CS230 : Computer Graphics

Lecture 7: Texture Mapping

Tamar Shinar
Computer Science & Engineering
UC Riverside

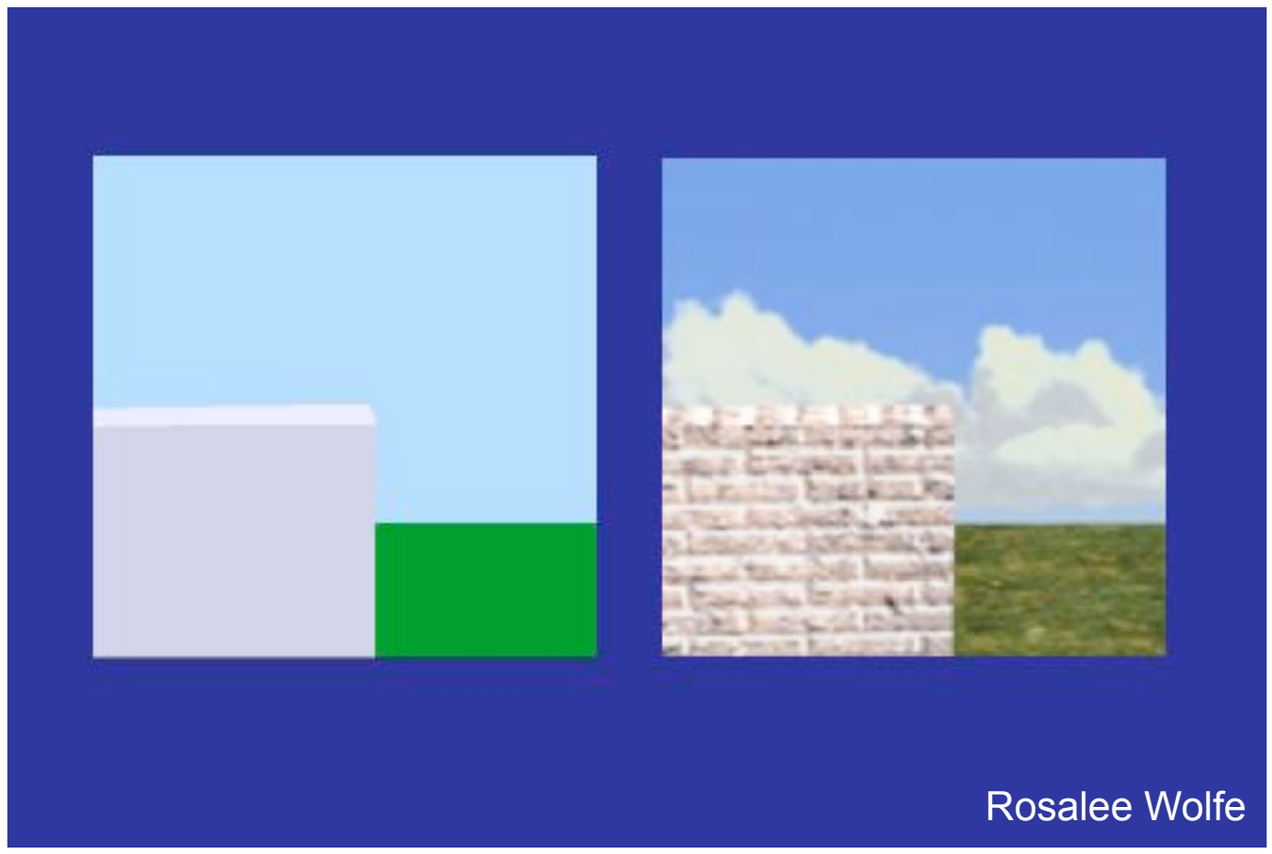
There are limits to geometric modeling

http://www.beinteriordecorator.com

National Geographic

Although modern GPUs can render millions of triangles/ sec, that's not enough sometimes...

Use texture mapping to increase realism through detail

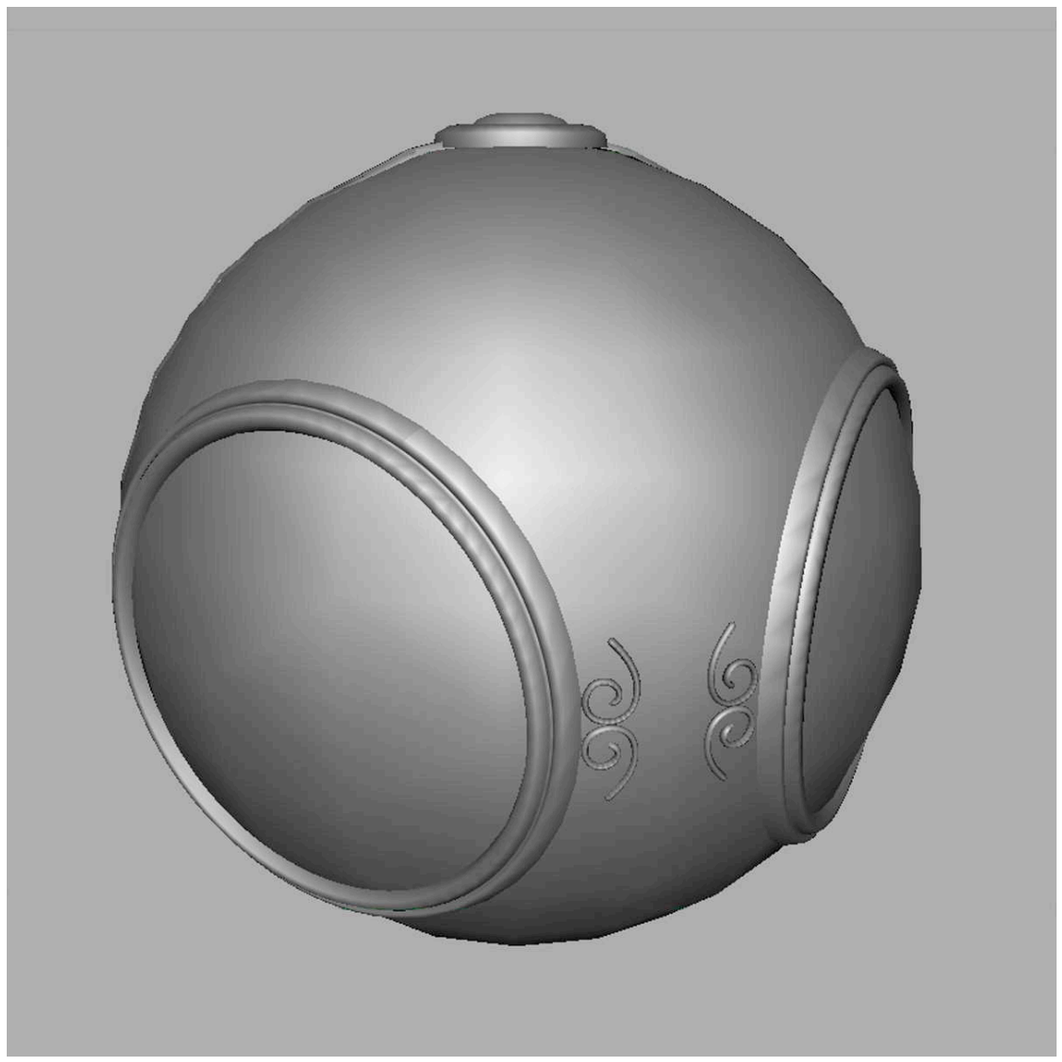


This image is just 8 polygons!

Add visual complexity.

http://www.siggraph.org/education/materials/HyperGraph/mapping/r_wolfe/r_wolfe_mapping_1.htm

Example



Geometry

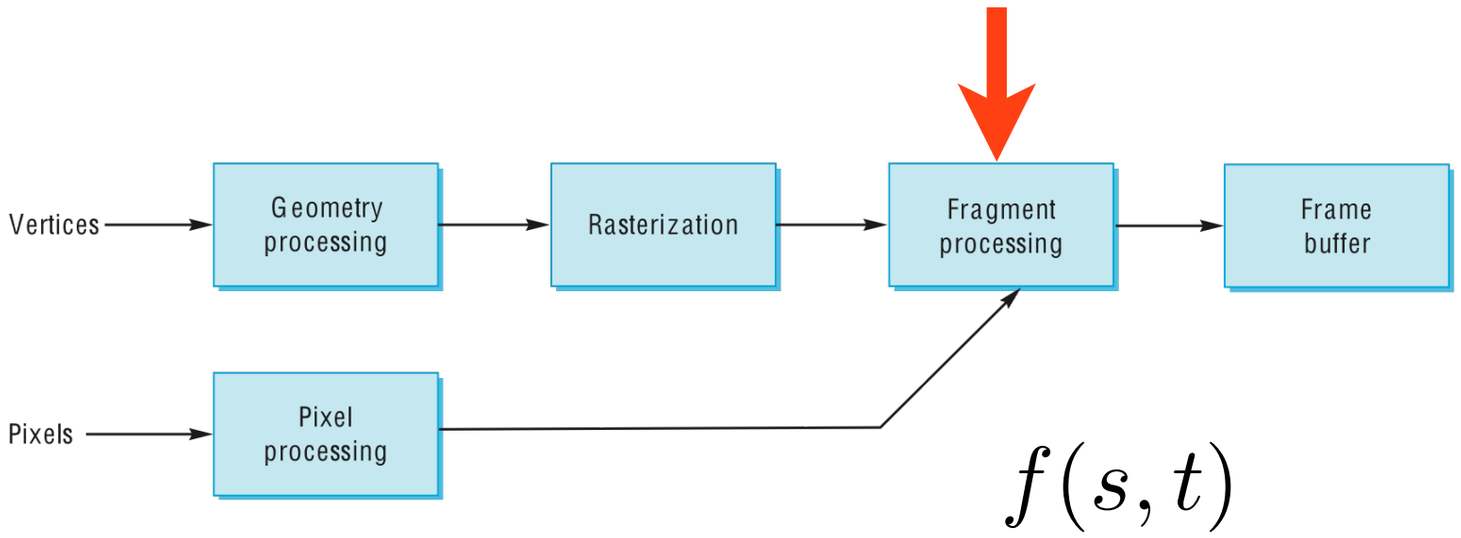
With texture

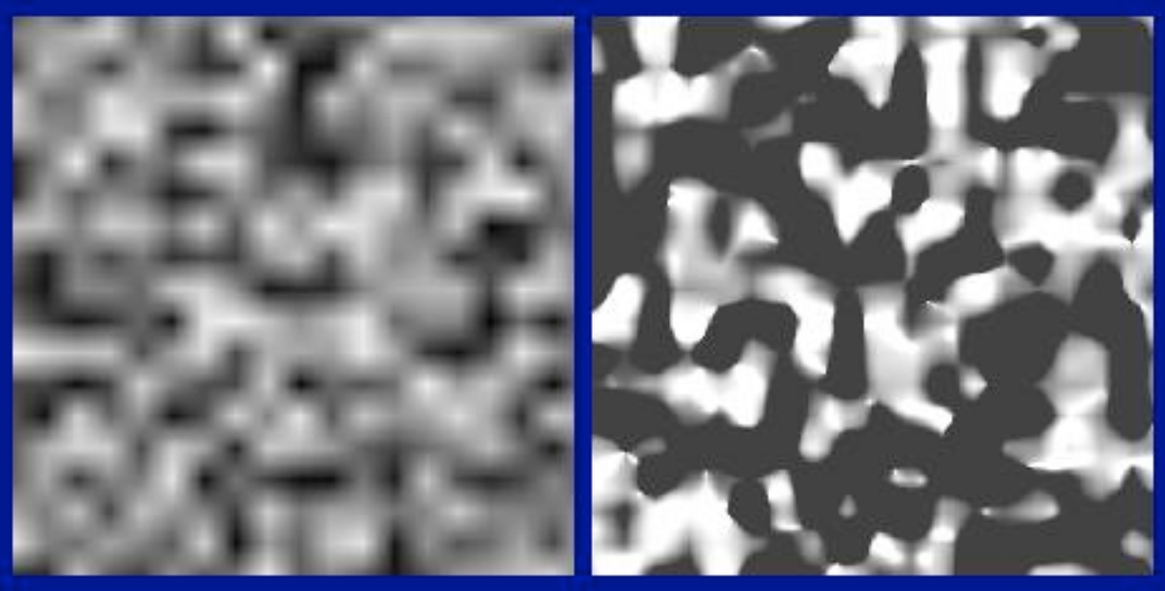
Angel and Shreiner 2012

Example

Pixar - Toy Story

Store 2D images in buffers and lookup pixel reflectances





procedural

photo

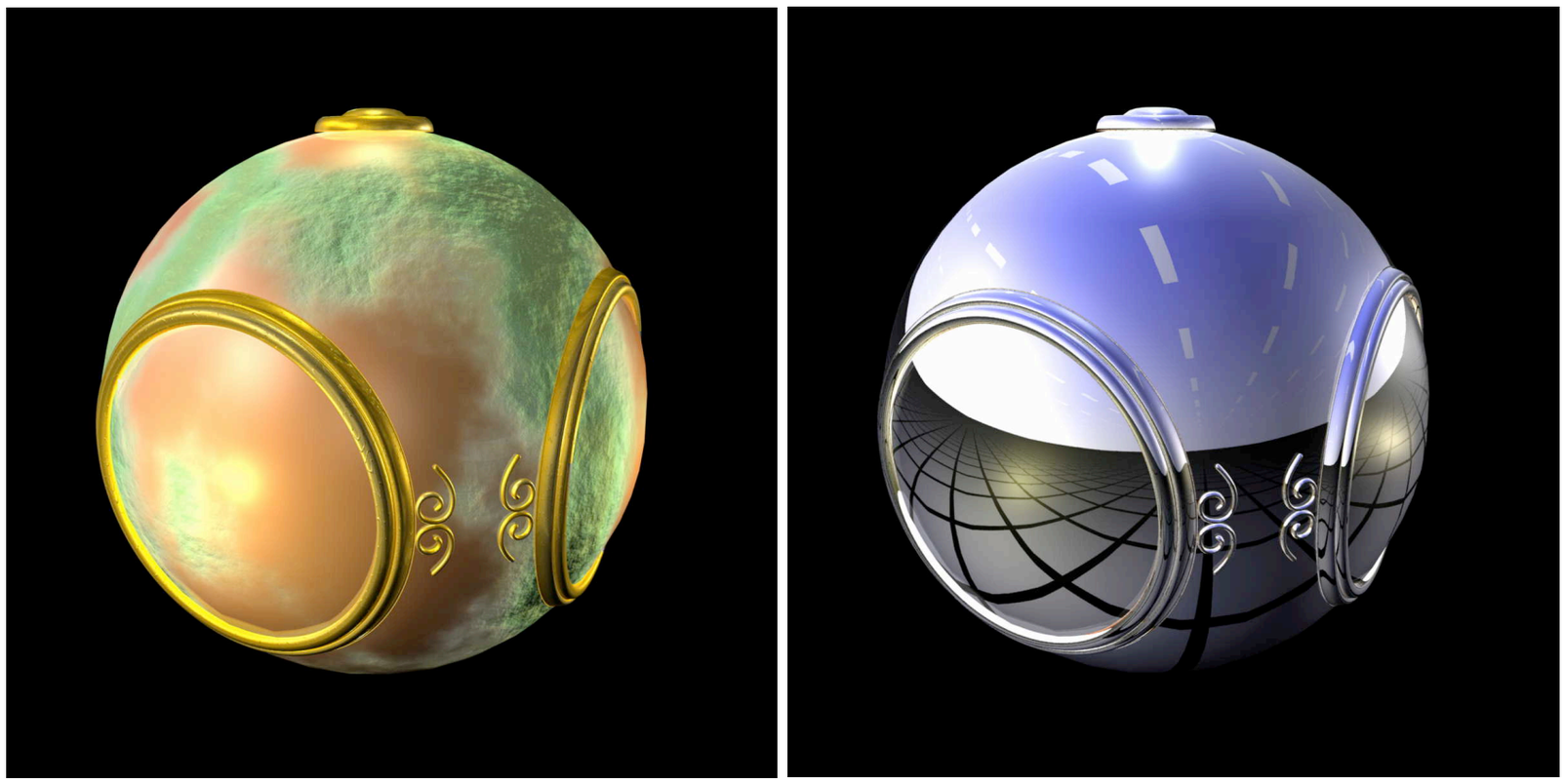
Textures can be anything that you can lookup values in -- photo, procedurally generated, or even a function that computes a value on the fly

3D solid textures

Dong et al., 2008

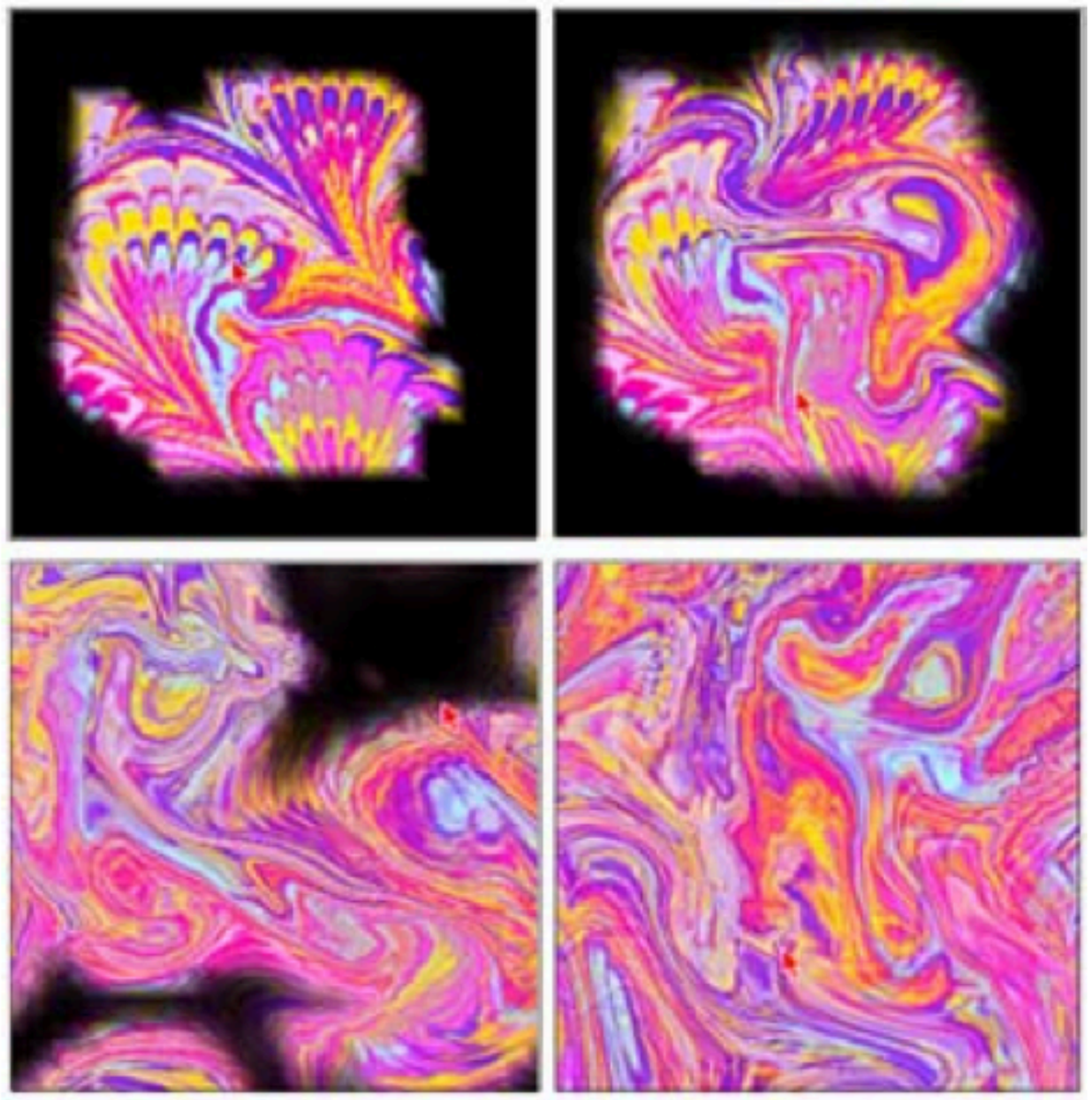
Other uses of textures...

