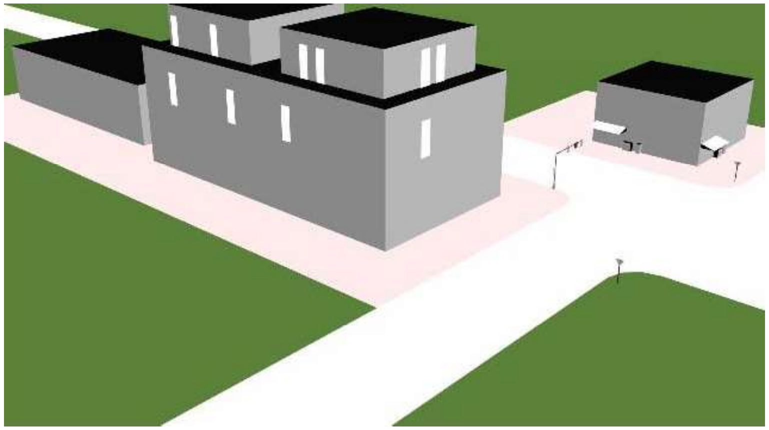
Large triangles

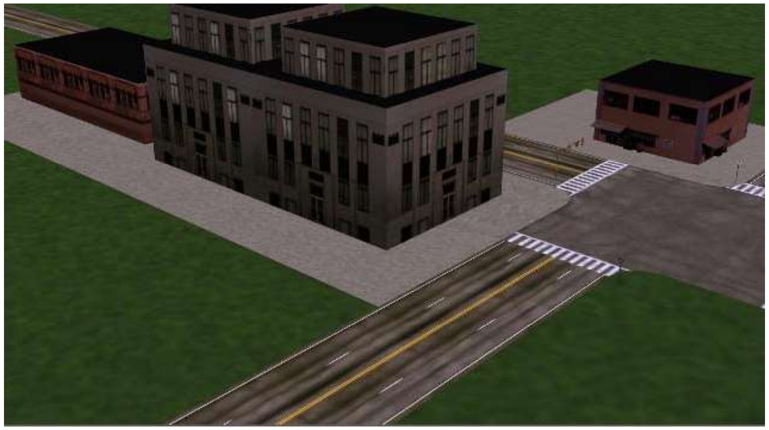
- Advantages
  - Often sufficient for simple geometry
  - Fast to render
- Disadvantages
  - Per vertex colors look boring and computer-generated



Based on slides courtesy of Jurgen Schulze

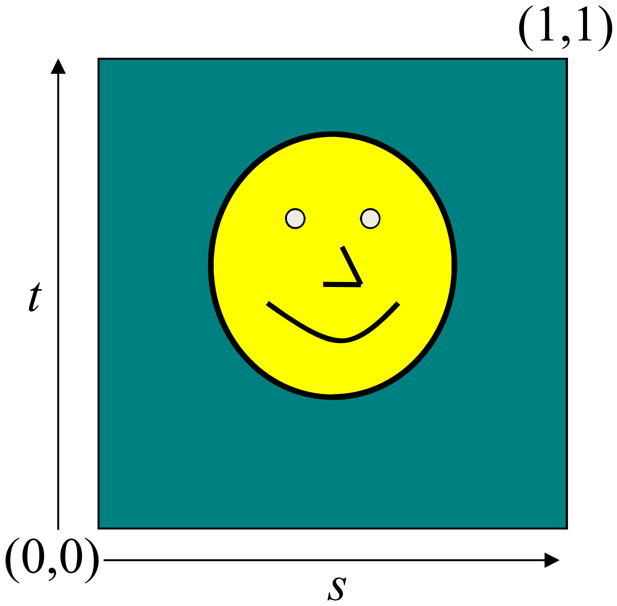
Texture mapping

- Map textures (images) onto surface polygons
- Same triangle count, much more realistic appearance



Texture mapping

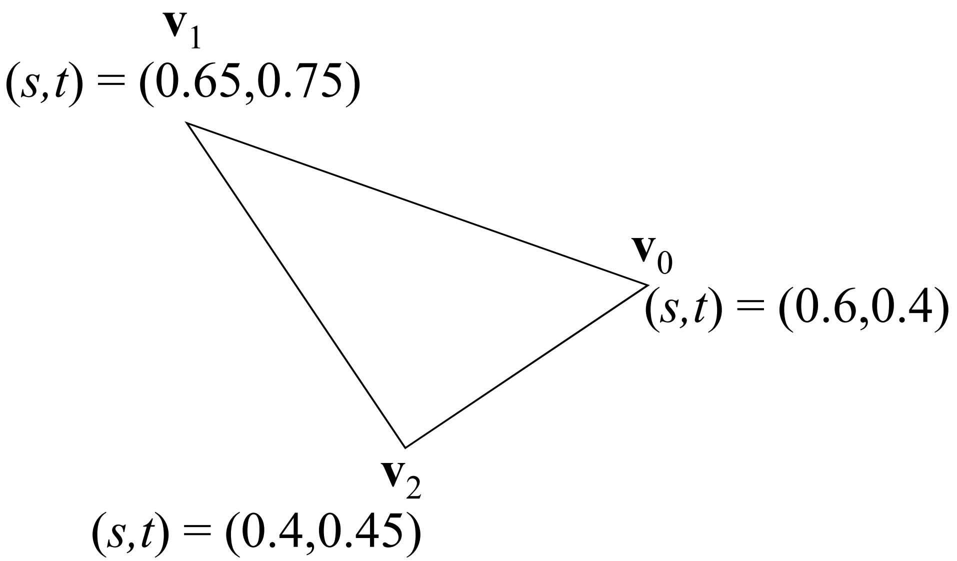
- Objective: map locations in texture to locations on 3D geometry
- Texture coordinate space
  - Texture pixels (texels) have texture coordinates (*s*,*t*)
- Convention
  - Bottom left corner of texture is at (*s*,*t*) = (0,0)
  - Top right corner is at (*s*,*t*) = (1,1)



