

Textures

Modélisation de l'apparence

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

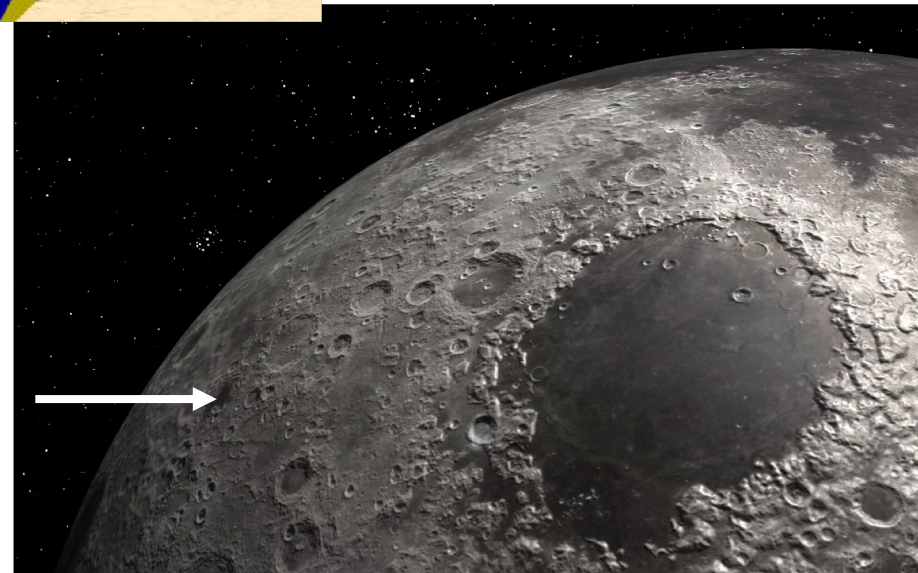
On veut garder une seule face, mais plusieurs couleurs



⇒ **Texture de couleurs**

Des micro-polygones seraient nécessaires


⇒ **Texture de normales**

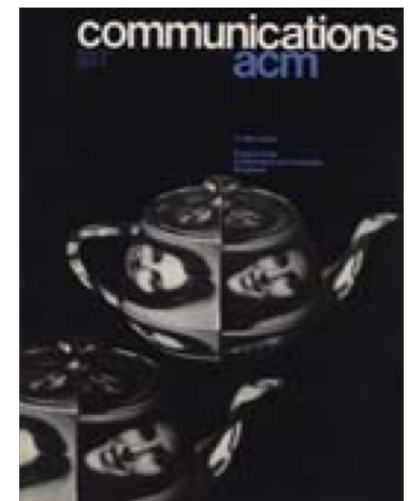
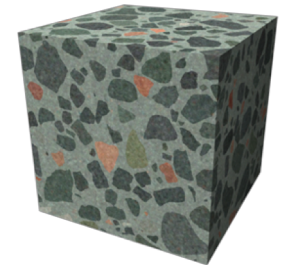
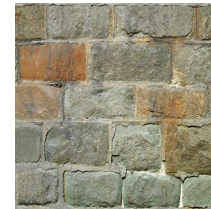


Les textures

Champ scalaire discret (*lookup table*)

défini sur un domaine :

- linéaire (1D) 
- rectangulaire (2D)
- cubique (3D)

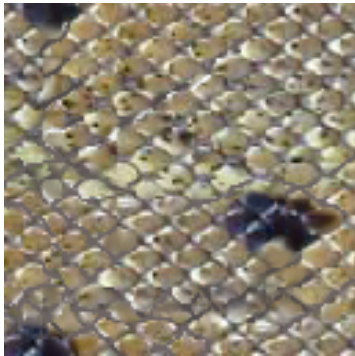


Texture and reflection in computer generated images.
J. F. Blinn & M.E. Newell. 1976

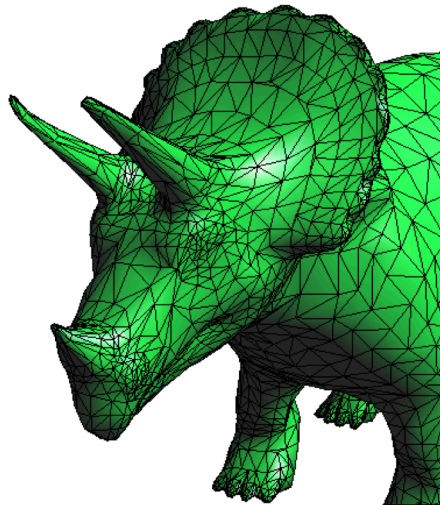
Les textures

Appliquées sur la surface par des **coordonnées de texture**

- spécifiées en chaque sommet
- interpolées par fragment



x



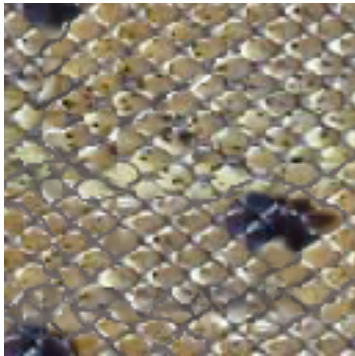
=



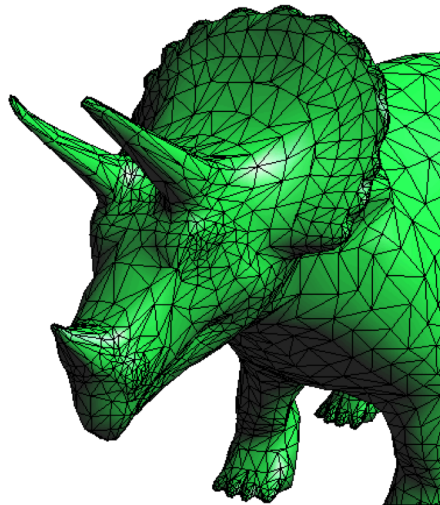
Les textures

Support matériel

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



x



=



Les textures

Peuvent être utilisées pour spécifier :



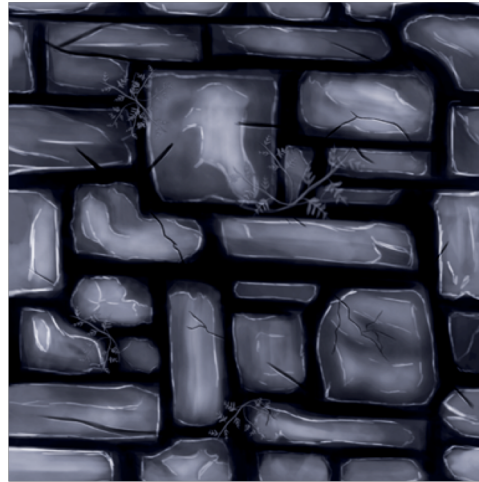
Les textures

Peuvent être utilisées pour spécifier :

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



Diffuse Map



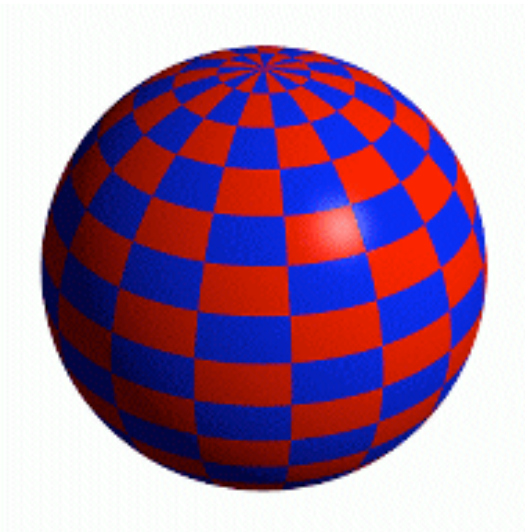
Specular Map



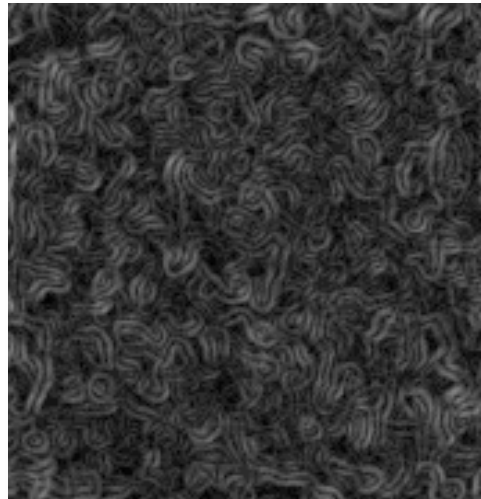
Les textures

Peuvent être utilisées pour spécifier :

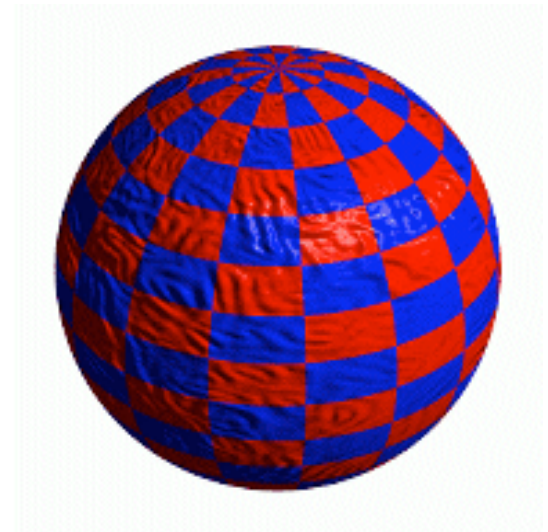
- La couleur
- Les **normales** (*bump / normal mapping*)



Texture Diffuse mappée
sur la sphère



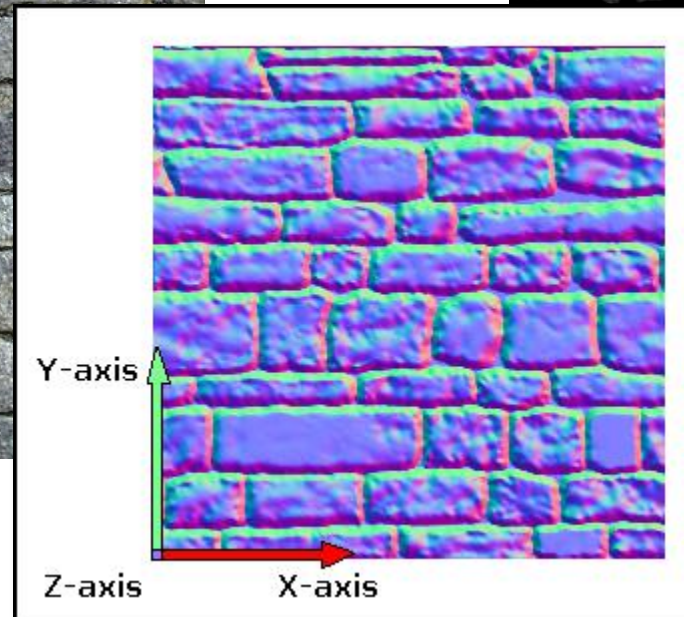
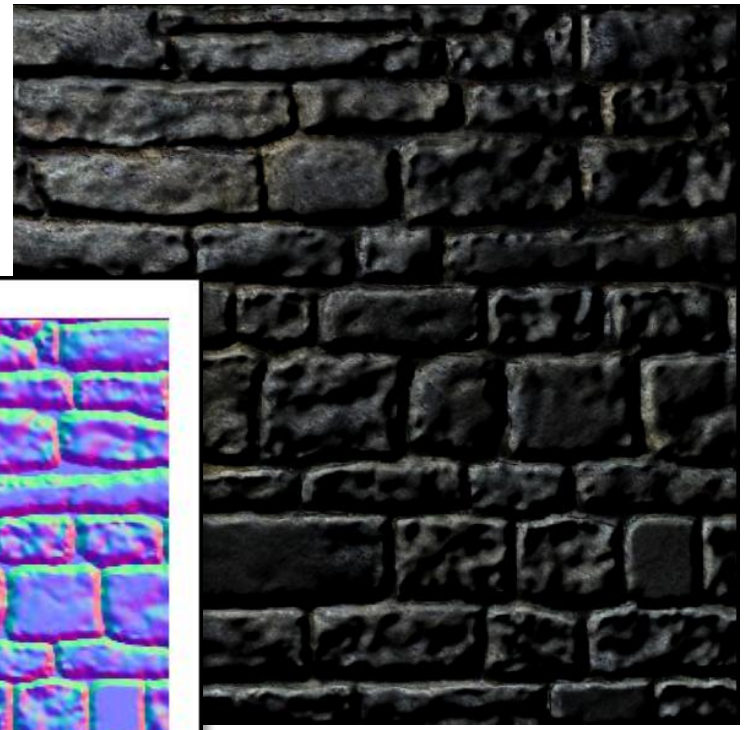
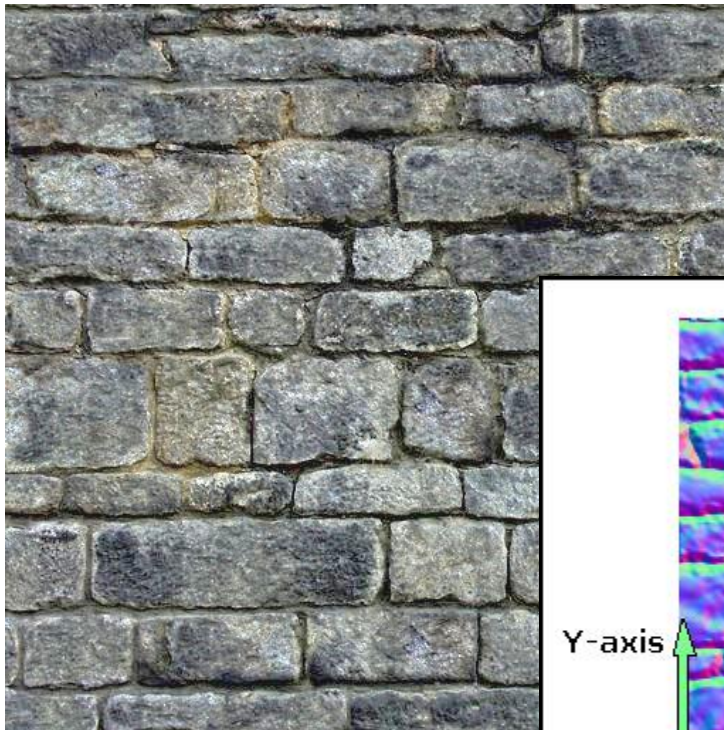
Bump Map



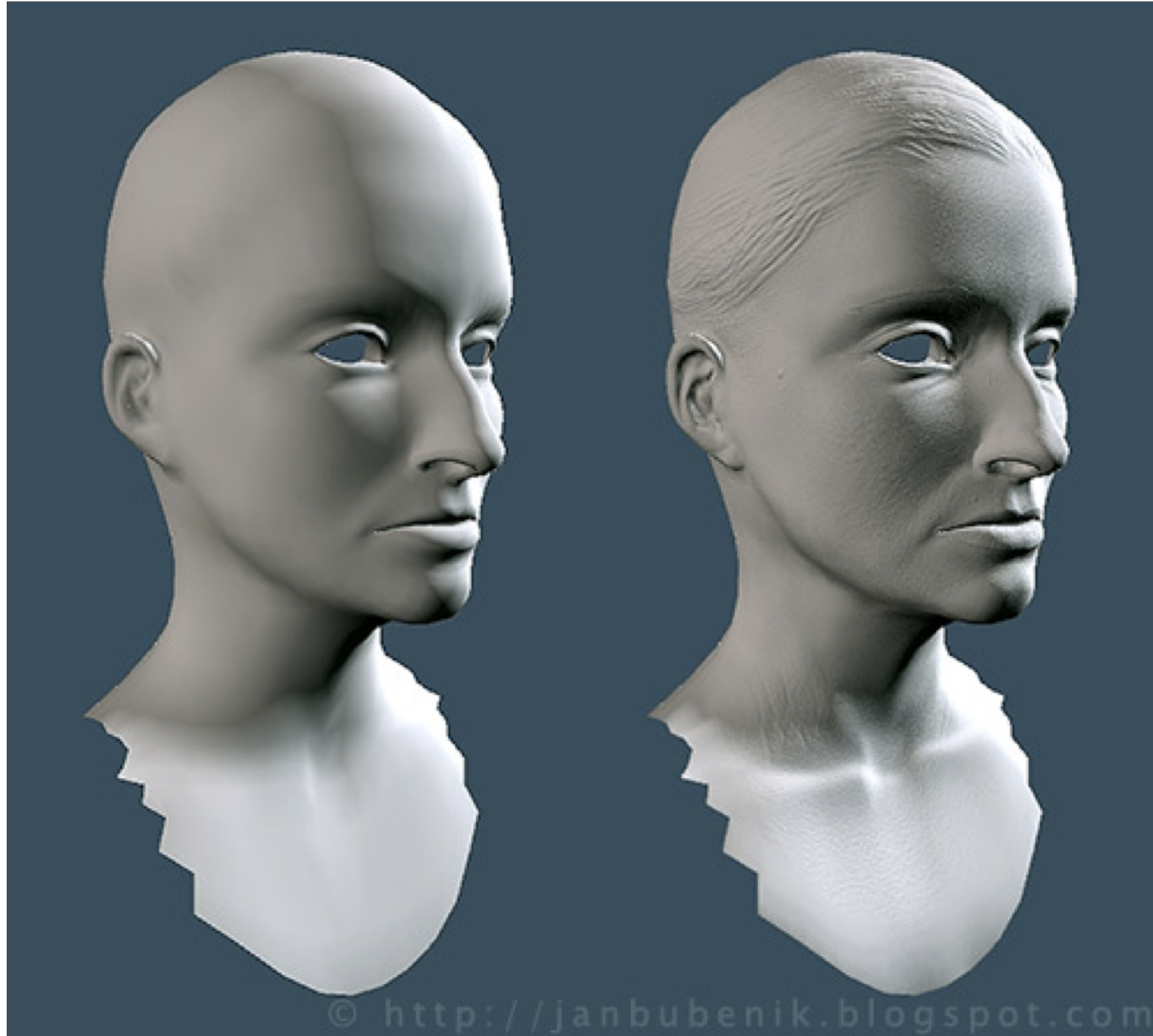
Textures combinées

Perturbation des normales

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



Perturbation des normales



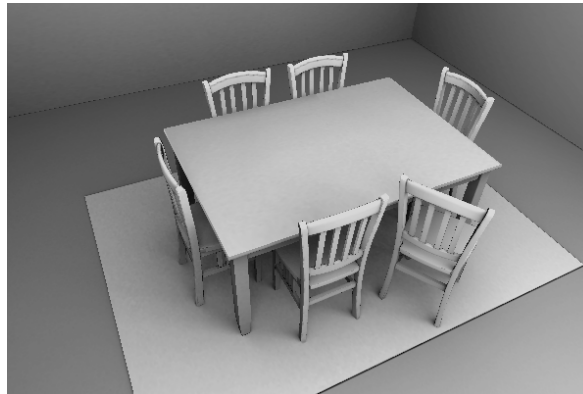
Les textures

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

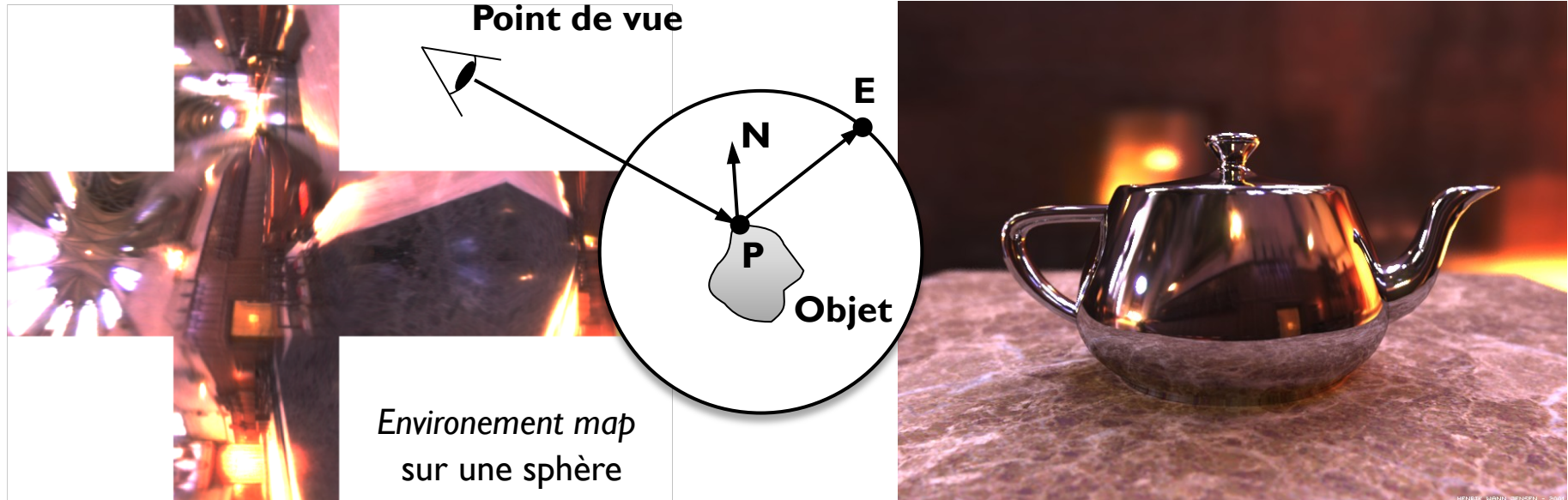


Textures combinées

Les textures

Peuvent être utilisées pour spécifier :

