CS230 : Computer Graphics

Lecture 9: Texture Mapping

Tamar Shinar
Computer Science & Engineering
UC Riverside

Texture Mapping

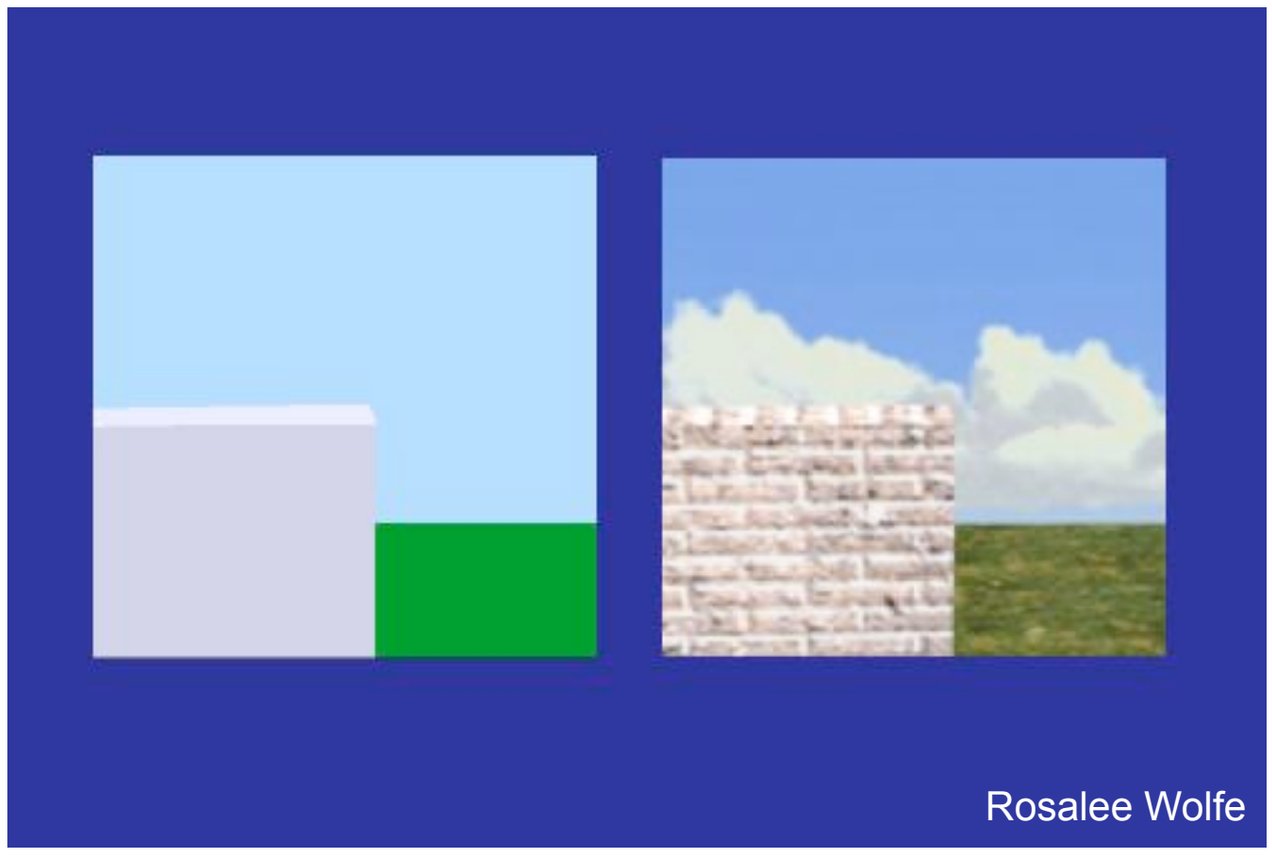
There are limits to geometric modeling

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

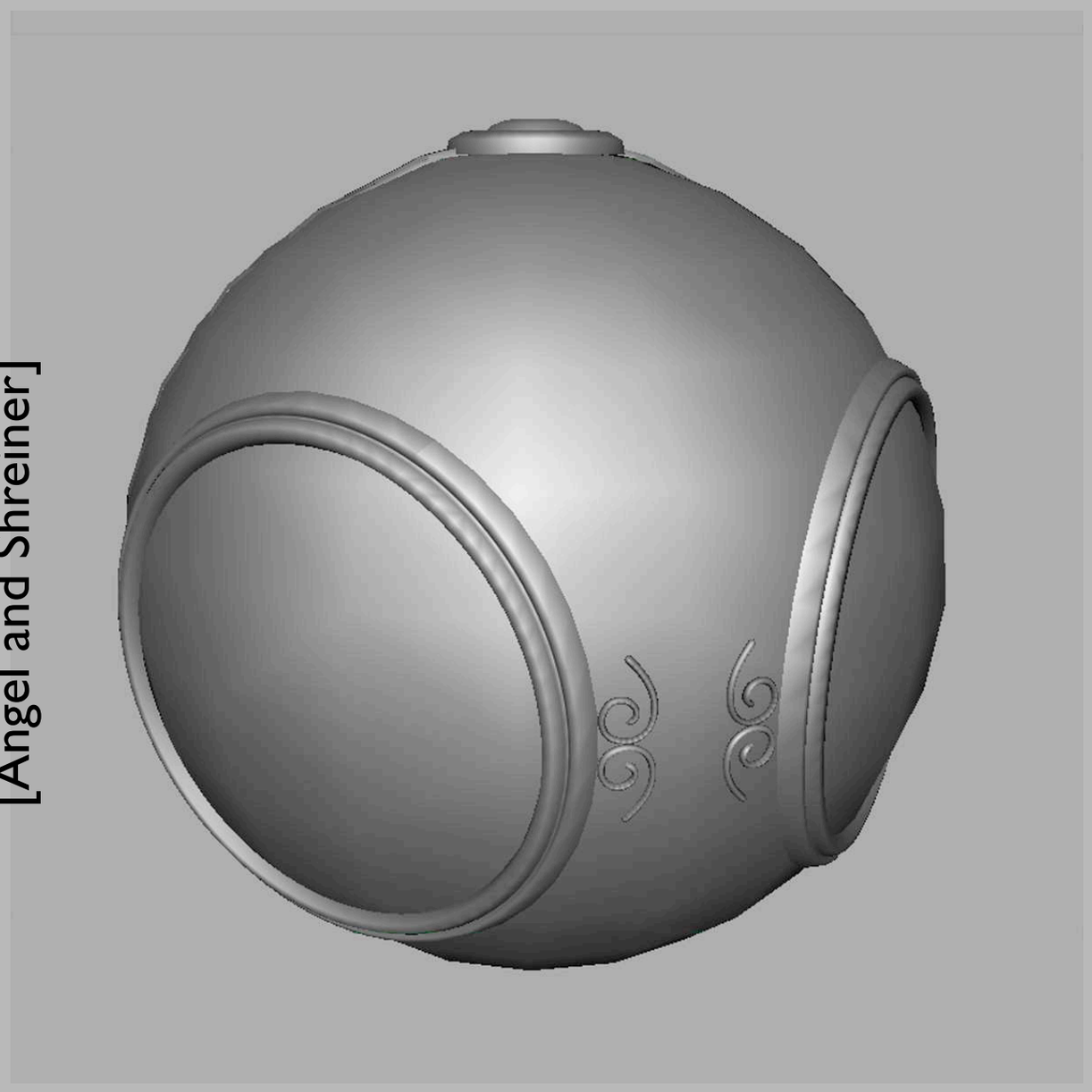


This image is just 8 polygons!

Add visual complexity.

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

[Angel and Shreiner]

