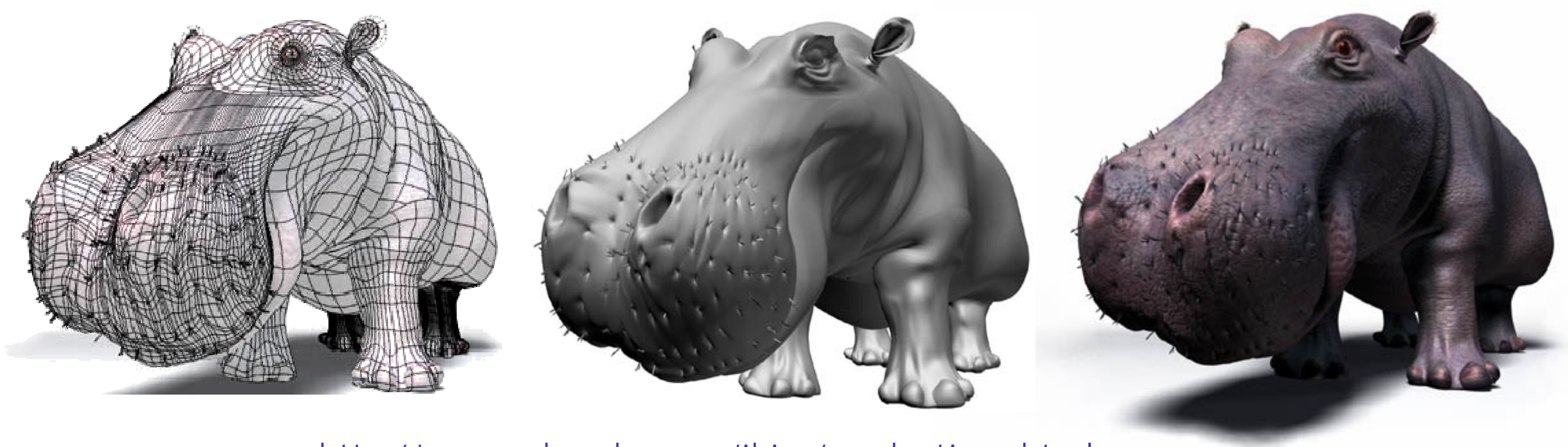
Texture Mapping

Motivation

Wireframe Model + Lighting & Shading + Texture Mapping

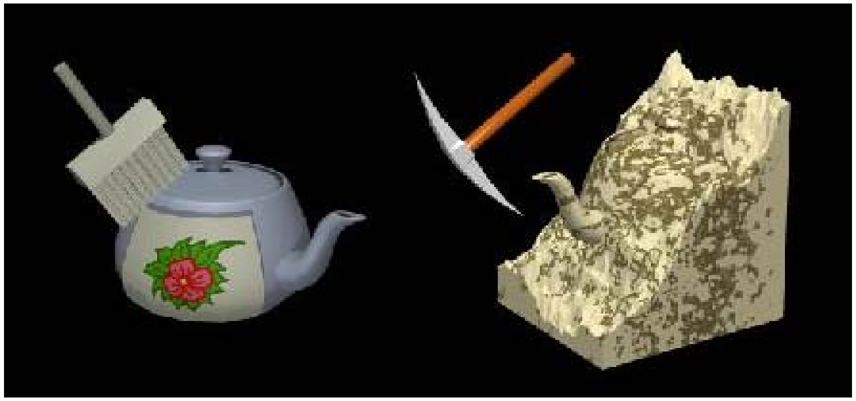


http://www.3drender.com/jbirn/productions.html

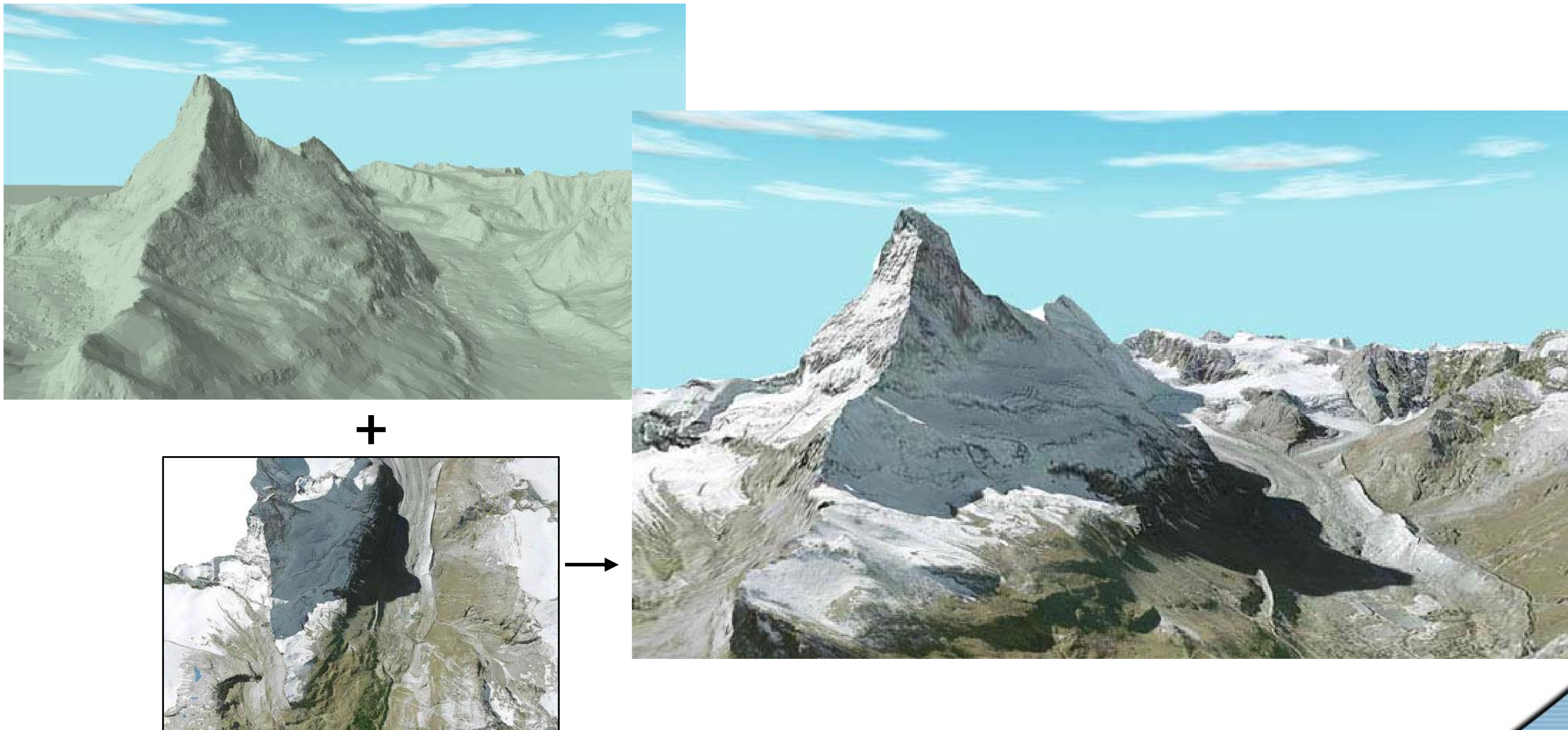
towards more realism

Idea

- Add surface detail
  without raising geometric complexity
- Textures can be images or procedures
- Textures can be 2D or 3D