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



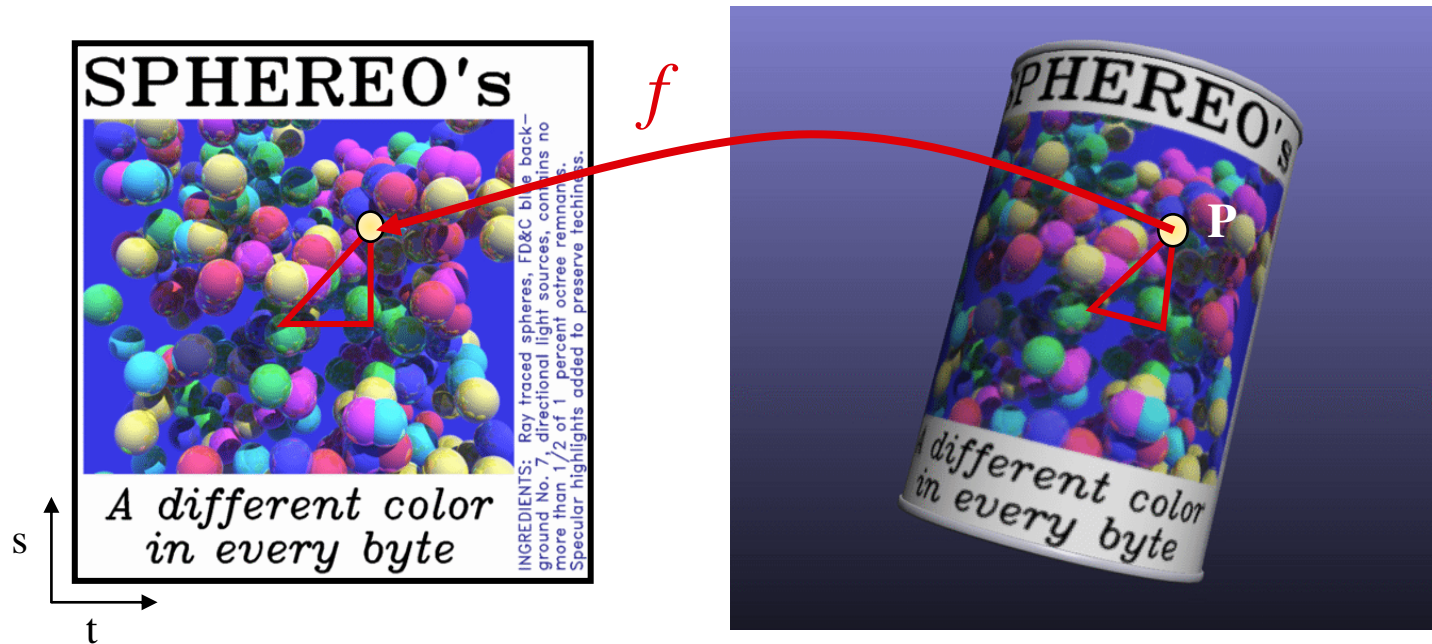
Les textures 2D



Image $I(s,t)$

Fonction de placage (*mapping*)

$$f: \mathbf{P}(x,y,z) \rightarrow (s,t) \text{ dans } [0,1]$$



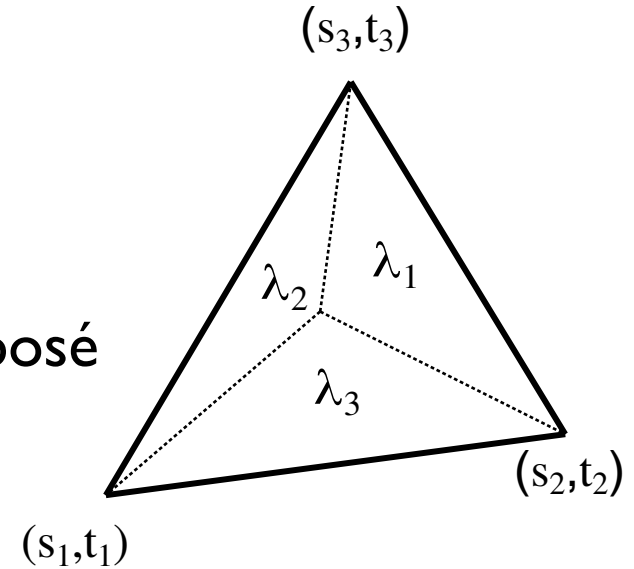
Coordonnées de texture

Définies aux sommets

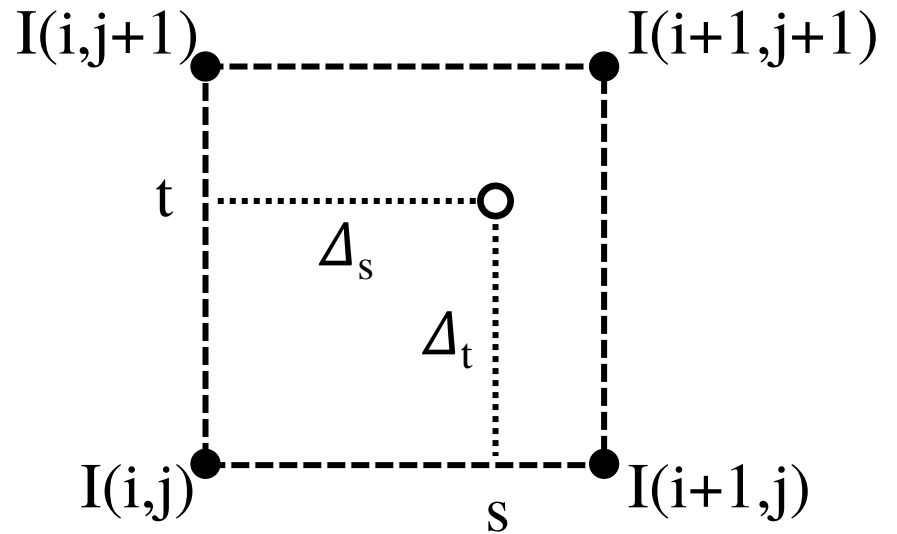
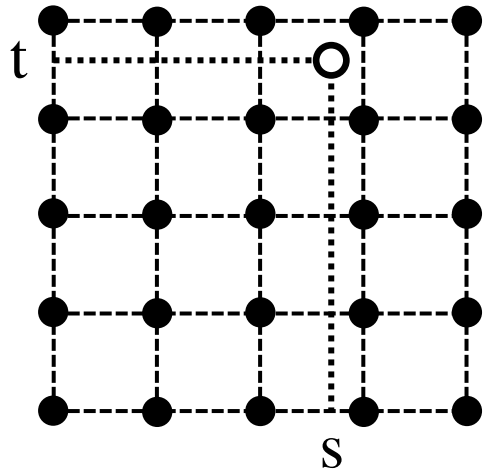
⇒ **Interpolation barycentrique**

$$(s,t) = \lambda_1(s_1,t_1) + \lambda_2(s_2,t_2) + \lambda_3(s_3,t_3)$$

avec λ_i l'aire relative signée du triangle opposé

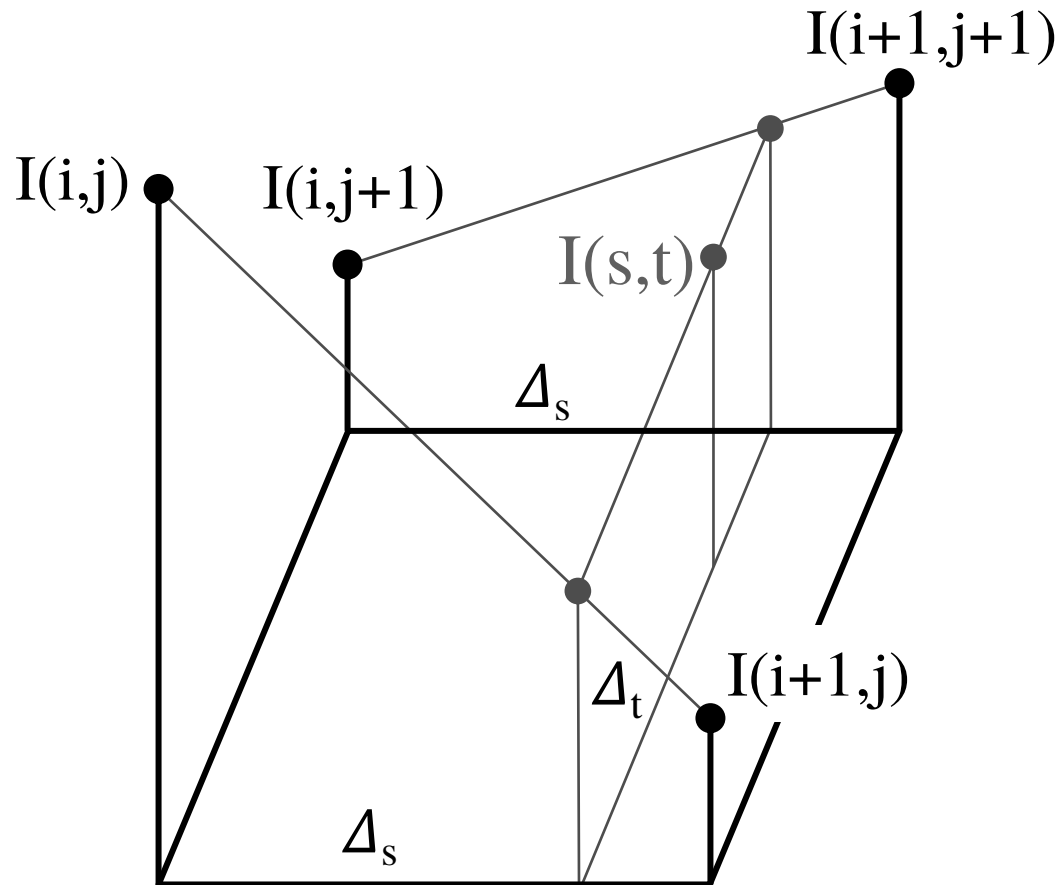


Ré-échantillonnage

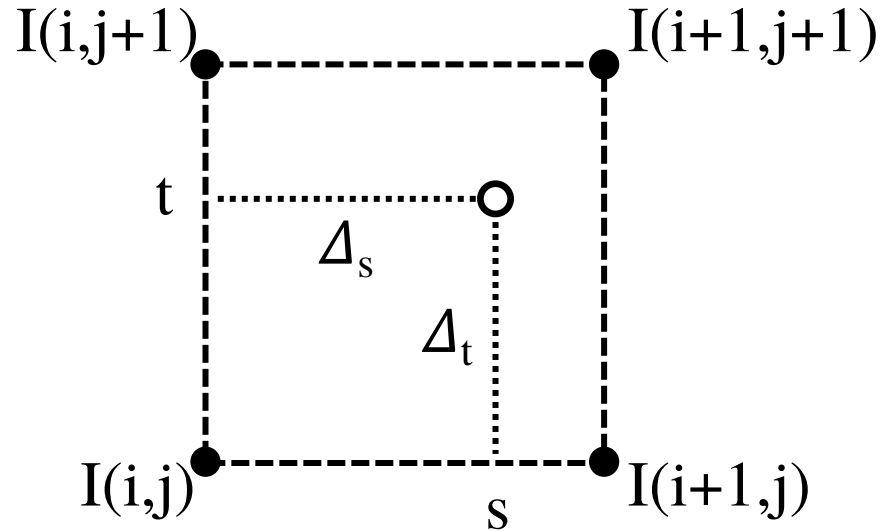
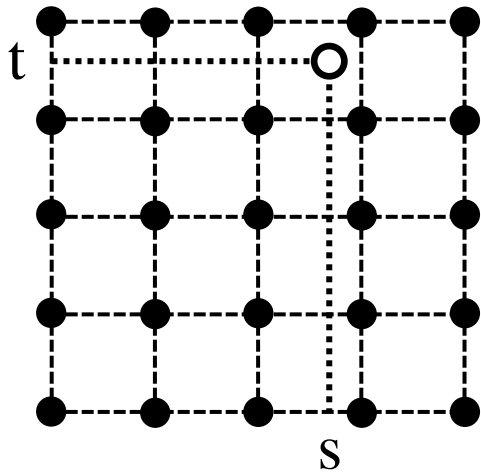


$$I(s,t) = I(i+\Delta_s, j+\Delta_t)$$

Interpolation bilinéaire



Interpolation bilinéaire



Interpolation bilinéaire

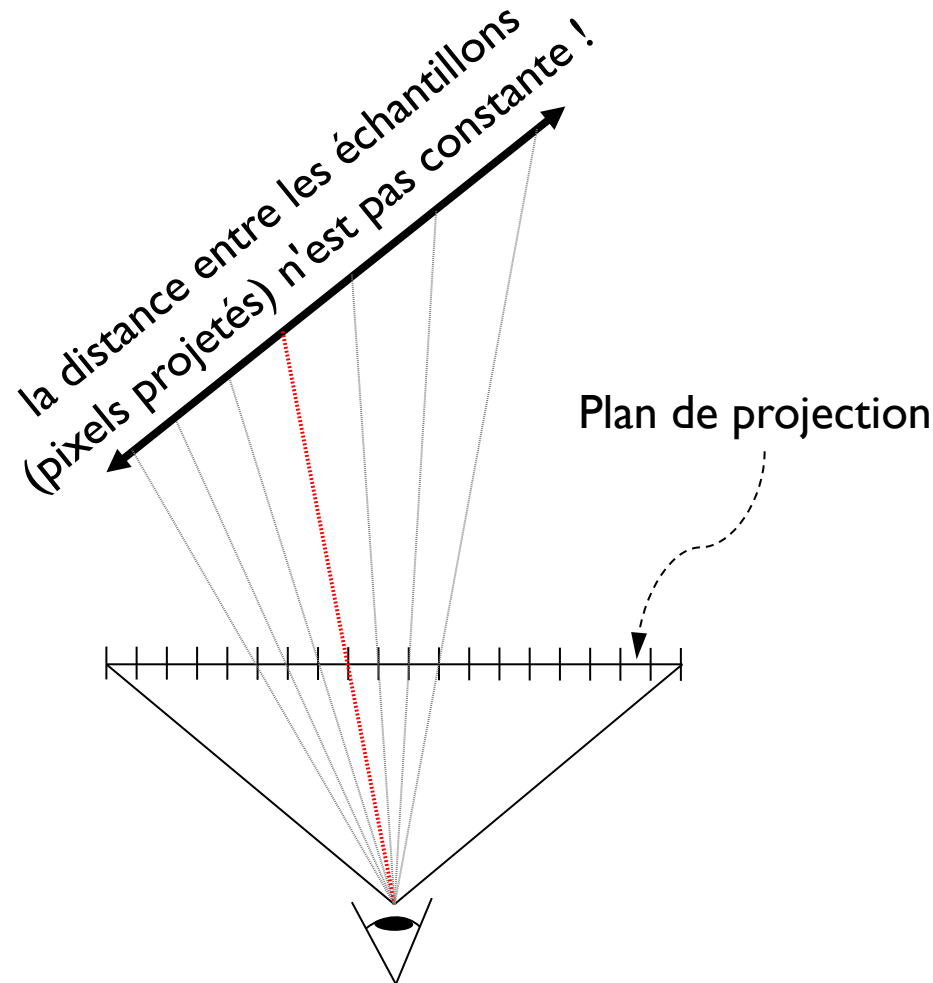
$$\begin{aligned} I(s,t) &= I(i+\Delta_s, j+\Delta_t) \\ &= (1-\Delta_s)(1-\Delta_t) I(i,j) + \Delta_s(1-\Delta_t) I(i+1,j) \\ &\quad + (1-\Delta_s)\Delta_t I(i,j+1) + \Delta_s\Delta_t I(i+1,j+1) \end{aligned}$$

...pour chaque composante RGB

Interpolation des coordonnées

La projection perspective n'est pas linéaire !

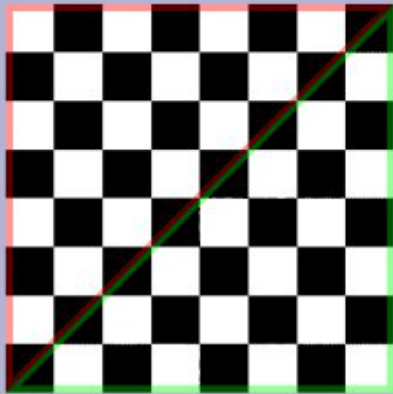
⇒ On ne peut pas interpoler linéairement dans le plan image 2D



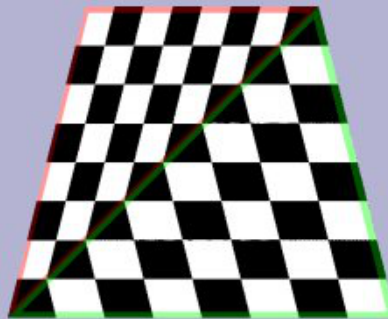
Attention à la perspective !

Solution : calculer les coordonnées barycentrique de l'espace objet à partir de celles en espace image

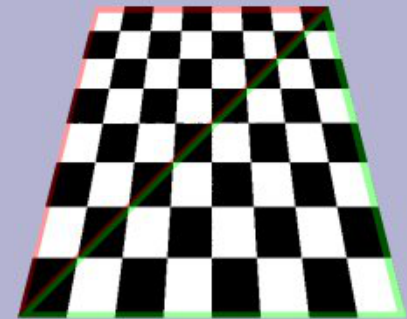
Interpolation
à z constant



Interpolation linéaire
en espace image



**Prise en compte
de la perspective**



Prise en compte de la perspective

Extrémités d'un segment en espace monde :

$$\mathbf{p}_1^w = (x_1, y_1, z_1) \qquad \mathbf{p}_2^w = (x_2, y_2, z_2)$$

⇒ points correspondants en espace image :

$$\mathbf{p}_1 = \mathit{near} (x_1/z_1, y_1/z_1) \qquad \mathbf{p}_2 = \mathit{near} (x_2/z_2, y_2/z_2)$$

Point sur le segment $\mathbf{p}_1^w \mathbf{p}_2^w$ en espace monde :

$$\mathbf{p}^w(t) = (1-t) \mathbf{p}_1^w + t \mathbf{p}_2^w \qquad 0 \leq t \leq 1$$

⇒ point correspondant en espace image :

$$\mathbf{p}(s) = \mathit{near} \left(\frac{(1-s) x_1 + s x_2}{(1-s) z_1 + s z_2}, \frac{(1-s) y_1 + s y_2}{(1-s) z_1 + s z_2} \right)$$

Prise en compte de la perspective

Correspond à une **interpolation en espace image** de paramètre t :

$$(1-t) x_1/z_1 + t x_2/z_2 = \frac{(1-s) x_1 + s x_2}{(1-s) z_1 + s z_2}$$

Résoudre cette équation en s :

$$s = \frac{t z_1}{z_2 + t (z_1 - z_2)}$$

⇒ Calculer t en **espace image**, utiliser cette équation pour calculer s en **espace monde**, et utiliser s pour interpoler les coordonnées aux sommets

Prise en compte de la perspective

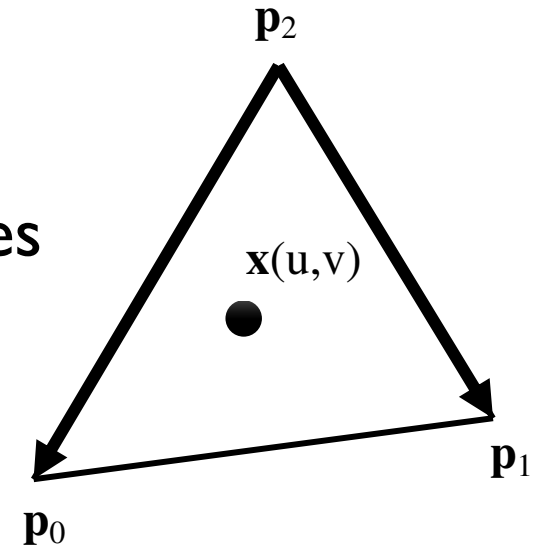
Coordonnées barycentrique d'un pixel pour un triangle en **espace image** :

$$\mathbf{x} = u \mathbf{p}_0 + v \mathbf{p}_1 + (1 - u - v) \mathbf{p}_2$$

Coordonnées correspondantes exprimées en **espace objet** :

$$u^w = \frac{z_1 z_2 u}{z_0 z_1 + z_1 u (z_2 - z_0) + z_0 v (z_2 - z_1)}$$

$$v^w = \frac{z_0 z_2 v}{z_0 z_1 + z_1 u (z_2 - z_0) + z_0 v (z_2 - z_1)}$$



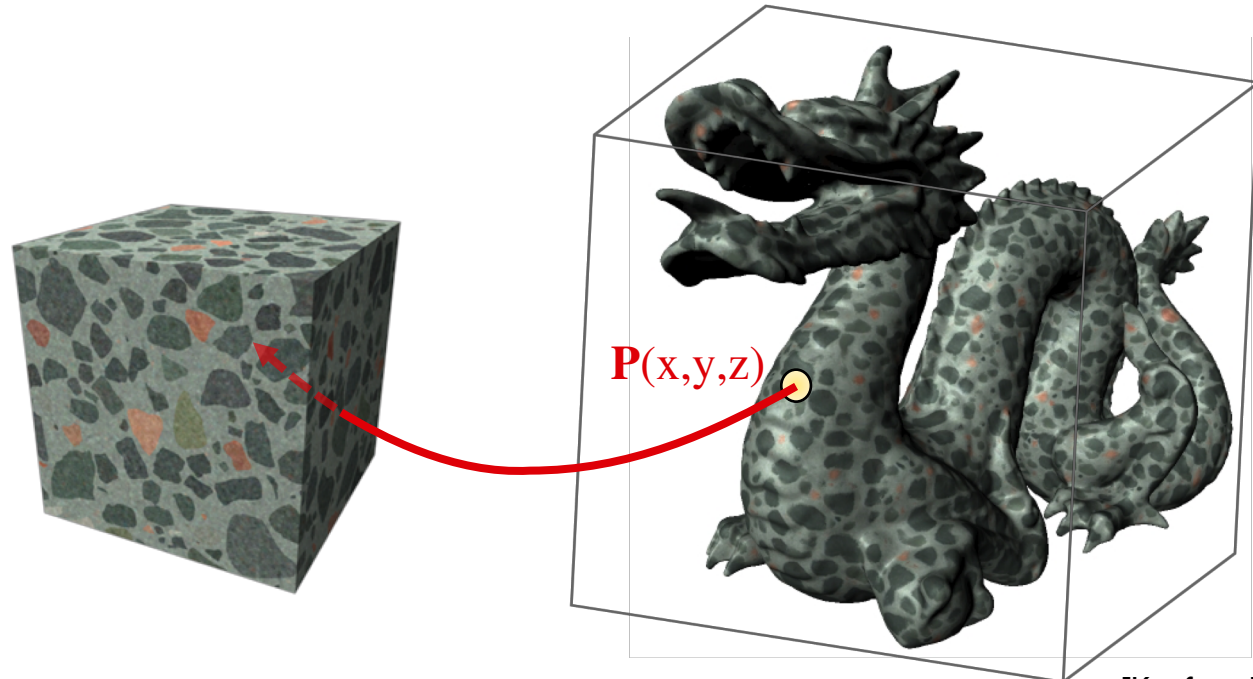
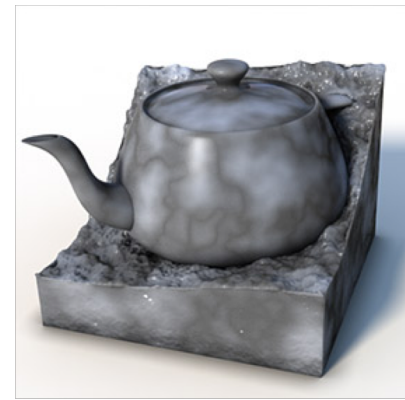
Les textures 3D

