Textures

CS 4620 Lecture 19

Announcements

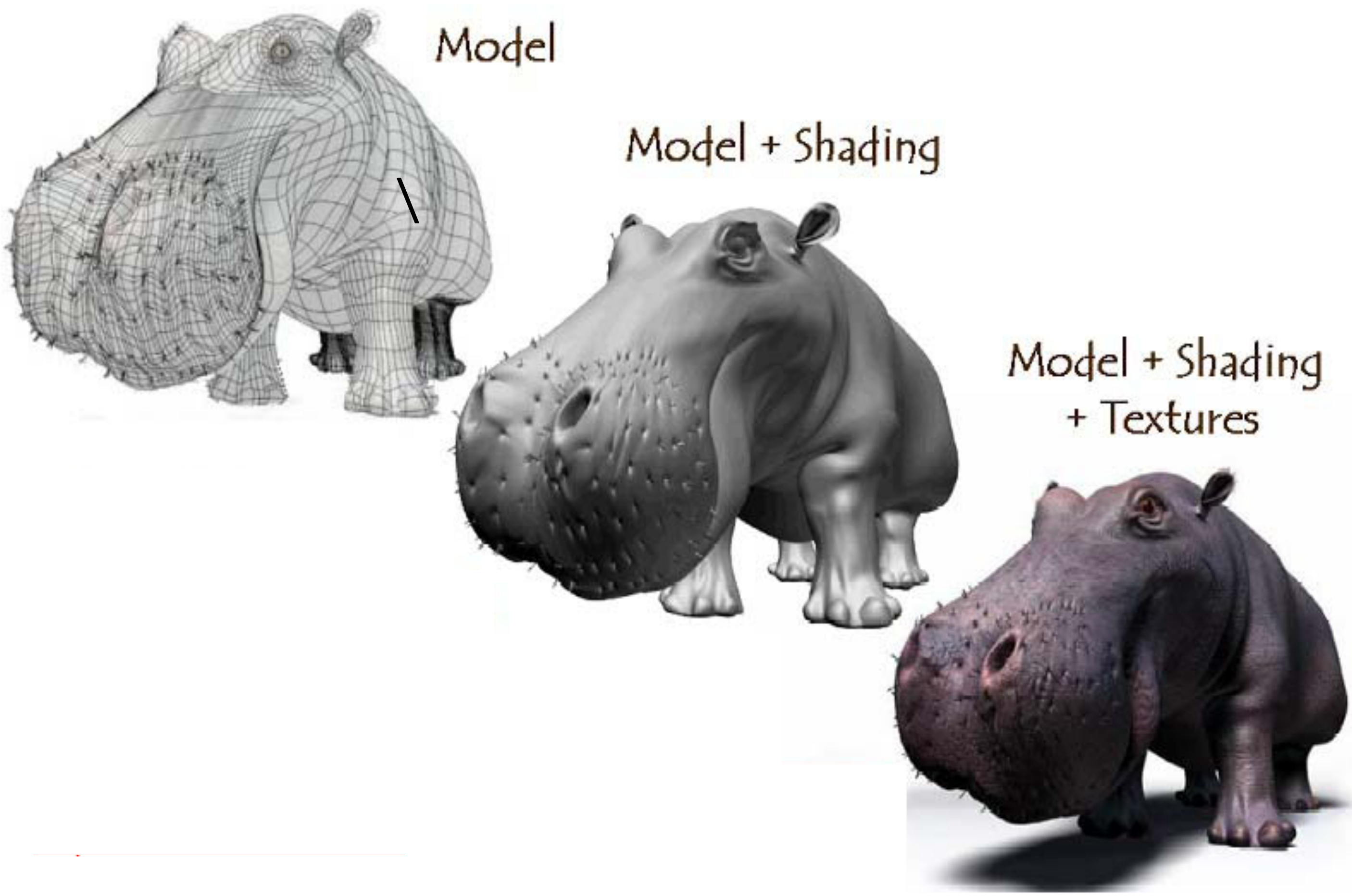
- A4 out this week. Schedule has been shifted accordingly
- A3 grading is today
  - Send cs4620-staff-l@cornell.edu mail if hard constraints
- Prelim next week
  - Oct 20th 2015, 7:30, Olin Hall 155

Texture mapping

- Objects have properties that vary across the surface

Texture Mapping

- Cannot model every single change using primitives
- Instead we make the shading parameters (and other properties) v



Texture Mapping: applications

- Surprisingly simple idea but with big results
  - Again uses memory (like the z buffer)

A definition

Texture mapping: a technique of defining surface properties in such a way that they vary as a function of position on the surface

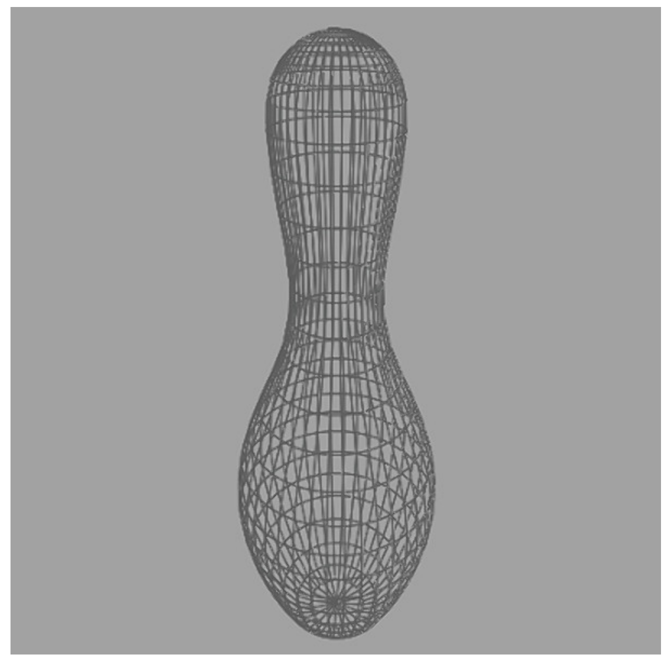
Texture mapping

- Actually more than diffuse material properties
- Textures increase apparent visual complexity of geometry and material
  - Diffuse material properties
  - Specular properties
  - Normals
  - Positions
  - Lighting....

- Increases realism, but very very simple

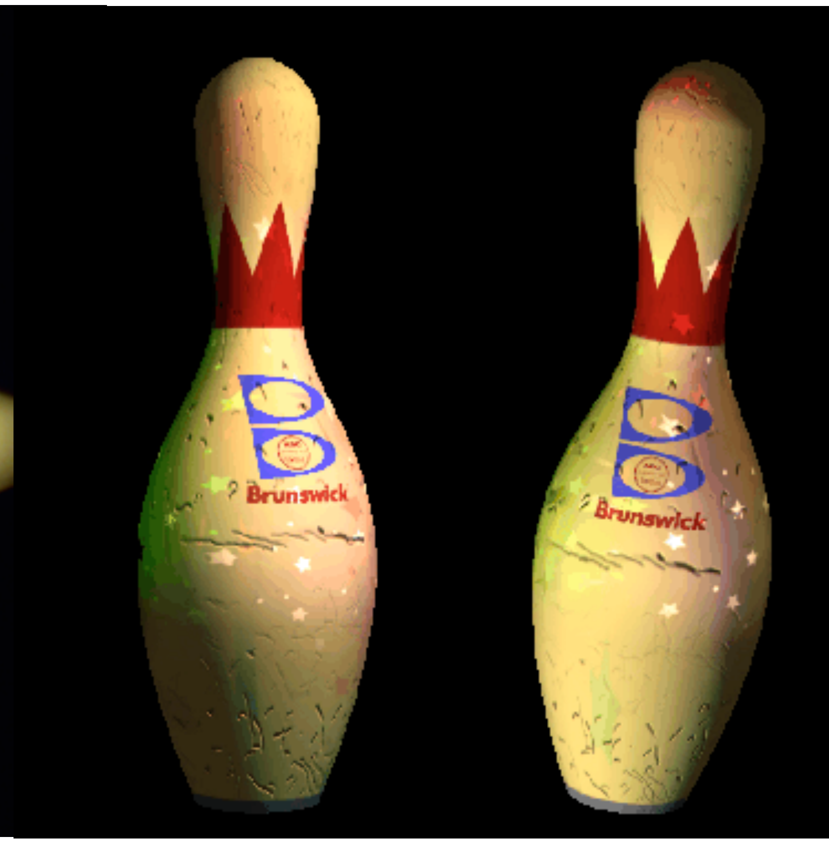
Texture mapping

- Surface properties are not the same everywhere
  - diffuse color (k_d) varies due to changing pigmentation
  - brightness (k_s) and sharpness (p) of specular highlight varies due to changing roughness and surface contamination



Texture mapping

- Surface properties are not the same everywhere
  - diffuse color (k_d) varies due to changing pigmentation
  - brightness (k_s) and sharpness (p) of specular highlight varies due to changing roughness and surface contamination



Examples

- Wood gym floor with smooth finish
  - diffuse color k_D varies with position
  - specular properties k_S, n are constant
- Glazed pot with finger prints
  - diffuse and specular colors k_D, k_S are constant
  - specular exponent n varies with position
- Adding dirt to painted surfaces
- Simulating stone, fabric, …
  - to approximate effects of small-scale geometry
    - they look flat but are a lot better than nothing