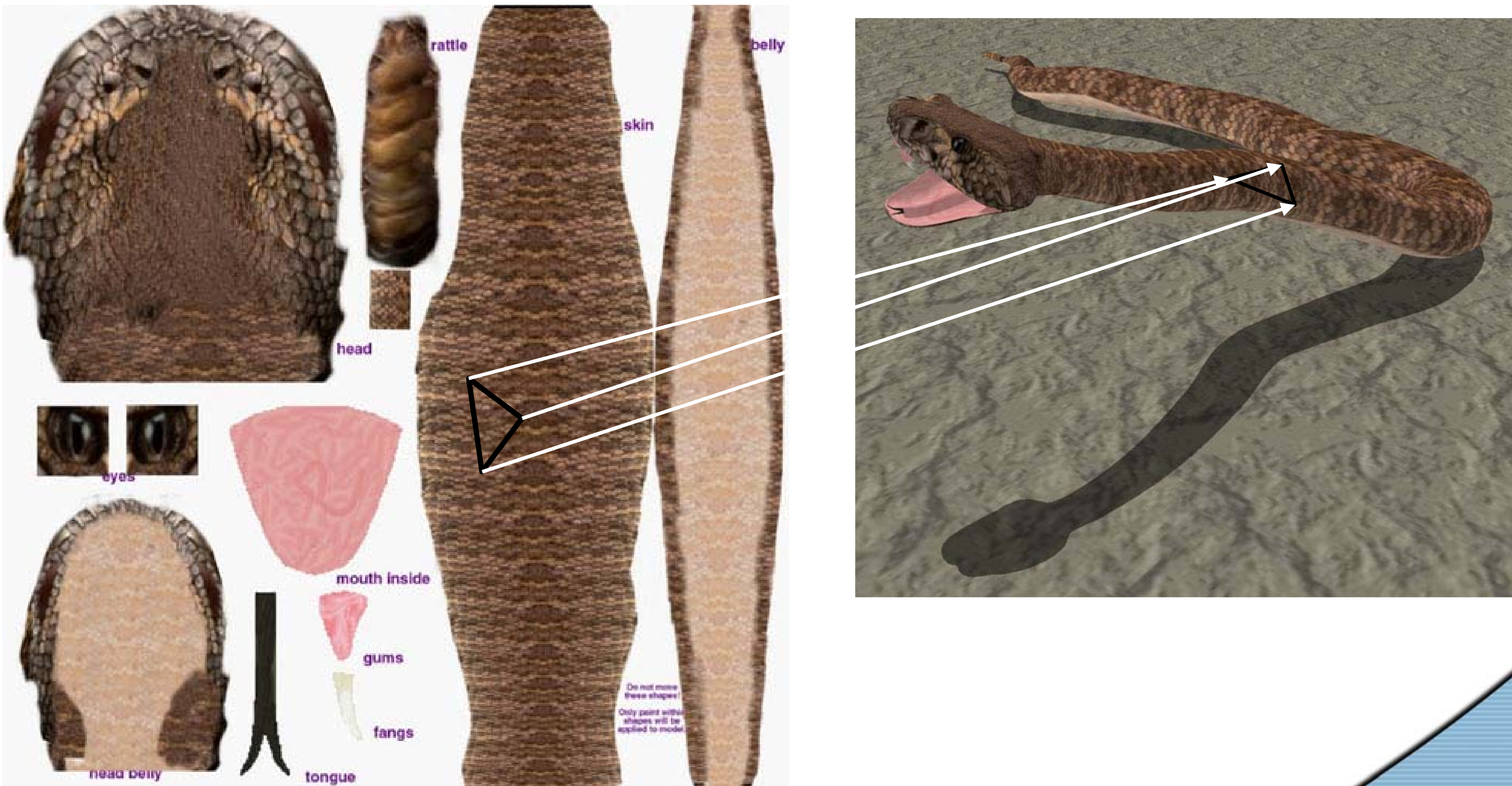


Examples – Image Textures



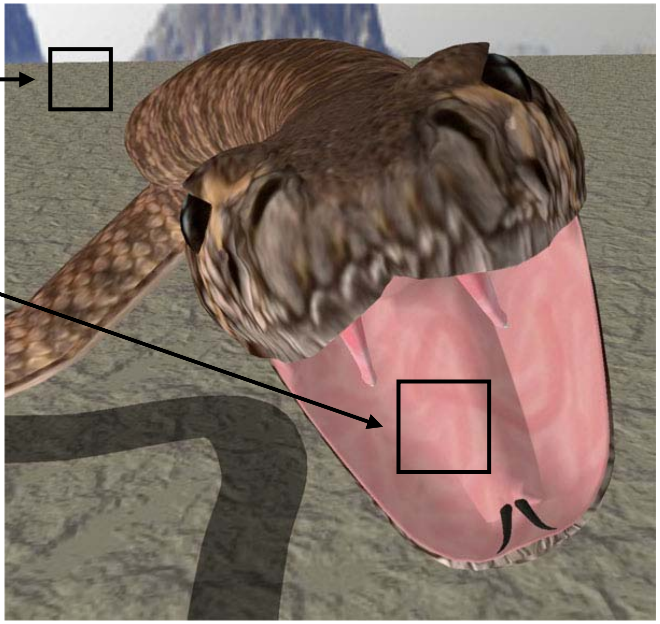
Flytastic II (www.endoxon.ch)

