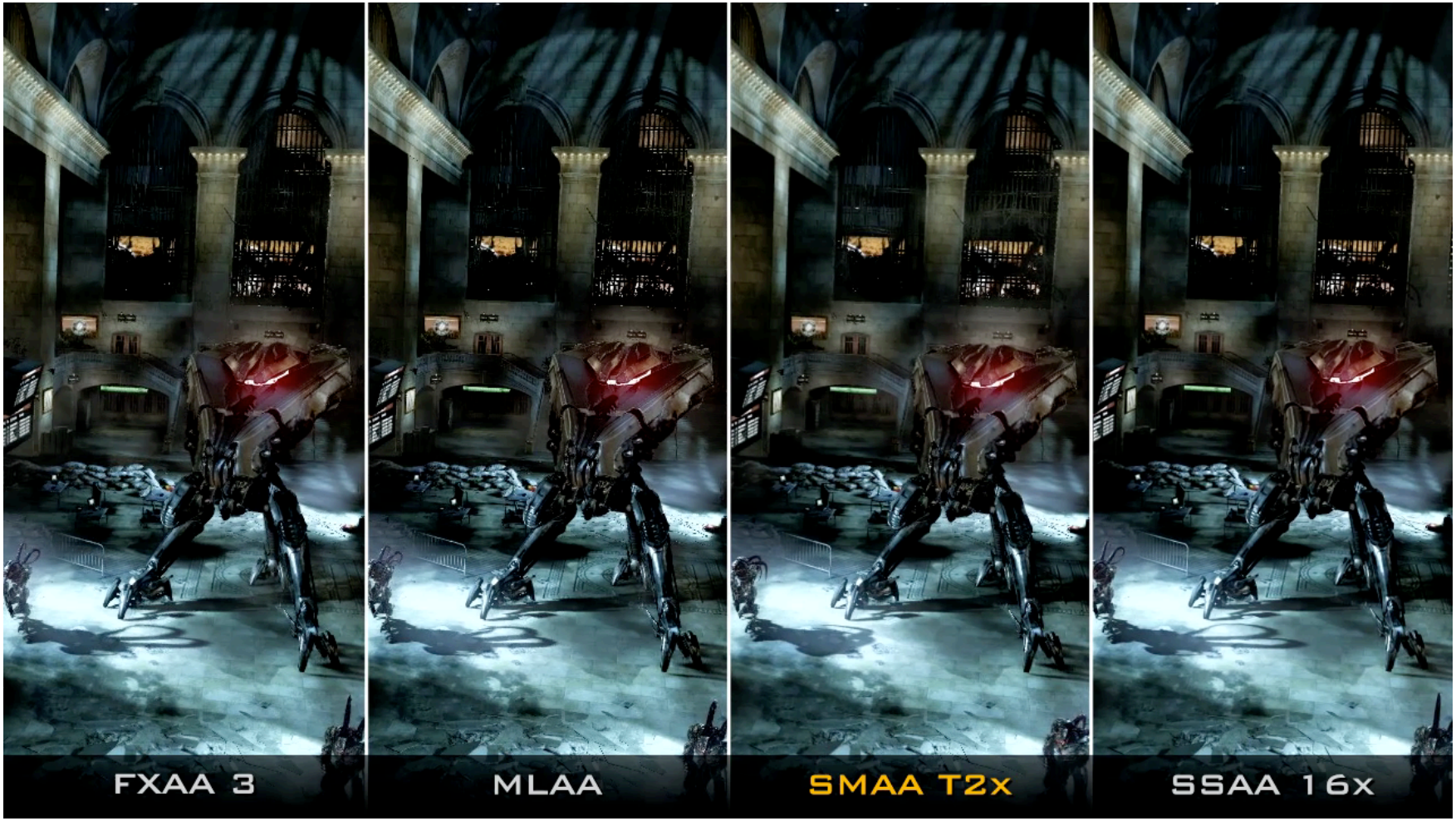


MLAA



SMAA [Jimenez et al. 2011]

Subpixel Morphological Antialiasing



Problèmes

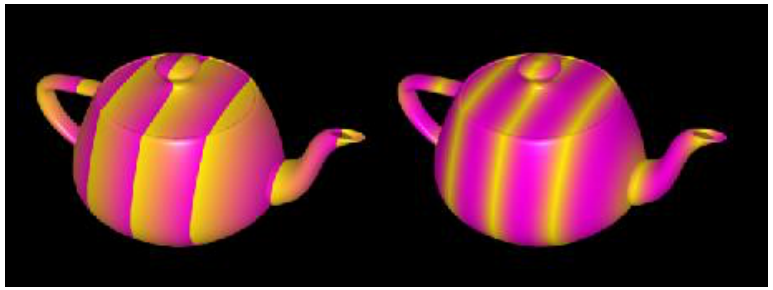
- Plaquer une texture sans distorsion
- Réduire le Crénelage, l'aliasing
- **Synthétiser** une texture