Texture mapping

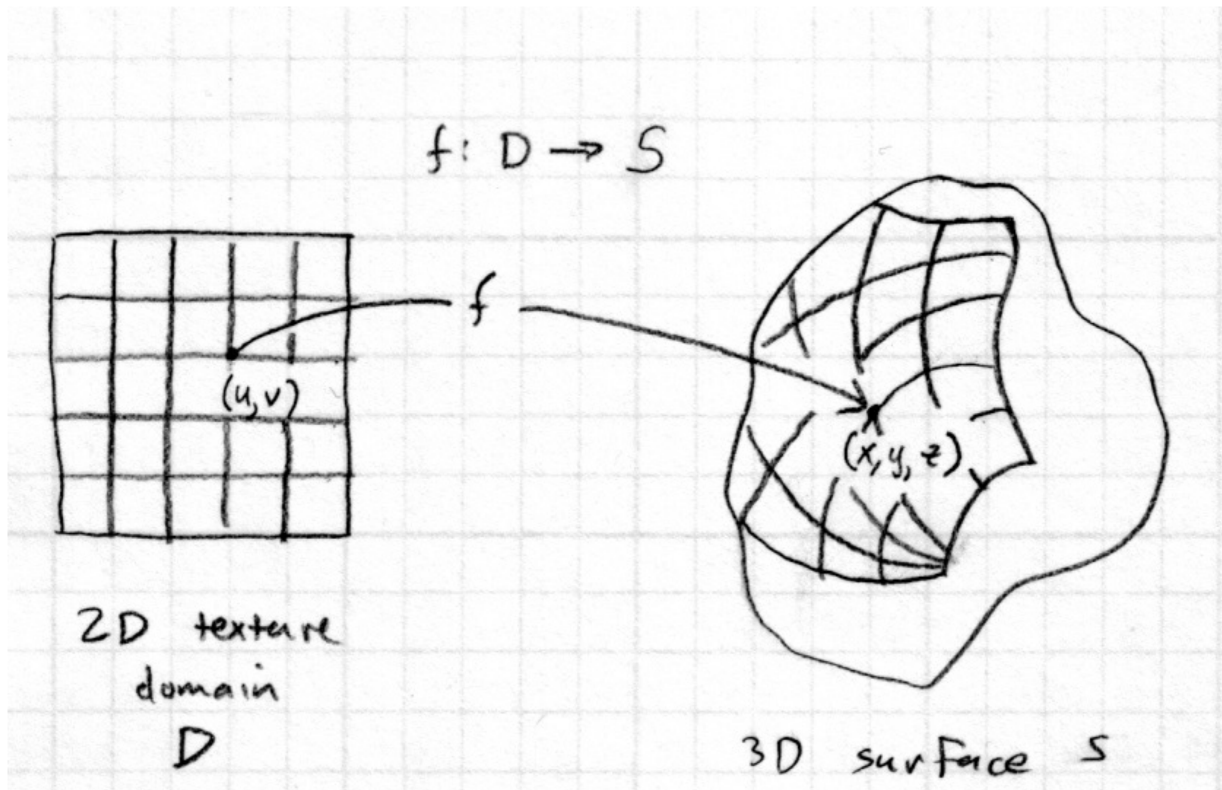
- Want functions that assign properties to points on the surface
  - the surface is a 2D domain
  - given a surface parameterization, just need function on plane
  - images are a handy way to represent such functions
  - can represent using any image representation
  - raster texture images are very popular

Mapping textures to surfaces

- Usually the texture is an image (function of *u*, v)
  - the big question of texture mapping: where on the surface does the image go?
  - obvious only for a flat rectangle the same shape as the image
  - otherwise more interesting

Mapping textures to surfaces

- "Putting the image on the surface"
  - this means we need a function *f* that tells where each point on the image goes
  - this looks a lot like a parametric surface function
  - for parametric surfaces (e.g. sphere, cylinder) you get *f* for free

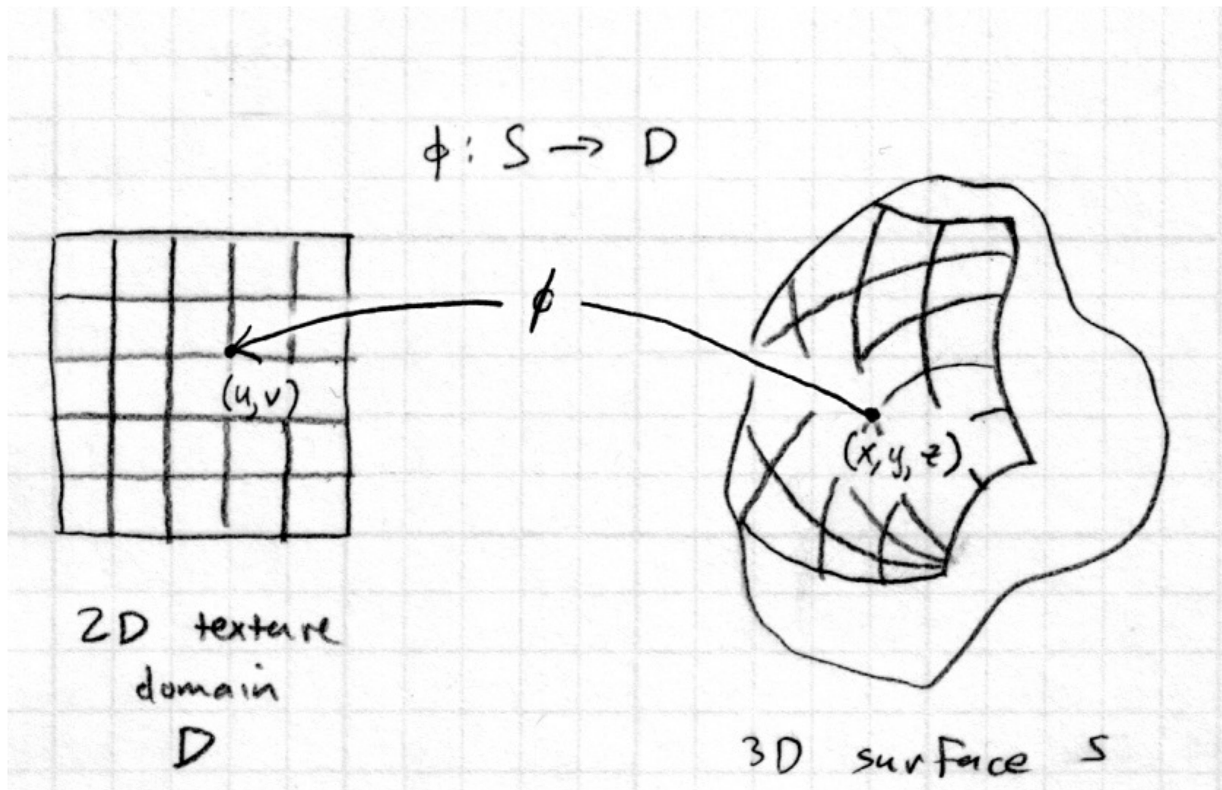


Texture coordinate functions

- Non-parametrically defined surfaces: more to do
  - can't assign texture coordinates as we generate the surface
  - need to have the *inverse* of the function *f*
- Texture coordinate fn.

  $$\phi : S \to \mathbb{R}^2$$

  - when shading **p** get texture at $\phi(\mathbf{p})$



Texture coordinate functions

- Define texture image as a function

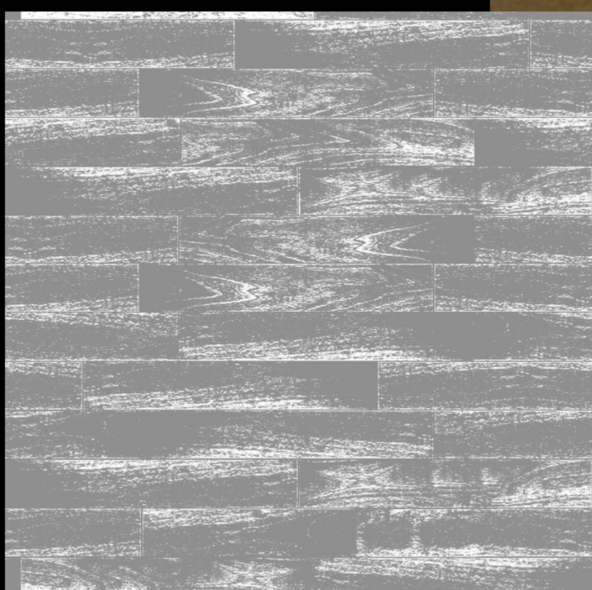
$$T : D \to C$$

  - where *C* is the set of colors for the diffuse component

- Diffuse color (for example) at point **p** is then

$$k_D(\mathbf{p}) = T(\phi(\mathbf{p}))$$



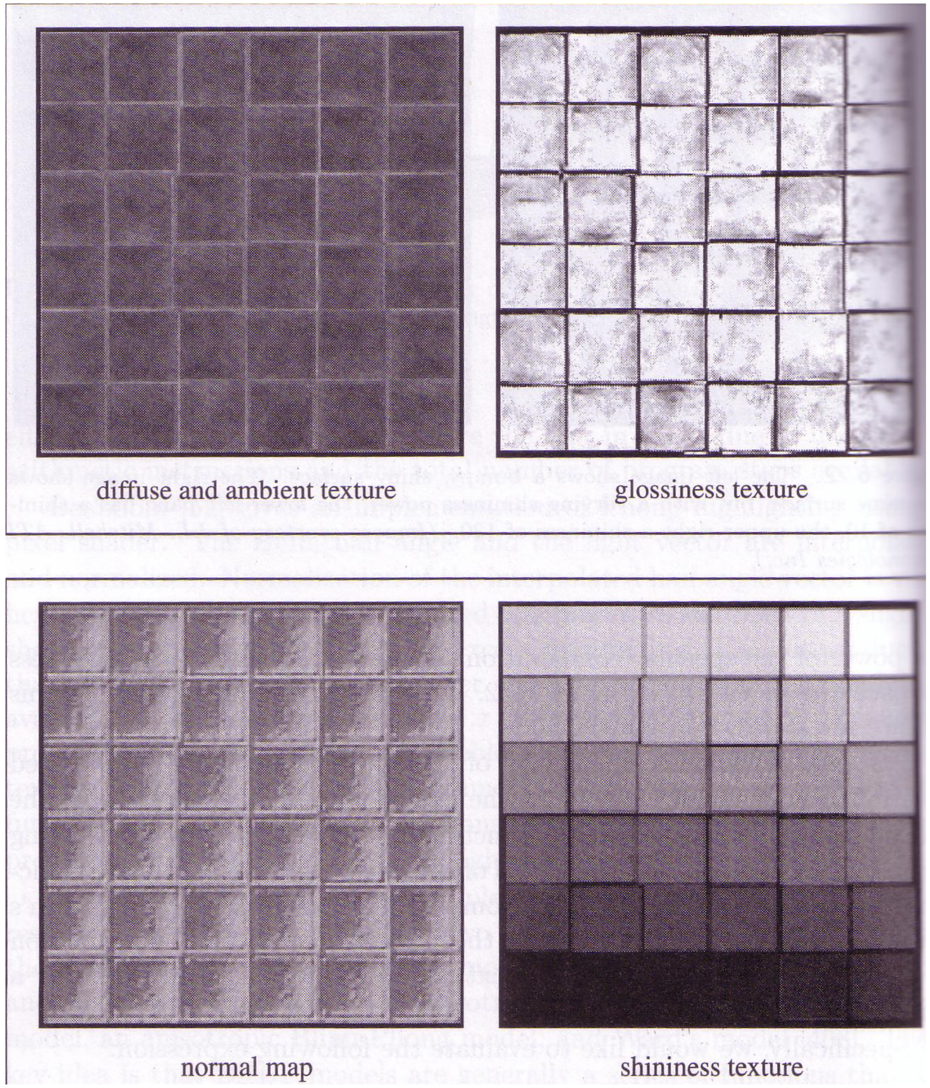




Power of Texture Mapping

n.l d + (n.h)^m g

Effects: Example

- Diffuse color texture
- Gloss map: strength of specular
- Shininess map for power of specular
- Normal map for bumpiness