Light maps
Shadow maps
Environment maps
Bump maps
Opacity maps



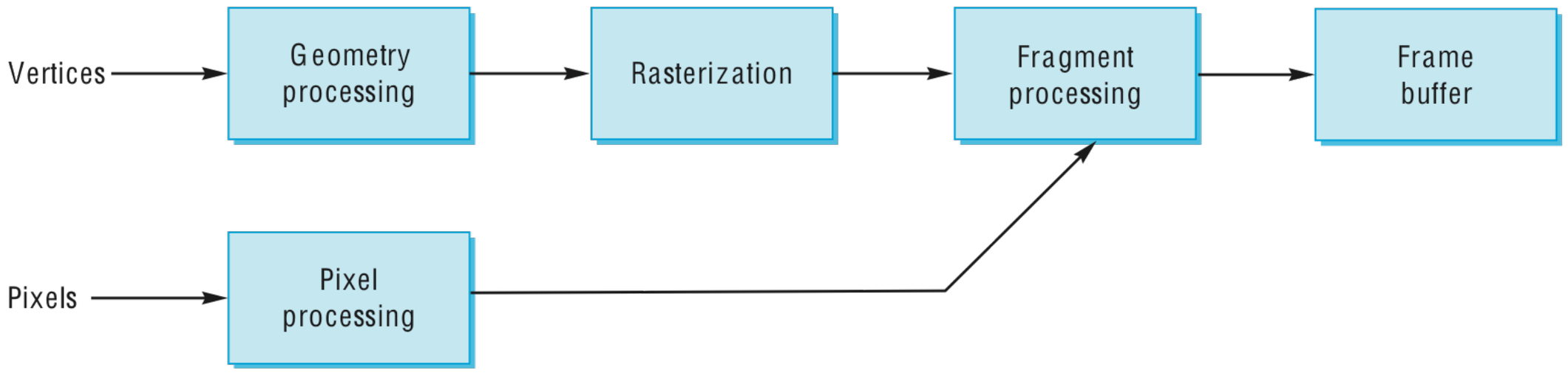
Angel and Shreiner 2012

Texture advection



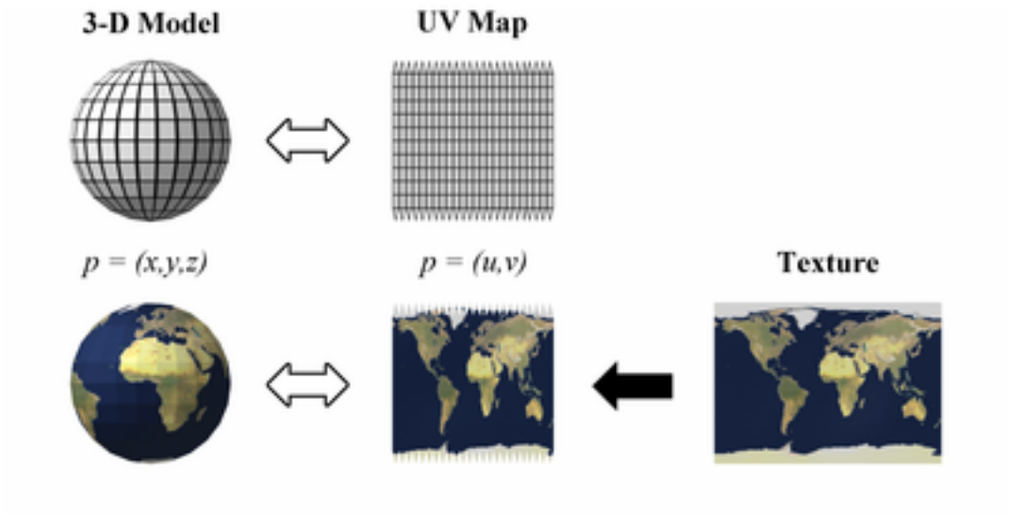
Jos Stam, “Stable Fluids,” SIGGRAPH 99

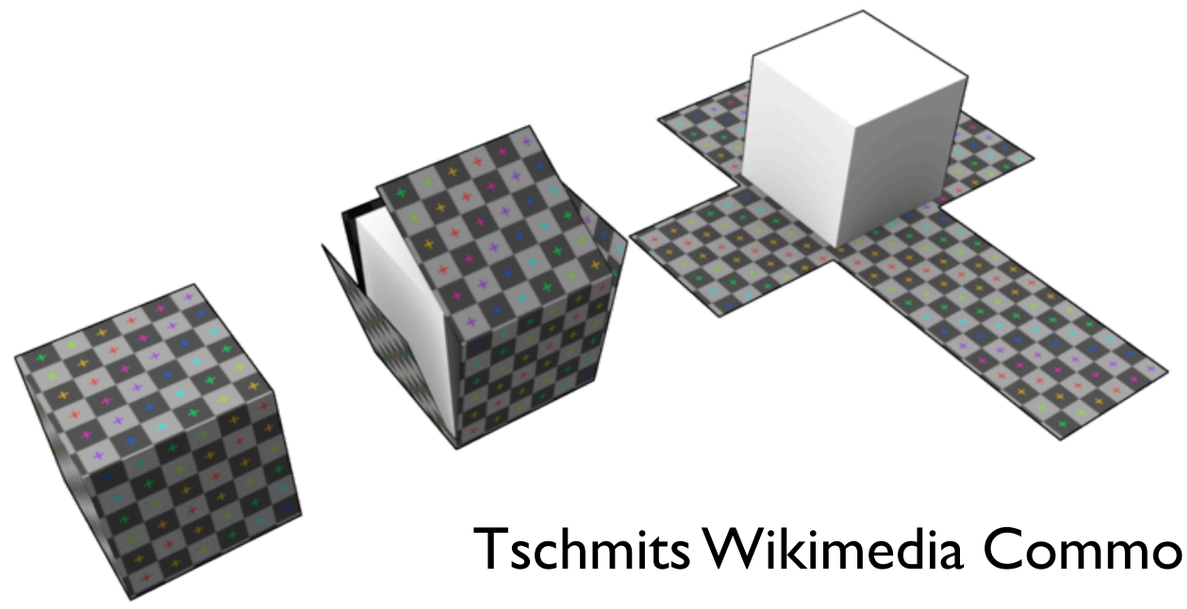
Texture mapping in the OpenGL pipeline



- Geometry and pixels have separate paths through pipeline
- meet in **fragment processing** - where textures are applied
- texture mapping applied at end of pipeline - efficient since relatively few polygons get past clipper

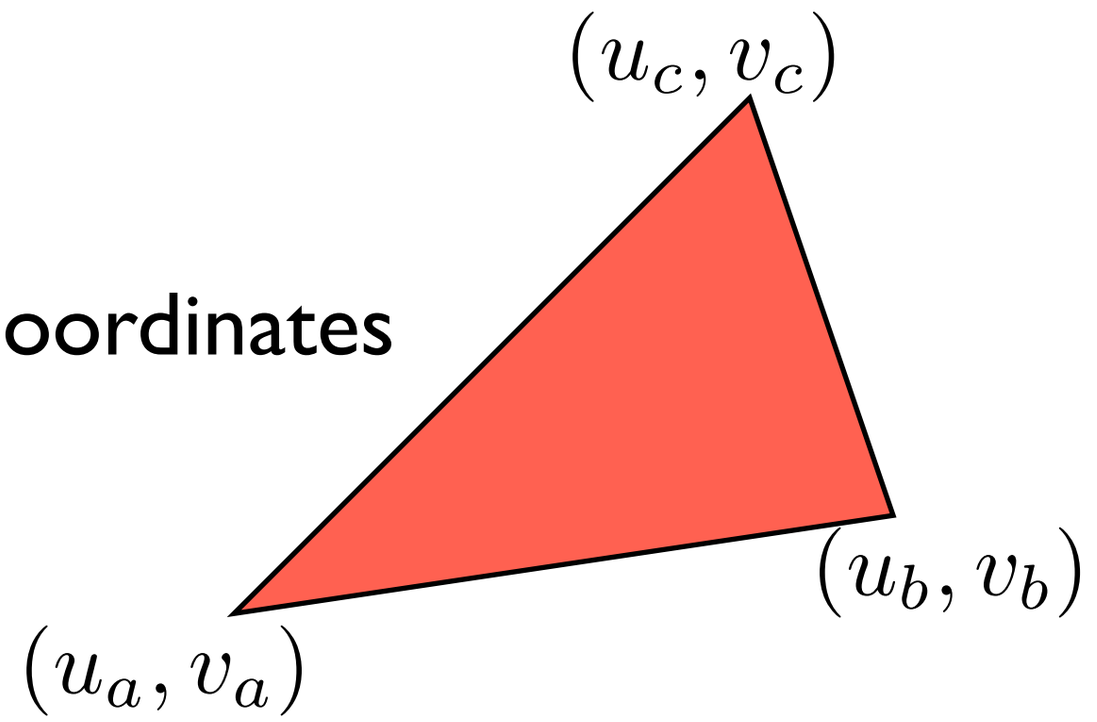
UV Mapping





Tschmits Wikimedia Commons

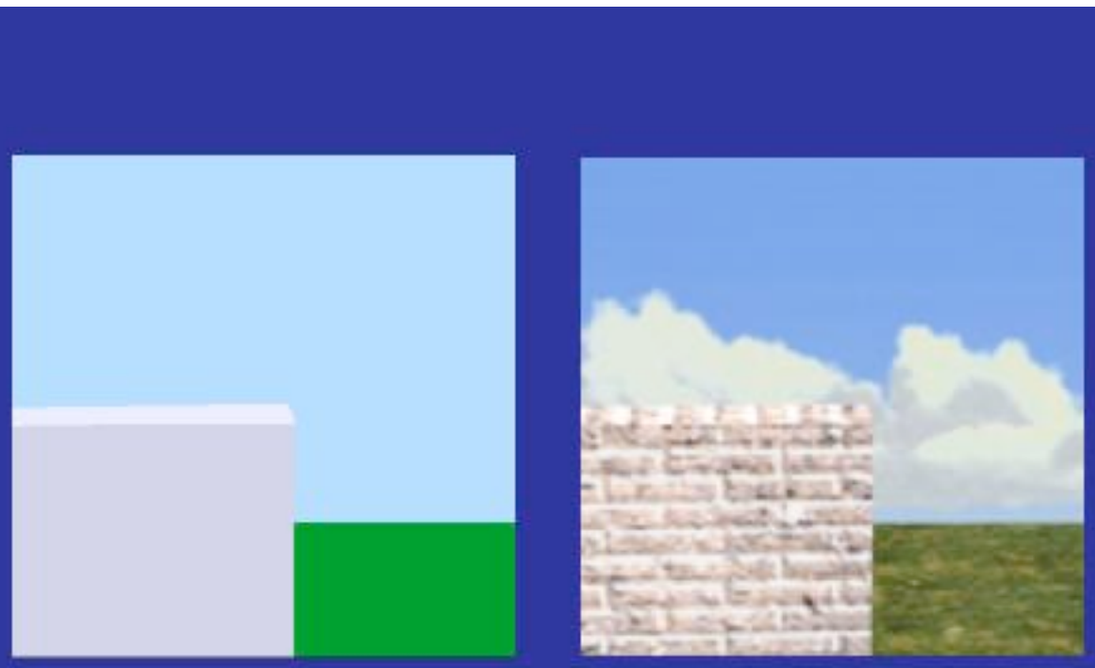
- Texture is parameterized by (u,v)
- Assign polygon vertices texture coordinates
- Interpolate within polygon

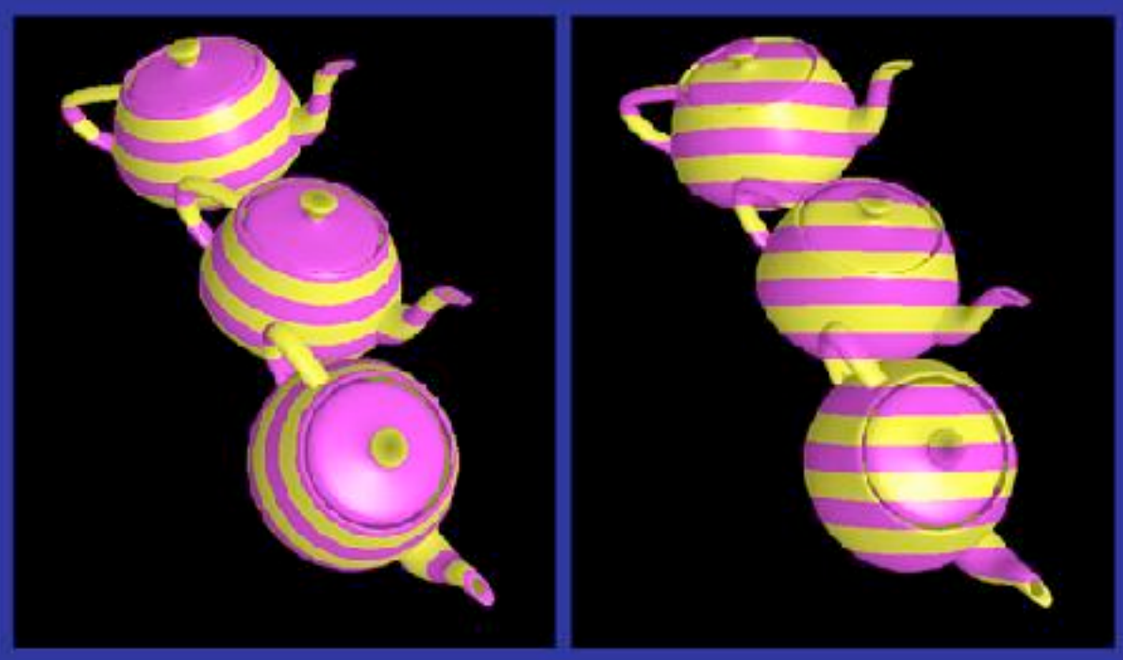


Texture coordinates are per-vertex data - a position in the (u,v) space
can interpolate tex coordinates with barycentric coordinates

The major issues in texture mapping...

- What should the actual mapping be?





easy: rectangular surface

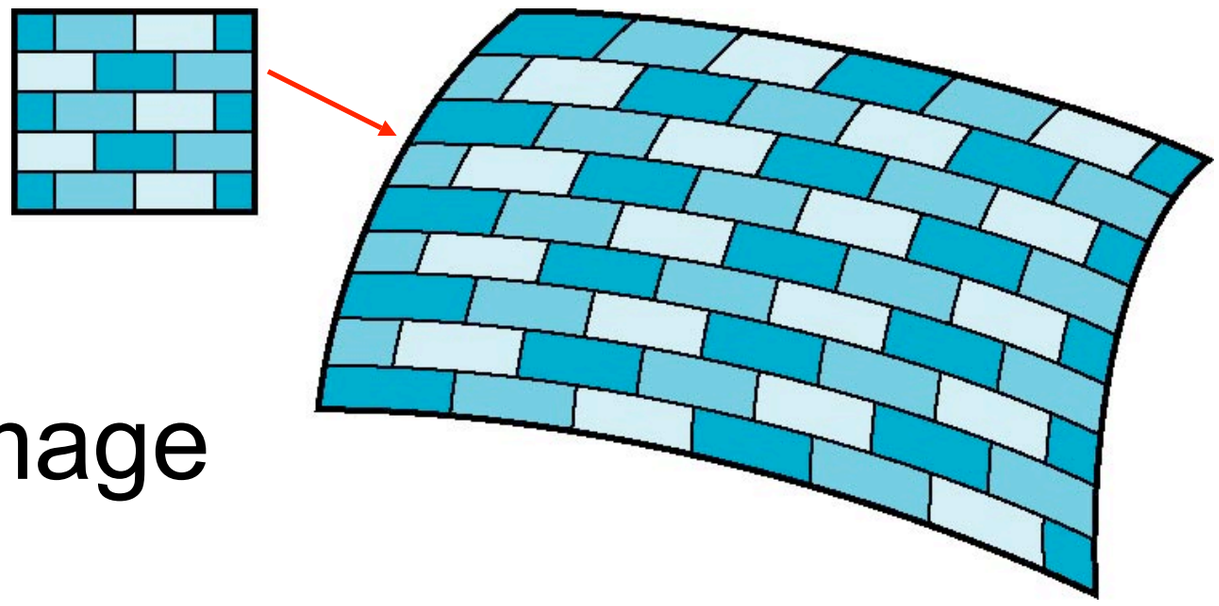
harder: parametric surface

Teapot: Which image looks better? The image on the left uses **object coordinates** in the texture mapping – this makes more sense. The image on the **right** uses **world coordinates** – texture ends up changing relative to the object

want a nice map that doesn't look distorted

Is it simple?

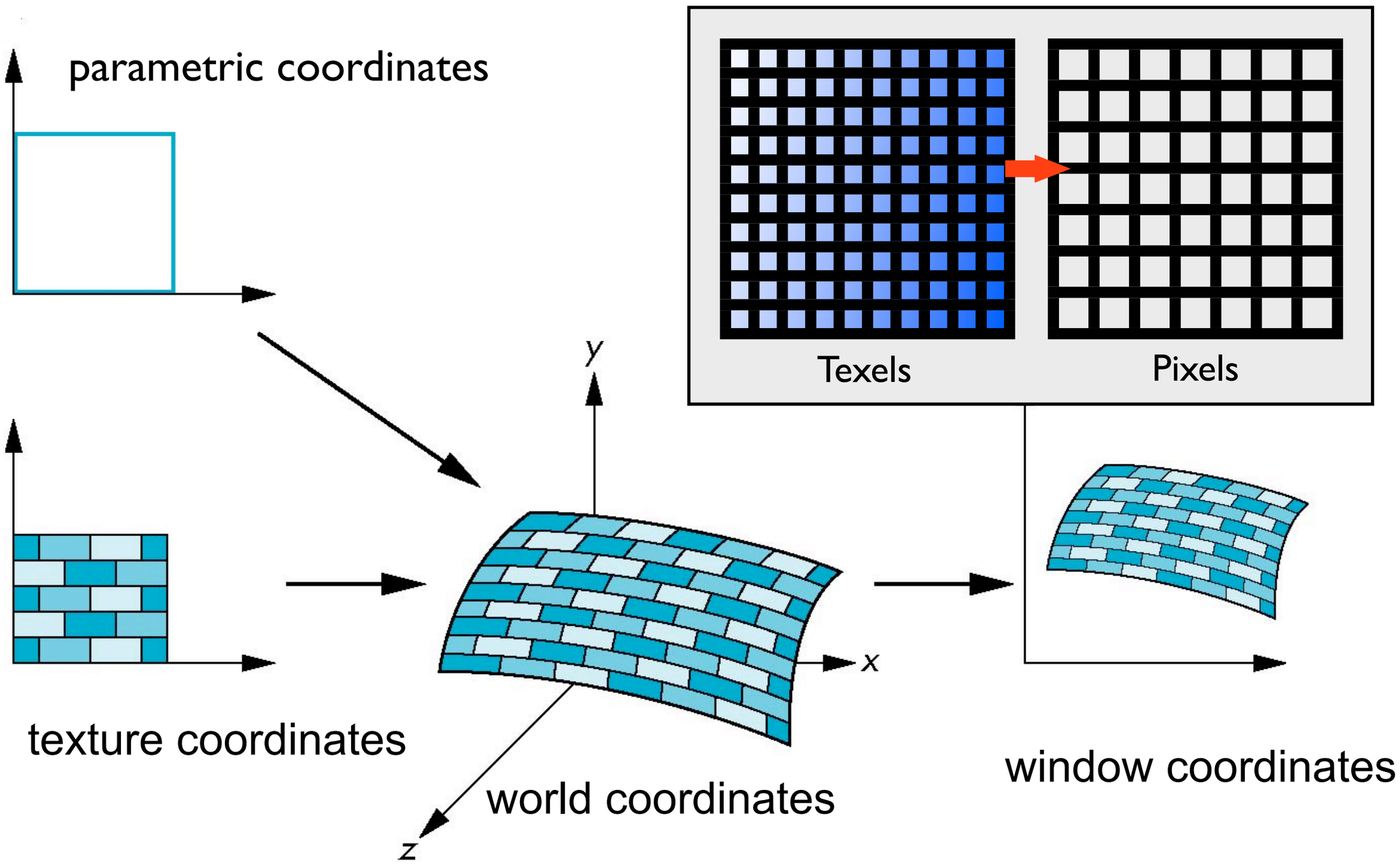
- Although the idea is simple---map an image to a surface---there are 3 or 4 coordinate systems involved



2D image

3D surface

Texture Mapping



- Parametric coordinates
  - May be used to model curves and surfaces
- Texture coordinates
  - Used to identify points in the image to be mapped
- Object or World Coordinates
  - Conceptually, where the mapping takes place
- Window Coordinates
  - Where the final image is really produced

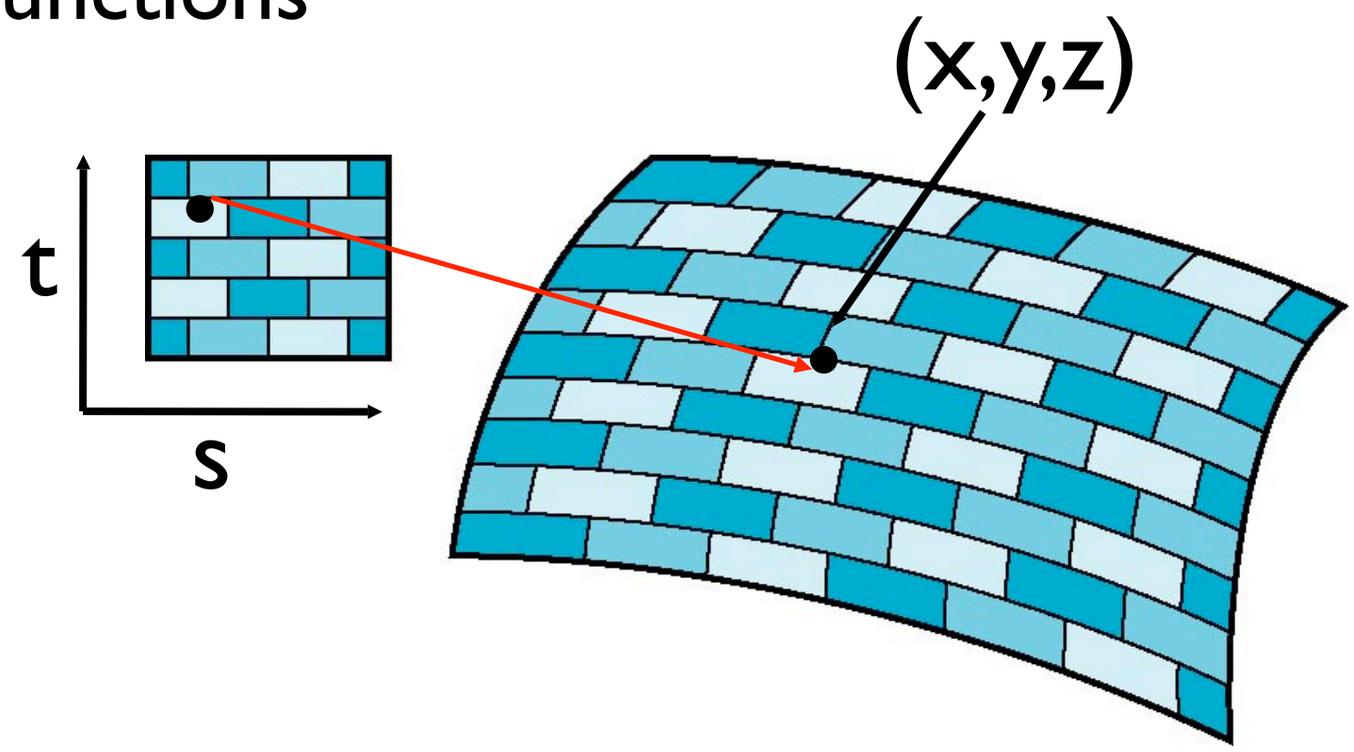
Mapping Functions

- Basic problem is how to find the maps
- Consider mapping from texture coordinates to a point a surface
- Appear to need three functions

$$x = x(s,t)$$

$$y = y(s,t)$$

$$z = z(s,t)$$



- But we really want

to go the other way

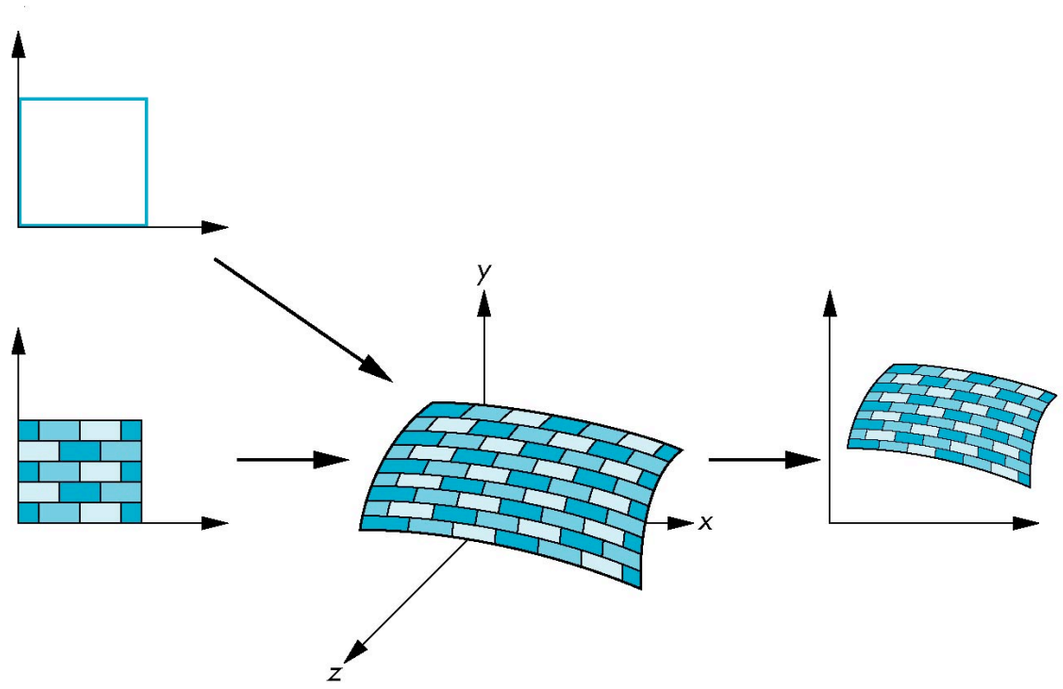
Backward Mapping

- We really want to go backwards
  - Given a pixel, we want to know to which point on an object it corresponds
  - Given a point on an object, we want to know to which point in the texture it corresponds
- Need a map of the form

