Introduction to Computer Graphics

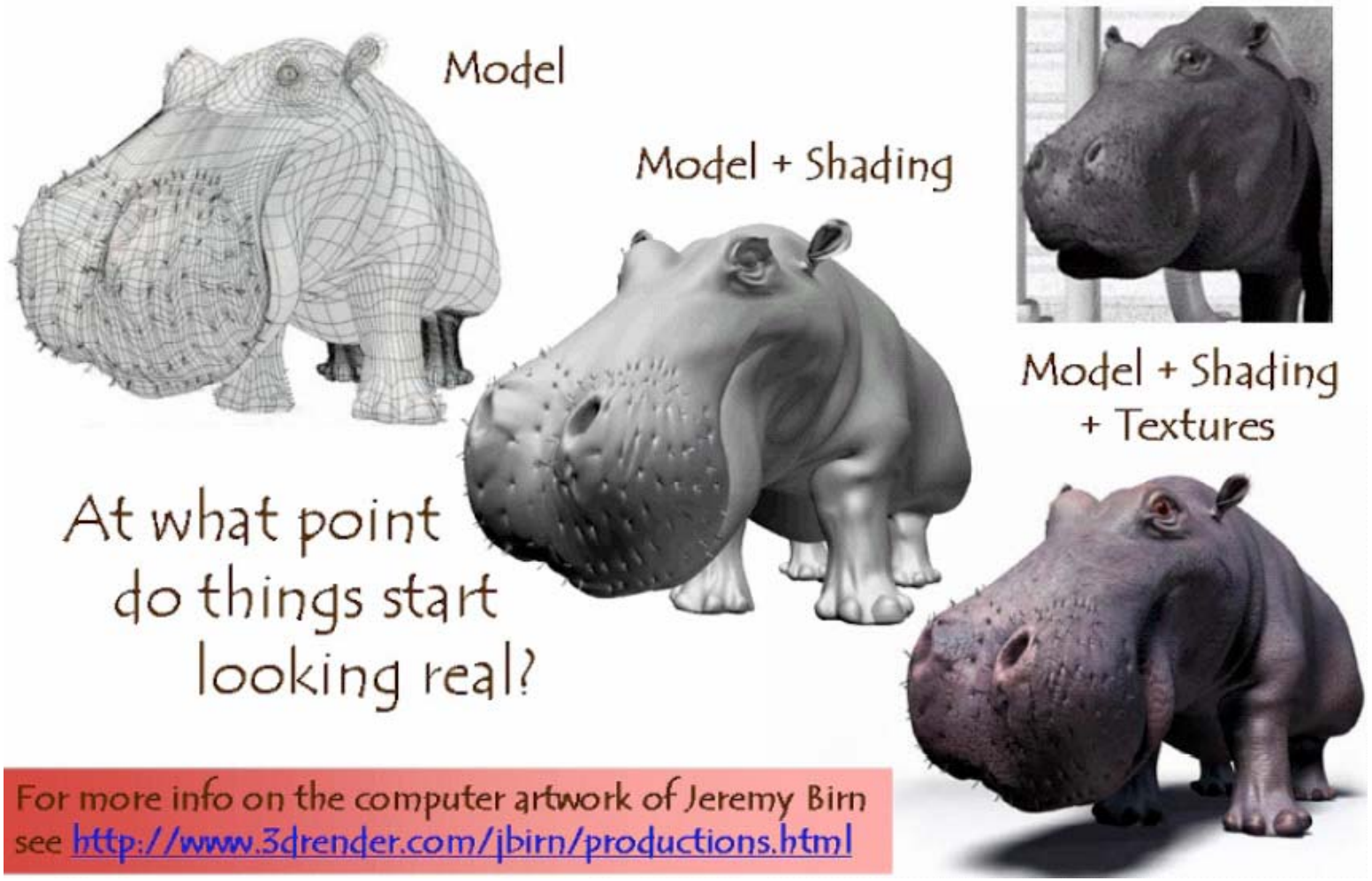
1. Rendering
2. Modeling
3. **Textures**
4. Animation
5. Complex models

Course schedule (3h a week, A009 or ARV)

Marie-Paule Cani & Estelle Duveau

04/02 Introduction & projective rendering
11/02 Prodedural modeling, Interactive modeling : parametric surfaces
25/02 Introduction to OpenGL + lab: first steps & modeling
04/03 Implicit surfaces 1 + C/lab: transformations & hierarchies
11/03 Implicit surfaces 2 + Lights & materials in OpenGL
18/03 **Textures** + Lab: Lights & materials in OpenGL
25/03 Textures in OpenGL: lecture + lab
01/04 Procedural & kinematic animation + lab: procedural anim
08/04 Physics: particle systems + lab: physics 1
22/04 Physics: collisions, control + lab: physics 2
29/04 Animating complex objects + Realistic rendering
06/05 Talks: results of cases studies

