CS130 : Computer Graphics

Lecture 9: Texture Mapping (cont.)

Tamar Shinar
Computer Science & Engineering
UC Riverside

The major issues in texture mapping...

- What should the actual mapping be?

[Rosalee Wolfe]

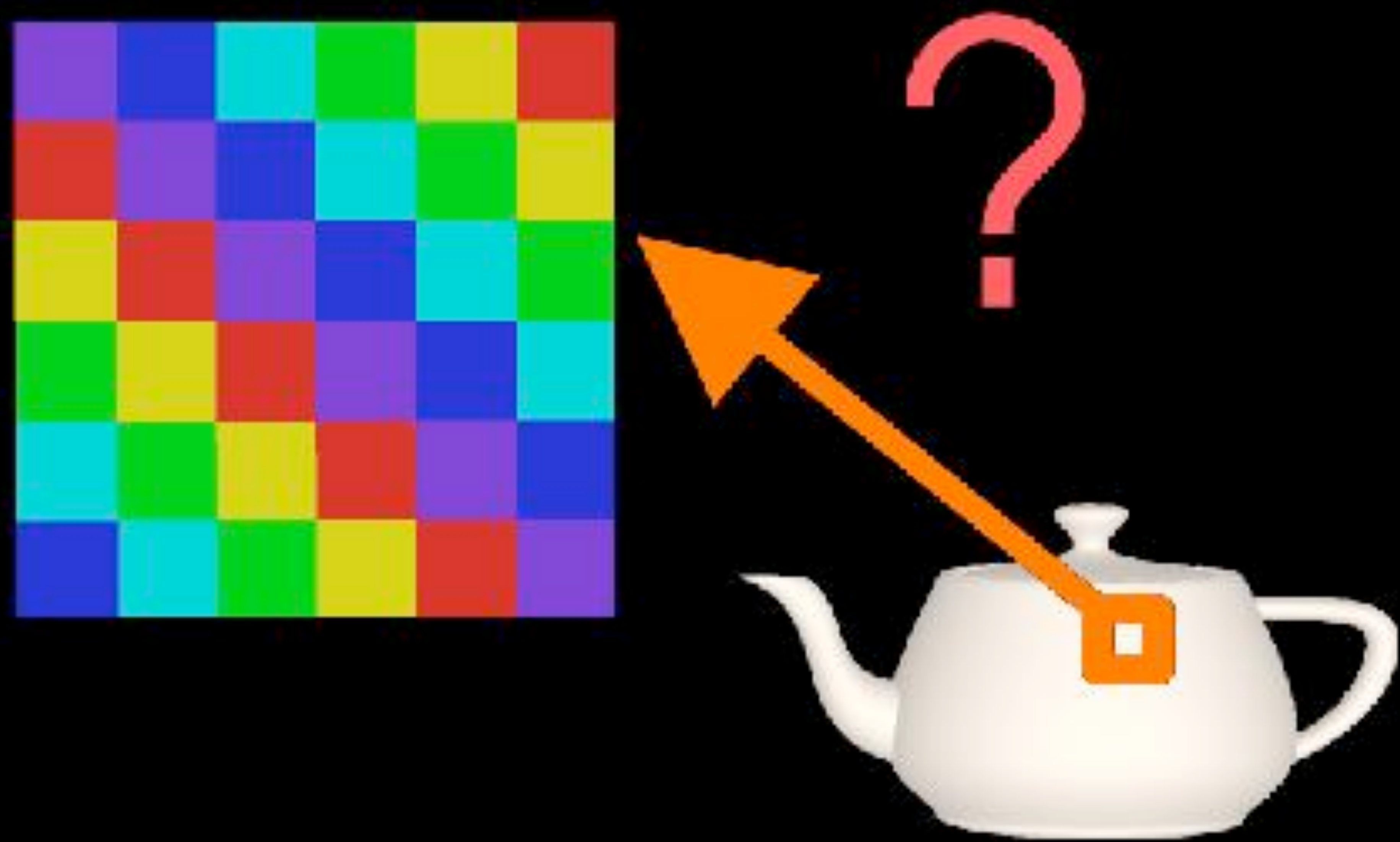
easy: rectangular surface

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

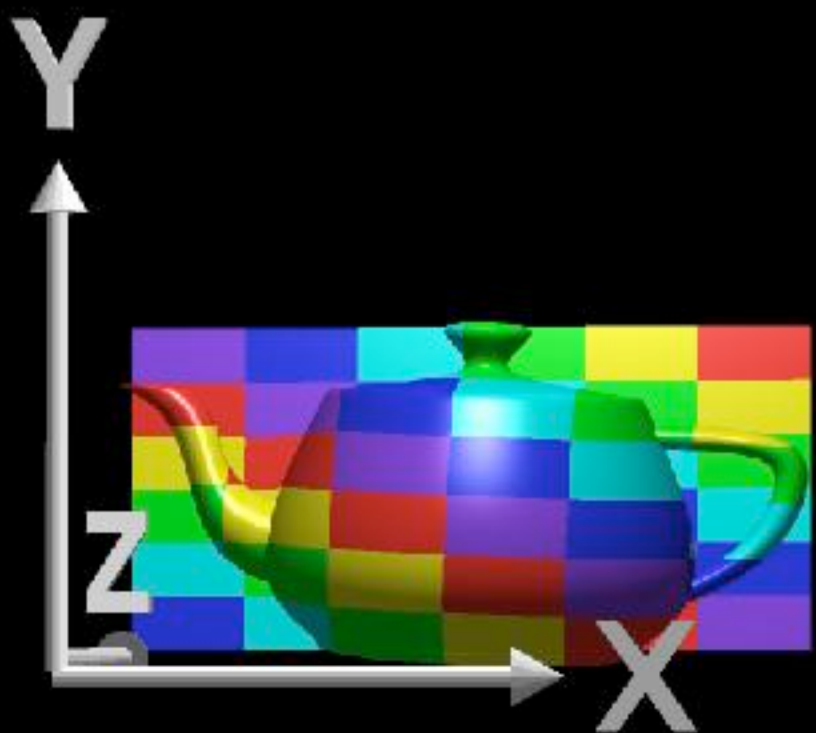
want a nice map that doesn't look distorted

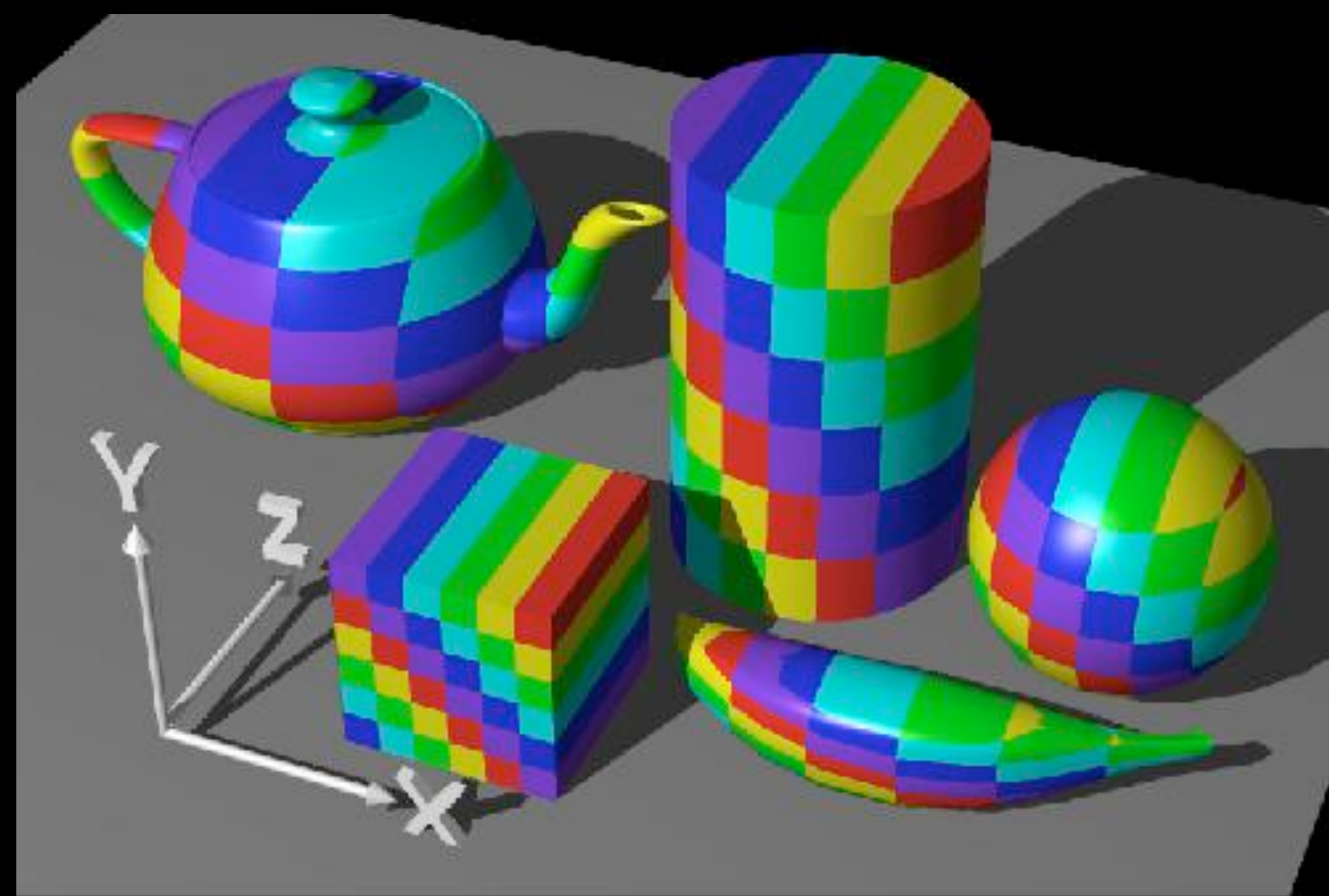
Given a point on the object **(x,y,z)**, what point **(u,v)** in the texture we use?

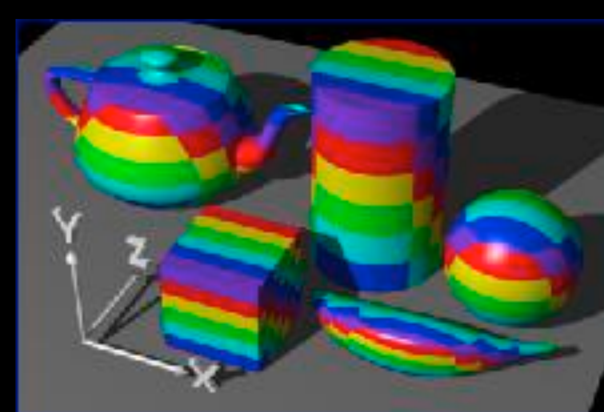


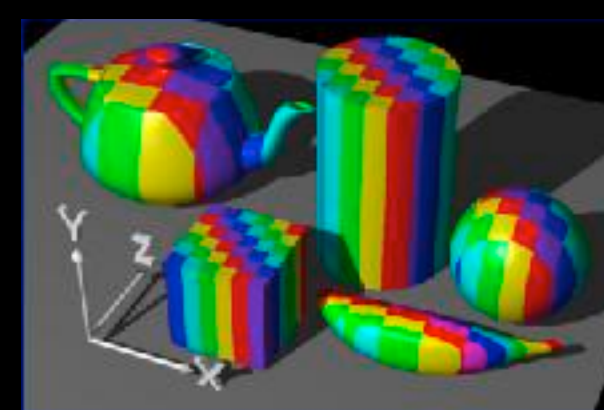
[Rosalee Wolfe]

Example: planar mapping





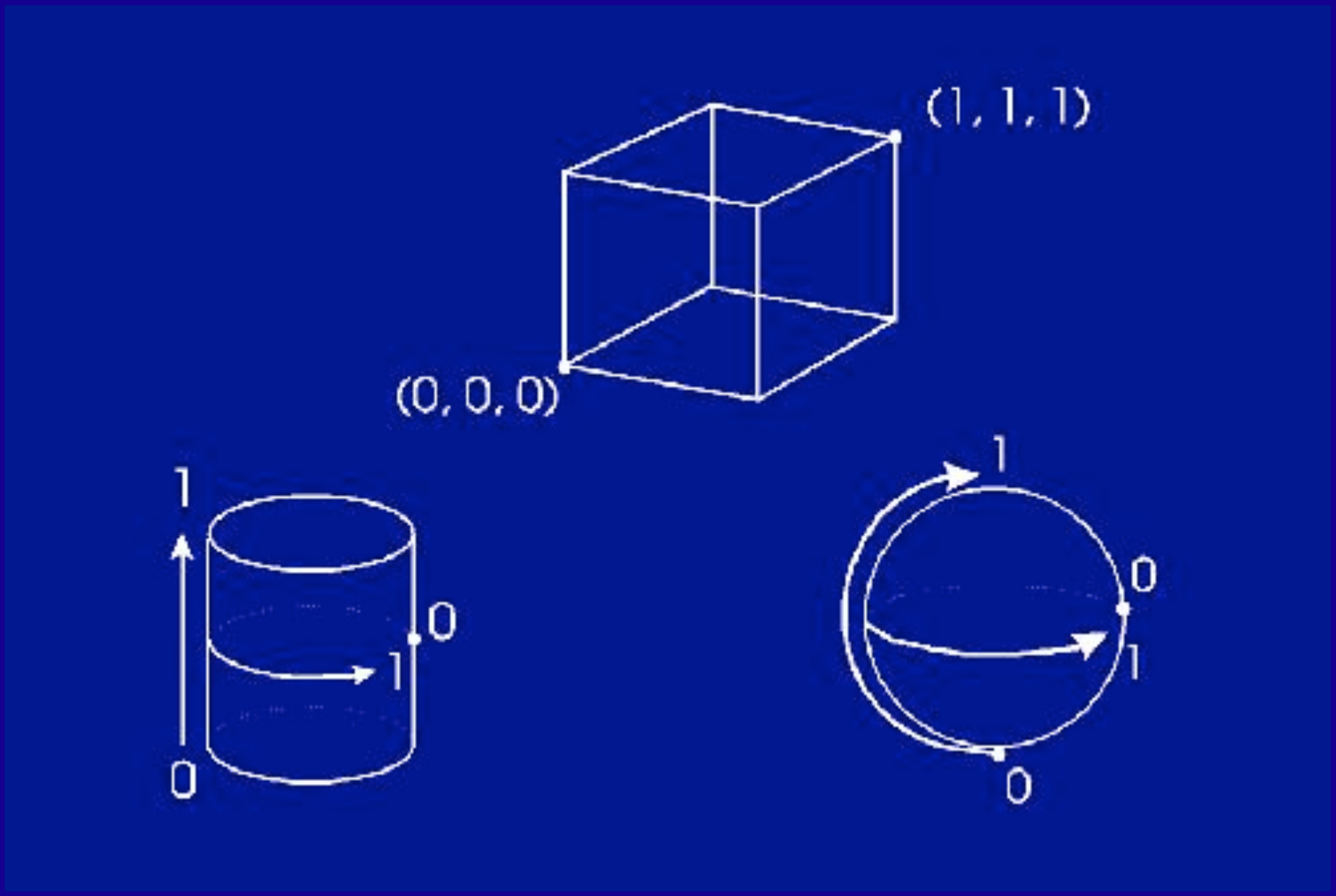




[Rosalee Wolfe]