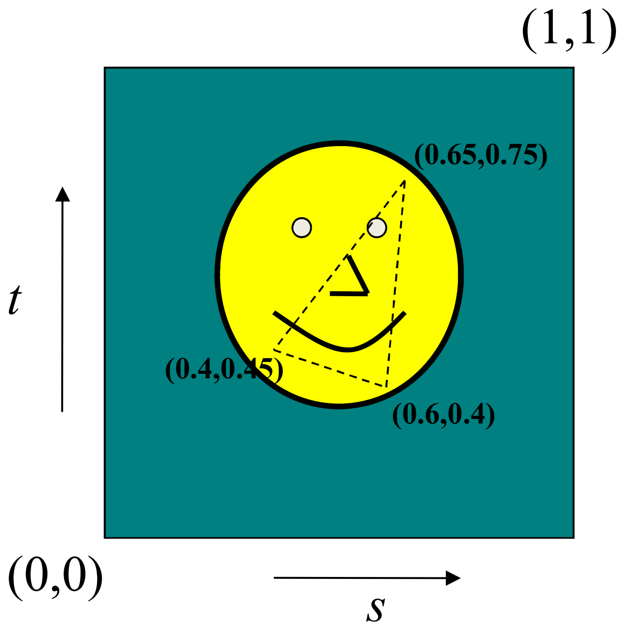
Texture coordinates

Texture mapping

- Store 2D texture coordinates *s*,*t* with each triangle vertex



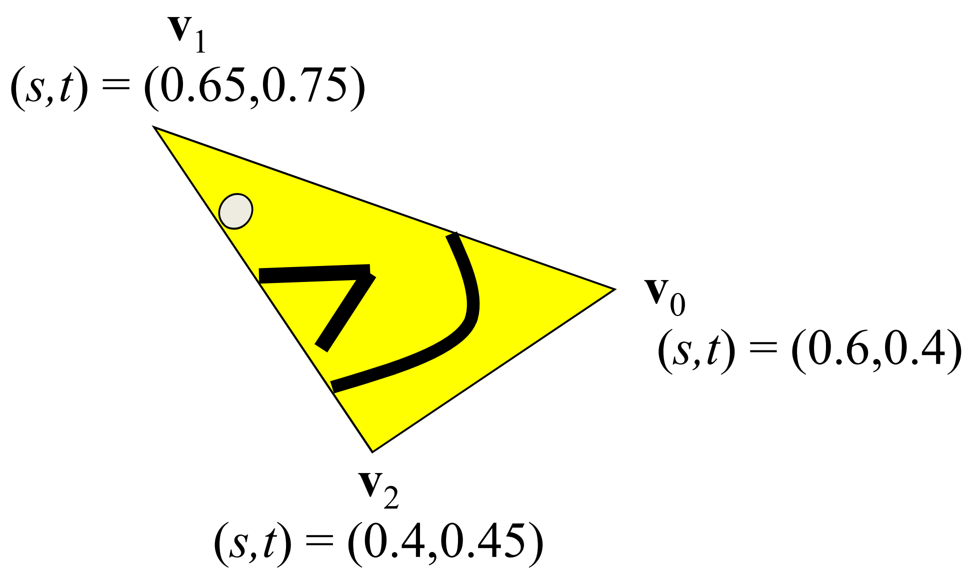
Triangle in any space before projection



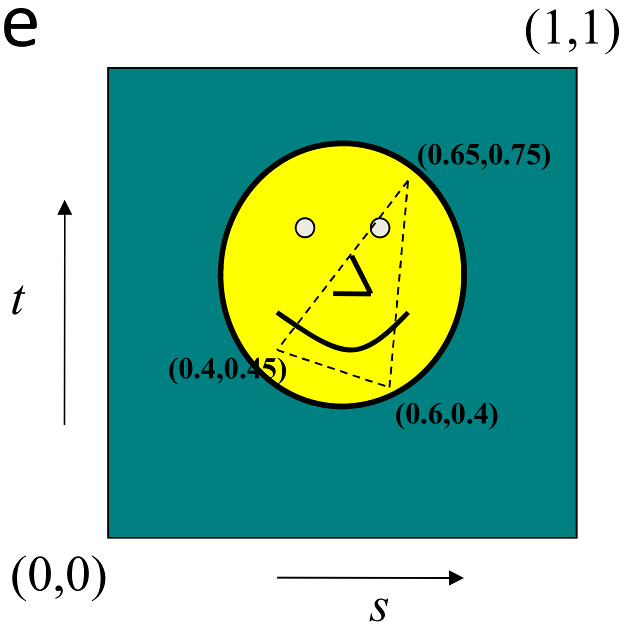
Texture coordinates

Texture mapping

- Each point on triangle gets color from its corresponding point in texture

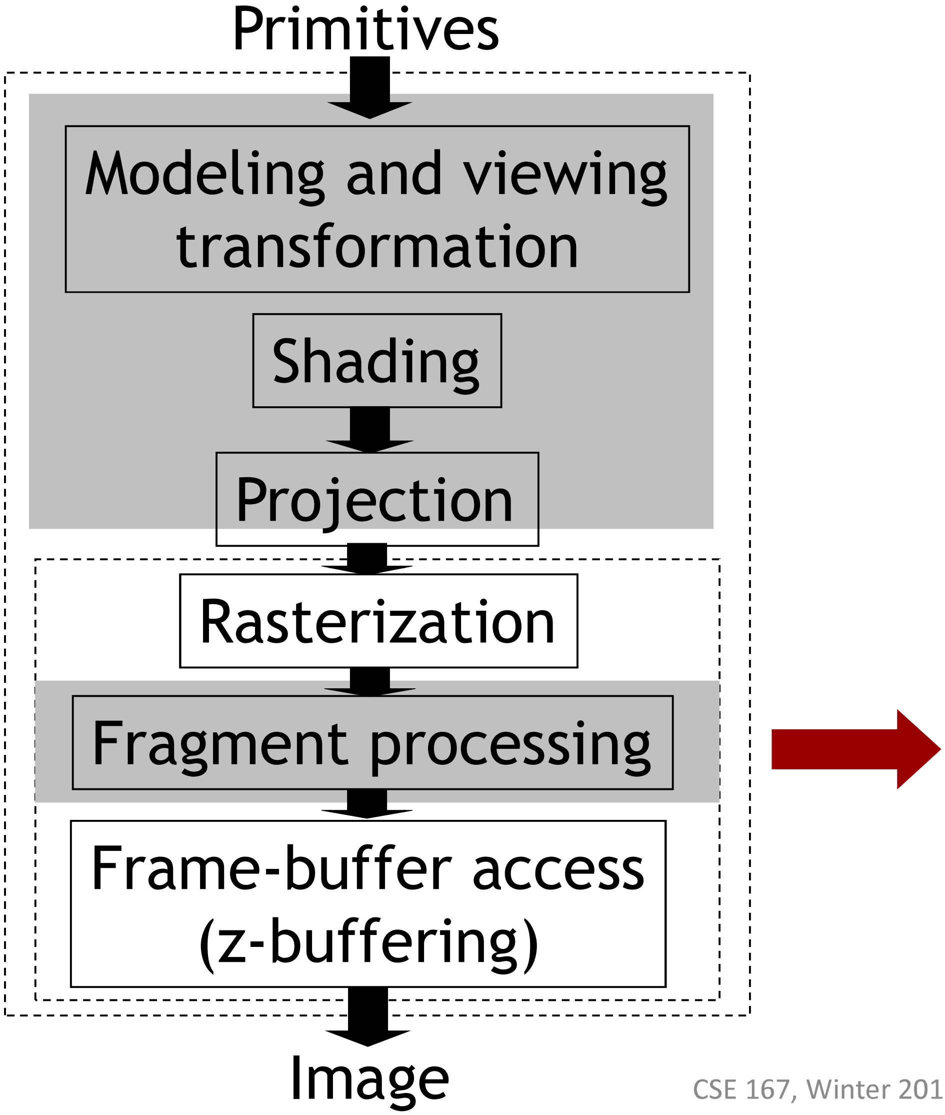


Triangle in any space before projection



Texture coordinates

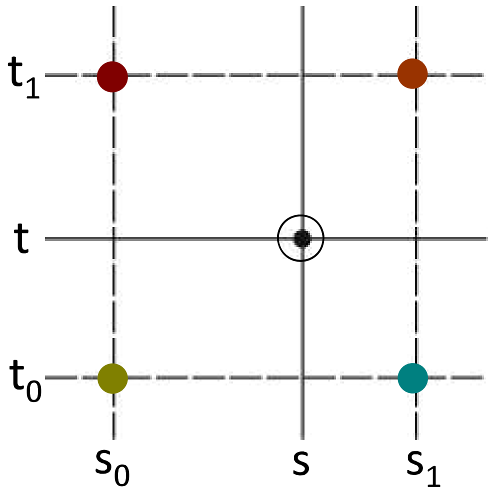
Texture mapping


Primitives
Modeling and viewing transformation
Shading
Projection
Rasterization
Fragment processing
Frame-buffer access (z-buffering)
Image

Includes texture mapping

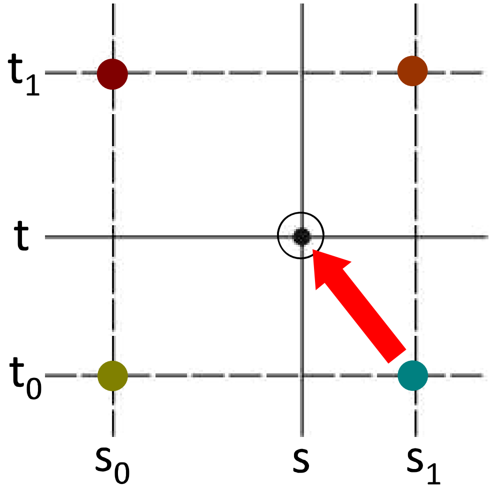
Texture look-up

- Given texture coordinates (s,t) at current pixel
- Closest four texels in texture space are at (s_0,t_0), (s_1,t_0), (s_0,t_1), (s_1,t_1)
- How to compute pixel color? Interpolate



Nearest neighbor interpolation

- Use color of closest texel



- Simple, but low quality and aliasing

Bilinear interpolation

1. Linear interpolation horizontally

   Ratio in *s* direction is $r_s = \dfrac{s - s_0}{s_1 - s_0}$

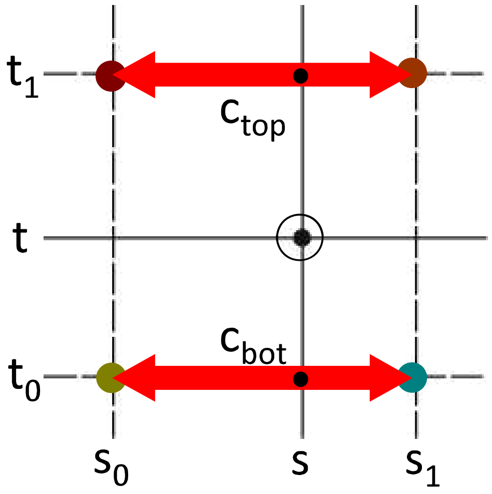
   c_{top} = tex(s_0,t_1) (1-r_s) + tex(s_1,t_1) r_s

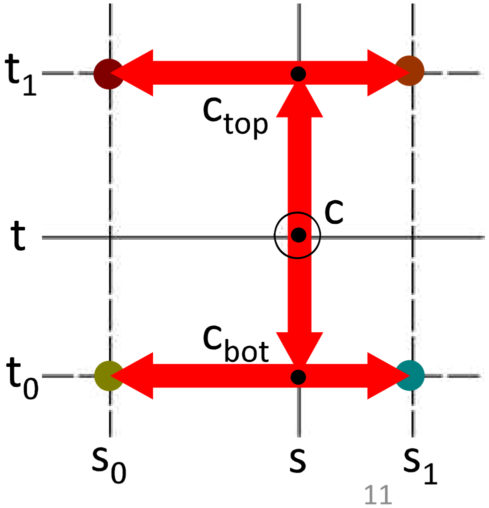
   c_{bot} = tex(s_0,t_0) (1-r_s) + tex(s_1,t_0) r_s

2. Linear interpolation vertically

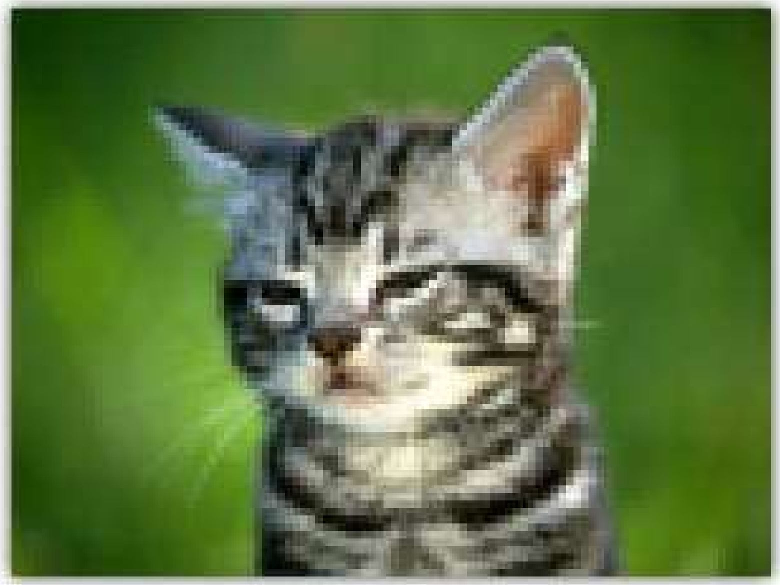
   Ratio in *t* direction is $r_t = \dfrac{t - t_0}{t_1 - t_0}$

   $c = c_{\text{bot}}\,(1\text{-}r_t) + c_{\text{top}}\,r_t$

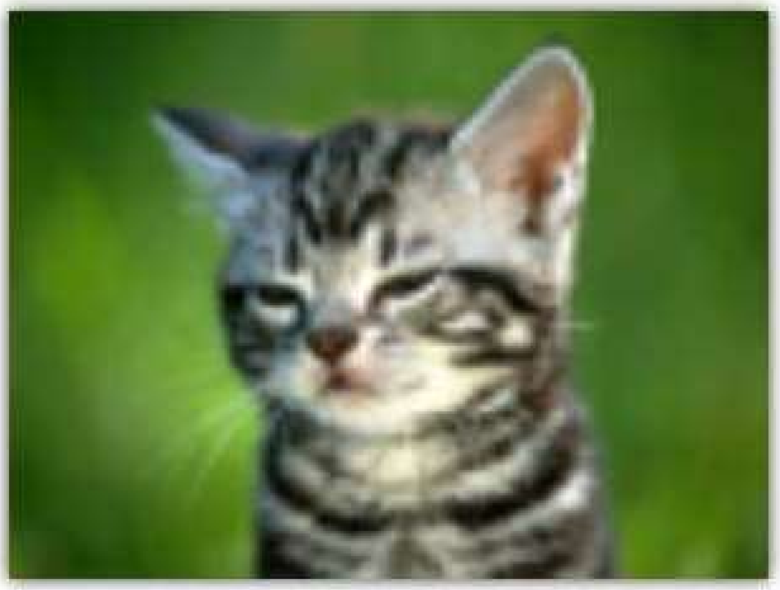




Interpolation



Nearest neighbor



Bilinear

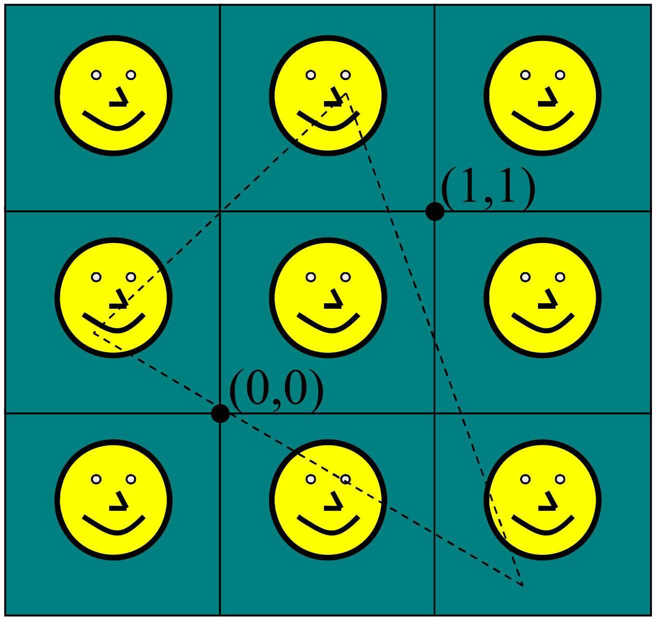
- OpenGL
  - – GL_NEAREST: Nearest neighbor interpolation
  - – GL_LINEAR: Bilinear interpolation