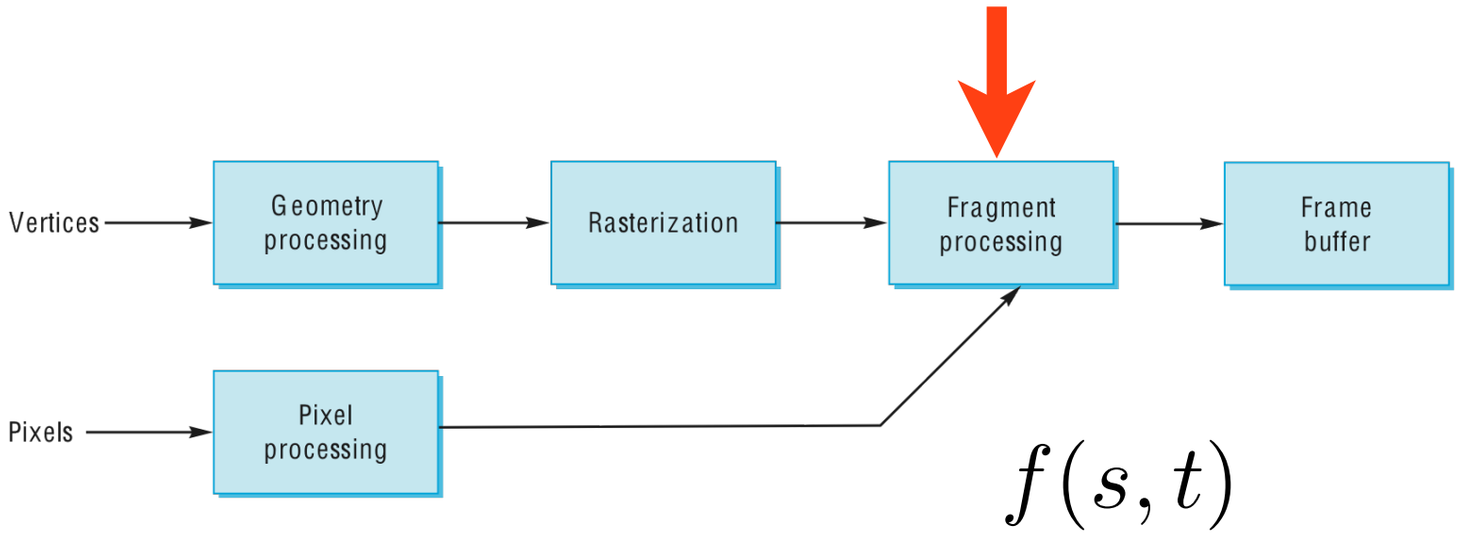


No texture

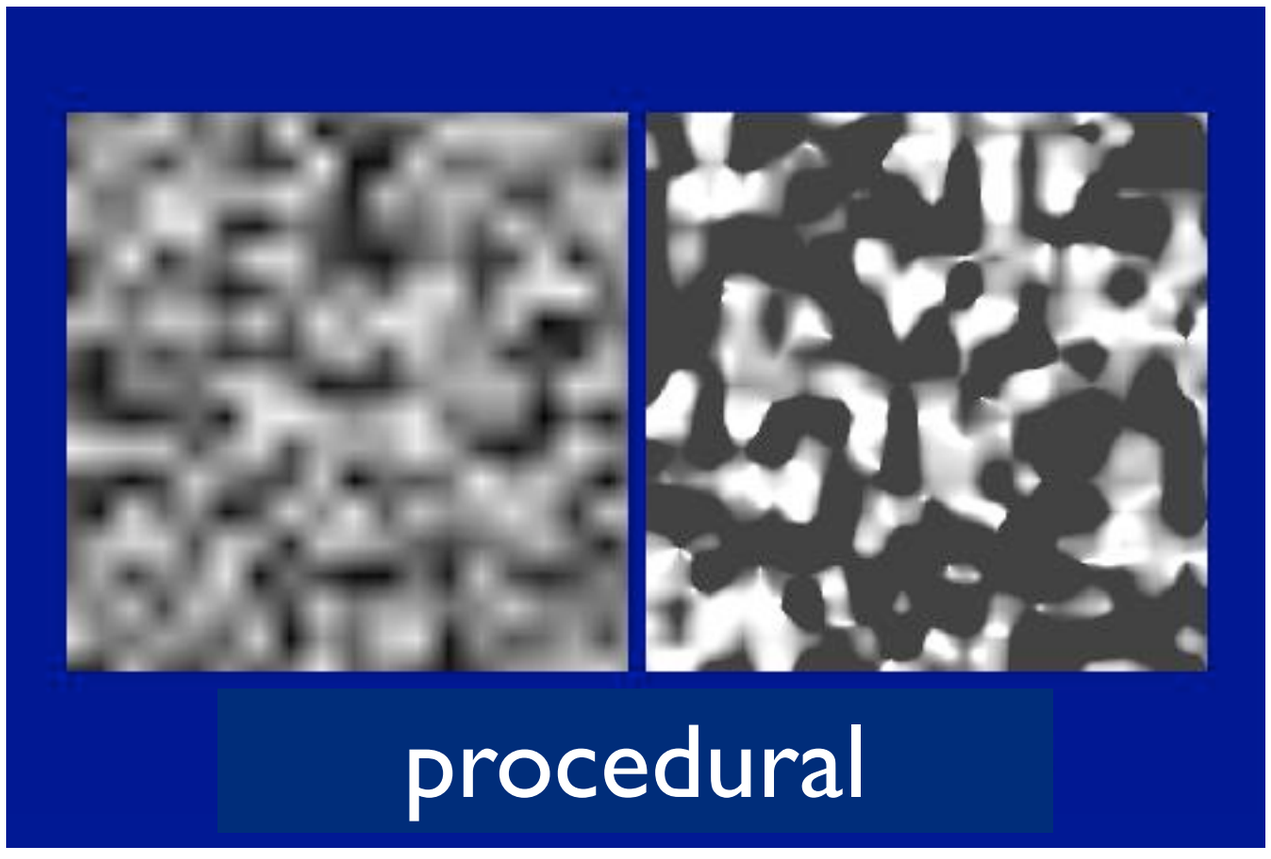
With texture

Pixar - Toy Story

Store 2D images in buffers and lookup pixel reflectances



$f(s,t)$