Intermediate surfaces

First map the texture to a simpler, intermediate surface

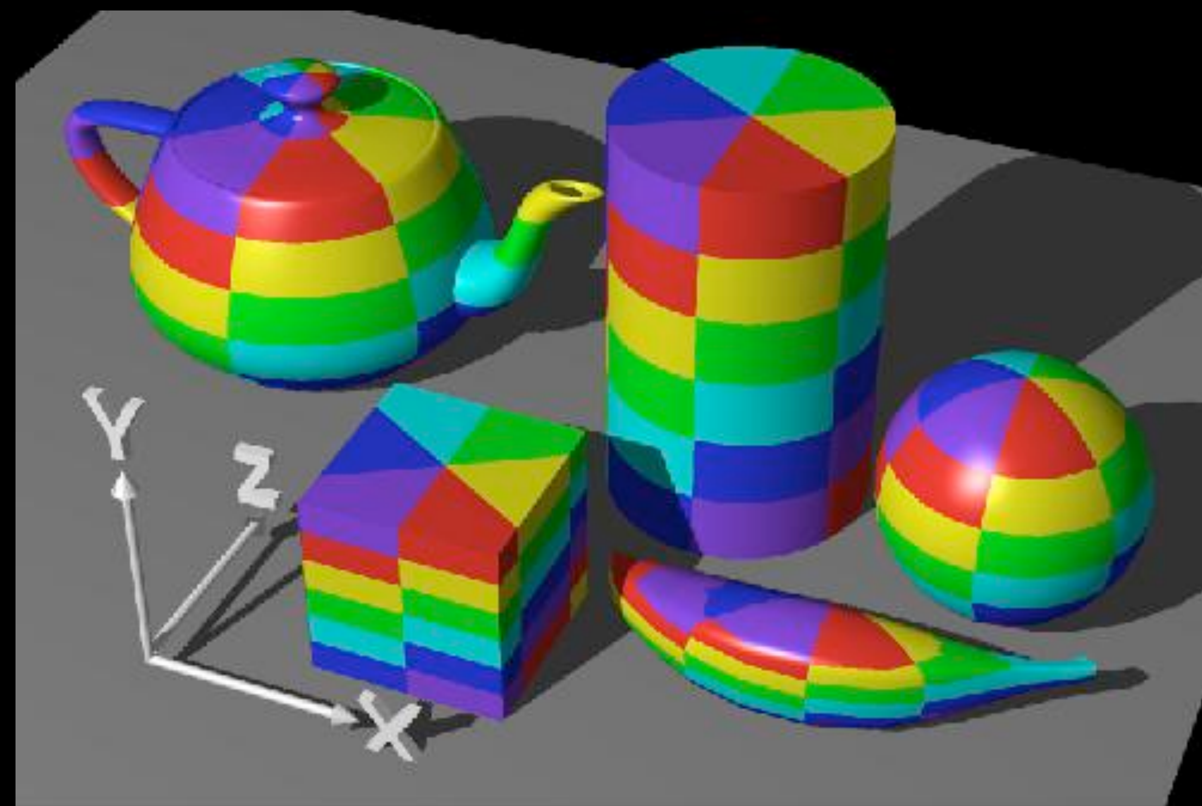


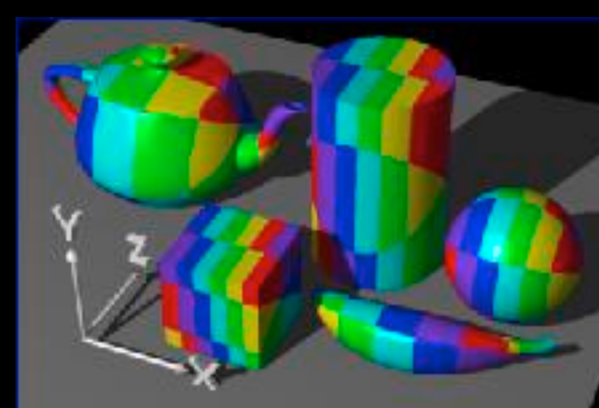
[Rosalee Wolfe]

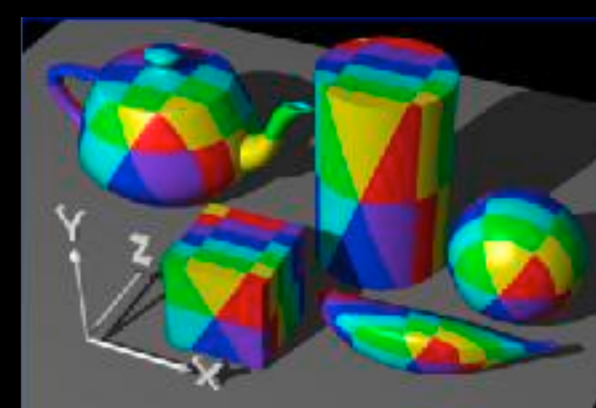
Cylindrical mapping

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

[Rosalee Wolfe]





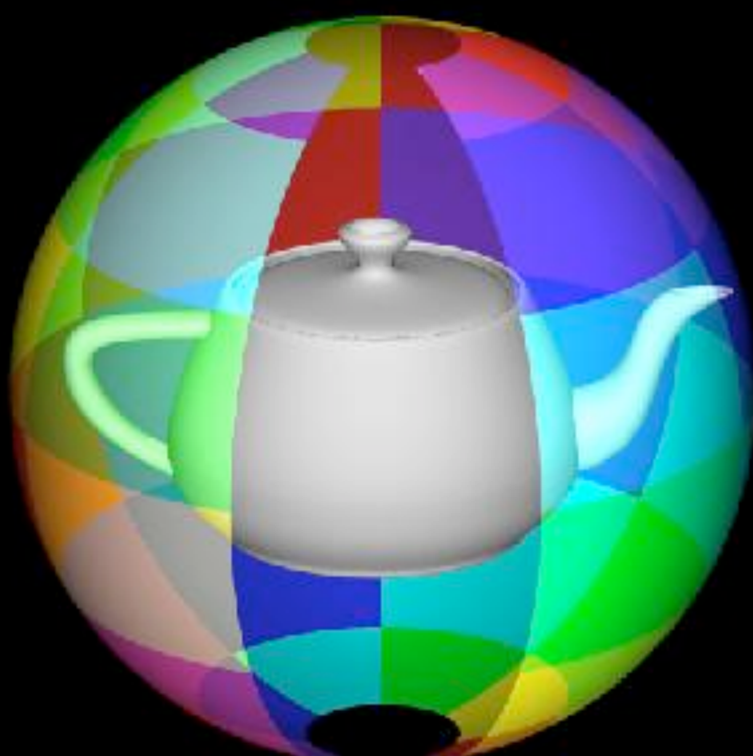


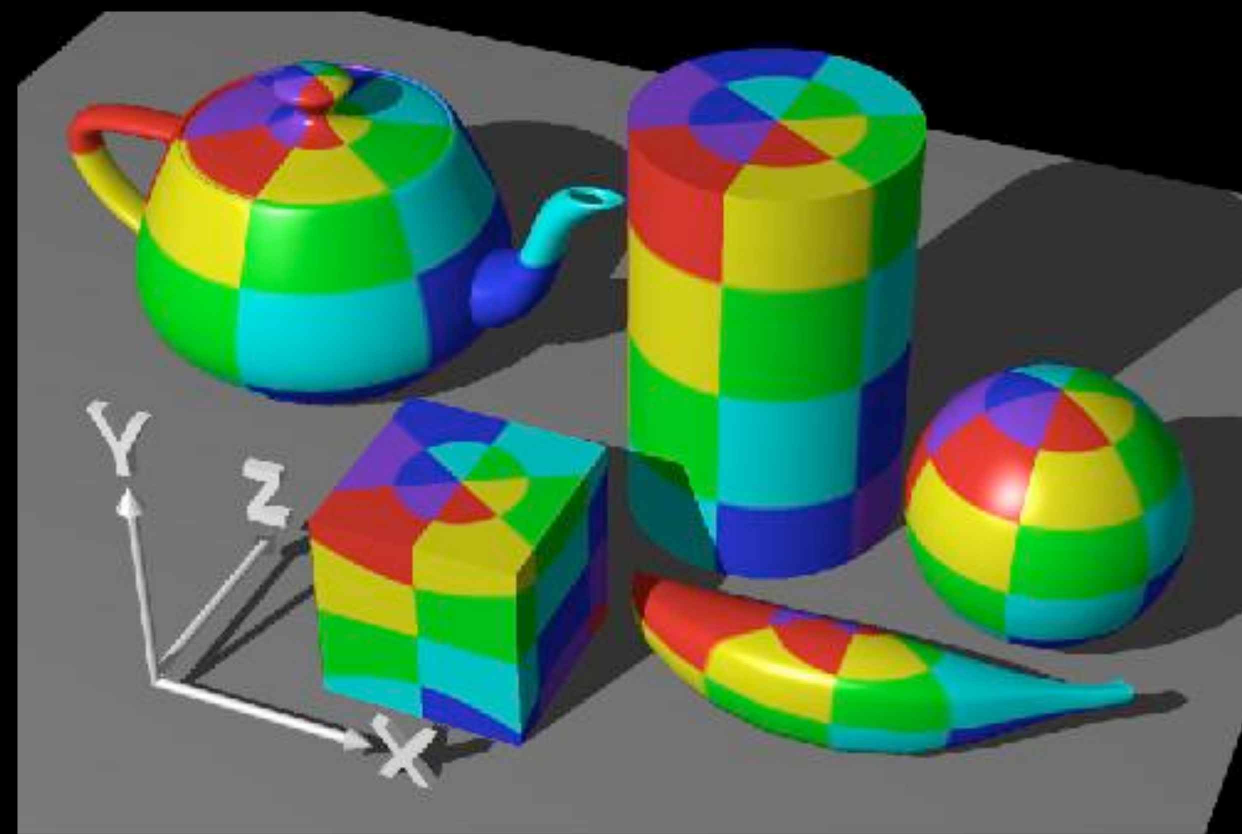
- note “pie slice” phenomena
- which coordinate axis is parallel to the cylinder axis?

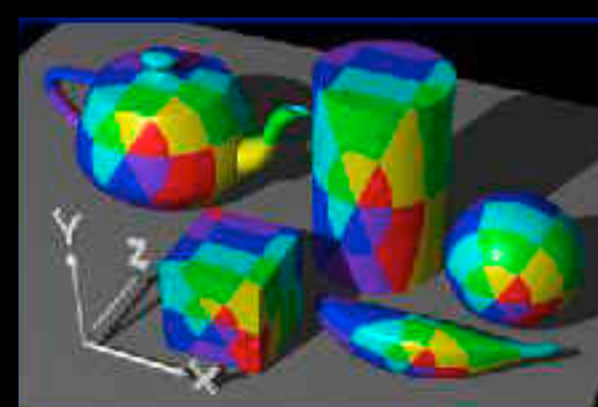
Spherical Mapping

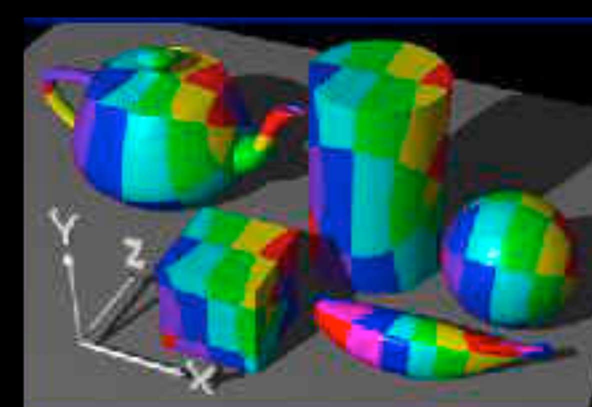
(x,y,z) -> (latitude,longitude)
-> (u,v)

[Rosalee Wolfe]