Volume $V(s,t,r)$

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

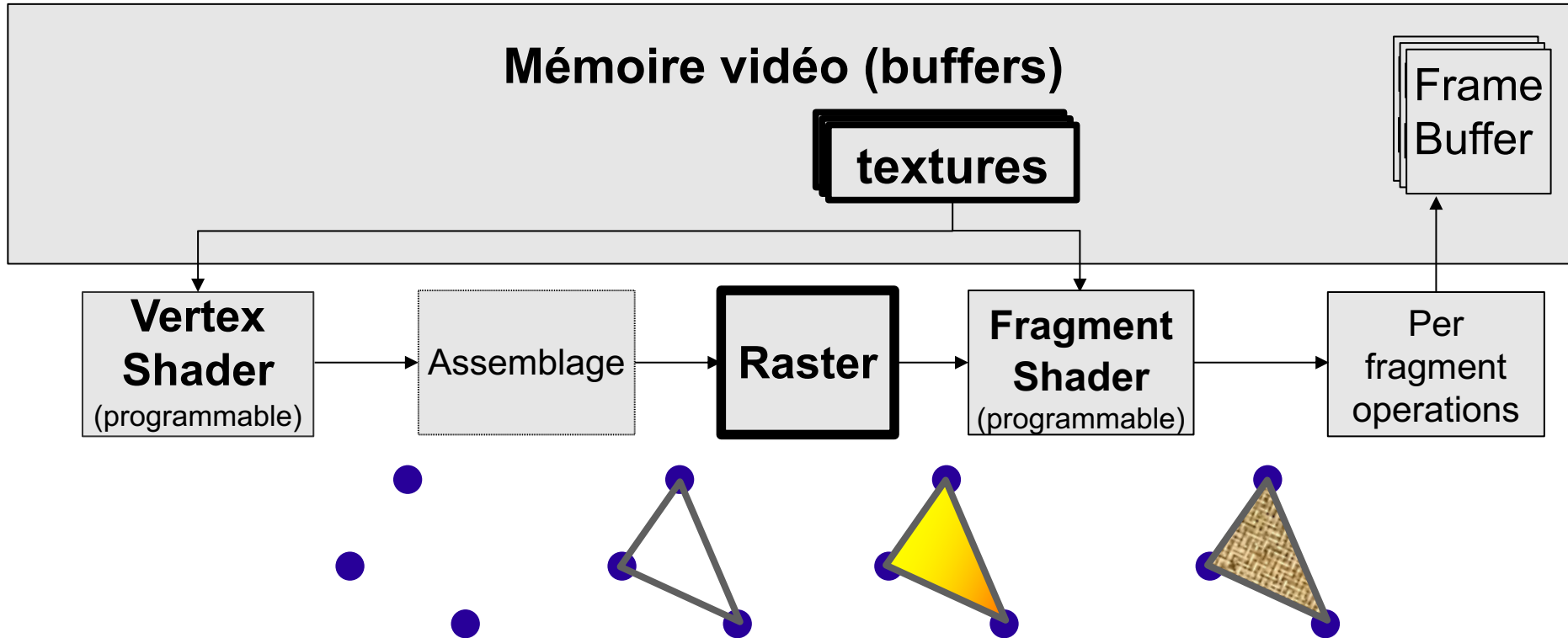
$$x = s, y = t, z = r$$

Attention : coût mémoire important



[Kopf et al. 2006]

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;

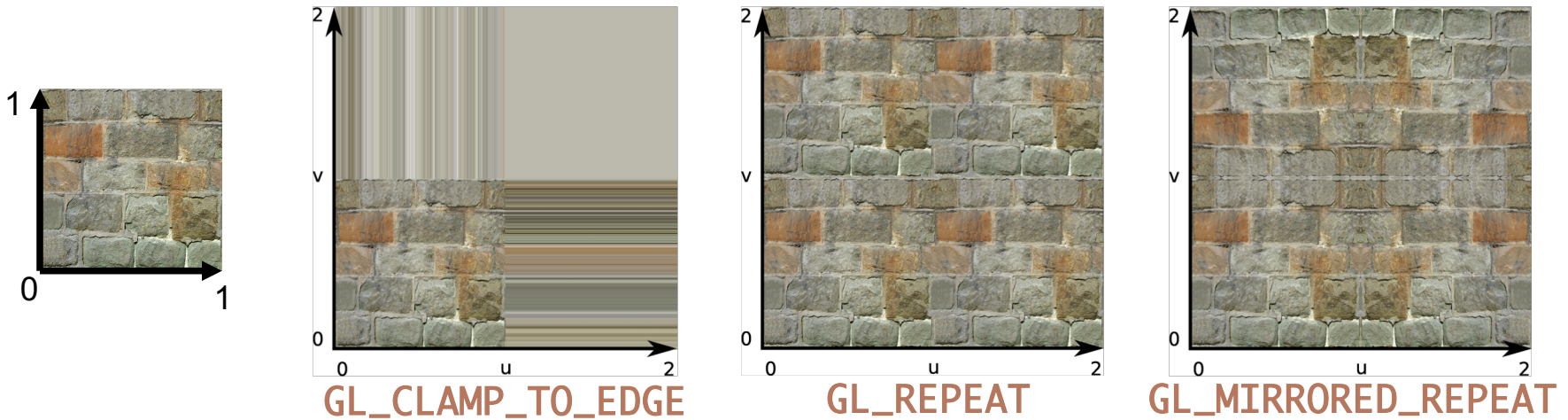
out vec4 out_color;

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**



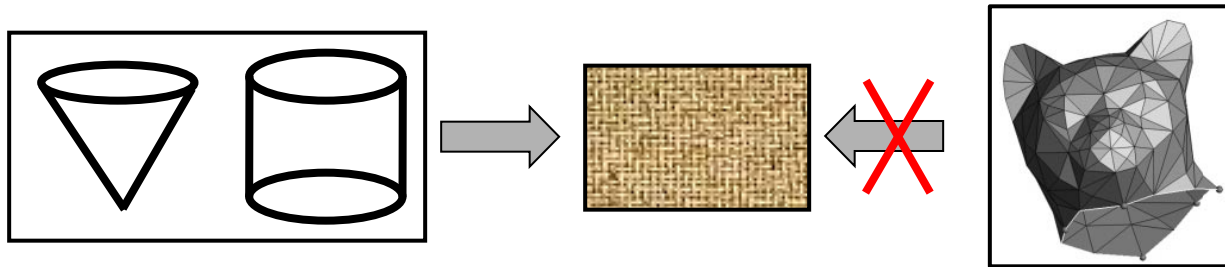
Possibilité d'appliquer une **matrice de transformation**
⇒ effet de changement d'échelle, glissement, etc.

Problèmes

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

Fonction de placage

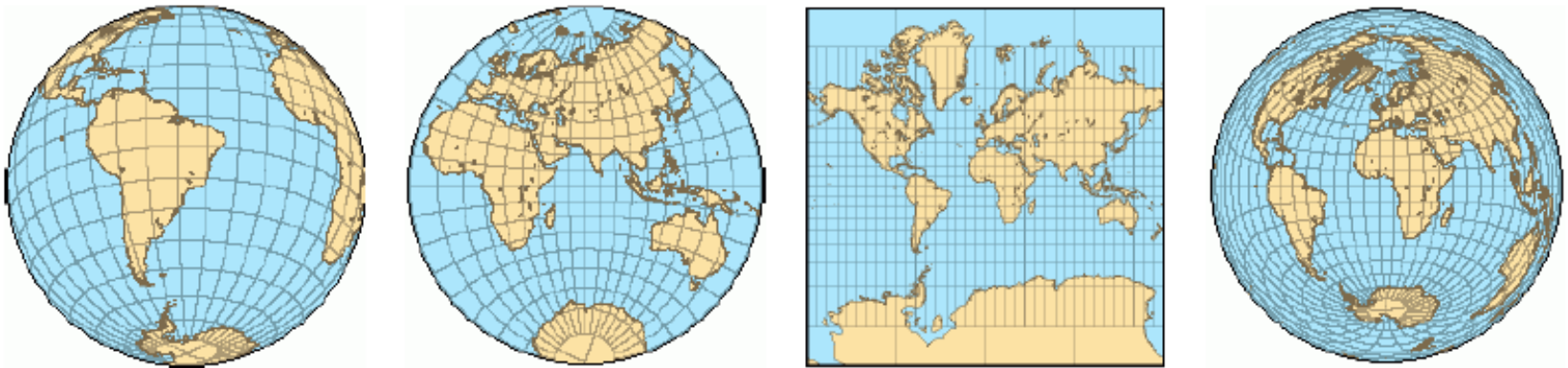
Équivalent à un **dépliage**



Pas toujours possible : **surface développable**
⇒ rendre le problème local

Fonction de placage

Problème bien connu en cartographie



⇒ **Distorsion** globale ou locale

⇒ Choix de conserver les angles, distances, ...

Solutions simples

$$f : (x,y,z) \rightarrow [0,1] \times [0,1]$$

Planaire : projection

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

Cylindrique

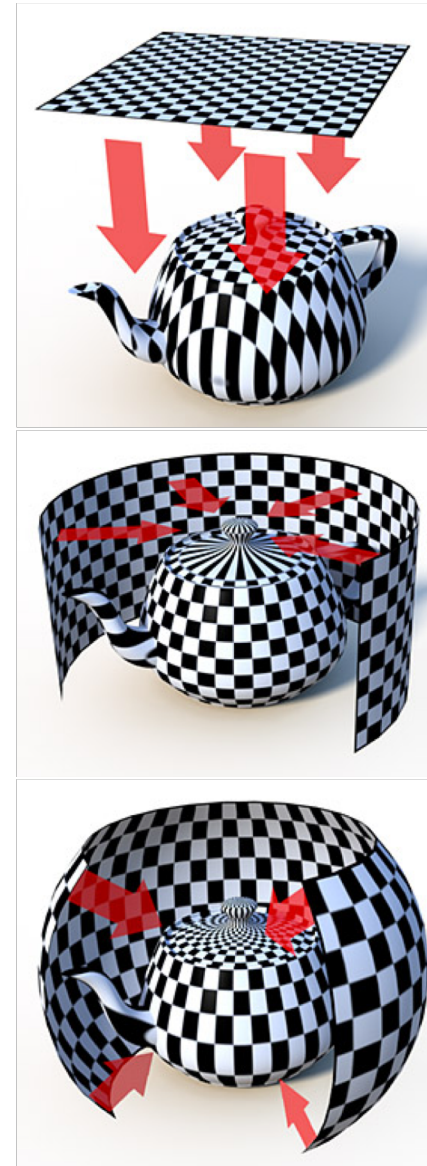
$$f(\theta,y) = (\theta/2\pi , y)$$

$$\text{avec } \theta = \text{atan}(z/x)$$

Sphérique

$$f(\theta,z) = (\theta/2\pi , \Phi/\pi)$$

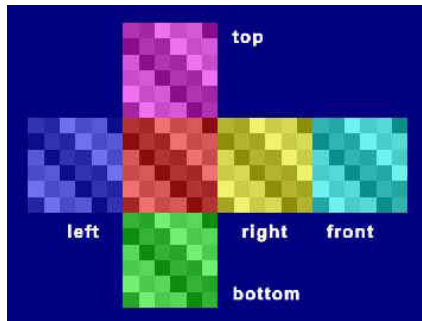
$$\text{avec } \theta = \text{atan}(z/x) \text{ et } \Phi = \text{asin}(y)$$



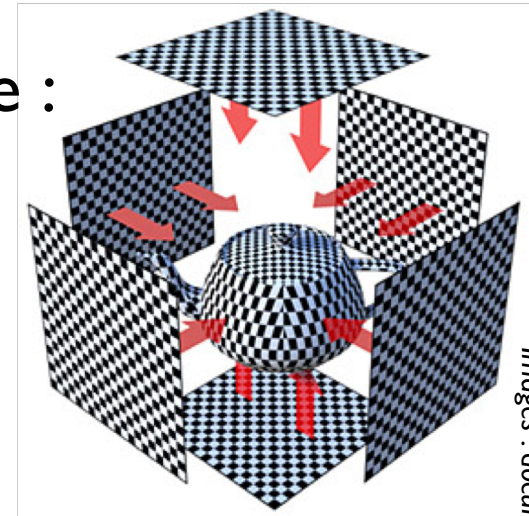
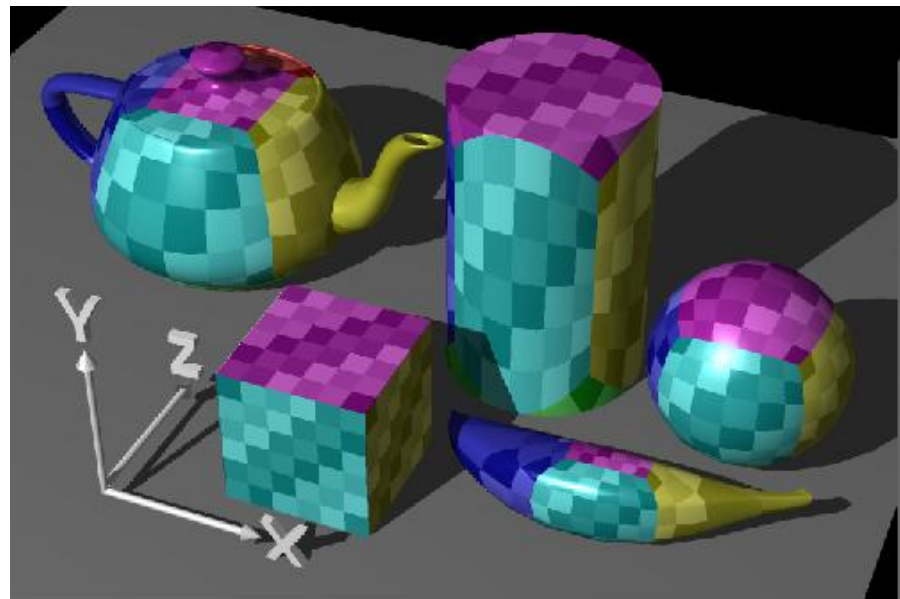
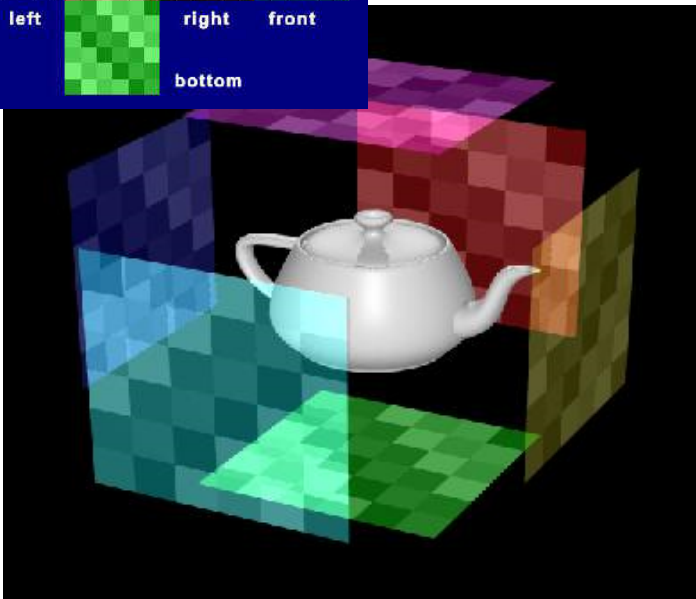
Solutions simples

Passer par un **objet simple** intermédiaire :

1. Dépliage
2. Projection

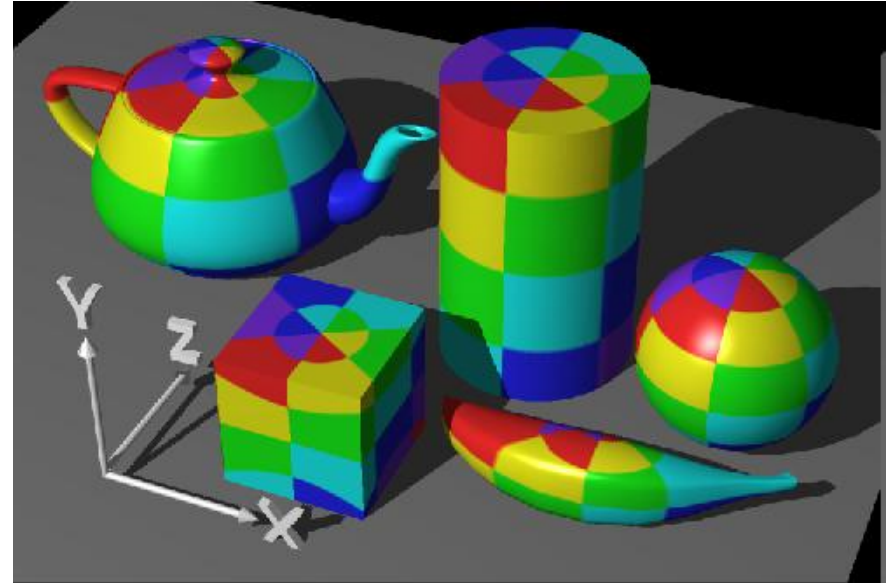
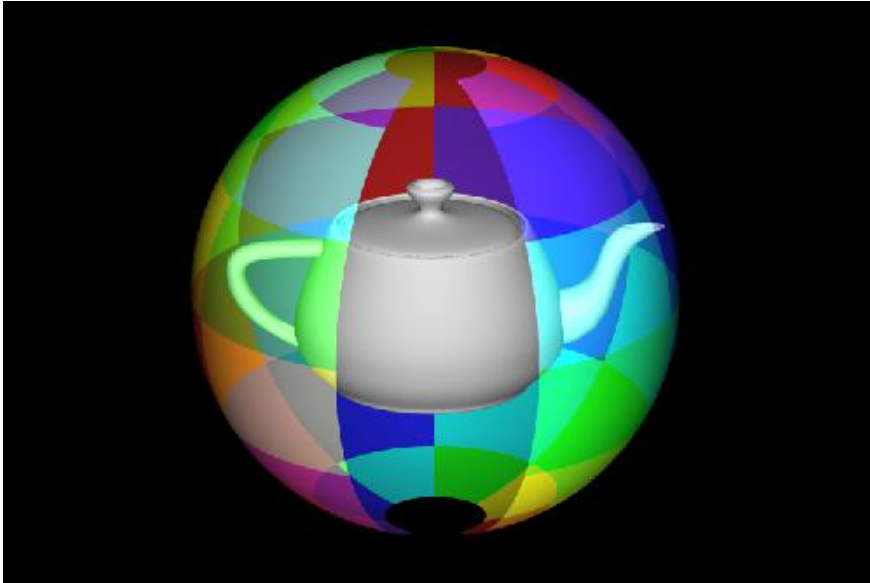


Cube mapping

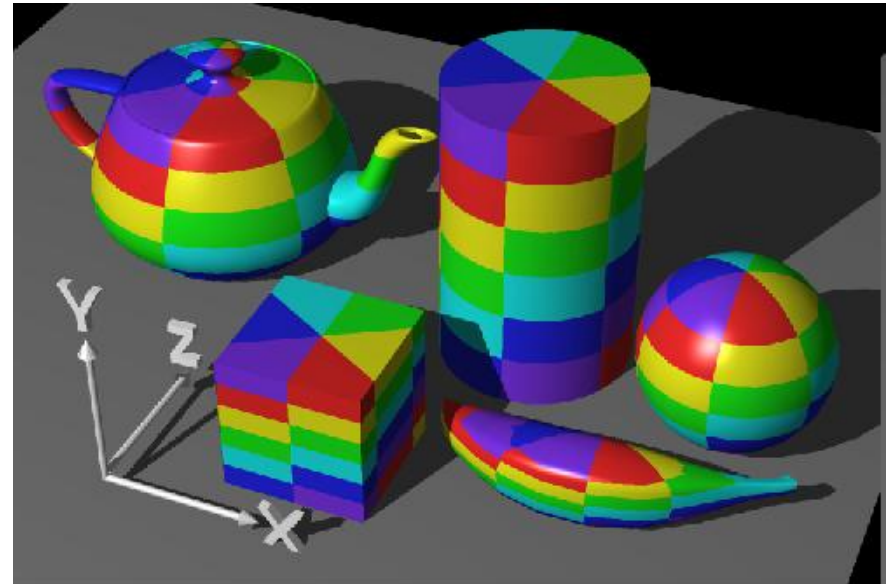


Images : documentation « modo »