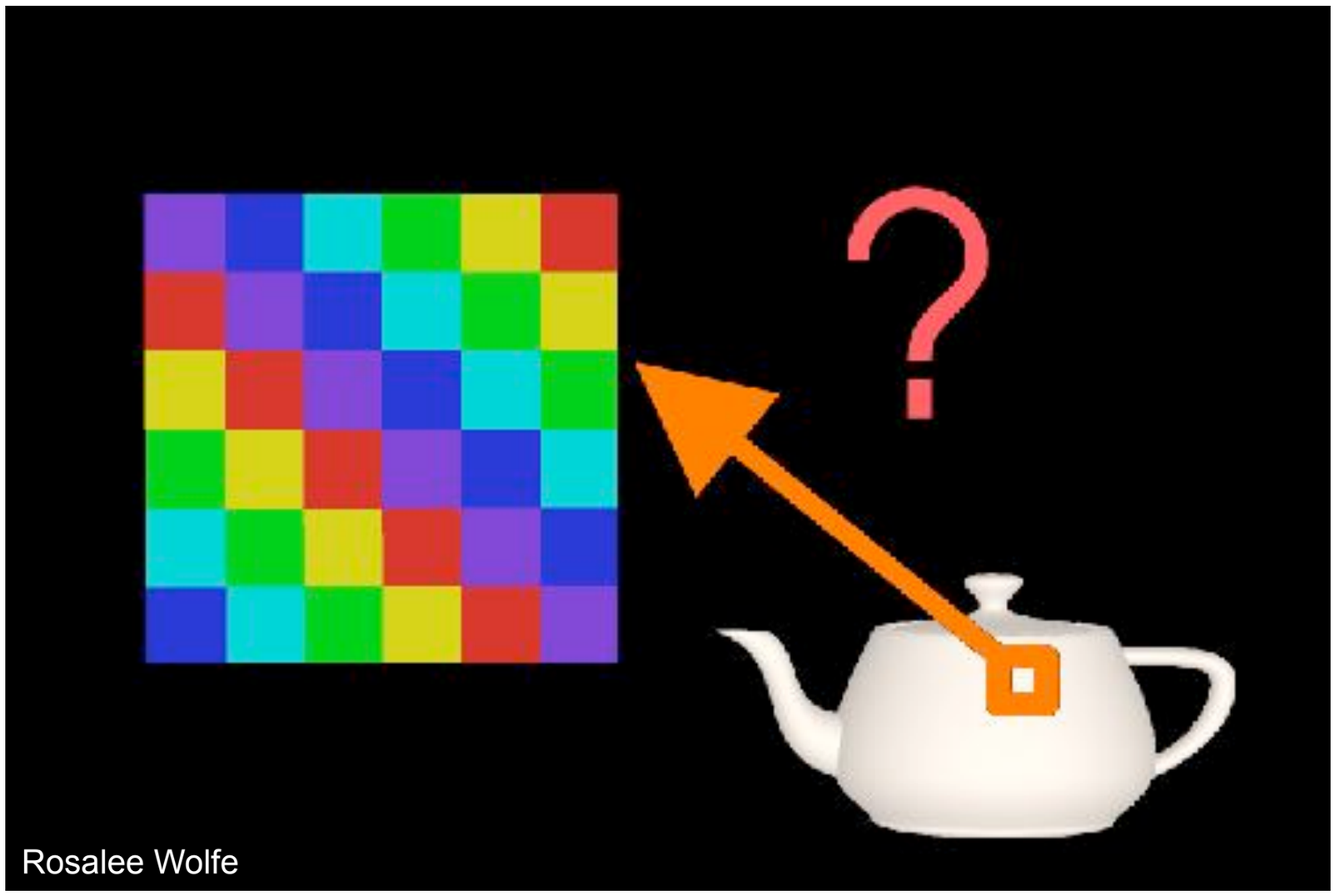
$$s = s(x,y,z)$$

$$t = t(x,y,z)$$

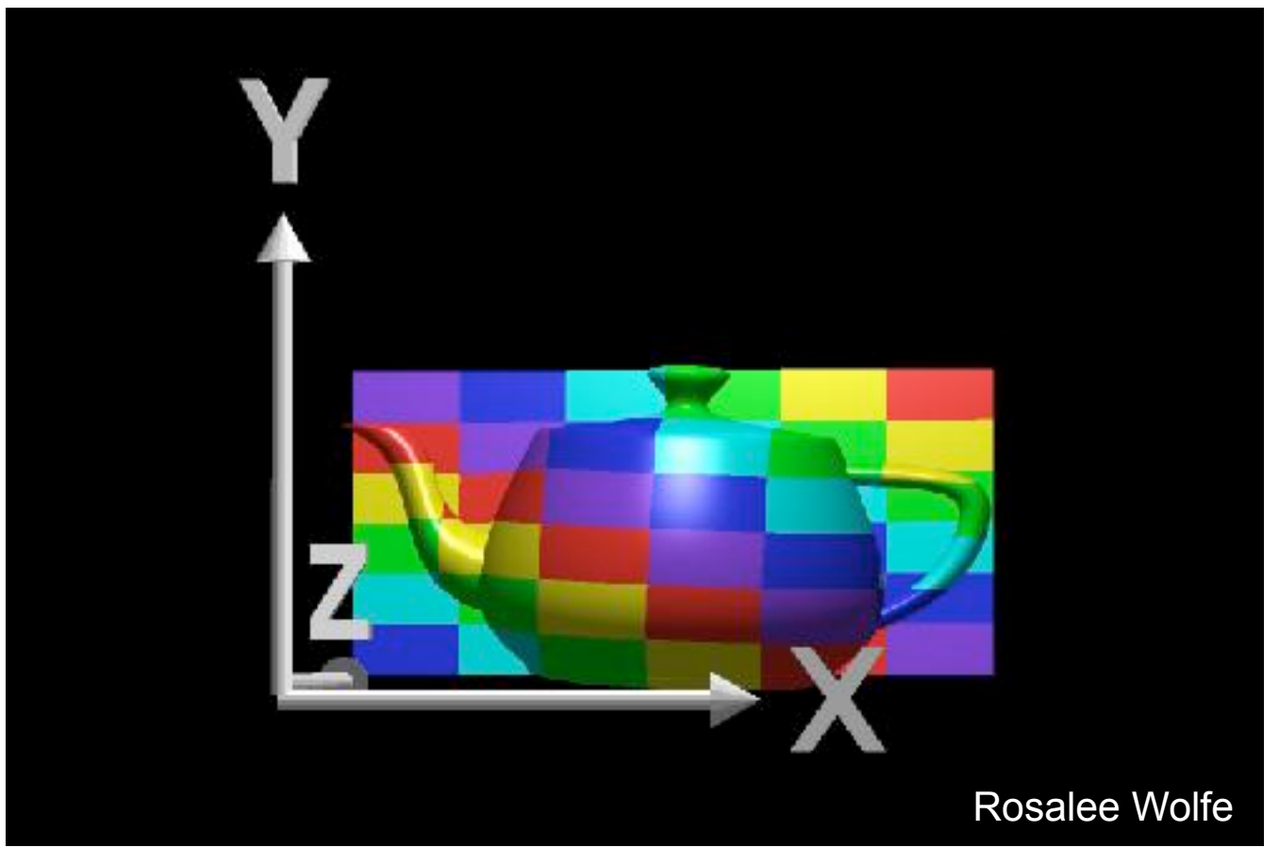
- Such functions are difficult to find in general

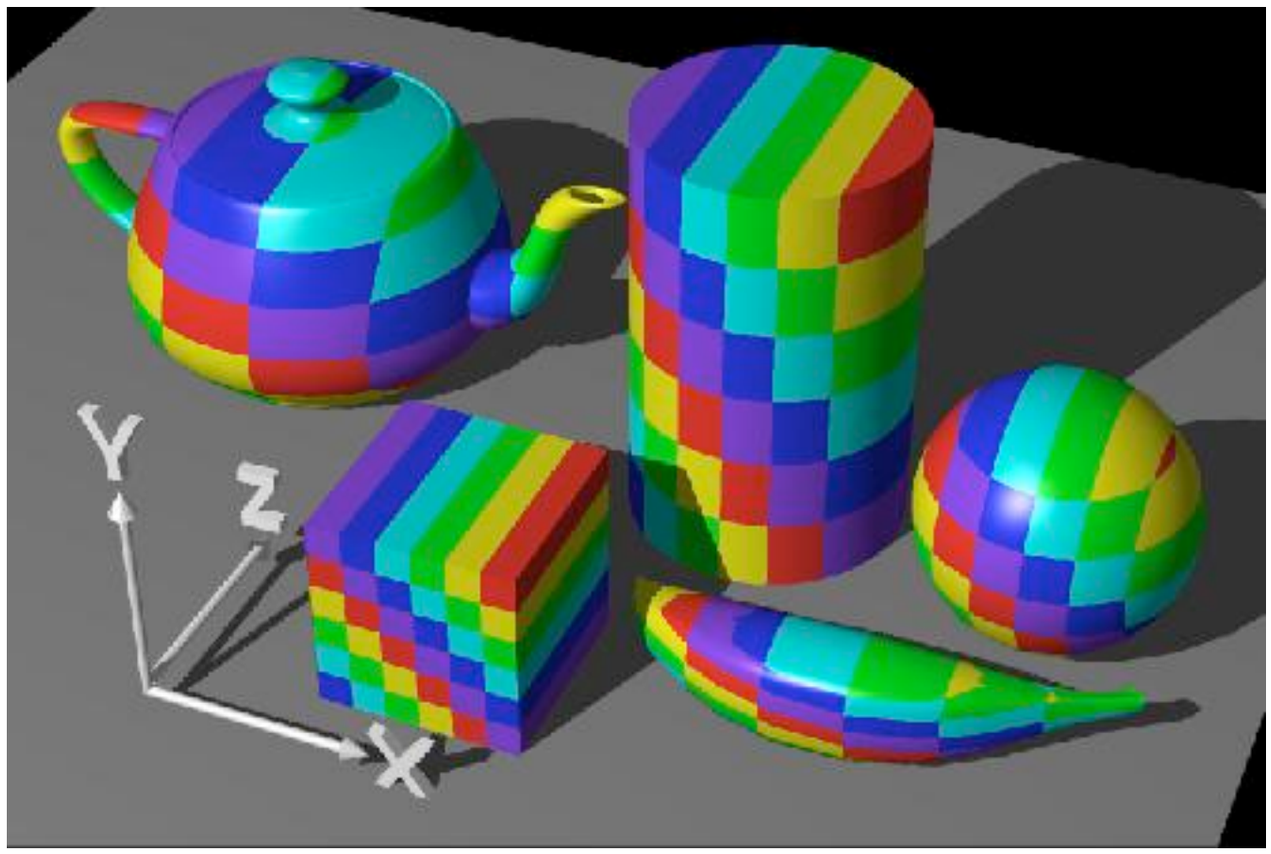


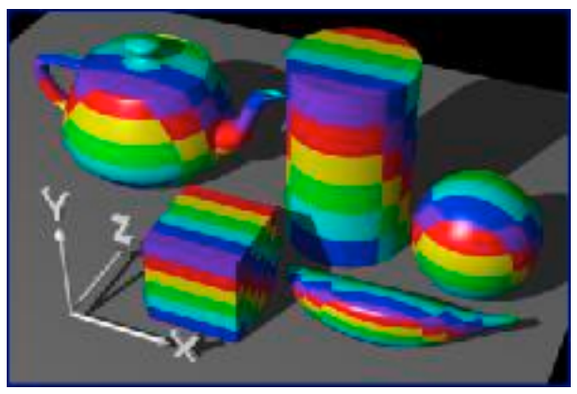
Given a point on the object **(x,y,z)**, what point **(u,v)** in the texture we use?

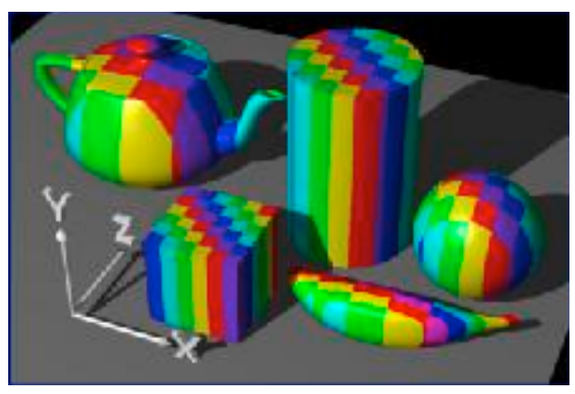


Example: planar mapping



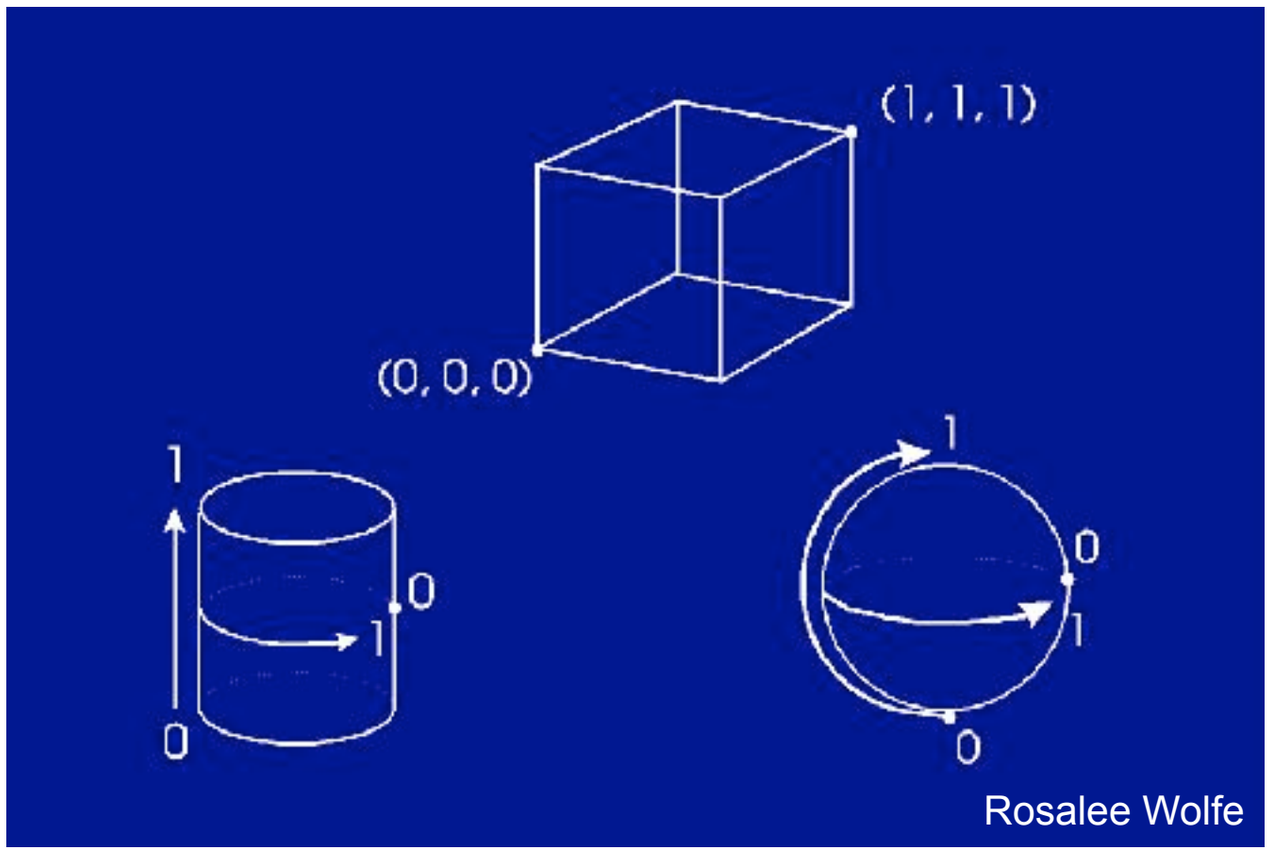






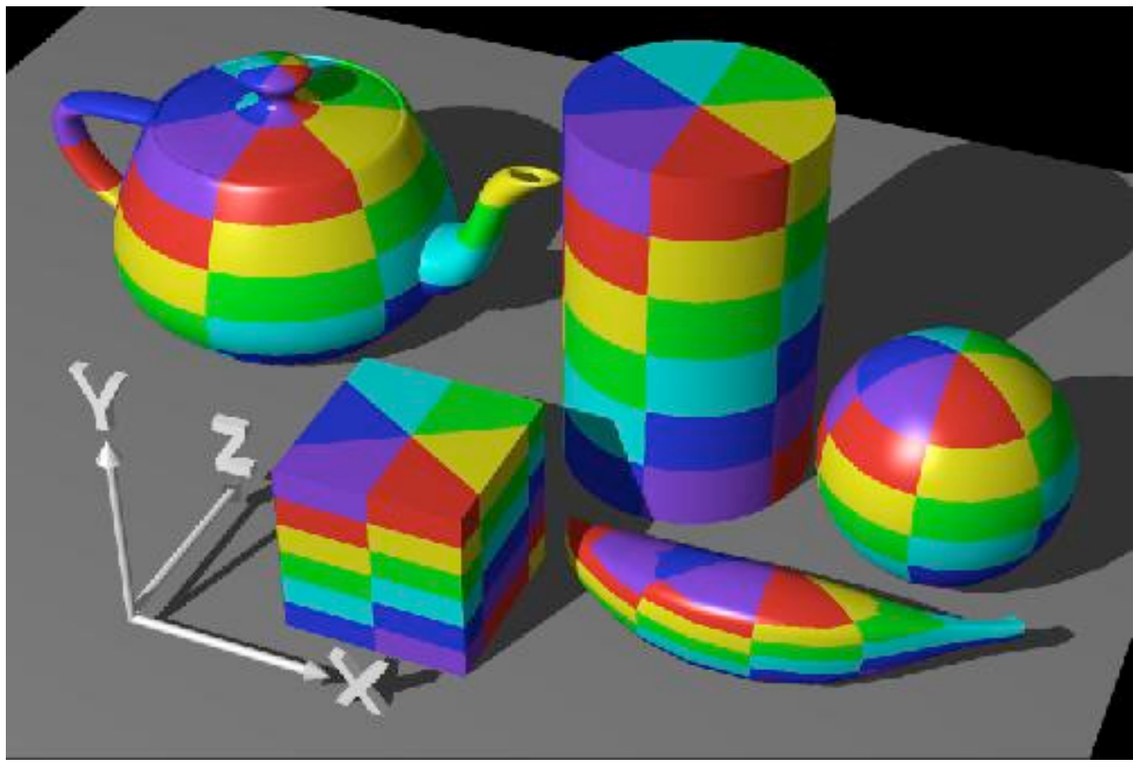
Intermediate surfaces

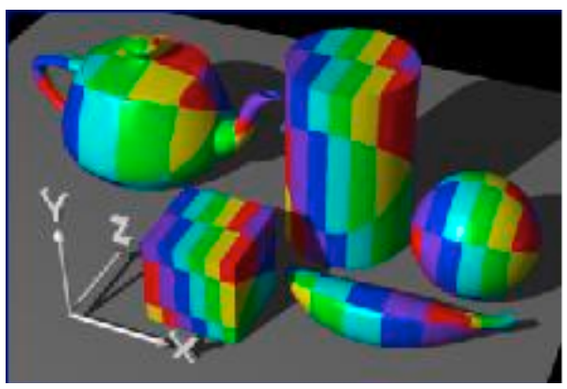
First map the texture to a simpler, intermediate surface

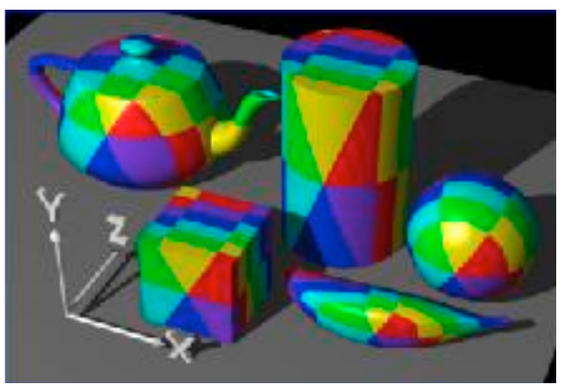


Cylindrical mapping

(x,y,z) -> (theta, h) -> (u,v)