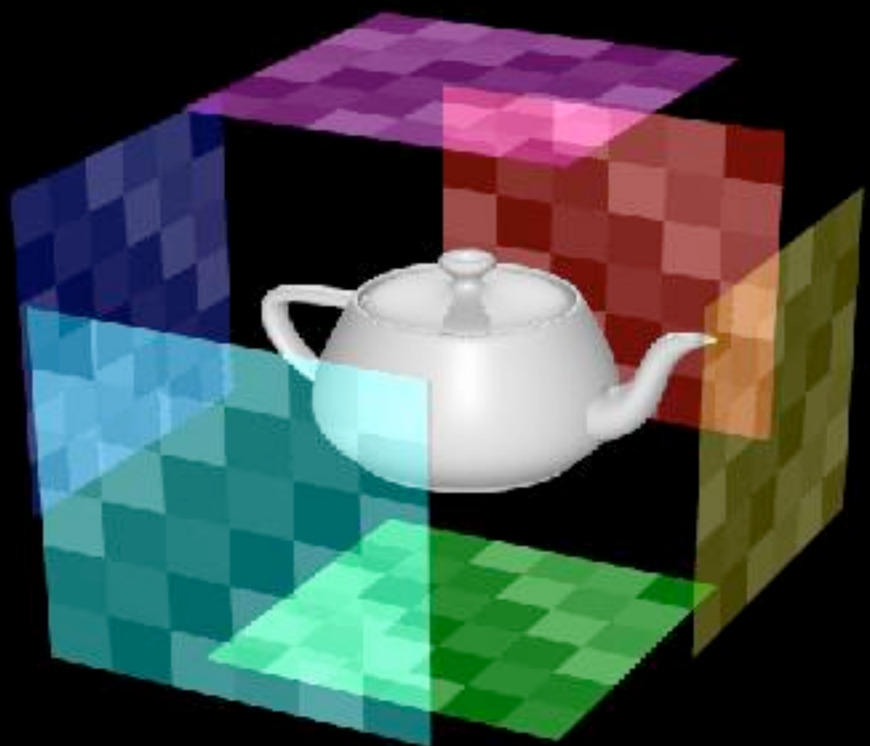


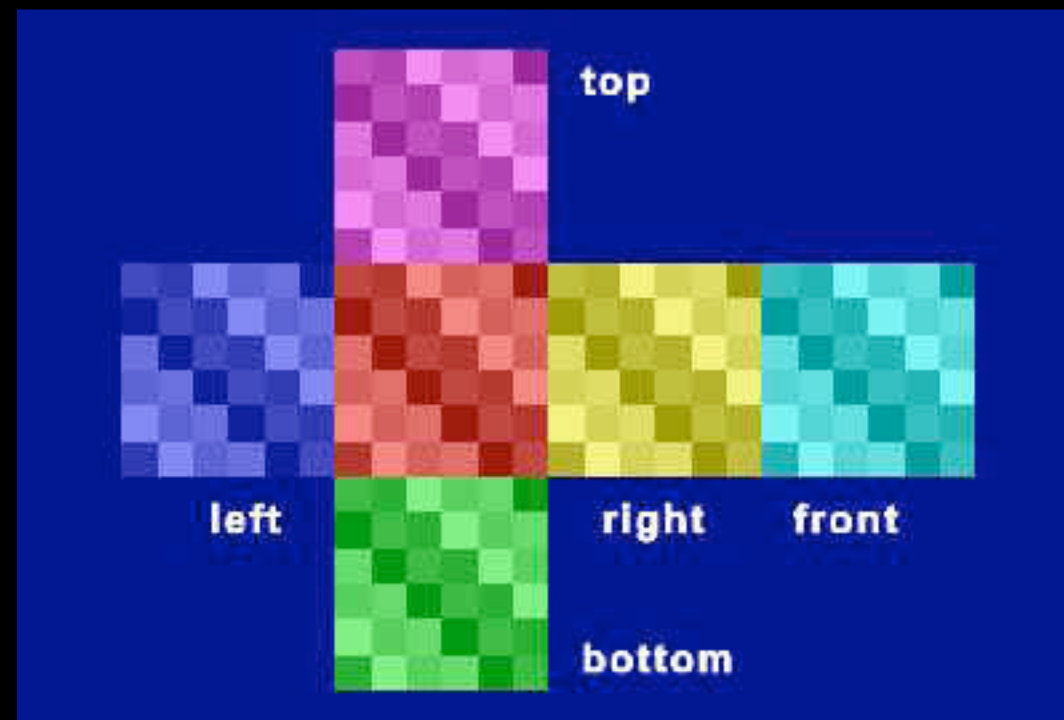


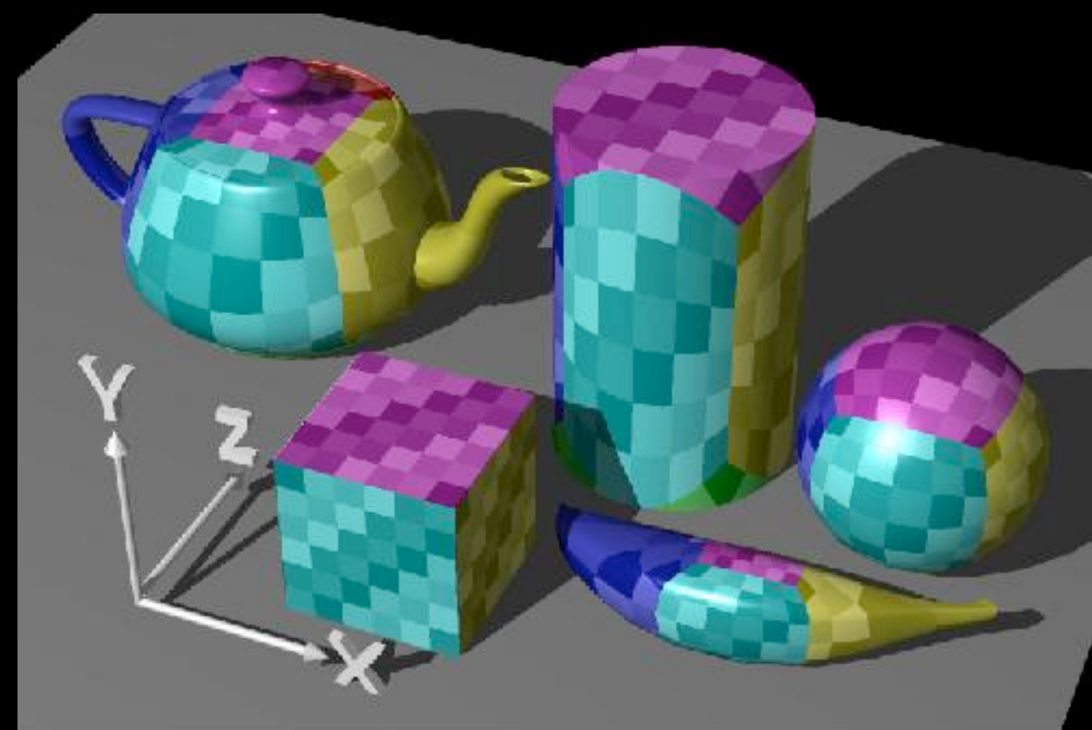


spherical map stretches squares at equator and squeezes squares at poles

Box Mapping



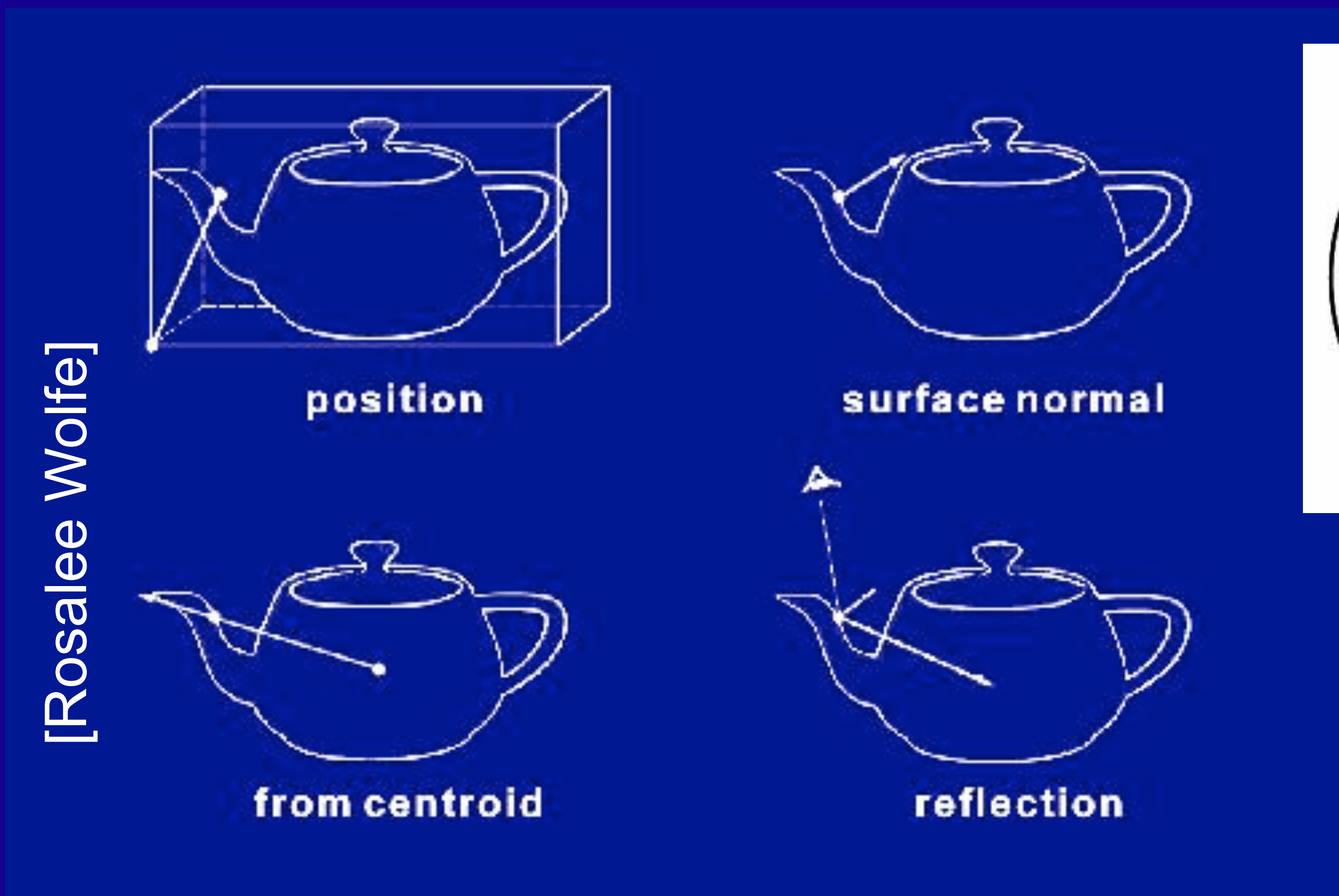


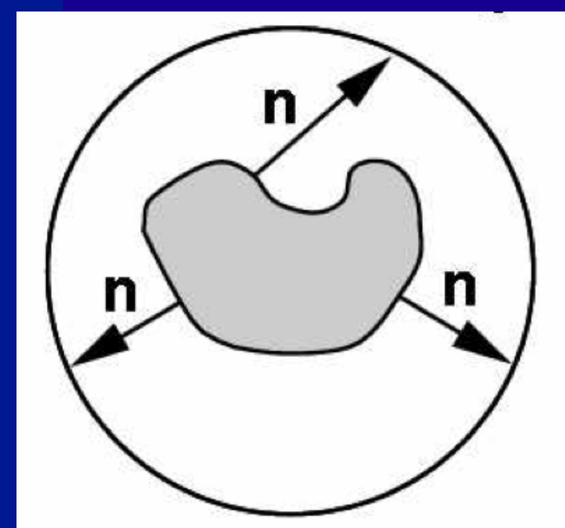


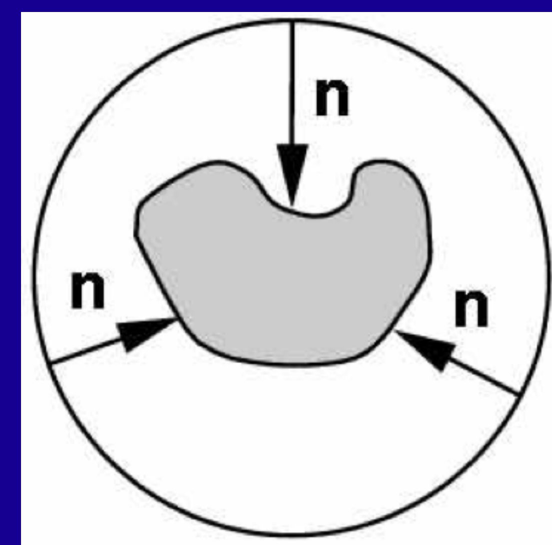
[Rosalee Wolfe]

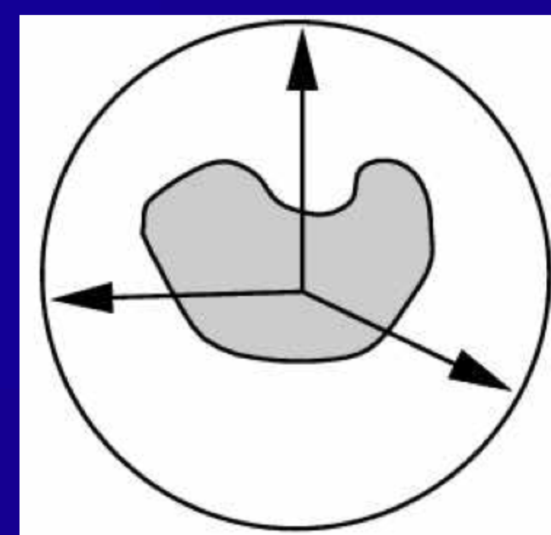
- similar to planar mapping
- planar projection -- choose which plane to project onto

How do we map between intermediate and actual objects?





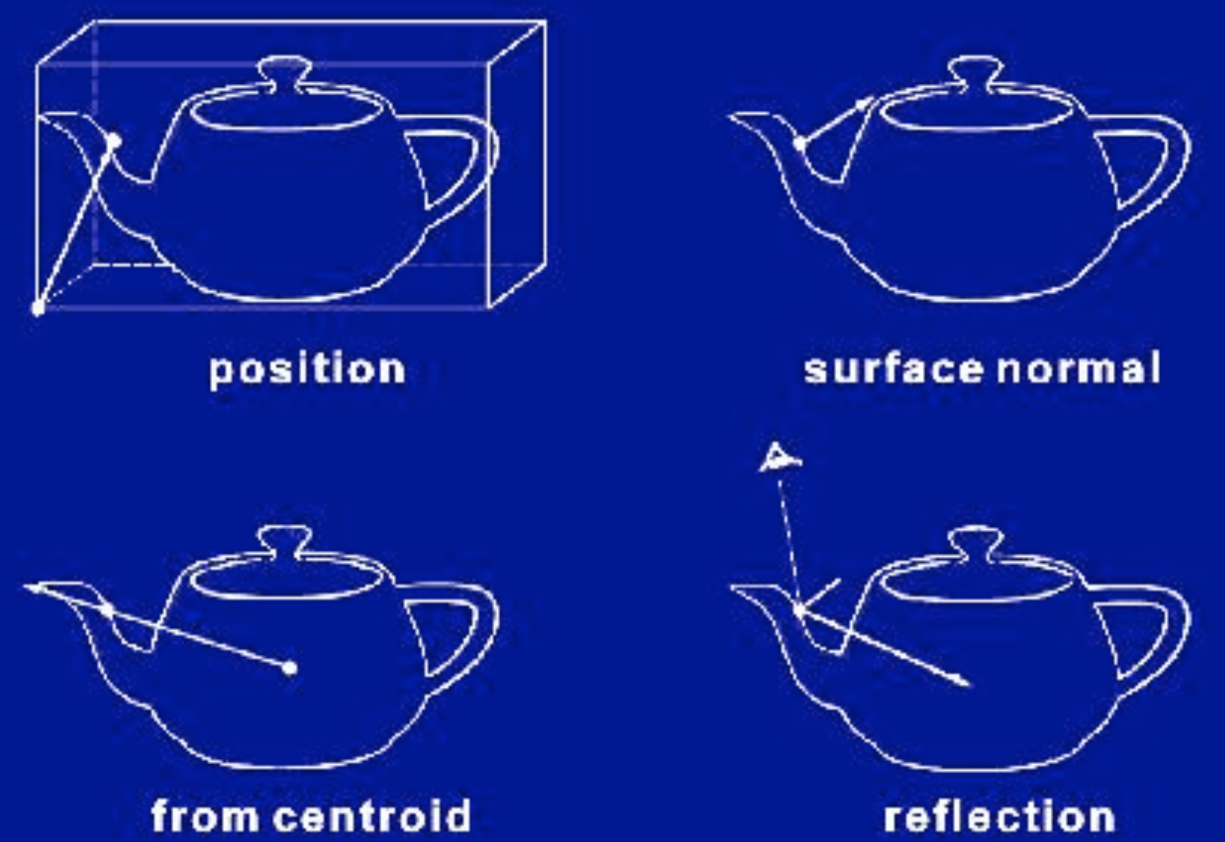


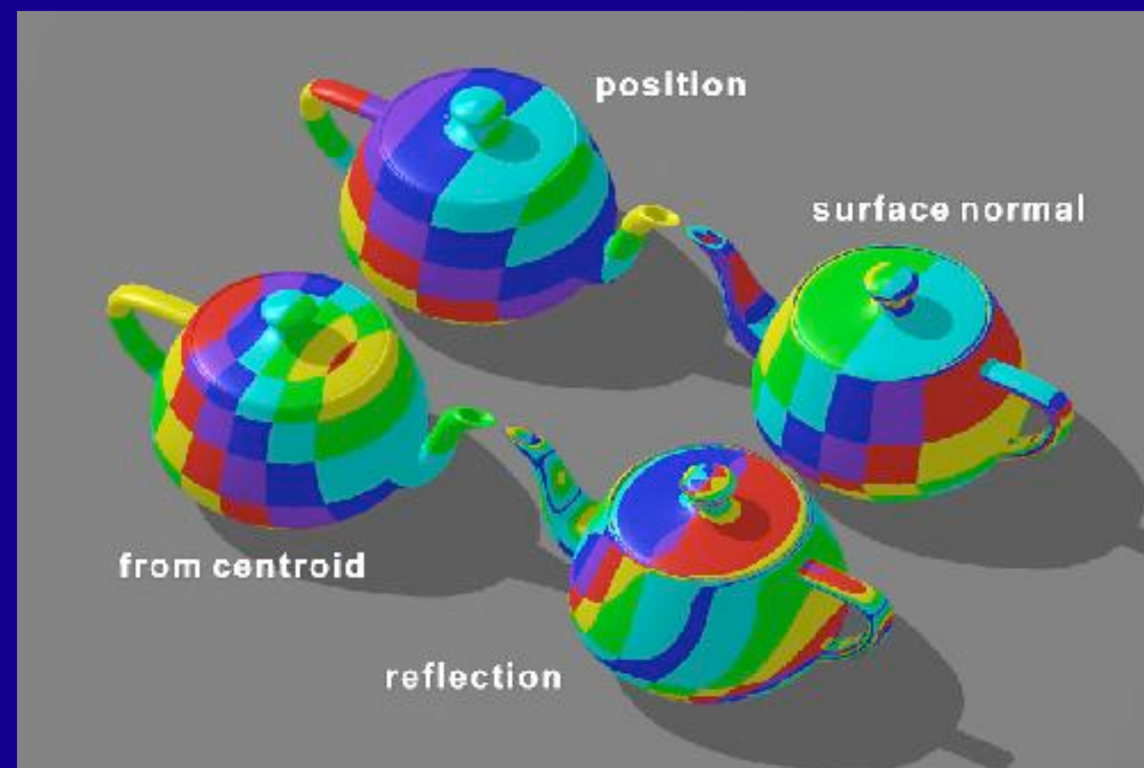


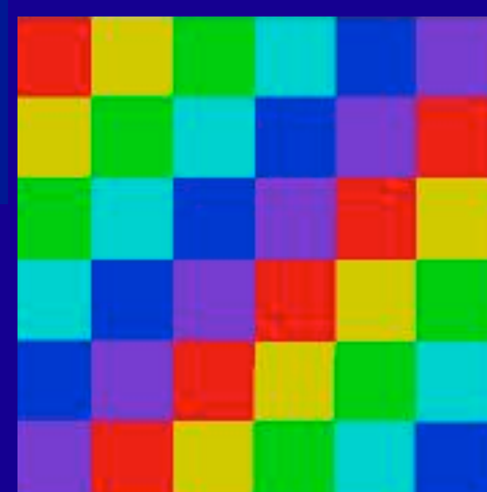
We associated (x,y,z) on the intermediate object with the texture (u,v). But which point on the actual object is this?
We choose both the **intermediate shape** and the **mapping from the actual shape to the intermediate shape**
1. a point on the object relative to its bounding box
2. see where surface 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?