Wrapping

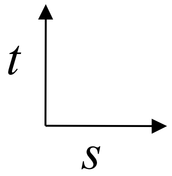
- Texture image extends from [0,0] to [1,1] in texture space
  - What if (*s*,*t*) texture coordinates are beyond that range? Wrap
    - Repeat (and optionally mirror) texture
    - Clamp texture

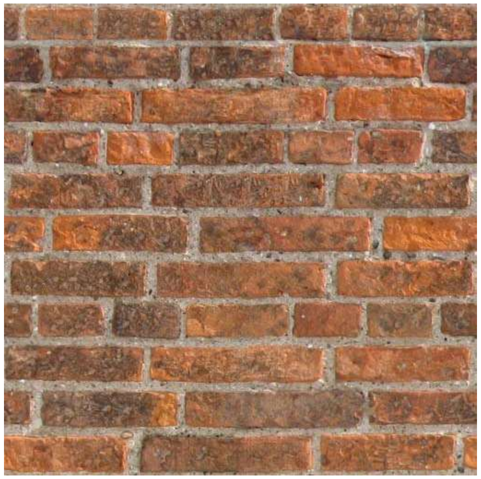
Wrapping: repeat

- Repeat the texture
  - Creates discontinuities at edges, unless texture is designed to line up



Texture Space

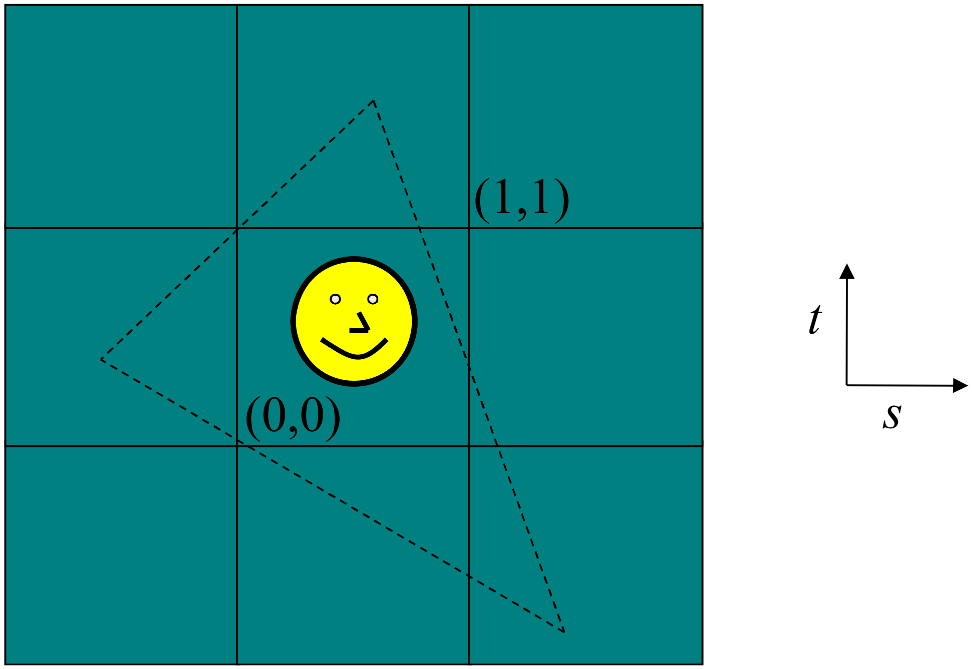




Seamless brick wall texture
(by Christopher Revoir)

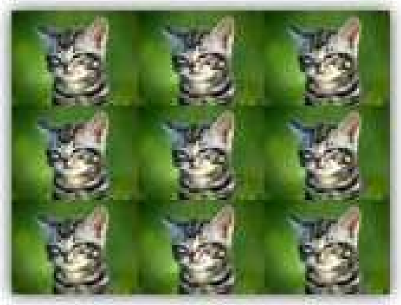
Wrapping: clamp

- Use edge or specified color everywhere outside data range [0..1]

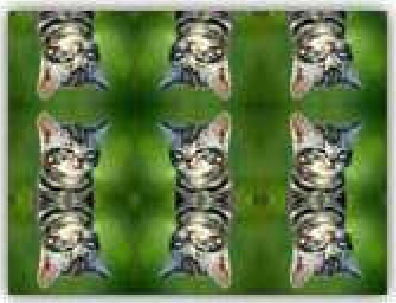


Texture Space

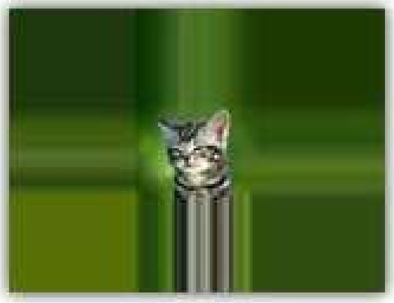
Wrapping



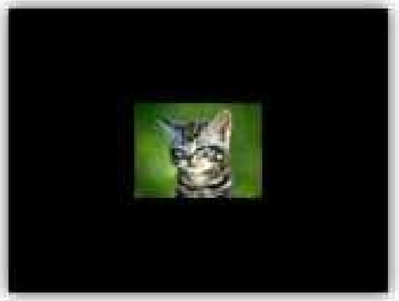
Repeat



Repeat and mirror



Clamp to
edge color



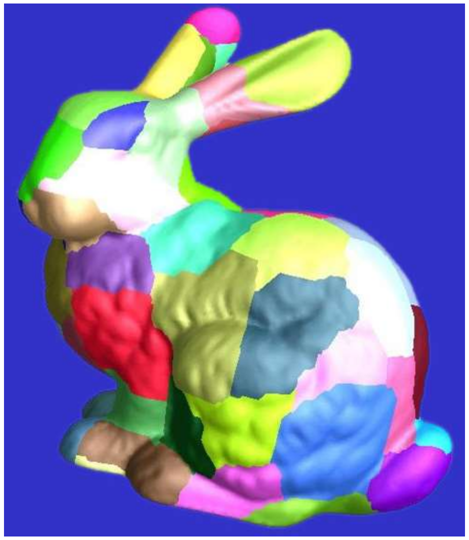
Clamp to
specified color

- OpenGL
  - GL_REPEAT
  - GL_MIRRORED_REPEAT
  - GL_CLAMP_TO_EDGE: repeats last pixel in the texture
  - GL_CLAMP_TO_BORDER: requires setting border color

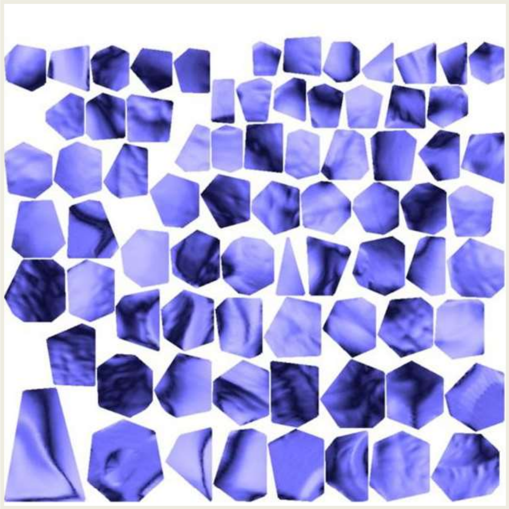
Texture coordinates

- What if texture extends across multiple polygons? Parameterize surface
  - Mapping between 3D positions on surface and 2D texture coordinates
    - Defined by texture coordinates of triangle vertices
  - Example mappings
    - Cylindrical
    - Spherical
    - Orthographic
    - Cube
    - Parametric
    - Skin