Examples – Image Textures



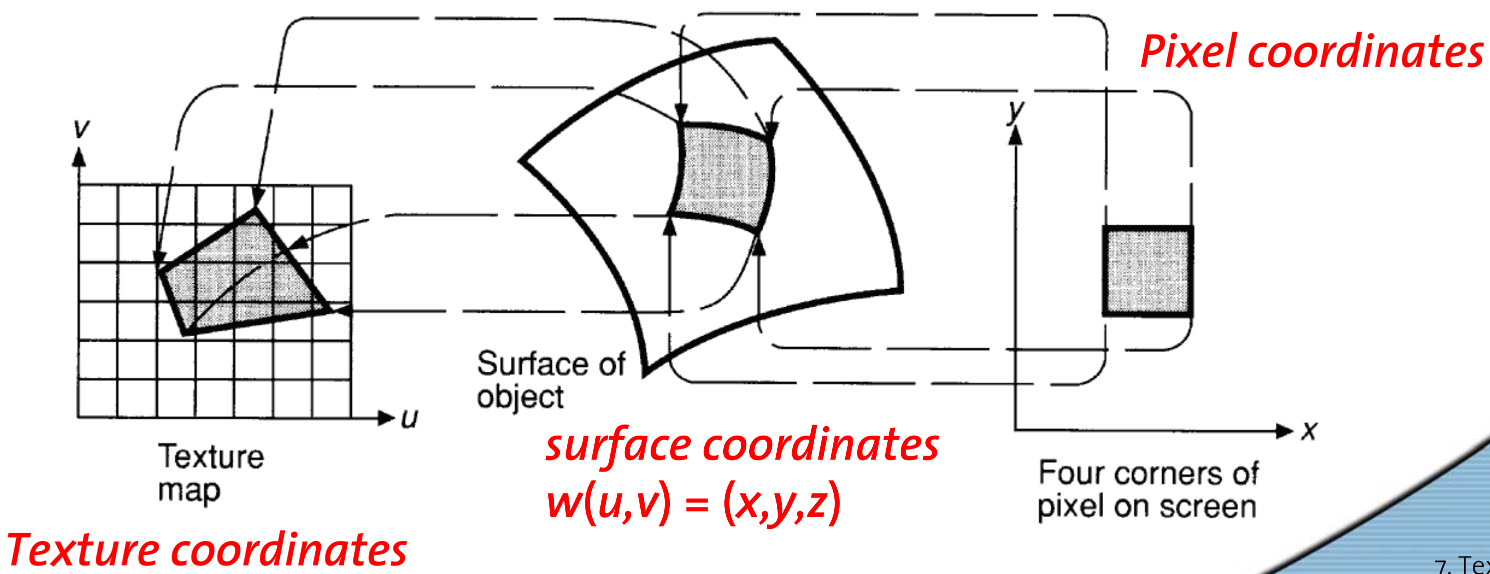
Issues

- Definition of texture coordinates
- Surface parameterization
- Anti aliasing
- Texture filtering
- Level-of-Detail
- Hardware acceleration



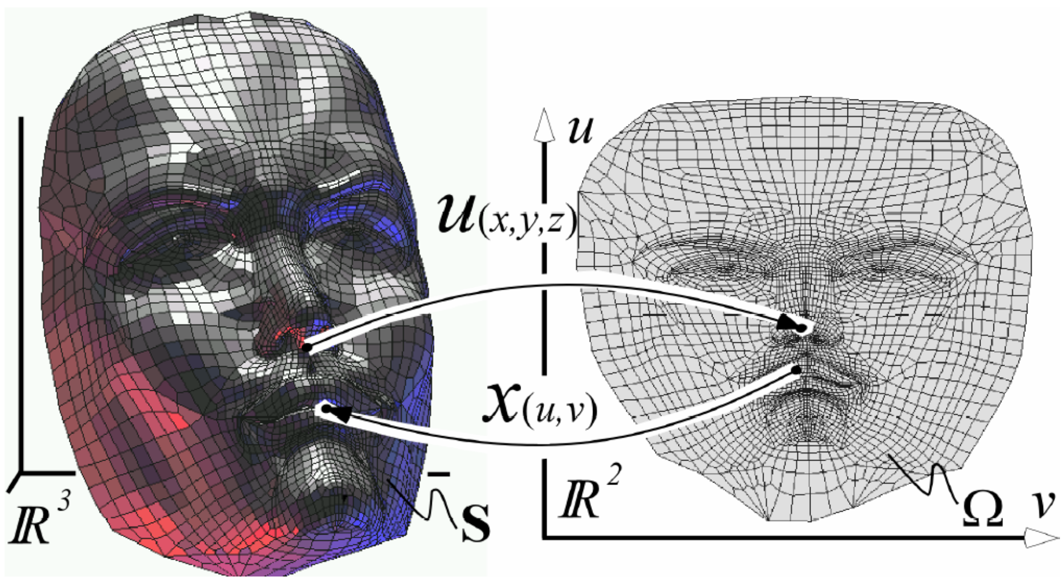
Concept of Texture Mapping

- Find mappings between different coordinate systems
- Invert transformation from texture coordinates to image pixel