Motivation
Quest for Visual Realism



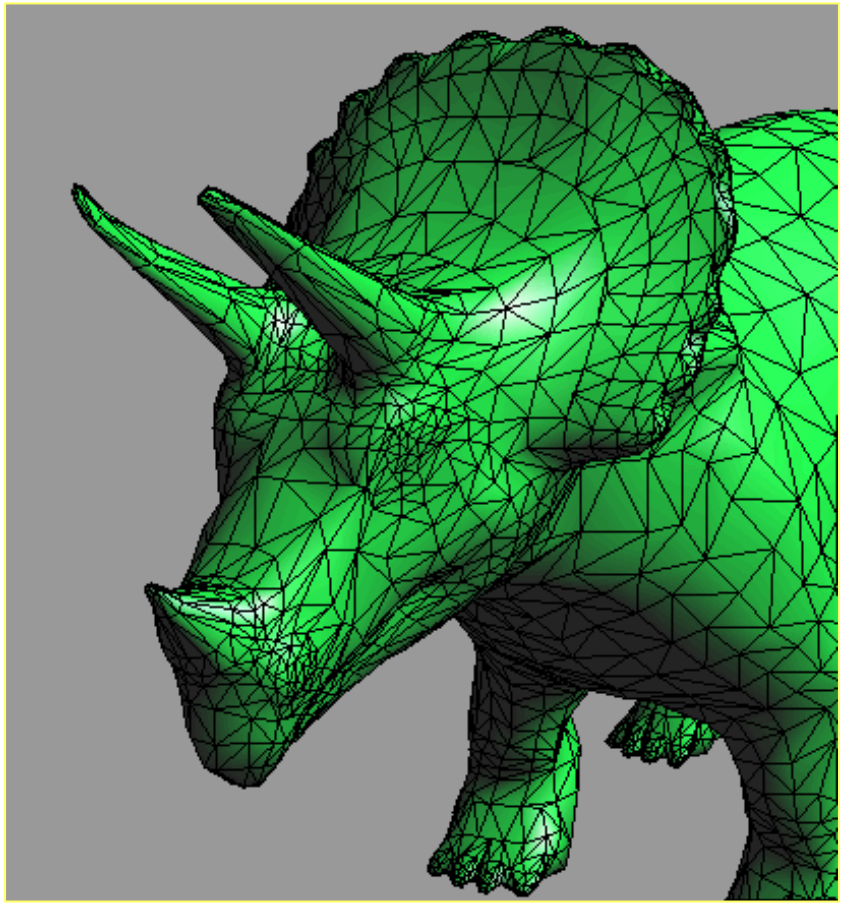
Reminder: from Modeling to Rendering

Object =

list of points + normals,

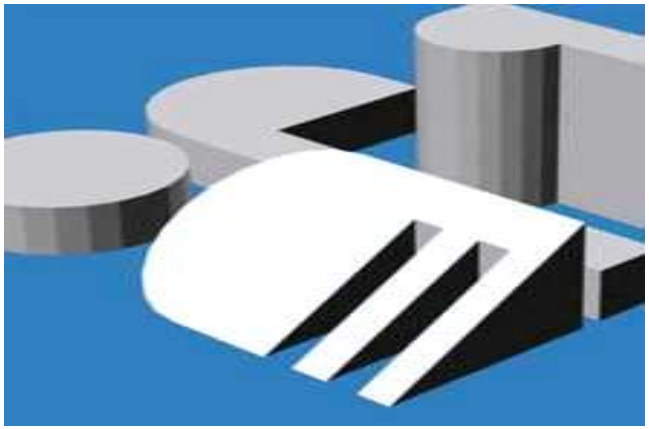
grouped into faces

- Faces are noticeable on the apparent silhouette

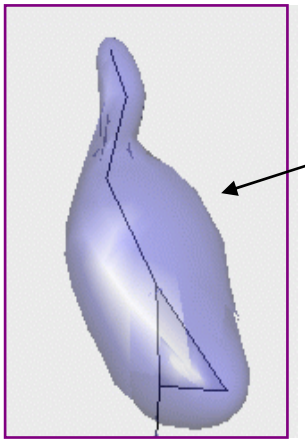


Reminder: from Modeling to Rendering

- Normals are used to compute shadows and reflection



Flat shading = normals to faces



Faces still visible on silhouettes!

Smooth surface: exact normals

- Soft shading for meshes:

  normal (P) = average of normals of adjacent faces

Modeling object's « material »

Add attributes (color, transparency…) to each point

- Color will be multiplied by the shading coefficient

Problem : We should not model everything at the scale of geometry!

Still a single face, but several colors?

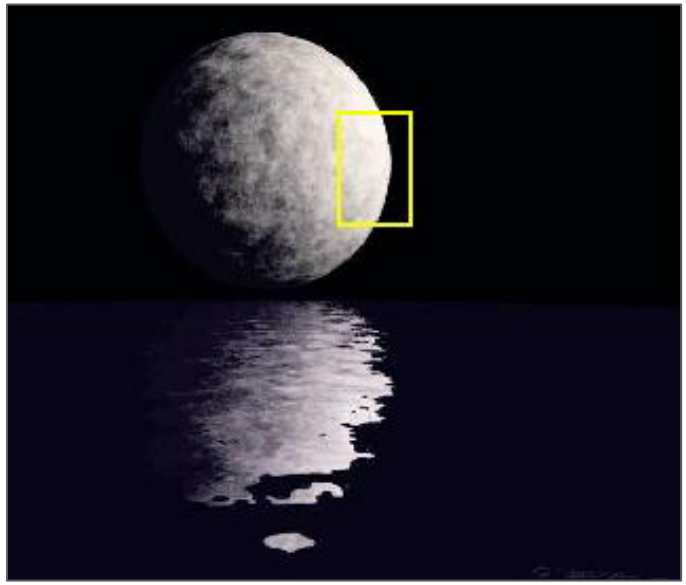
Modeling object's « material »

Add attributes (color, transparency…) to each point

- Color will be multiplied by the shading coefficient

Problem : We should not model everything at the scale of geometry!

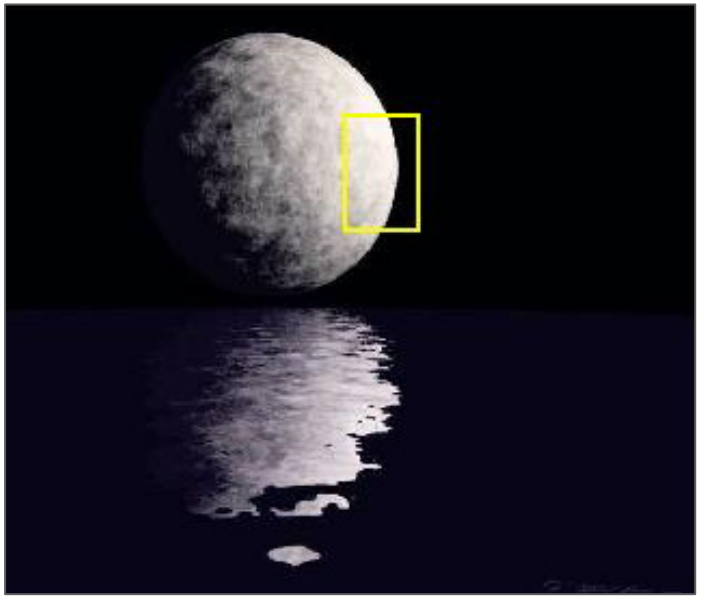
Micro-polygons would be needed!



Textures

- Enable the attributes to vary inside each face

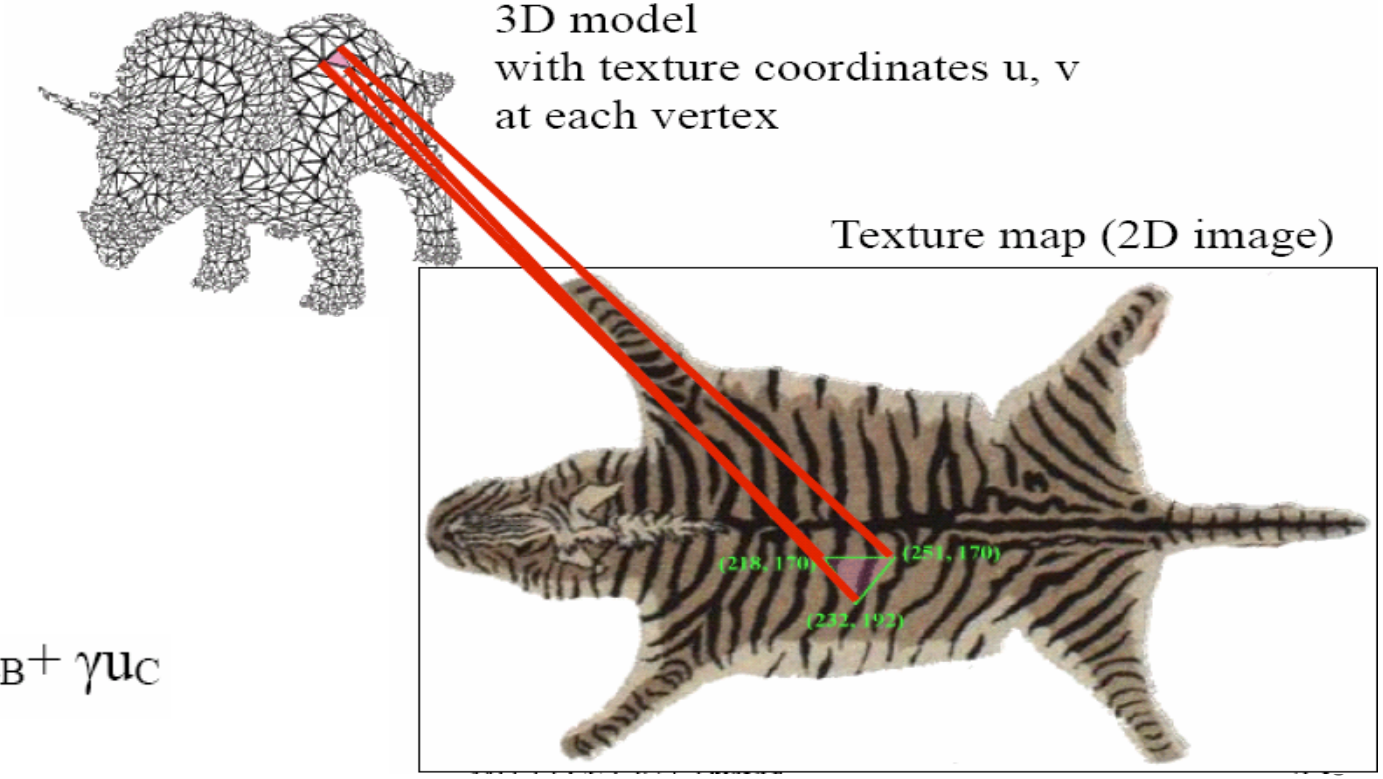
Texture for colors



Texture for normals

2D Textures

- Planar image I (u,v) + mapping P(x,y,z) → (u,v)
- Store: mesh point + normal + texture coordinates (u,v)