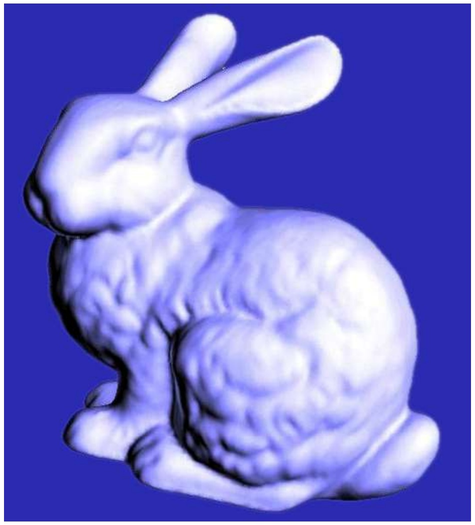
Texture atlas



charts



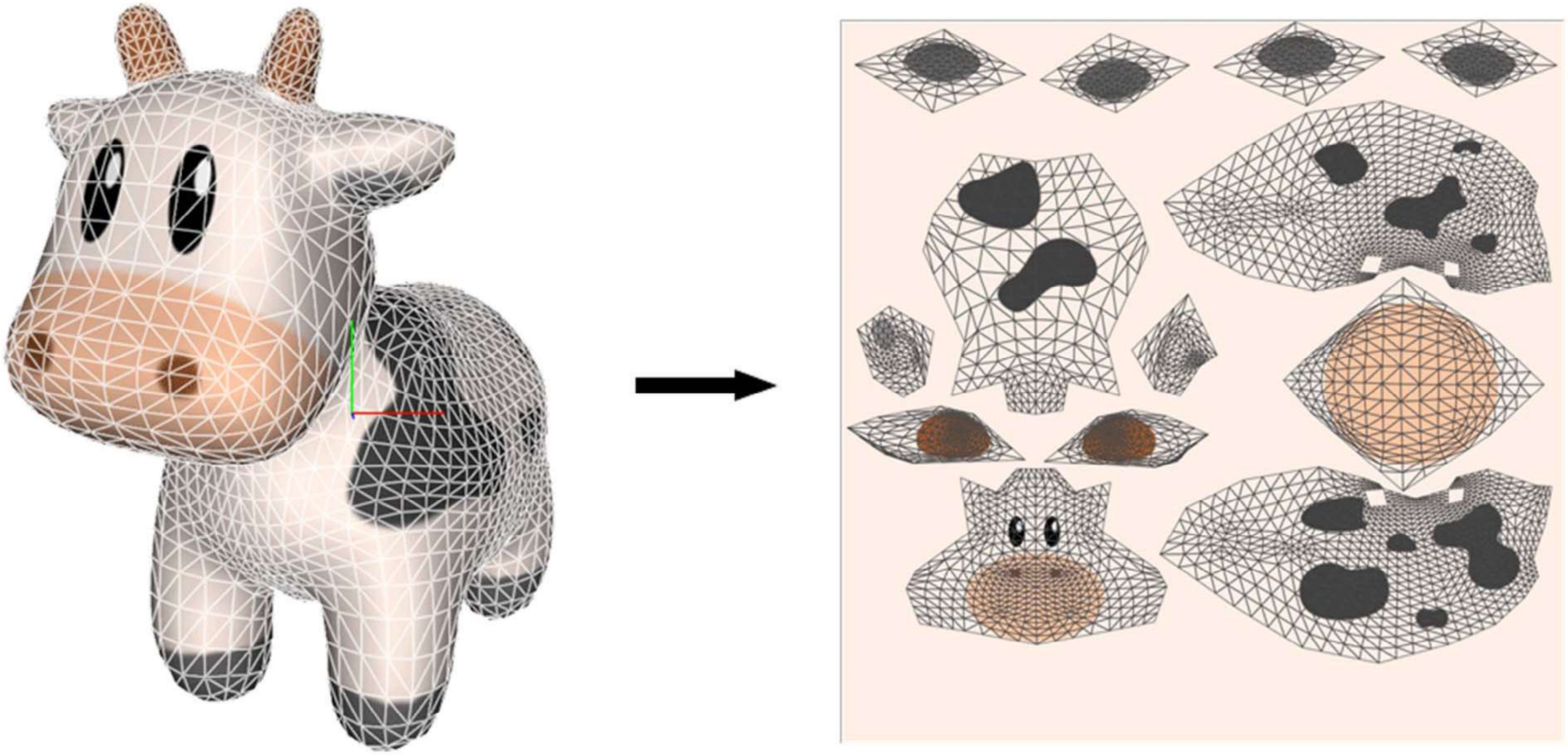
atlas



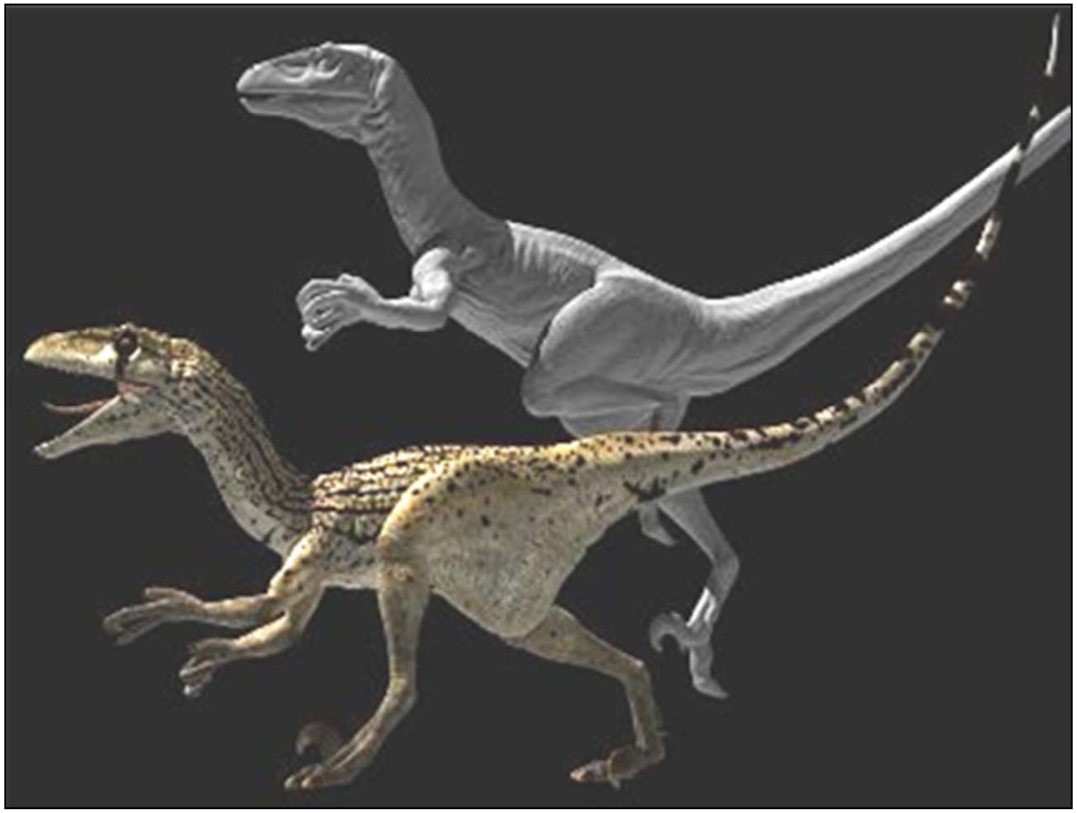
surface

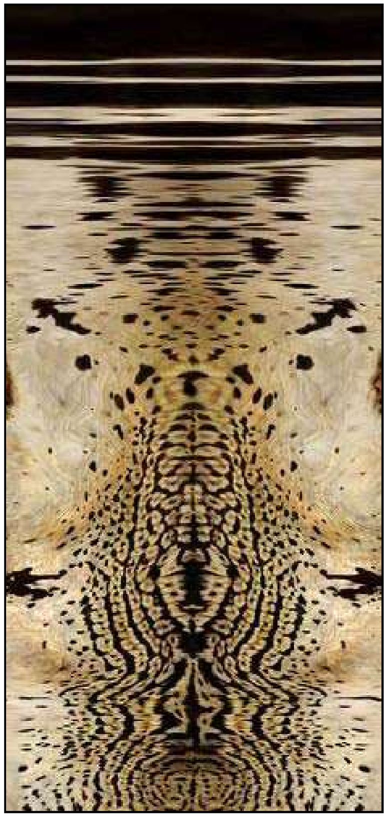
[Sander2001]