[Rosalee Wolfe]

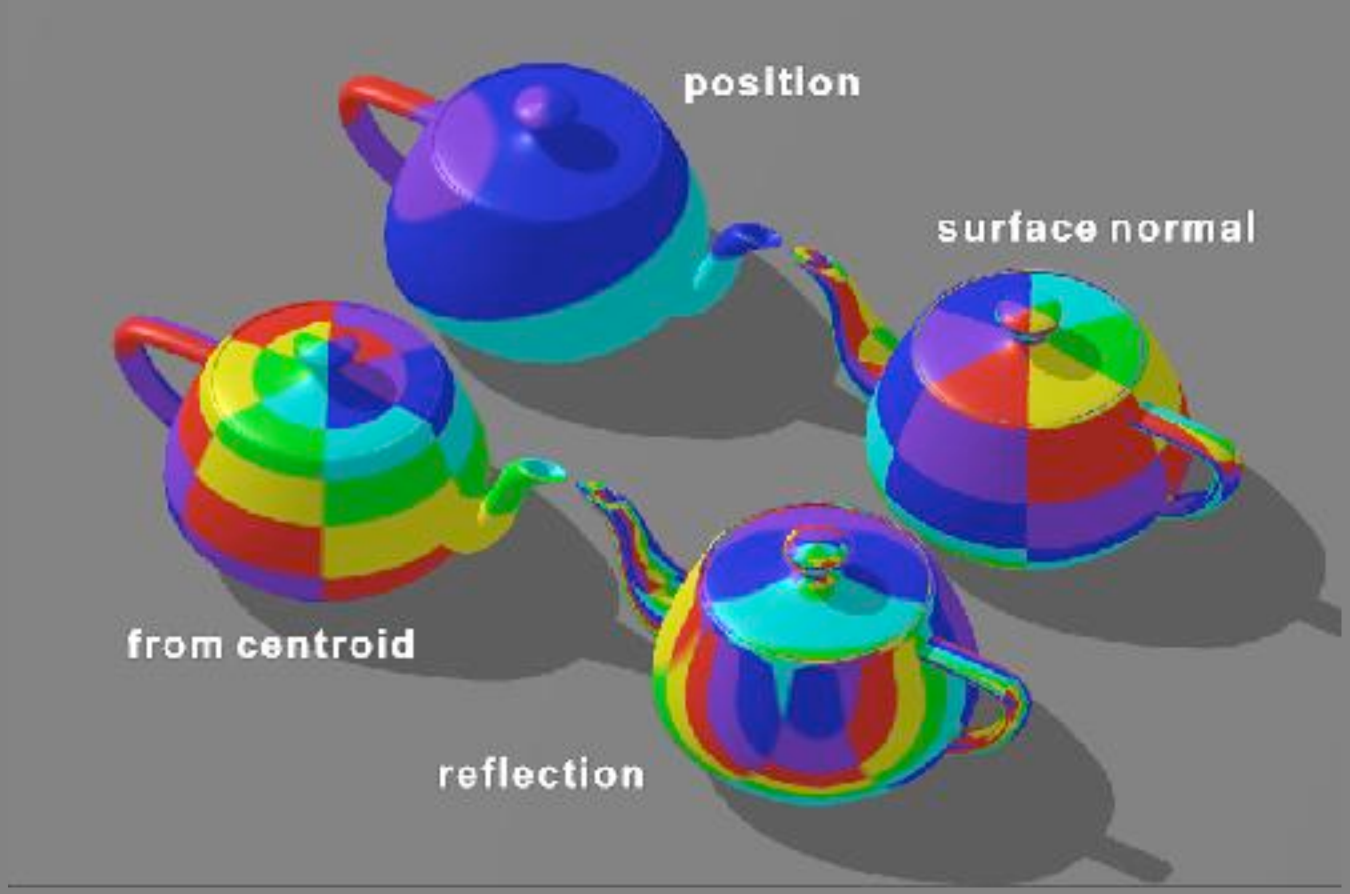




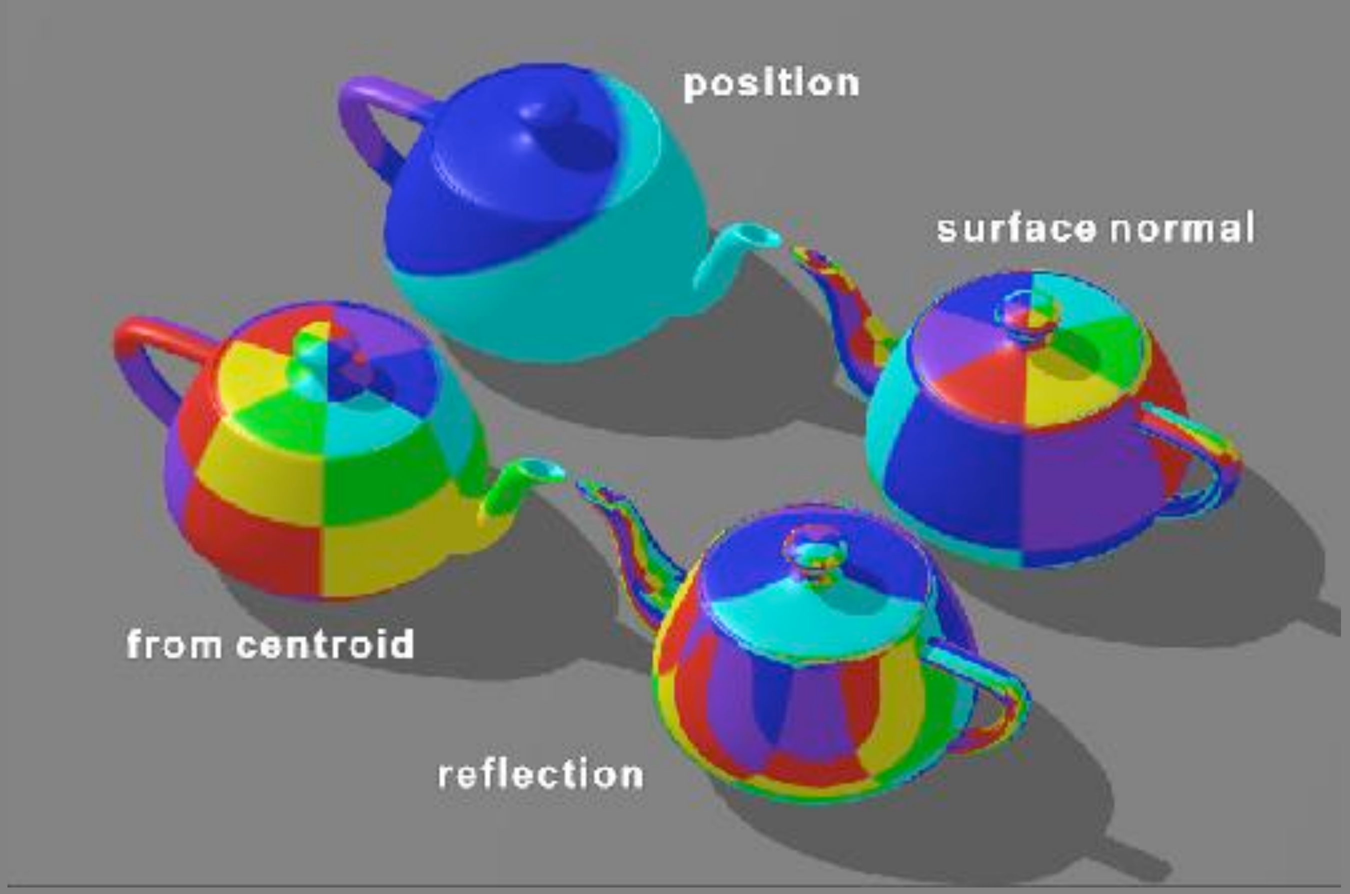


What intermediate shape was used here?

Can you tell what intermediate shape was used?
Planar map - in xy plane

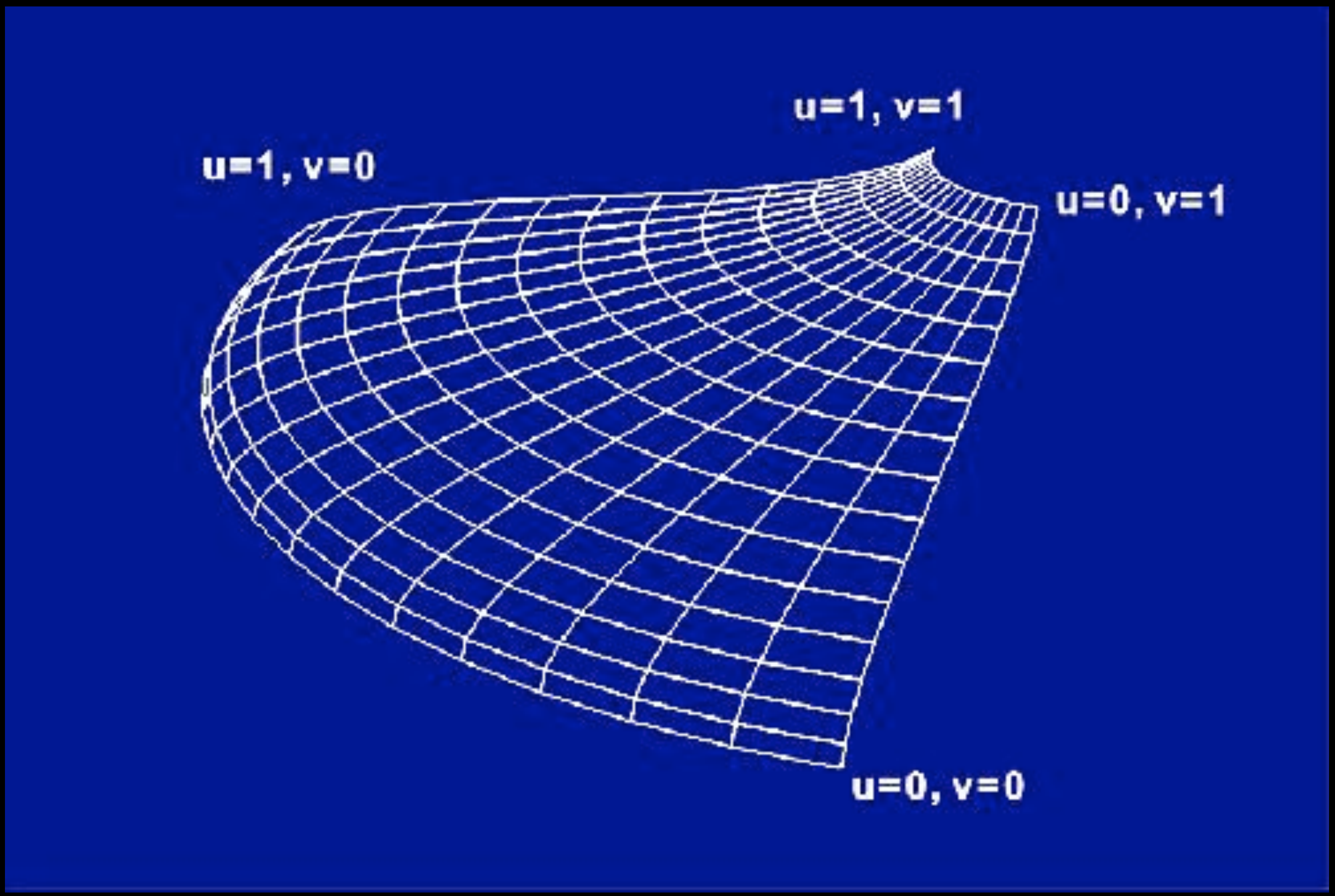


Cylindrical



Spherical

Parametric Surfaces



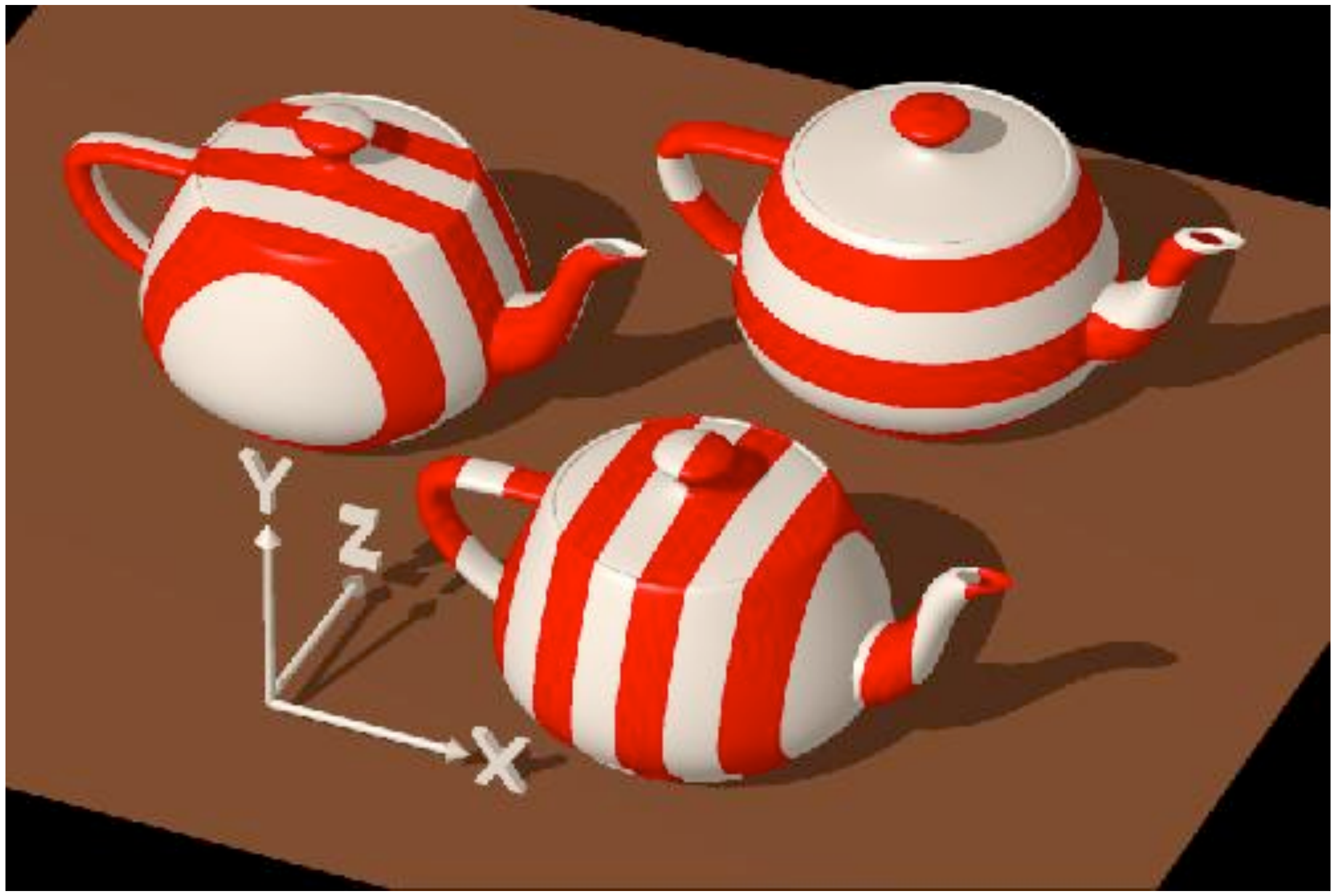
32 parametric patches

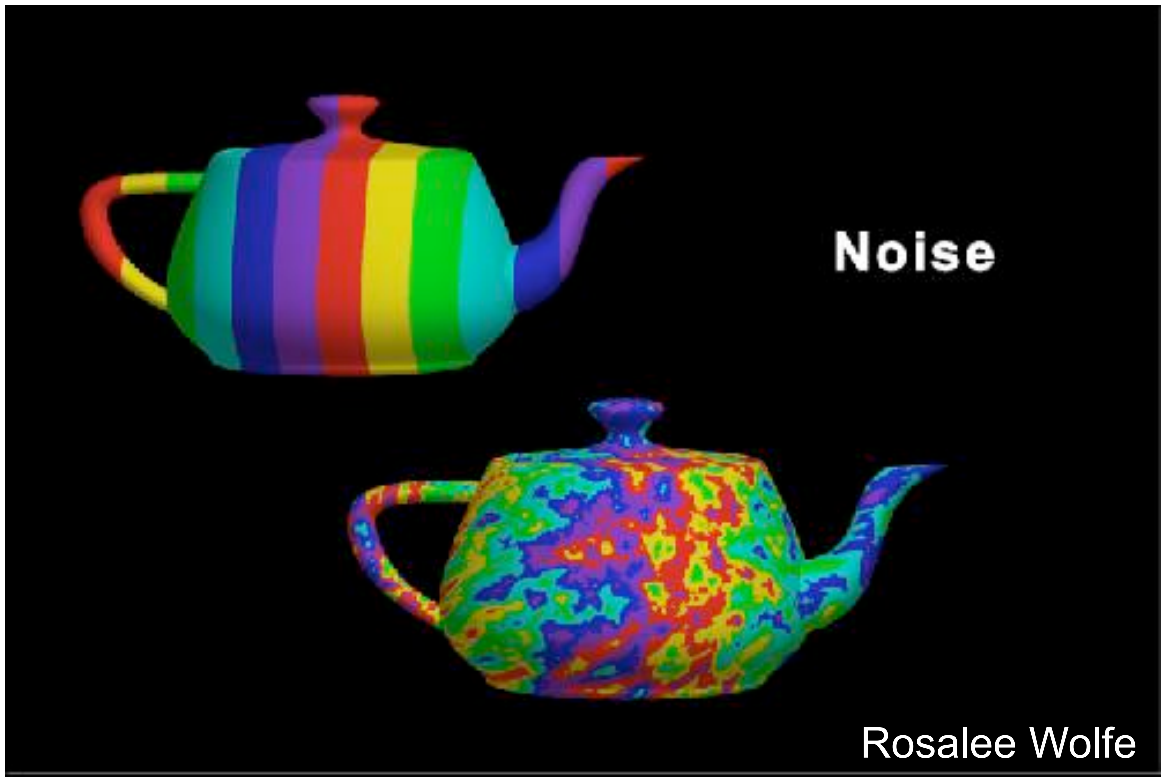
3D solid textures

[Dong et al., 2008]