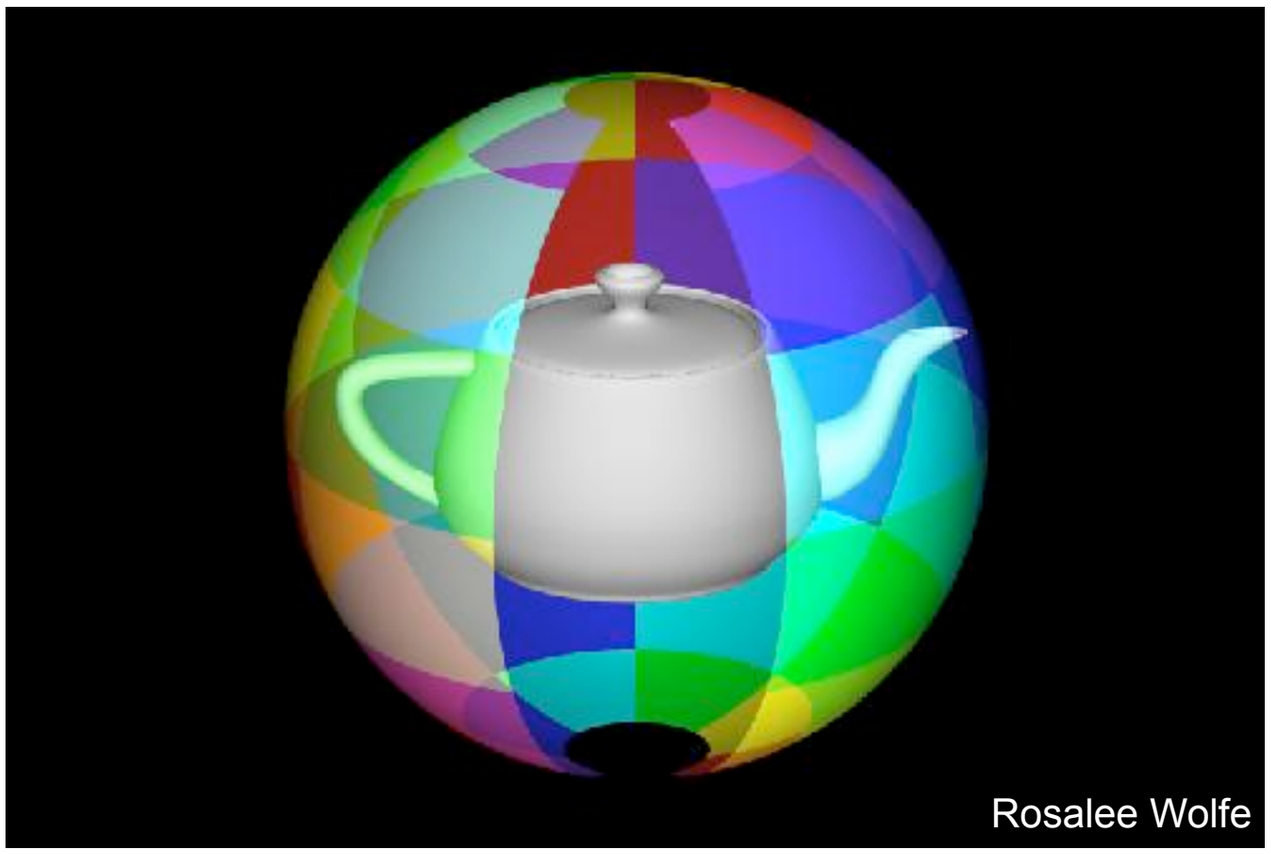


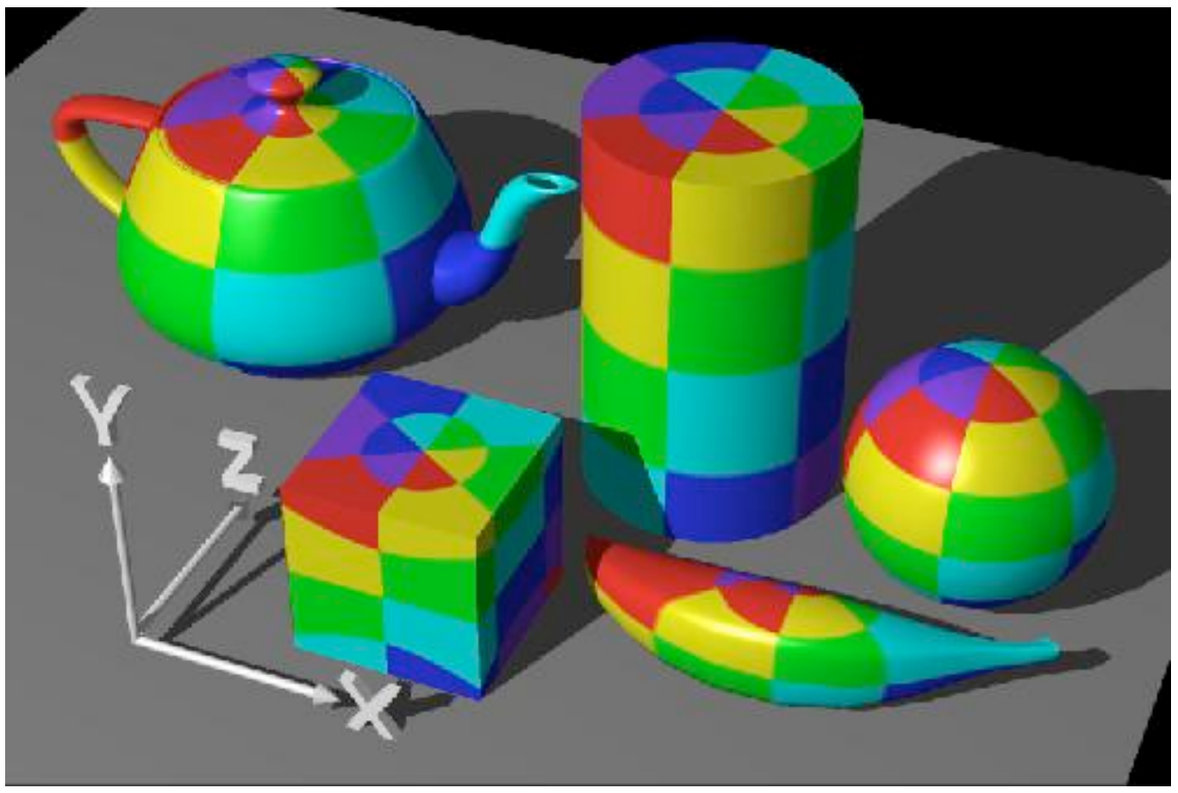


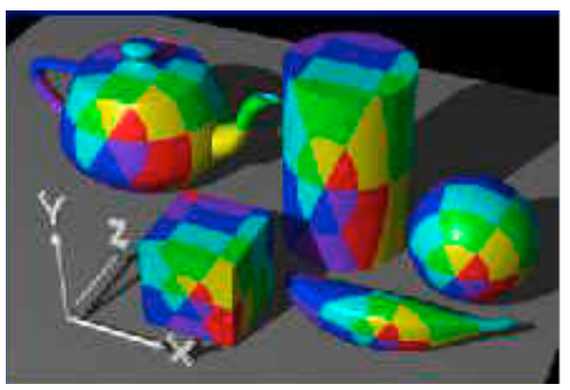
- note “pie slice” phenomena
- which coordinate axis is parallel to the cylinder axis?

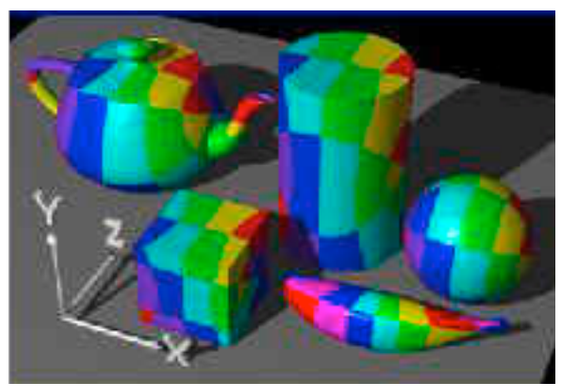
Spherical Mapping

(x,y,z) -> (latitude,longitude)
-> (u,v)



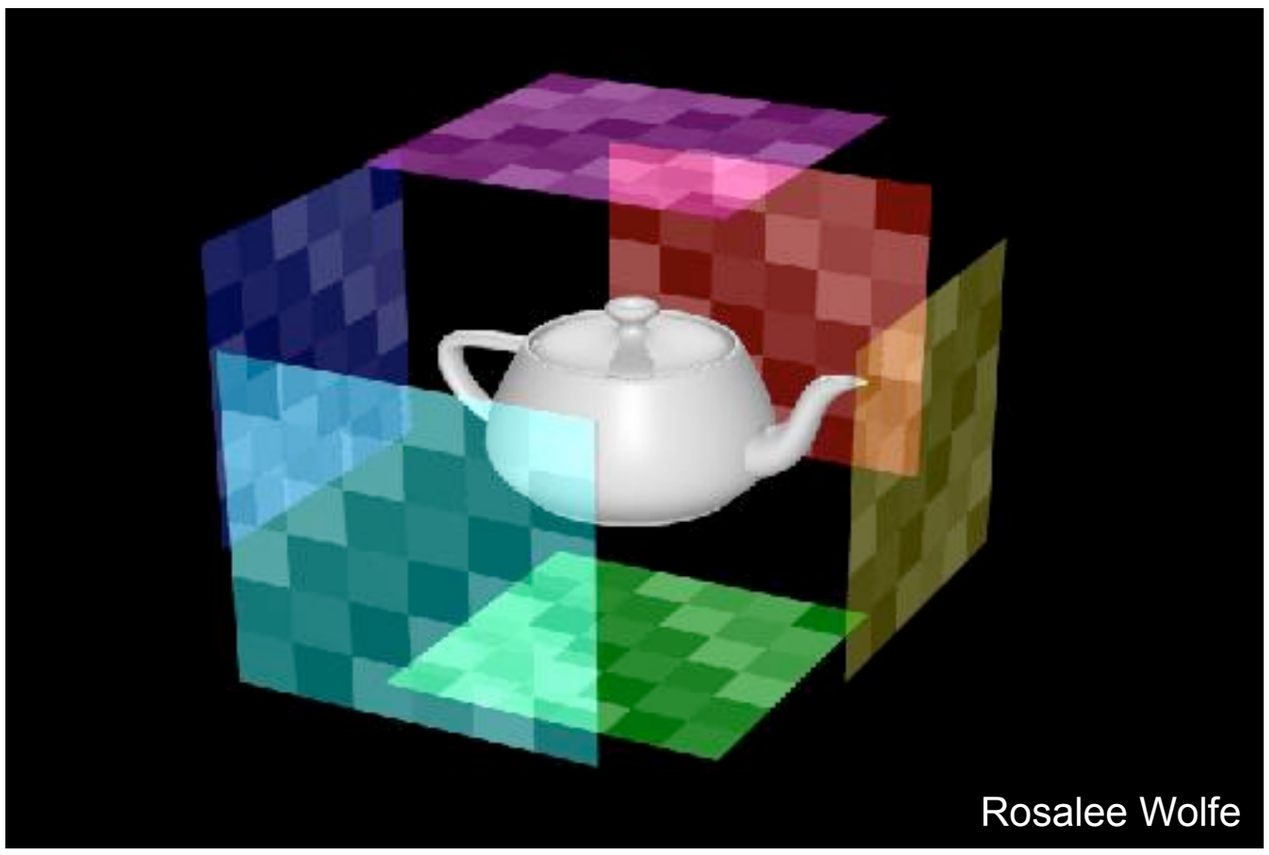


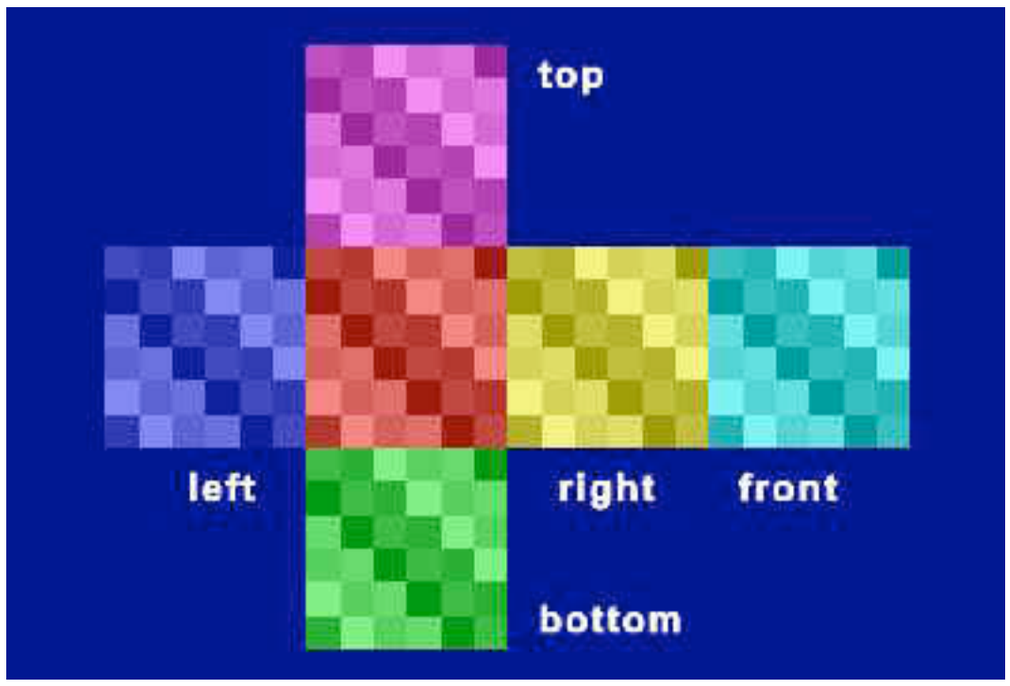


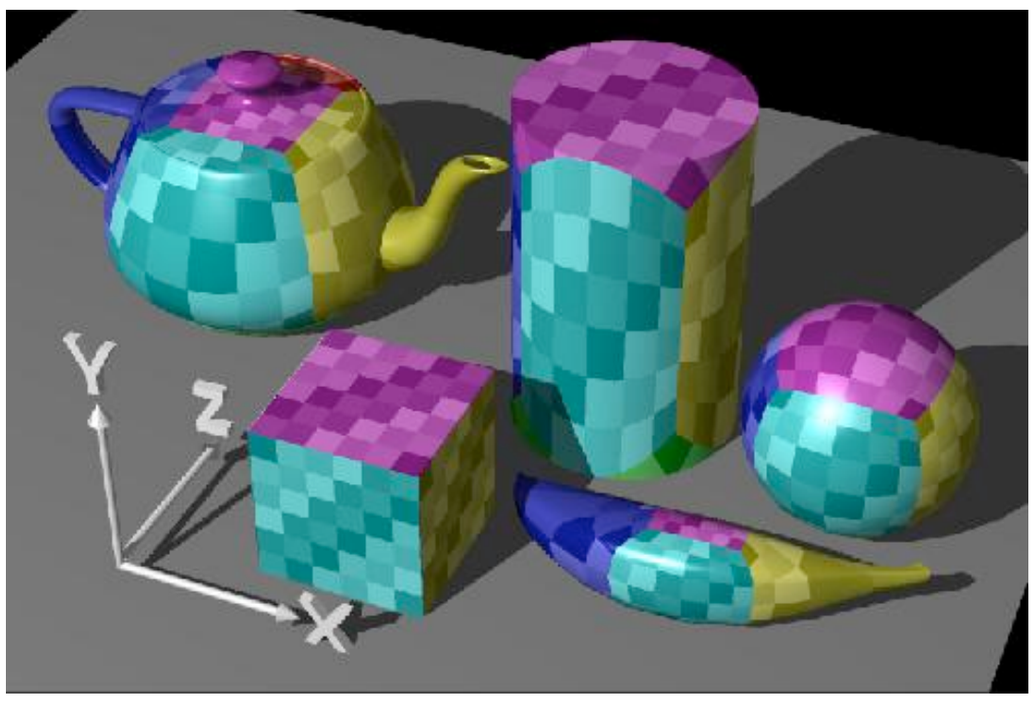


spherical map stretches squares at equation and squeezes squares at poles

Box Mapping

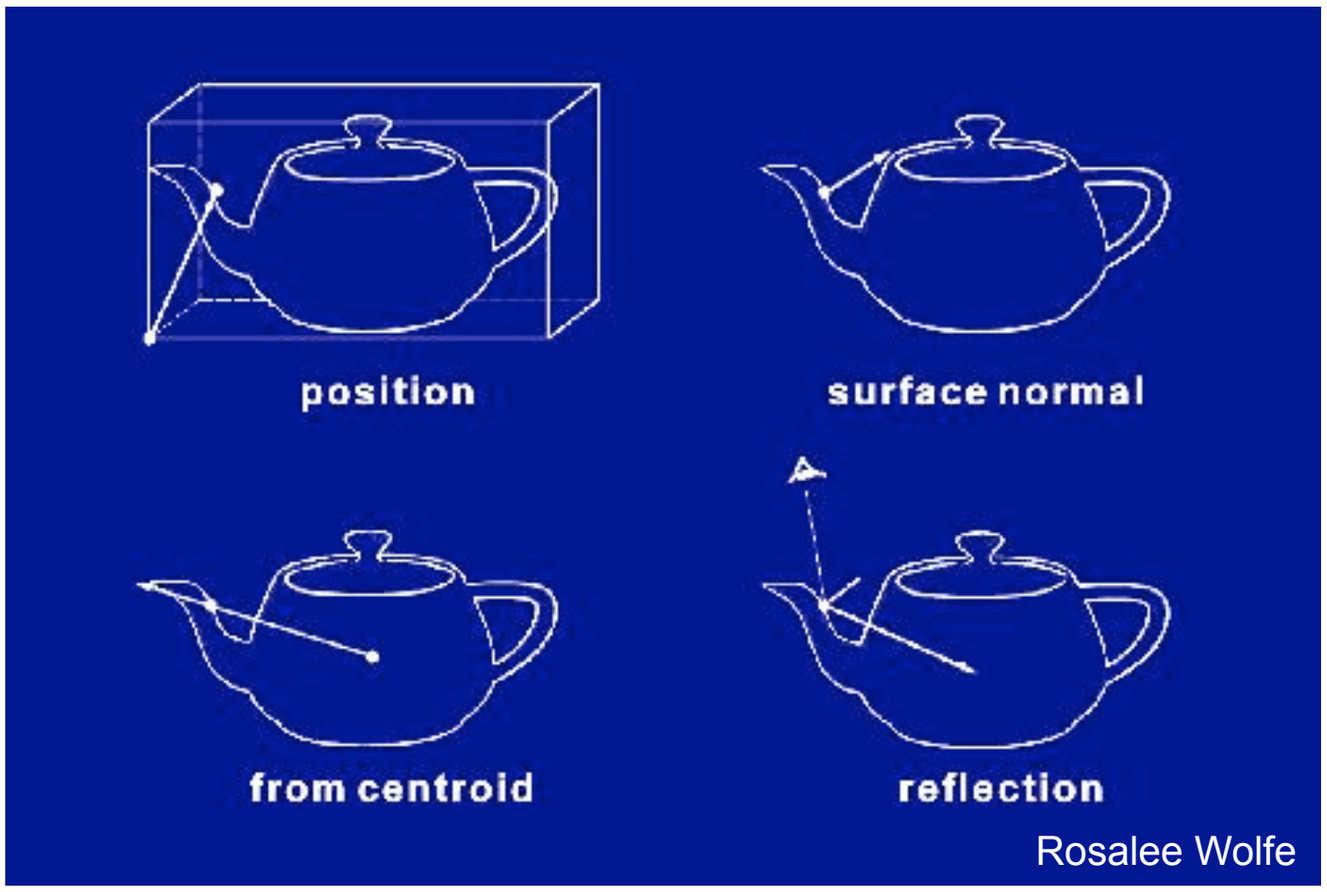






- similar to planar mapping
- planar projection -- choose which plane to project onto

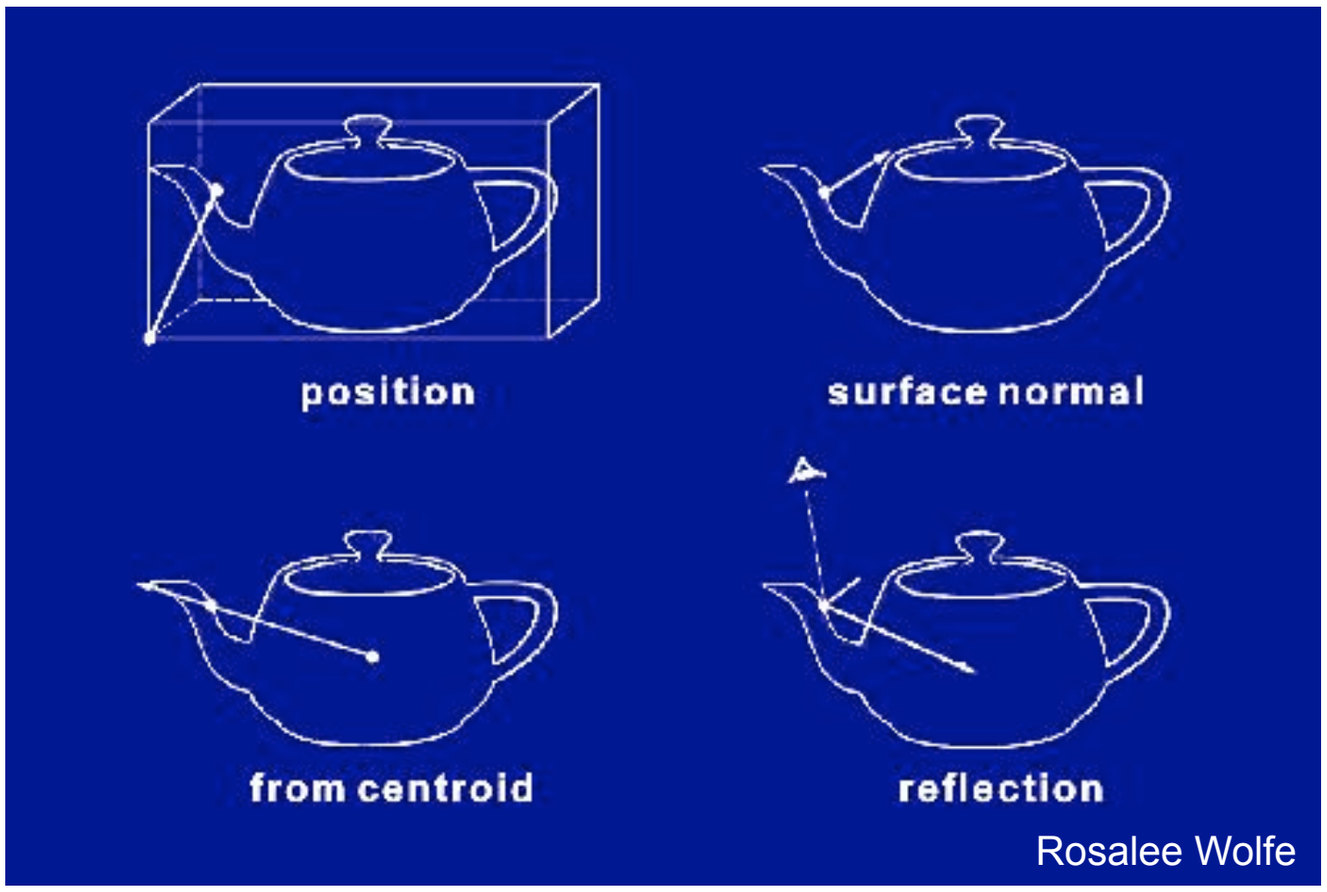
How do we map between intermediate and actual objects?