Skin mapping

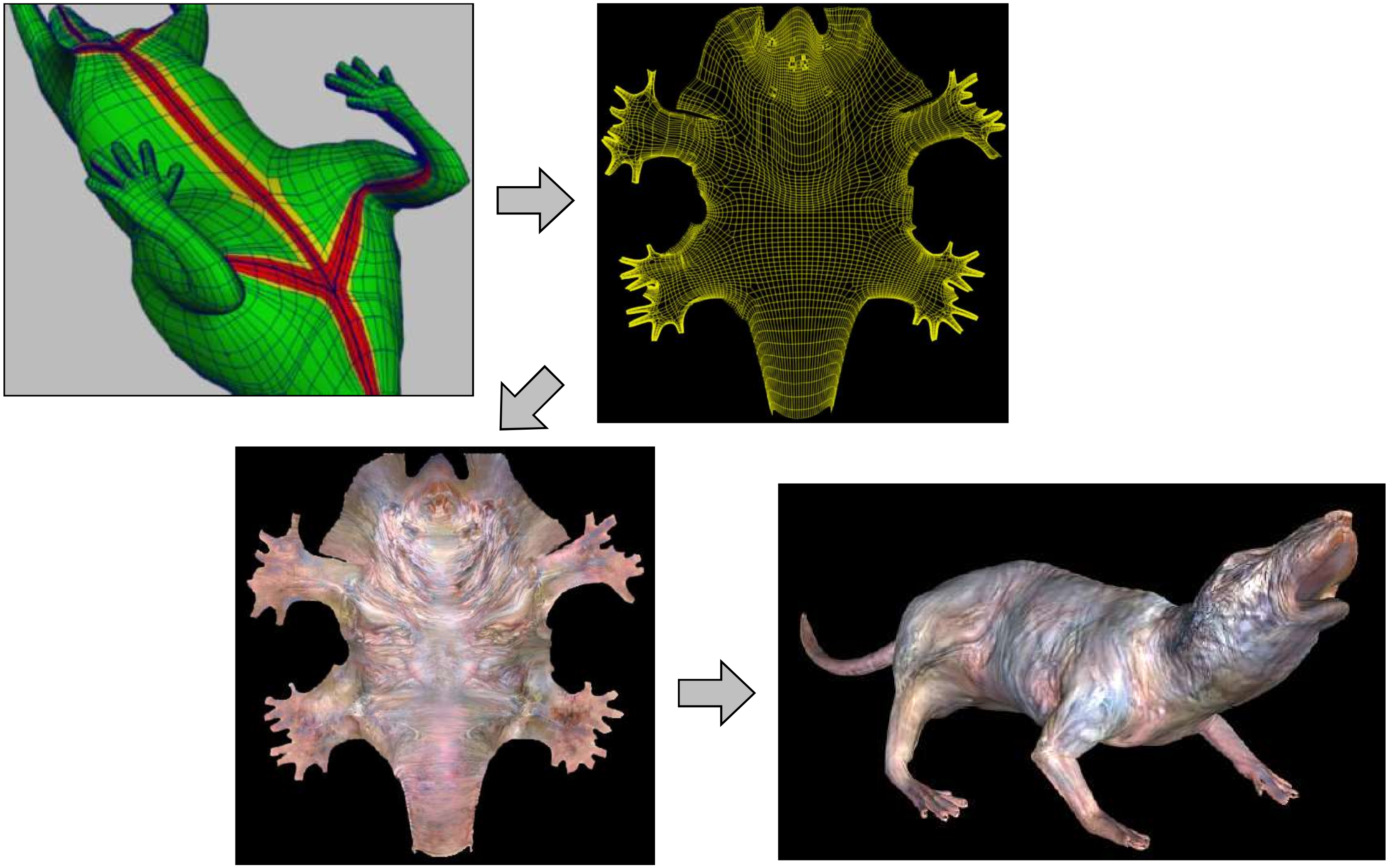


Skin mapping

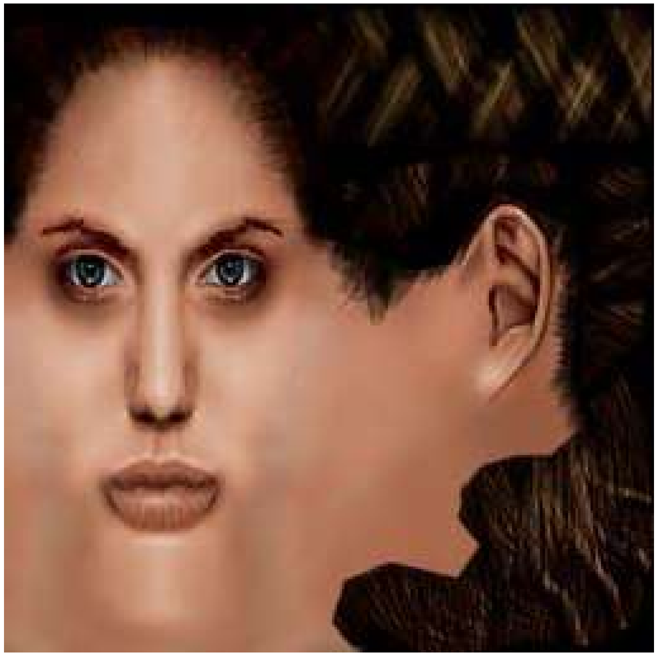


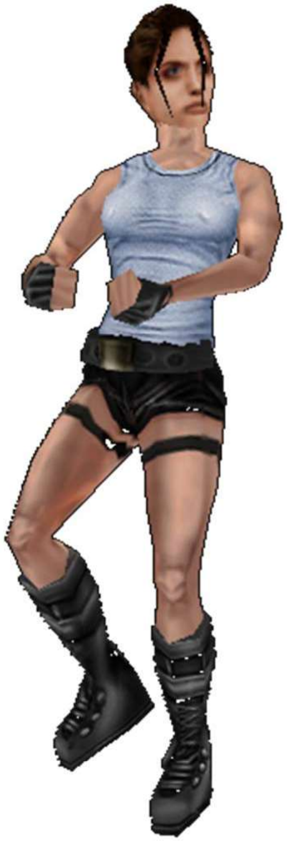


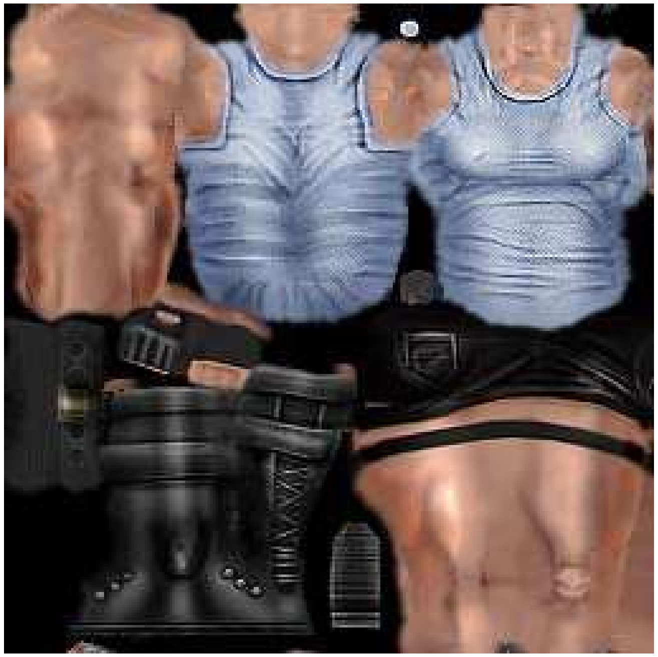
Skin mapping



Skin mapping



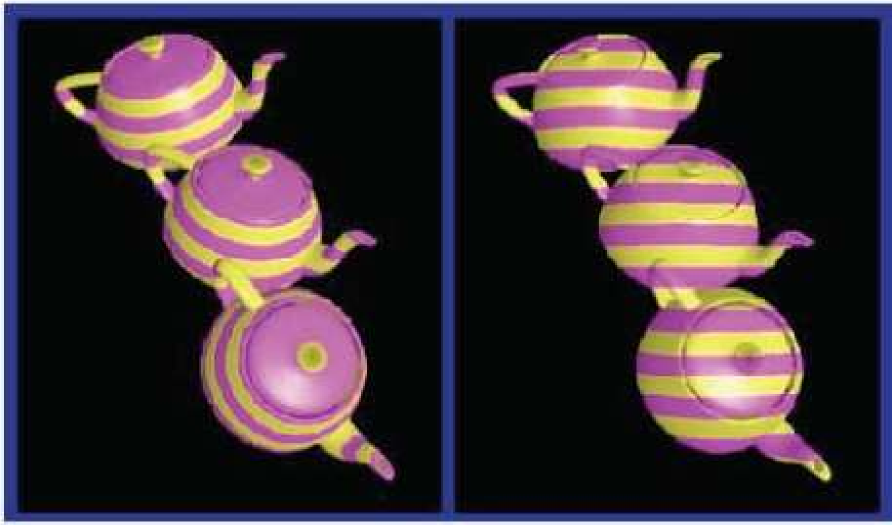




Orthographic mapping

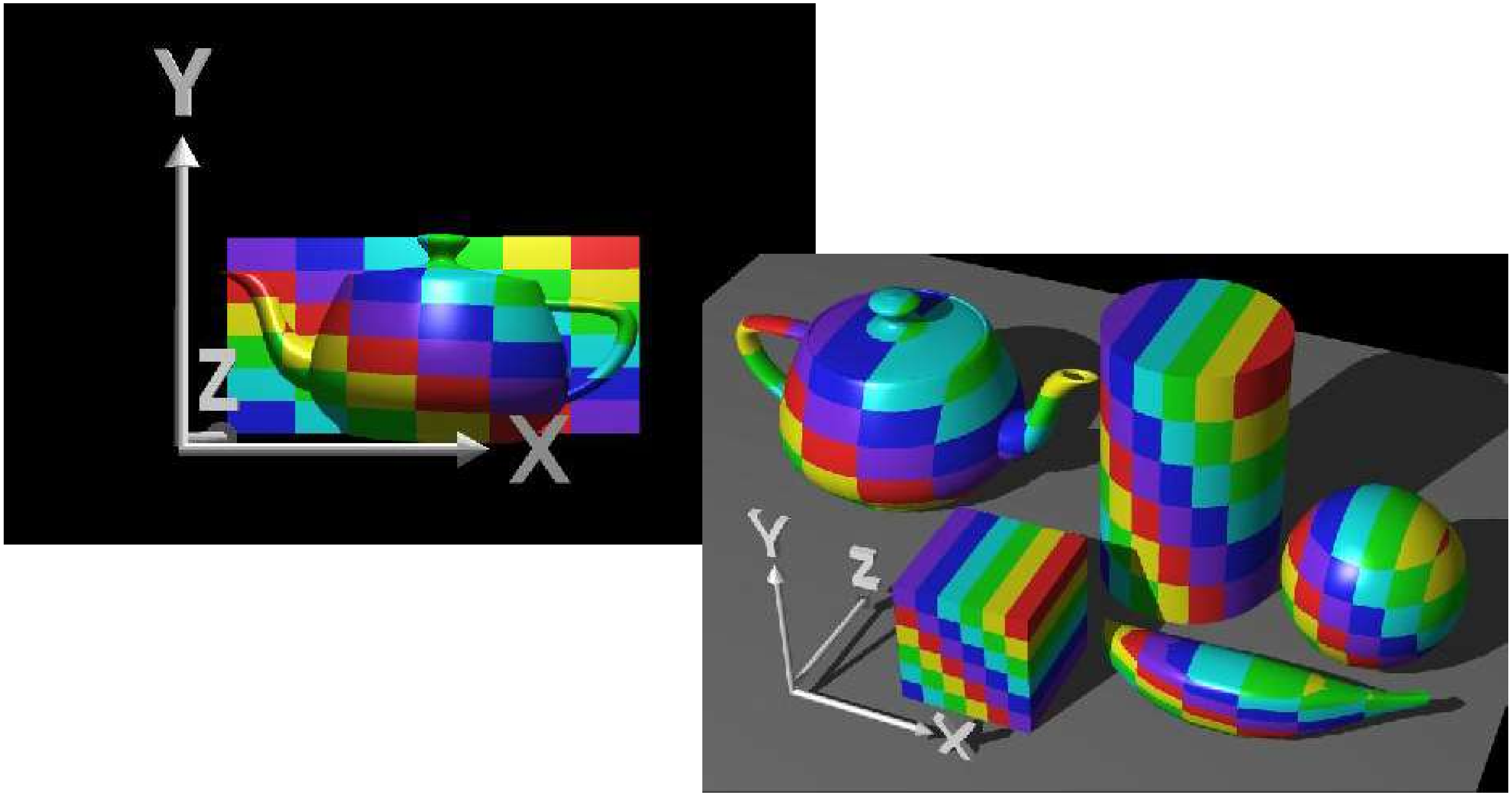
- Use linear transformation of object's XYZ coordinates