diffuse and ambient texture
glossiness texture
normal map
shininess texture

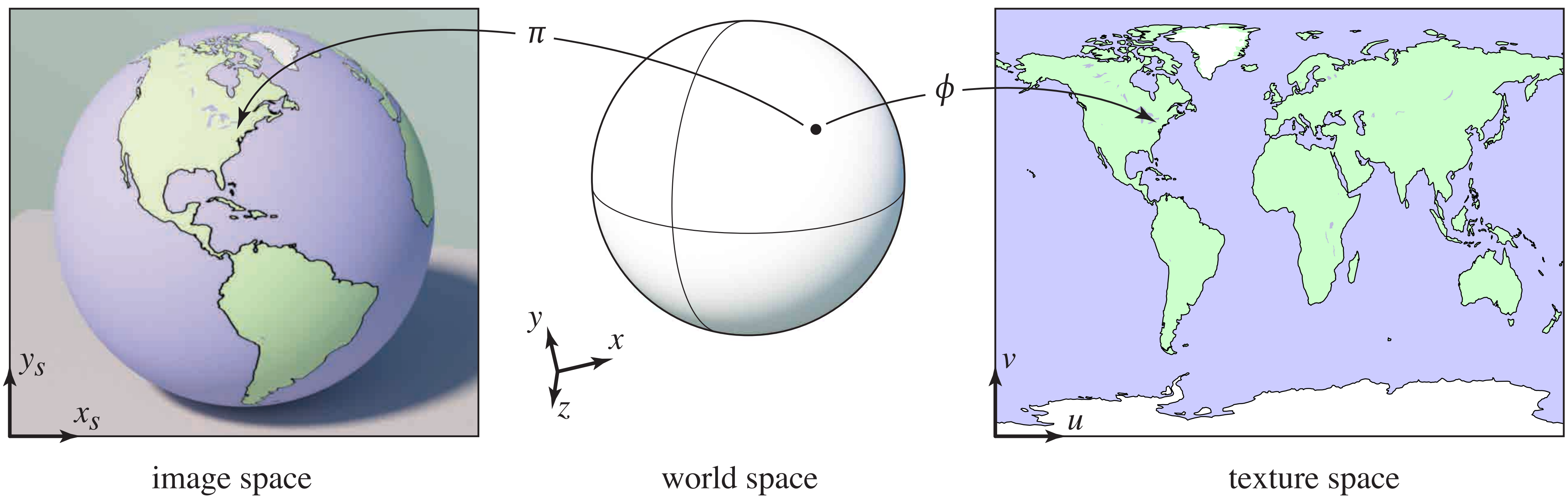
Variable Specular Power

Key concept: everything is a texture lookup (and an arbitrary fn)!

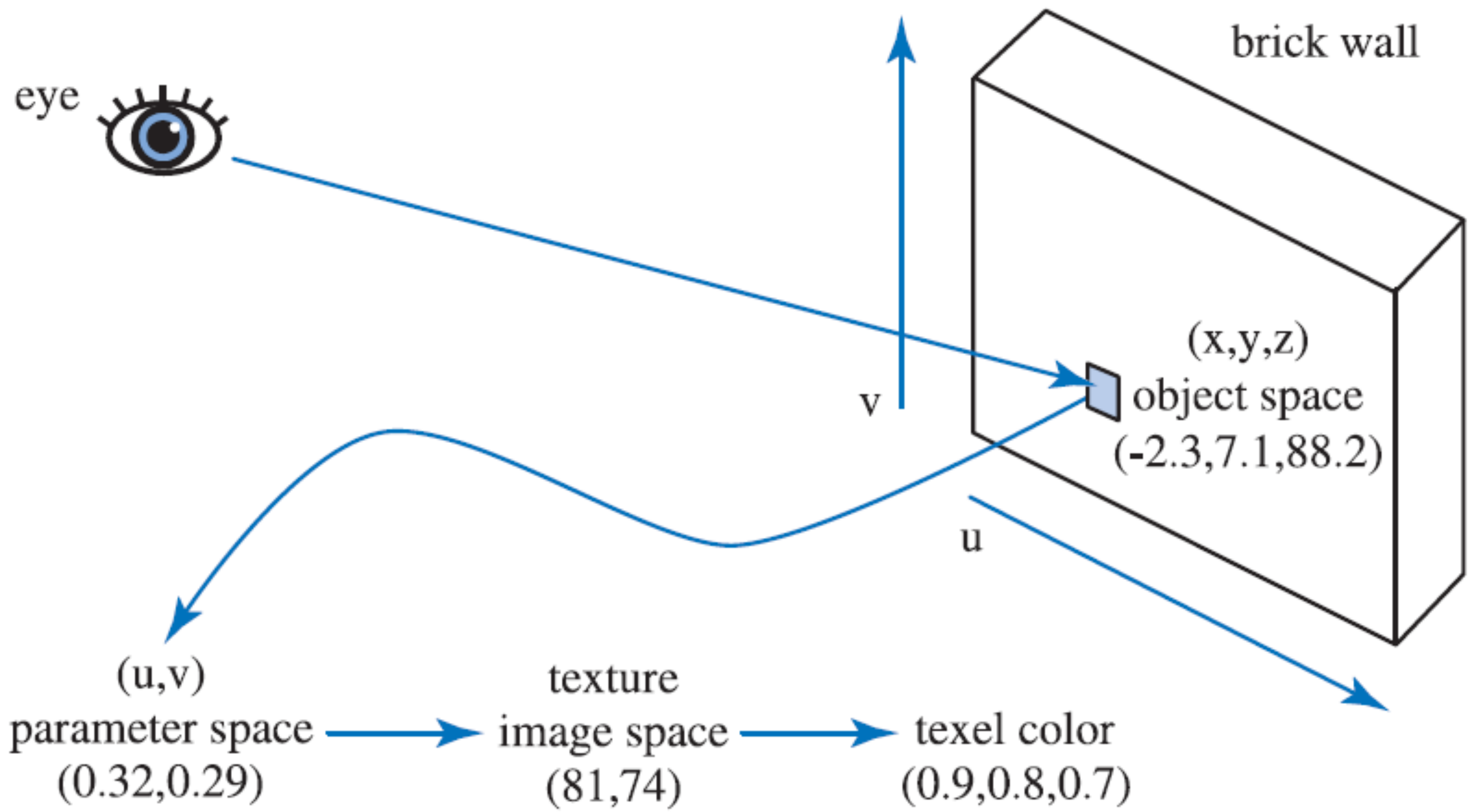
(a) High-res geometry (b) Real-time hybrid map rendering (c) Offline SSS rendering

Three spaces

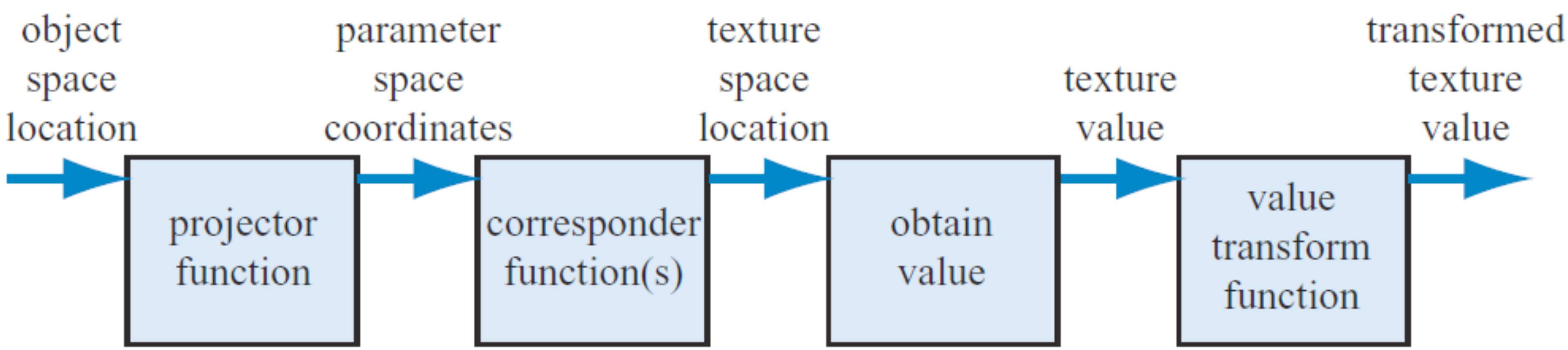
- Surface lives in 3D world space
- Every point has a place it goes in the image
  - … and a place it maps to in the texture.



How does it work?