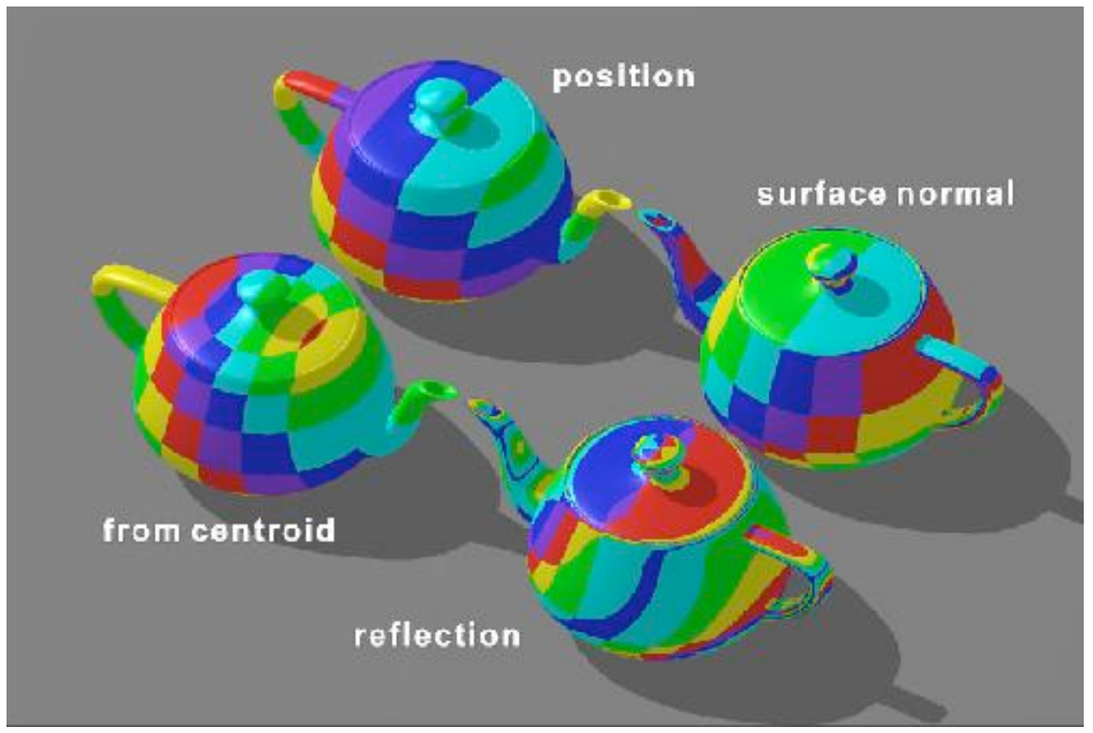


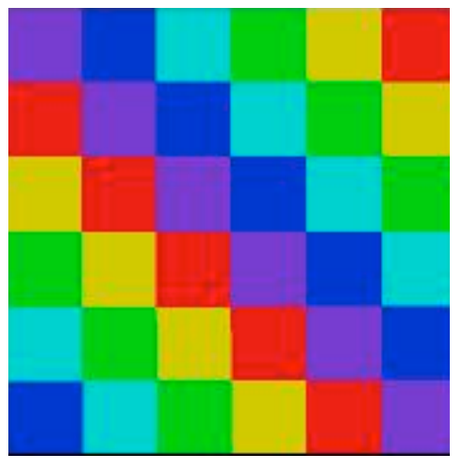
We associated (x,y,z) on the intermediate object with the texture (u,v). But which point on the actual object is this?
We choose both the **intermediate shape** and the **mapping from the actual shape to the intermediate shape**

1. a point on the object relative to its bounding box
2. see where surface normal intersects intermediate surface
3. shoot ray from centroid through surface point to intermediate surface
4. use the reflection vector (depends on the viewer position and normal)

How do we map between intermediate and actual objects?



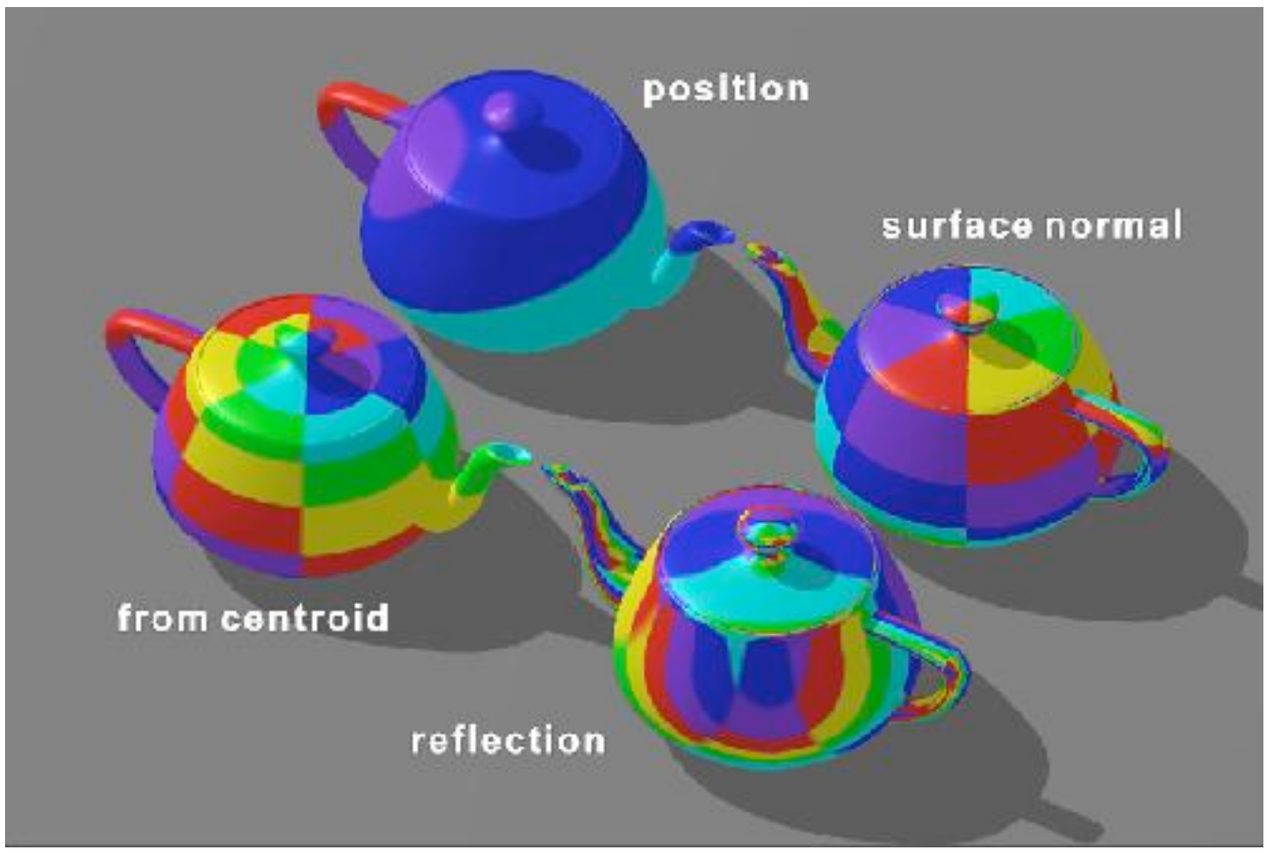




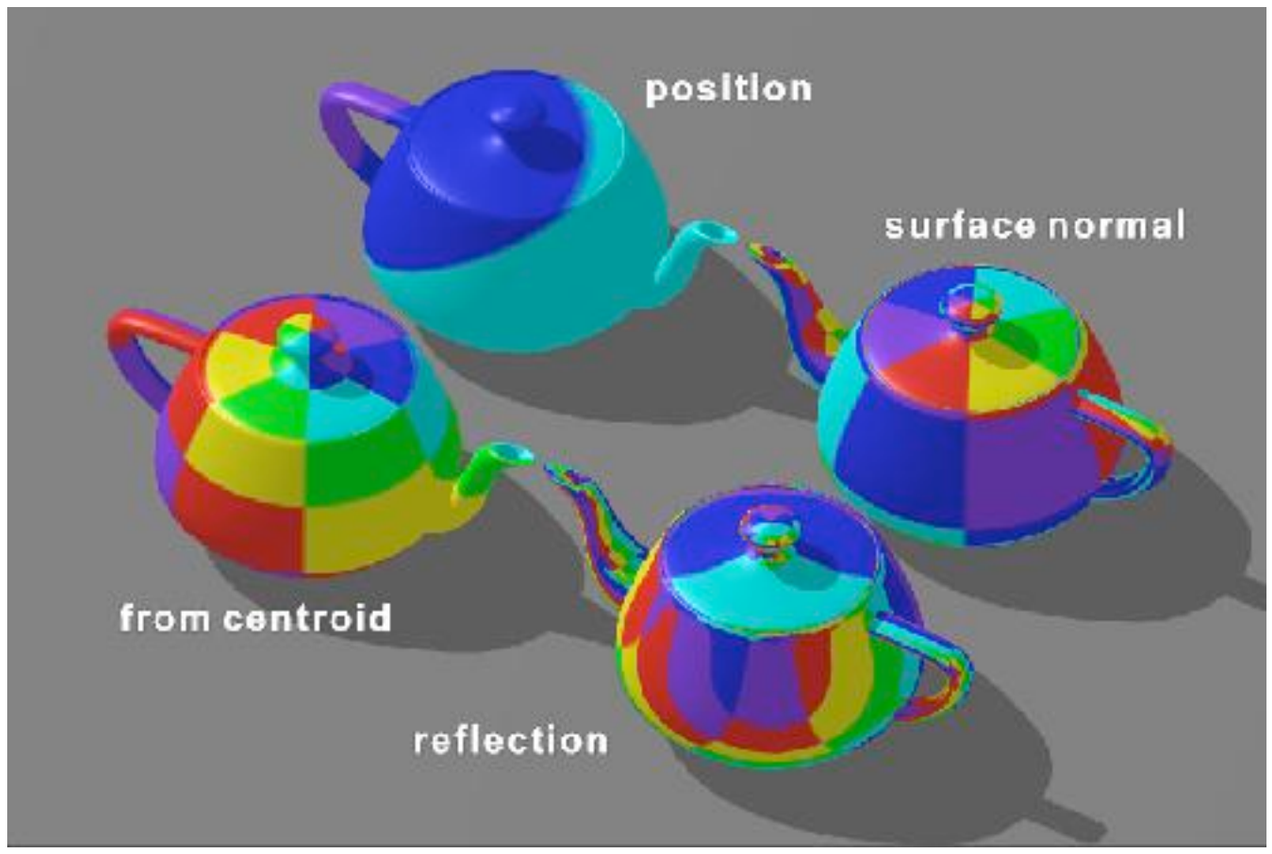
What intermediate shape was used here?

Can you tell what intermediate shape was used?

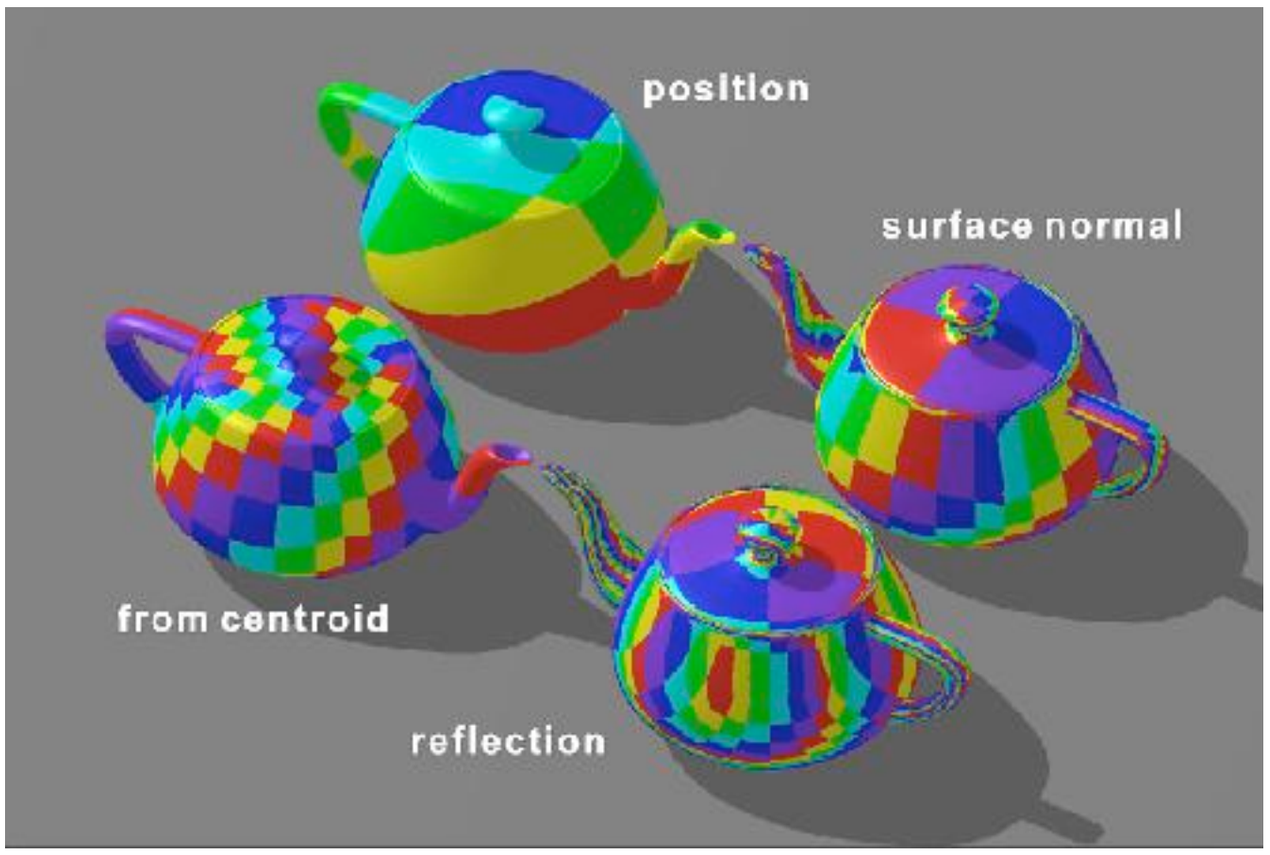
Comparison

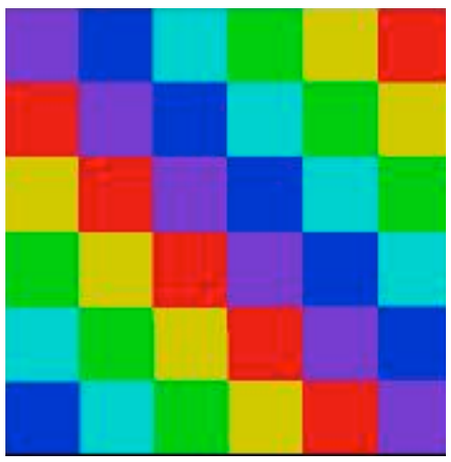


Cylindrical



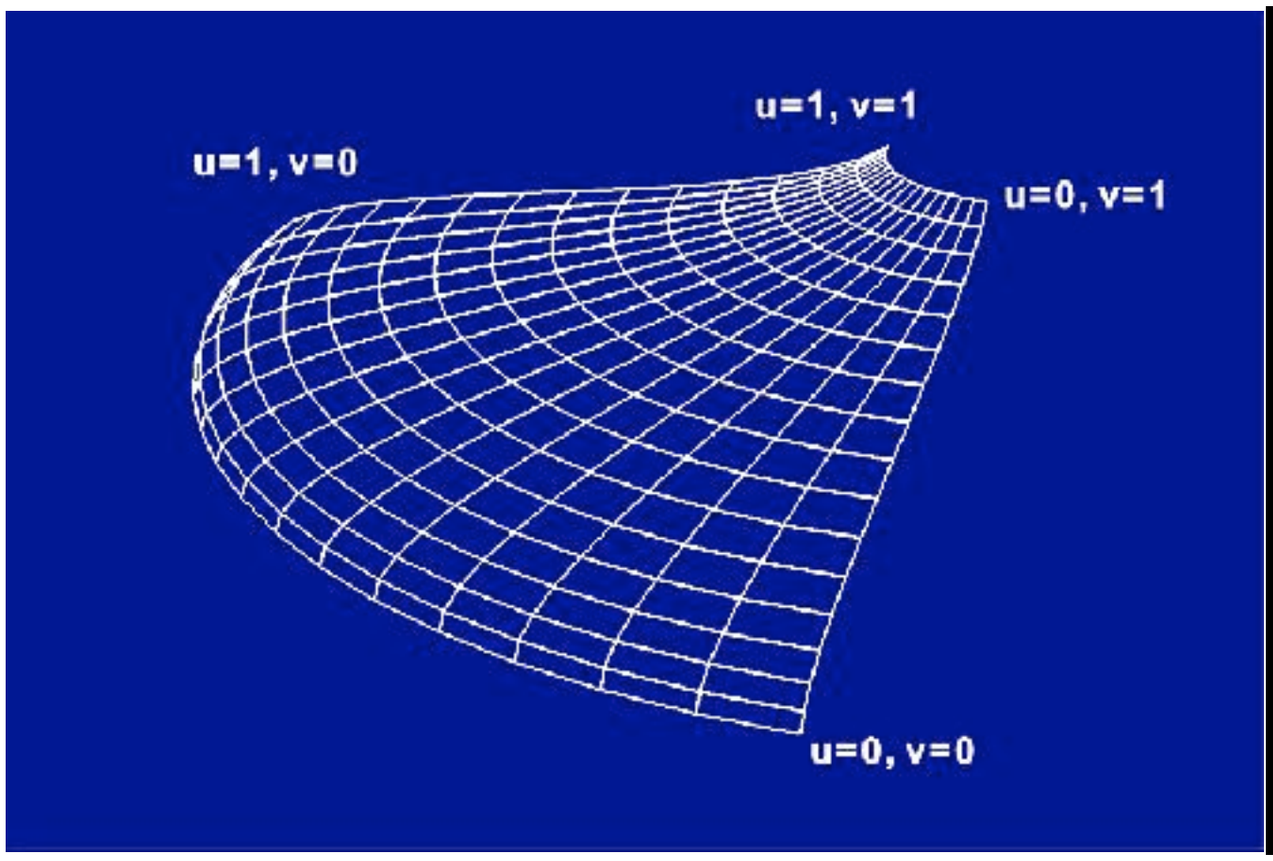
Spherical

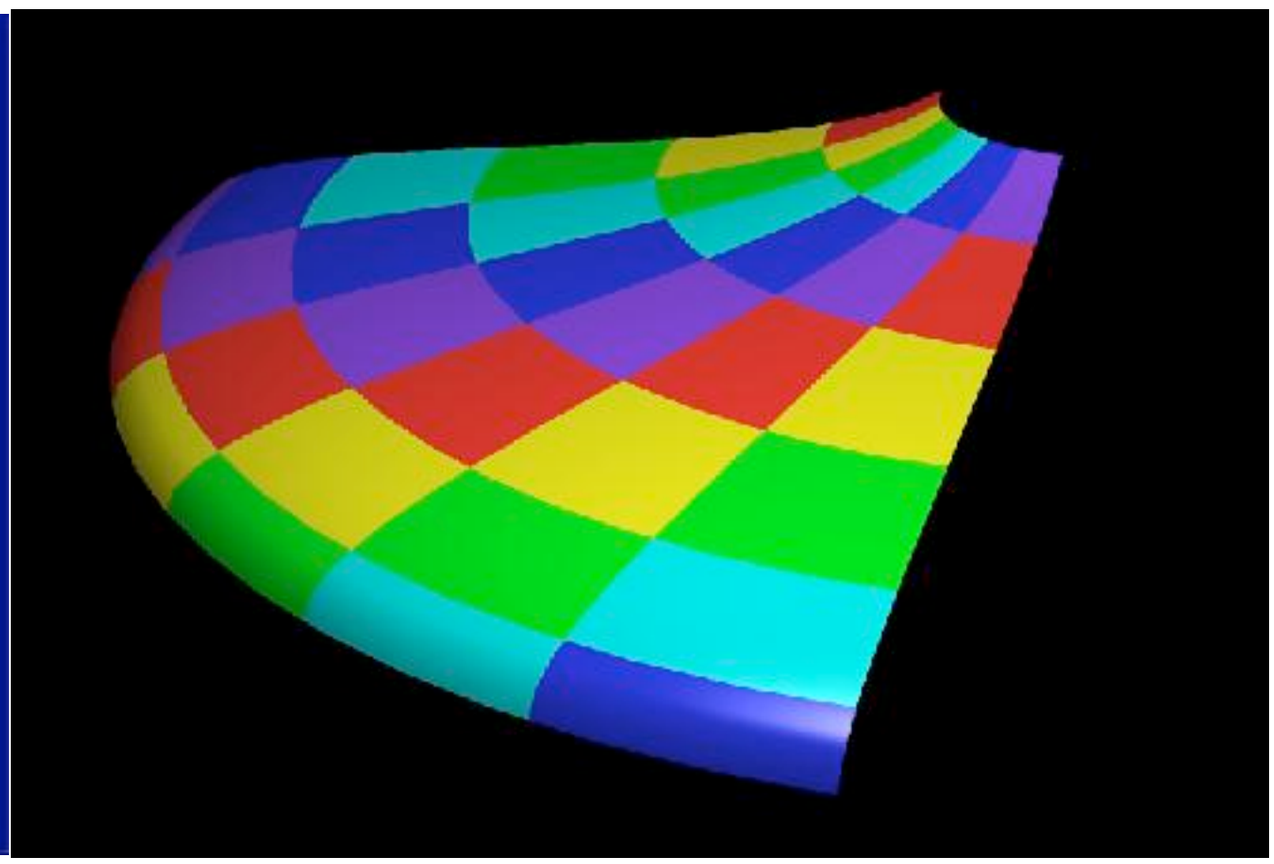




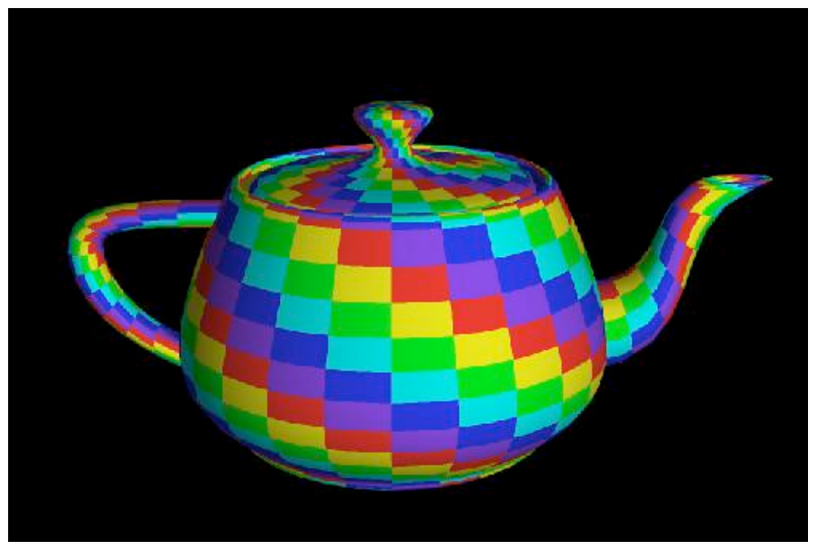
What intermediate shape was used here?