photo

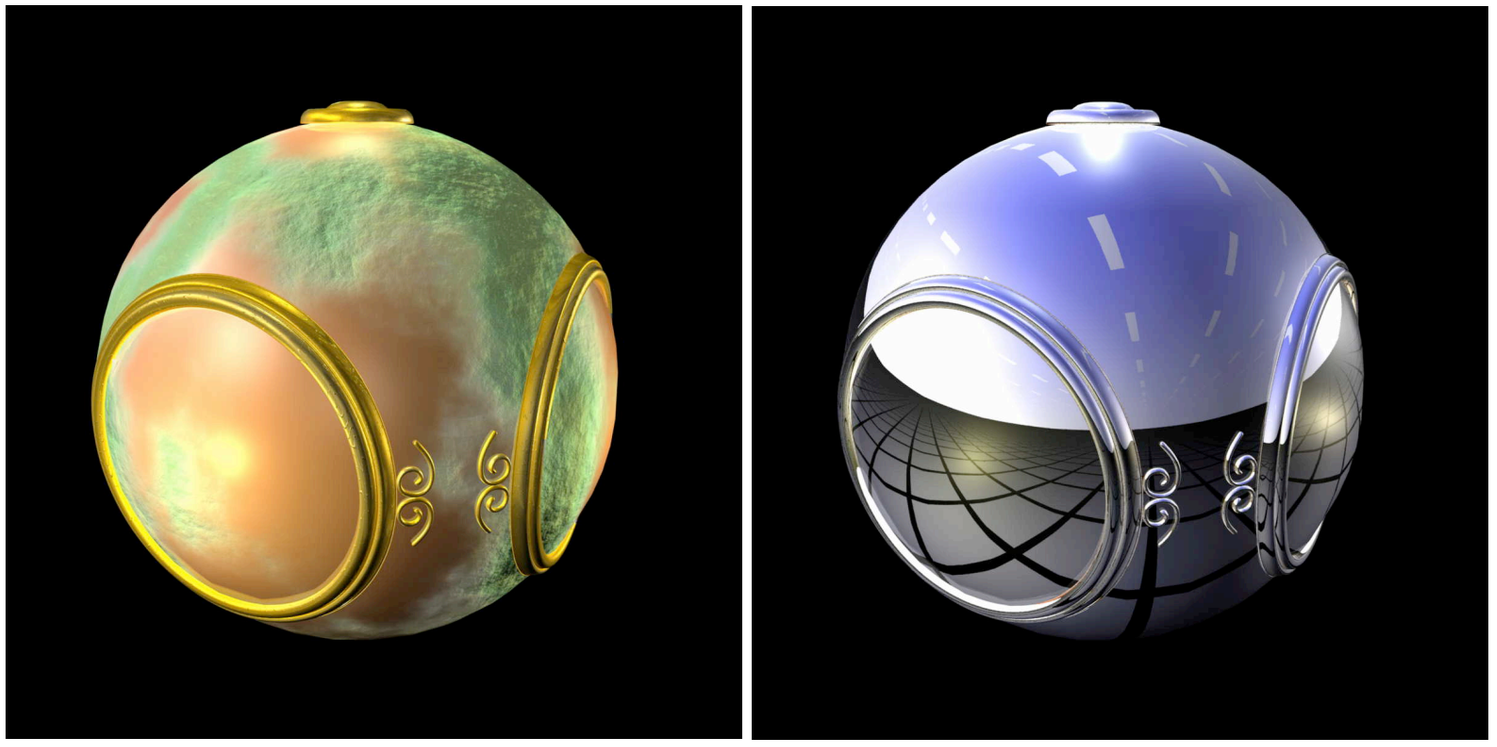
Textures can be anything that you can lookup values in -- photo, procedurally generated, or even a function that computes a value on the fly