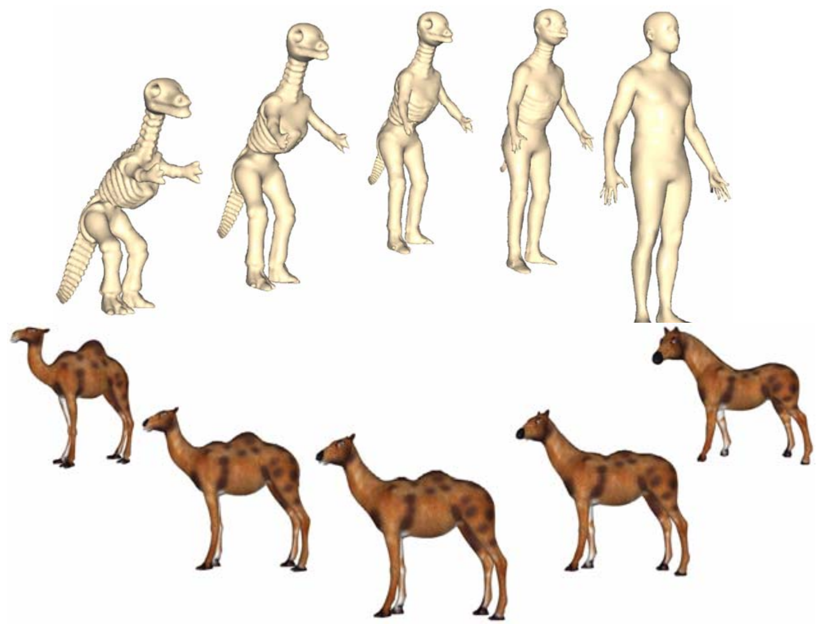
Parameterization

- Find a one-to-one mapping between given surface and 2D parameter domain



Parameterization

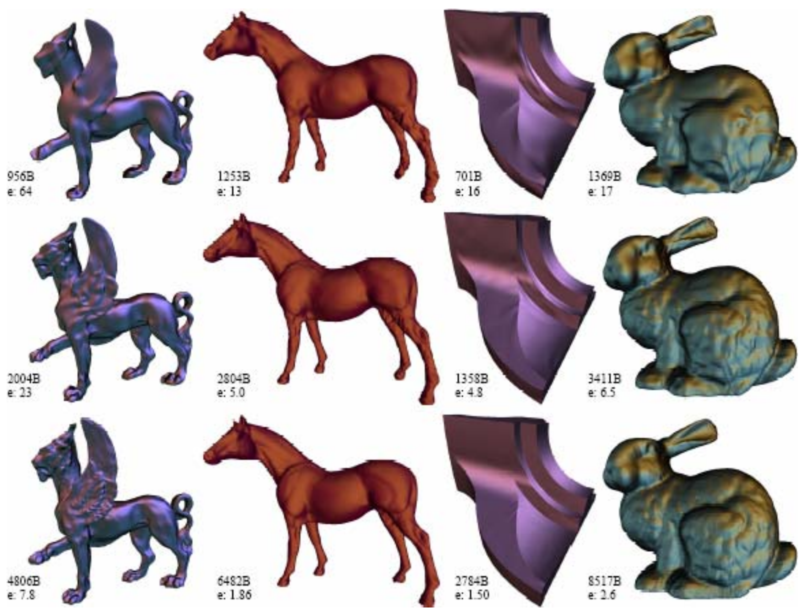
- Fundamental concept in graphics
- Many different applications
  - Morphing



Kraevoy, Sheffer: *Cross-Parameterization and Compatible Remeshing of 3D Models*, SIGGRAPH, 2004

Parameterization

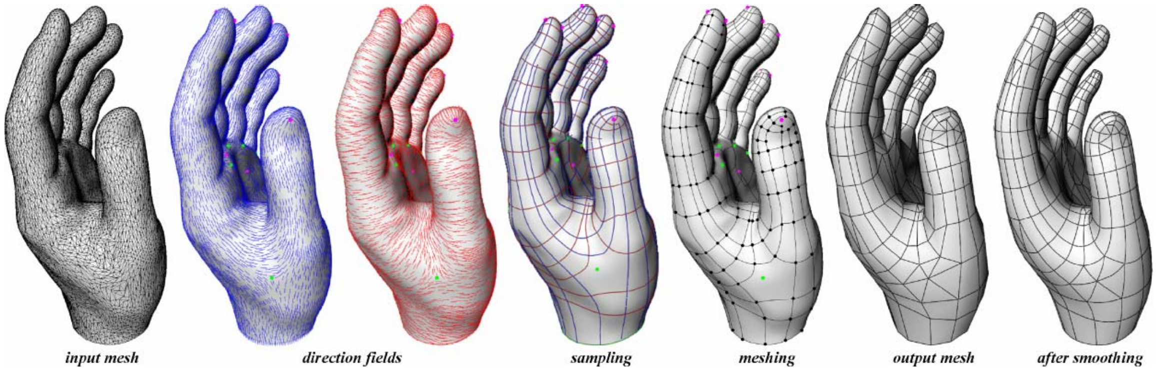
- Fundamental concept in graphics
- Many different applications
  - Morphing
  - Compression



Khodakovsky, Schroeder, Sweldens: *Progressive Geometry Compression*, SIGGRAPH, 2000

Parameterization

- Fundamental concept in graphics
- Many different applications
  - Morphing
  - Compression
  - Remeshing



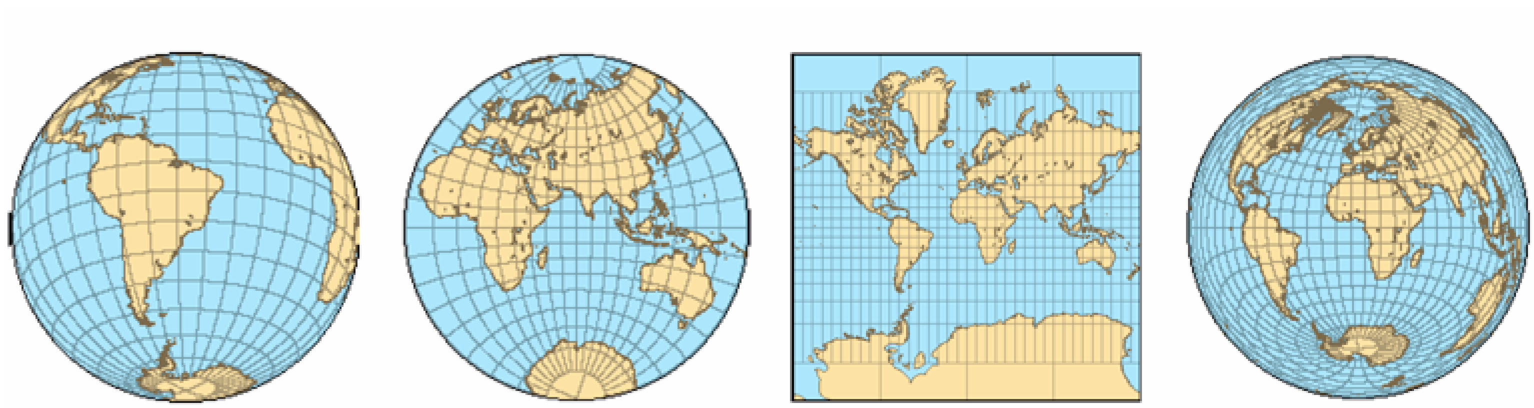
Alliez, Cohen-Steiner, Devillers, Levy, Desbrun: *Anisotropic Polygonal Remeshing*, SIGGRAPH, 2003

Parameterization

- Fundamental concept in graphics
- Many different applications
  - Morphing
  - Compression
  - Remeshing
  - Texture Mapping

Some History

- Cartography



orthographic — stereographic — Mercator — Lambert

stereographic ↑ preserves angles = conformal

Lambert ↑ preserves area = equiareal

Floater, Hormann: *Surface Parameterization: A Tutorial and Survey*, Advances in Multiresolution for Geometric Modeling, 2005