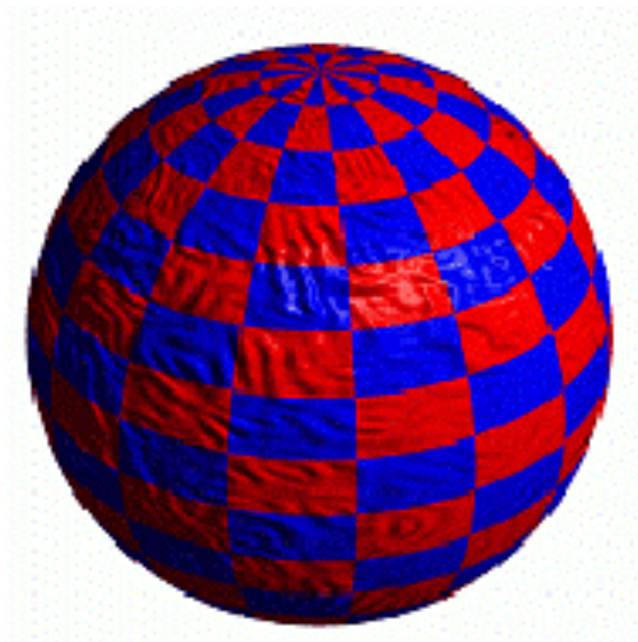


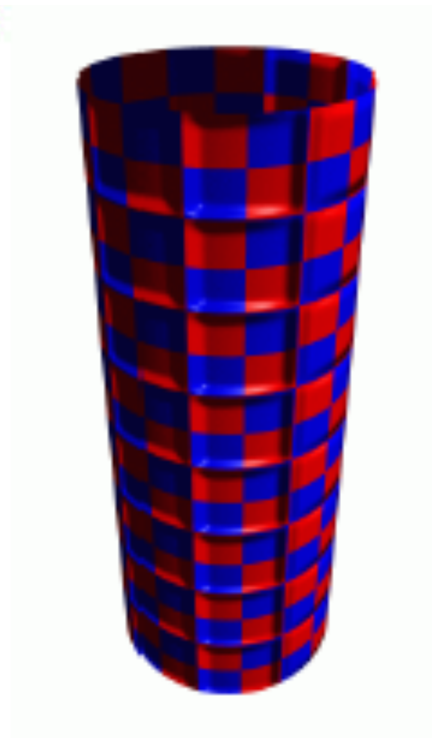
Texture Pipeline



Projector Functions

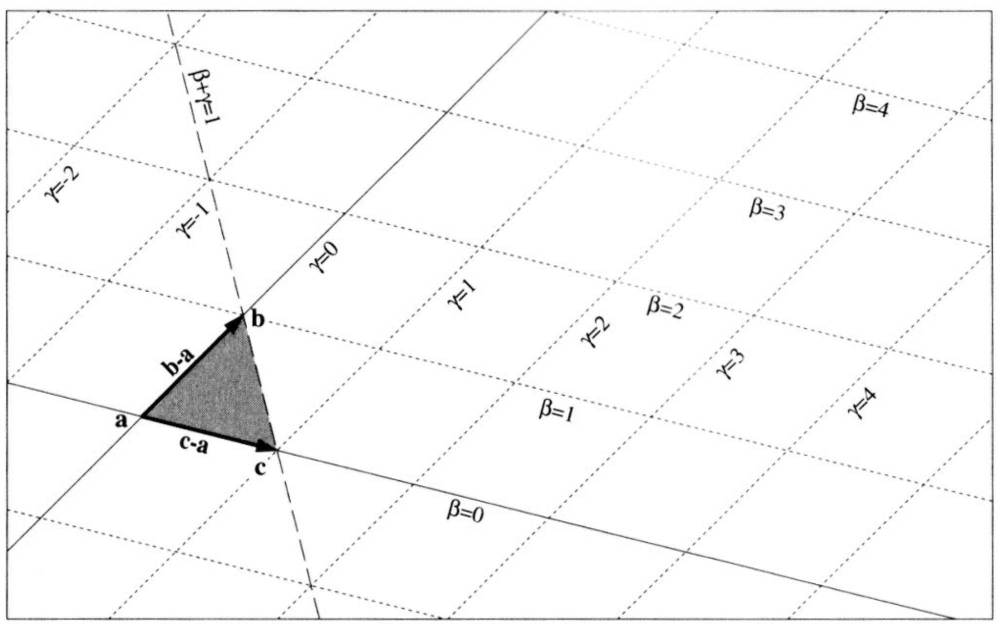
- Planes, cylinders, spheres





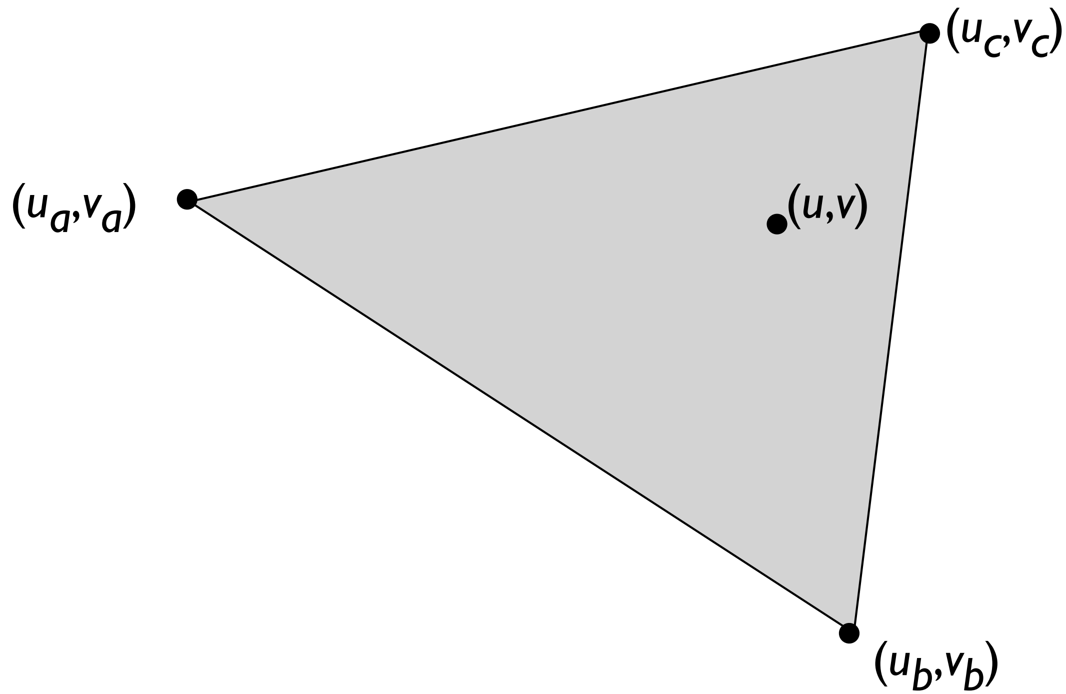
Examples of projector functions

- A square/rectangle
  - image can be mapped directly, unchanged
- An arbitrary plane
  - simple affine transformation (rotate, scale, translate)
- A triangle



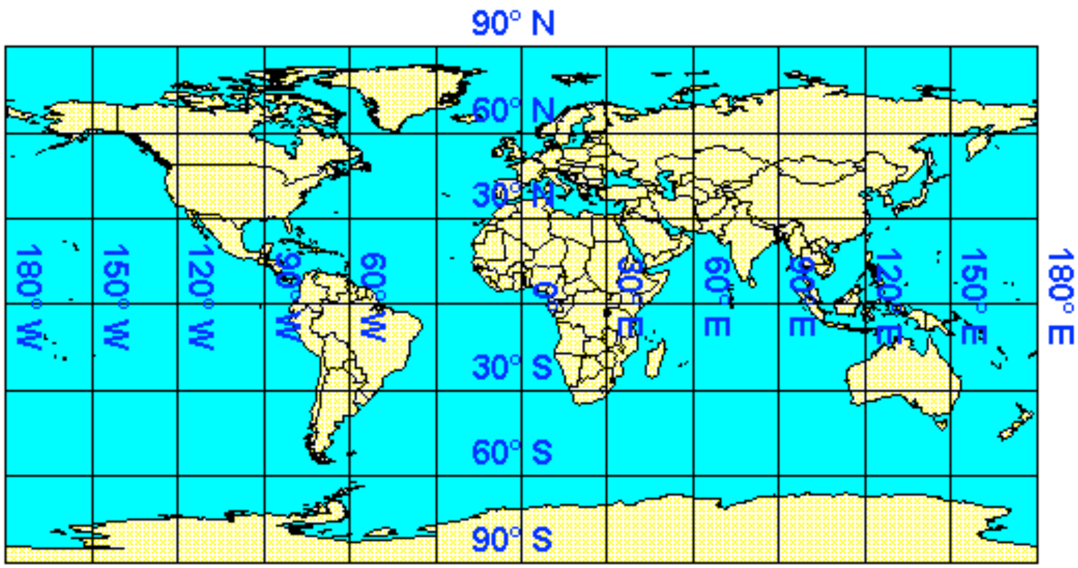
Examples of projector functions

- Triangles
  - specify (*u*,*v*) for each vertex
  - define (*u*,*v*) for interior by linear interpolation