$$\begin{bmatrix} s \\ t \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

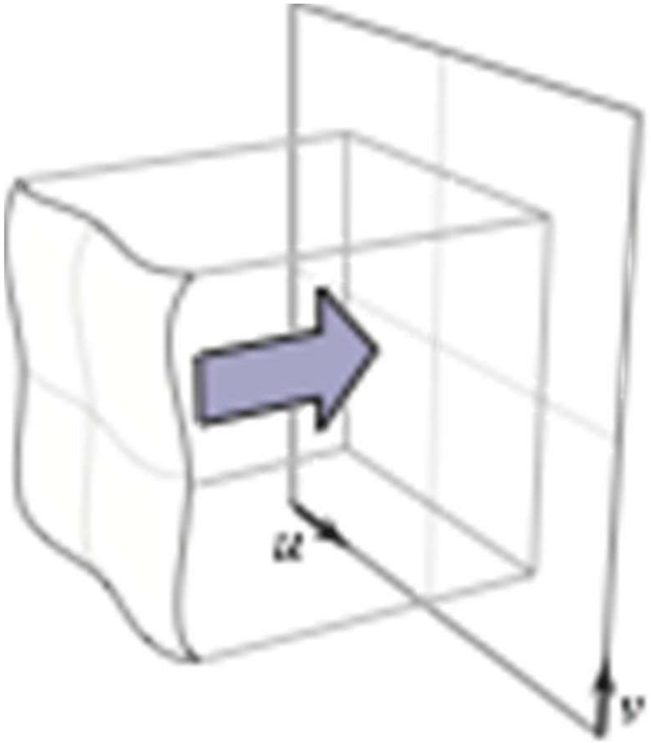


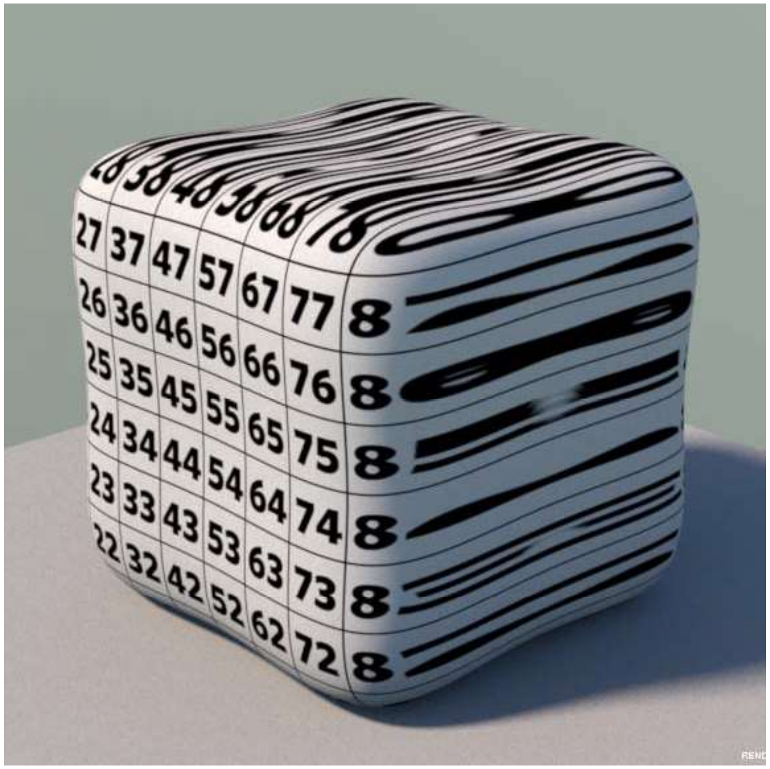
xyz in object space *xyz in camera space*