3D solid textures

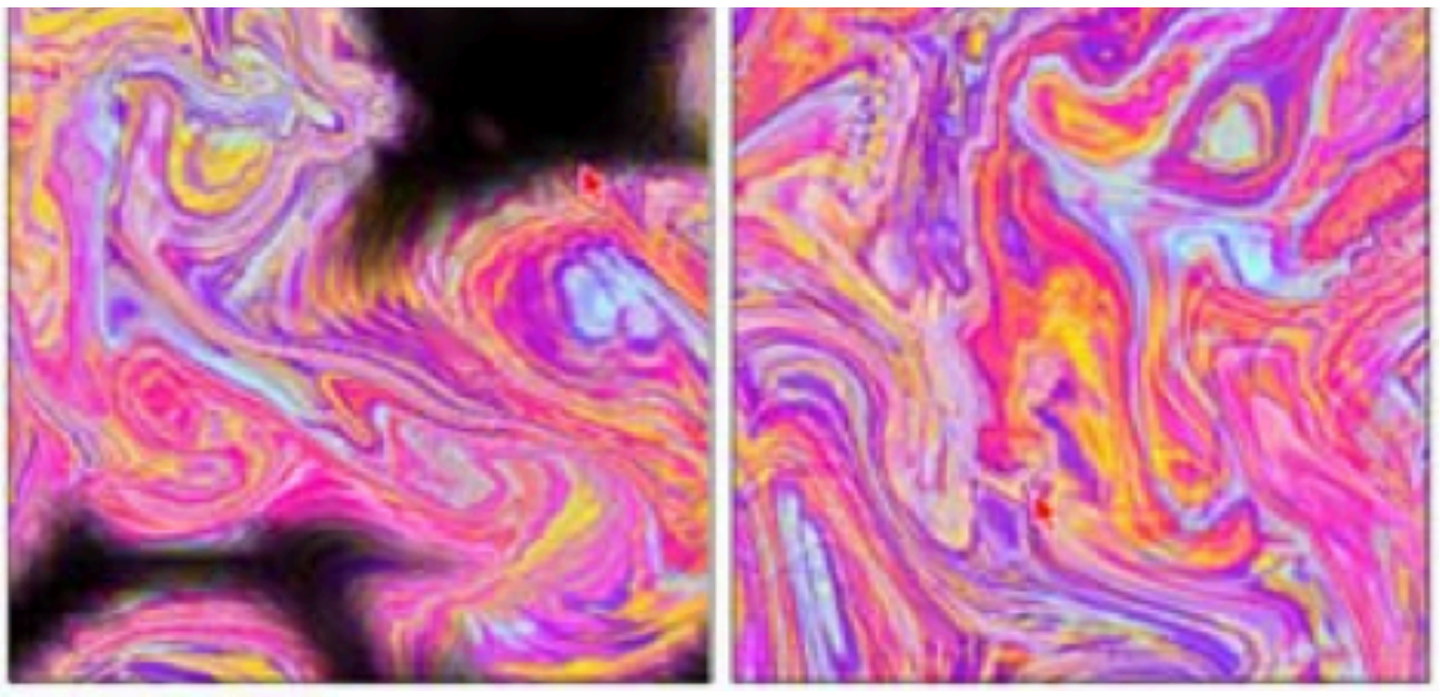
[Dong et al., 2008]

Other uses of textures...

Light maps
Shadow maps
Environment maps
Bump maps
Opacity maps
Animation

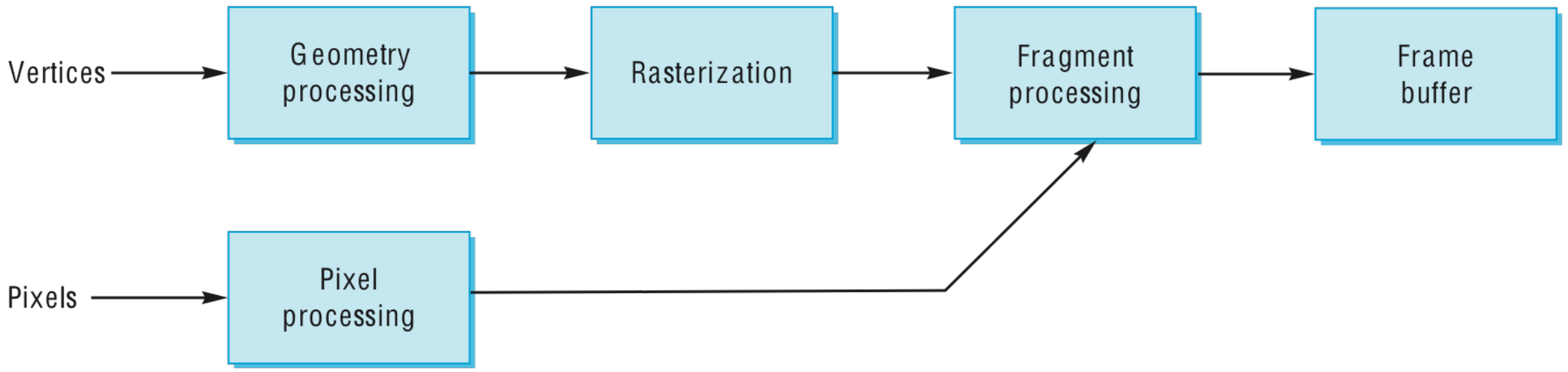


[Angel and Shreiner]