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

Procedural textures

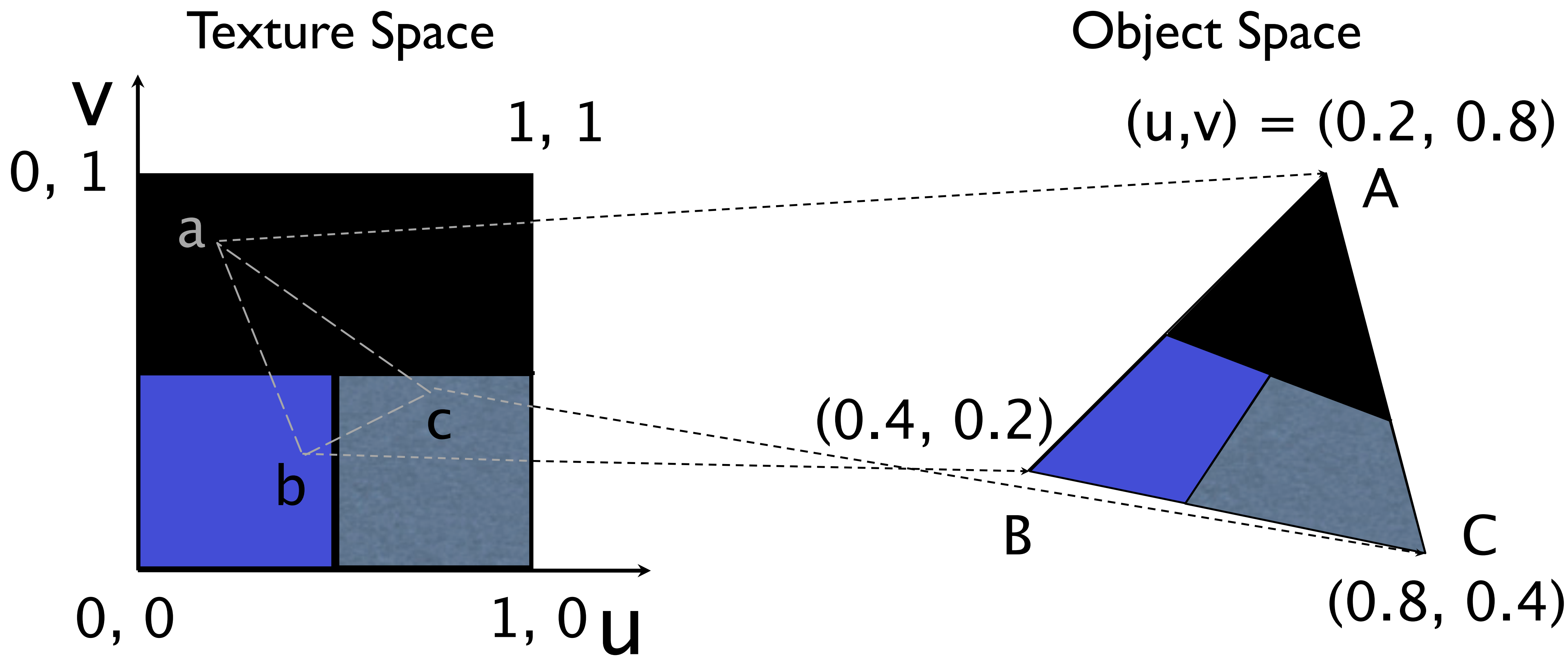




e.g., Perlin noise

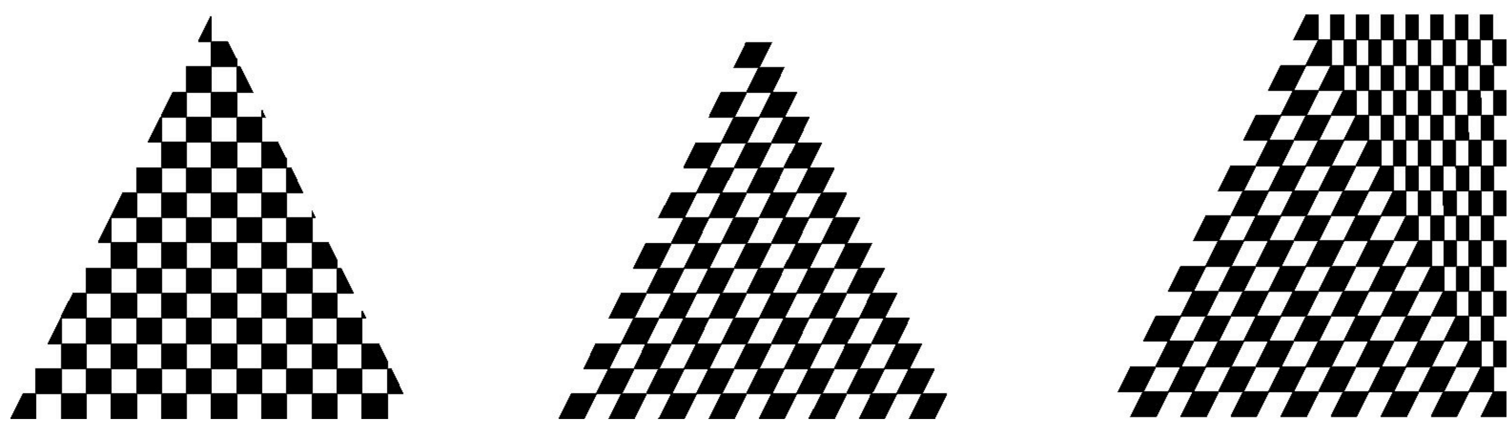
Triangles

Texturing triangles

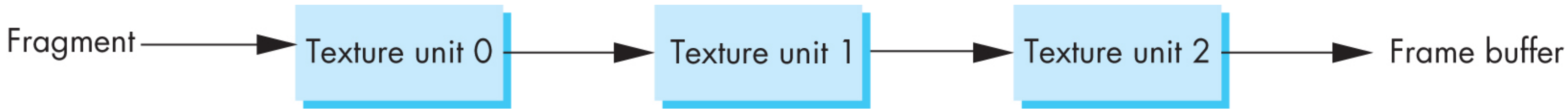


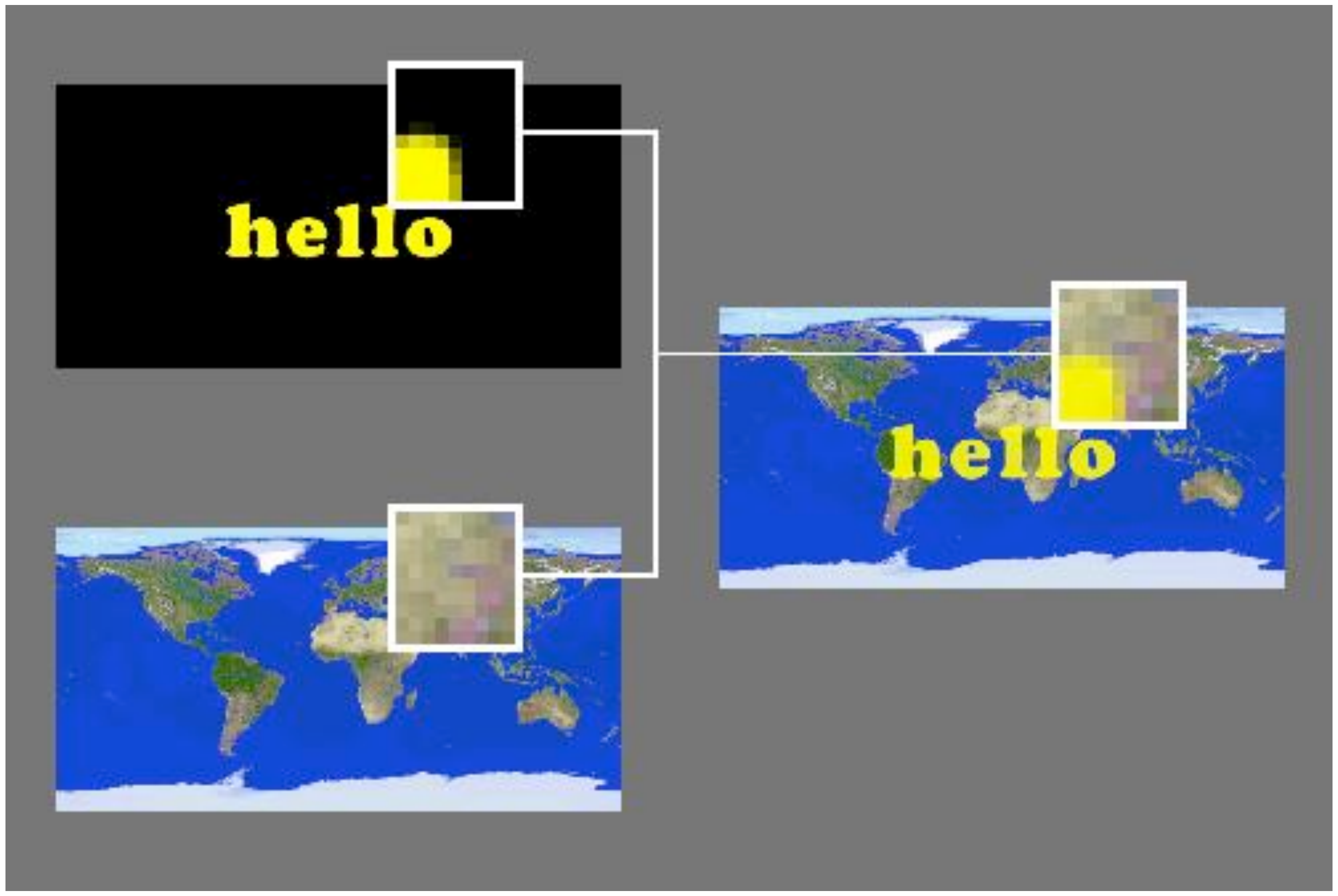
[Angel and Shreiner]