Spherical mapping



Cylindrical mapping



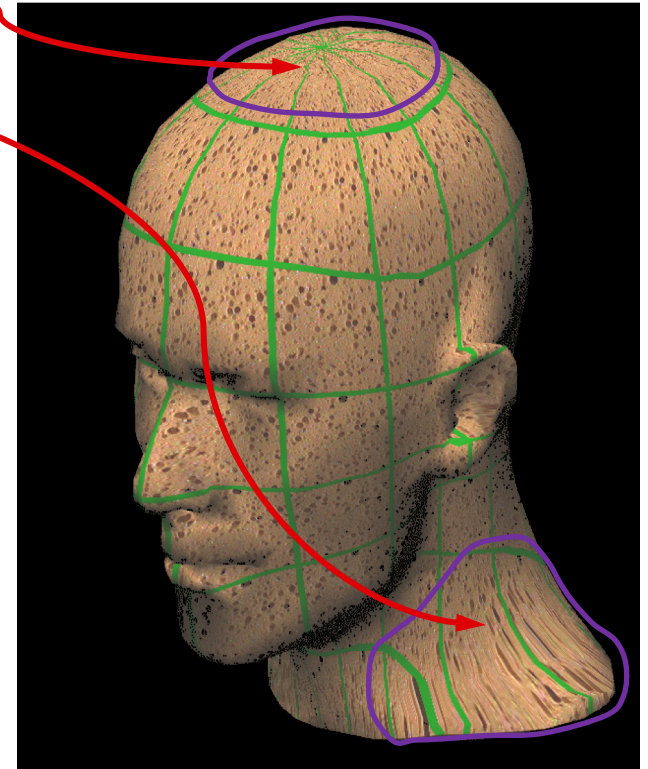
Cas général

Problème d'optimisation de la paramétrisation pour minimiser

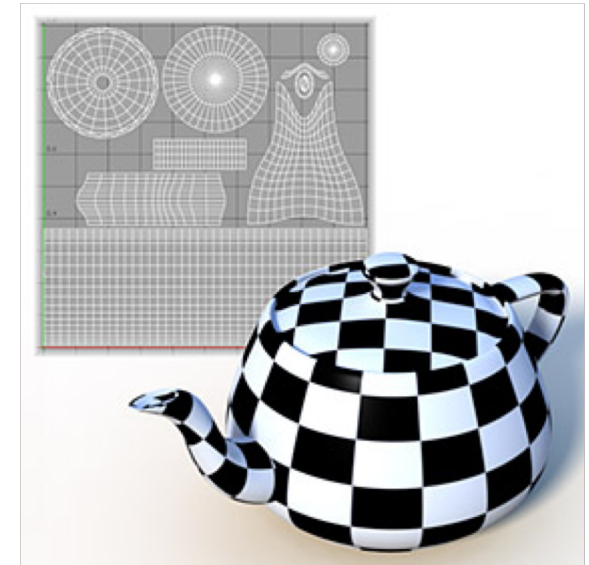
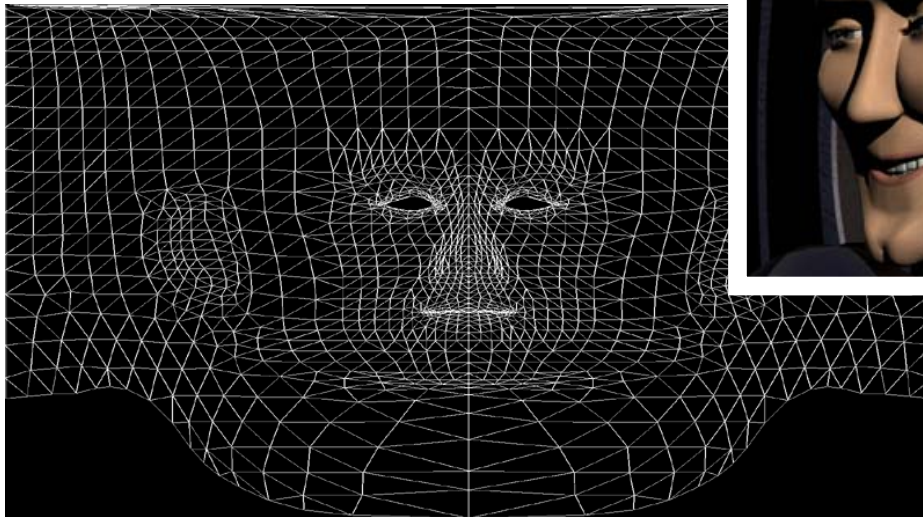
- les **singularités** (pôles)
- les **distorsions**

⇒ Réalisée à la main par les artistes

⇒ Techniques automatiques



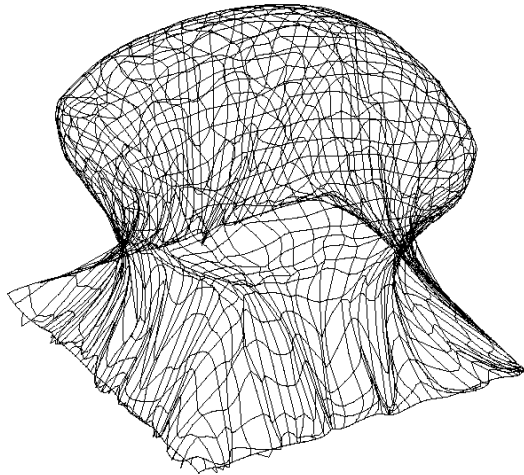
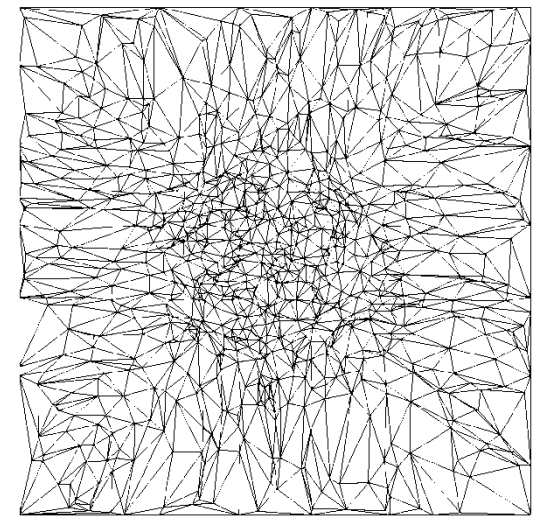
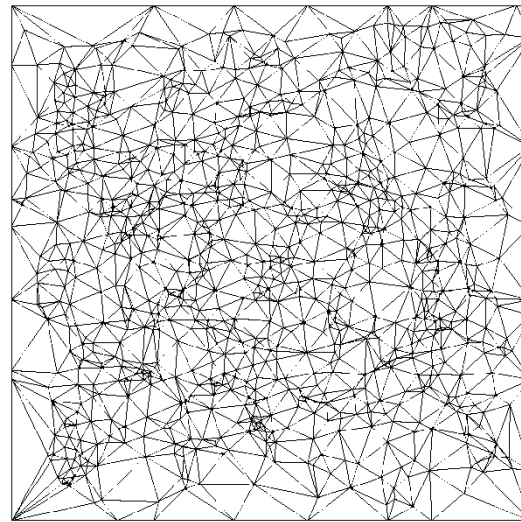
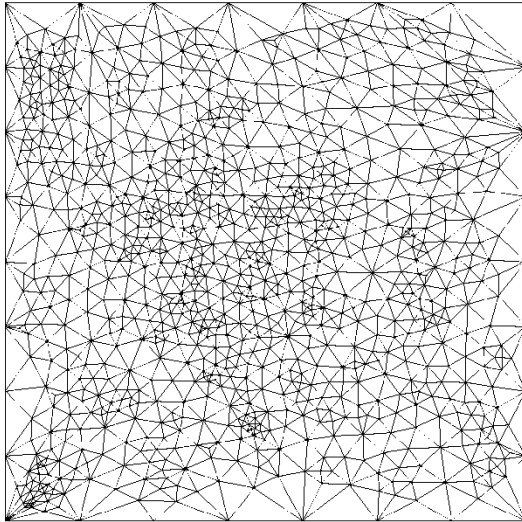
Paramétrisation manuelle



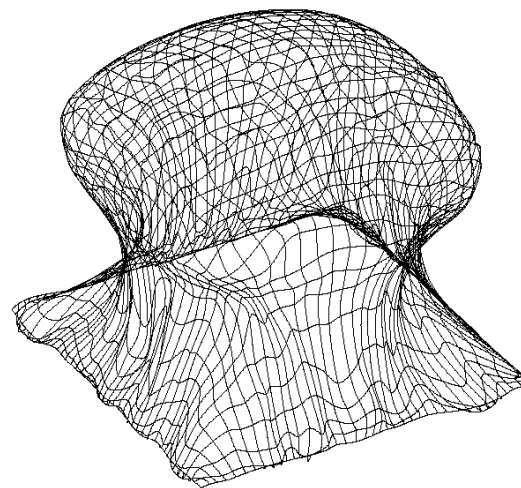
documentation « modo »

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

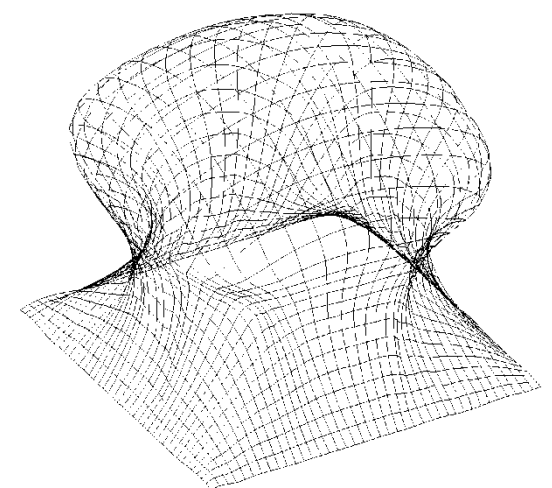
Exemples d'optimisations automatiques



Iso-barycentre



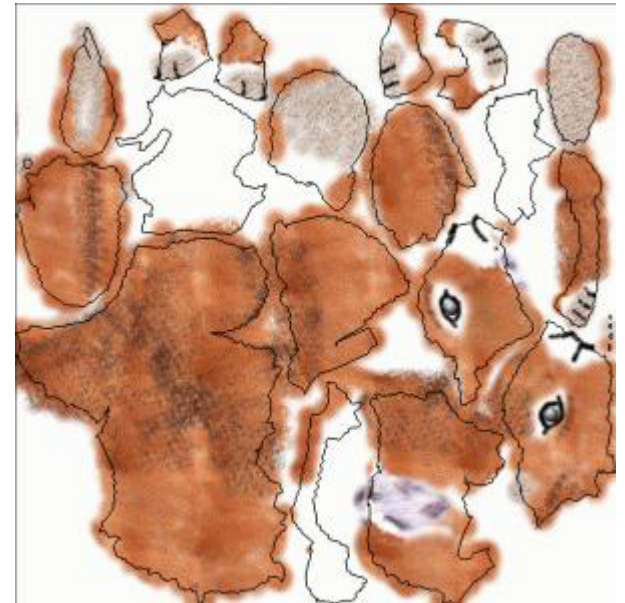
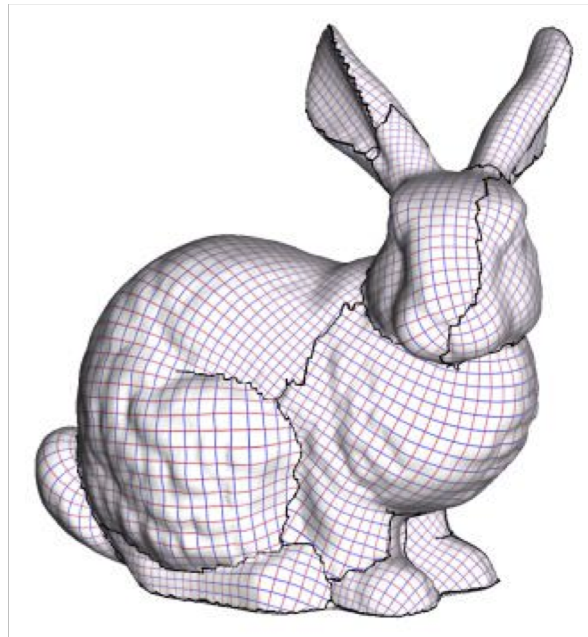
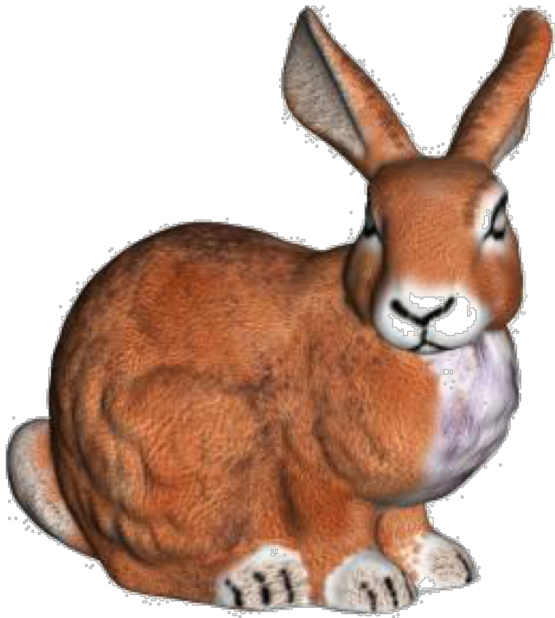
**Conservation
des distances**



Barycentre (3 pts)

Atlas de texture

Décomposition en différents patches



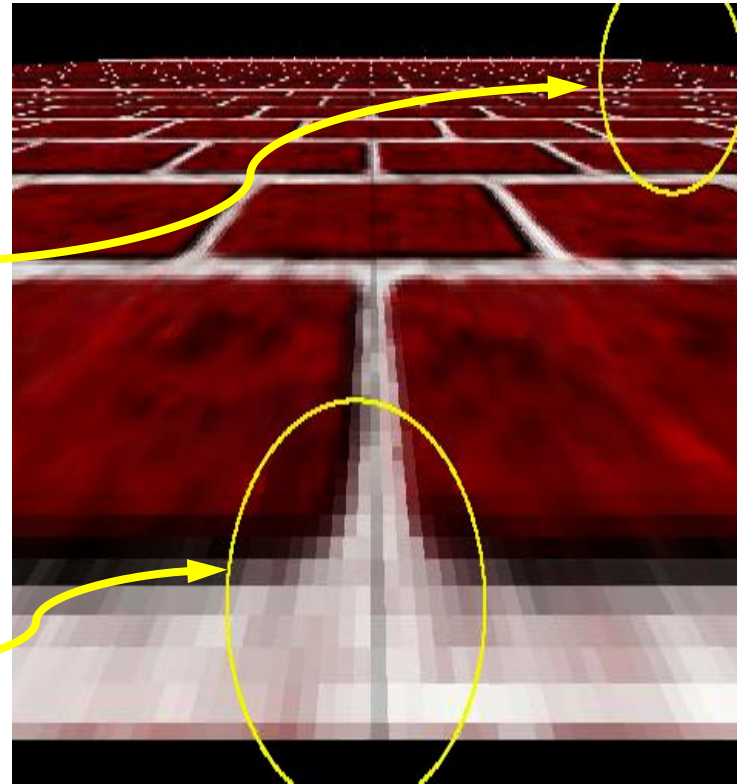
[Lévy et al. 2002]

Problèmes

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

Plusieurs couleurs pour un pixels

Pixels visibles



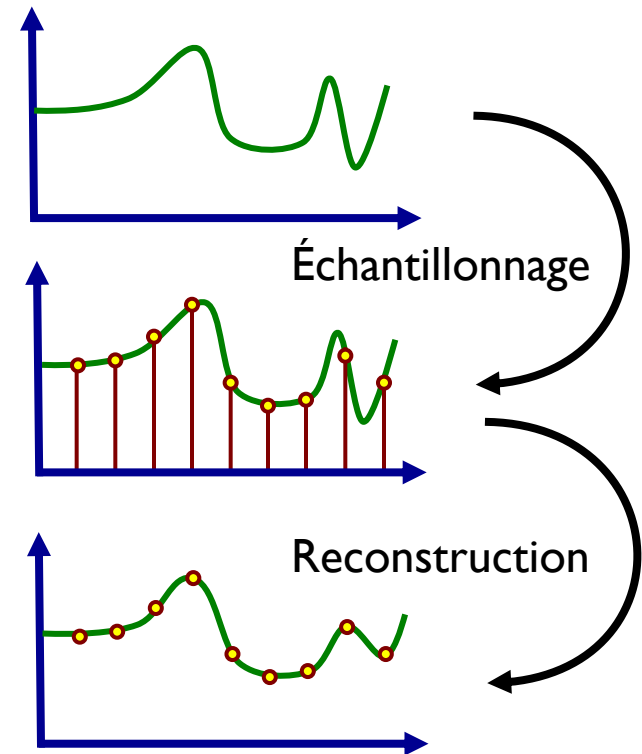
Échantillonnage

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

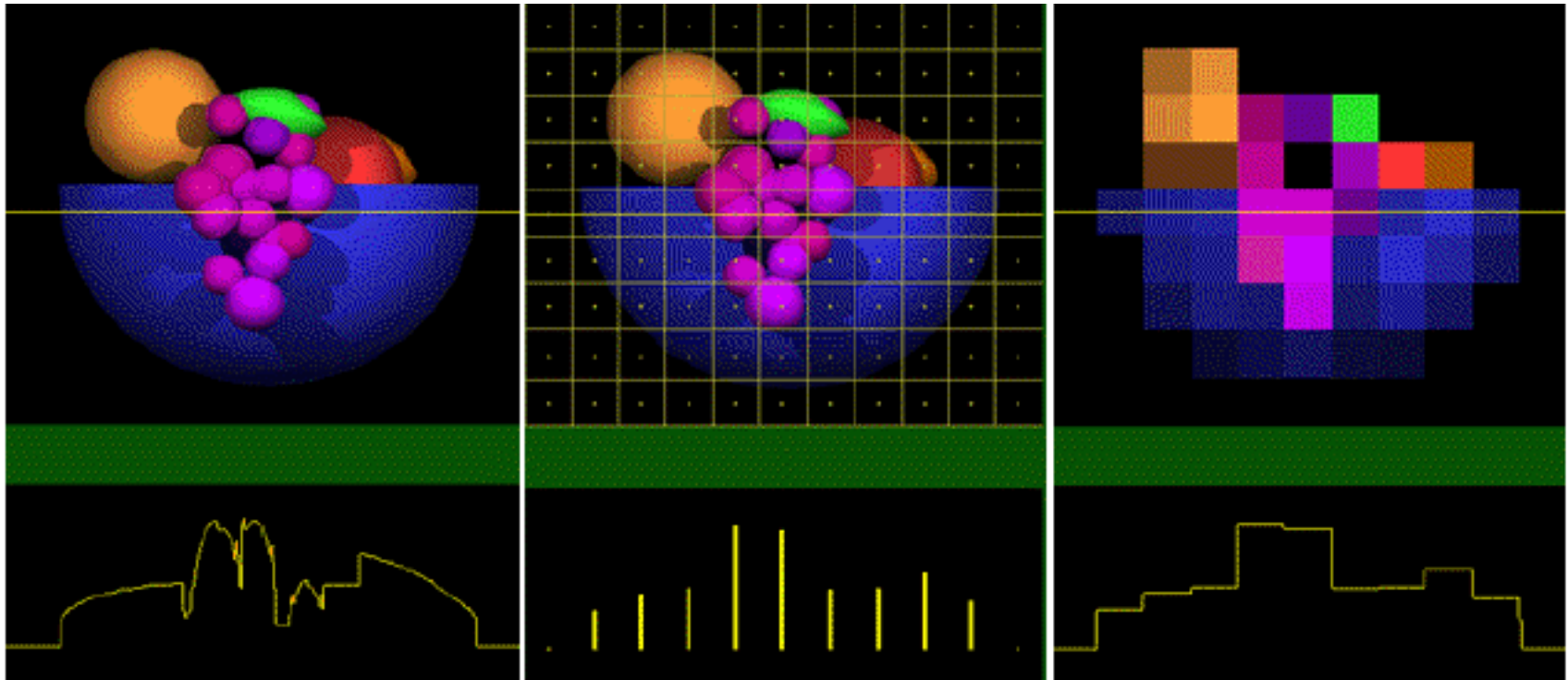
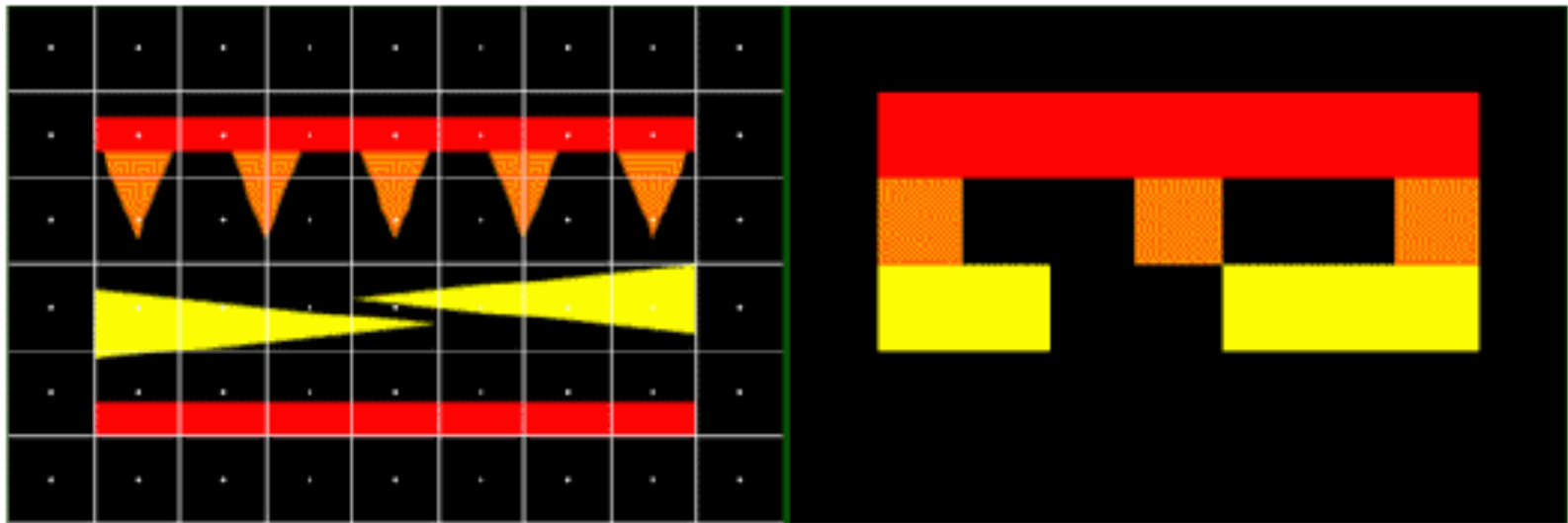