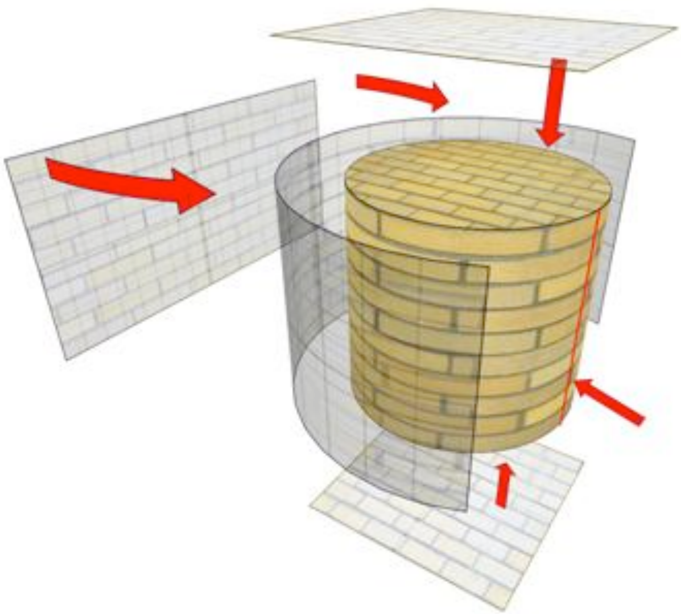
Examples of projector functions

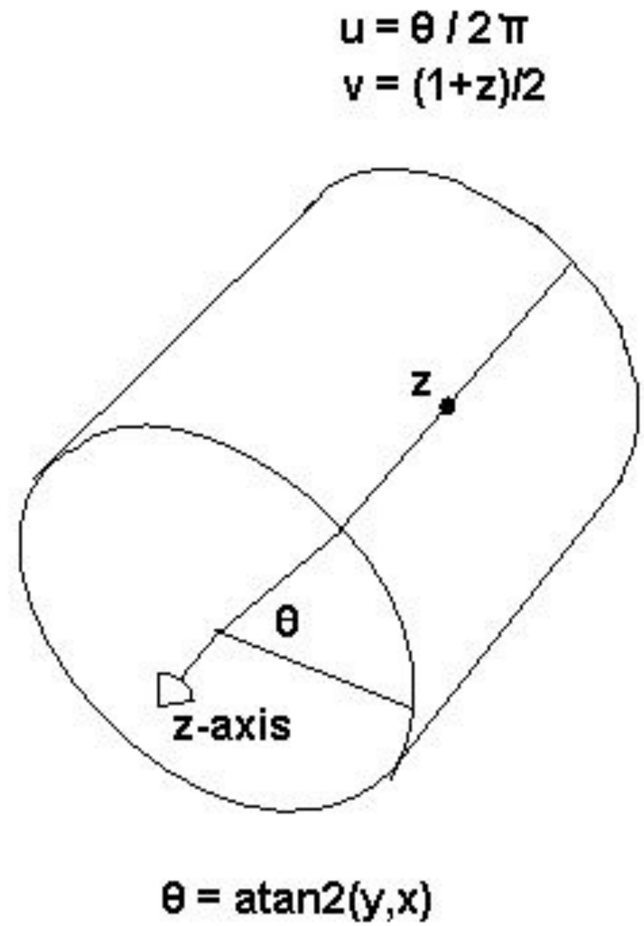
- For a sphere: latitude-longitude coordinates
  - ϕ maps point to its latitude and longitude



[map: Peter H. Dana]

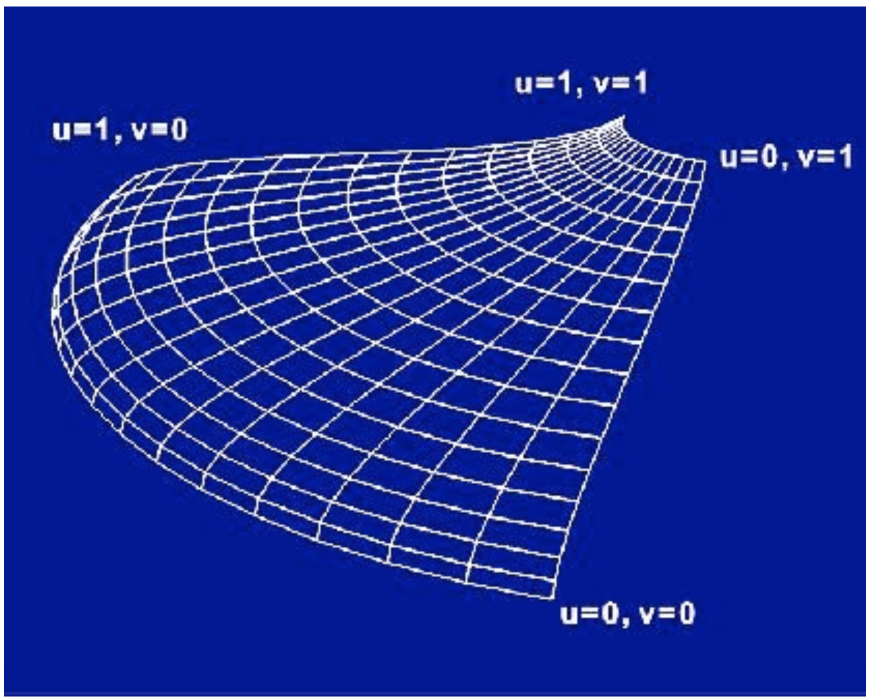
Cylinder

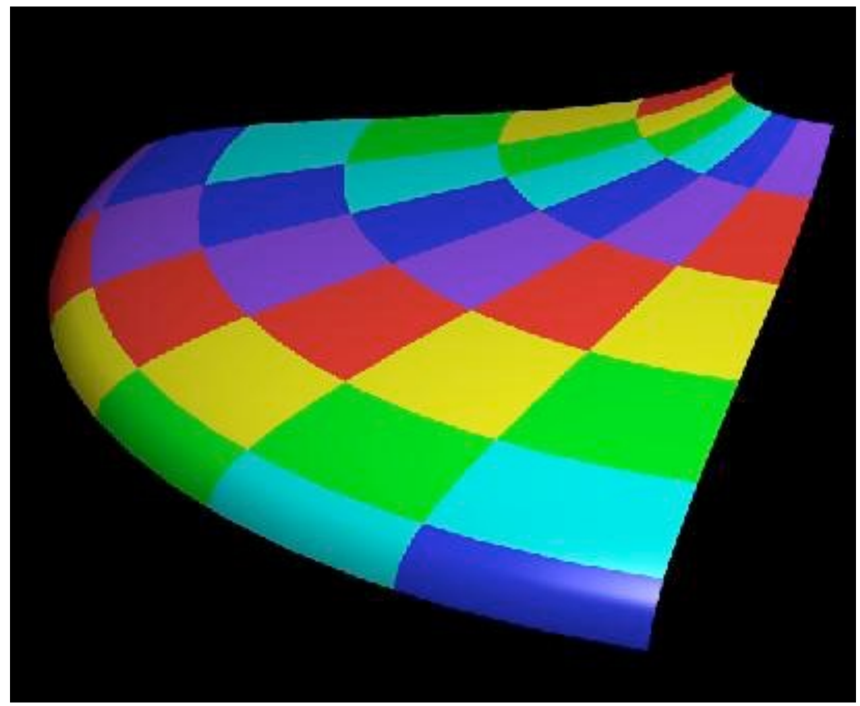




Examples of projector functions

- A parametric surface (e.g. spline patch)
  - surface parameterization gives mapping function directly (well, the inverse of the projector function)

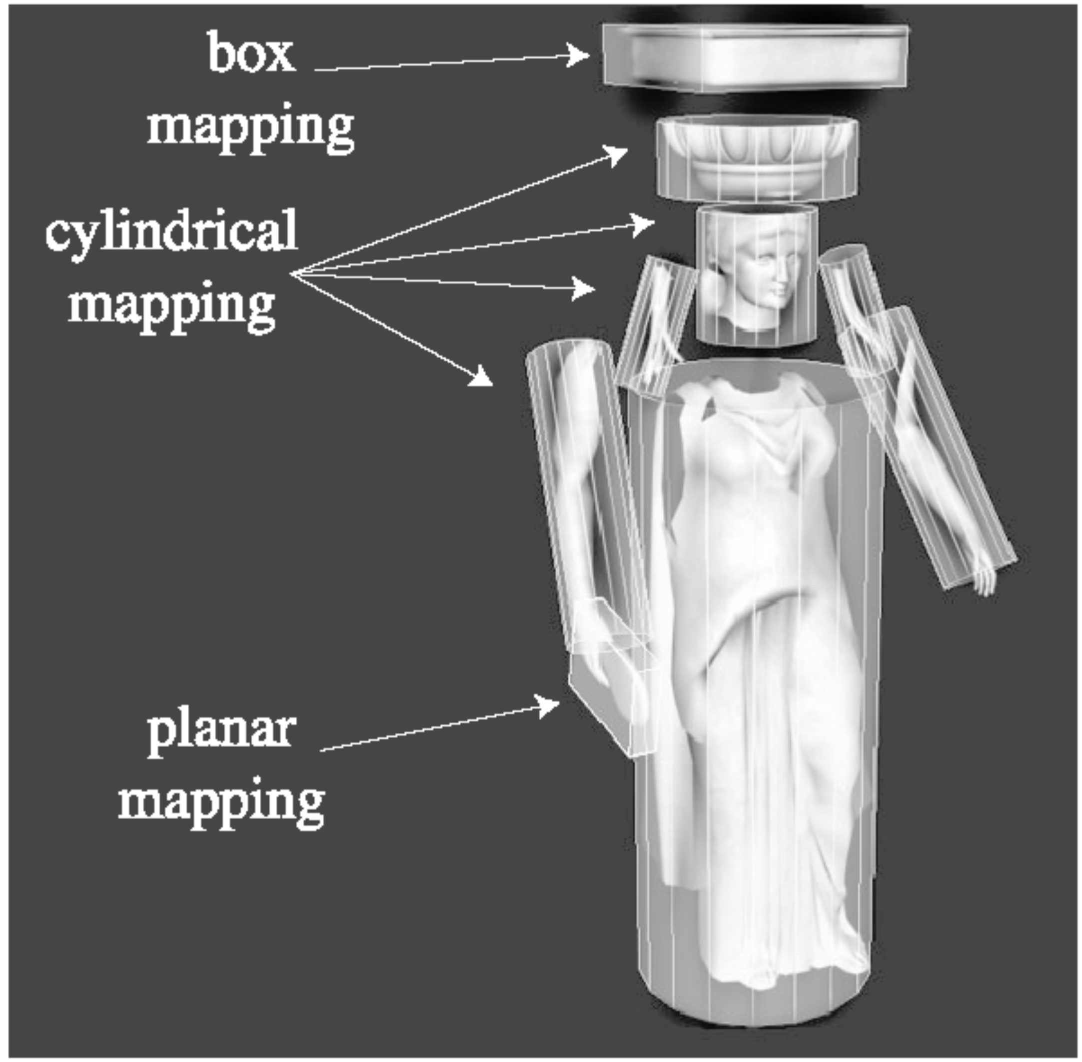




[Wolfe / SG97 Slide set]