Parametric Surfaces





32 parametric patches



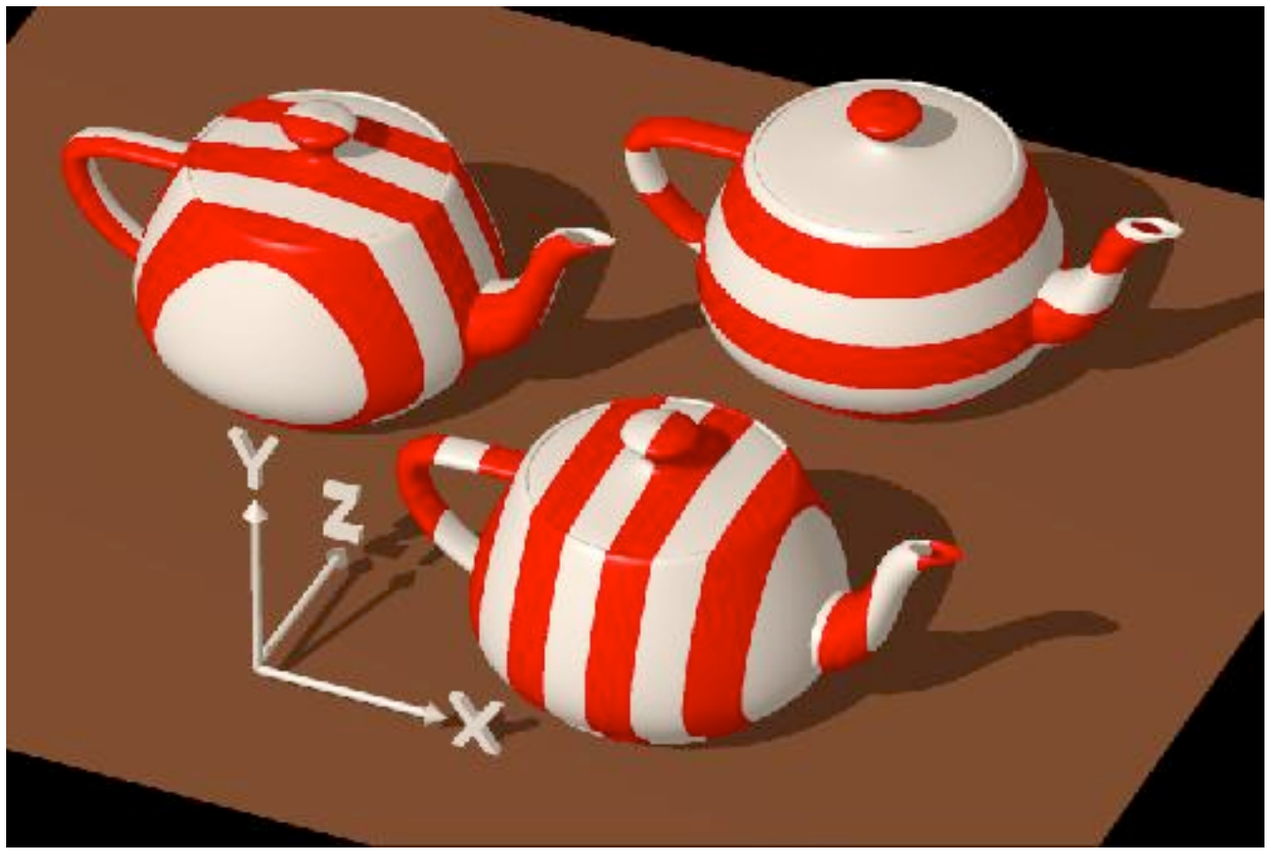
3D solid textures

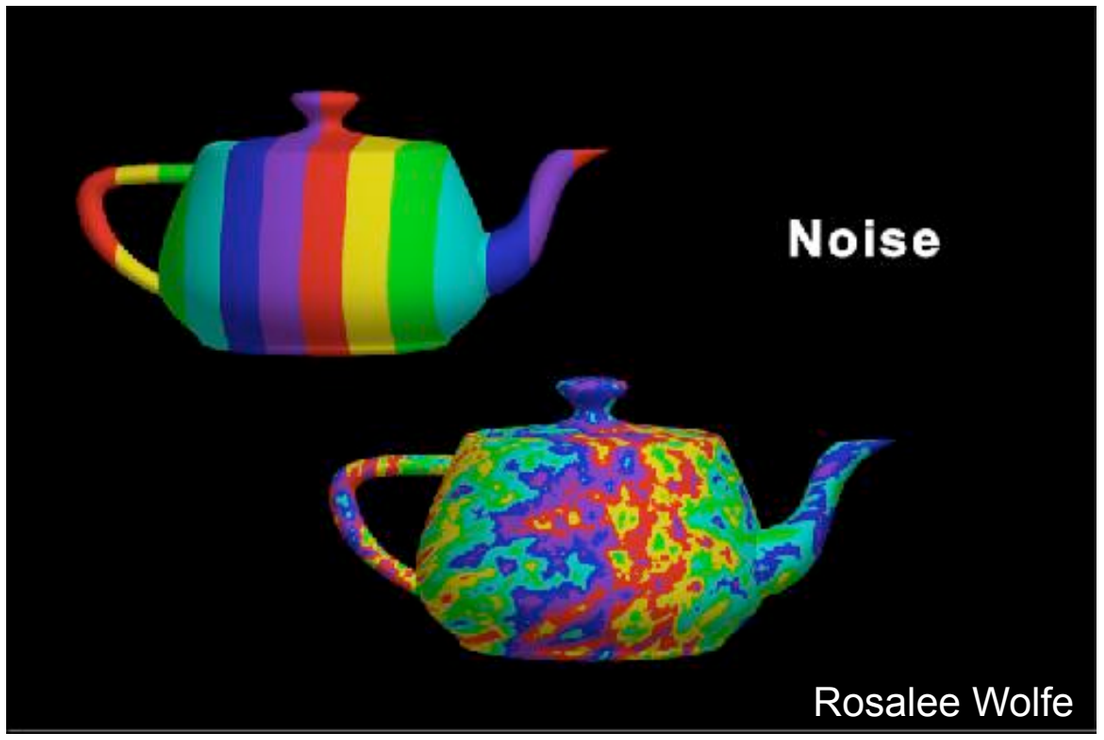
Dong et al., 2008

can map object (x,y,z) directly to texture (u,v,w)

no

Procedural textures

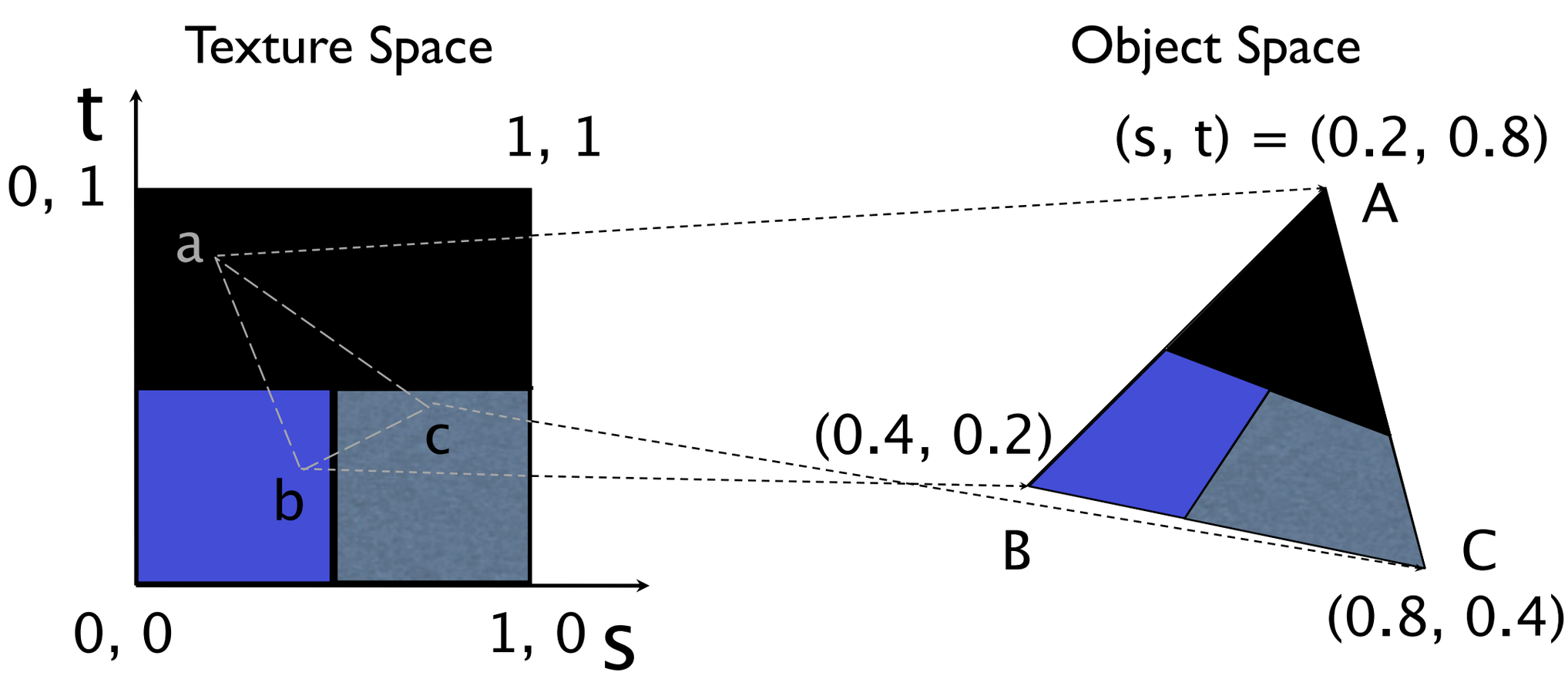




e.g., Perlin noise

Texturing triangles

- Based on parametric texture coordinates
- **`glTexCoord*()`** specified at each vertex

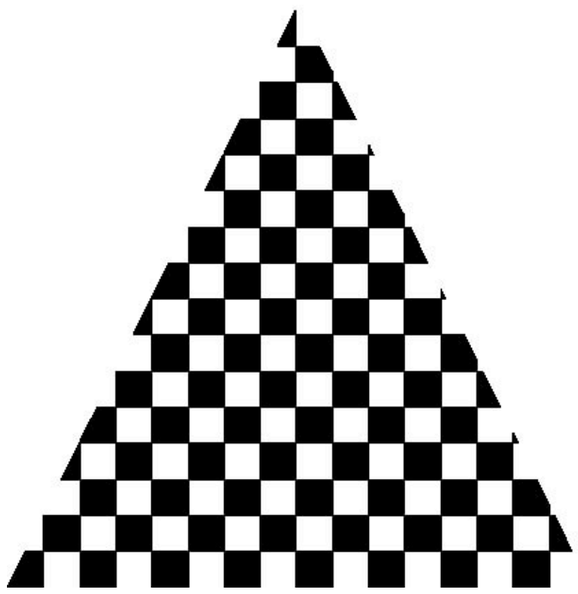


Interpolation

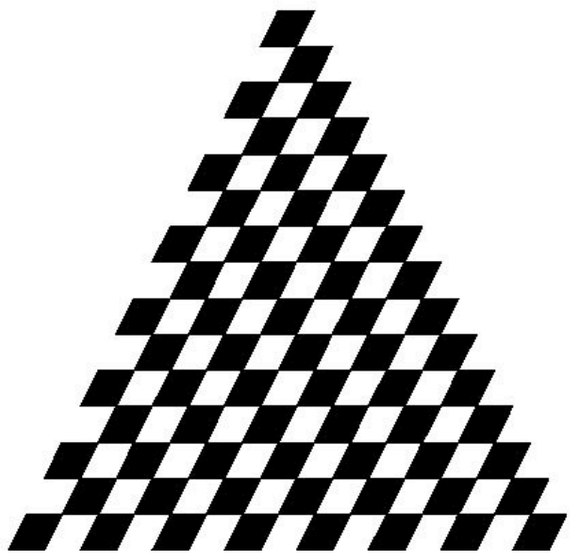
For polygonal mesh, want to assign (u,v) to vertices

OpenGL uses interpolation to find proper texels from specified texture coordinates

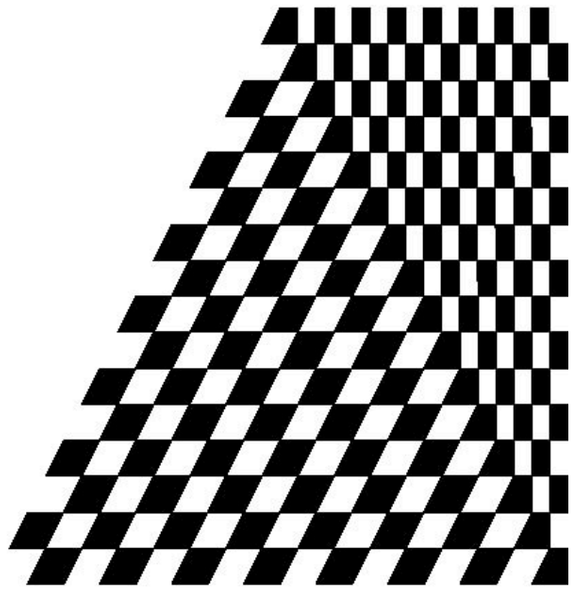
Can be distortions



good selection
of tex coordinates

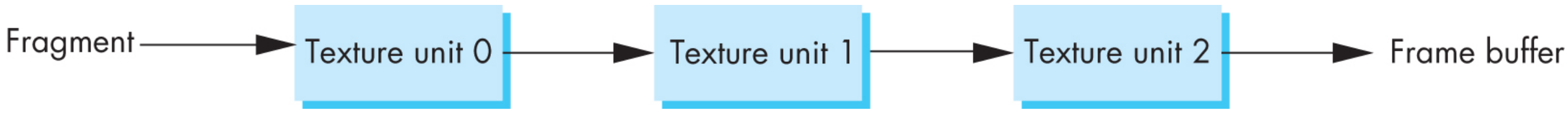


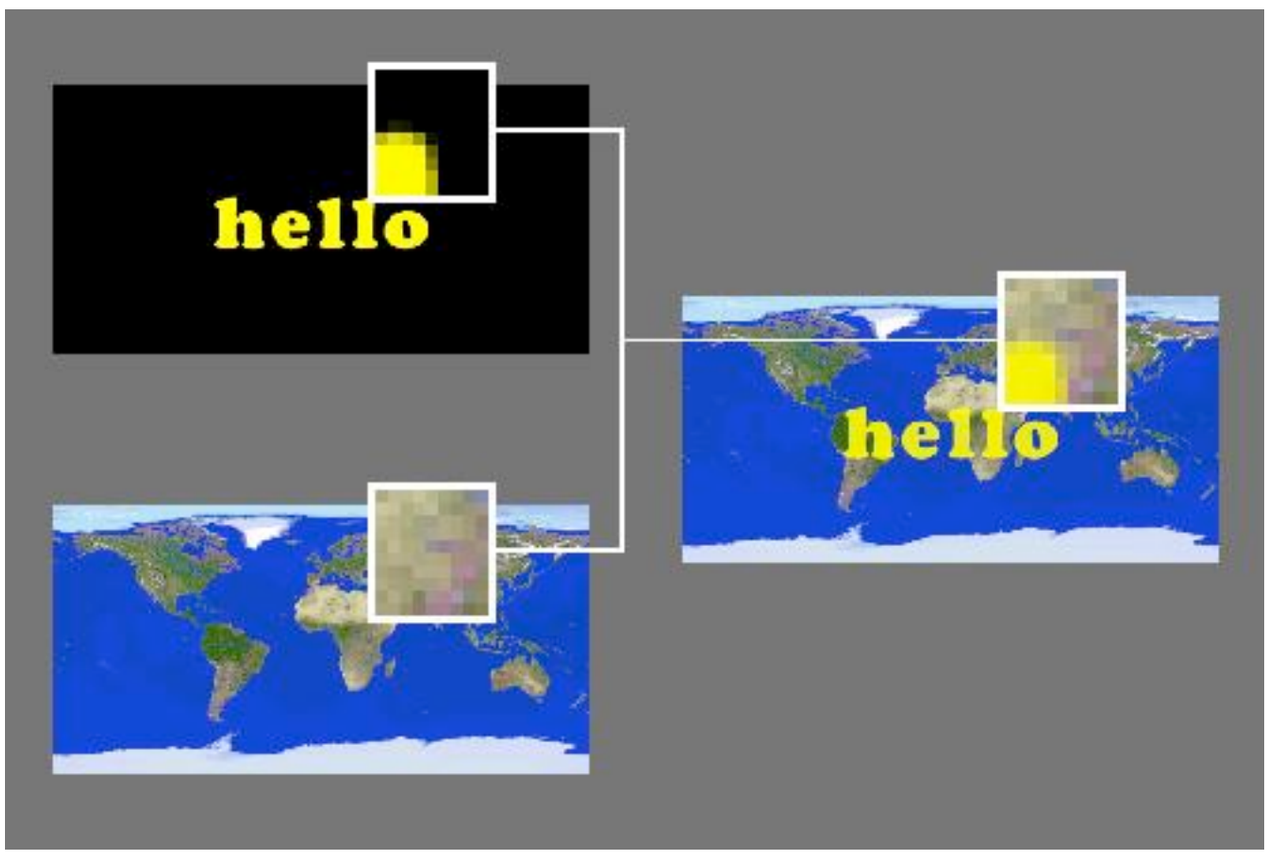
poor selection
of tex coordinates



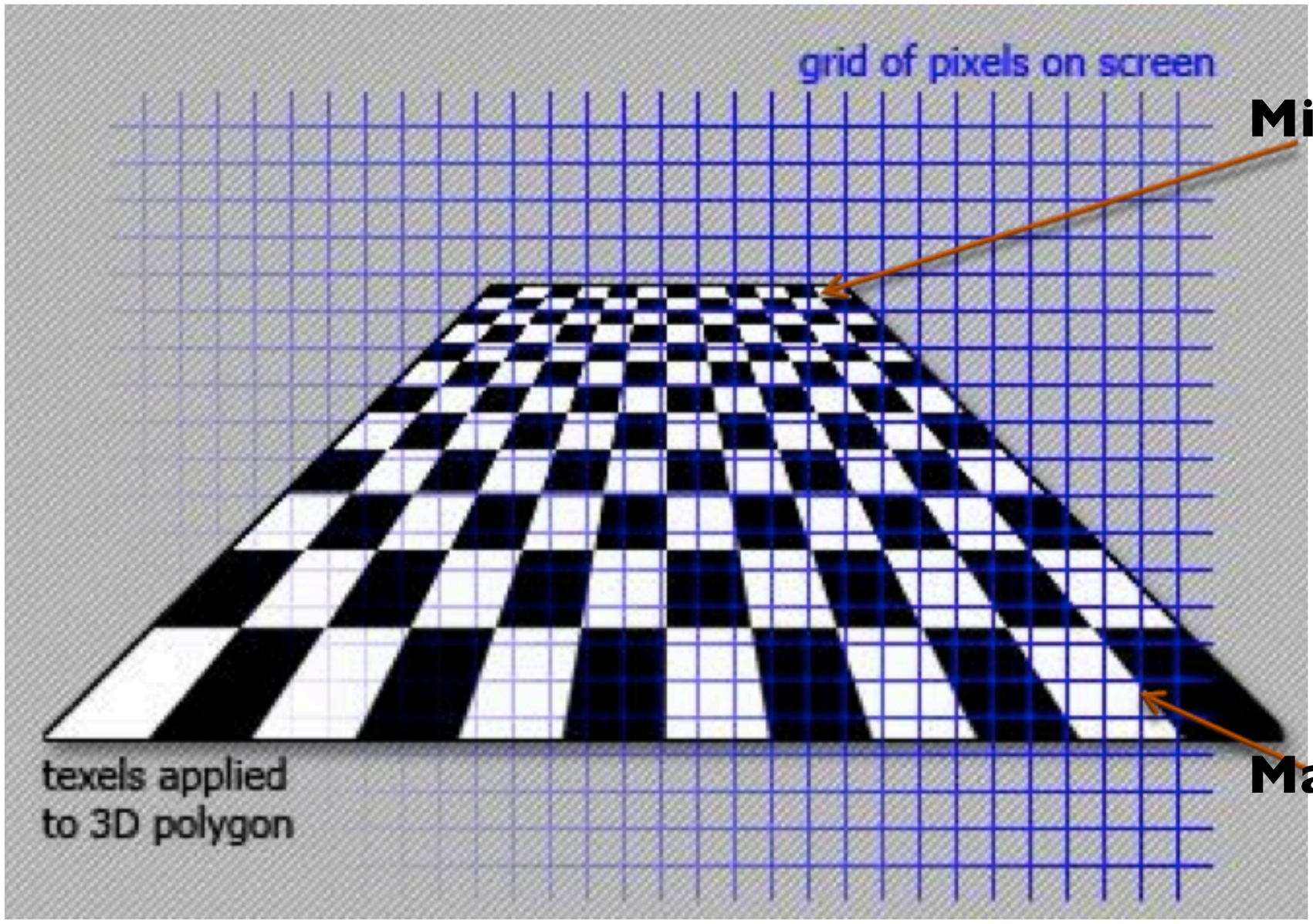
texture stretched
over trapezoid
showing effects of
bilinear interpolation