Image d'origine

Échantillons

Reconstruction

Aliasing – perte de détails

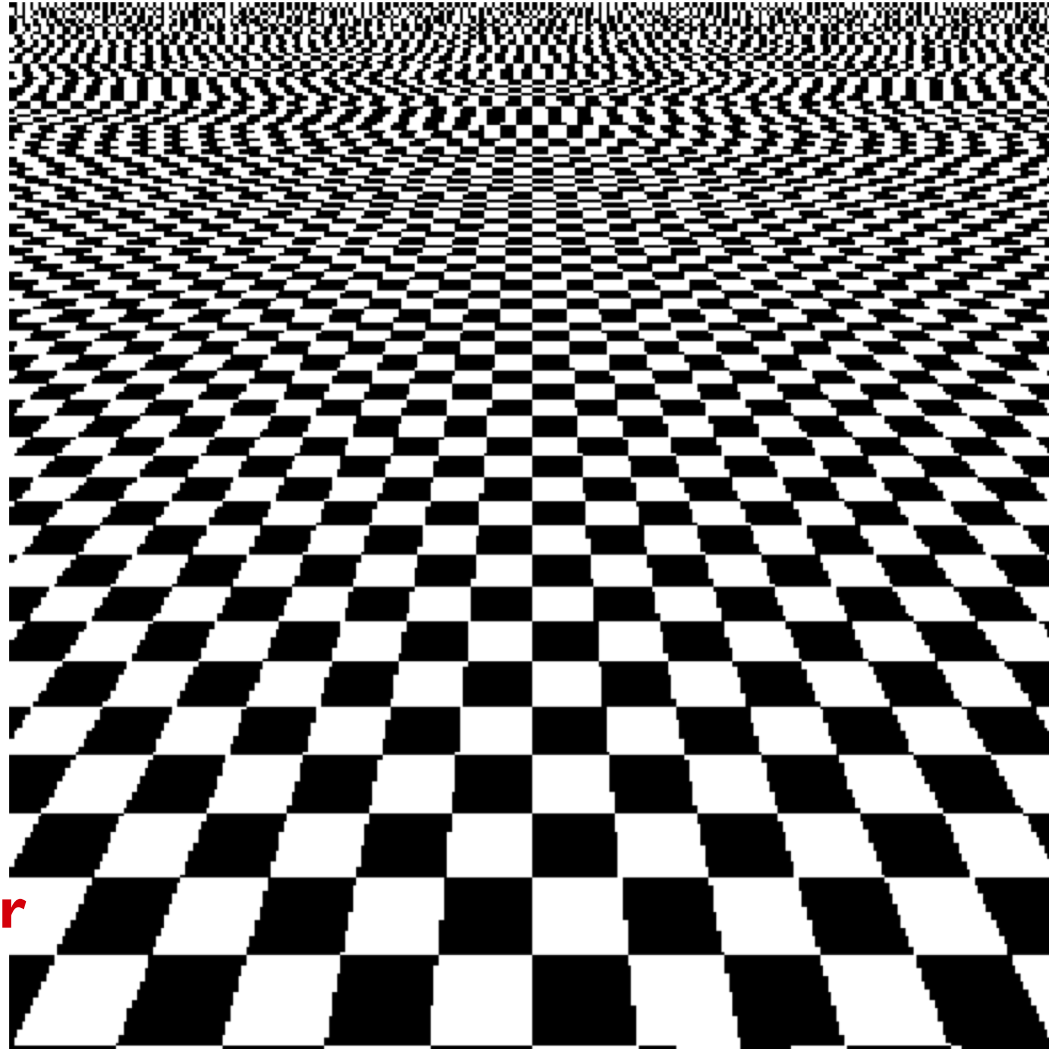


Échantillons

Reconstruction

Aliasing = effet d'escalier + moiré

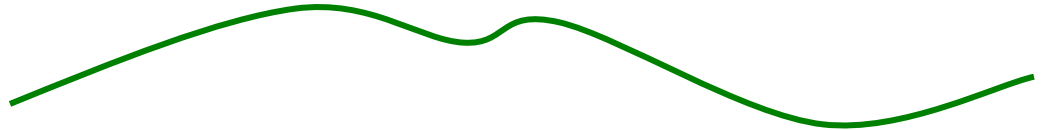
Moiré



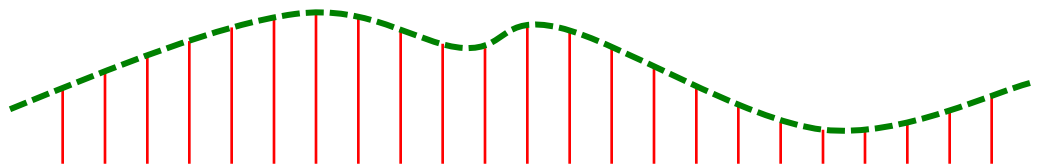
Effet d'escalier

Signal basse fréquence

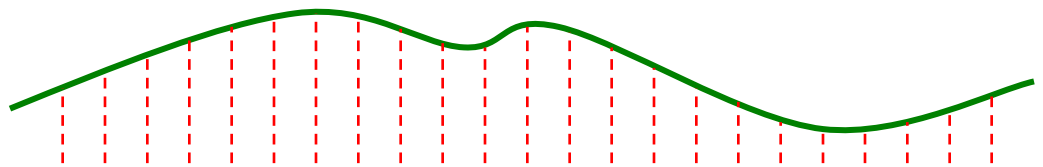
Signal original



Échantillonnage
haute fréquence

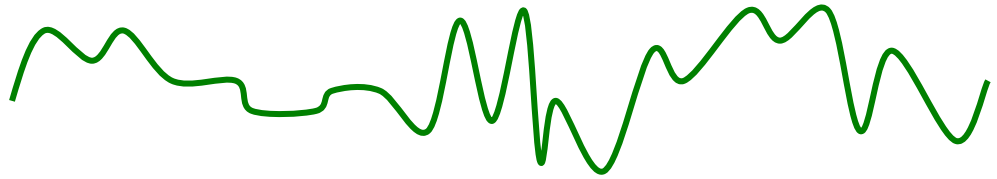


Signal reconstruit

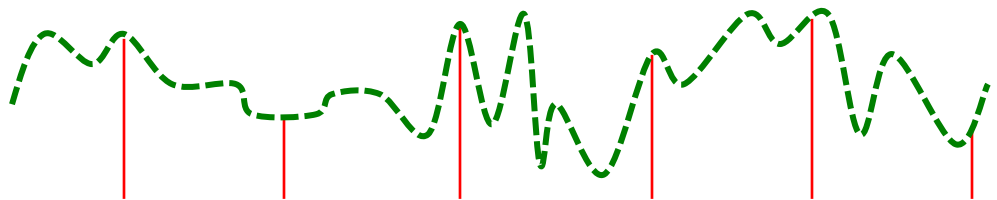


Signal haute fréquence

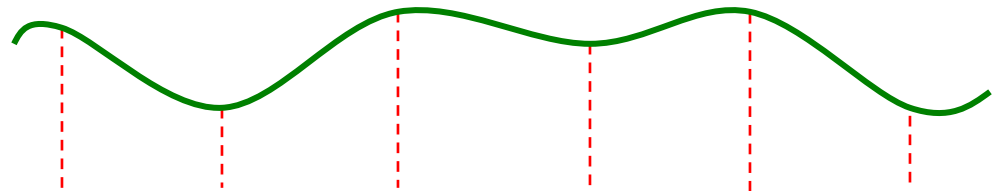
Signal original



Échantillonnage
basse fréquence

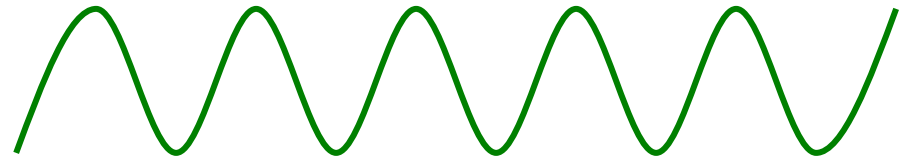


Signal reconstruit

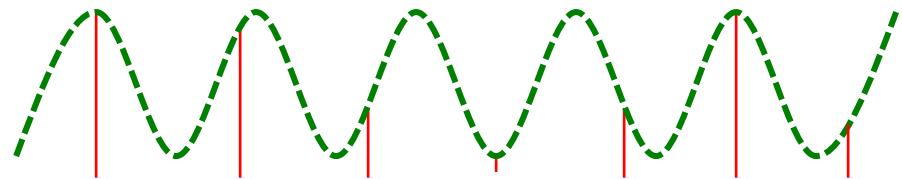


Signaux périodiques

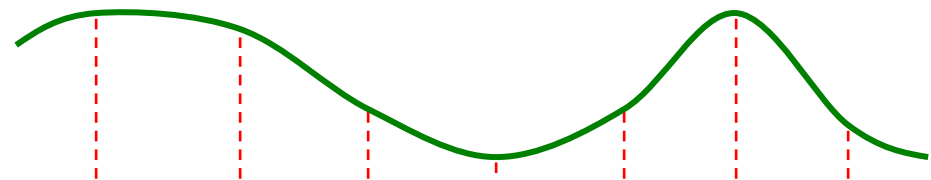
Signal original



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 $2f$

⇒ **fréquence de Nyquist**

Valide pour les signaux à **support fréquentiel fini**.

Problème :

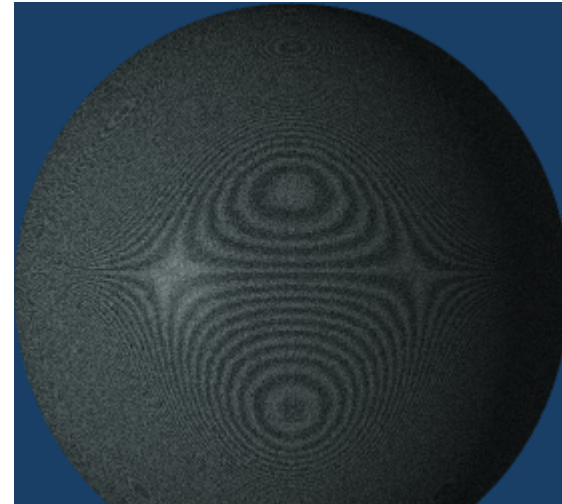
la plupart des images comportent des **discontinuités** (lignes, ombres dures, textures, taches spéculaires, etc.)

⇒ **fréquences infinies !**

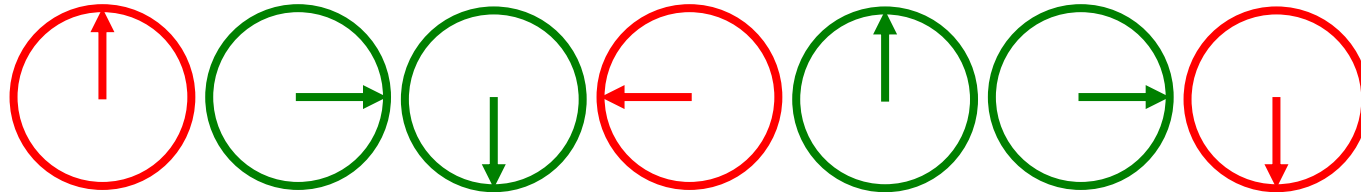
Aliasing

Spatial

- Effets d'escalier, Moiré, etc.



Temporel



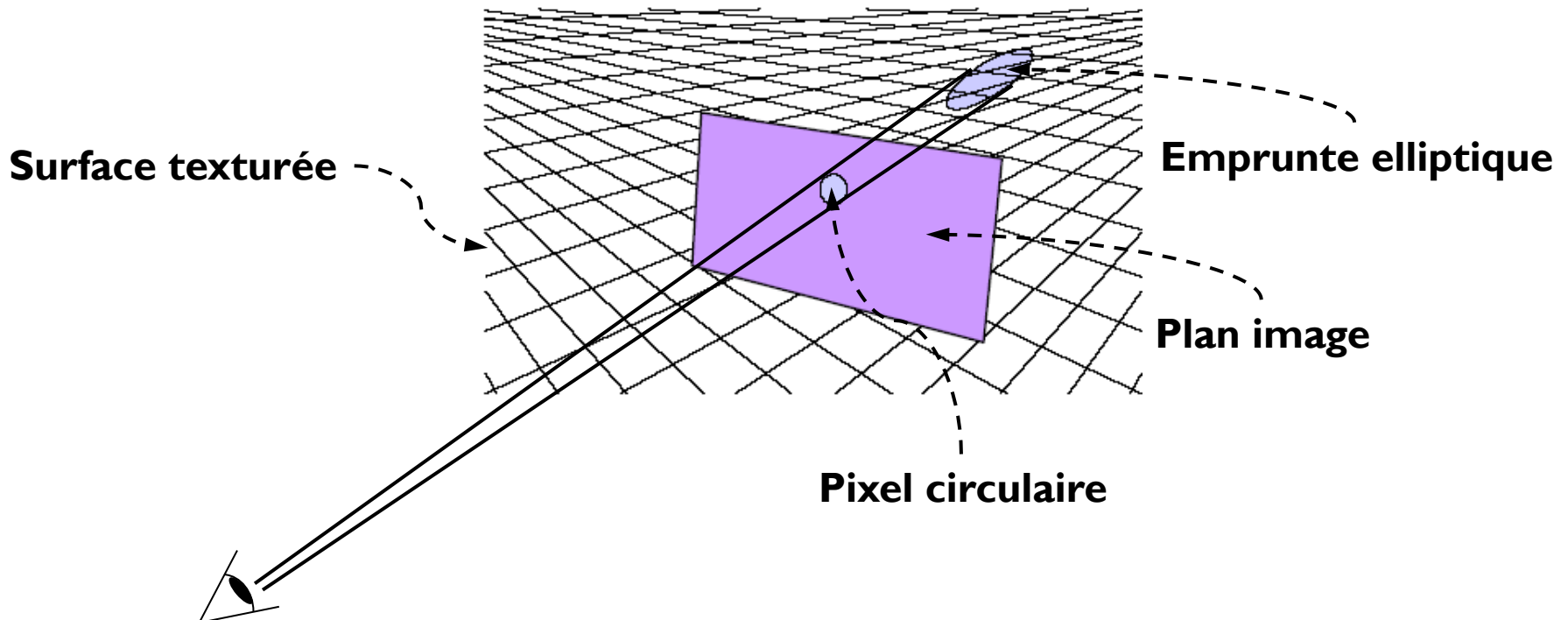
Antialiasing

Théorie :

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

⇒ Très couteux

En pratique : **approximations !**

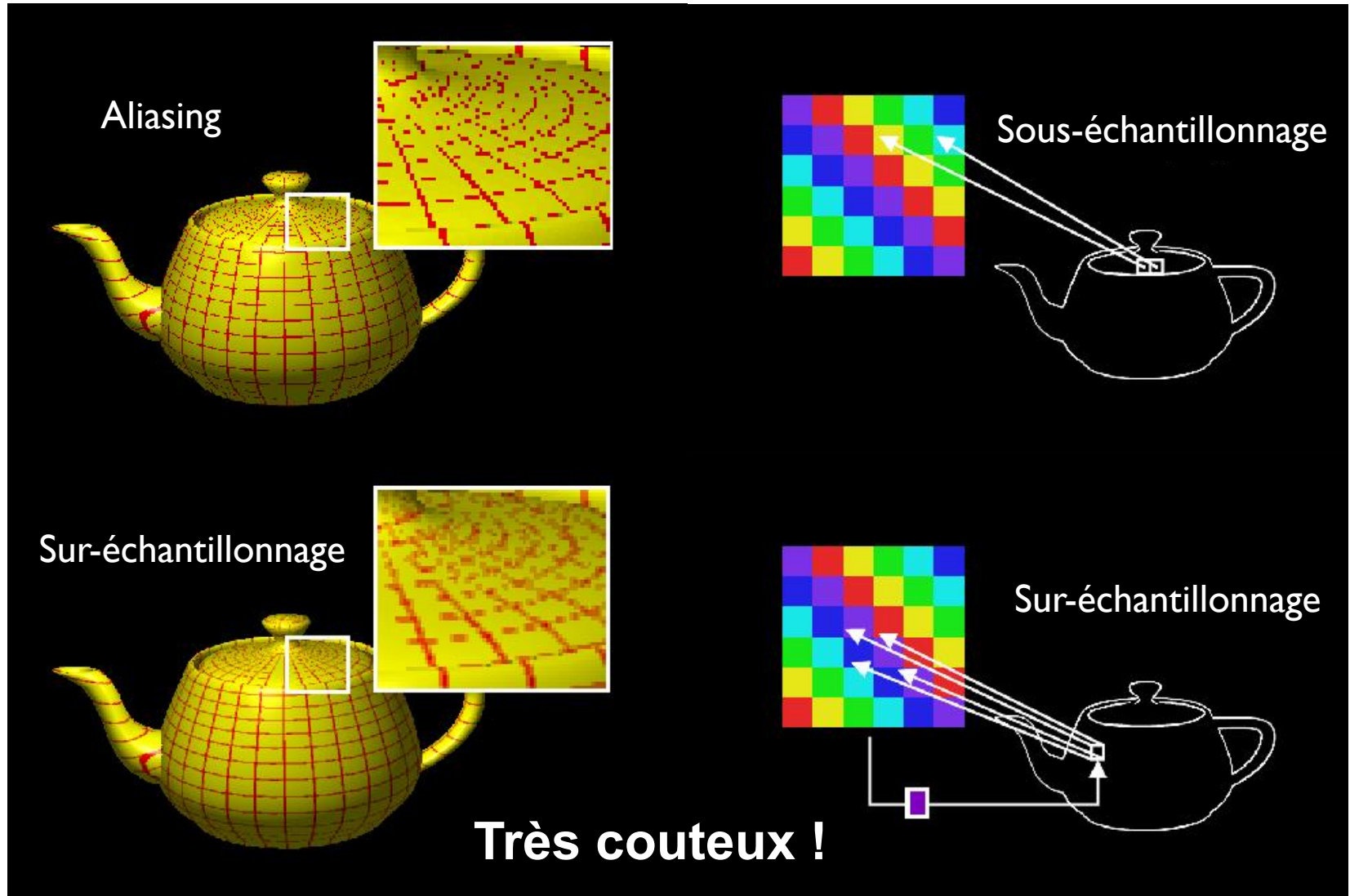


Antialiasing

Échantillonner à **plus haute fréquence**

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

Sur-échantillonnage



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*)
- Compromis entre flou et aliasing

Pré-filtrage

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

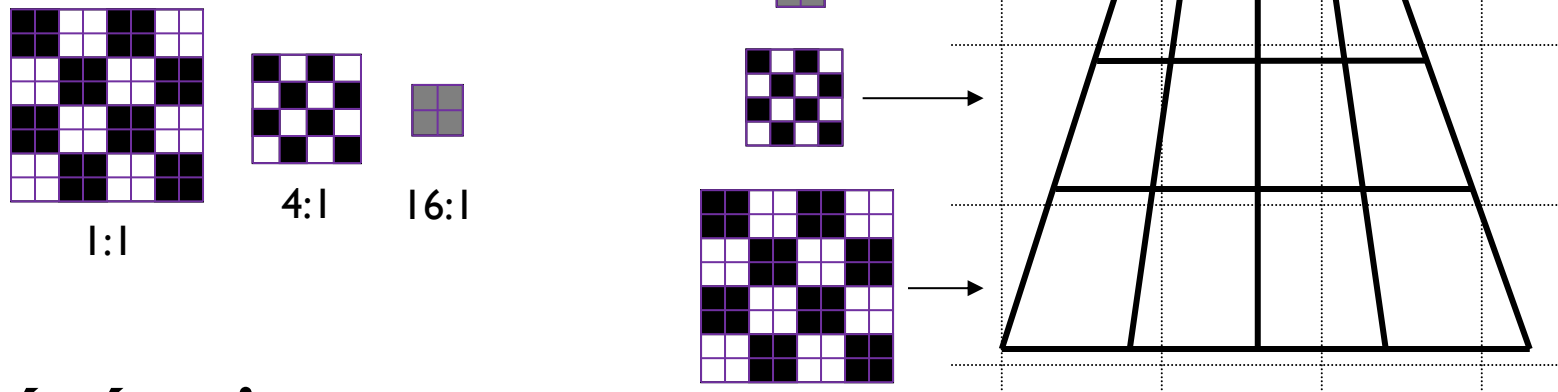
Intégration (convolution) **variant spatialement** en chaque pixel

⇒ **trop coûteux**

⇒ en pré-calculer une **approximation**

MIP Mapping [Williams 1983]

Pyramide d'images pré-filtrées



Génération

- automatique
`GL_GENERATE_MIPMAP`, `glGenerateMipmap()`
- manuelle \Rightarrow choix du filtre

MIP Mapping [Williams 1983]

Sélection du niveau basée sur le **gradient** des coordonnées de texture par fragment