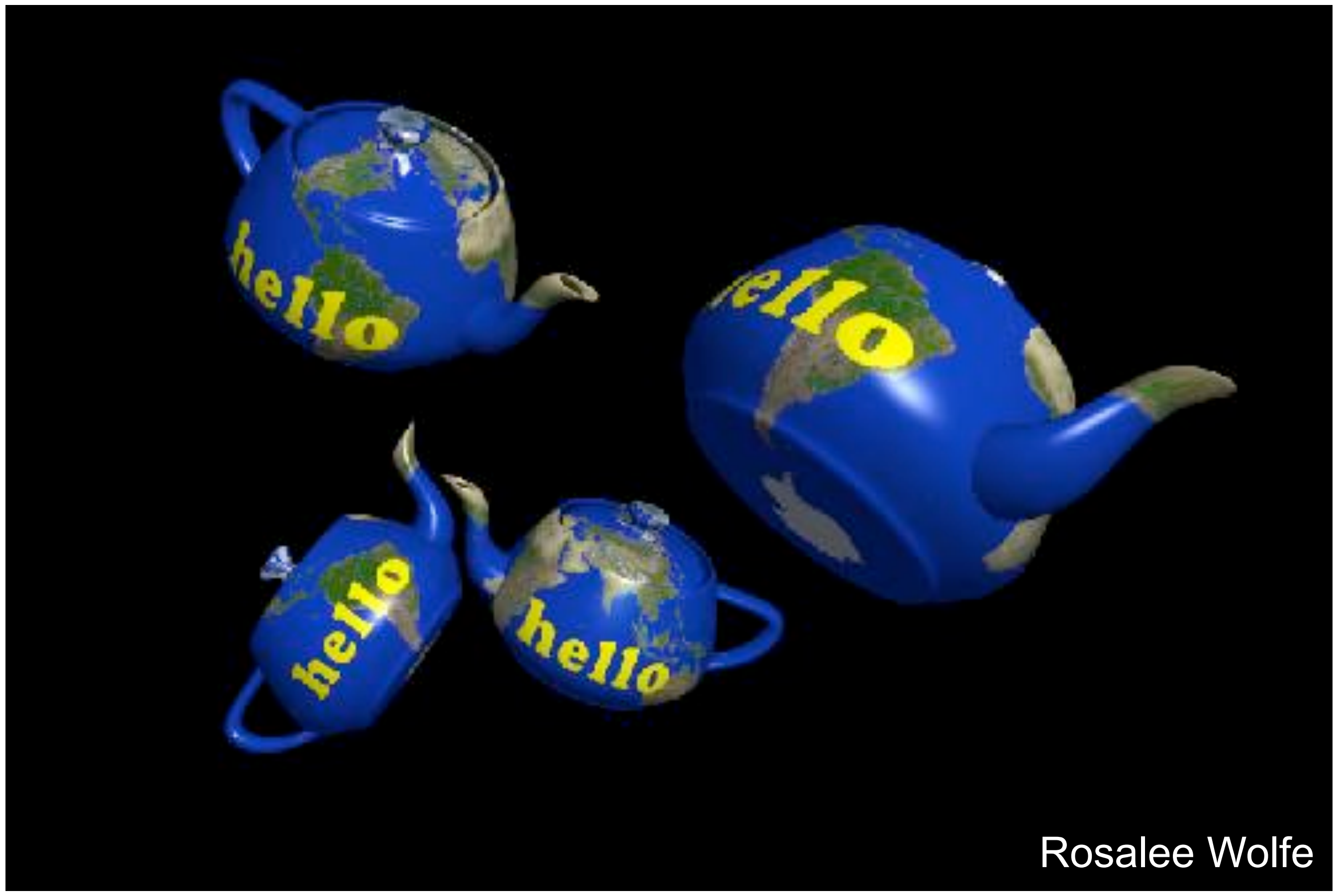
`glTexCoord*()`



Multitexturing

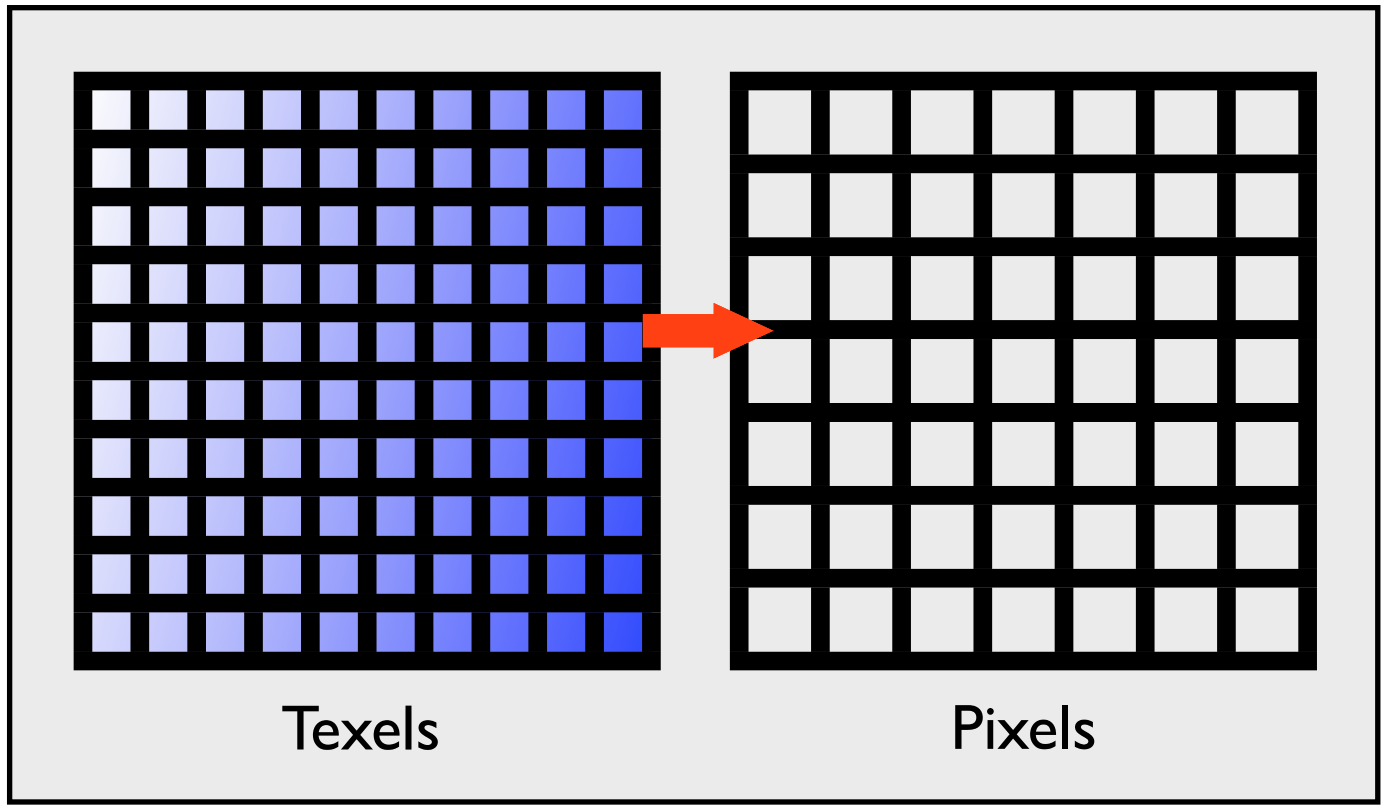


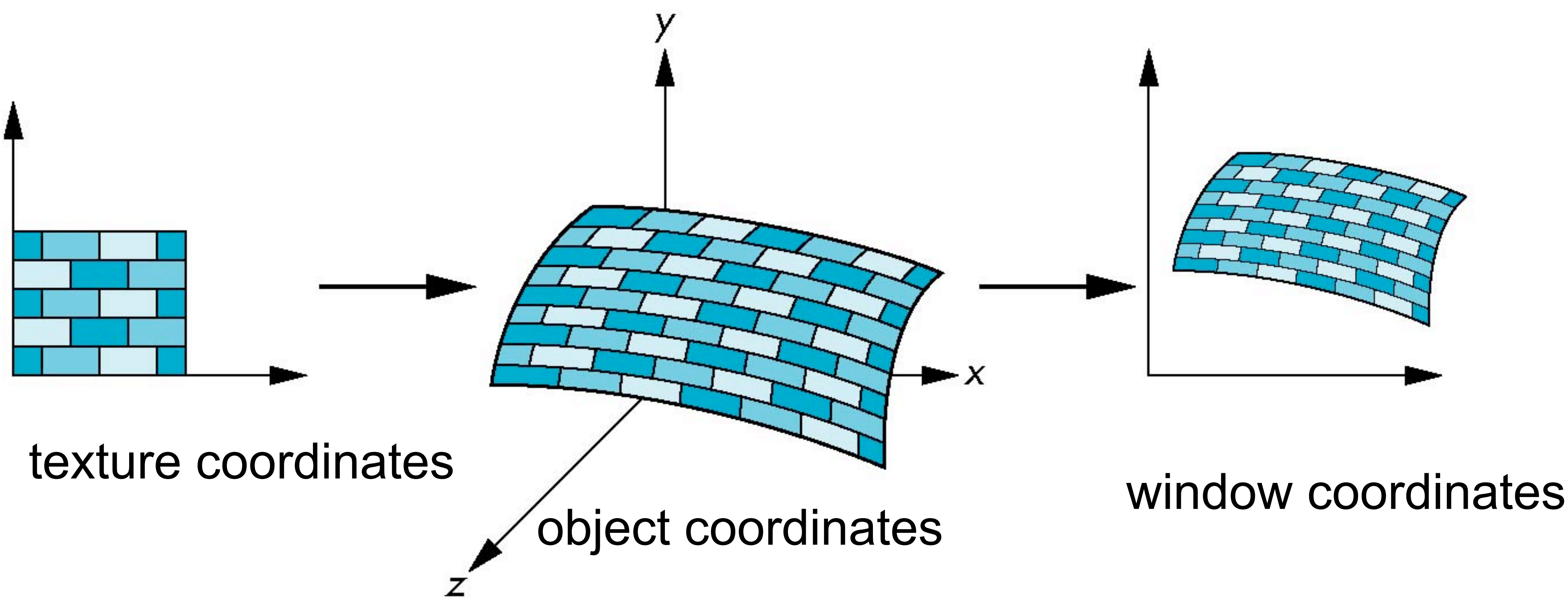




Texture Sampling

Texture Mapping



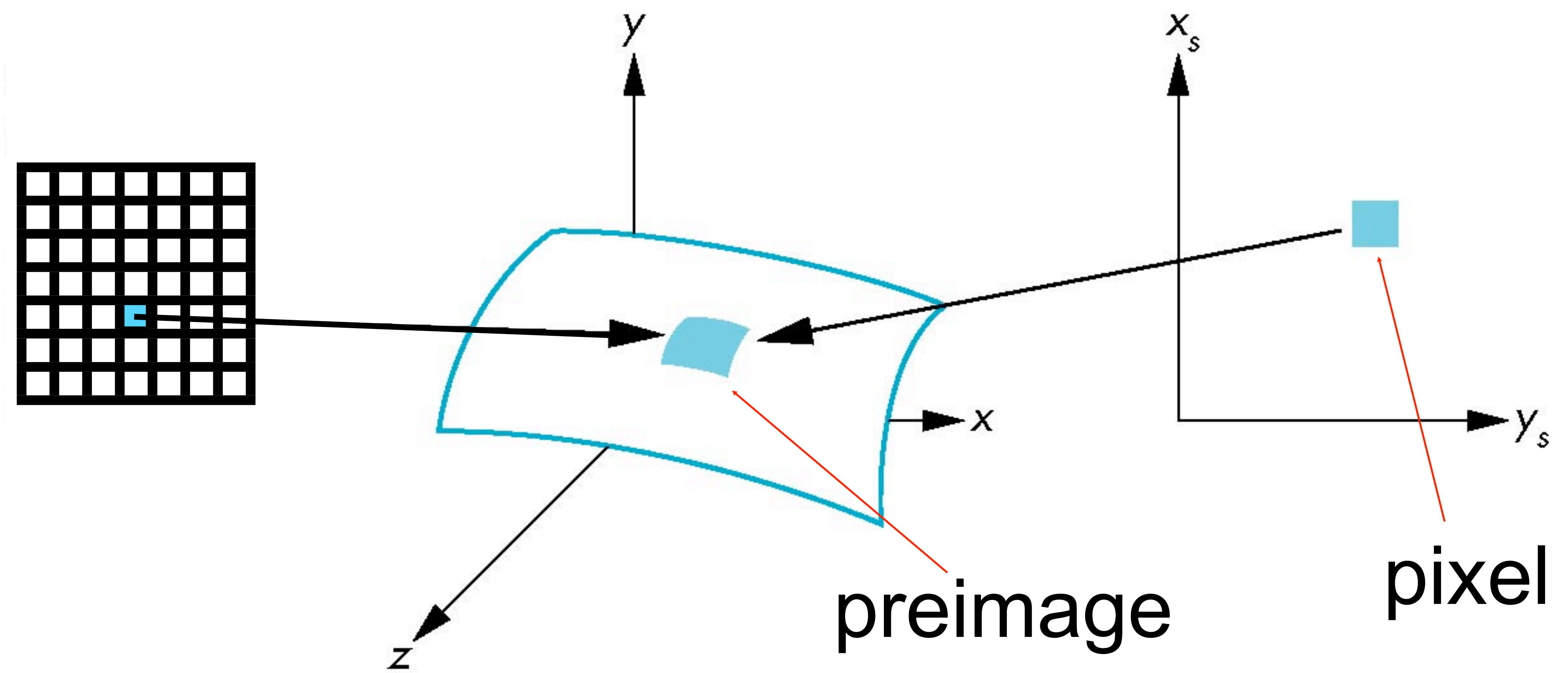


[Angel and Shreiner]

- Texture coordinates: Used to identify points in the image to be mapped
- Object Coordinates: Conceptually, where the mapping takes place
- Window Coordinates: Where the final image is really produced

Point Sampling

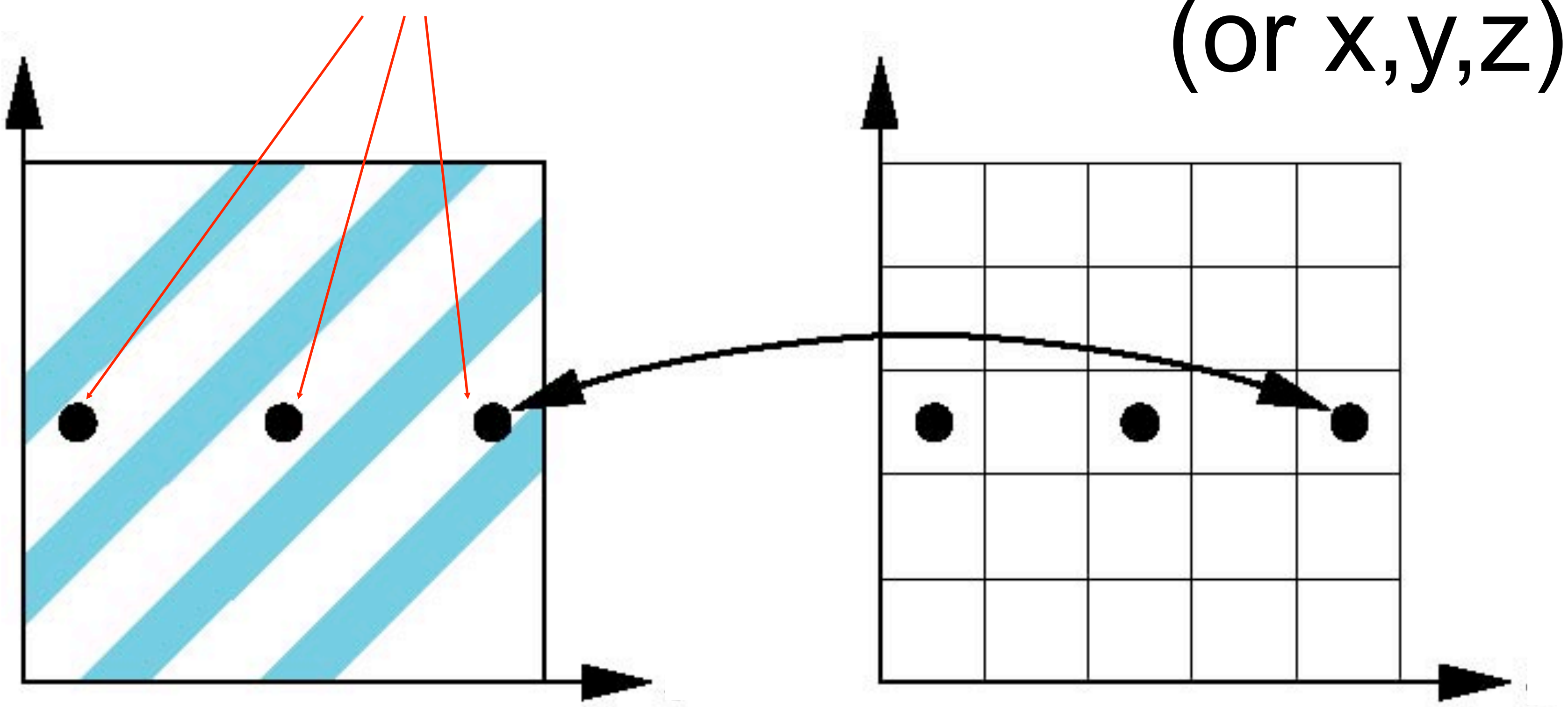
Map back to texture image and use the **nearest texel**



[Angel and Shreiner]

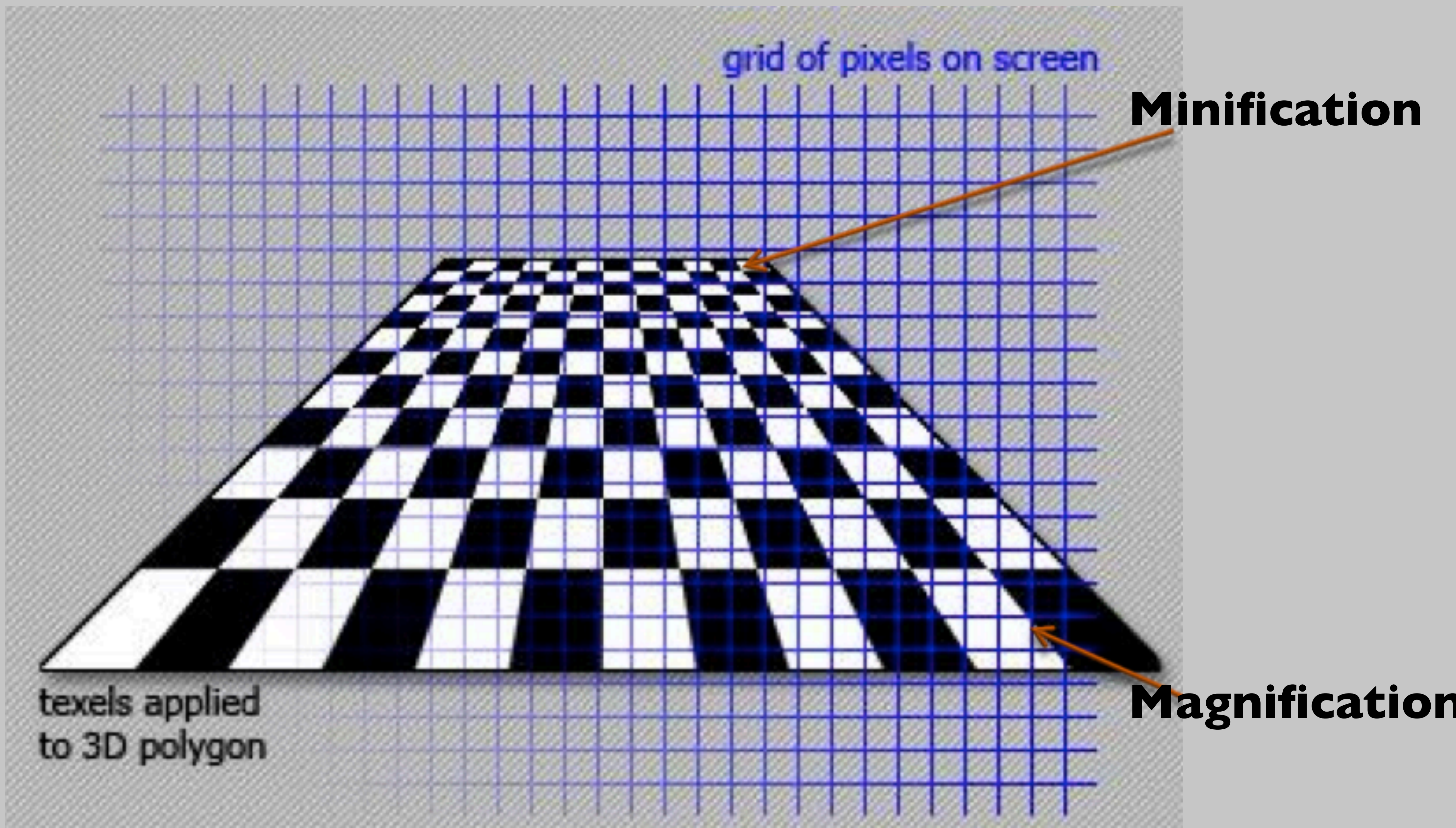
Aliasing

Point sampling of the texture can lead to aliasing artifacts



[Angel and Shreiner]