[Stam 99]

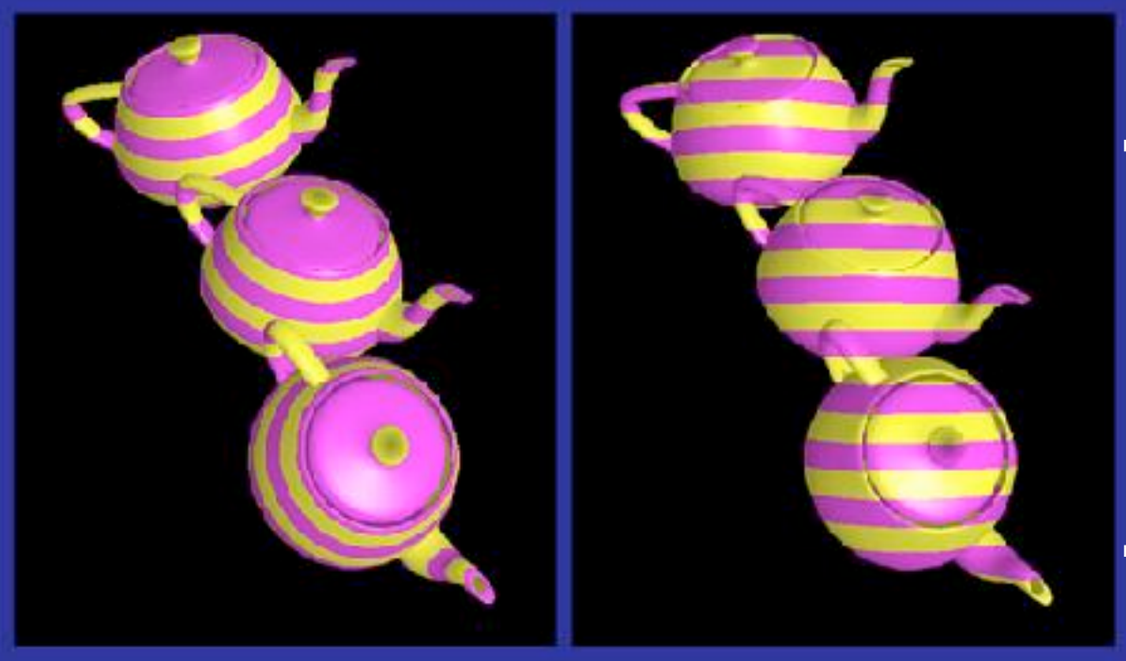
Texture mapping in the OpenGL pipeline



- 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

The major issues in texture mapping...

- What should the actual mapping be?



[Rosalee Wolfe]