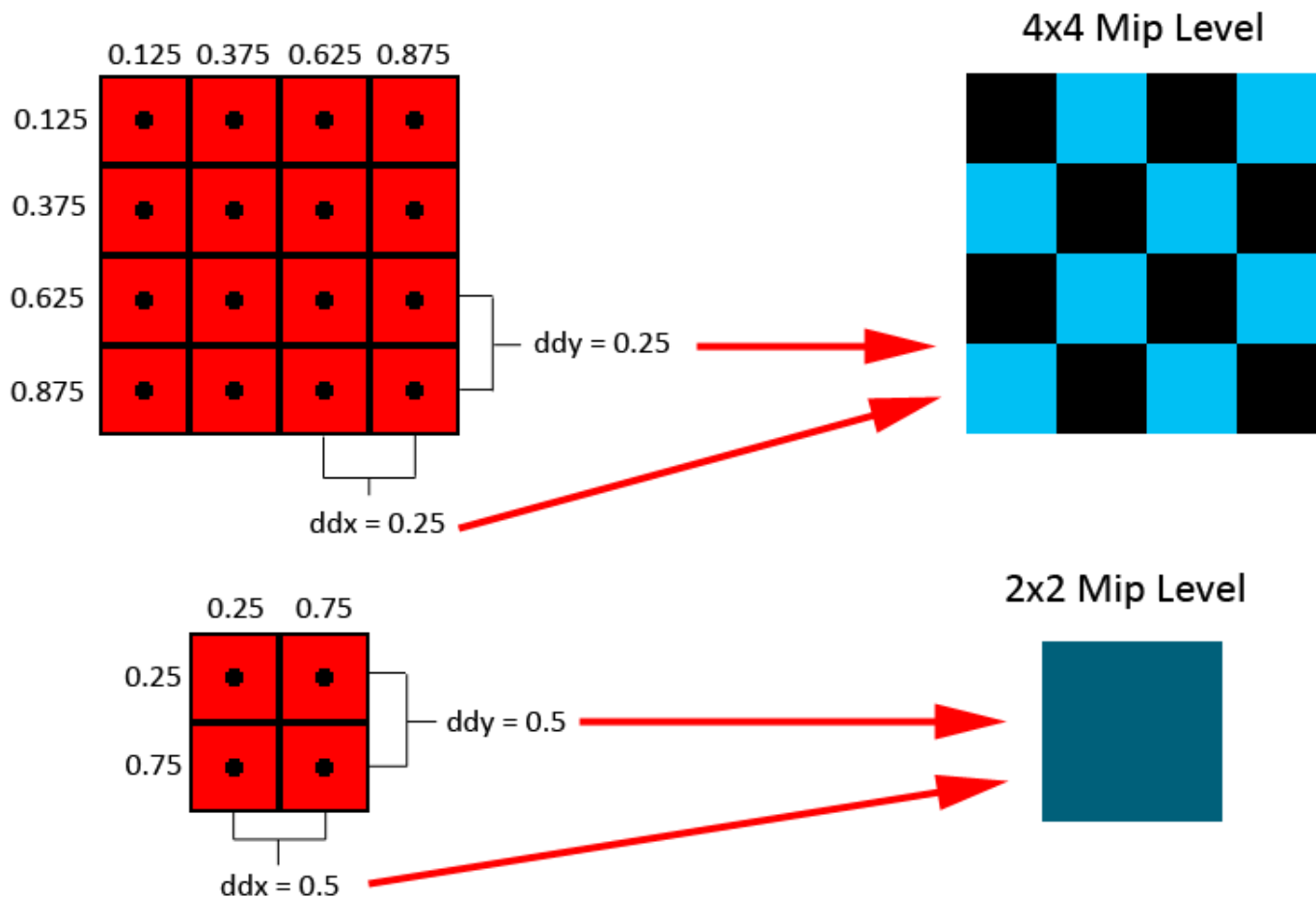
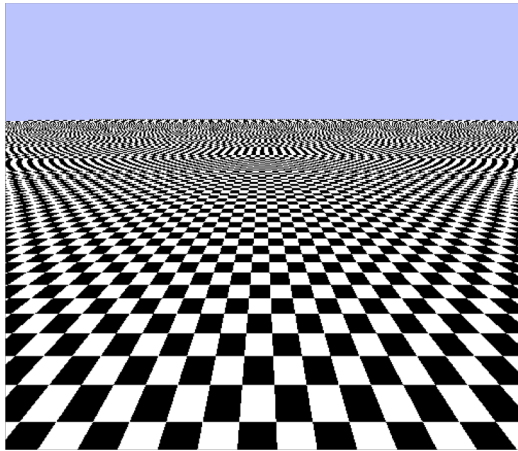


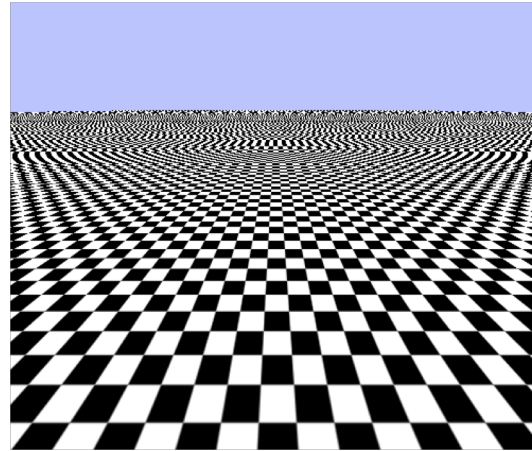
Image : <https://mynameismip.wordpress.com/2012/11/02/applying-sampling-theory-to-real-time-graphics/>

MIP Mapping [Williams 1983]

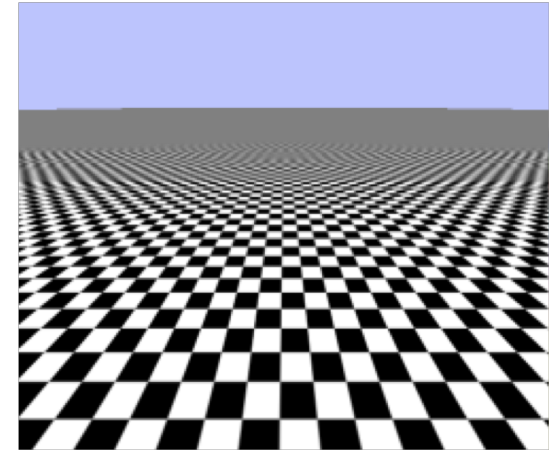
Filtrage lors des accès



Plus proche voisin
`GL_NEAREST`



Filtrage linéaire
`GL_LINEAR`

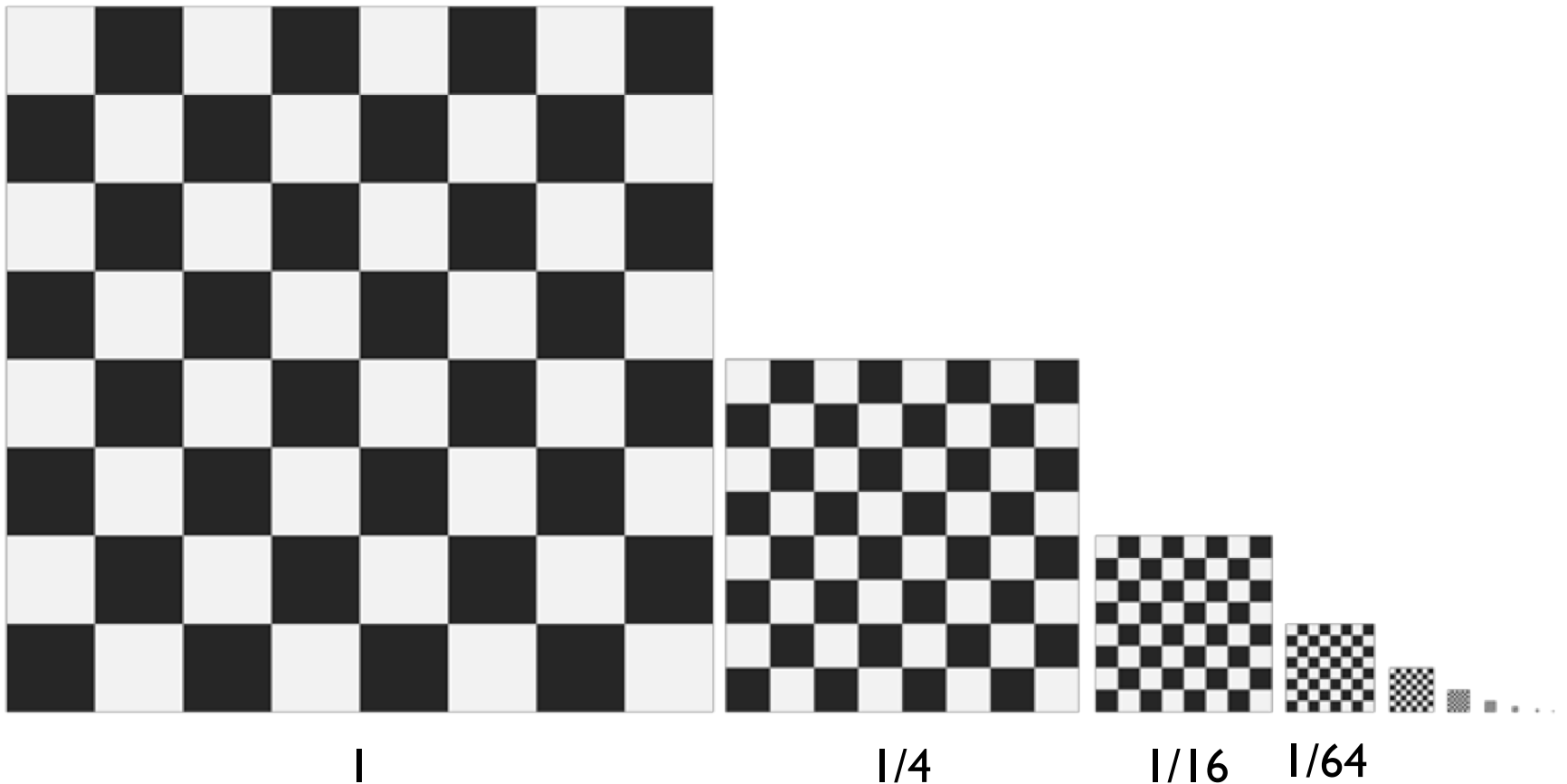


MIP mapping
`GL_LINEAR_MIPMAP_LINEAR`

MIP Mapping [Williams 1983]

Surcoût en mémoire de 33% (si carrée et puissance de 2)

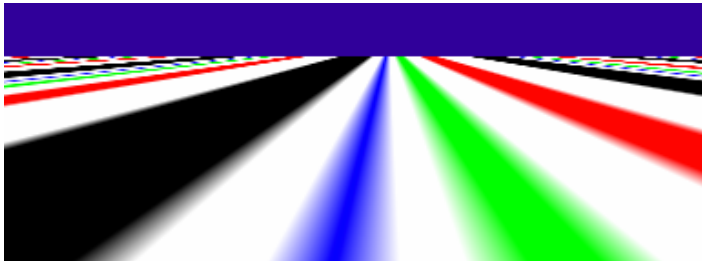
$$1/4 + 1/16 + 1/64 + 1/256... \rightarrow 1/3$$



MIP Mapping

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

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



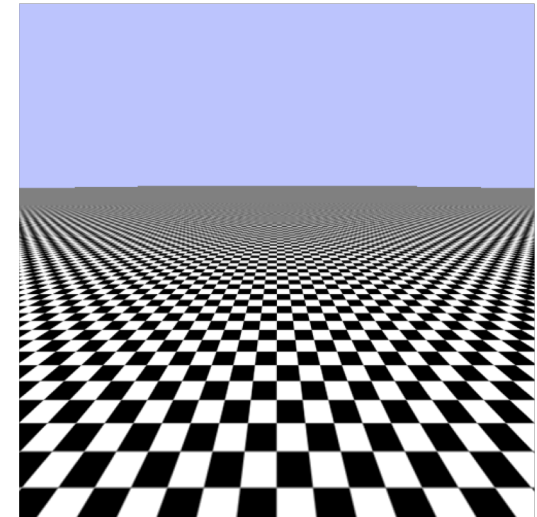
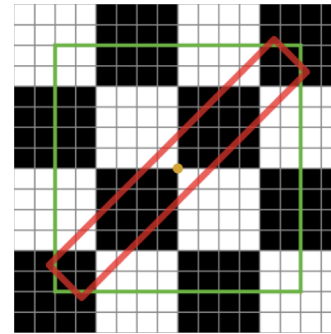
Plus proche voisin



MIP Mapping

Solutions (plus coûteuses)

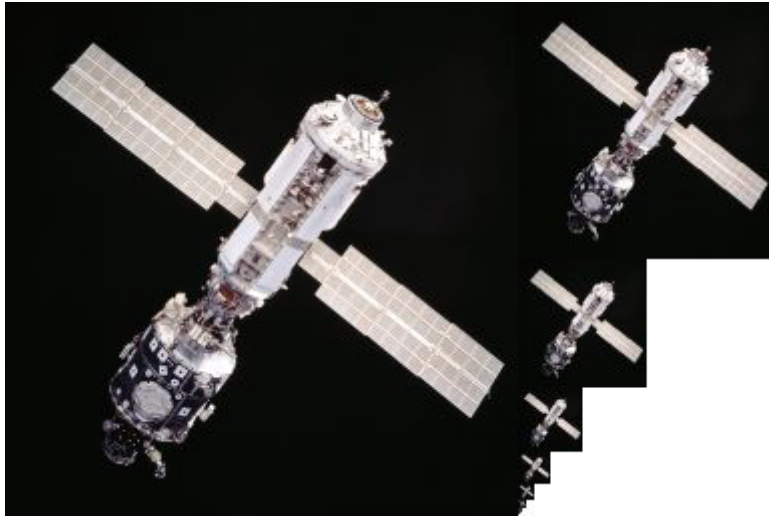
- Filtrage anisotrope matériel, mais « brute-force »
- *Elliptical weighted average* calcul de empreinte elliptique du pixel
- *RIP maps*



Filtrage anisotrope

RIP maps

4 fois plus coûteux à stocker...



MIP map



RIP map

Problèmes

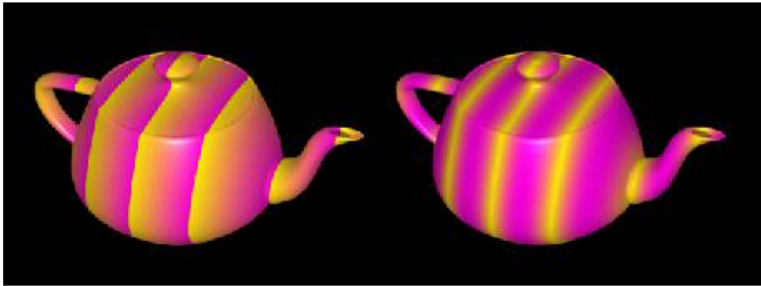
1. Plaquer une texture sans distorsion
2. Réduire le crénelage, l'*aliasing*
3. **Synthétiser** une texture



par Hugo Beyer, créés avec *Substance Allegorithmic*

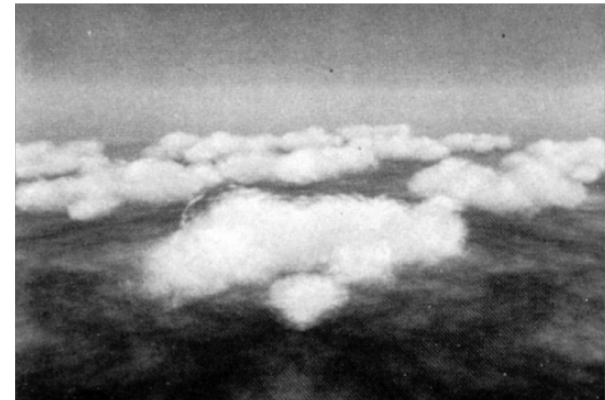
Textures procédurales

Résultat d'une **fonction mathématique**
ou d'un **programme** + *color map*

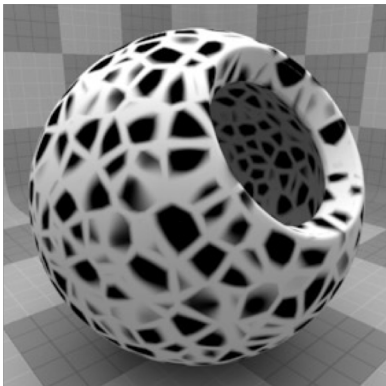


$\text{mod}(x,a)/a$

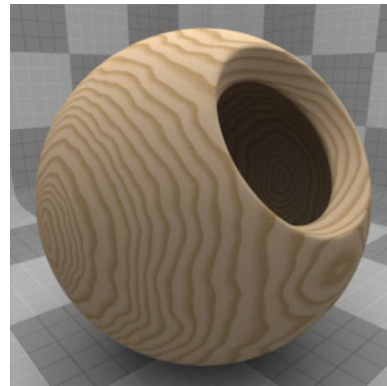
$(\sin(x)+1)/2$



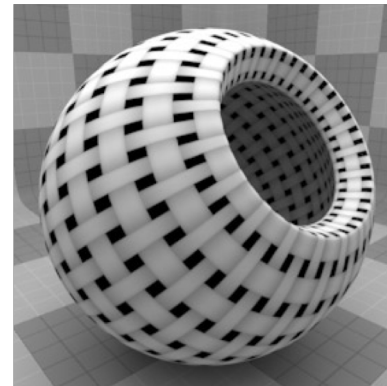
[Gardner 1985]



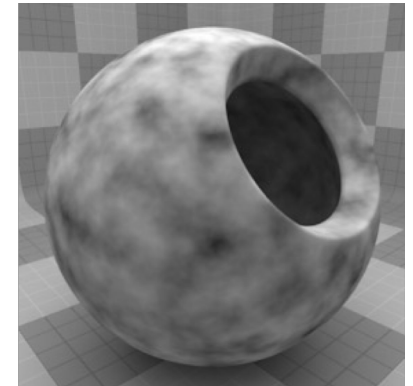
Texture cellulaire
[Worley 1996]



Texture de bois

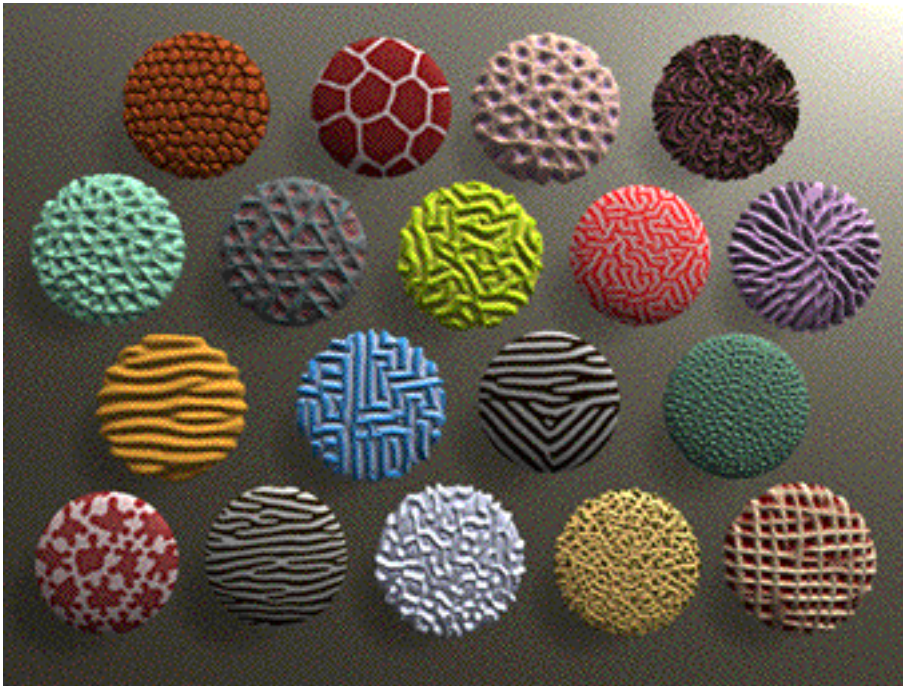


Texture de tissage

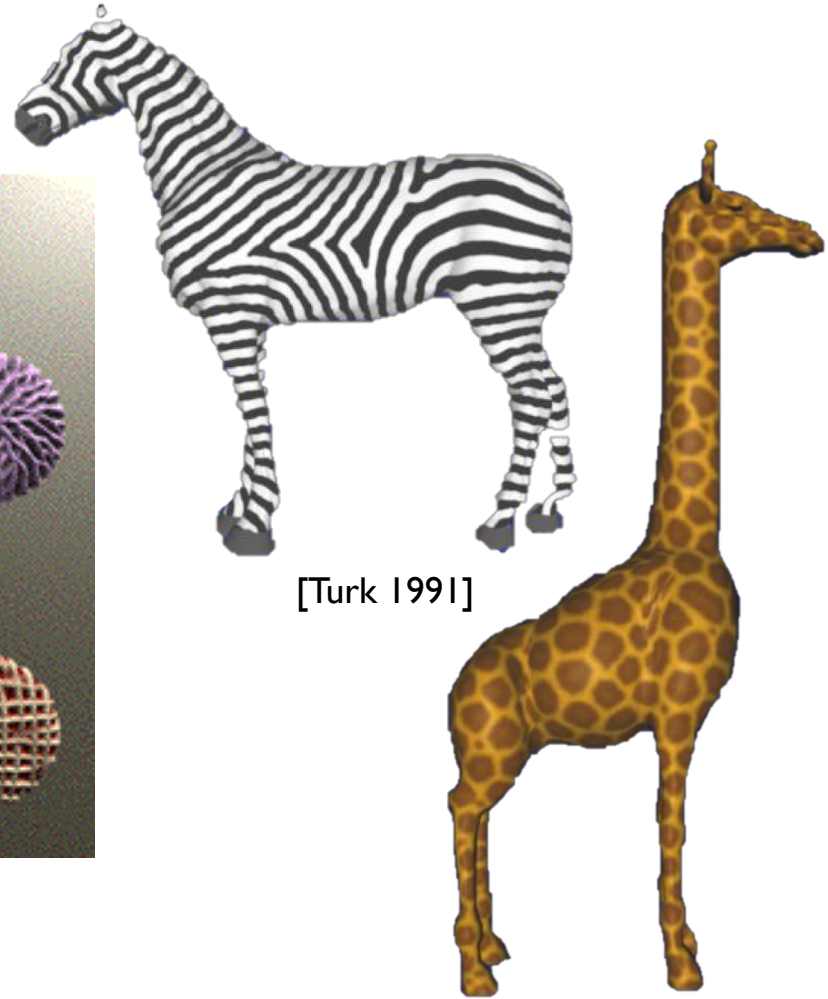


Texture de bruit

Réaction-diffusion



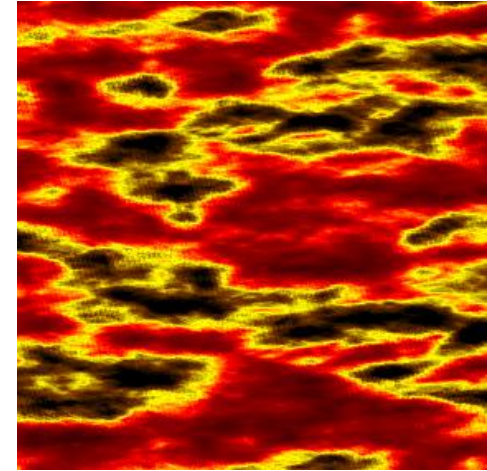
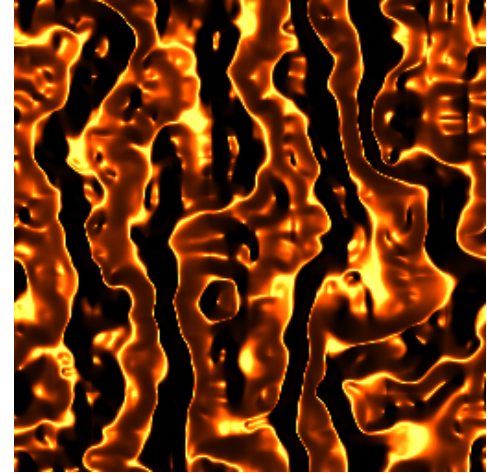
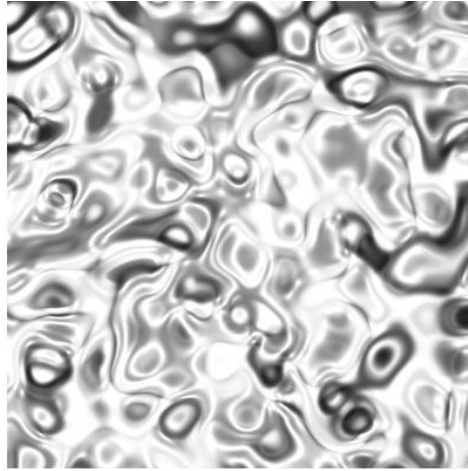
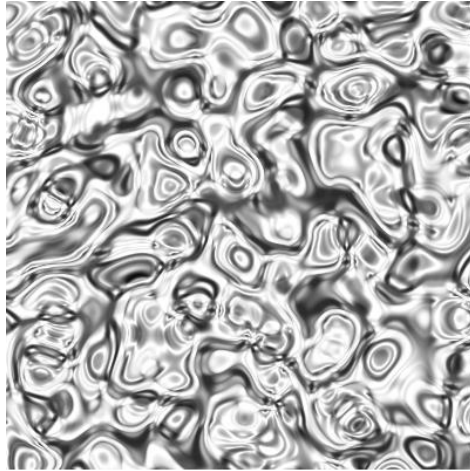
[Kass & Witkin 1991]