par Hugo Beyer, crées avec *Substance Allegorithmic*

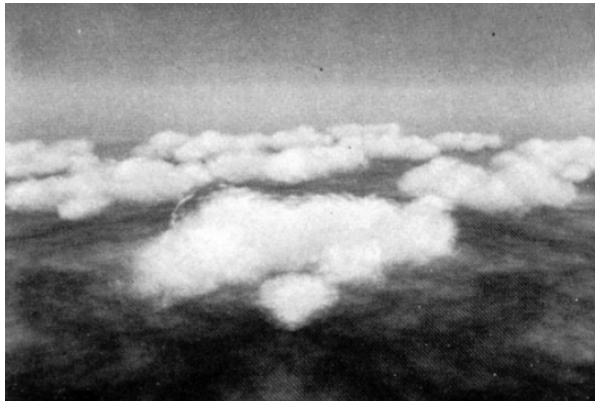
Textures procédurales

Résultat d'une **fonction mathématique**
ou d'un **programme**

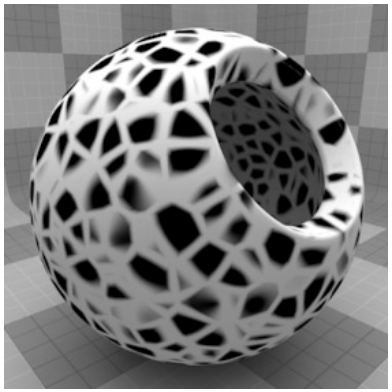
+ *color map*



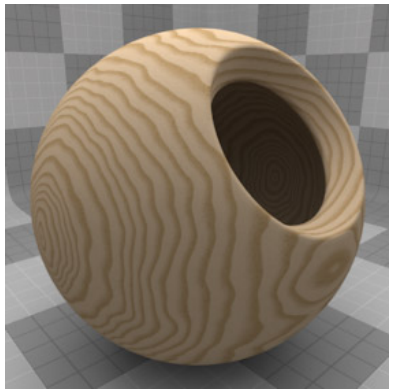
mod(x,a)/a *(sin(x)+1)/2*



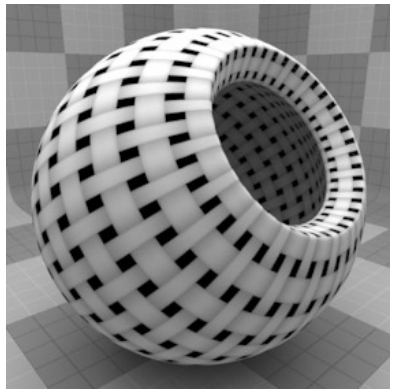
[Gardner1985]



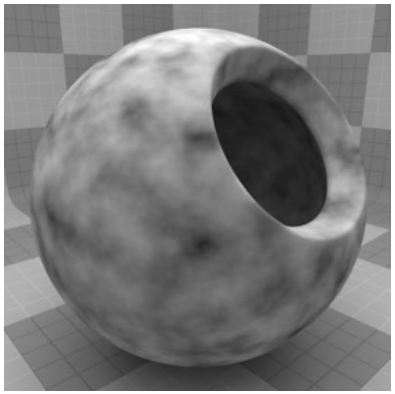
Texture cellulaire
[Worley 1996]



Texture de bois



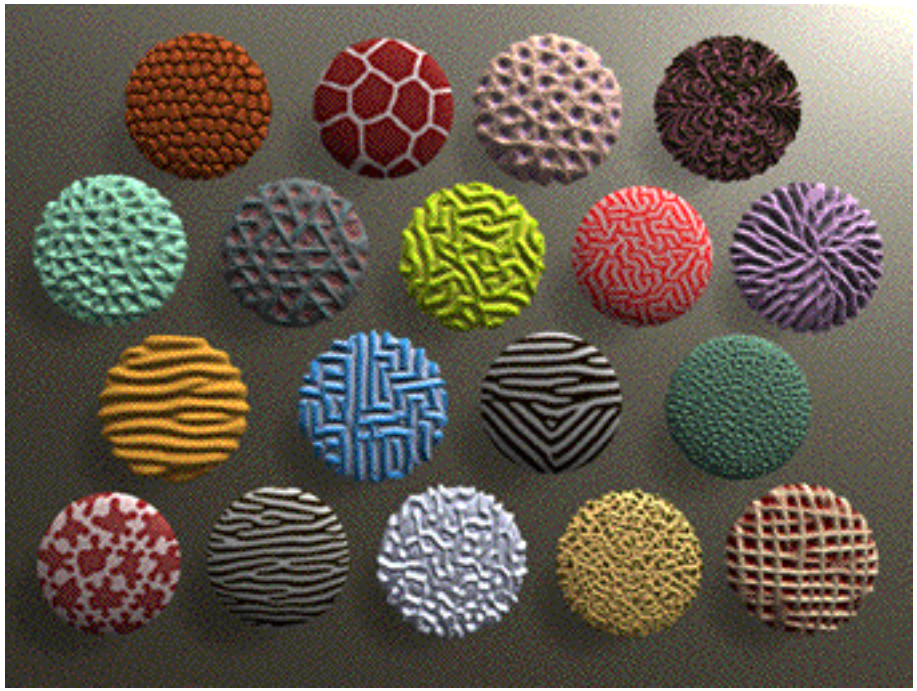
Texture de tissage



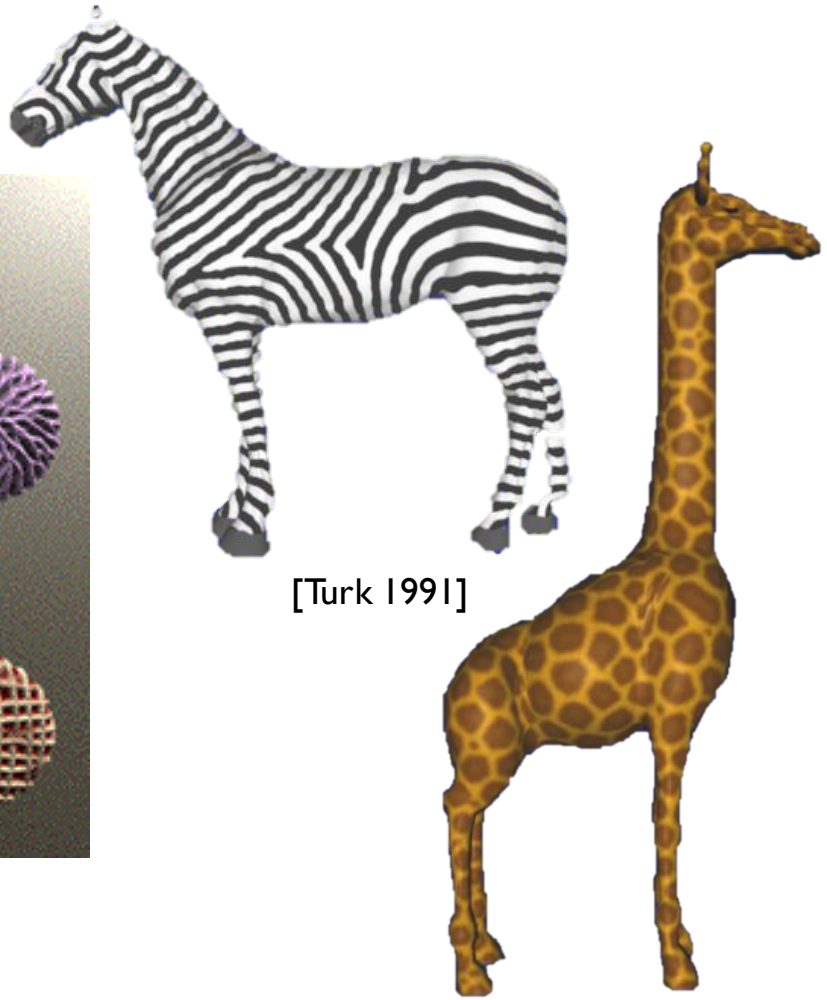
Texture de bruit

Images: documentation « modo »