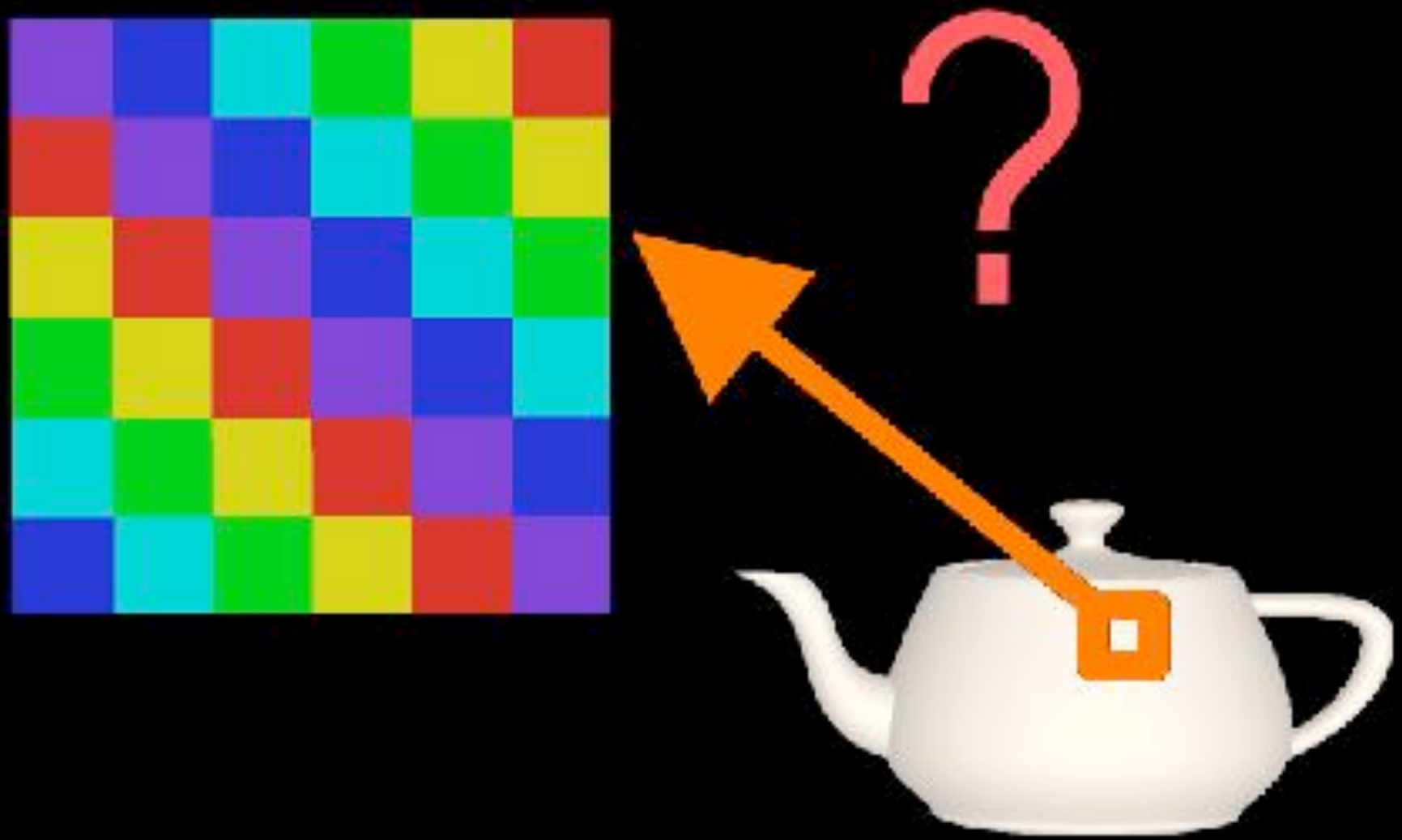
easy: rectangular surface

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

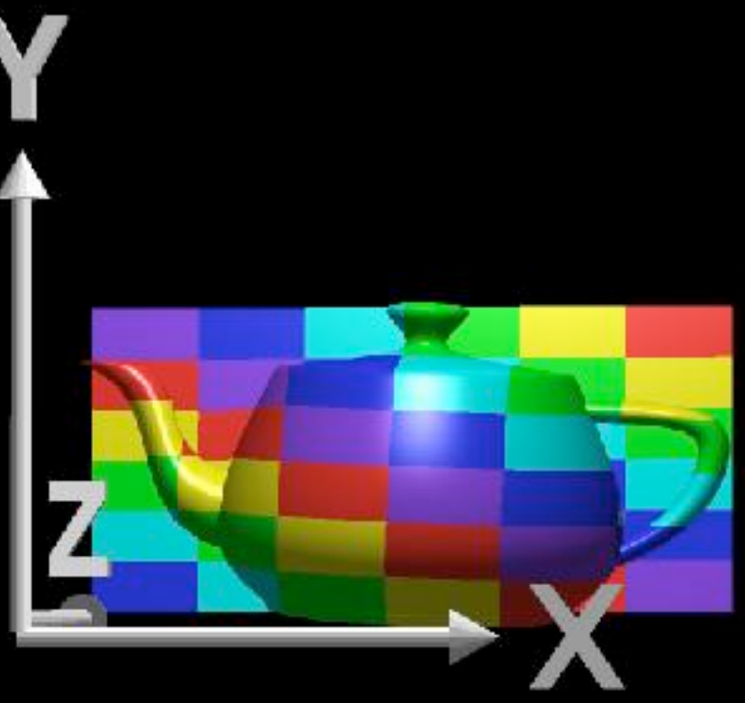
want a nice map that doesn't look distorted