[Turk 1991]

Textures de Perlin

Bruit fractal continu + *color map*



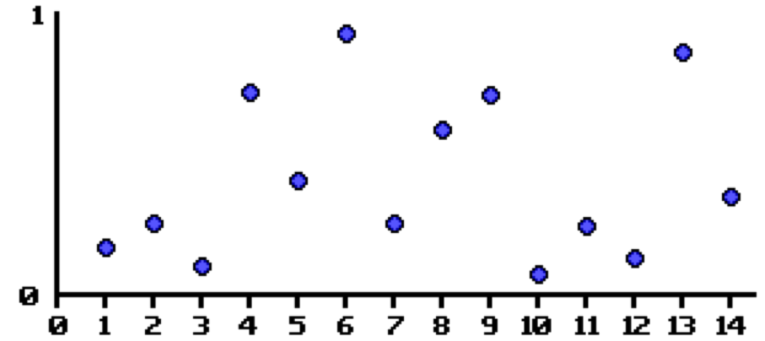
Fonction de bruit

Fonctions de base (1D)

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

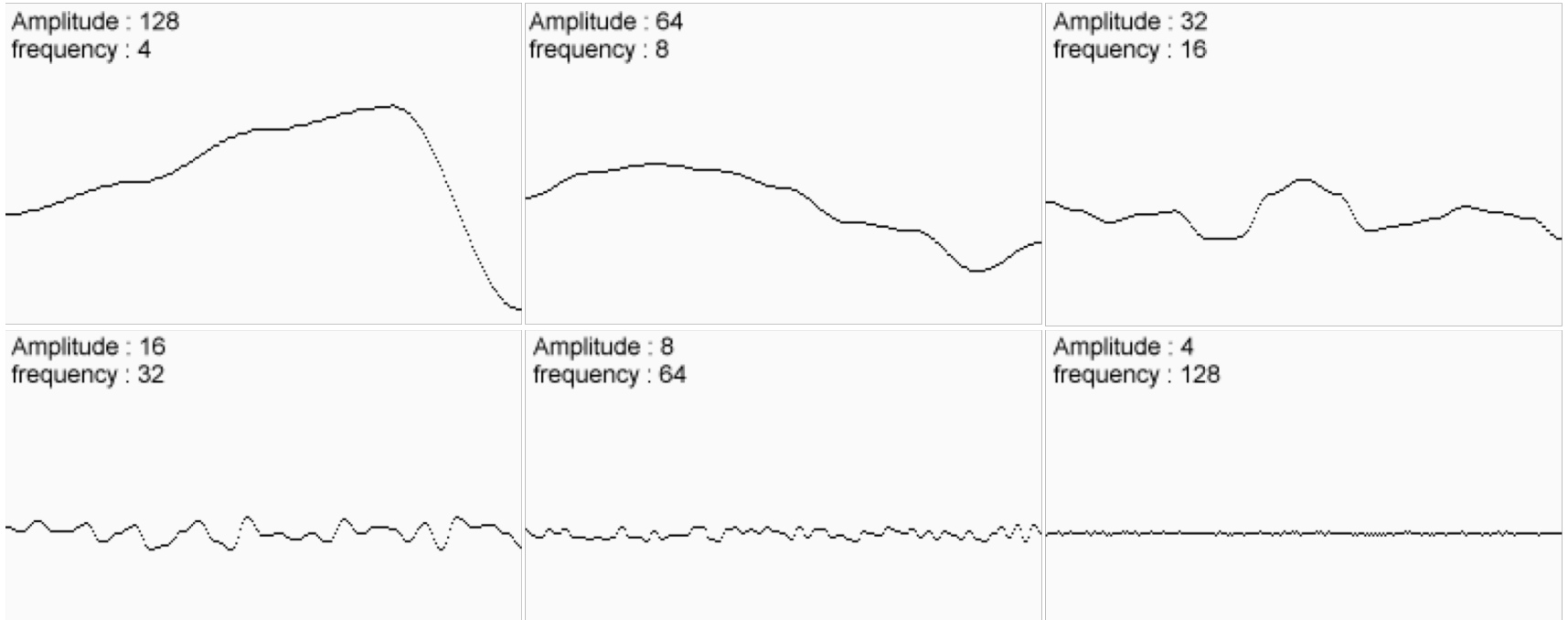
Pré-calcul des valeurs (tableau 1D)

Perlin : interpolation de gradients aléatoires unitaires

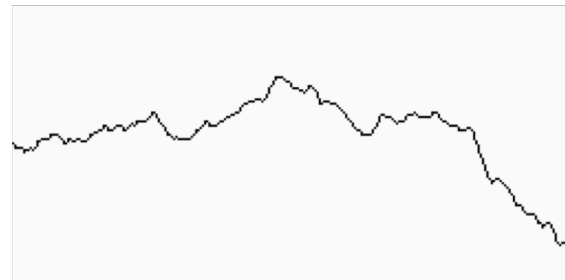


Bruit turbulent

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

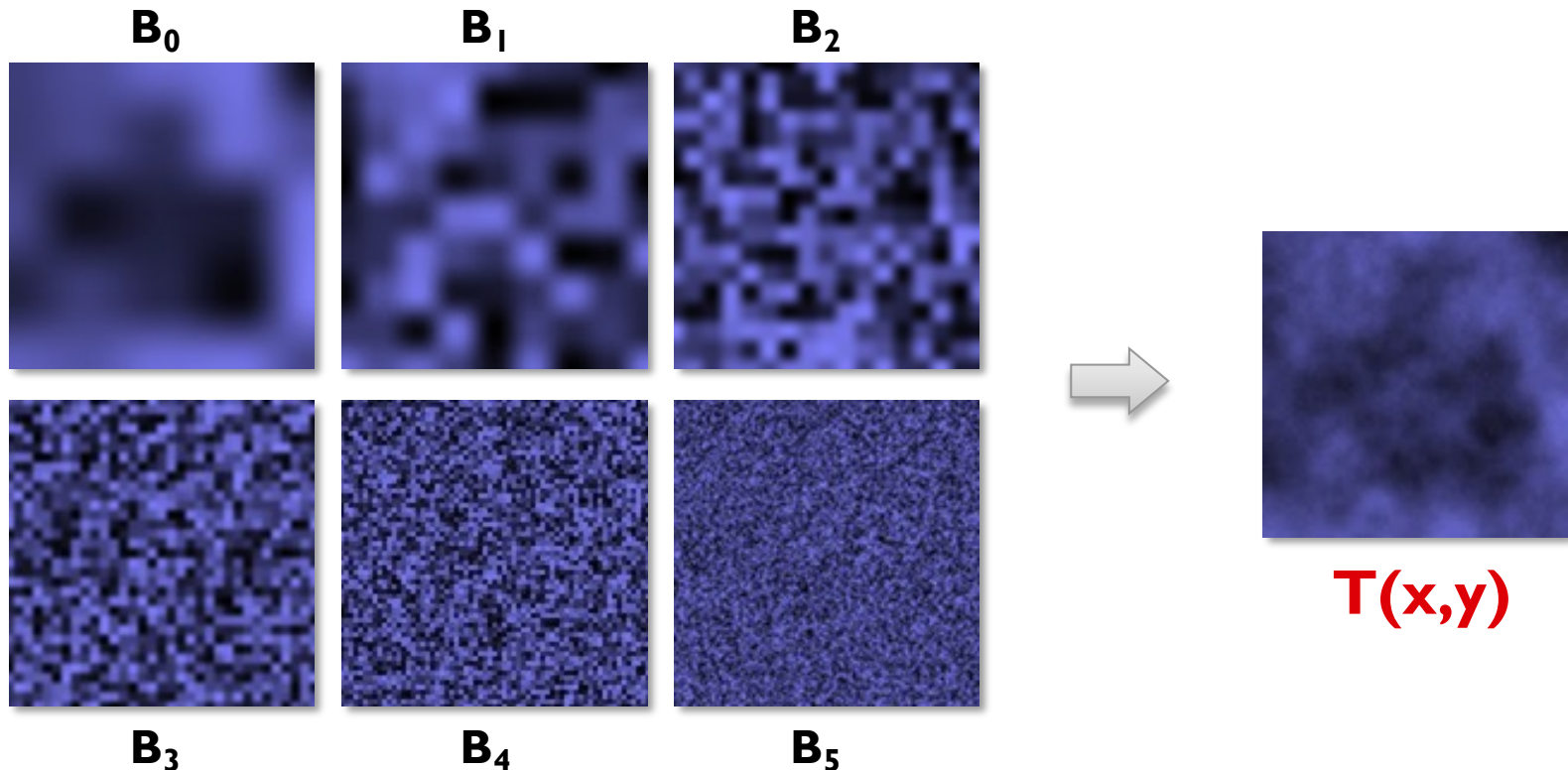


$$\Rightarrow T(x) = \sum 1/2^i B(2^i x)$$



Texture

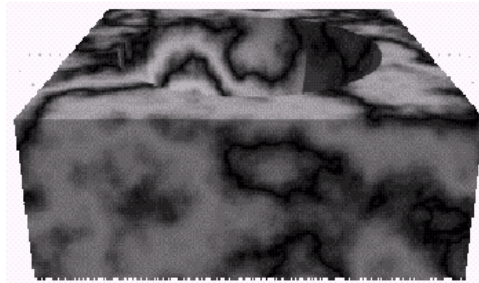
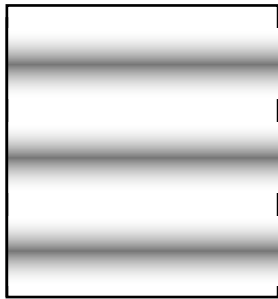
Extension des bruits en 2D, 3D, nD



Paramètres : **amplitude** et **fréquence** des B_i
ici, amplitude $1/2^i$ et fréquence $2^i \Rightarrow$ « octave »

Utilisation

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



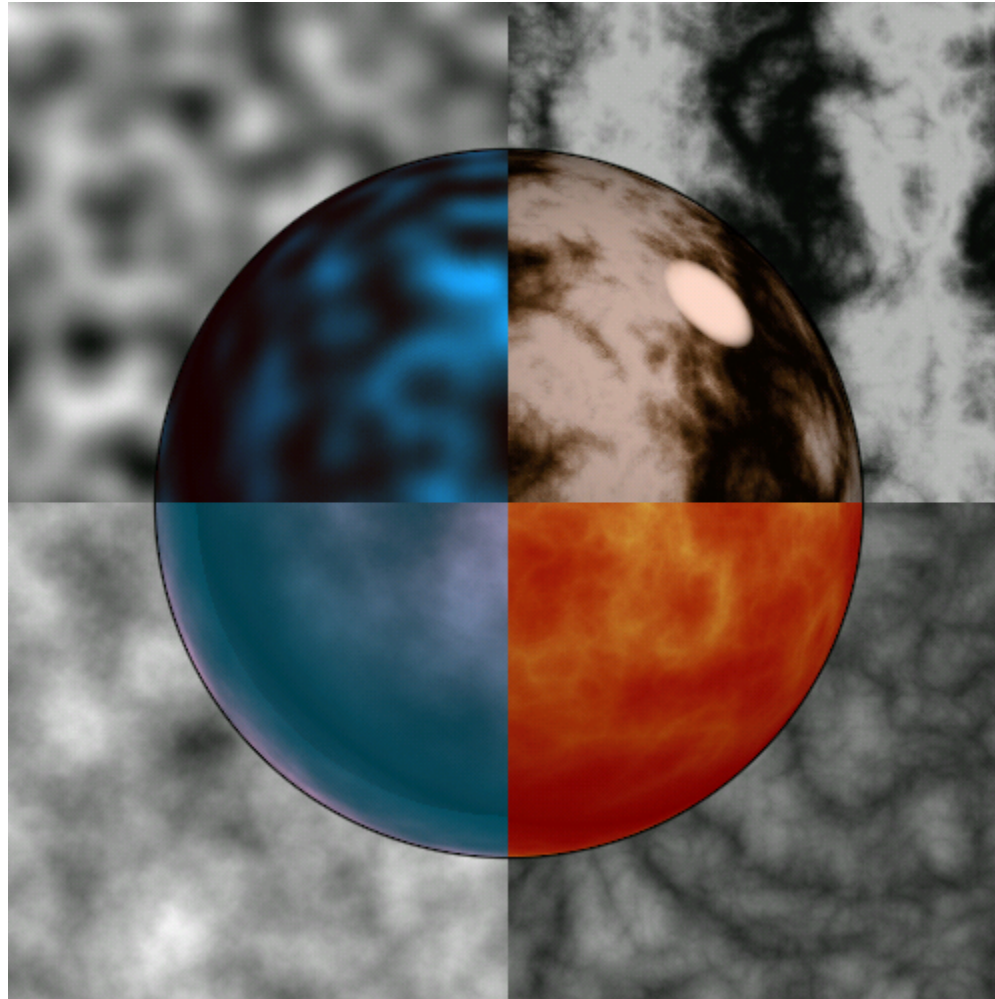
$$I(x,y,z) = \cos(x + T(x,y,z))$$



Bruit de Perlin

Bruit B

$\sin(x + T)$



$$T = \sum 1/f(\mathbf{B})$$

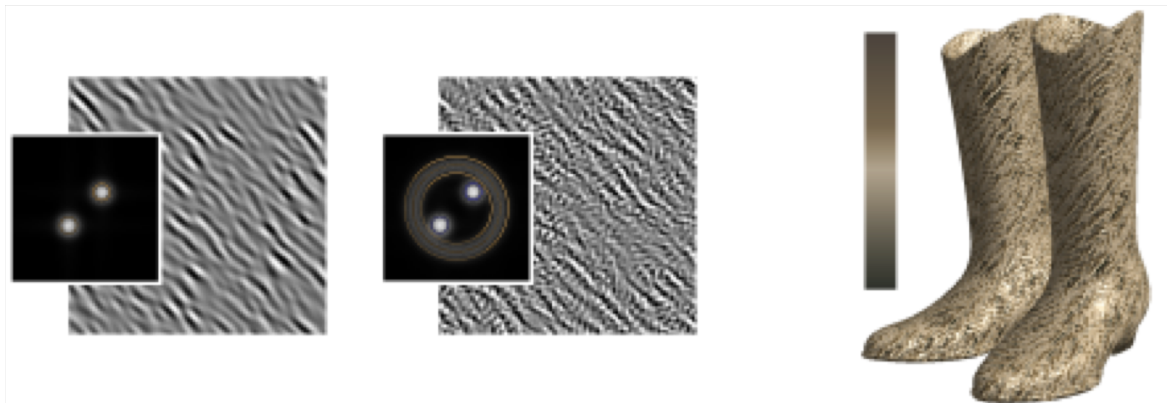
$$\sum 1/f(|\mathbf{B}|)$$

Bruits procéduraux

cf. "State of the Art in Procedural Noise Functions", Lagae et al. 2010

Nombreuses extensions

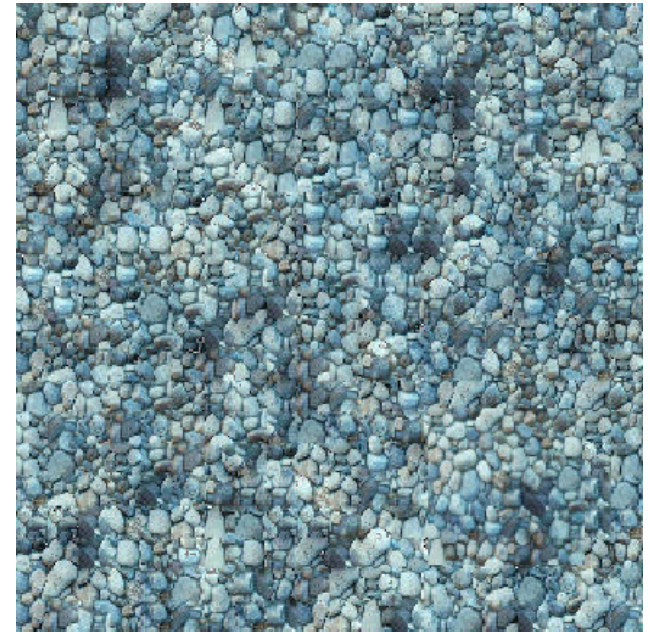
- **Simplex noise** [Per02]
plus rapide, meilleure qualité
- **Wavelet noise** [CD05]
contrôle du spectre, filtrage isotrope, plus lent
- **Anisotropic noise** [GZD08]
filtrage anisotrope, plus rapide, évaluation à la surface
- **Gabor noise** [LLDD09]
contrôle du spectre, filtrage, évaluation à la surface



Synthèse par l'exemple

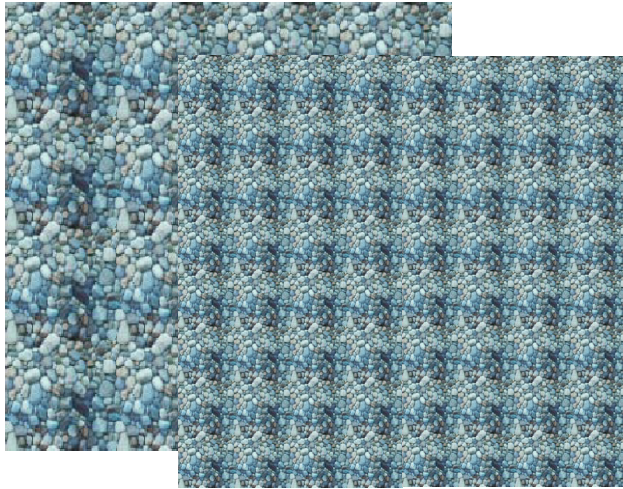


Entrée



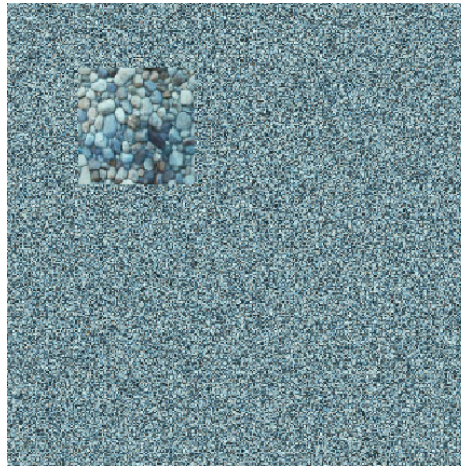
Sortie

Ce qui ne fonctionne pas...



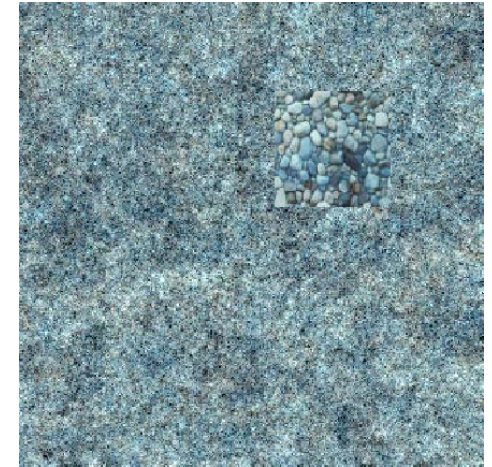
Recopier

- trop de répétition
- *aliasing*



Echantillonner les valeurs

- pas de structure
- cohérence horizontale



Echantillonner les fréquences (FFT)

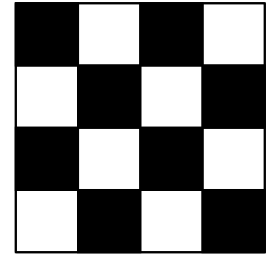
- pas de cohérence entre fréquences
- cohérence verticale

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.