In a face:

Interpolate (u,v) using barycentric coordinates

$$u_P=(1-\beta-\gamma)u_A+\beta u_B+\gamma u_C$$

Simple mappings

f: (x,y,z) → [0,1] x [0,1]

- Planar mapping

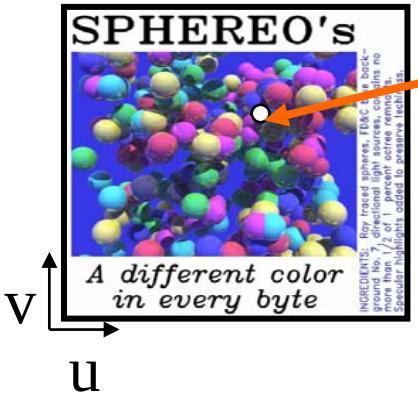
  f (x,y,z) = (||x|| , ||y||)

- Spherical mapping
  (part of a sphere)

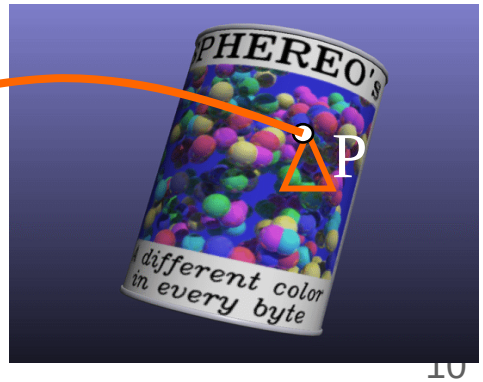
  f (θ,ψ) = (2θ/π , (π/2 – ψ) / π/4)

- Cylindrical mapping

  f (θ,z) = (θ/2π , z)

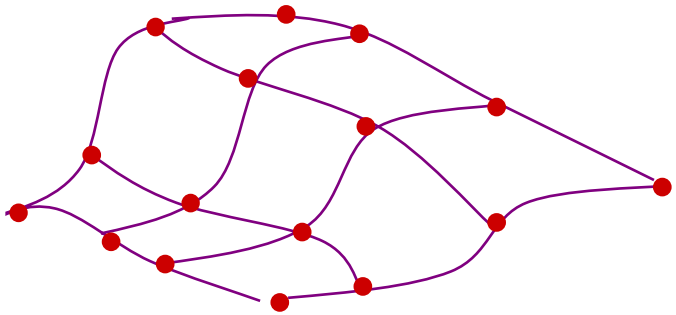


f



Mapping on free-form surfaces ?

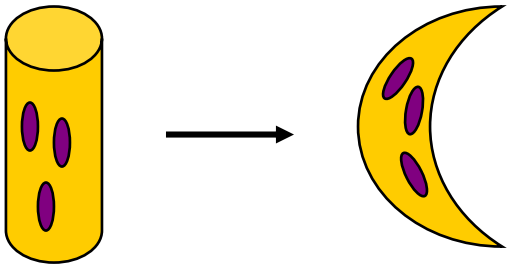
f: (x,y,z) → [0,1] x [0,1]



- On a spline surface S(u,v)

  f (S(u,v)) = (u,v)

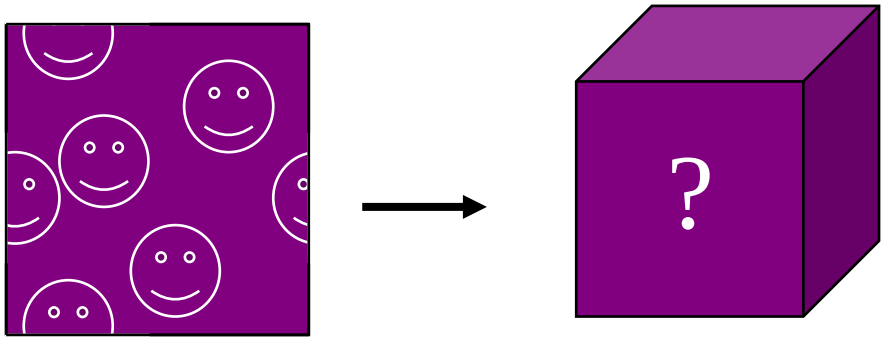
- Other??? (free form mesh...)
  - Map the texture before applying the deformation

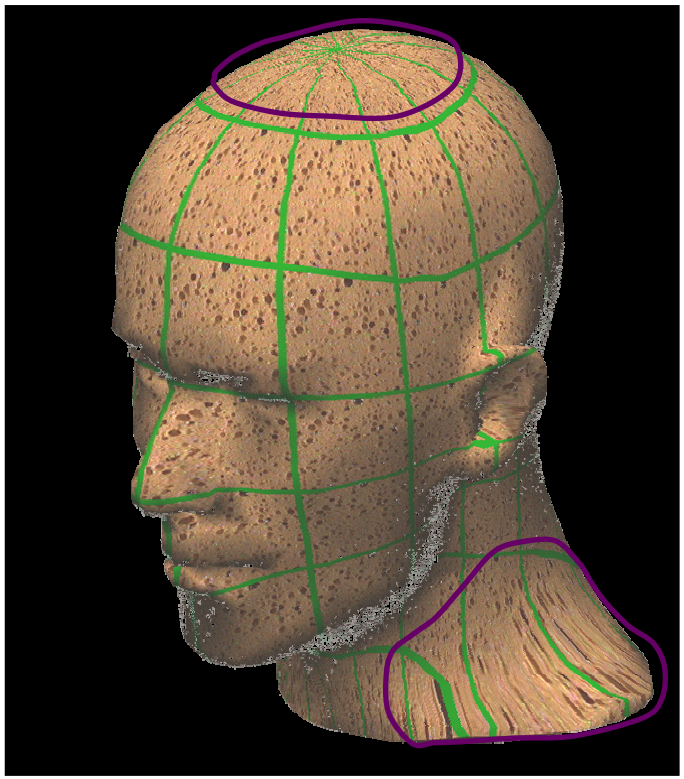


  - Or paint the texture on the mesh!

Mapping 2D textures: problems!

- Mapping on free-form shapes
  - Singularities (poles)
  - Distorsions
- Topological problems !