Multitexturing





Magnification and Minification



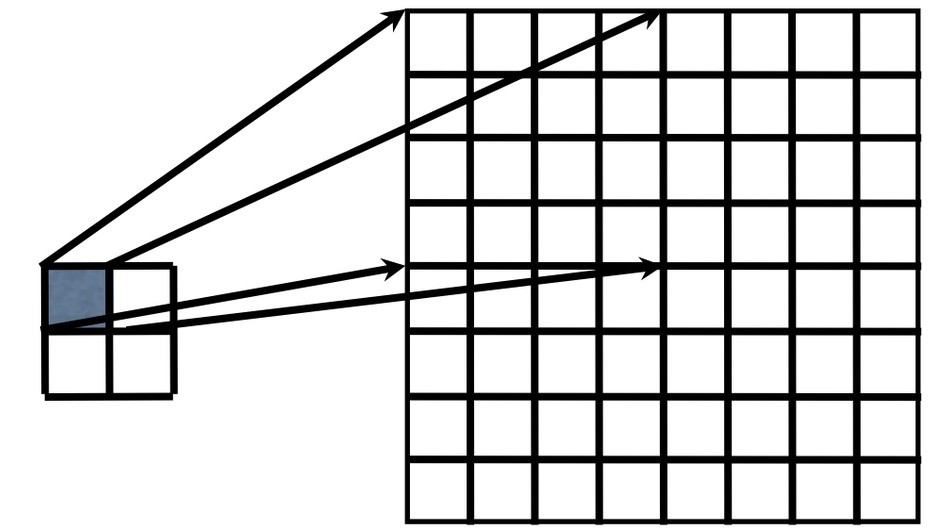
Minification

Magnification

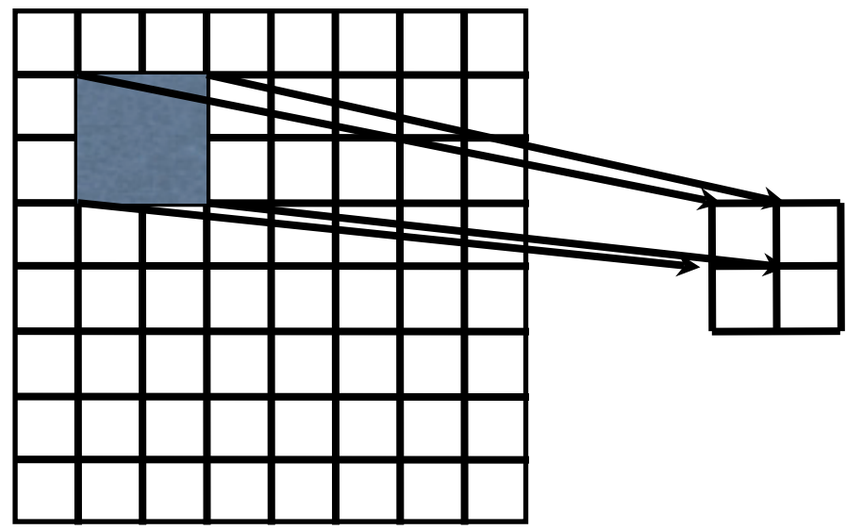
Magnification and Minification

More than one texel can cover a pixel (*minification*) or more than one pixel can cover a texel (*magnification*)

Can use point sampling (nearest texel) or linear filtering (2 x 2 filter) to obtain texture values



Magnification

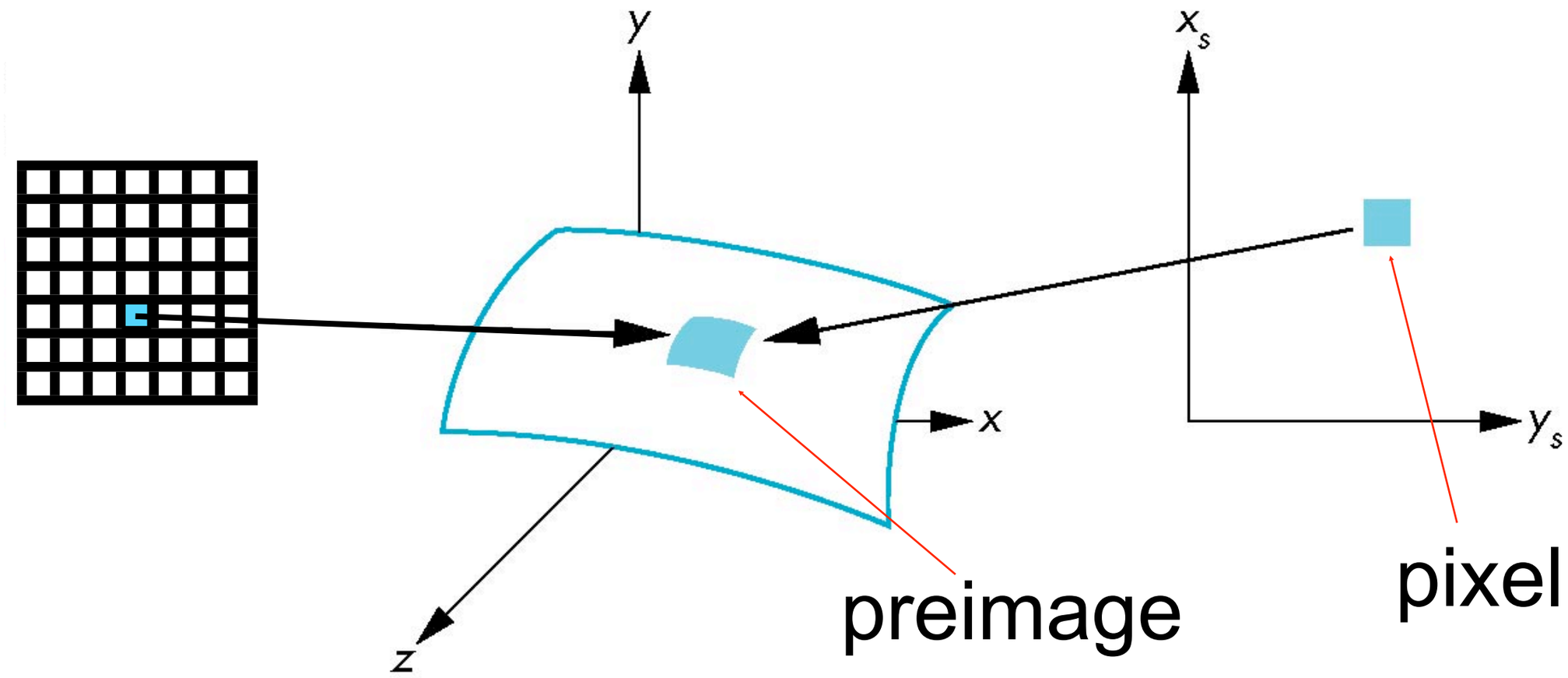


Minification

Texture Sampling

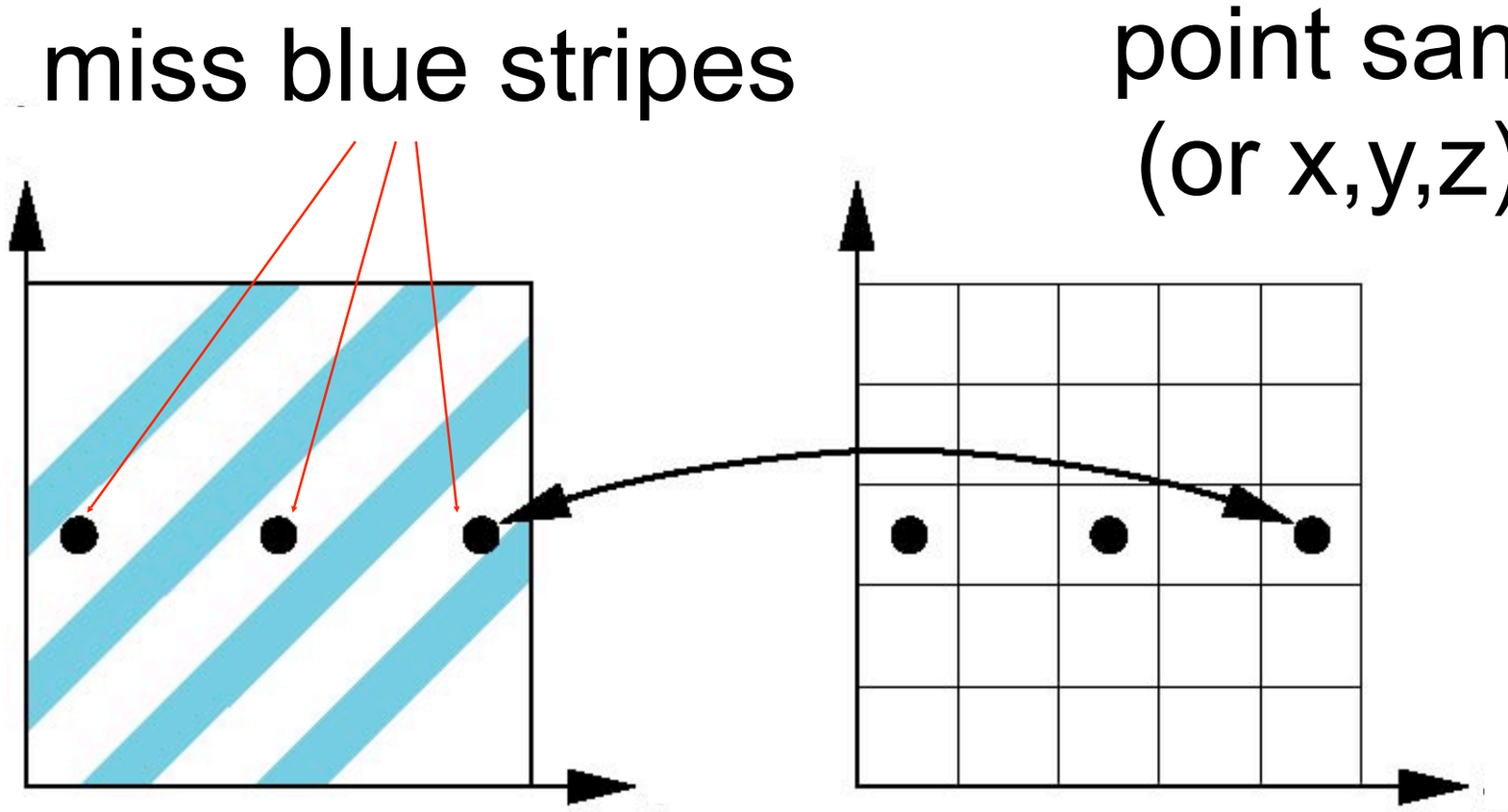
Point Sampling

Map back to texture image and use the **nearest texel**

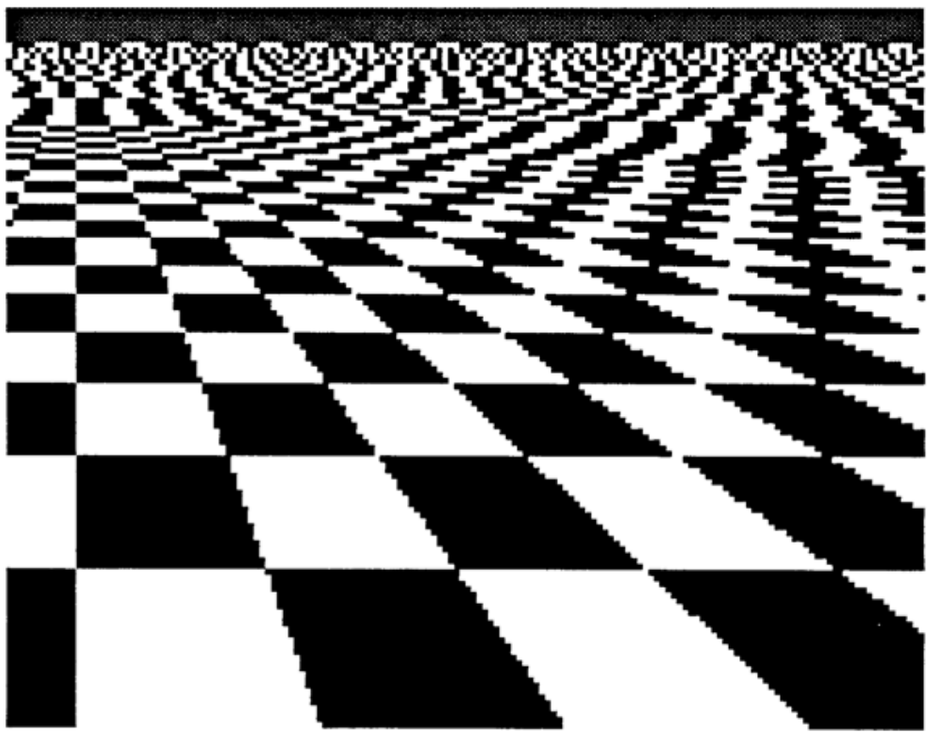


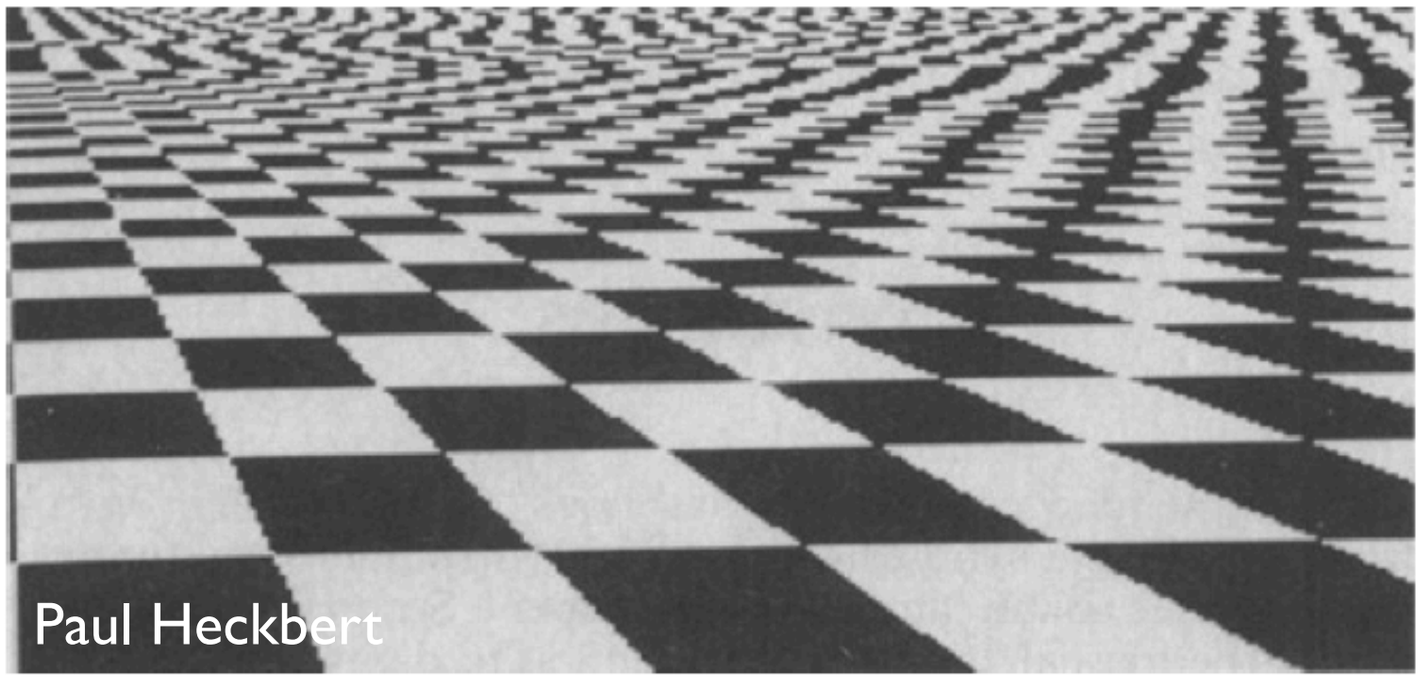
Aliasing

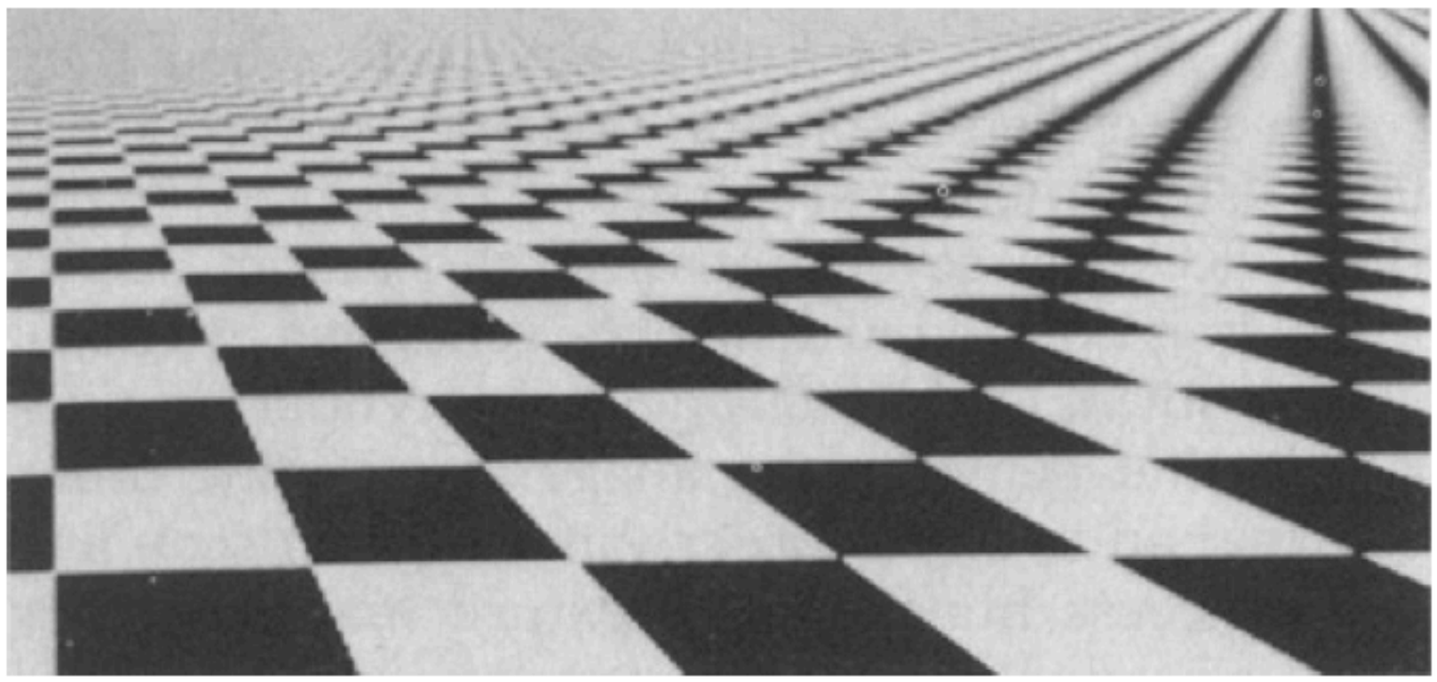
Point sampling of the texture can lead to aliasing artifacts