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

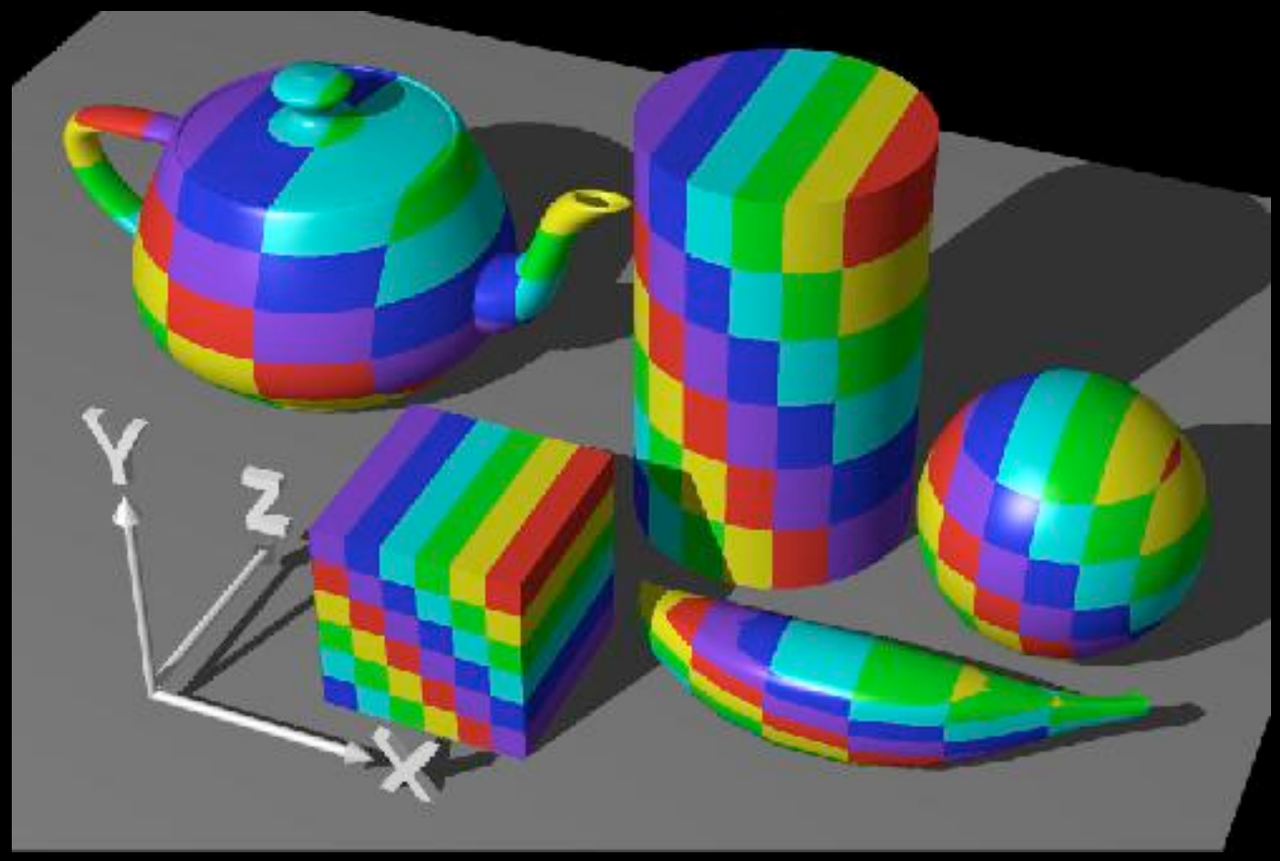


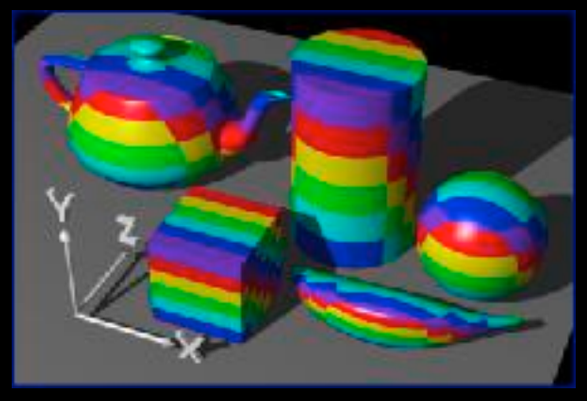
[Rosalee Wolfe]

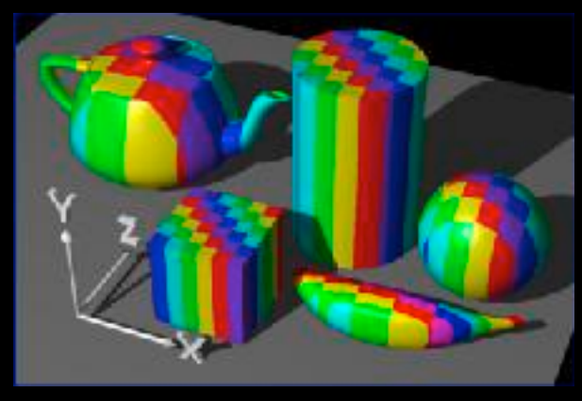
Example: planar mapping



[Rosalee Wolfe]