Réaction-diffusion



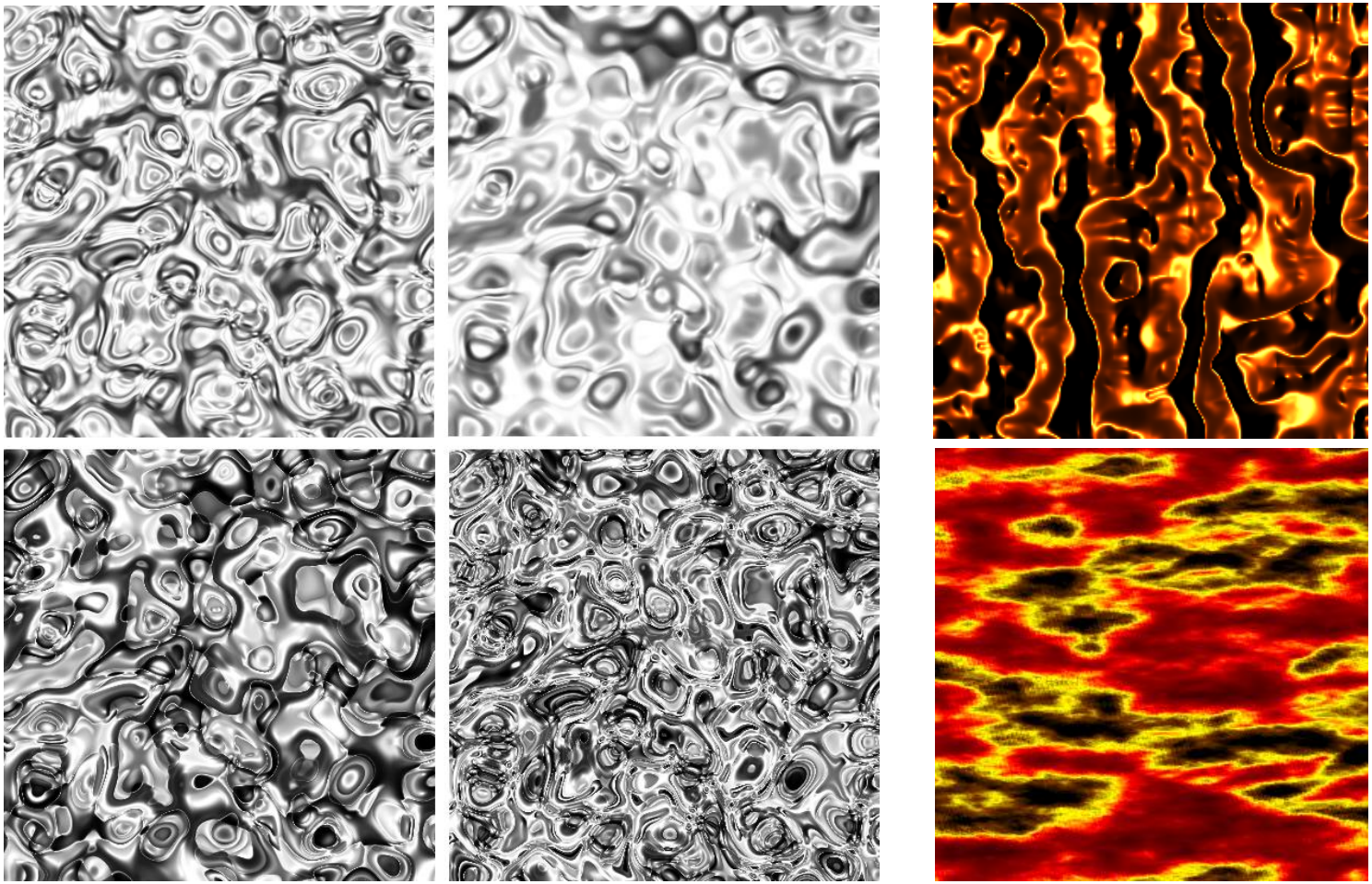
[Kass & Witkin 1991]



[Turk 1991]

Textures de Perlin

Bruit fractal continu + *color map*



An Image Synthesizer. Ken Perlin, 1985

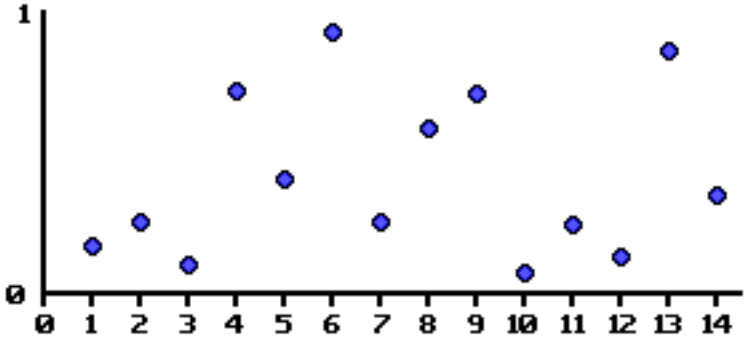
Fonction de bruit

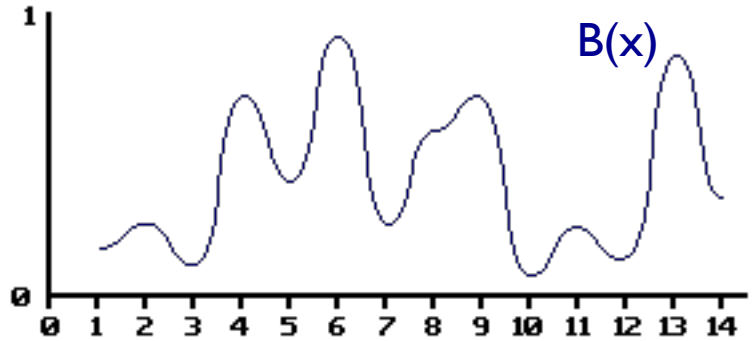
Fonctions de base (1D)

B(x) = **interpolation de valeurs aléatoires**, en des points régulièrement espacés

Pré-calcul des valeurs (tableau 1D)