Analytical 3D Sufraces

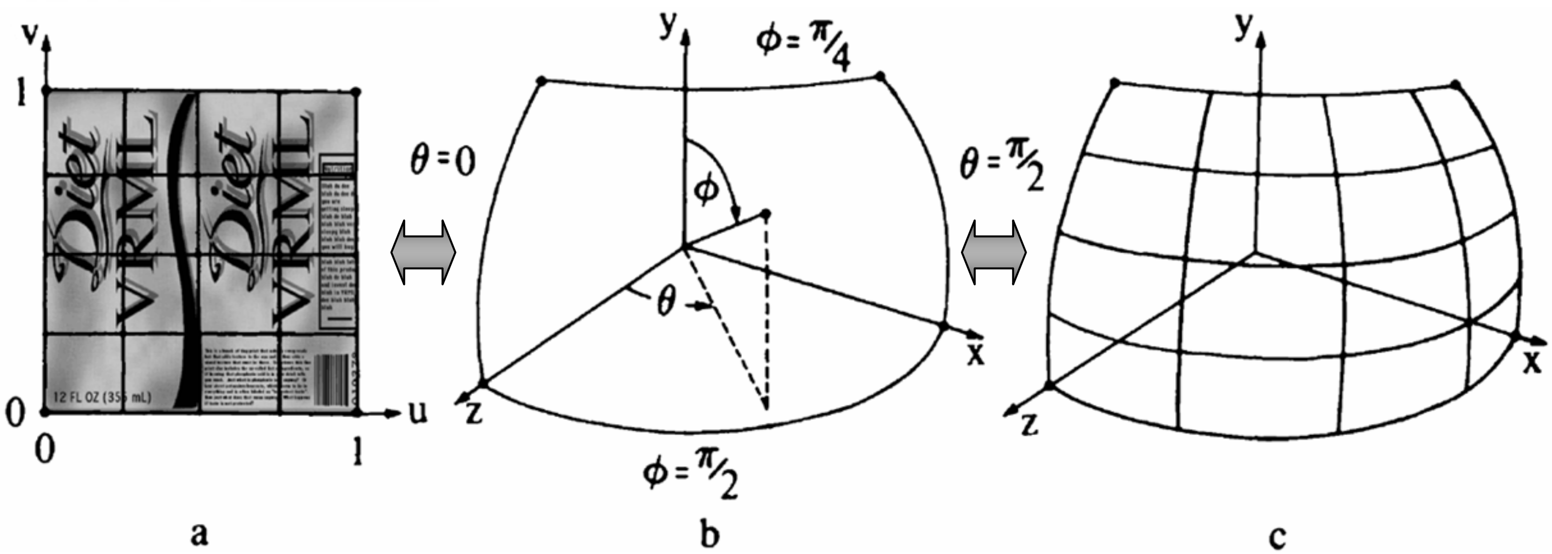
1. Key to texture mapping: ***Parameterization***

$$\begin{bmatrix} s \\ t \end{bmatrix} \rightarrow \begin{bmatrix} x(s,t) \\ y(s,t) \\ z(s,t) \end{bmatrix} \qquad \text{sphere:} \quad \begin{bmatrix} \theta \\ \phi \end{bmatrix} \rightarrow \begin{bmatrix} \sin\theta \sin\phi \\ \cos\phi \\ \cos\theta \sin\phi \end{bmatrix}$$

2. Map parameters to texture coordinates

$$\begin{bmatrix} s \\ t \end{bmatrix} \rightarrow \begin{bmatrix} u(s,t) \\ v(s,t) \end{bmatrix} \qquad \text{inverse:} \quad \begin{bmatrix} u \\ v \end{bmatrix} \rightarrow \begin{bmatrix} s(u,v) \\ t(u,v) \end{bmatrix}$$

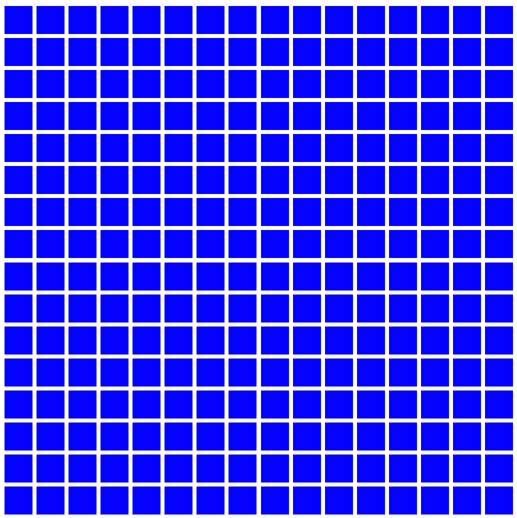
Mapping a texture onto a sphere

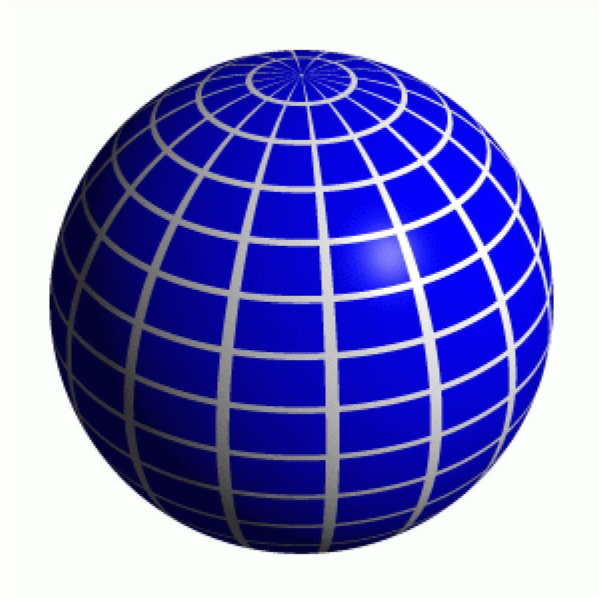


- Use linear map

$$\begin{bmatrix}\theta \\ \phi\end{bmatrix} = \begin{bmatrix}Au + B \\ Cv + D\end{bmatrix} \Rightarrow \begin{bmatrix}\theta \\ \phi\end{bmatrix} = \begin{bmatrix}\pi / 2 \cdot u \\ -\pi / 4 \cdot v + \pi / 2\end{bmatrix}$$