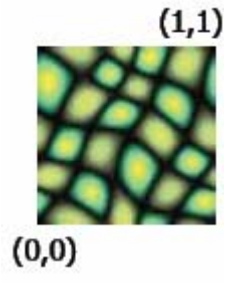




Case of pattern-based texturing

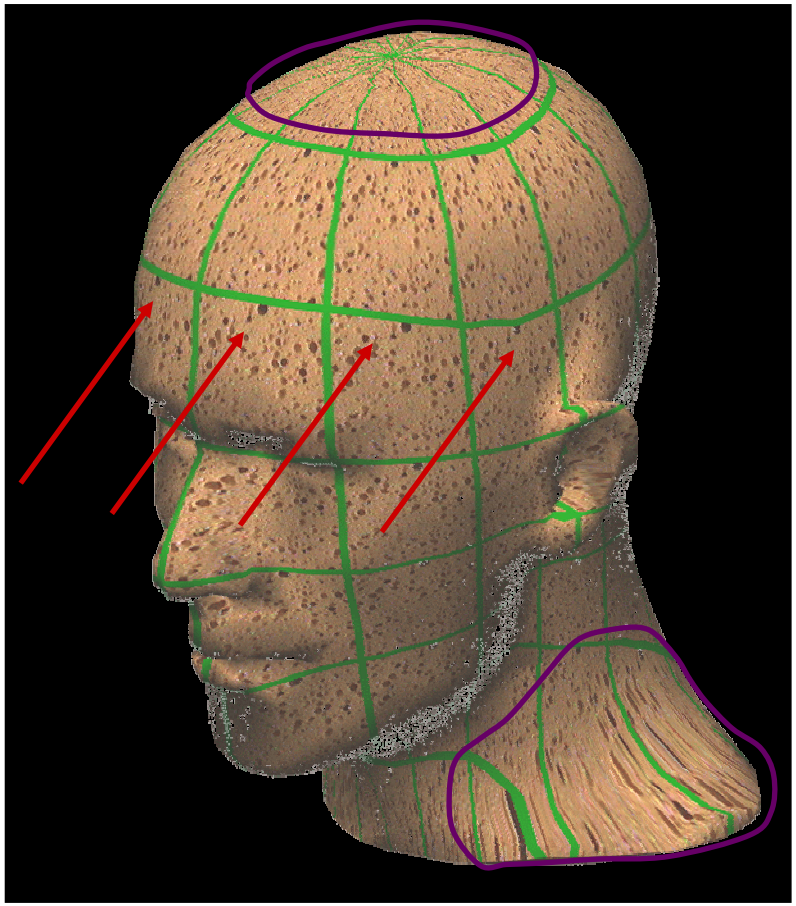
Image: pattern with toric topology

- Still poles and distorsions
- Too repetitive!



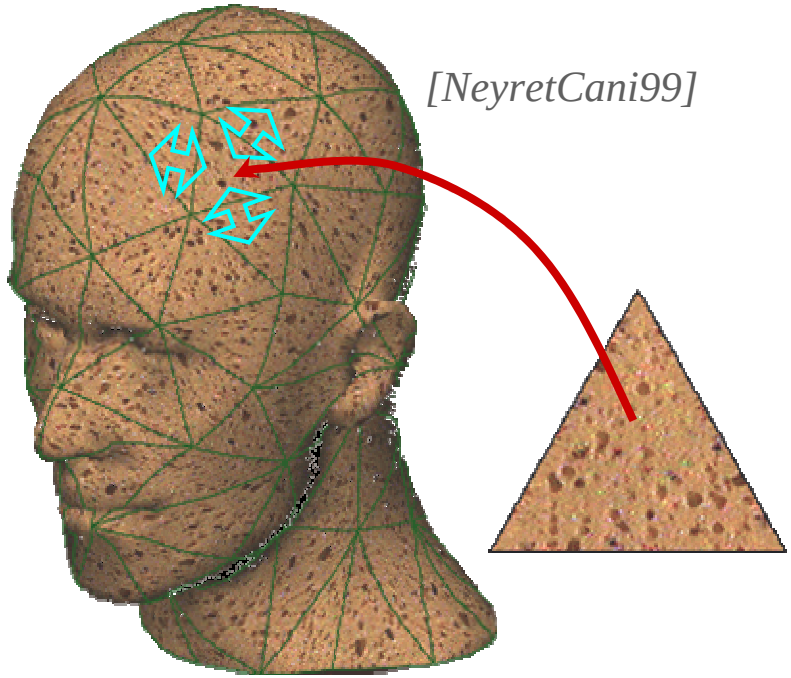
Paint of synthetise a similar texture on the surface ?