Perlin : interpolation des **gradients aléatoires**

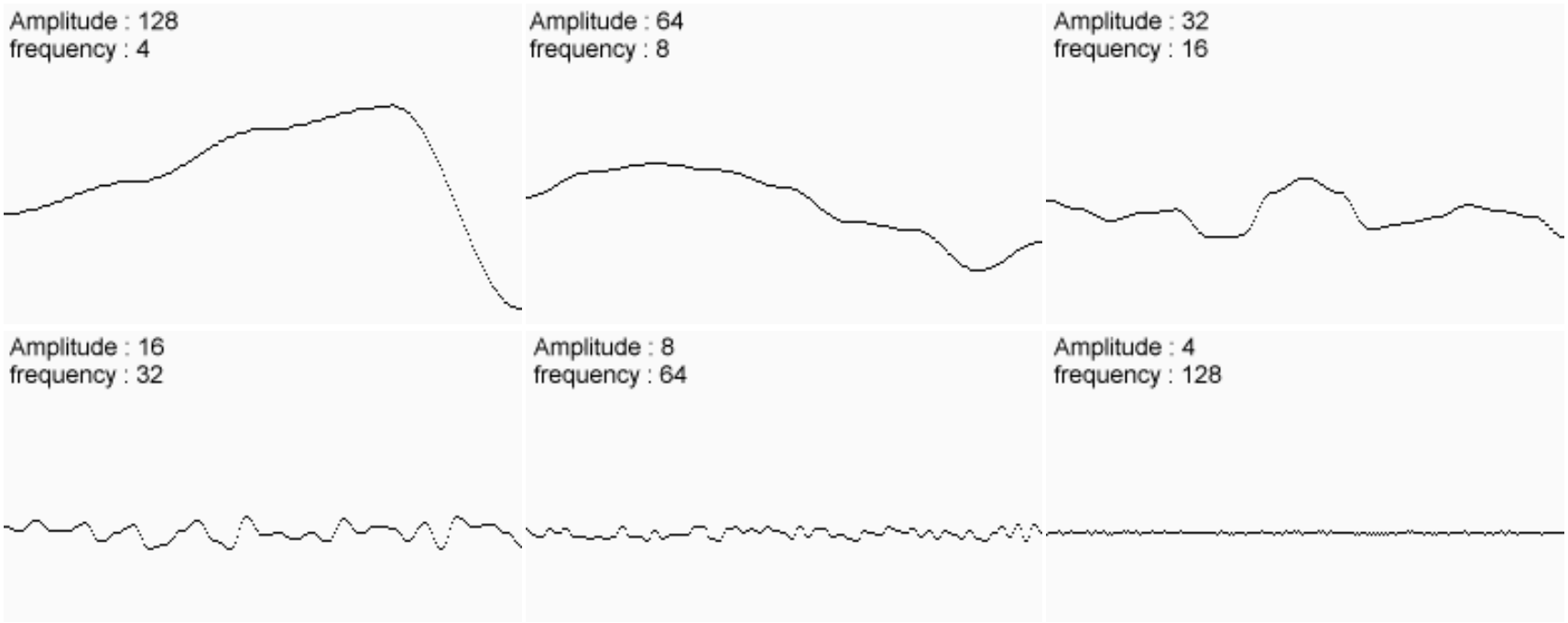




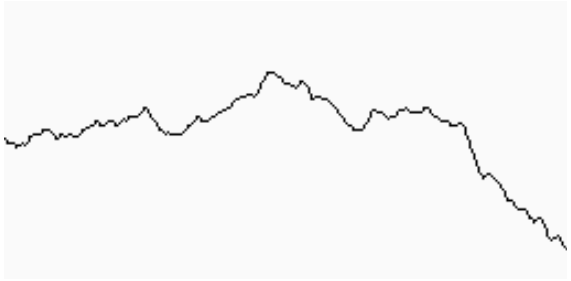
Images: Hugo Elias

Bruit turbulent

Sommer des copies de **B** à plusieurs échelles

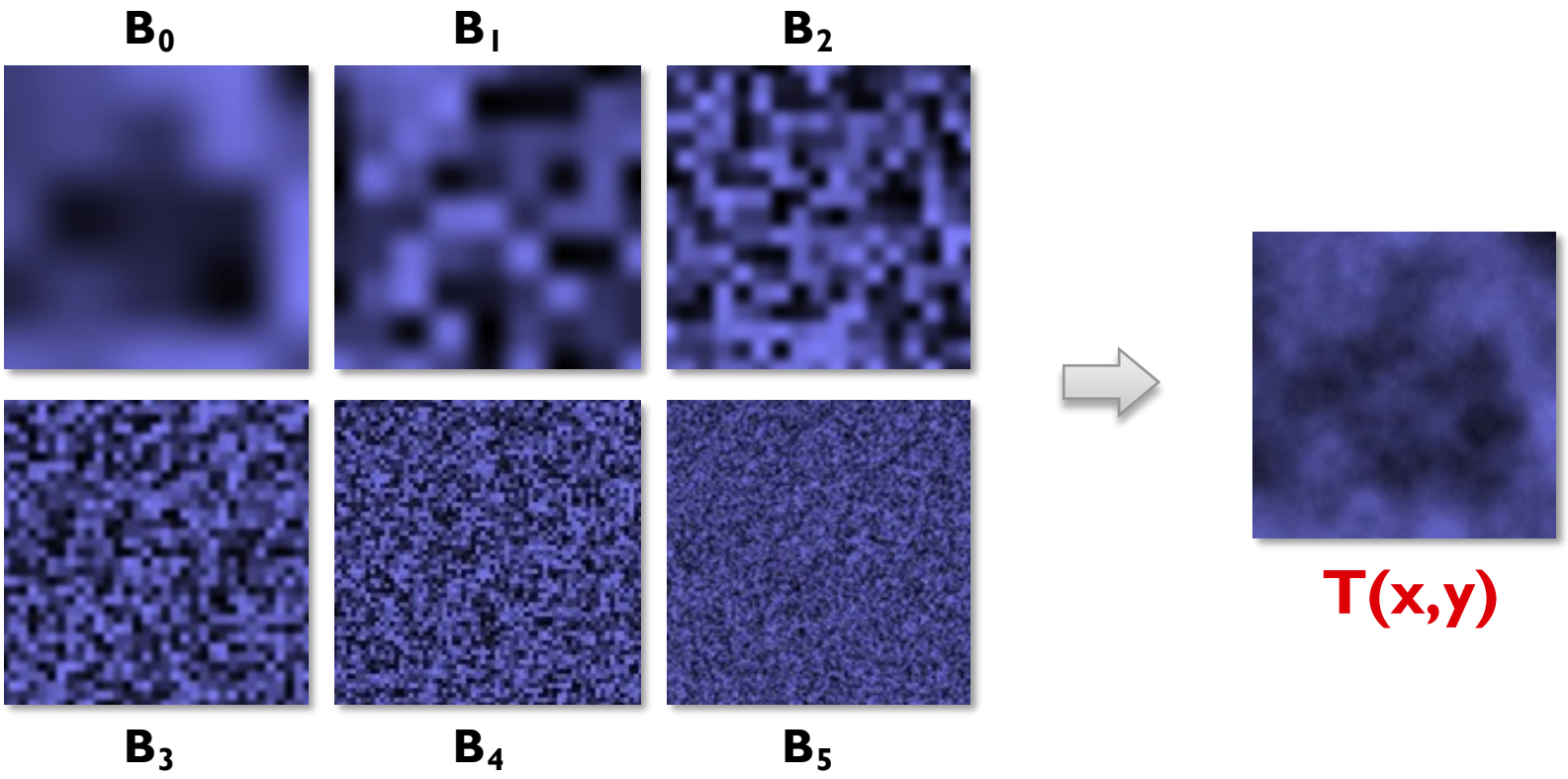


$$T(x) = \sum 1/2^i \; B(2^i x)$$



Texture

Extension des bruits **en 2D ou 3D**



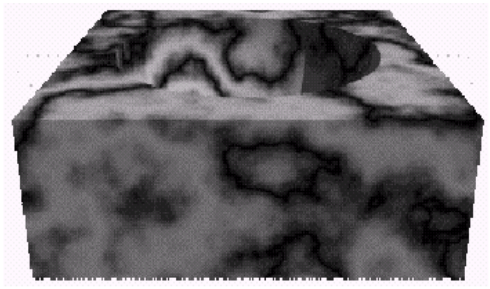
Paramètres : **amplitude** et **fréquence** des $\mathbf{B_i}$

ici, amplitude $1/2^i$ et fréquence $2^i \Longrightarrow$ « octave »