Aliasing artifacts



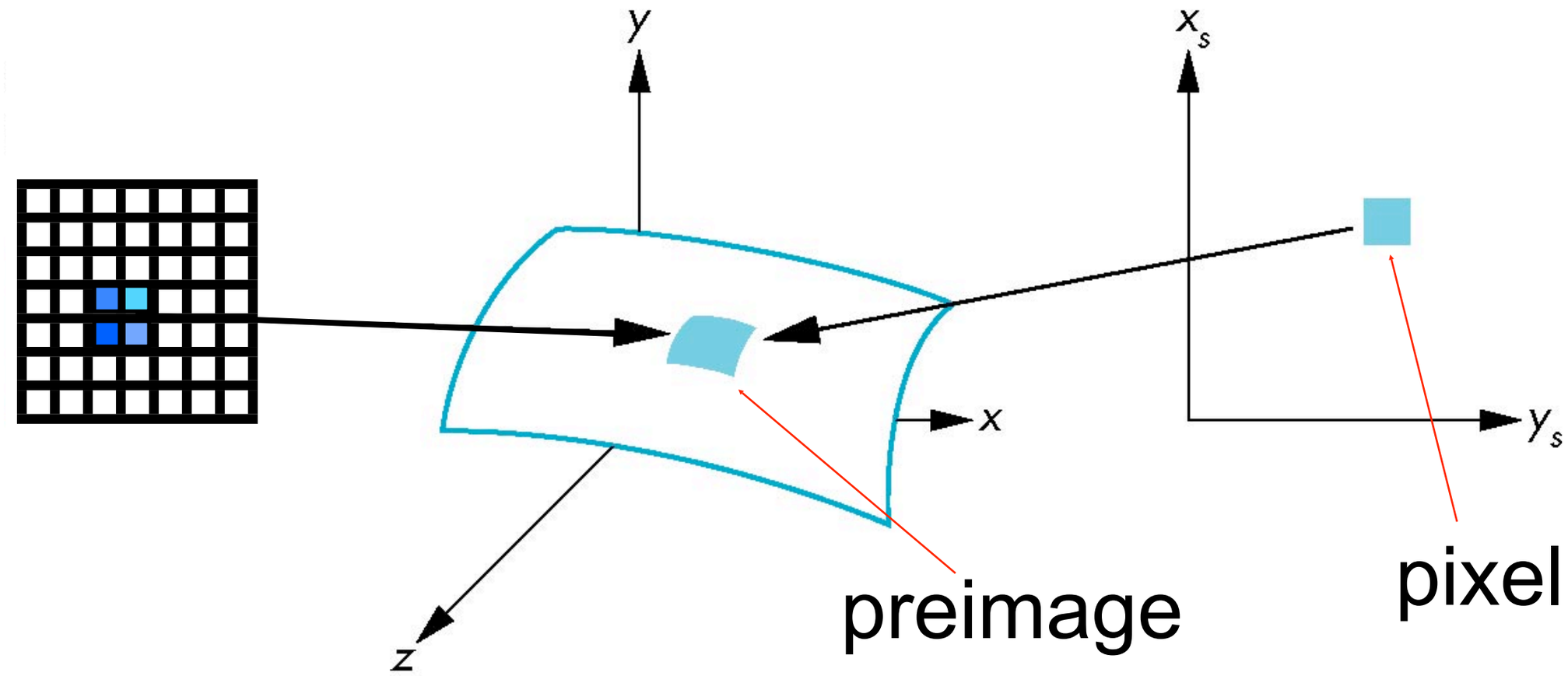




We apply **filtering** to reduce aliasing artifacts

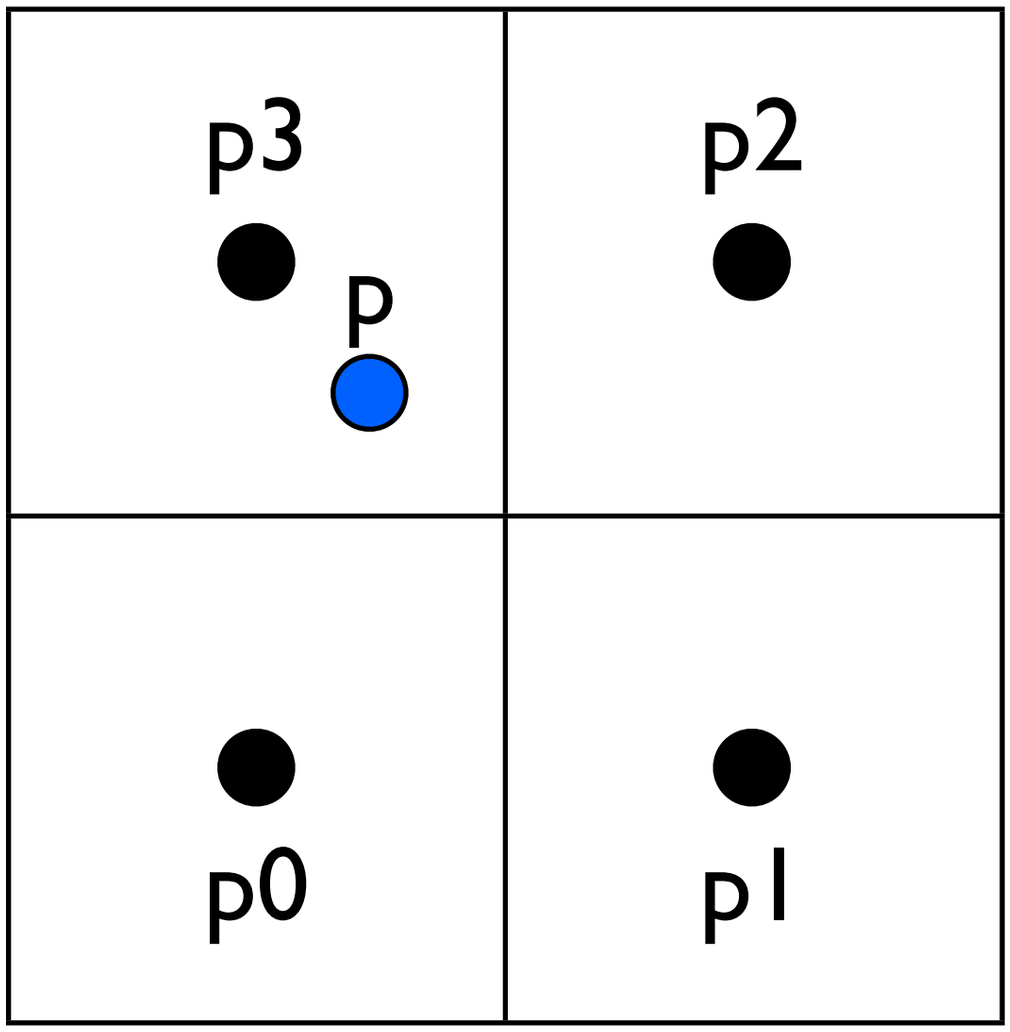
Area Averaging

A better but slower option is to use **area averaging**



Note that *preimage* of pixel is curved

Use bilinear filtering



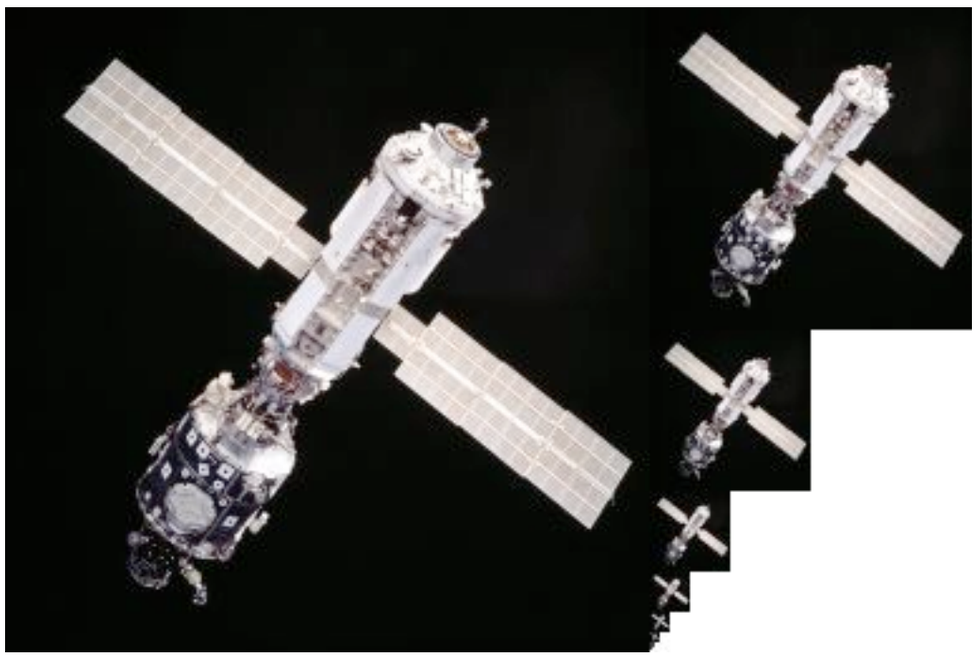
Wikipedia

nearest neighbor **bilinear** **bicubic**

mitigate magnification artifacts

smooths out the texture – no sharp boundaries

Mipmapping



Togikun, Wikimedia Commons

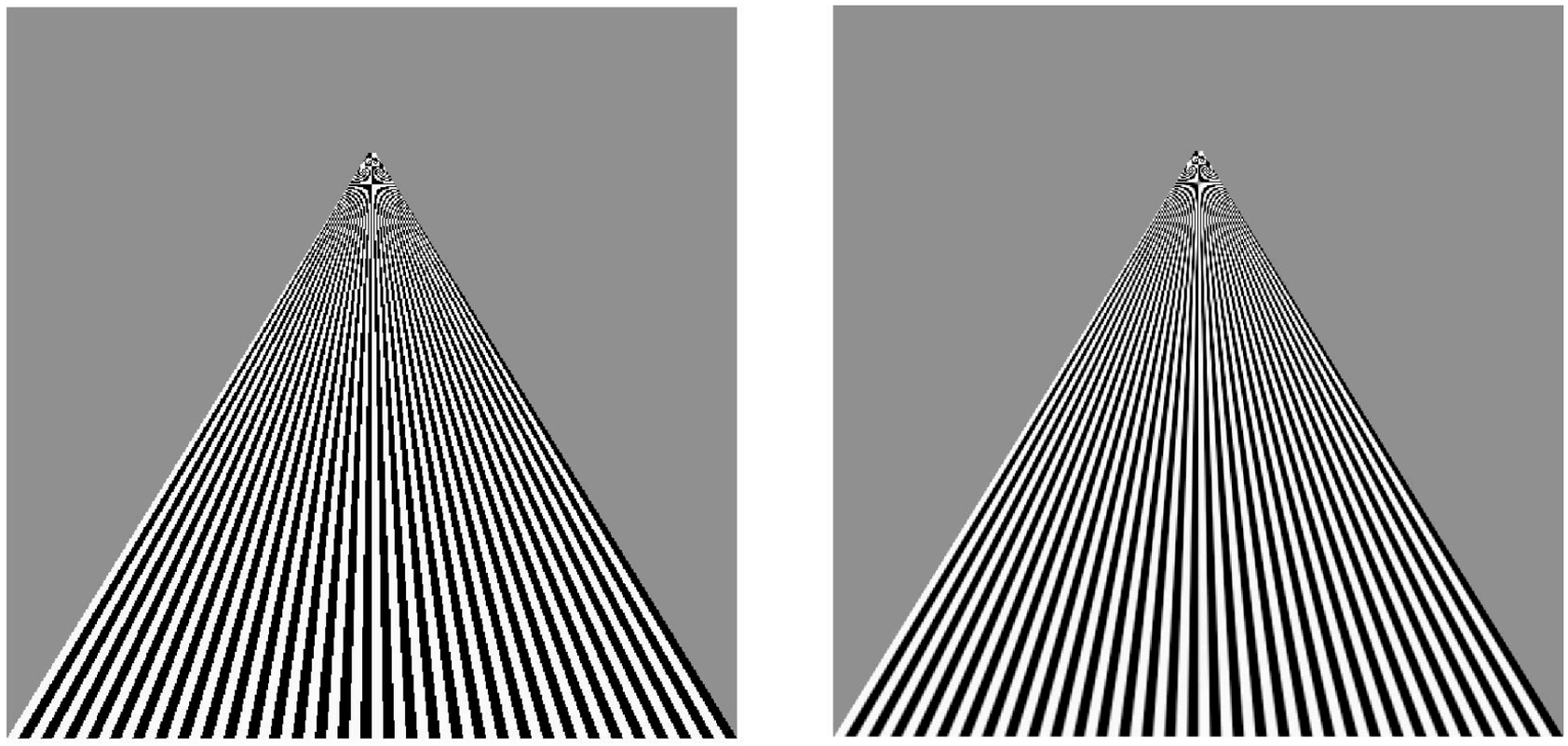
128×128, 64×64, 32×32, 16×16, 8×8, 4×4, 2×2, 1×1

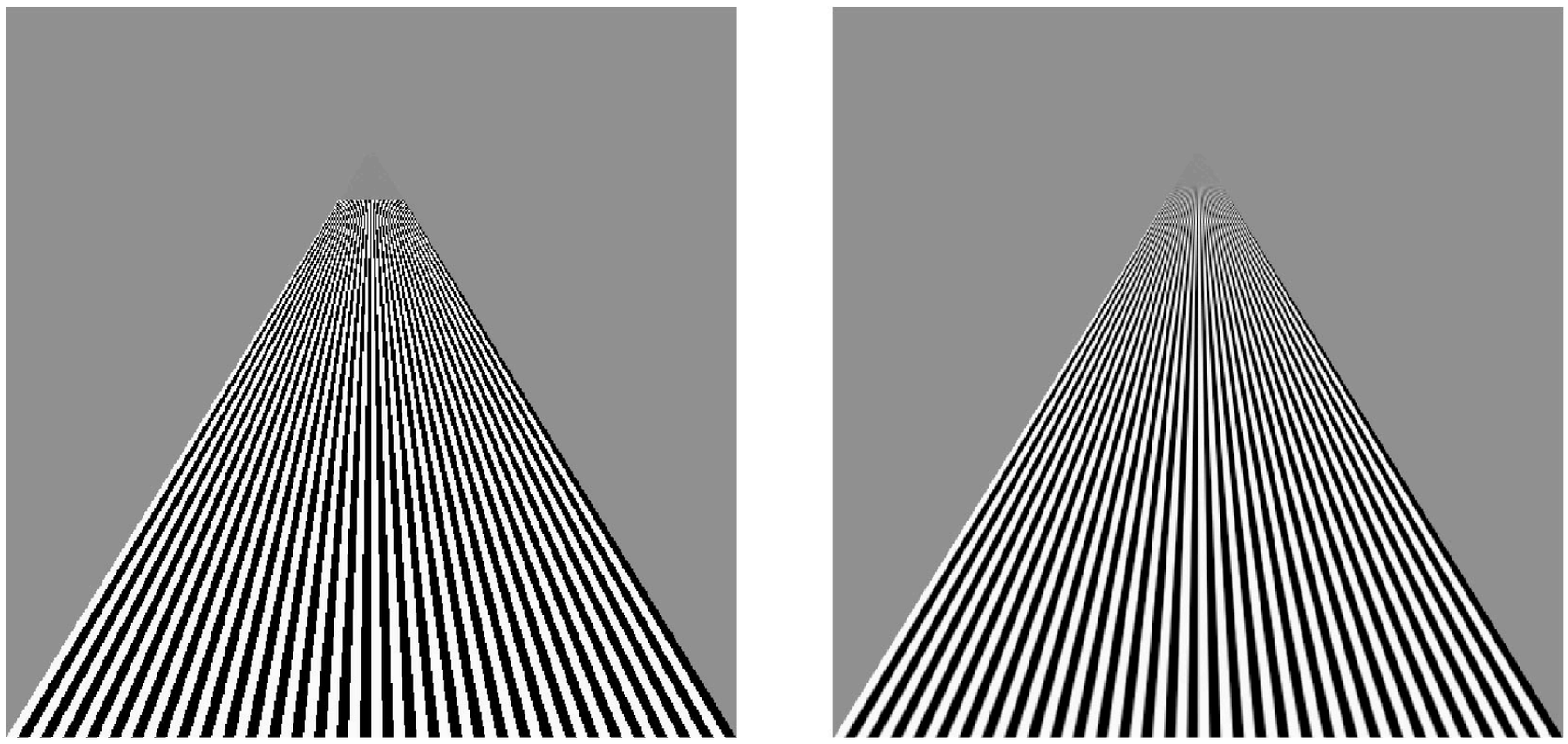
Reduce minification artifacts

Prefilter the texture to obtain reduced resolutions

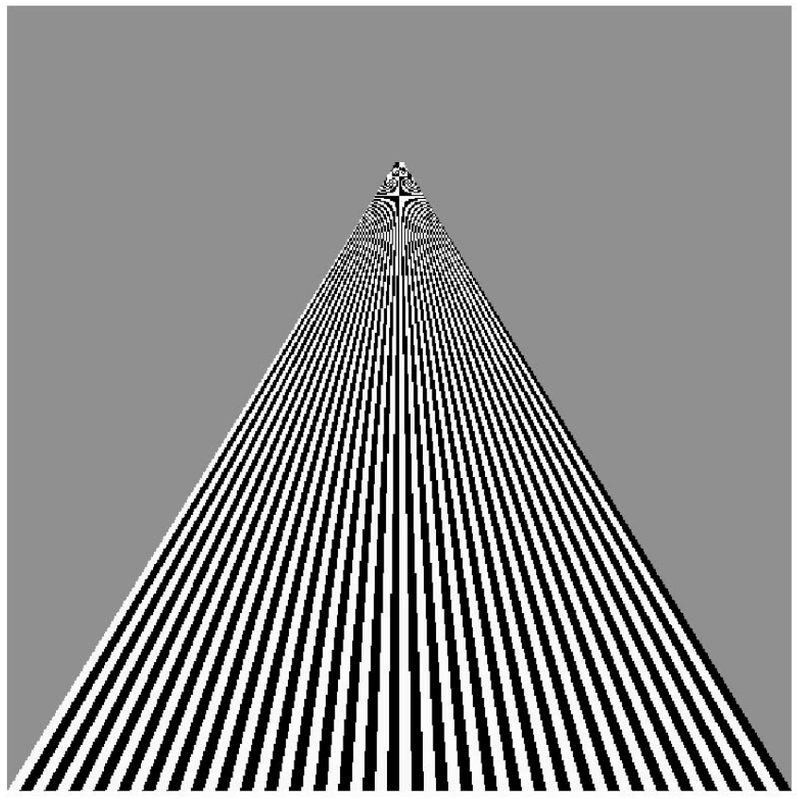
Requires 1/3 more space

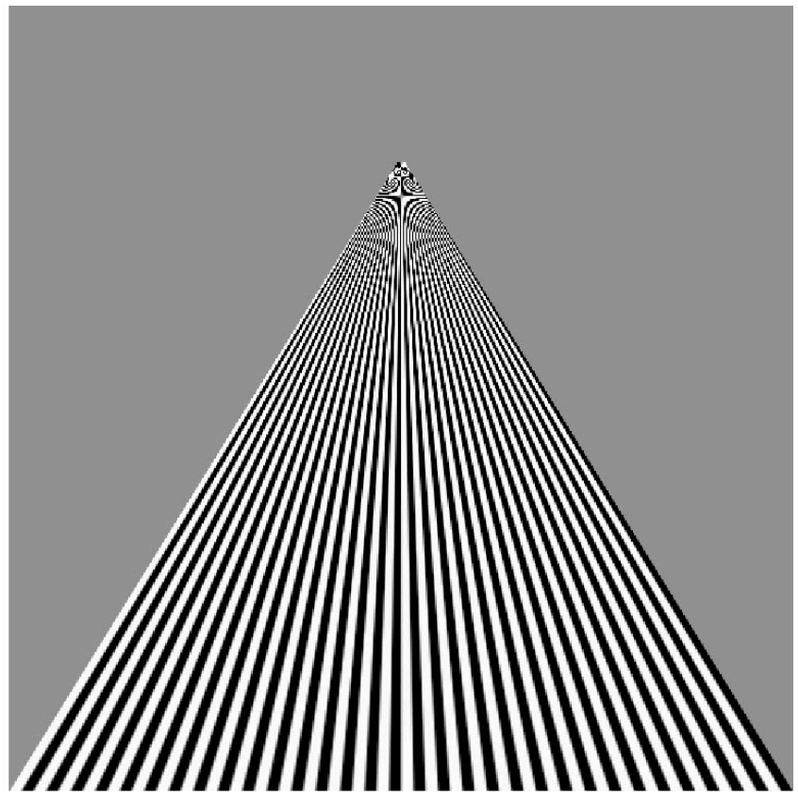
Get a texture hierarchy indexed by level





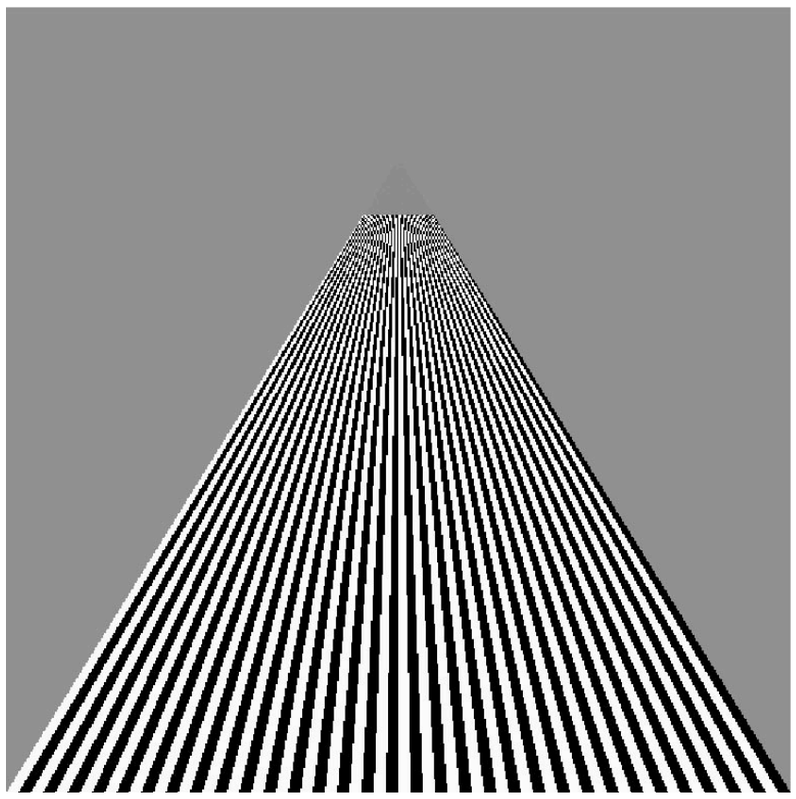
point sampling

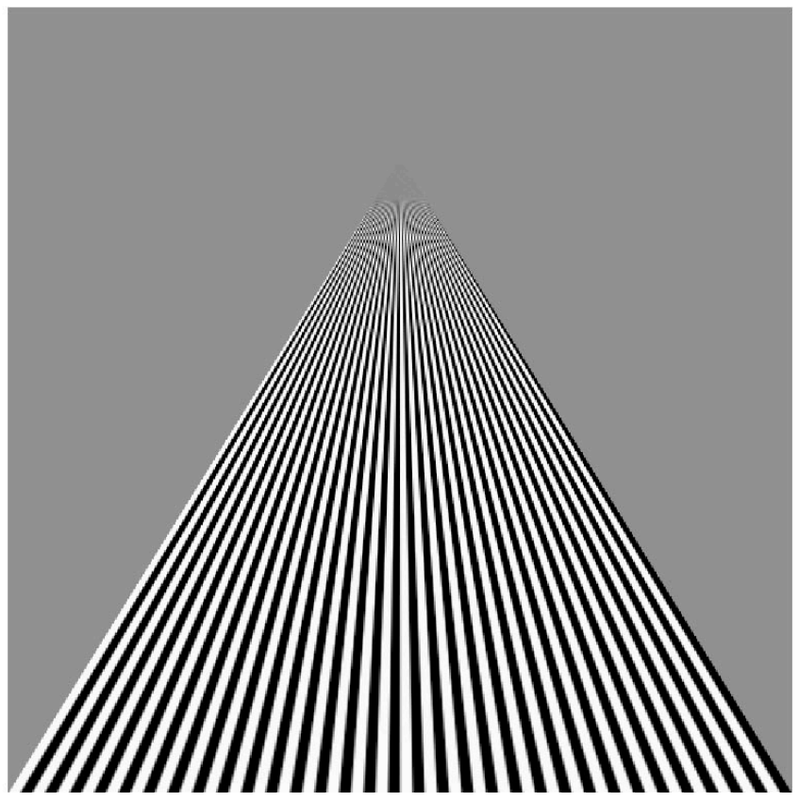




linear filtering

mipmapped point sampling





mipmapped linear filtering