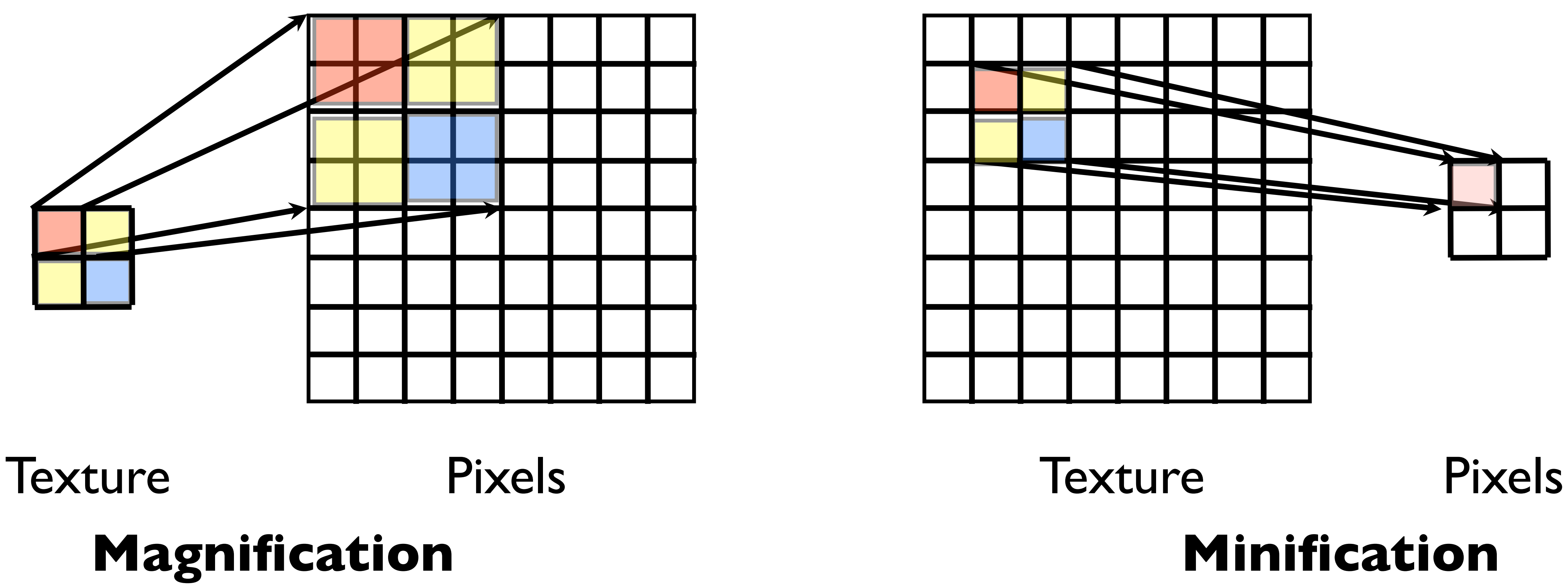
Magnification and Minification



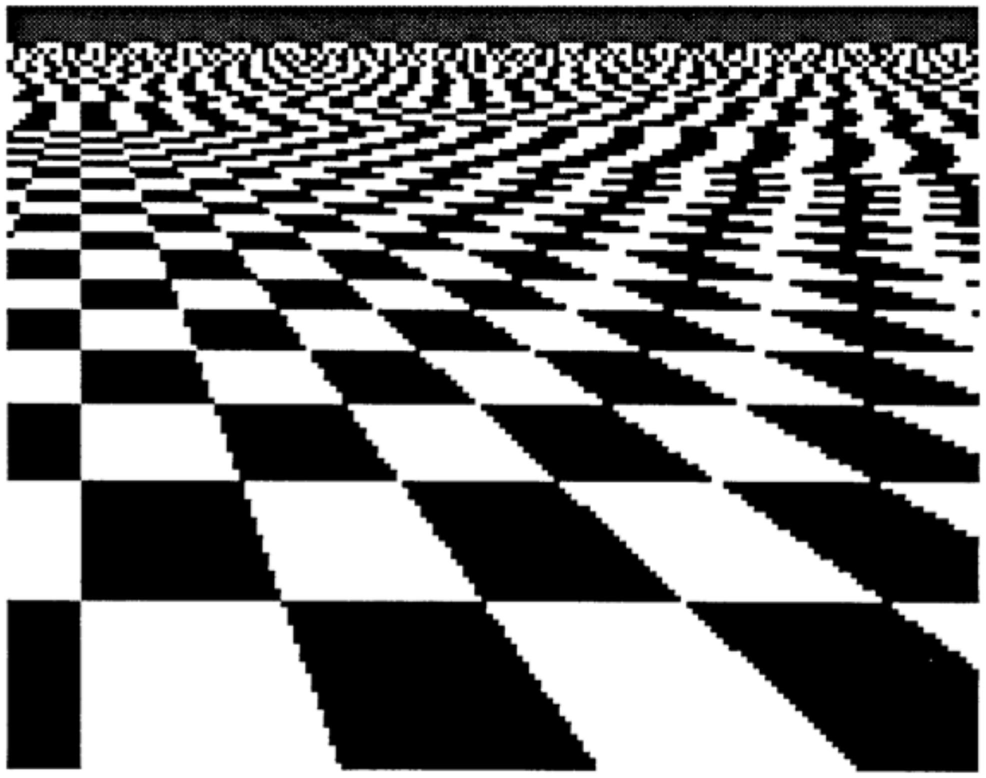
Magnification and Minification

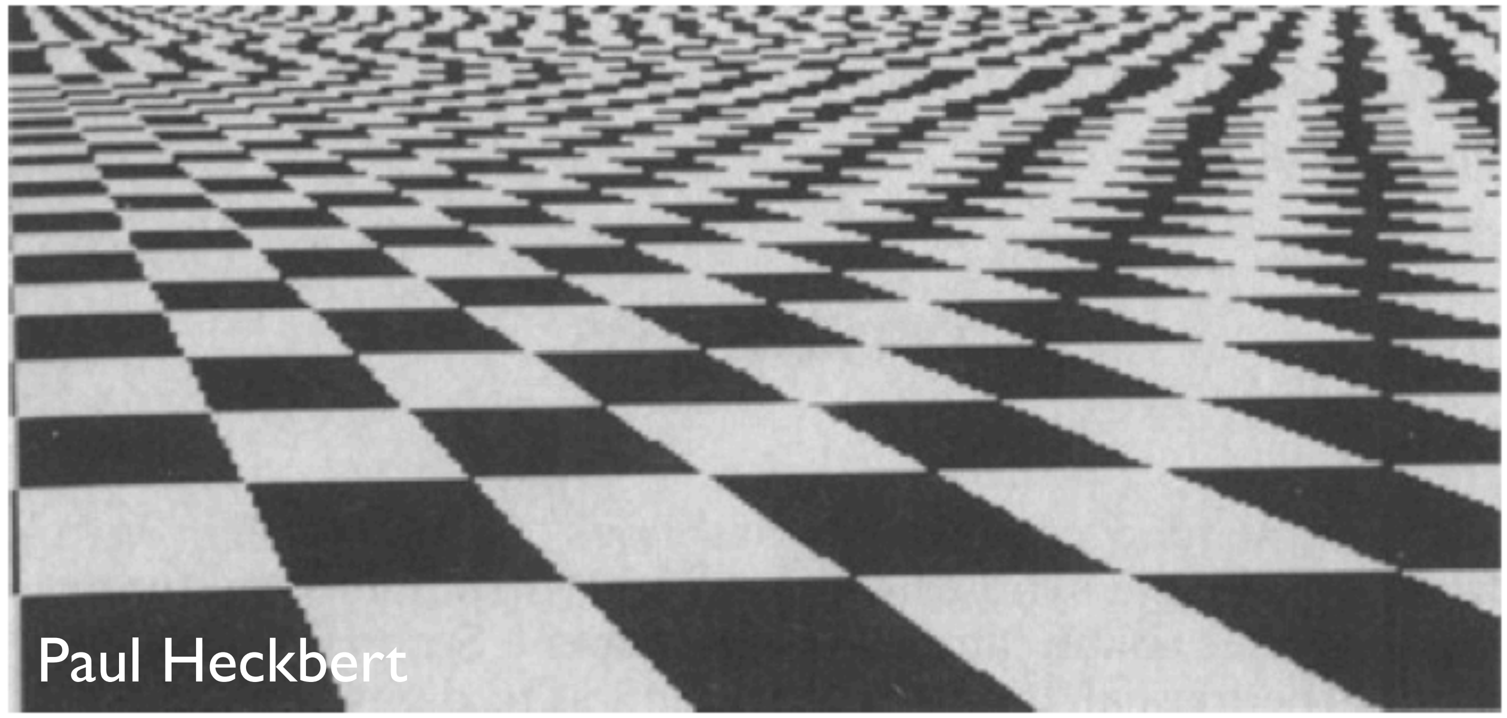
More than one texel can cover a pixel (*minification*) or more than one pixel can cover a texel (*magnification*)

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

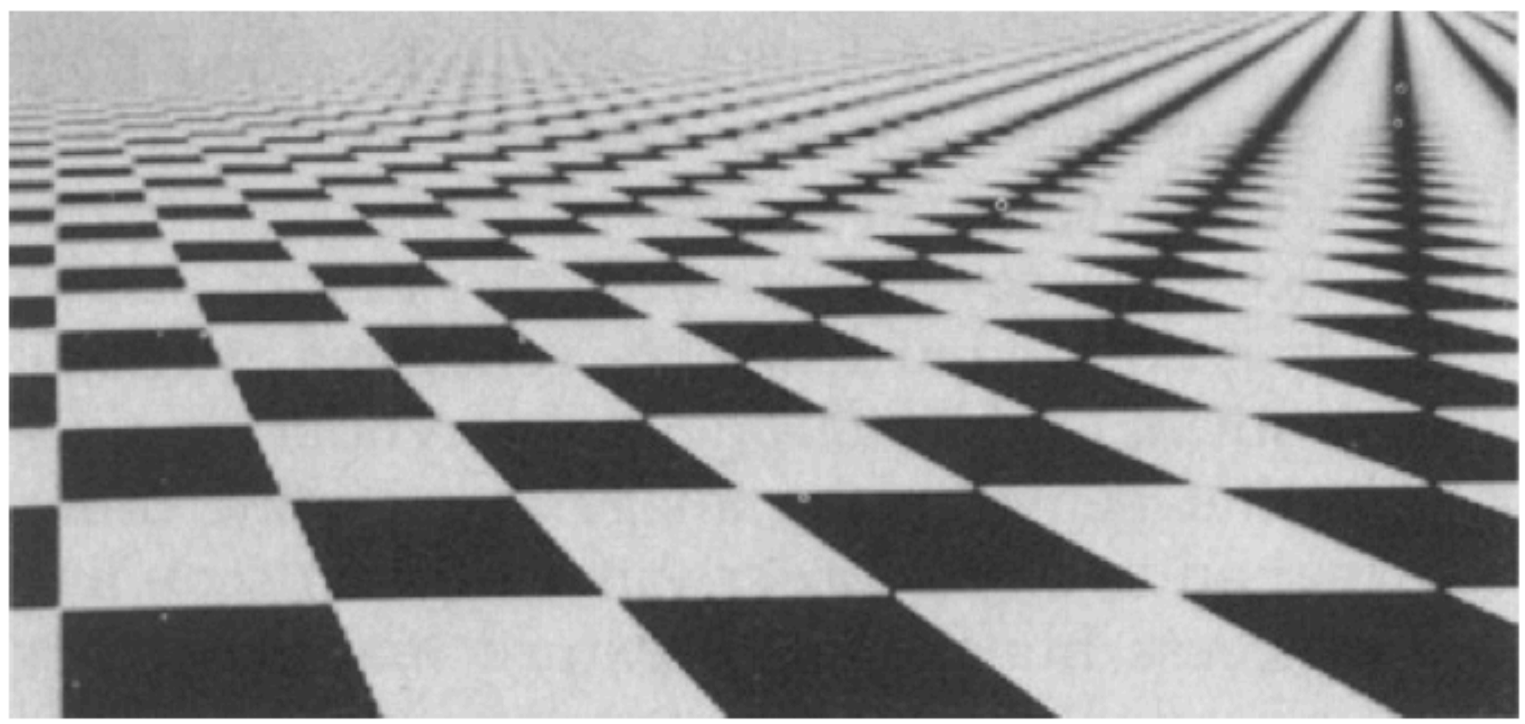


Aliasing artifacts



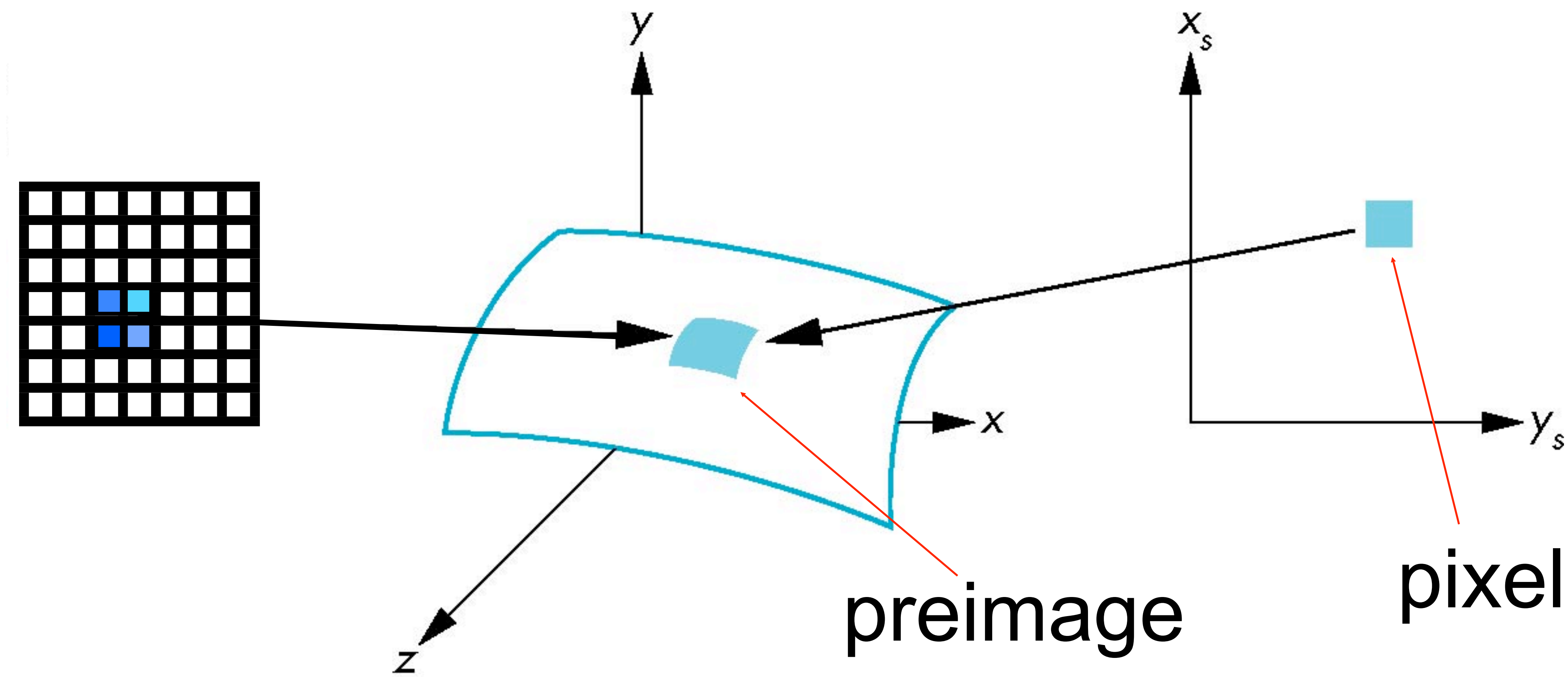


We apply **filtering** to reduce aliasing artifacts



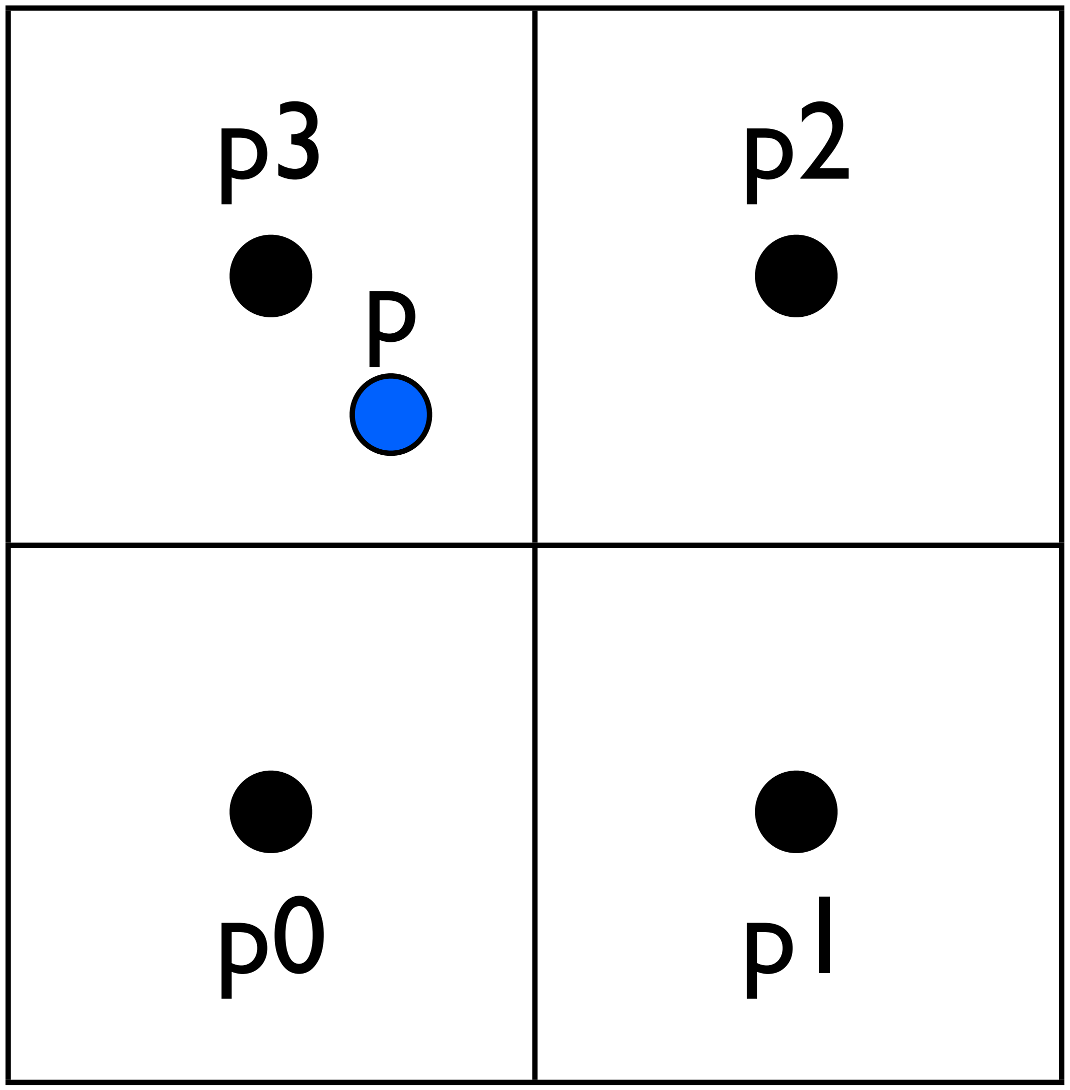
Area Averaging

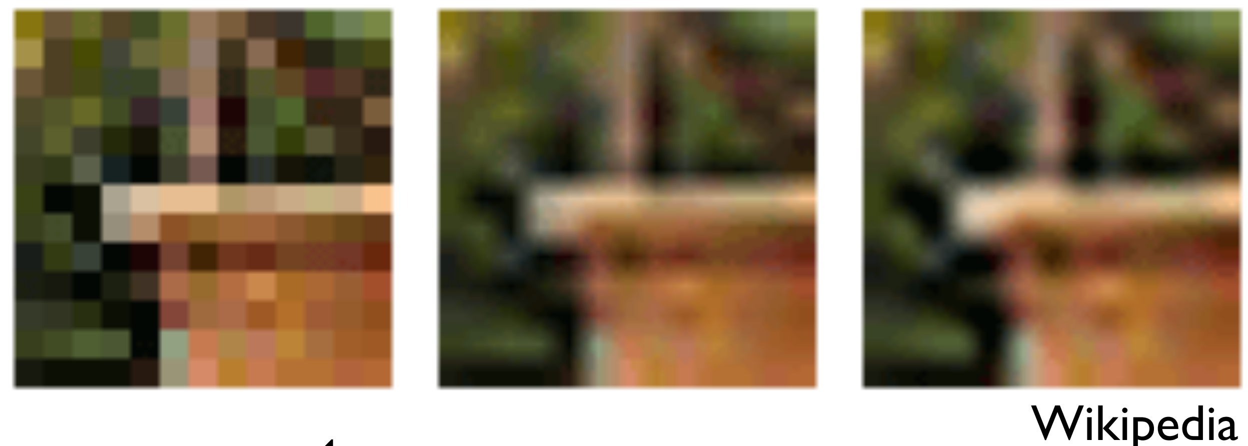
A better but slower option is to use **area averaging**