Images: Hugo Elias

Utilisation

Modification d'une **image** ou d'une **fonction simple**

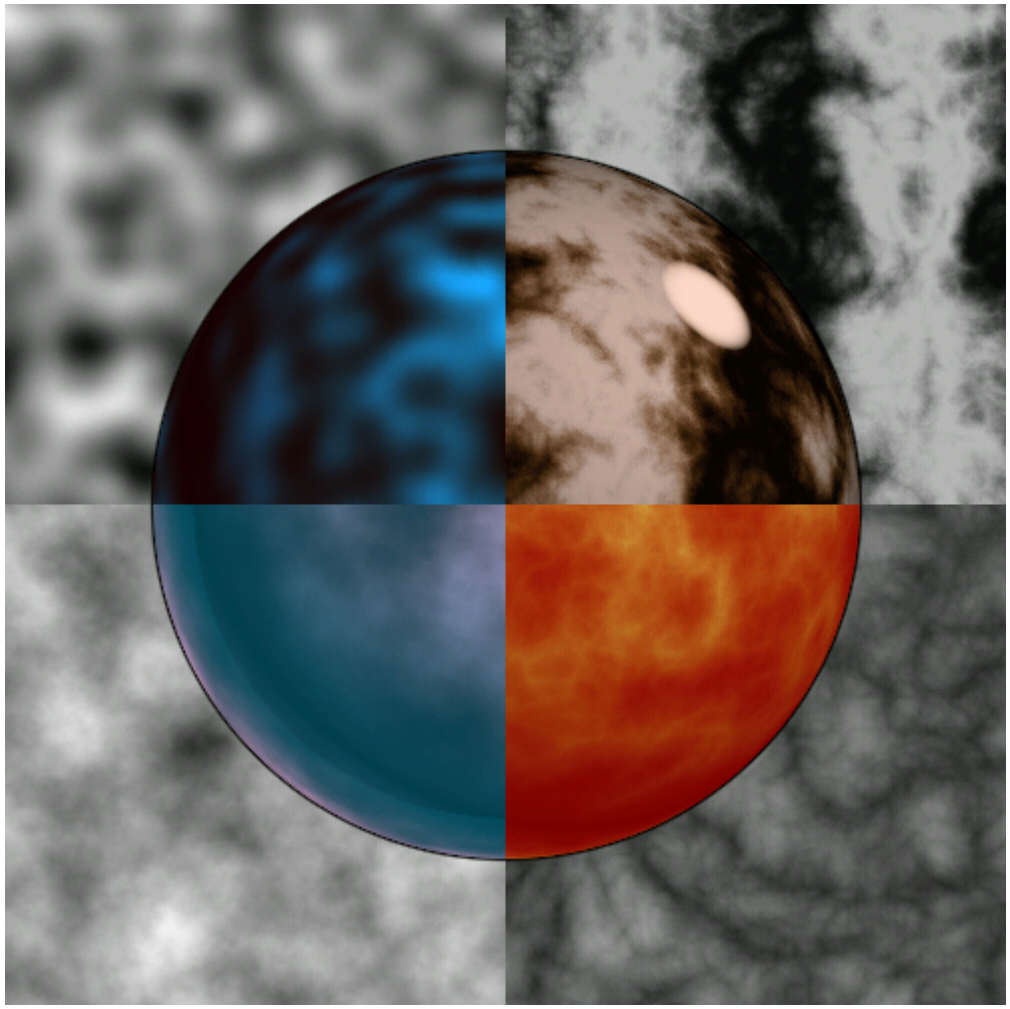


$$I(x,y,z) = \cos(x + T(x,y,z))$$

Bruit de Perlin

Bruit B

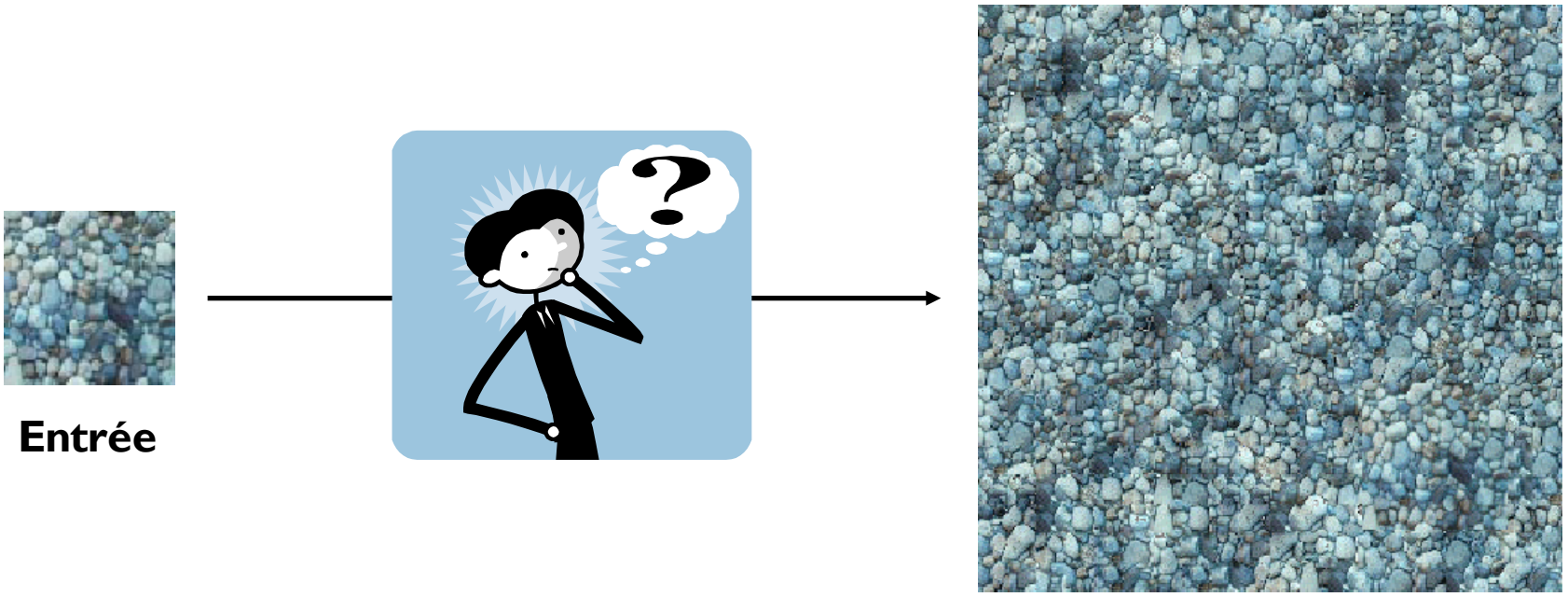
sin(x + T)



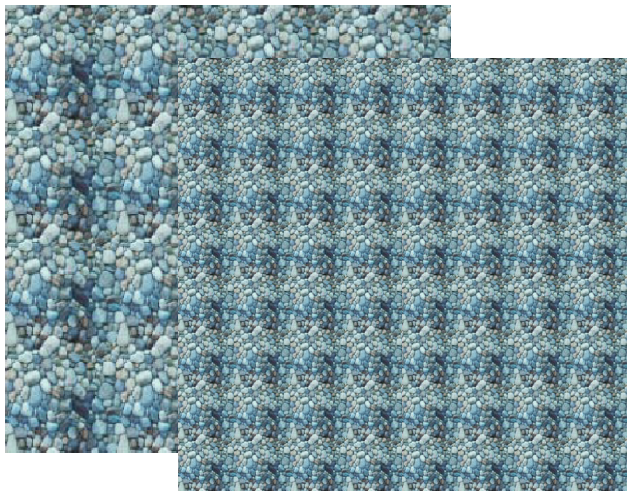
T= ∑ 1/f(B)

∑ 1/f(|B|)

Synthèse par l'exemple

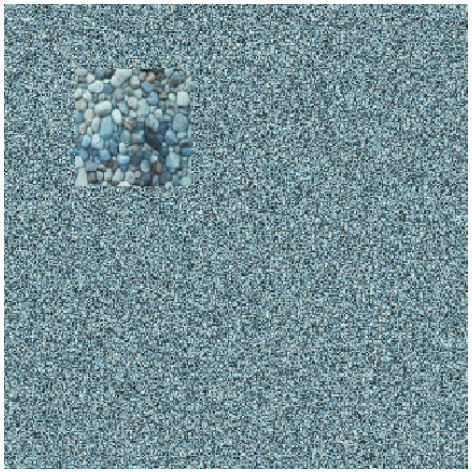


Ce qui ne fonctionne pas…



Recopier

- trop de répétition
- *aliasing*



Echantillonner les valeurs

- pas de structure