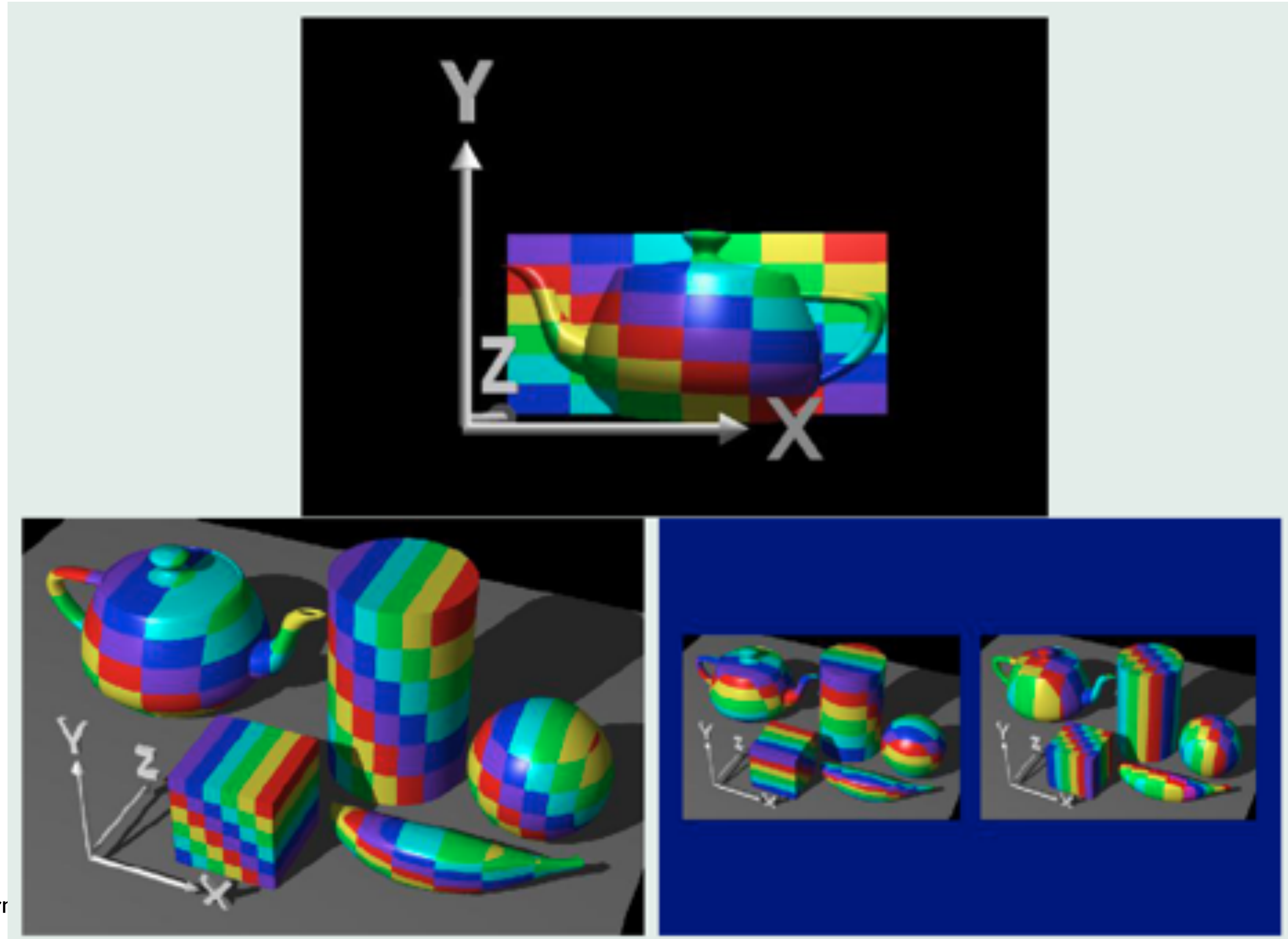
Projector Function: Arbitrary Surfaces

- Non-parametric surfaces: project to parametric surface

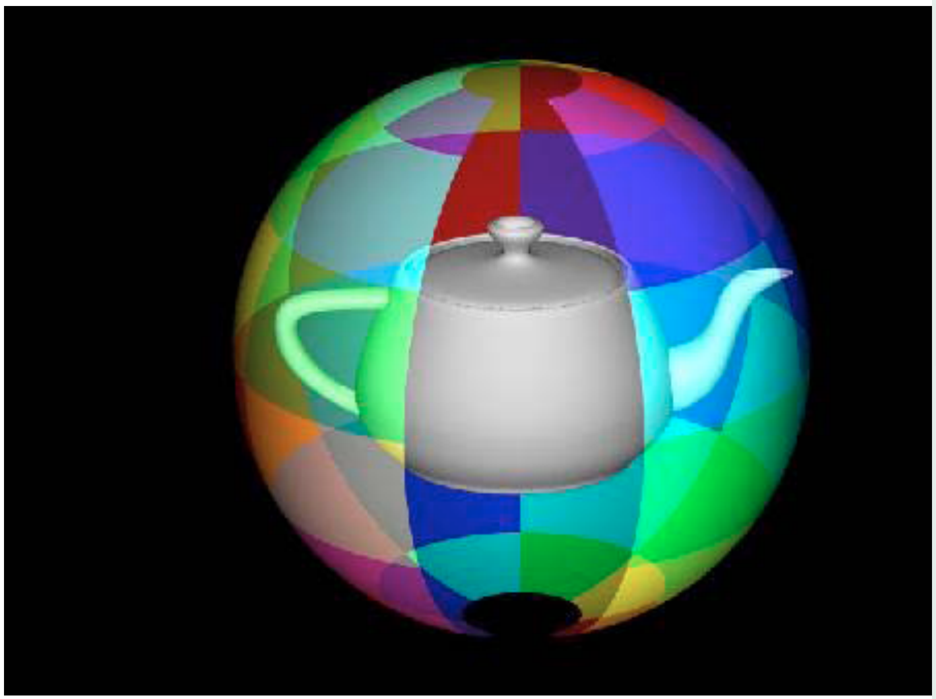



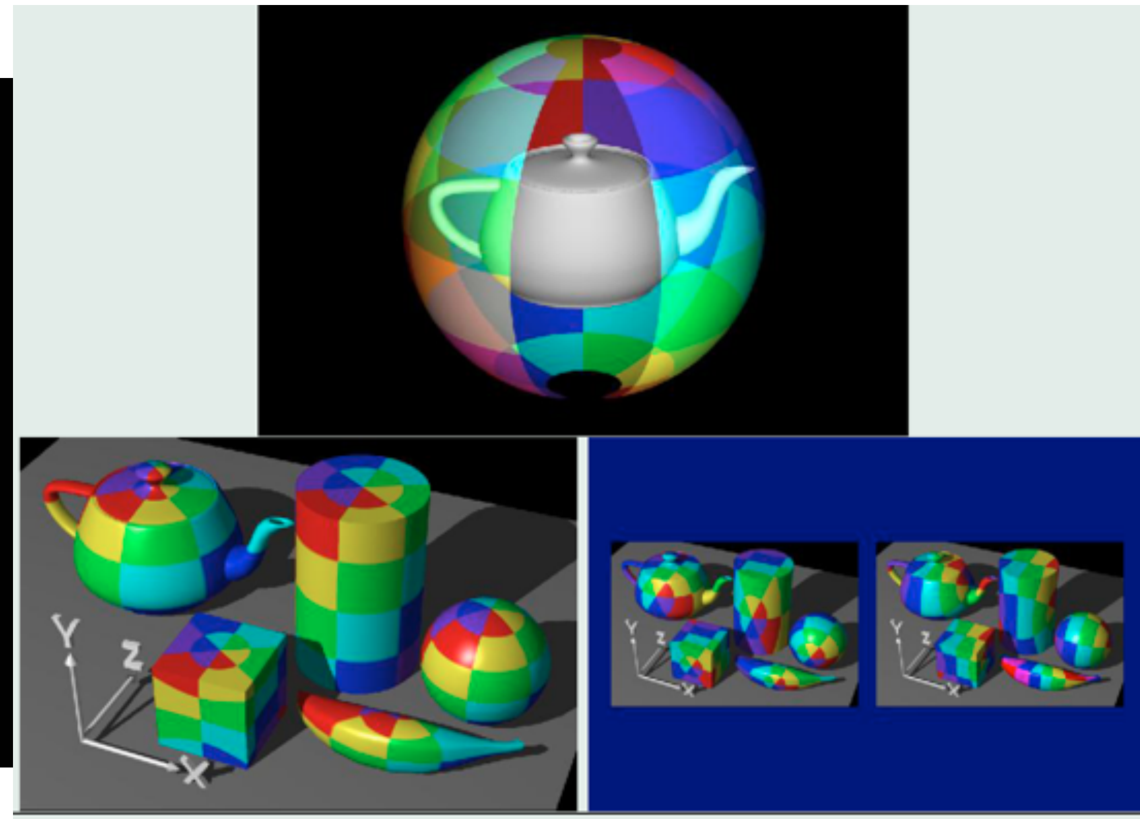
Images courtesy Tito Pagan

Planar Projection

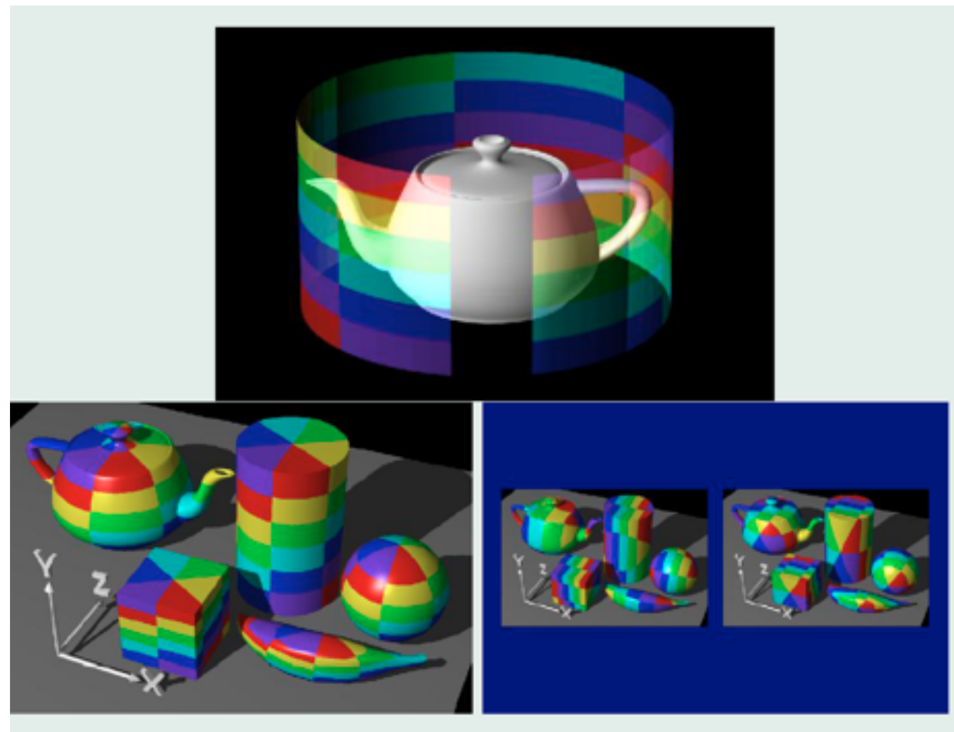


Spherical projection



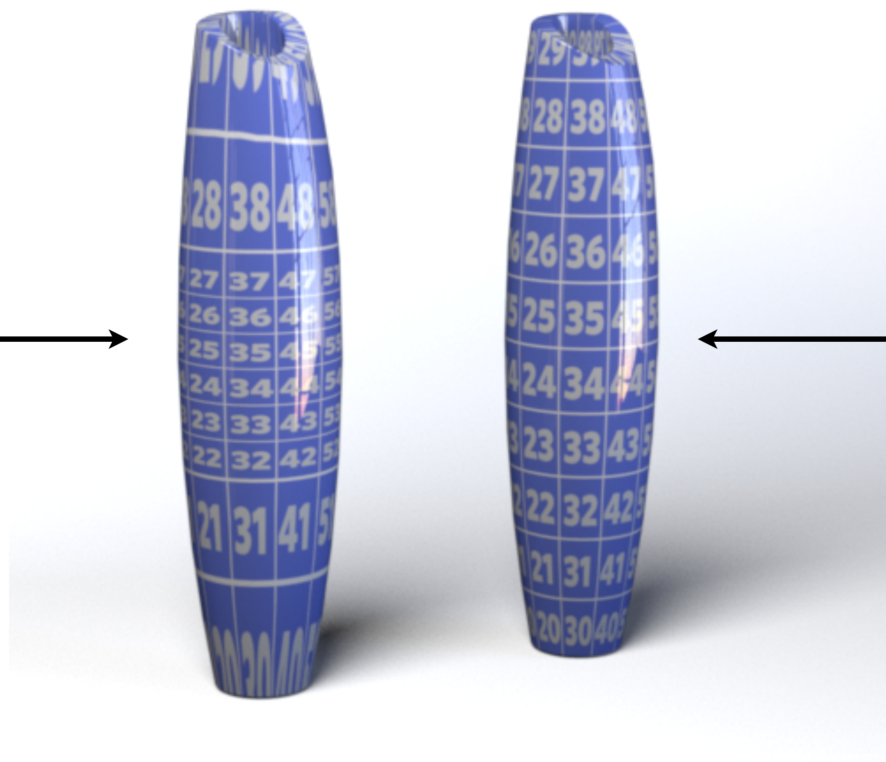


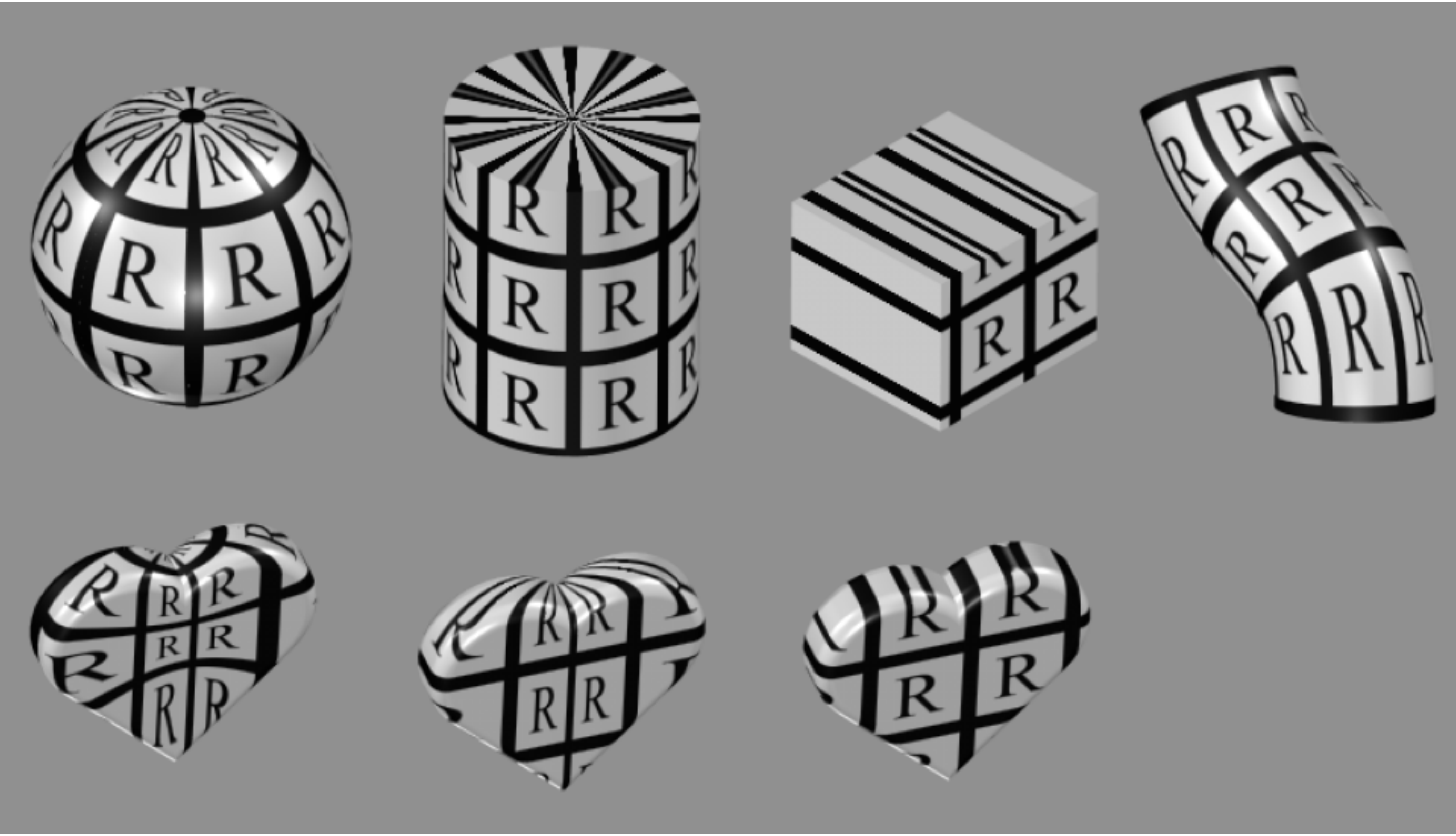
Cylindrical



Examples of coordinate functions

- Cylindrical projection