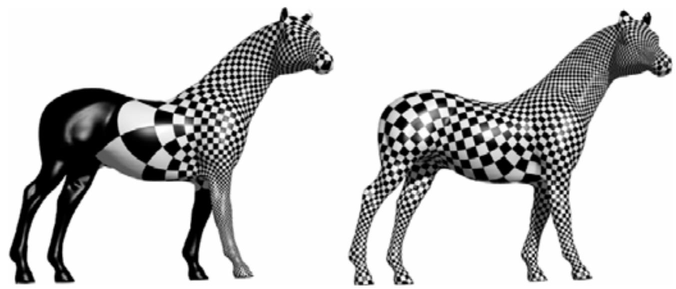
Example



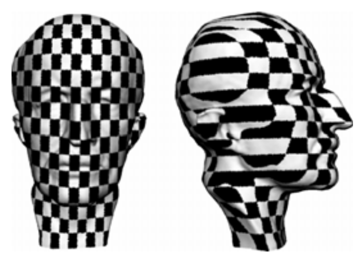


Desirable Properties

- Low distortion



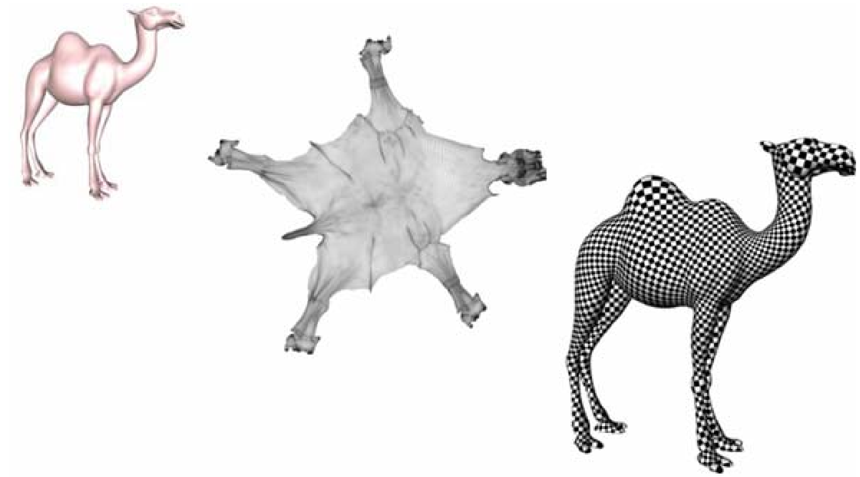
- Bijective mapping



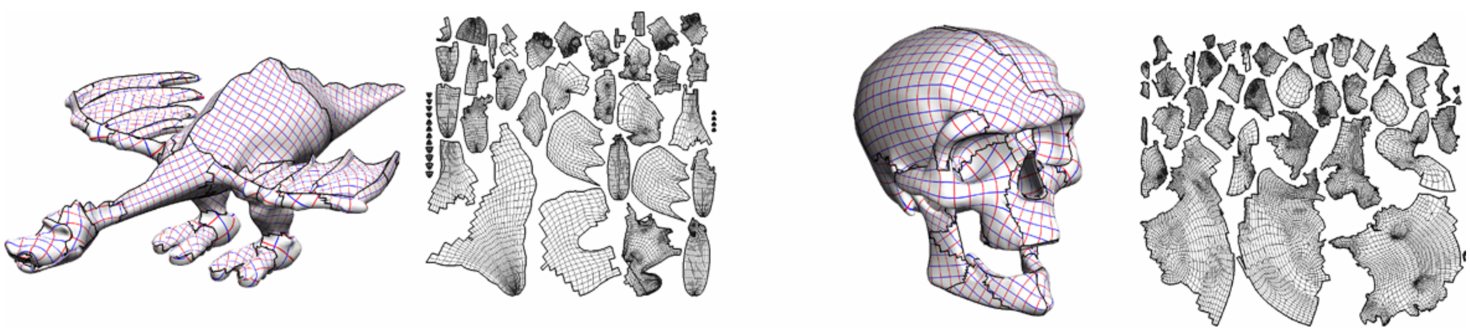
- Efficiently computable

Additional Issues

- Finding cuts



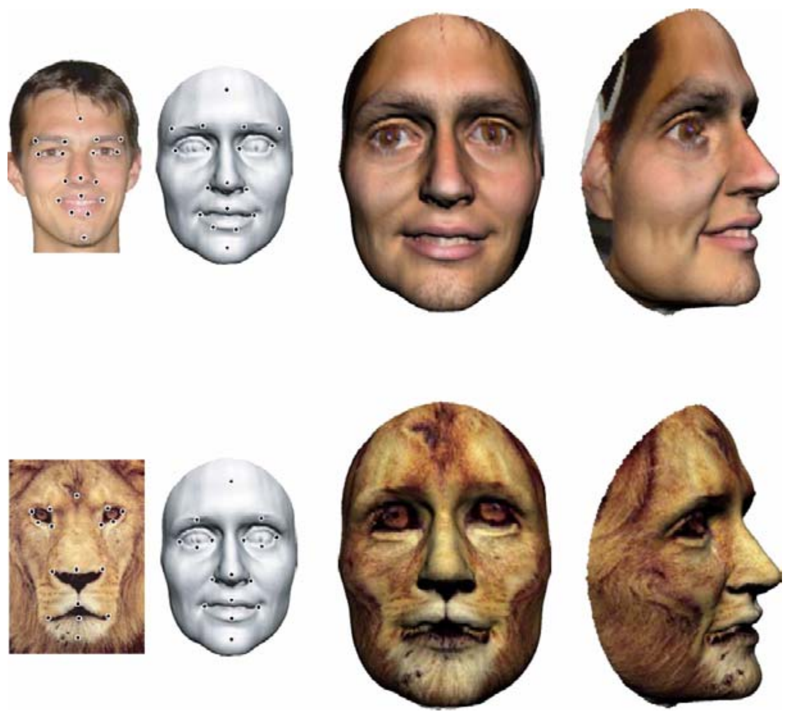
- Texture Atlases



Levy, Petitjean, Ray, Maillot: *Least Squares Conformal Maps for Automatic Texture Atlas Generation*, SIGGRAPH, 2002