- cohérence horizontale

Echantillonner les fréquences (FFT)

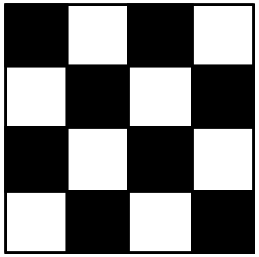
- pas de cohérence entre fréquences
- cohérence verticale

Approches paramétriques

En deux étapes :

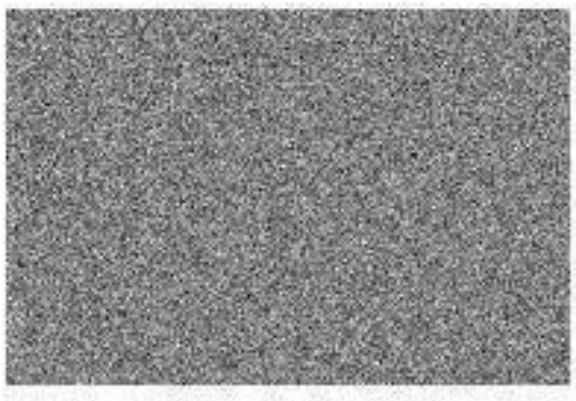
1. **Analyse**

   Extraire les paramètres d'un **modèle statistique** (distribution des intensités, gradients…)



2. **Synthèse**

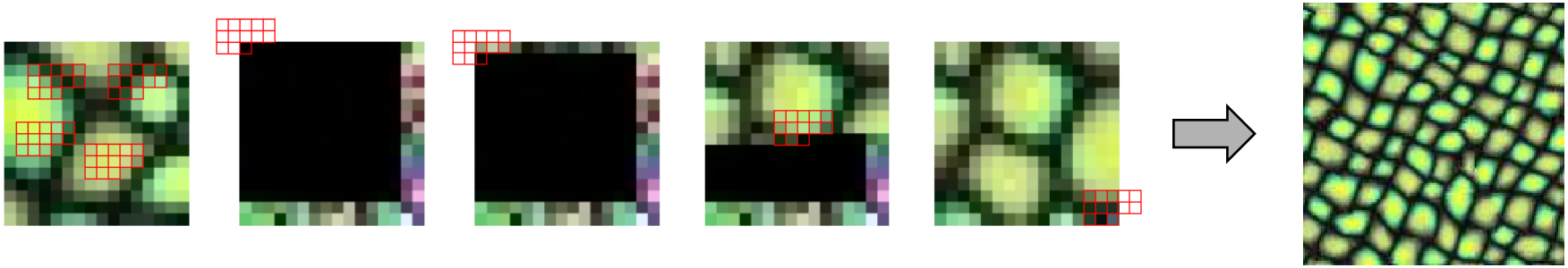
   Contraindre les statistiques d'une image de bruit vers celles analysées.