spherical →  ← cylindrical

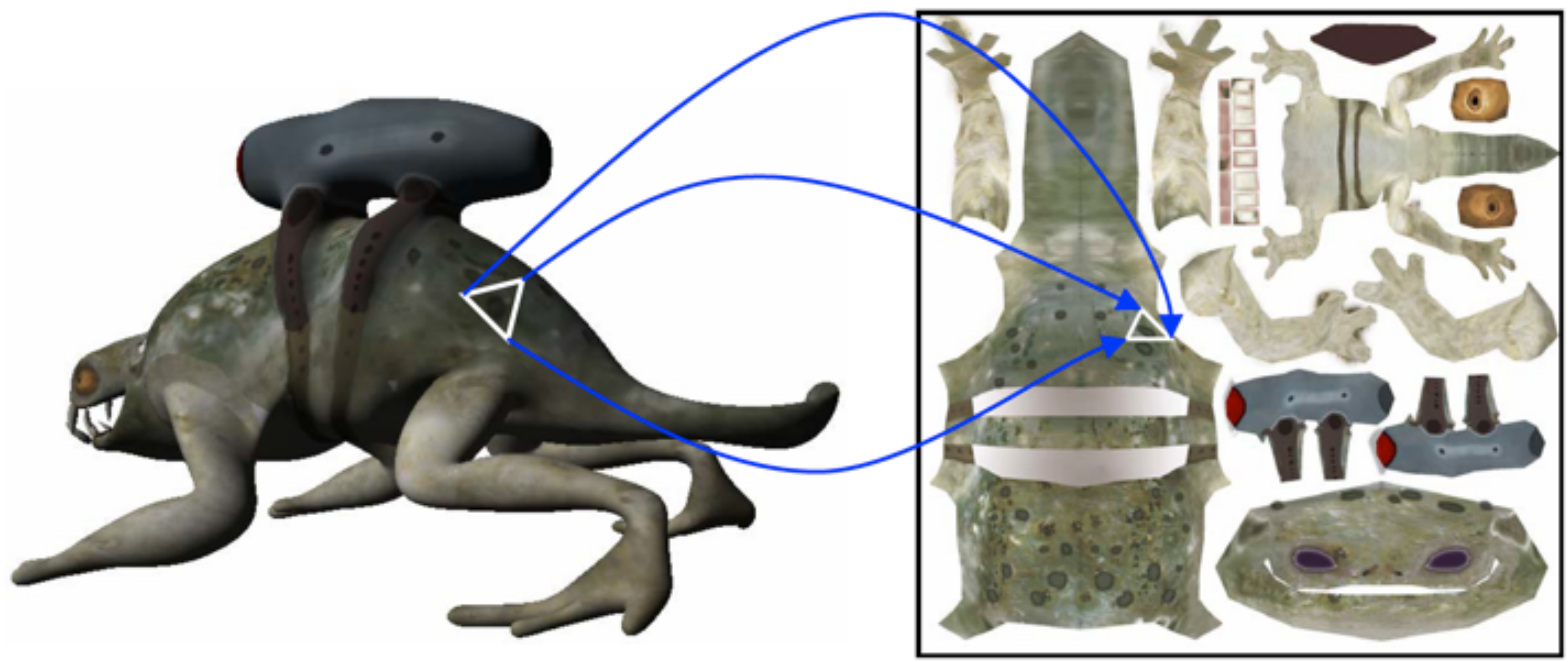


Texture coordinates on meshes

- Texture coordinates become per-vertex data like vertex positions
  - can think of them as a second position: each vertex has a position in 3D space and in 2D texure space
- How to come up with vertex (*u*,*v*)s?
  - use any or all of the methods just discussed
  - use some kind of optimization
    - try to choose vertex (*u*,*v*)s to result in a smooth, low distortion map