- Burden to create
- Costly to store

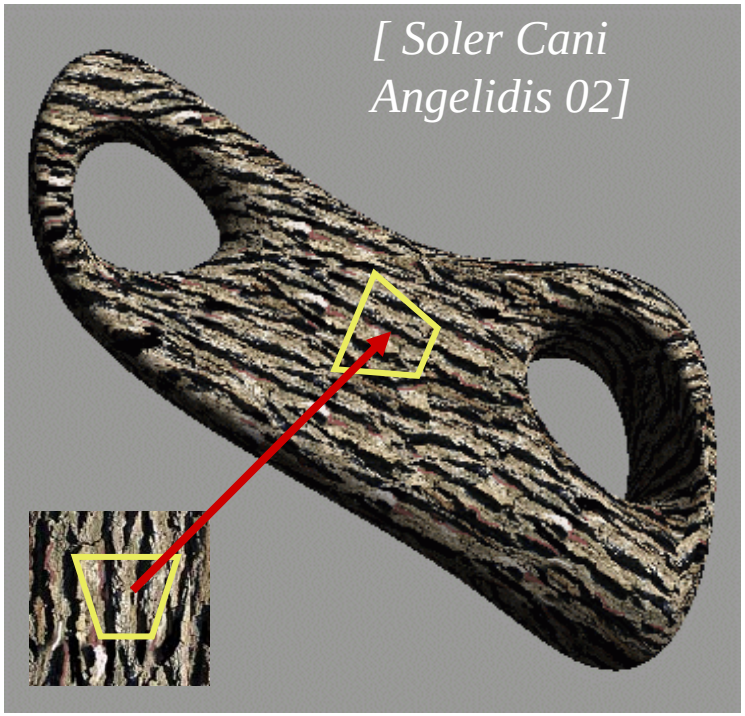


Pattern based texturing

- Some solutions

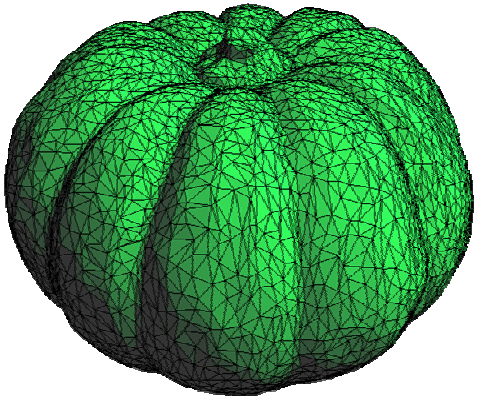


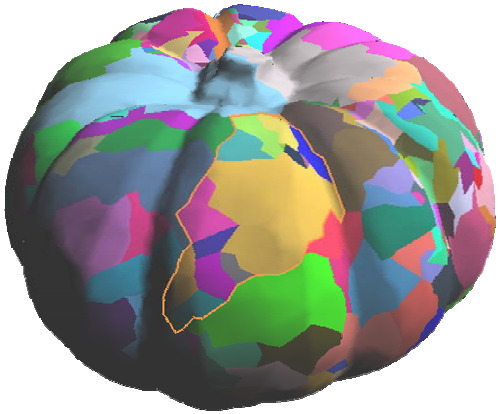
Isotropic textures

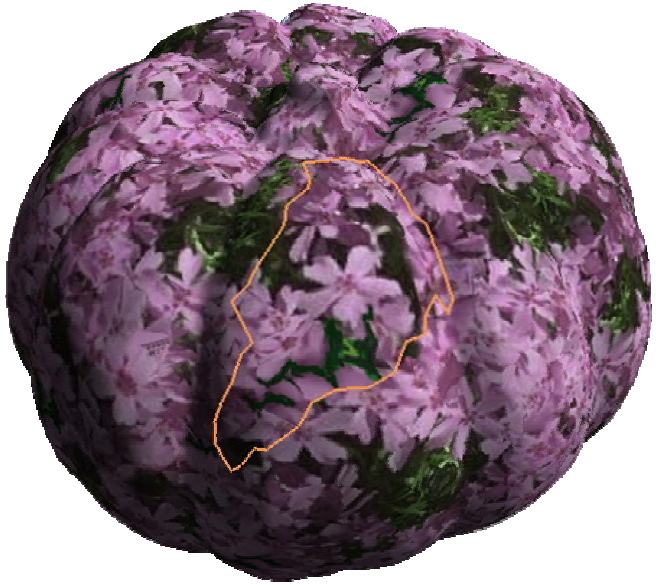


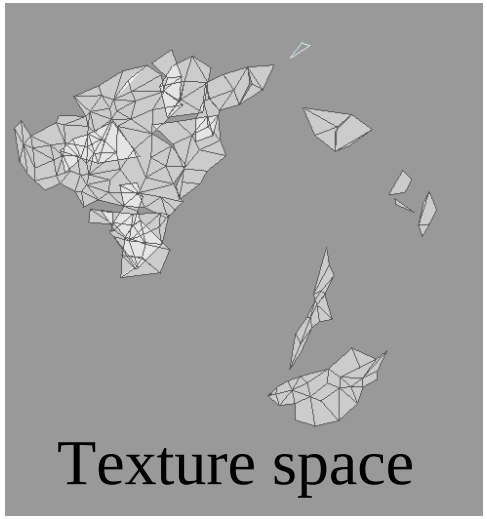
Arbitrary textures

Arbitrary pattern-based texturing









Solution by hierarchical local mapping

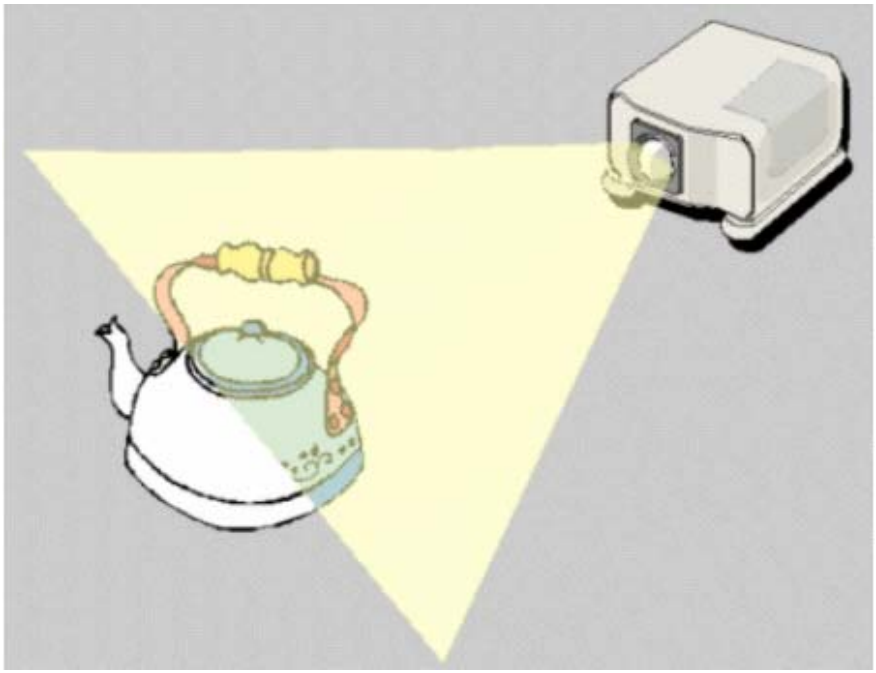
[Soler Cani Angelidis 02]

Projective textures

Use the texture like a slide projector

No need to specify texture coordinates explicitly!

- A good model for shading variations due to illumination
- A fair model for reflectance (can use pictures)
- Bad for animation!



Projective textures: example

Modeling from photograph: use input photos as texture

Figure from Debevec, Taylor & Malik
http://www.debevec.org/Research

3D textures

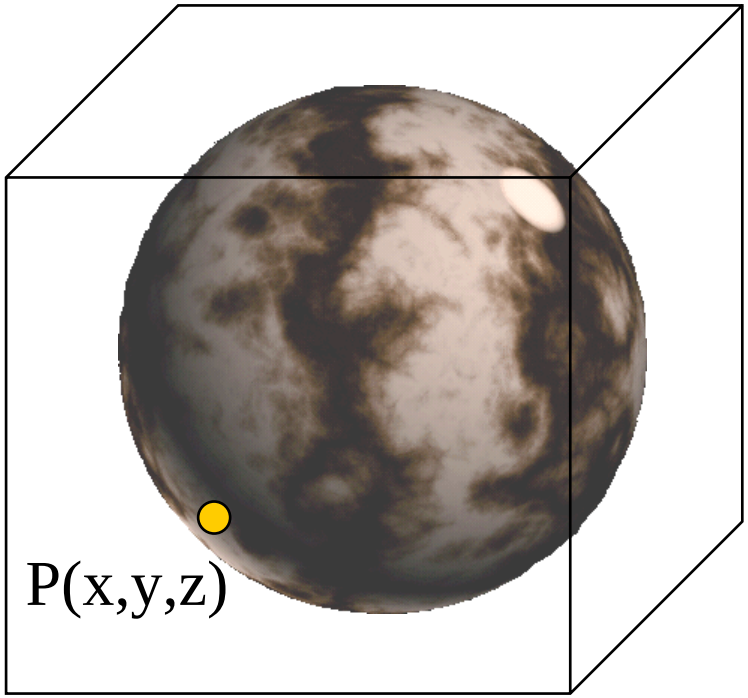
- Volume of material embedding the object
  - Wood, marble…

+ No mapping to compute !

u = x, v = y, w = z

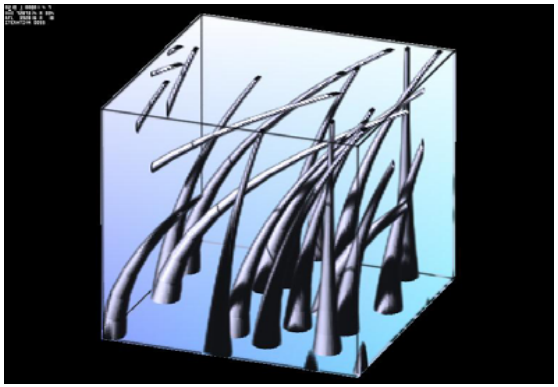
- Memory cost

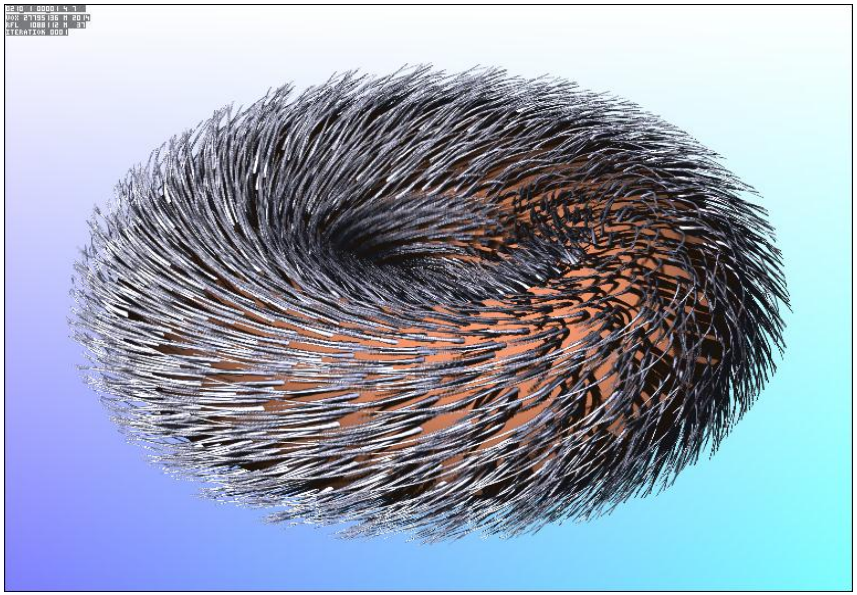
(ou of computational cost)



Texels : Some 3D along a surface

- Multiresolution voxels with 3D geometry
- Mapped on surface, like texture patterns