Additional Issues

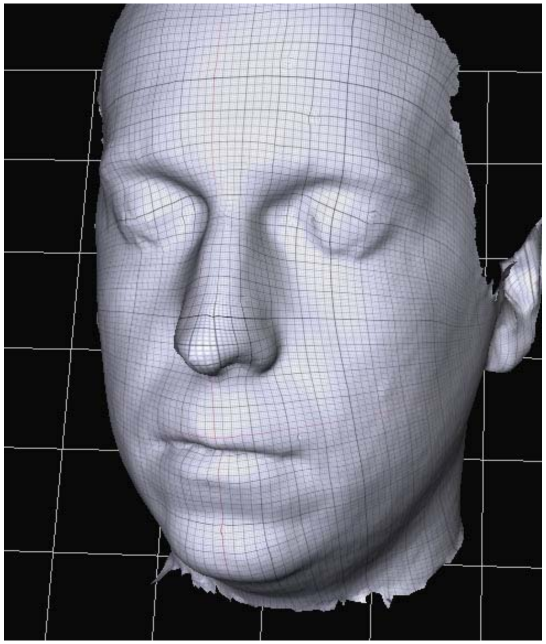
- Constraint Texture Mapping



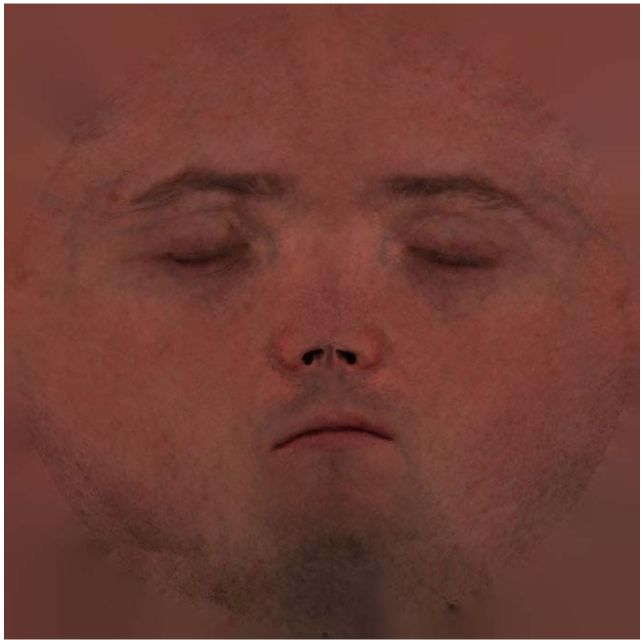
→ Demo

Texture Map

- Texture map corresponds to parameterization



stretched at nose tip



compressed at nose tip

Tim Weyrich
et. al.

Parametrized Triangle Mesh

OBJ Files:

```
v 0.131171 -0.113469 0.178314
v 0.130945 -0.114951 0.182474
v 0.130916 -0.115792 0.185402
…
vt 0.538446 0.4275
vt 0.550132 0.41427
vt 0.546491 0.427631
…
vn 0.609697 0.486474 0.625789
vn 0.799934 0.334347 0.498315
vn 0.942394 0.131824 0.307435
…
f 22/209/22 220/210/220 221/211/221
f 21/213/21 219/214/219 220/210/229
f 253/203/253 219/214/219 21/213/21
…
```

Vertex positions

Texture coordinates

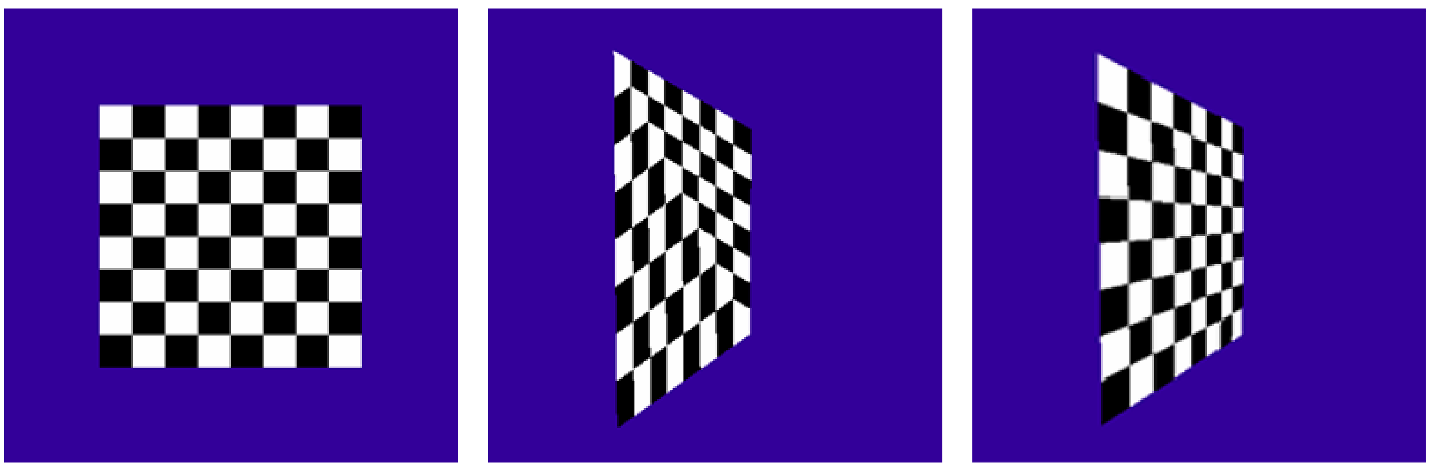
Normals

Faces (triangles)

`coordNr/texNr/normalNr`

Rasterization

- From texture coordinates of vertices to texture coordinates of pixels
- Linear interpolation in screen-space (as in Gouraud shading):

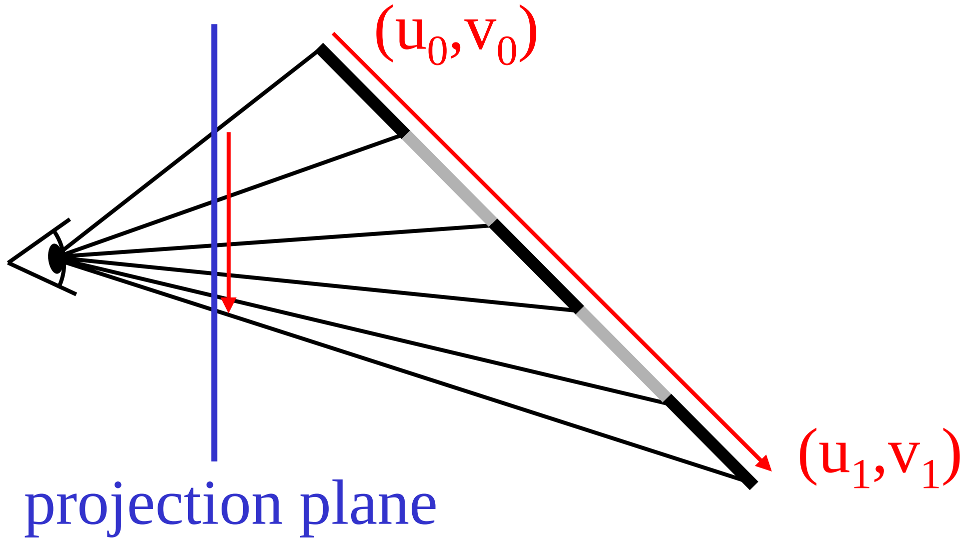


Images by Fredo Durand

texture source what we get what we want

Perspective Interpolation

- Linear variation in world coordinates yields non-linear variation in screen coordinates:

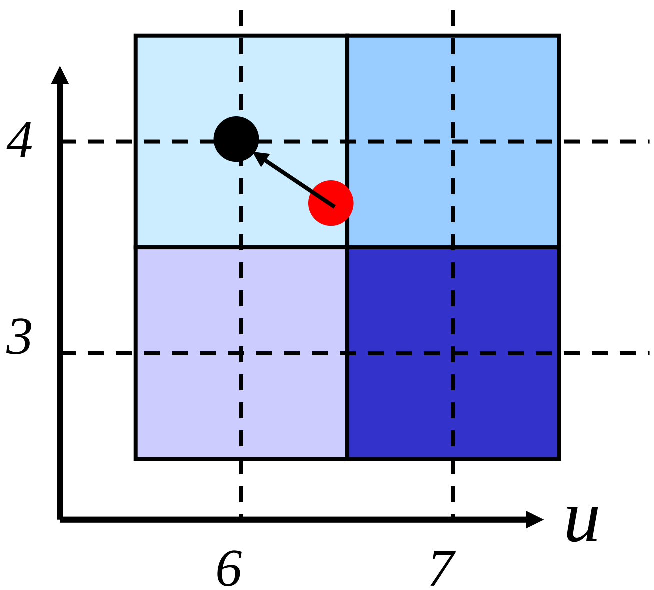


- Perspective interpolation implemented in today's graphics cards

Texture Filtering

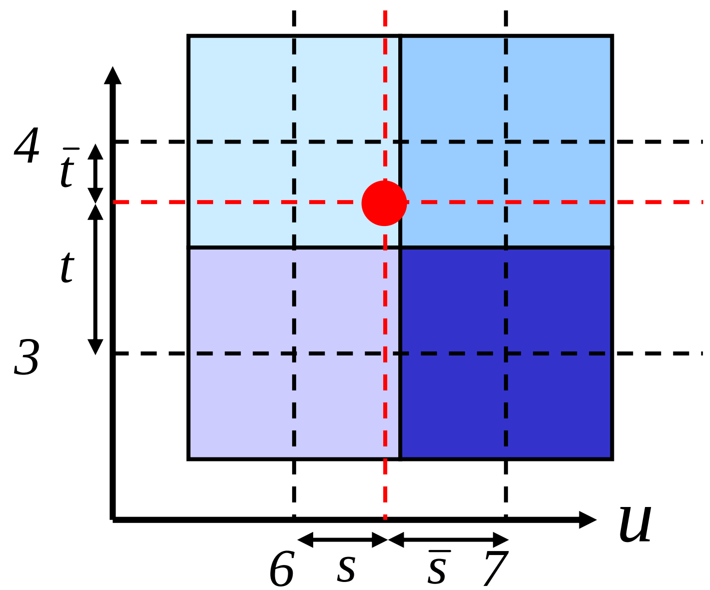
- (u,v) are real pixel coordinates, e.g. $(6.4, 3.7)$:

nearest



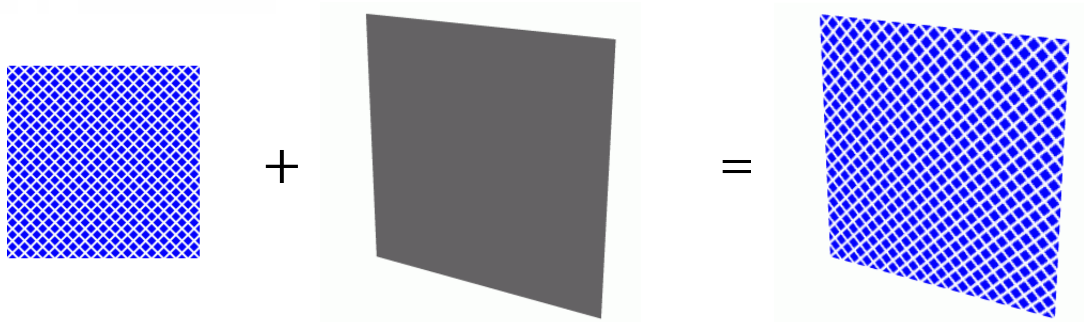
$color = map[6,4]$

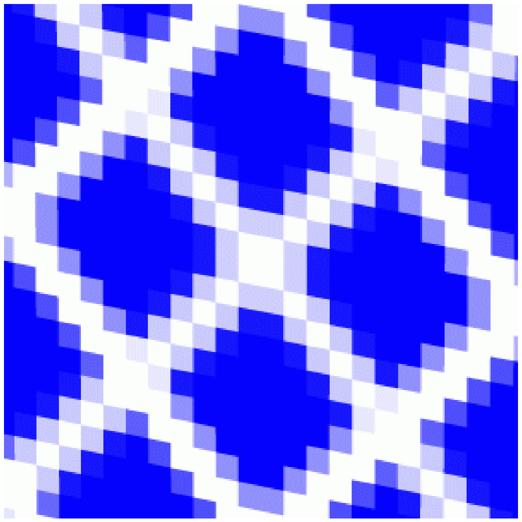
bi-linear



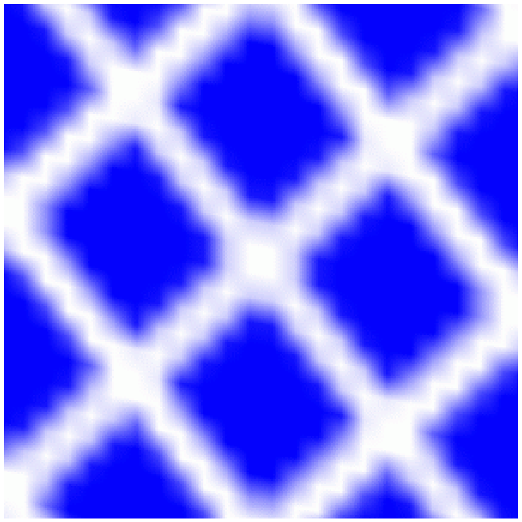
$$color = \bar{s}\cdot\bar{t}\cdot map[6,3] + s\cdot\bar{t}\cdot map[7,3] + \bar{s}\cdot t\cdot map[6,4] + s\cdot t\cdot map[7,4]$$

Texture Filtering





nearest



bi-linear

Texture Mapping in OpenGL

```
loadImage(&texture_data);
glGenTextures(1, &texId);
glBindTexture(GL_TEXTURE_2D, texId);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB,
     w, h, 0, GL_RGB, GL_UNSIGNED_BYTE,
    texture_data);
…

glBindTexture(GL_TEXTURE_2D, texId);
glBegin(GL_TRIANGLES);
    glTexCoord2f(u0,v0); glVertex(x0,y0,z0);
    glTexCoord2f(u1,v1); glVertex(x1,y1,z1);
    glTexCoord2f(u2,v2); glVertex(x2,y2,z2);
glEnd();
```

$$w = 2^n, h = 2^m$$

$$u, v \in [0 \ldots 1]$$

→ Tutor