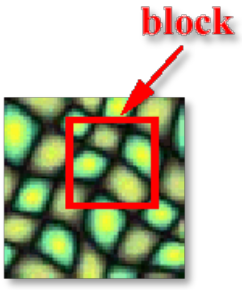
Approches non-paramétriques

Copie de **pixels** en respectant le voisinage

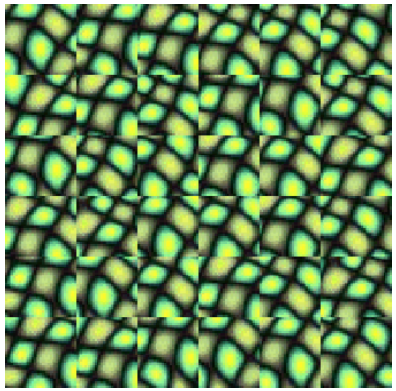
[Wei et al. 2000]



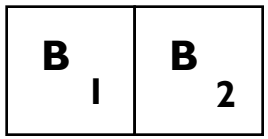
Copie de **patchs**

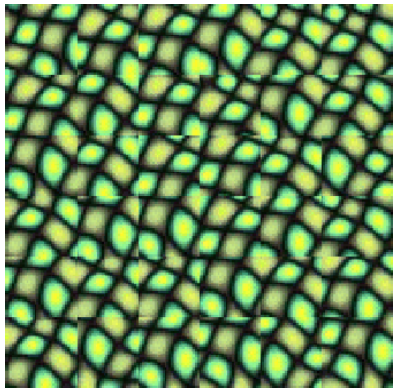


[Efros et al. 2001]

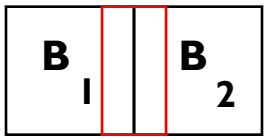


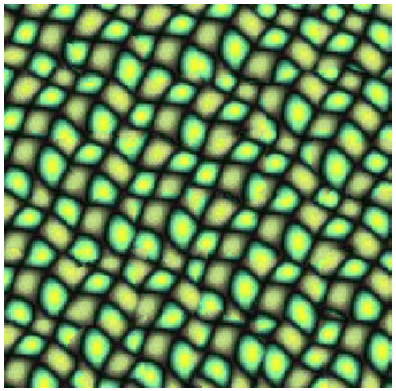
Placement aléatoire



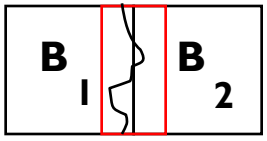


Contraintes aux bords

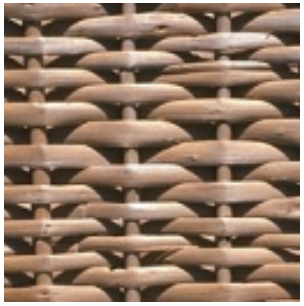




Optimisation du collage

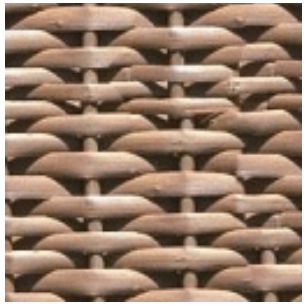


Approches non-paramétriques

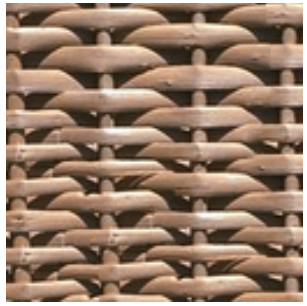


Exemples en entrée

Copie de pixels
[Wei et al. 2000]



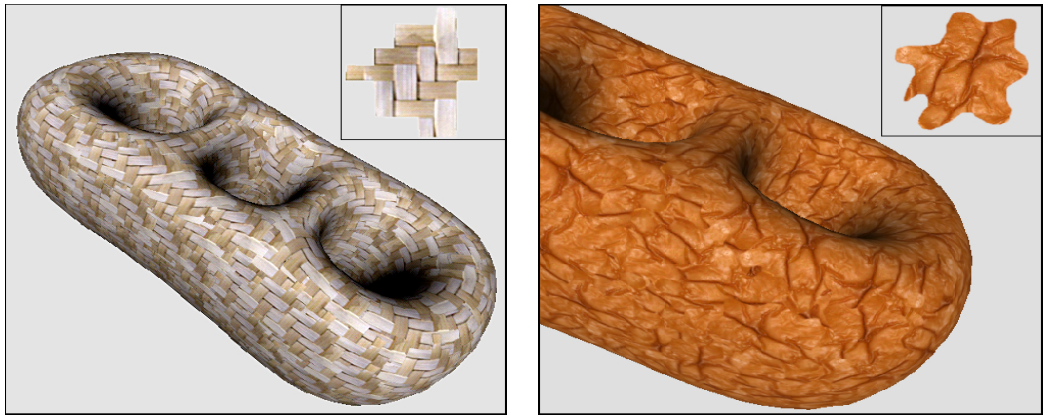
Copie de patches
[Efros et al. 2001]



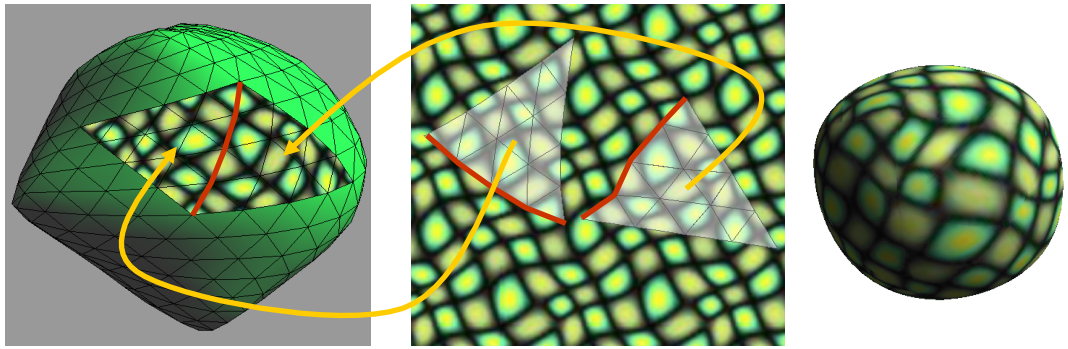
Optimisation globale
[Kwatra et al. 2005]