[Angel and Shreiner]

Use bilinear filtering





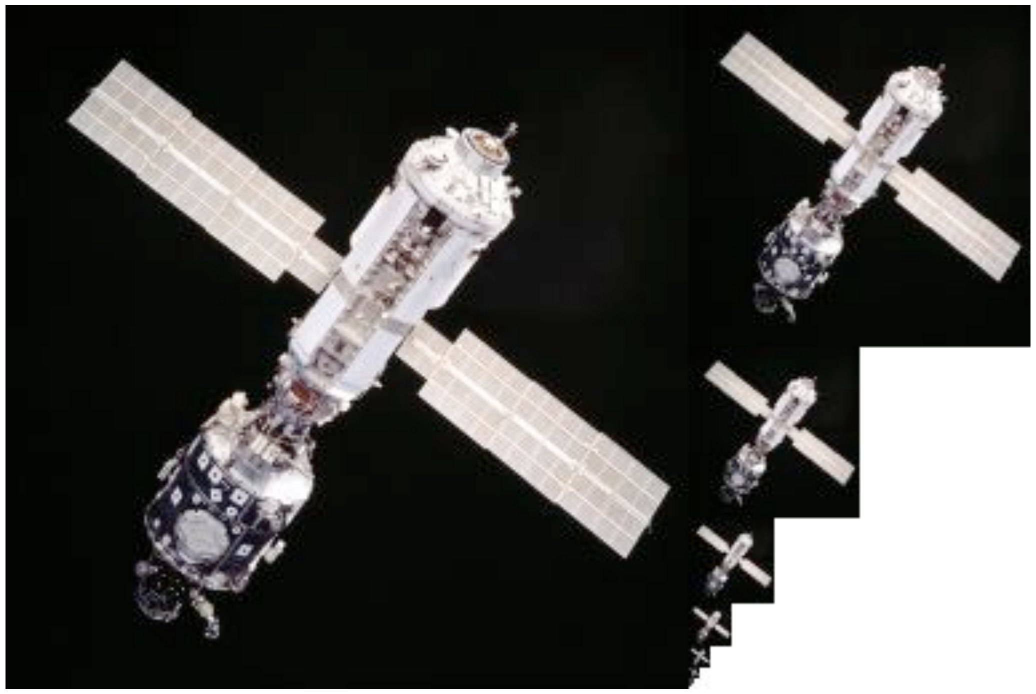
nearest neighbor **bilinear** **bicubic**

mitigate magnification artifacts

p = ?

smooths out the texture - no sharp boundaries

Mipmapping



Togikun, Wikimedia Commons

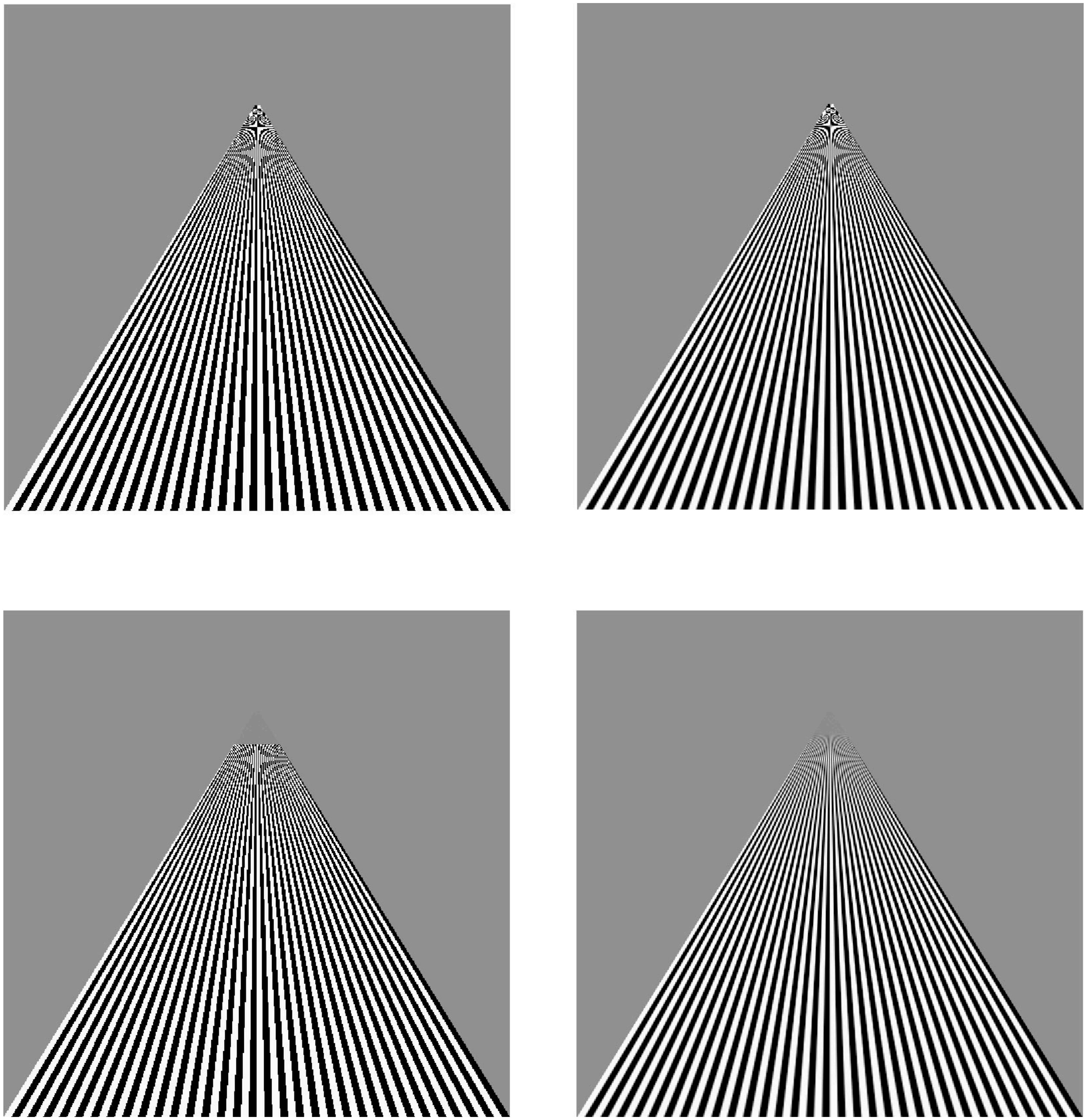
128×128, 64×64, 32×32, 16×16, 8×8, 4×4, 2×2, 1×1

Reduce minification artifacts

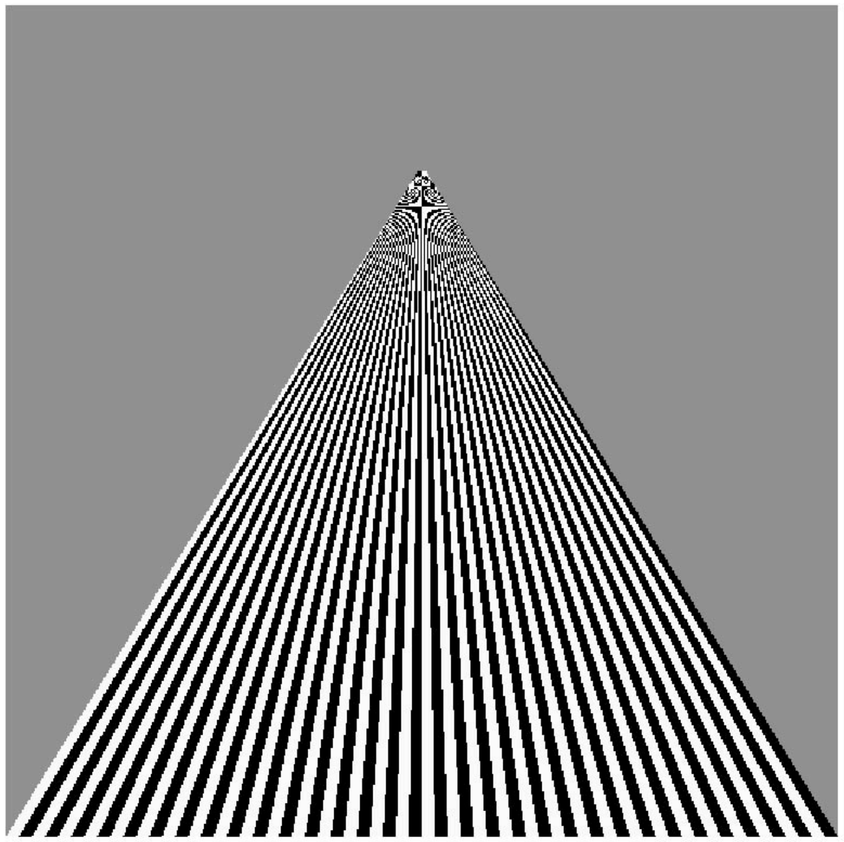
Prefilter the texture to obtain reduced resolutions

Requires 1/3 more space

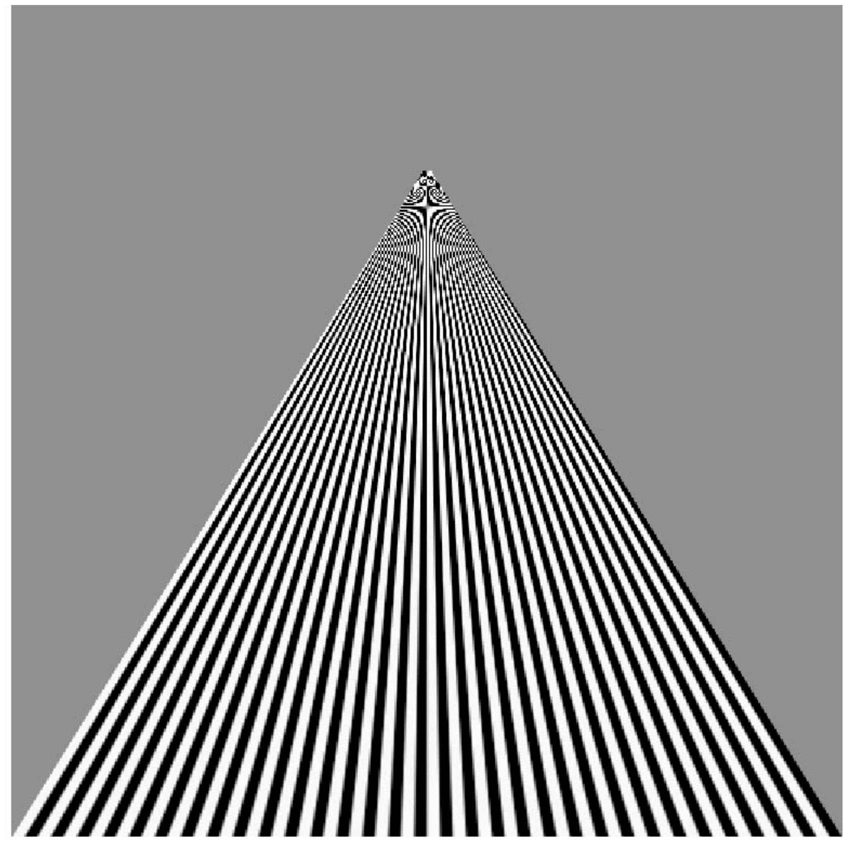
Get a texture hierarchy indexed by level



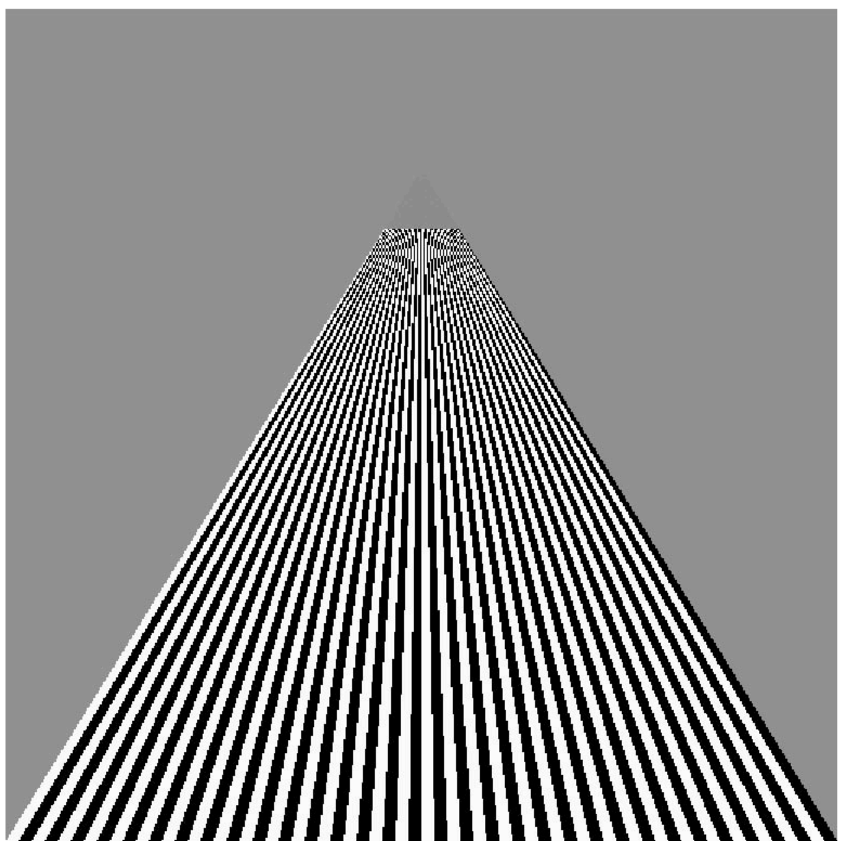
[Angel and Shreiner]



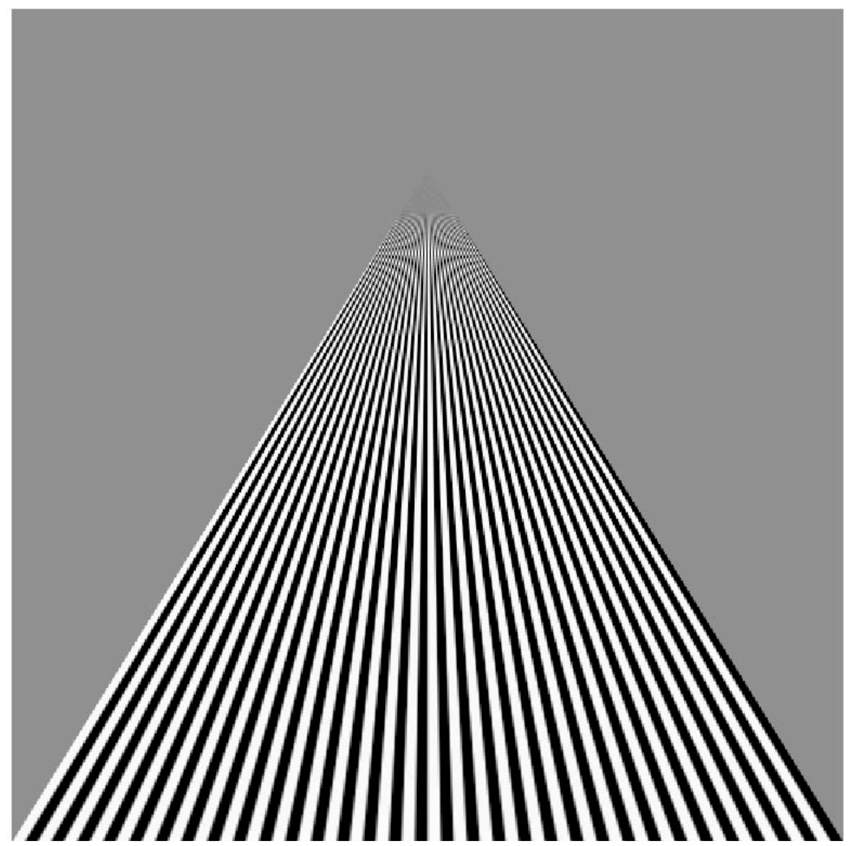
point sampling



linear filtering



mipmapped point sampling



mipmapped linear filtering

[Angel and Shreiner]