Creating a texture

- Real images
  - 2D textures only
  - Re-shading problem

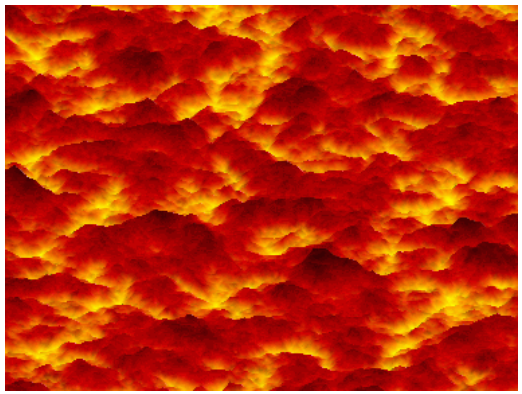
Snake skin (photograph)

Creating a texture

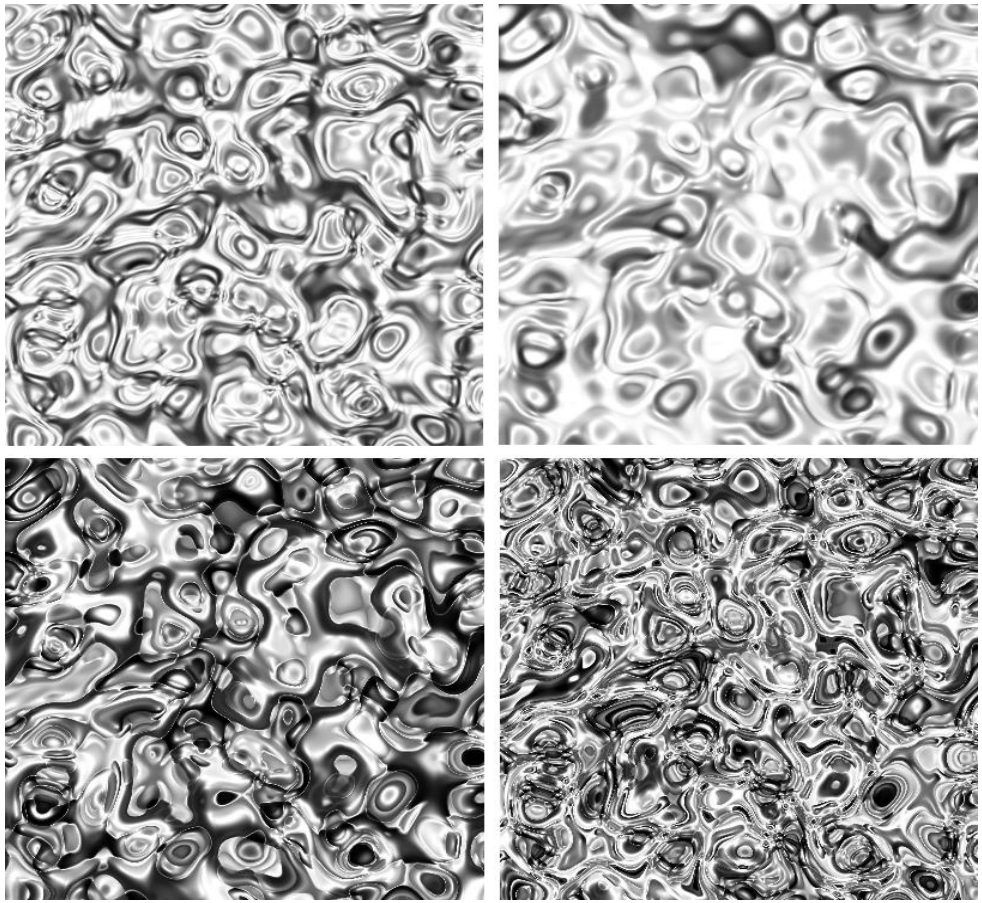
- Procedural textures (2D and 3D)
  - Statistical analysis of a material
  - Creation of a similar texture

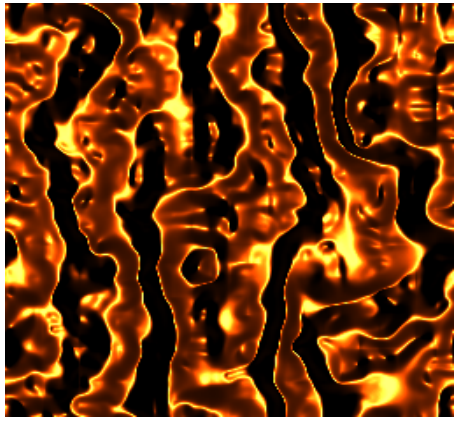
  - Continuous fractal noise

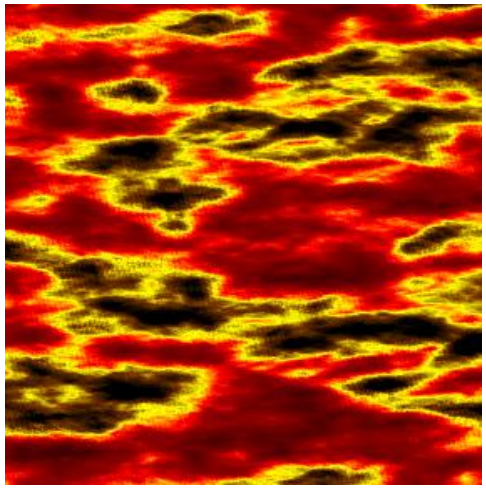
    (wood, marble, lava)



Perlin's textures (continuous fractal noise)