Synthèse 2D sur la surface

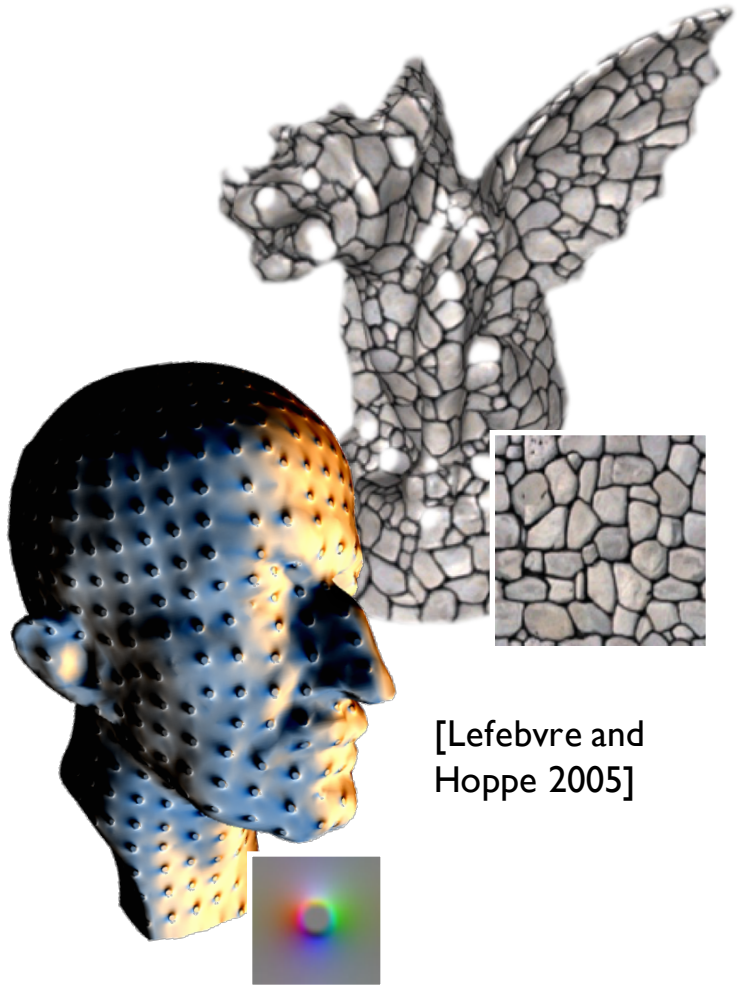
Idem en spécifiant le **voisinage sur le maillage**



[Praun et al. 2000]



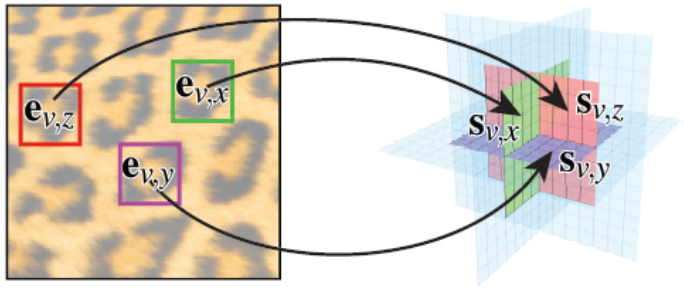
[Soler et al. 2002]

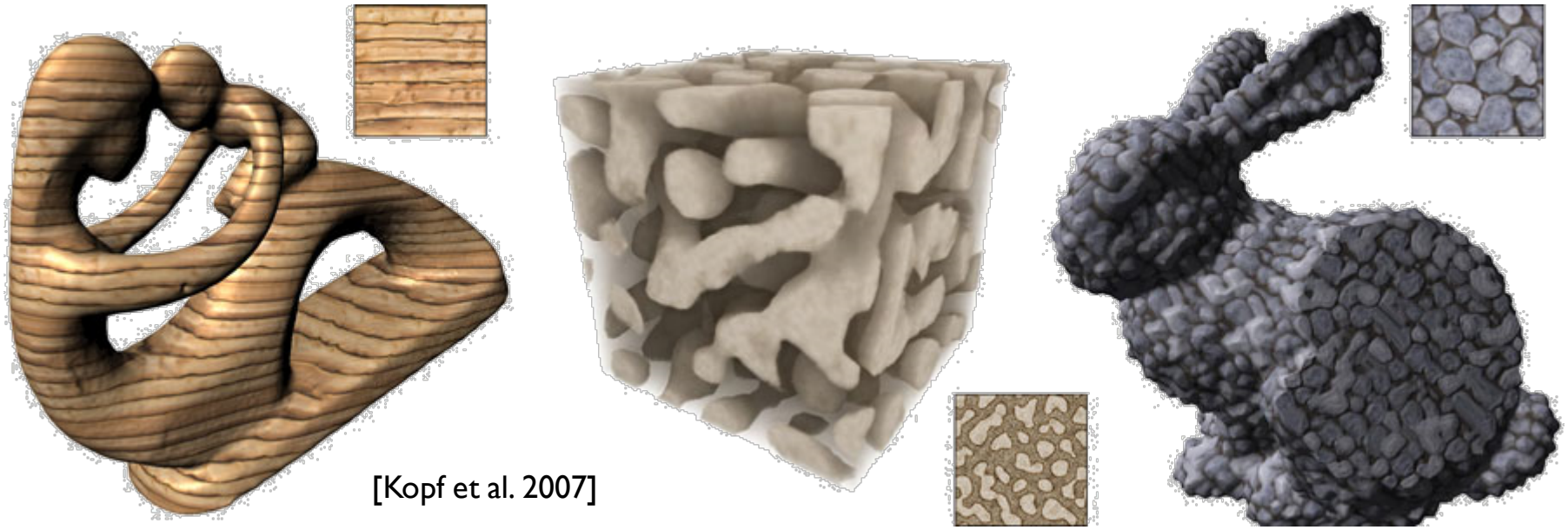


[Lefebvre and Hoppe 2005]

Textures volumiques

Synthèse par **voisinage selon 3 plans**





[Kopf et al. 2007]