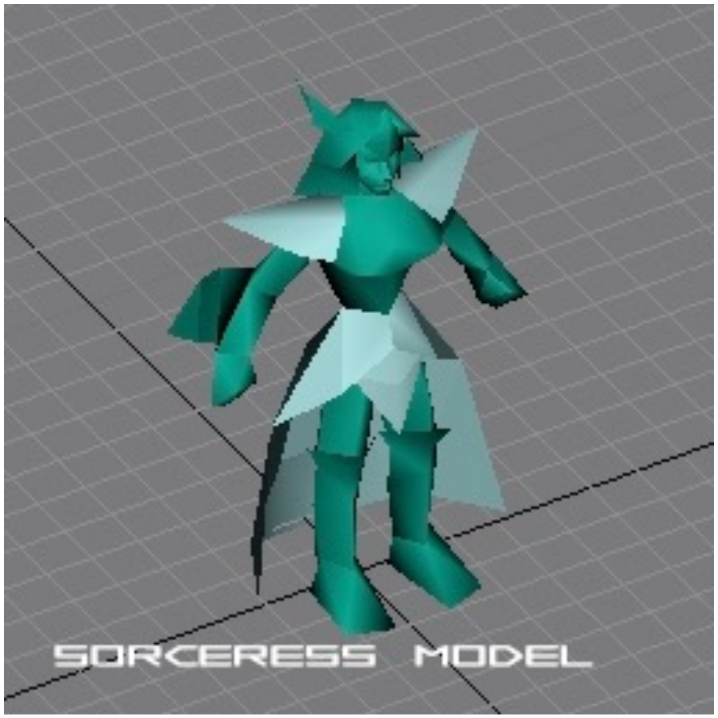
Arbitrary Meshes: uv mapping

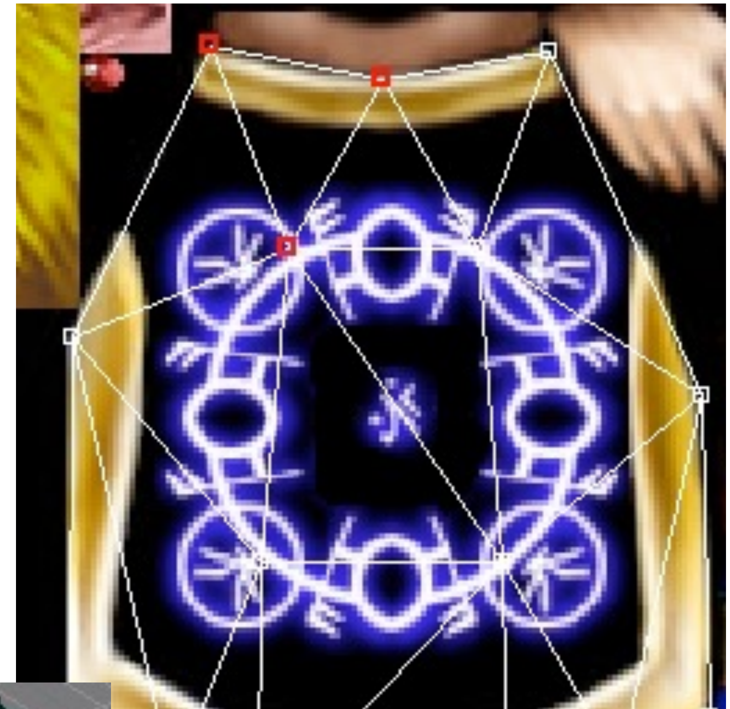
Arbitrary Meshes



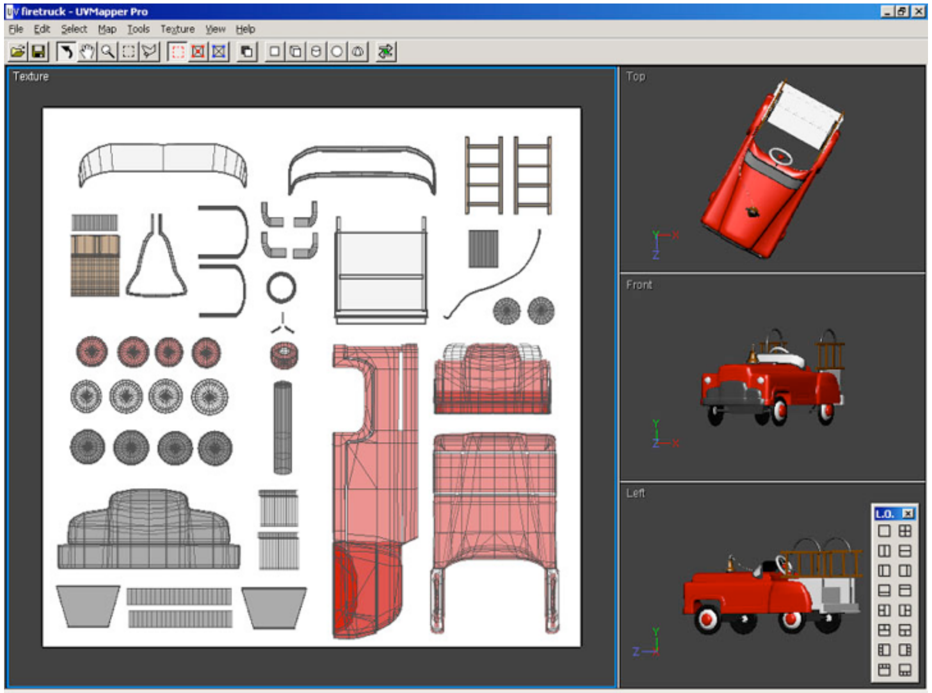
Projector Functions: User-Specified

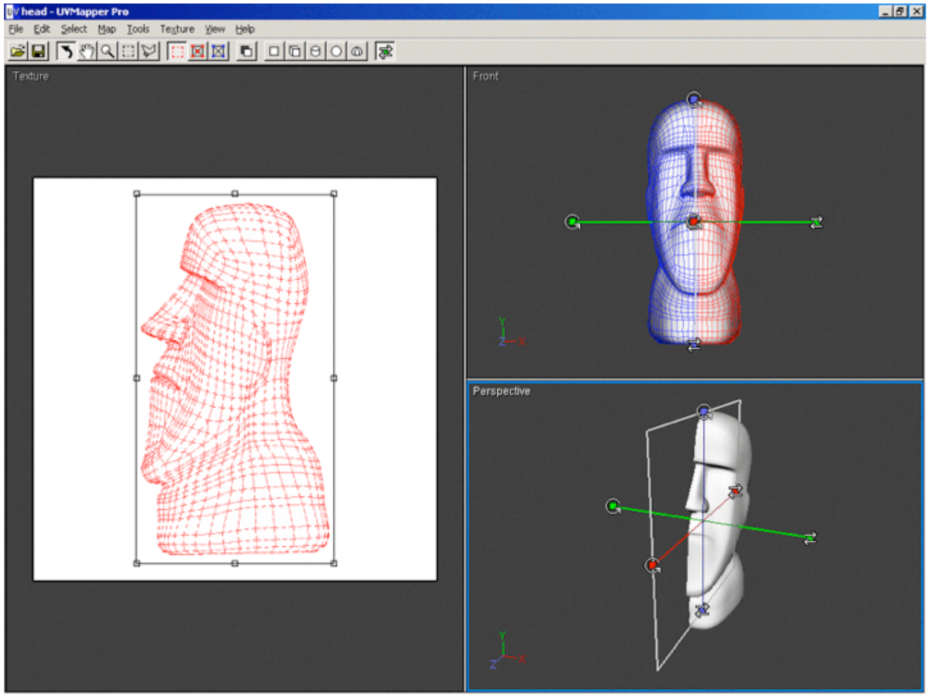
Warcraft III

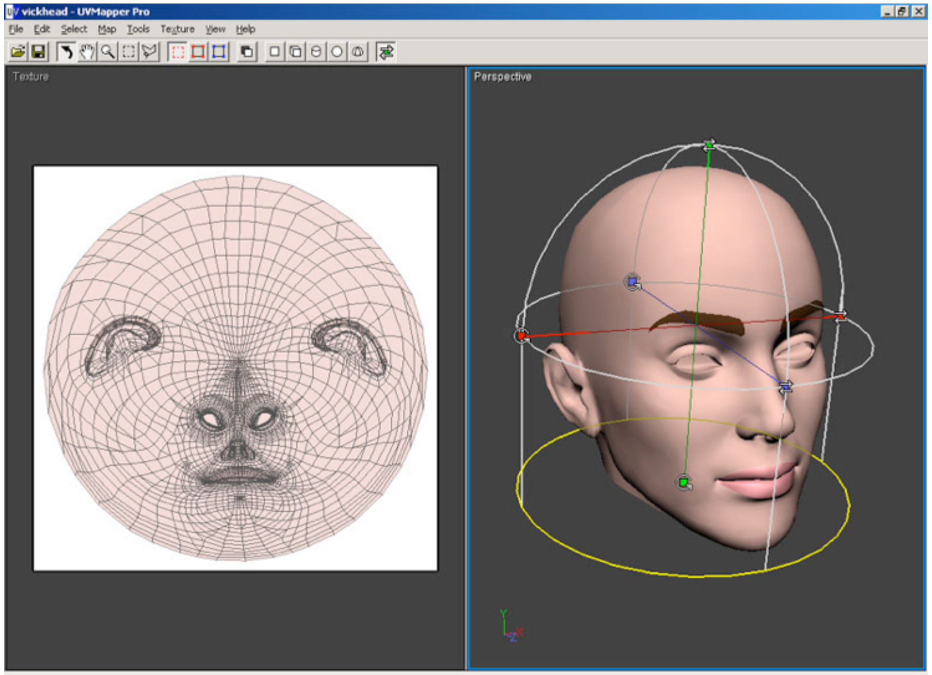




Example: UVMapper







Projector Function: Arbitrary Surfaces