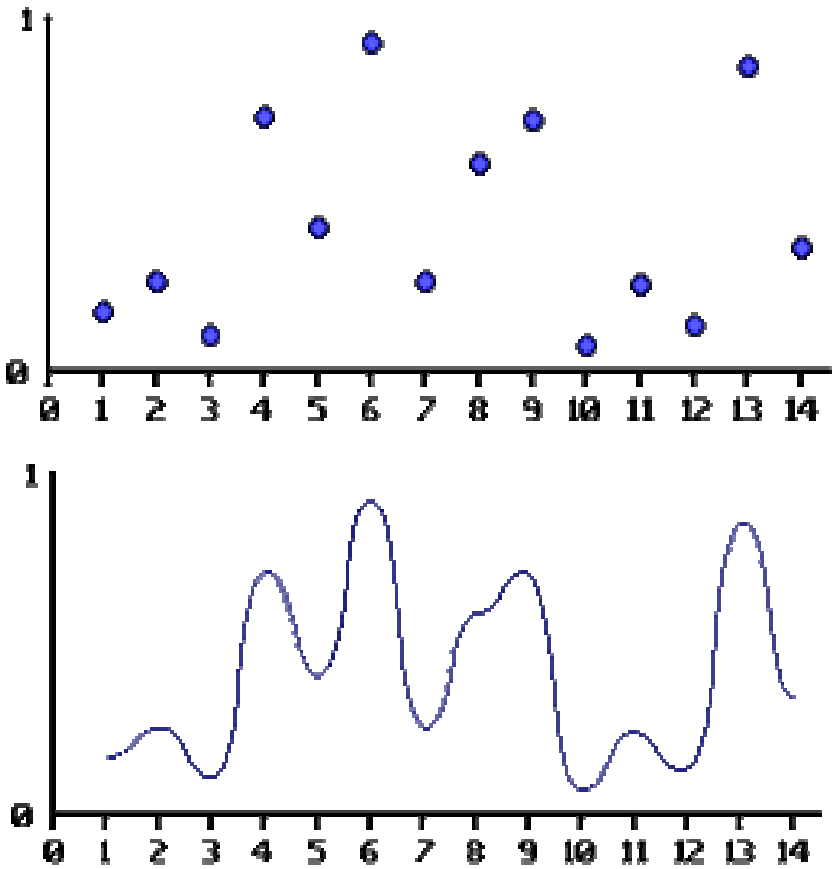




Perlin's noise

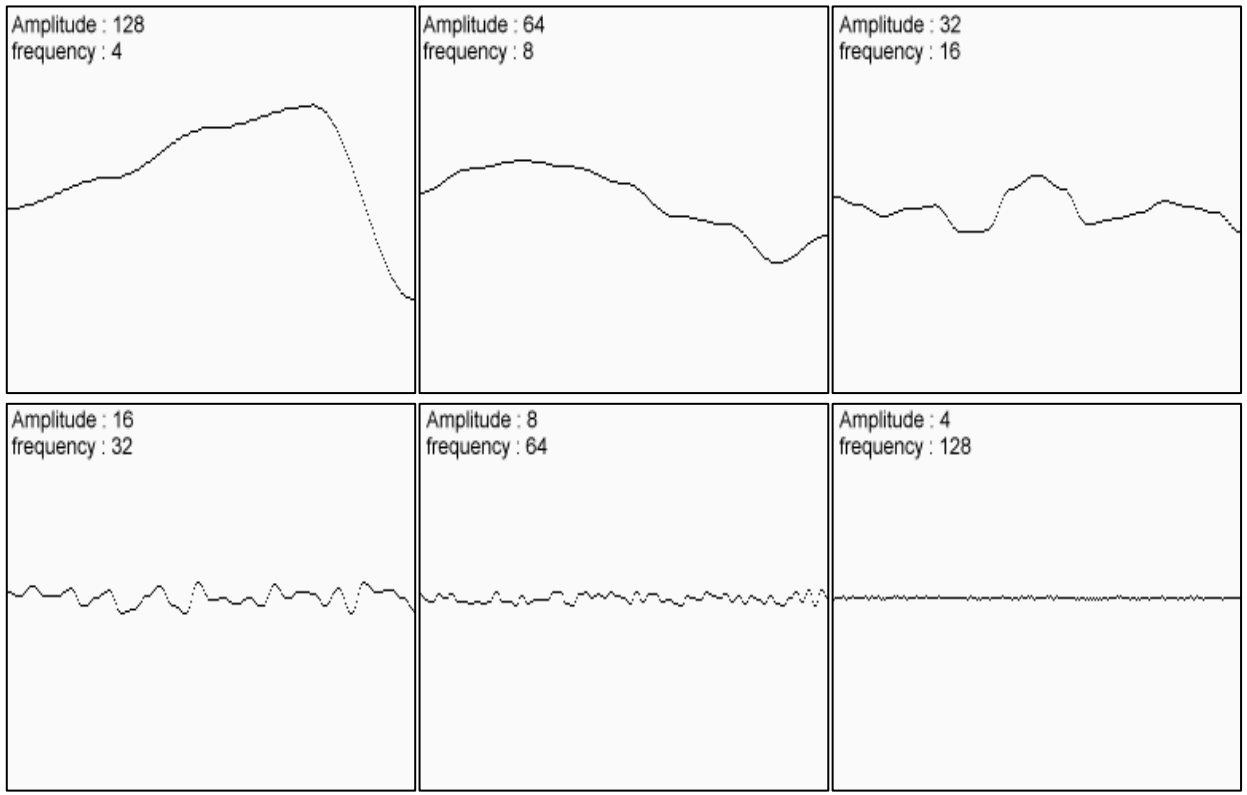
1D basis function

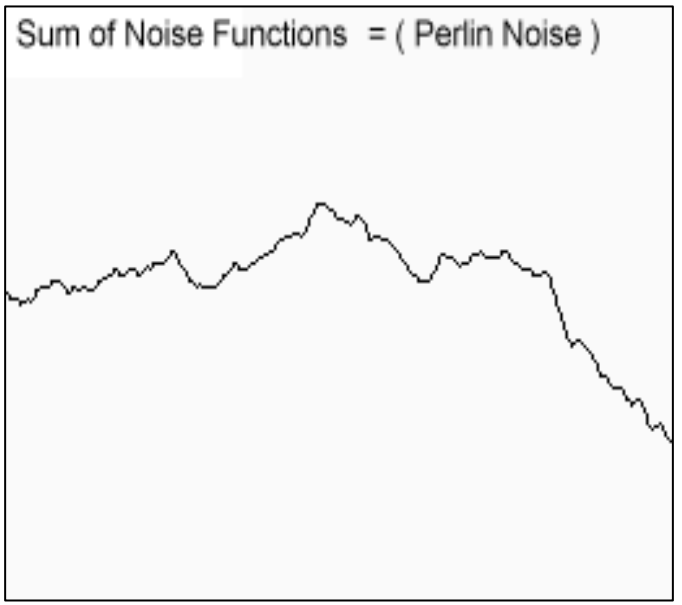
- ***B(x)*** = interpolation of evenly spaced random values
  - Pseudo-period!
- Pre-computation of values (1D table)
- To reduce smoothness
  B'(x) = | 2B(x) -1 |



Perlin's noise

Turbulent noise : sum copies of *B* at different scales

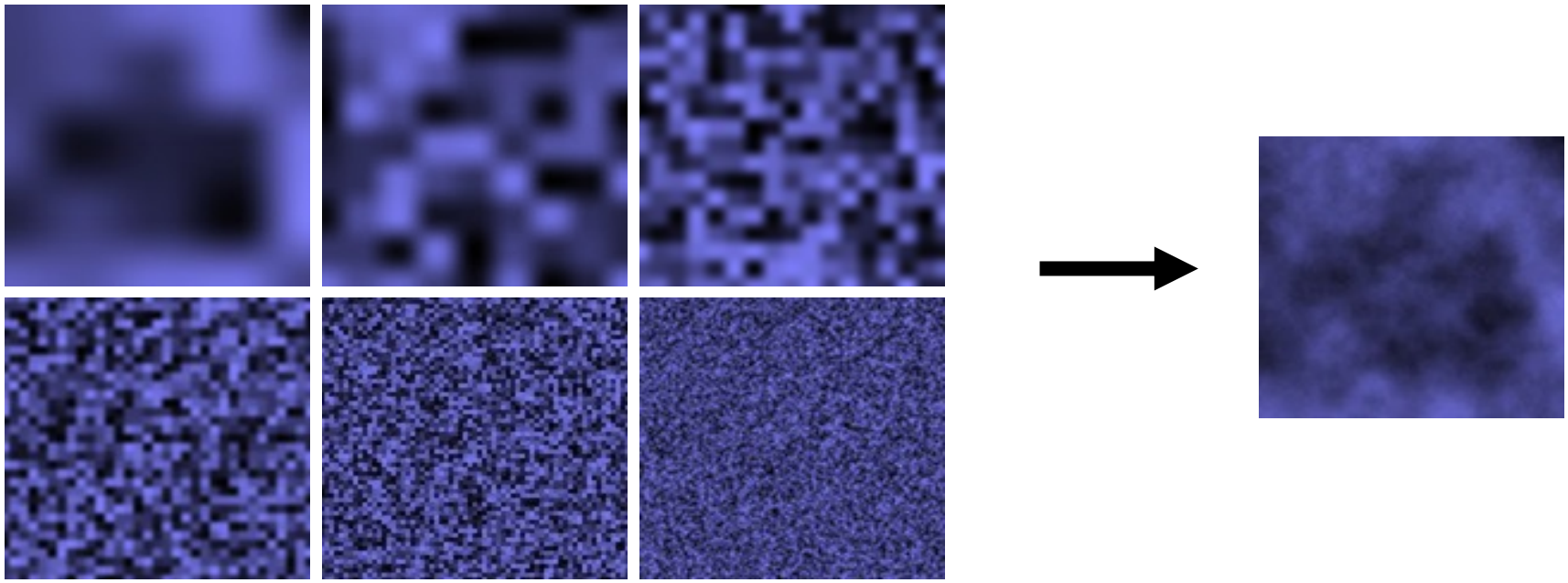




$$T(x) = \sum 1/2^i \; B(2^i x)$$

Perlin's textures

2D or 3D extensions of Perlin's noise



Random values (x,y) = 1D table [(x + permut(y)) mod n]

Perlin's texture : control ?