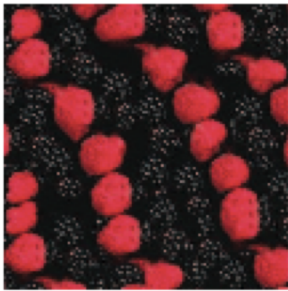


[Portilla & Simoncelli 2000]

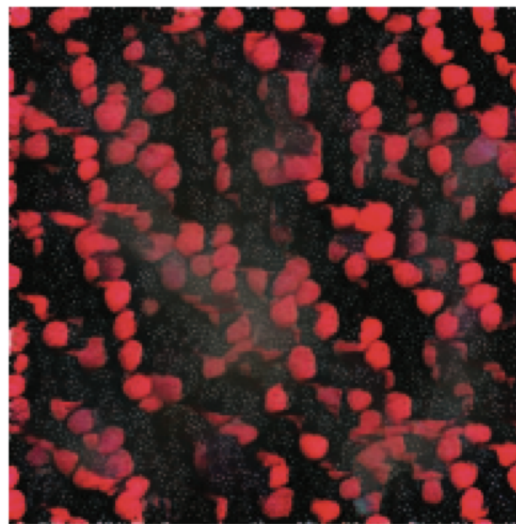
Approches paramétriques

Deep Learning

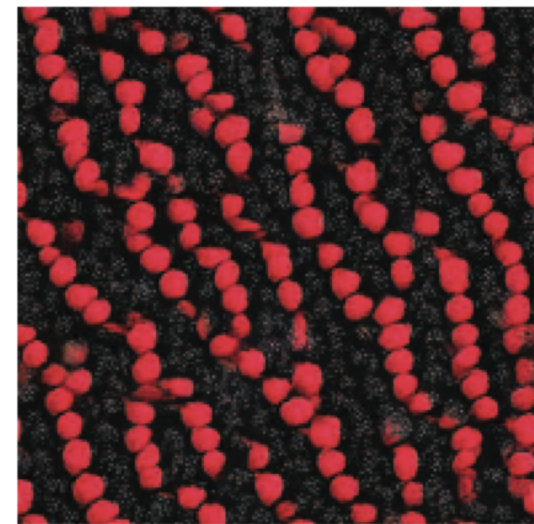
1. Extraction de *features* de l'image d'exemple avec un **réseau de neurones à convolution** (e.g., VGG [SZ14])
2. Synthèse par **optimisation d'une fonction de coût** sur certaines statistiques de ces *features*



Exemple



[Gatys et al. 2015]

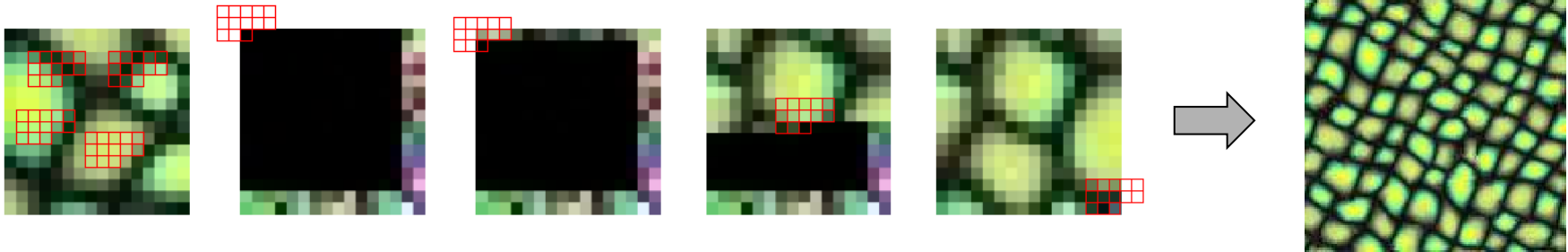


[Risser et al. 2017]

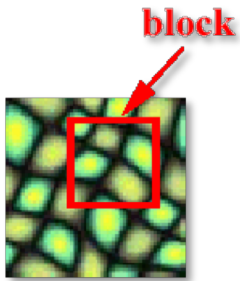
Approches non-paramétriques

Copie de **pixels** en respectant le voisinage

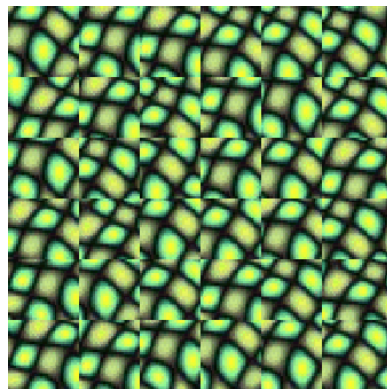
[Wei et al. 2000]



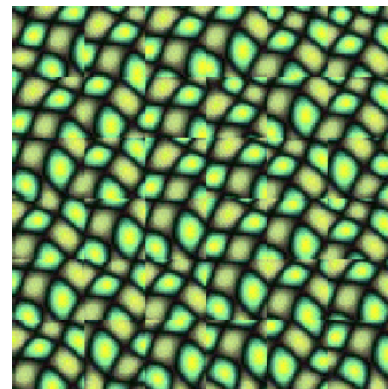
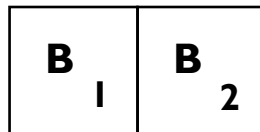
Copie de **patches**



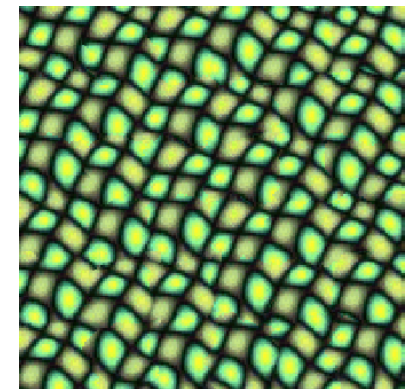
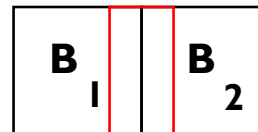
[Efros et al. 2001]



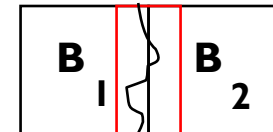
Placement
aléatoire



Contraintes
aux bords

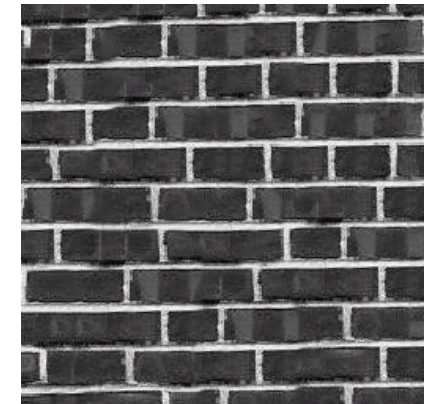
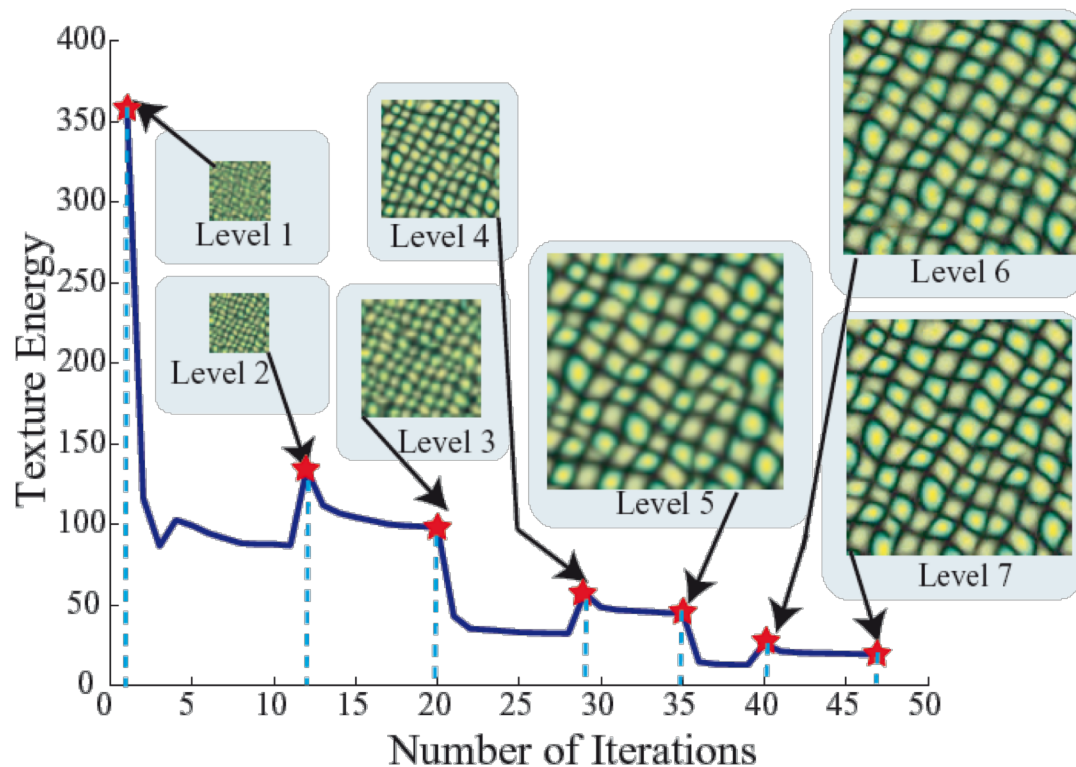


Optimisation
du collage



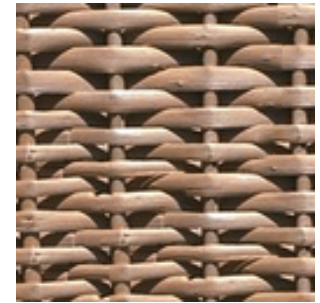
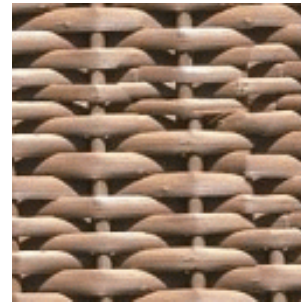
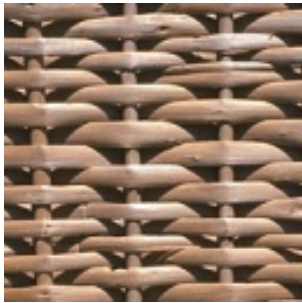
Optimisation globale

Minimisation itérative d'une **énergie globale**



[Kwatra et al. 2005]

Approches non-paramétriques



... of a visual cortical neuron—their
 ... describing the response of that neuro
 ... ht as a function of position—is perhap
 ... functional description of that neuron.
 ... seek a single conceptual and mathem
 ... scribe the wealth of simple-cell recep
 ... and neurophysiologically¹⁻³ and inferred
 ... especially if such a framework has the
 ... it helps us to understand the functio
 ... leeper way. Whereas no generic mo
 ... ussians (DOG), difference of offset C
 ... rivative of a Gaussian, higher derivati
 ... function, and so on—can be expect
 ... mple-cell receptive field, we noneth

... nounce tiarpm, nelolc swiom
 ... car es since,
 ... esoeao so eexcedi rep iacy ropas
 ... uogrs e—i—cisiae oi ones, m iul h
 ... y a—icciarcnesecacnoe uice dsioz
 ... neinto- eice sictm, ai asicr
 ... helialin—cicent-aimnri:cep
 ... ase onoiass as if amn.
 ... hal dell euecoronj jittlgmor rd th
 ... cingaretrnoqiscer tfr:eroc:s fulls
 ... Rnupactnewn cassa-153runnl.re .dl
 ... onl "—a hre anaeie ne w
 ... si onmooesl oile-can usansnlm i

... ition—is perk a single conceptual and
 ... of that neuribe the wealth of simple
 ... and matheurophysiologically¹⁻³ and
 ... simple-cell recially if such a framewor
 ... y¹⁻³ and inferlps us to understand th
 ... amework has perhay. Whereas no ge
 ... and the fumeuro:DOG), difference o
 ... no generic a single conceptual and m
 ... rence of offse the wealth of simple-ce
 ... , higher deriescribing the response of
 ... —can be expes a function of position—
 ... helps us to understand thription of th
 ... per way. Whereas no conceptual an
 ... sians (DOG), differencealth of simple

... iction of posiotuabe the wealth of sim
 ... | description of simirophroing the res
 ... gle conceposition—isies a function of
 ... reualth ion of that nectional descript
 ... reincti:ceptual and manus single con is
 ... g amcins of simple-ception of that
 ... cta single ally¹⁻³ and conceptual and
 ... nibe the wealth of sirealth of simple-ce
 ... europhysiologically¹⁻³ ologically¹⁻³ and
 ... ecially if such a frametch a framewor
 ... elps us to understand understand the
 ... per way. Whereas no hereas no gene
 ... ians (DOG), difference difference of
 ... tive of a Gaussian, highussian, higher

Exemples en
entrée

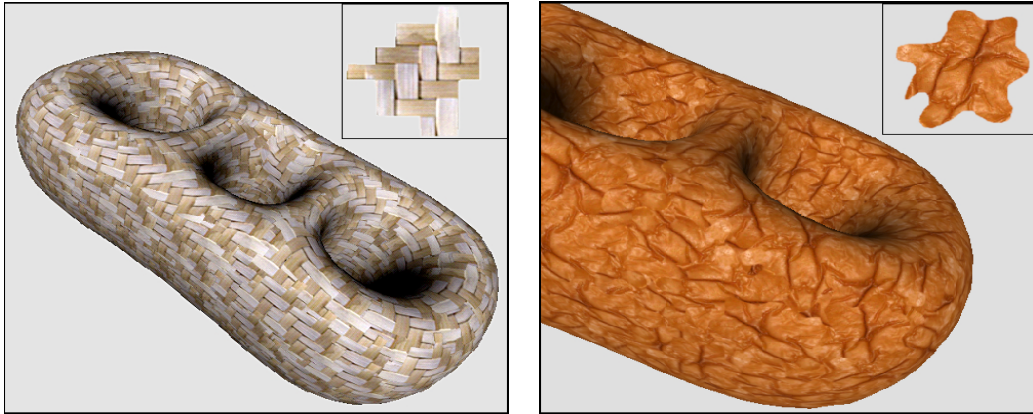
Copie de pixels
[Wei et al. 2000]

Copie de patches
[Efros et al. 2001]

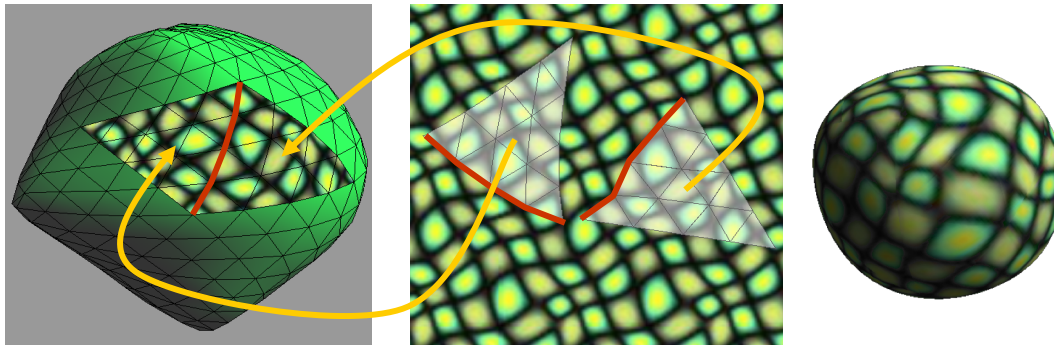
Optimisation globale
[Kwatra et al. 2005]

Synthèse 2D à la surface

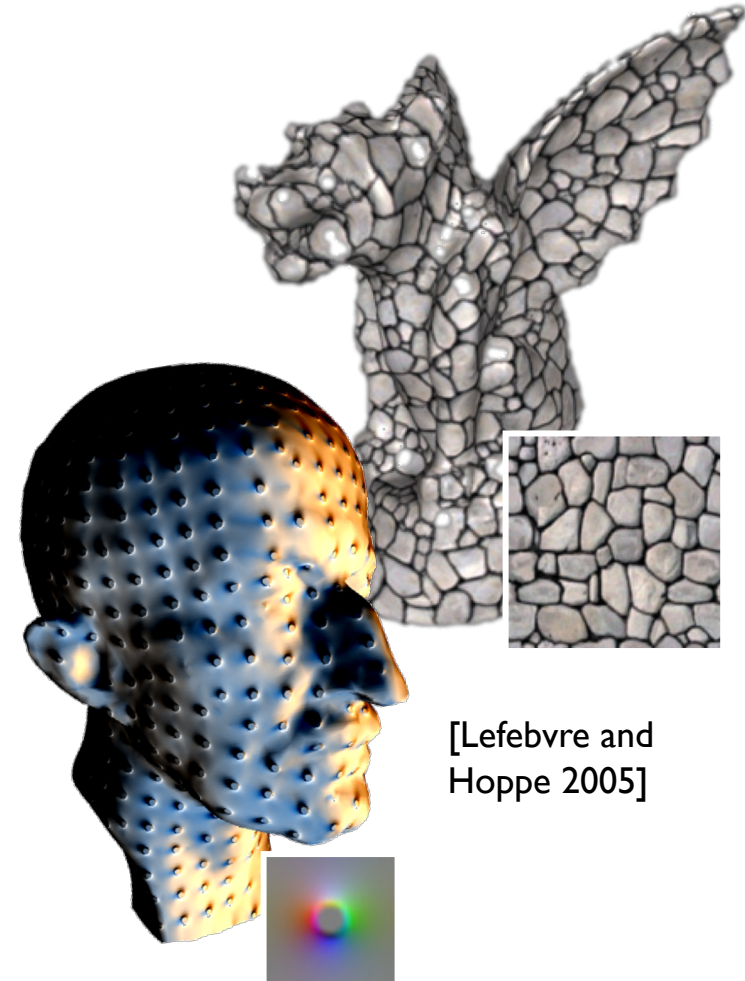
Idem en spécifiant le **voisinage sur le maillage**



[Praun et al. 2000]



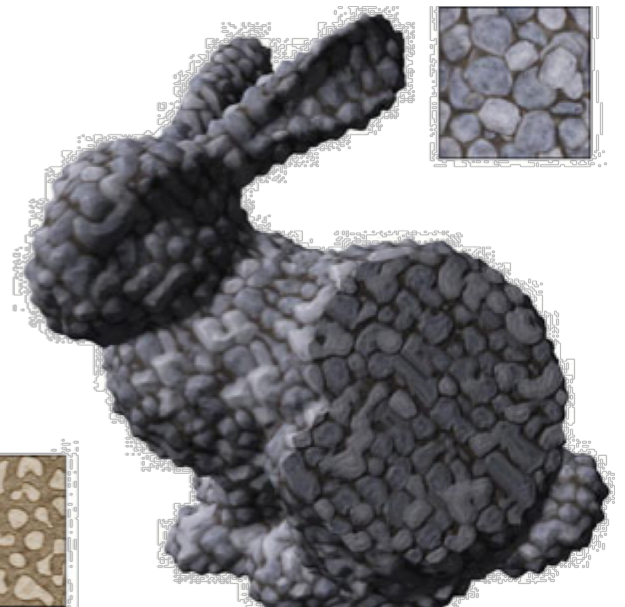
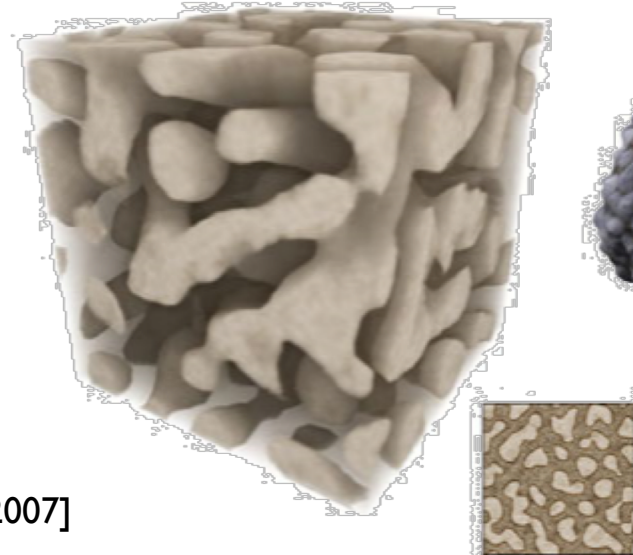
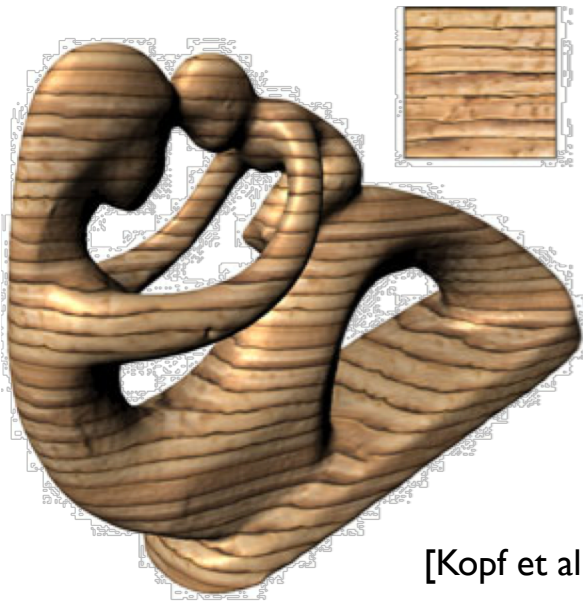
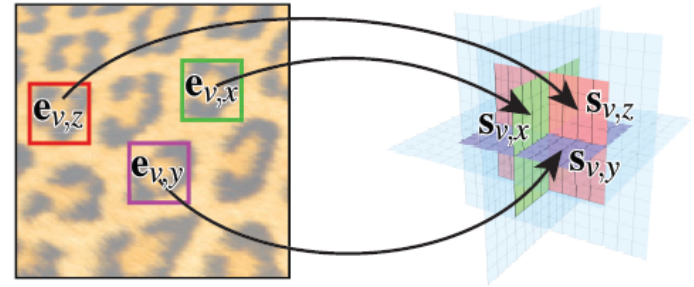
[Soler et al. 2002]



[Lefebvre and Hoppe 2005]

Textures volumiques

Synthèse par **voisinage**
selon **3 plans**



[Kopf et al. 2007]

Accélération

Recherche des plus proches voisins **très coûteuse**

⇒ méthodes d'accélération :

- Structures hiérarchiques (KD-Tree...)
- Pré-calcul (k-coherence)
- Échantillonnage stochastique (PatchMatch)