Planar mapping



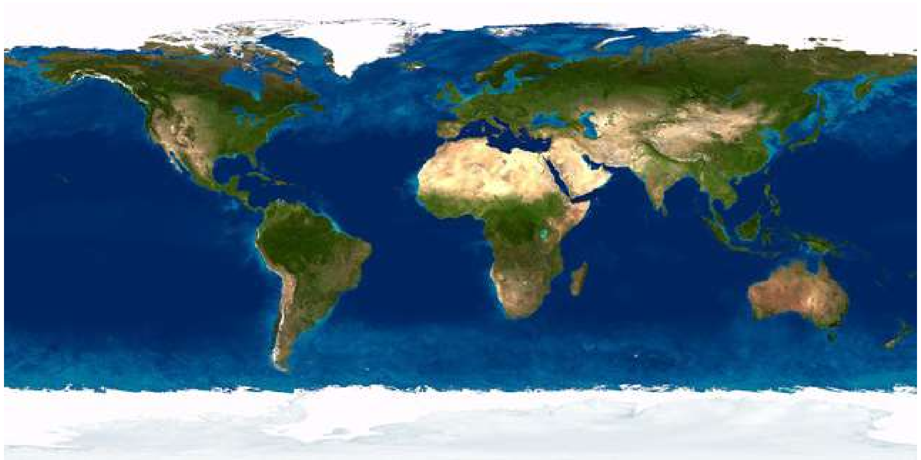
Planar mapping



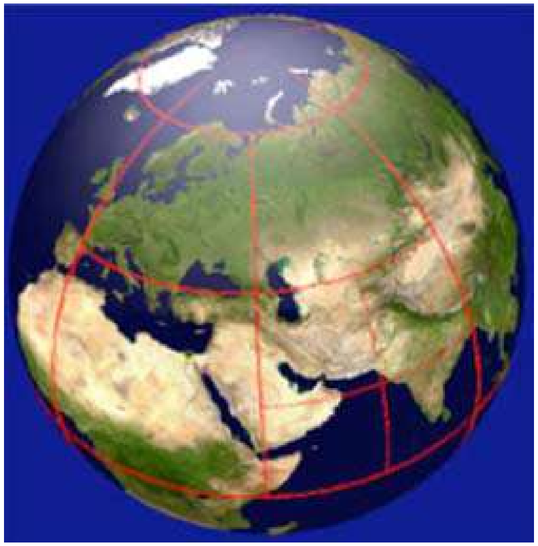


Spherical mapping

- Use spherical coordinates
- “Shrink-wrap” sphere to object

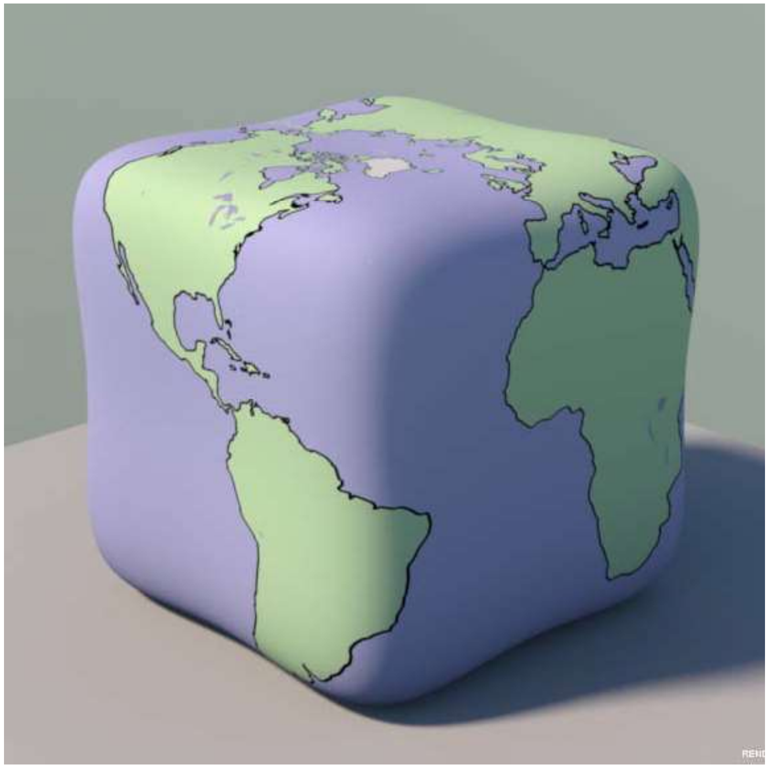


Texture map

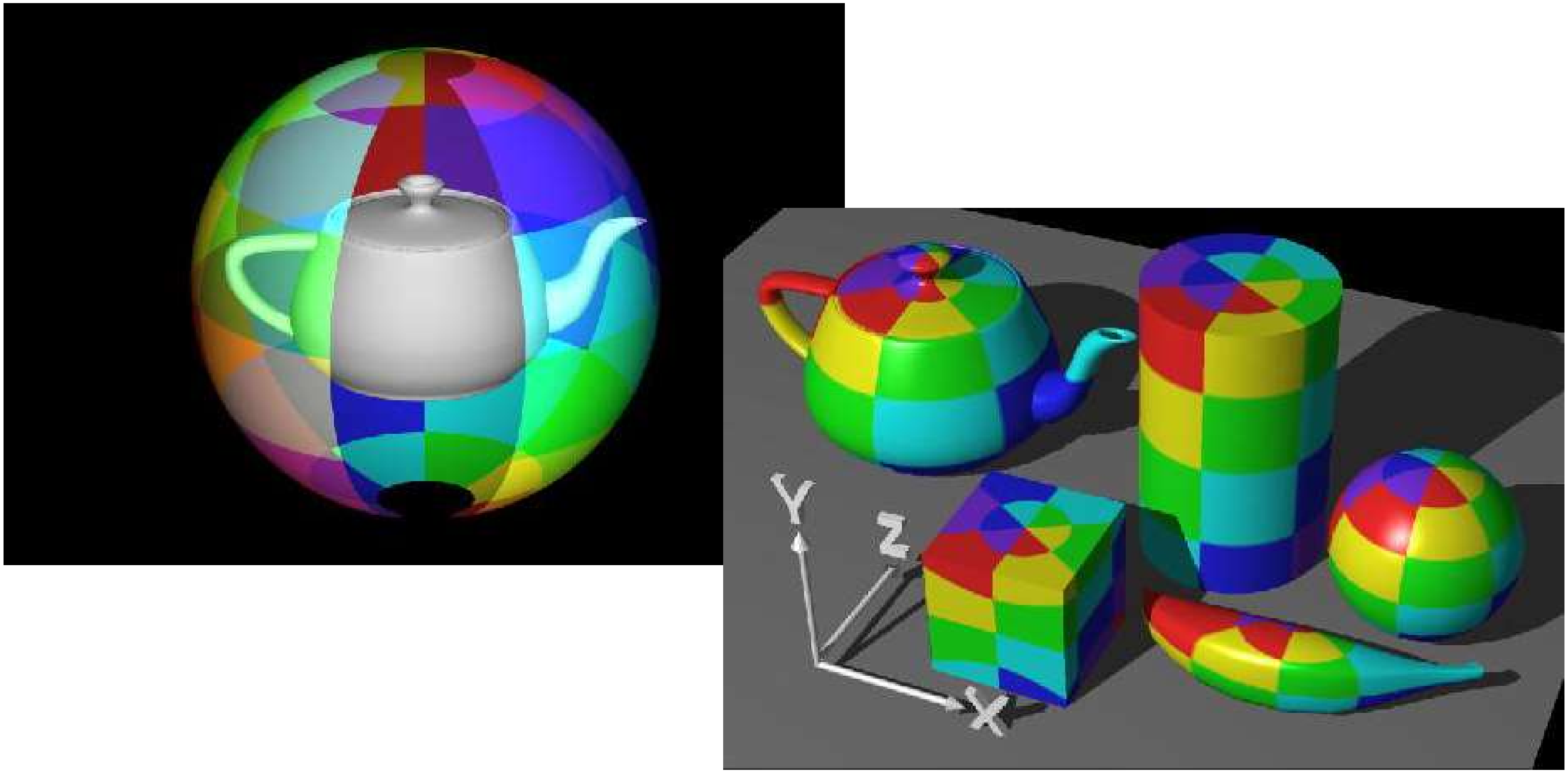


Mapping result

Spherical mapping

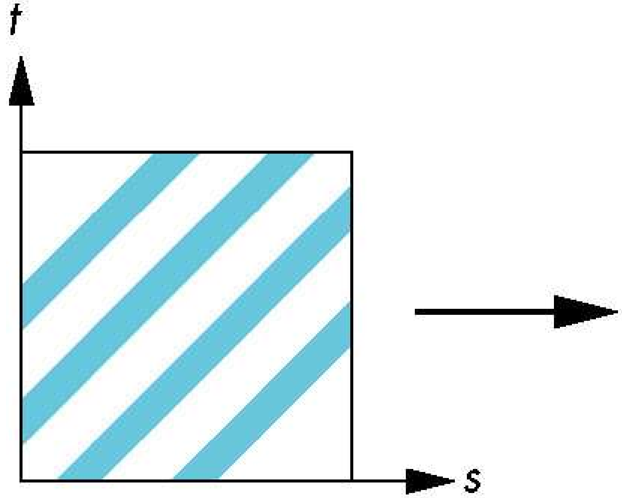


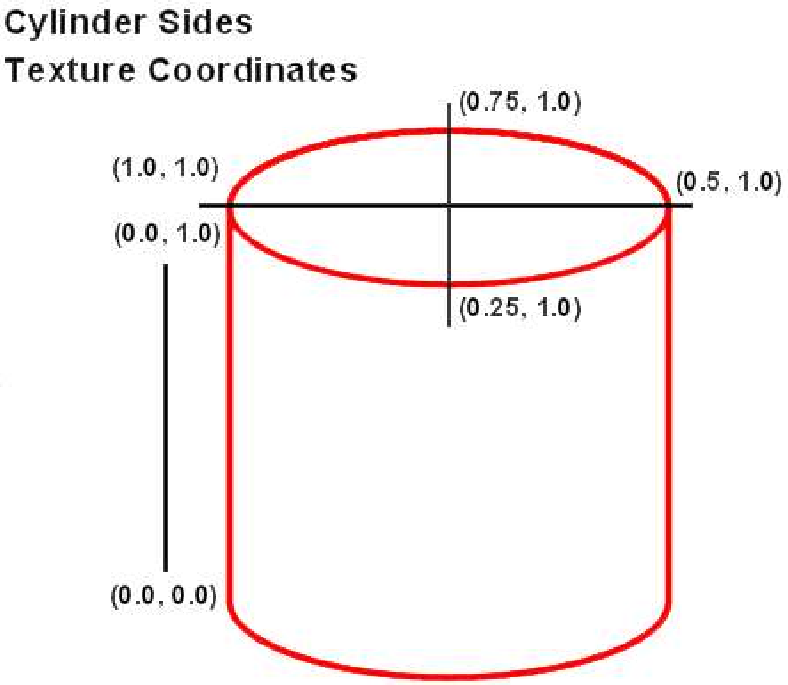
Spherical mapping



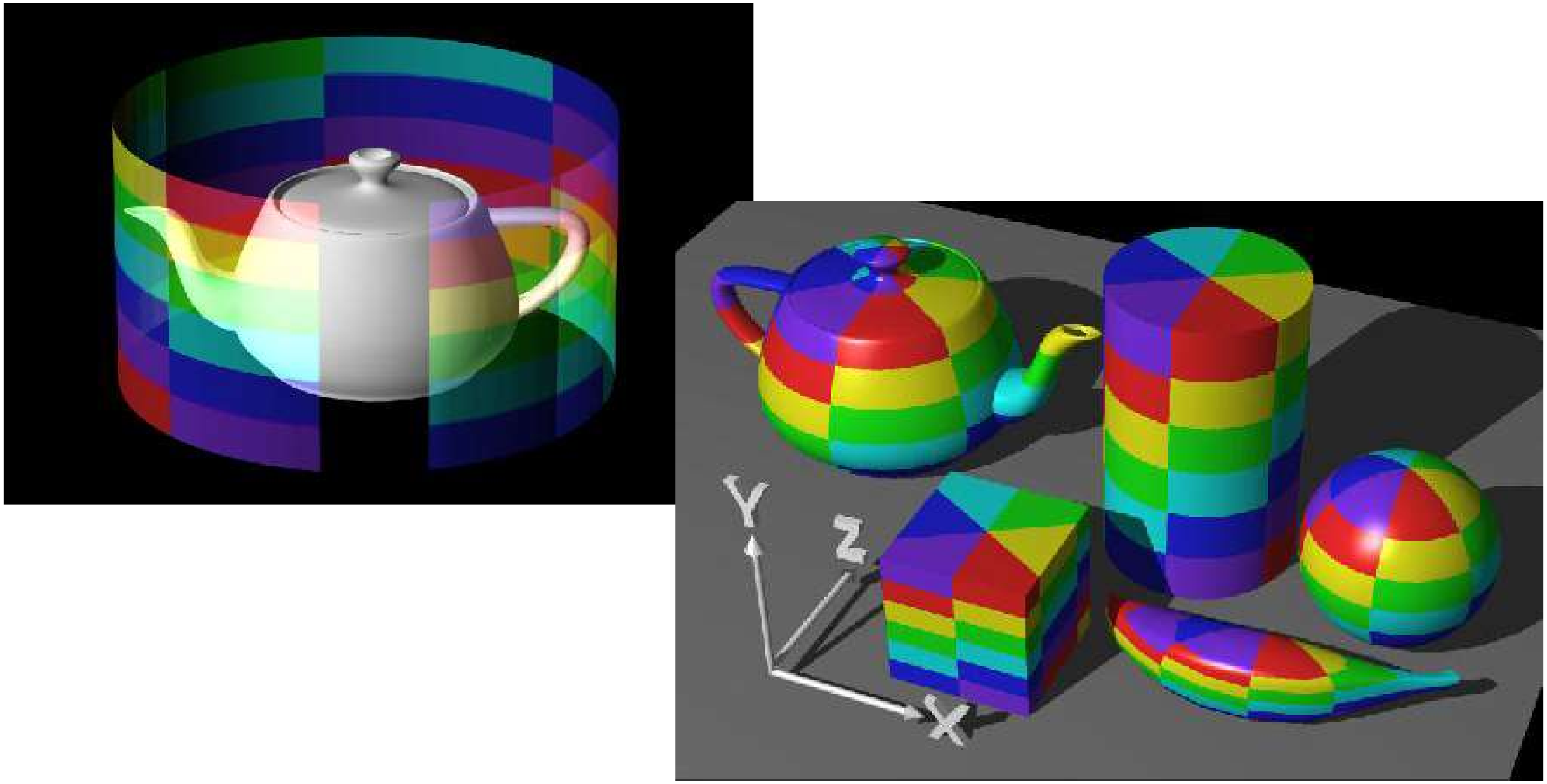
Cylindrical mapping

- Similar to spherical mapping, but with cylindrical coordinates