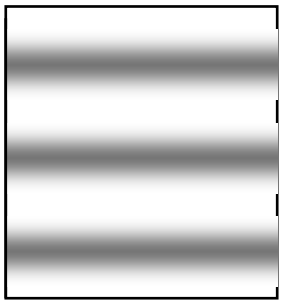
- Direct use by combining « boxes »
  - Noise linked to some of the material attributes
- Use to modify an image
  - Image : $I(x,y)$ replaced by $I(x+T_1(x,y), y+T_2(x,y))$

Perlin's texture : control ?

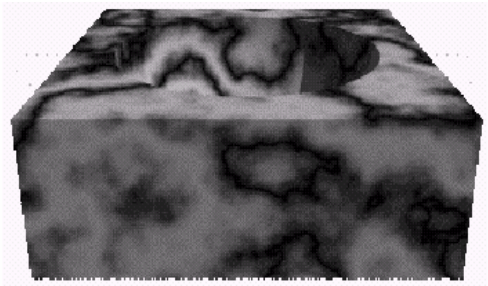
Modification of an image or a simple function

- Image : $I(x,y)$ replaced by $I(x+T_1(x,y), y+T_2(x,y))$
- Method applied to an « idealized » material

Example : Marble = veins + noise



cos(y)



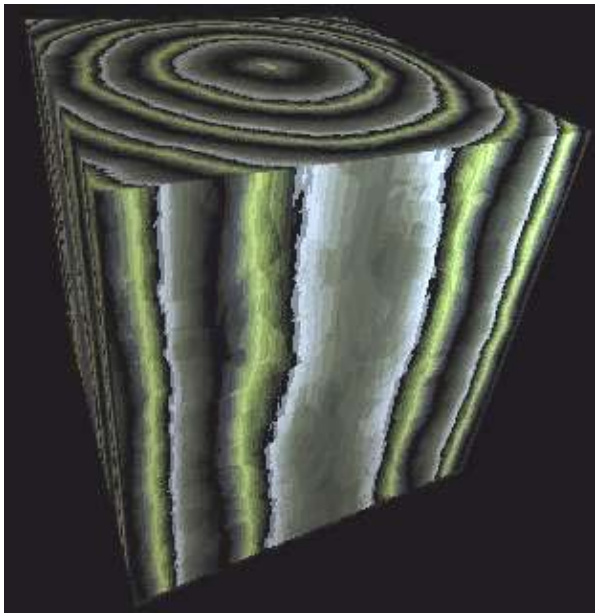
cos(y + k T(x,y,z))

Perlin's texture : control ?

Modification of an image or a simple function

- Image : $I(x,y)$ replaced by $I(x+T_1(x,y), y+T_2(x,y))$
- Method applied to an « idealized » material

Example : Wood
= series of cylinders



Textures of normals: « bump mapping »

Perturbation of normals

- Given by a Perlin noise, an image

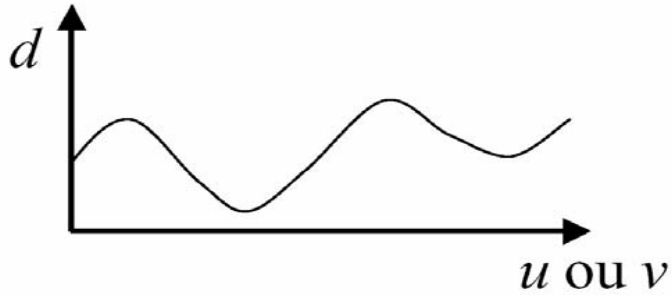
- Computed from a displacement

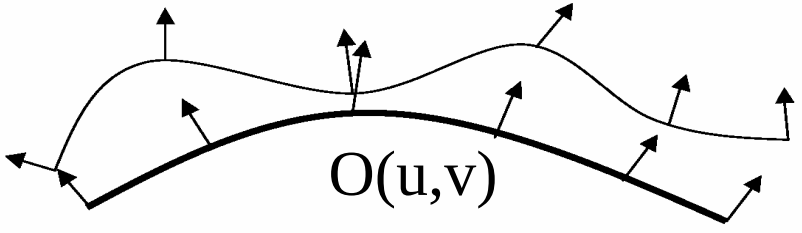
$O'(u,v) = O(u,v) + d(u,v)\, N(u,v)$

$N'(u,v) = N + \partial d/\partial u\, A + \partial d/\partial v\, B$

$A = N \wedge u,\ B = N \wedge v$

Gradient of d : finite differences





Textures of normals: « bump mapping »

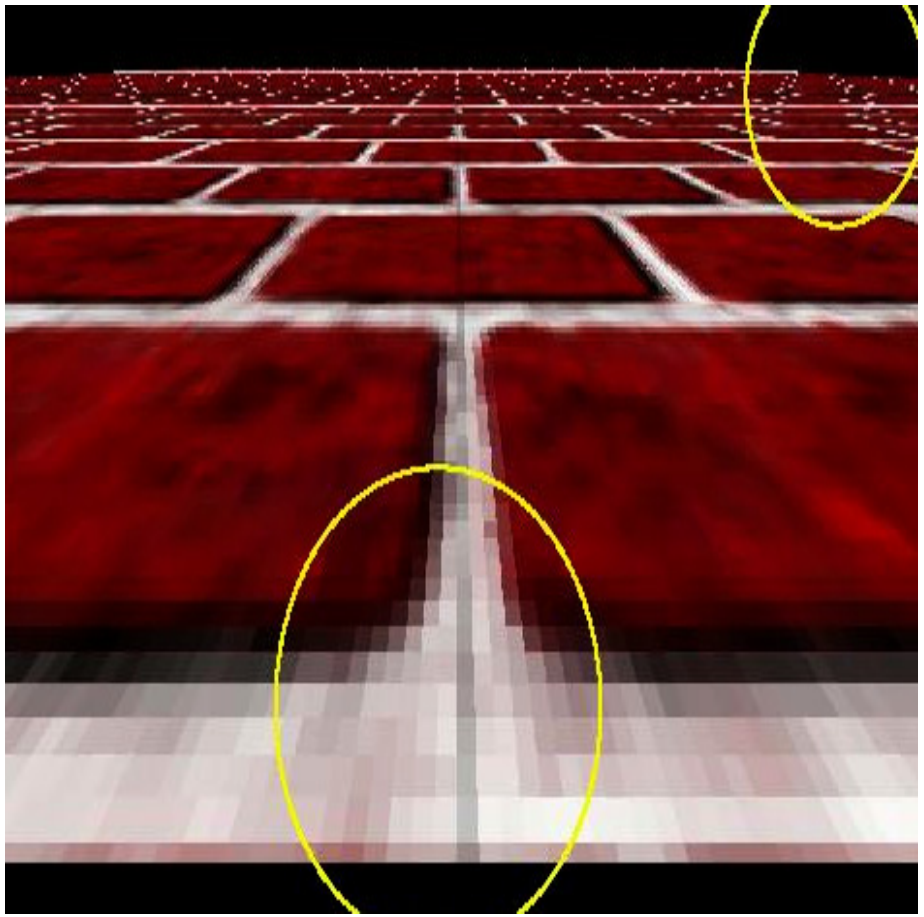
- Example : displacement given by an image
  - The geometry of faces has not been changed!

Aliasing problems for textures

- At the front, we see the pixels

- At the back

- Several colors to display in a pixel of the screen
  - An average would be good!
- Extra low frequencies called « aliases »



Aliasing problems for textures

Solutions

- Post-filtering (general case)
  - Image computed at a higher resolution
  - Smoothing filter

  The image we filter is incorrect!

  (the aliases are still there)

- Pre-filtering (textures only)
  - texture = image pyramid

  Bilinear « mip-mapping »