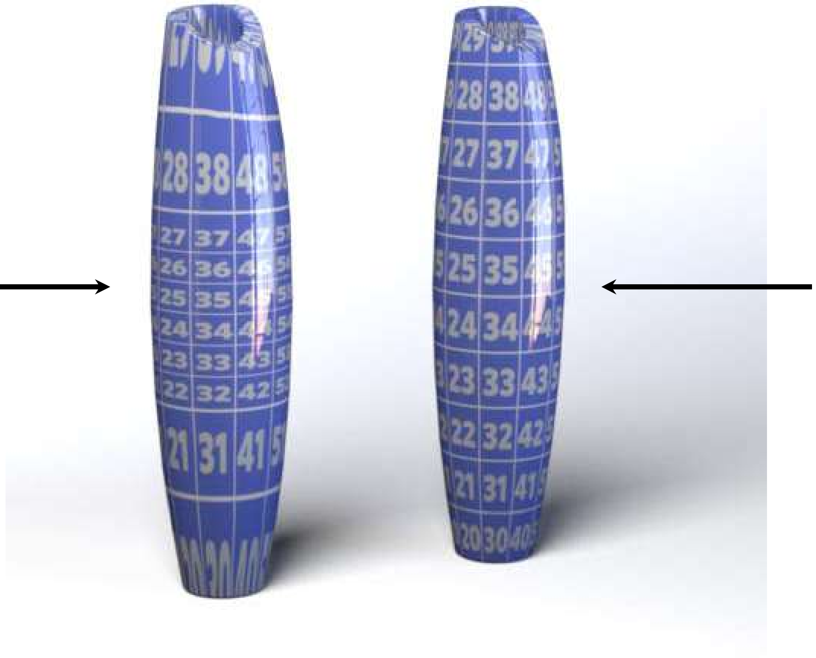




Cylindrical mapping



Spherical and cylindrical mapping

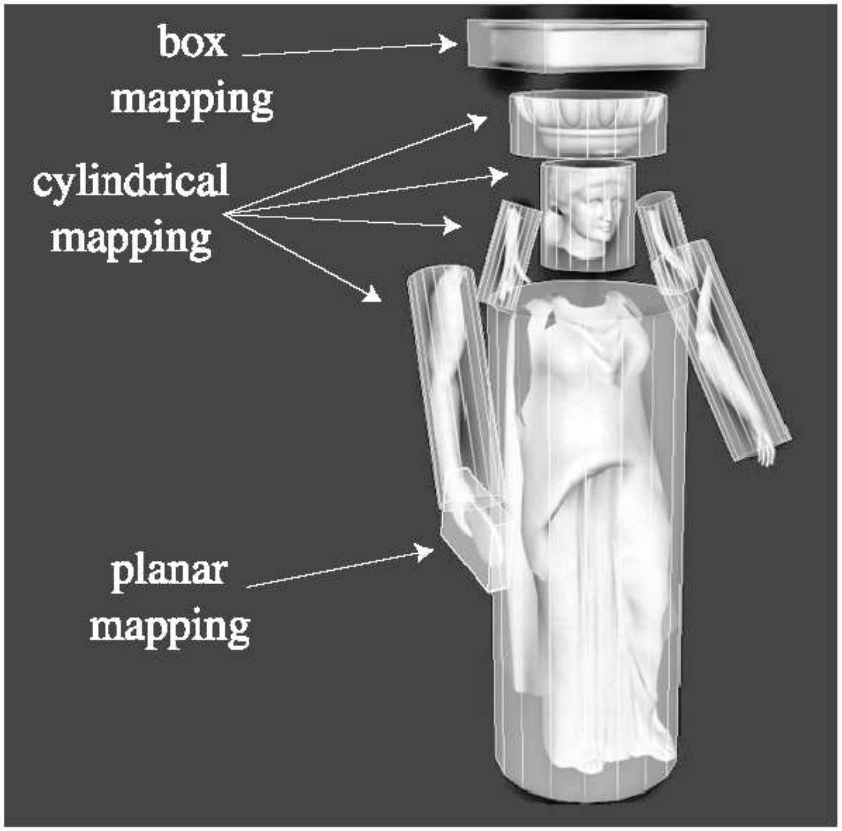


spherical

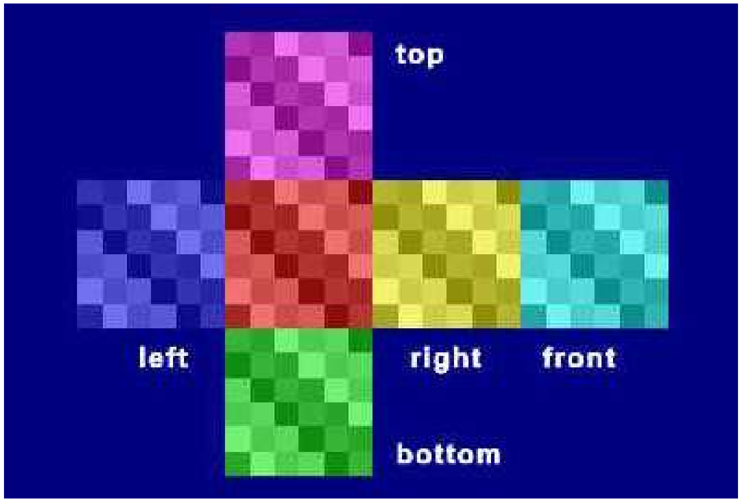
cylindrical

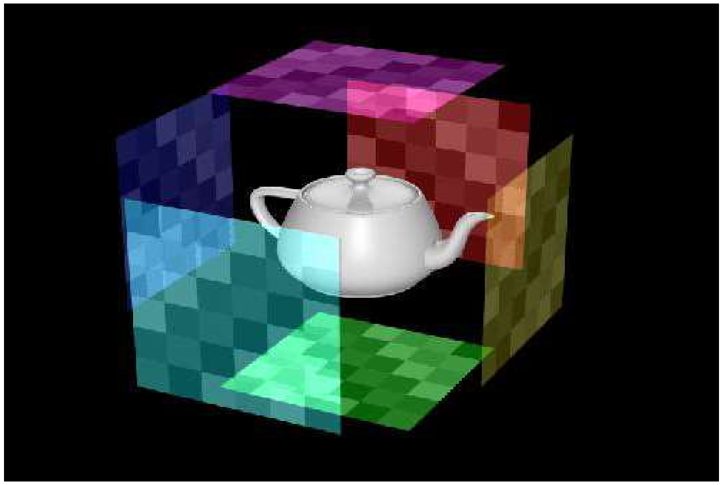
Multiple mappings

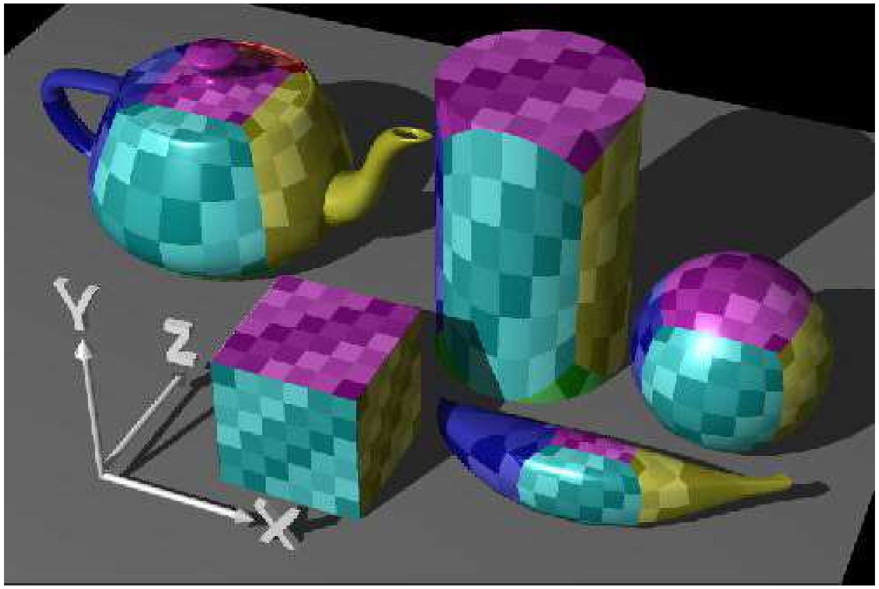
- Complex objects



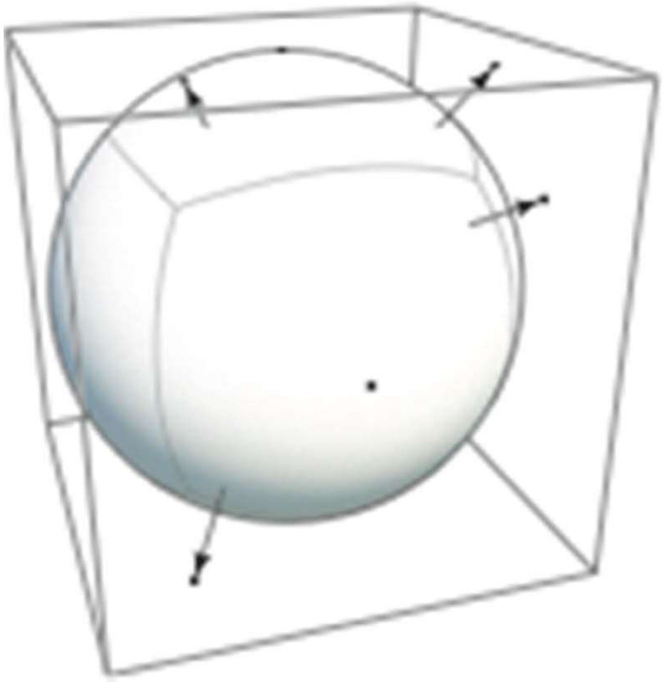
Cube mapping

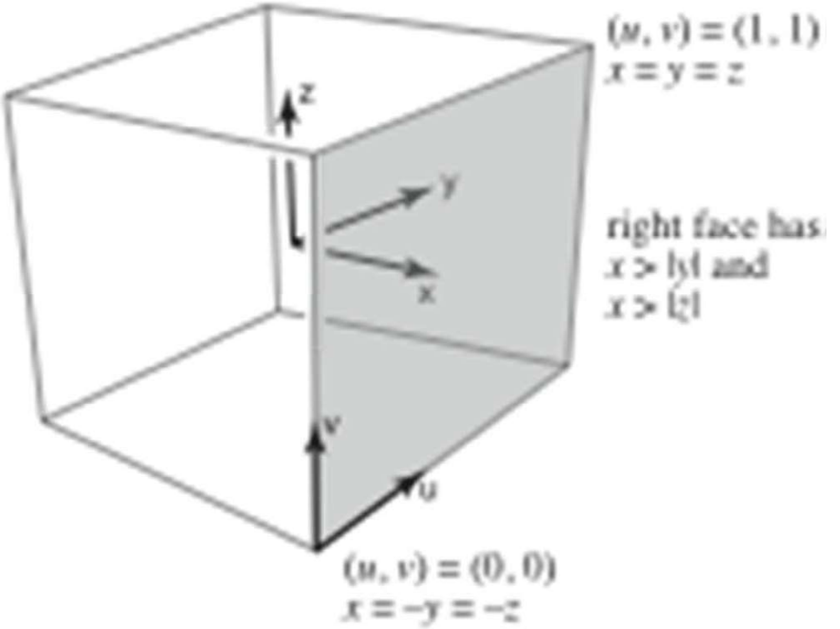




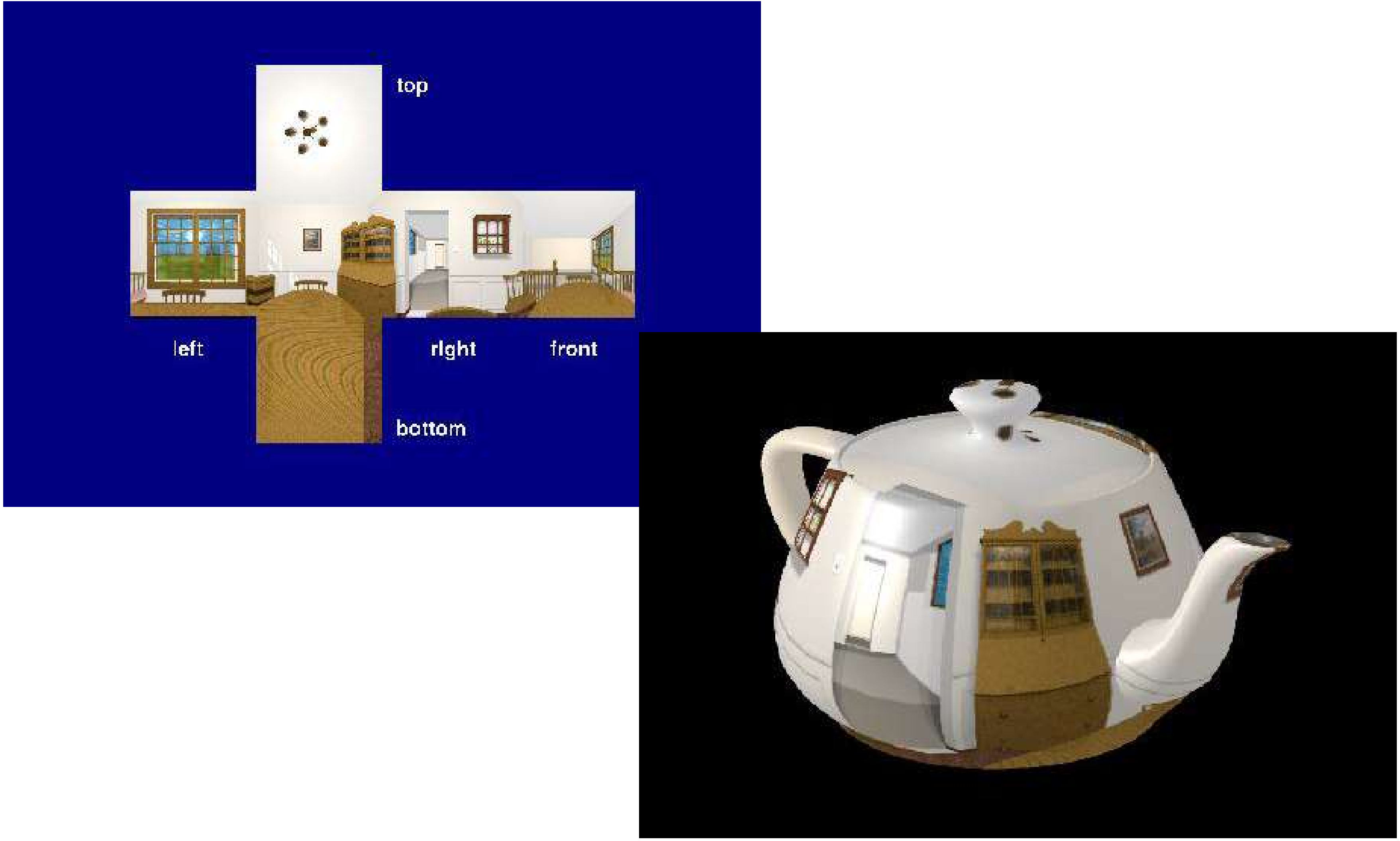


Cube mapping





Cube mapping


top
left
right
front
bottom