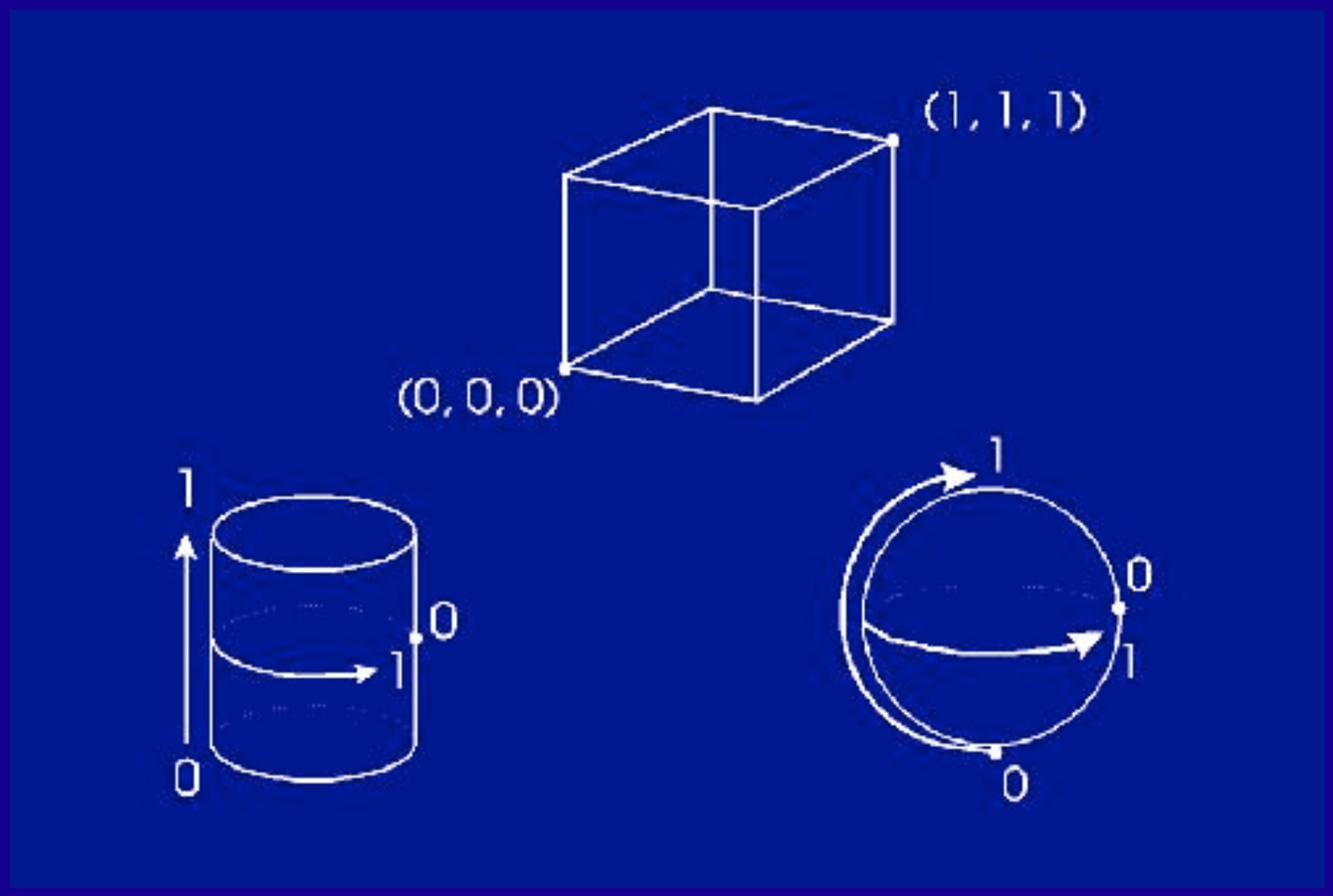






Intermediate surfaces

First map the texture to a simpler, intermediate surface

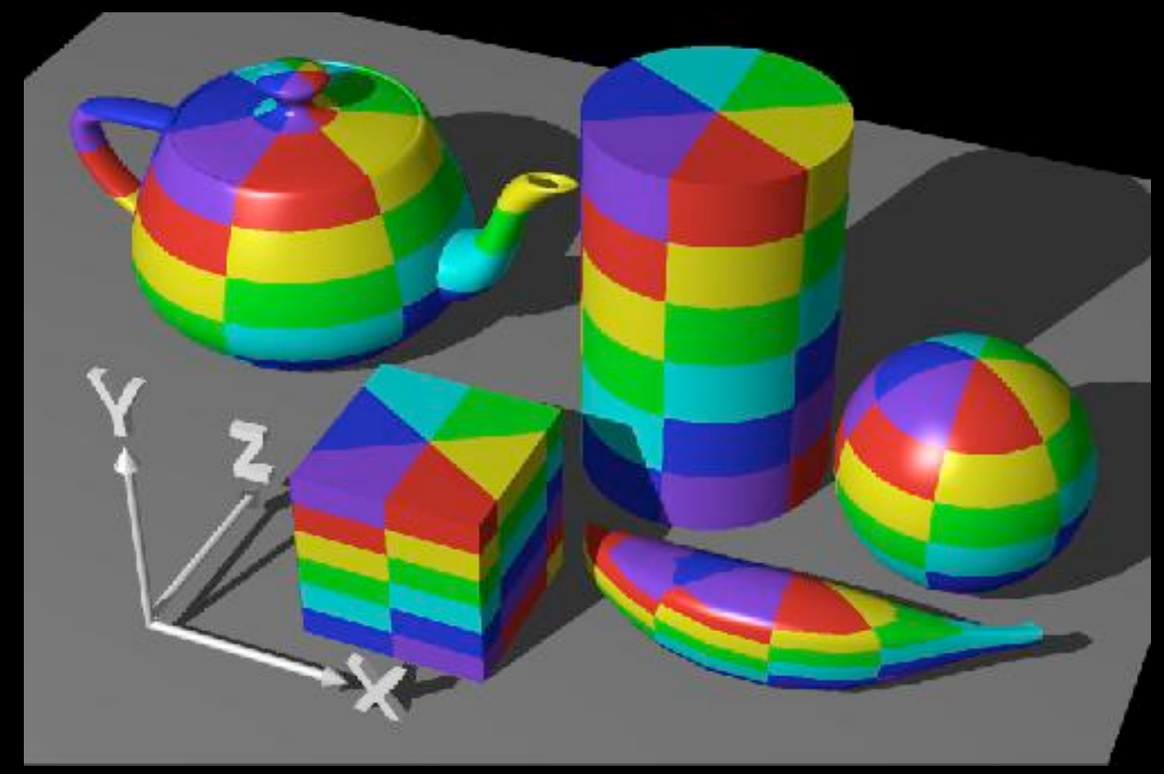


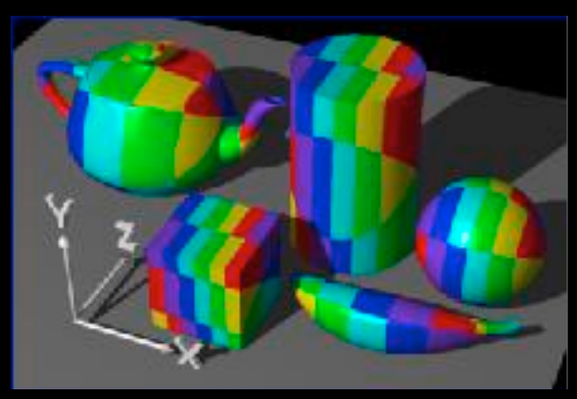
[Rosalee Wolfe]

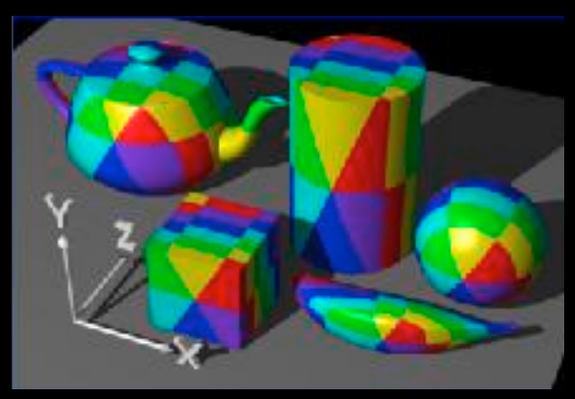
Cylindrical mapping

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

[Rosalee Wolfe]



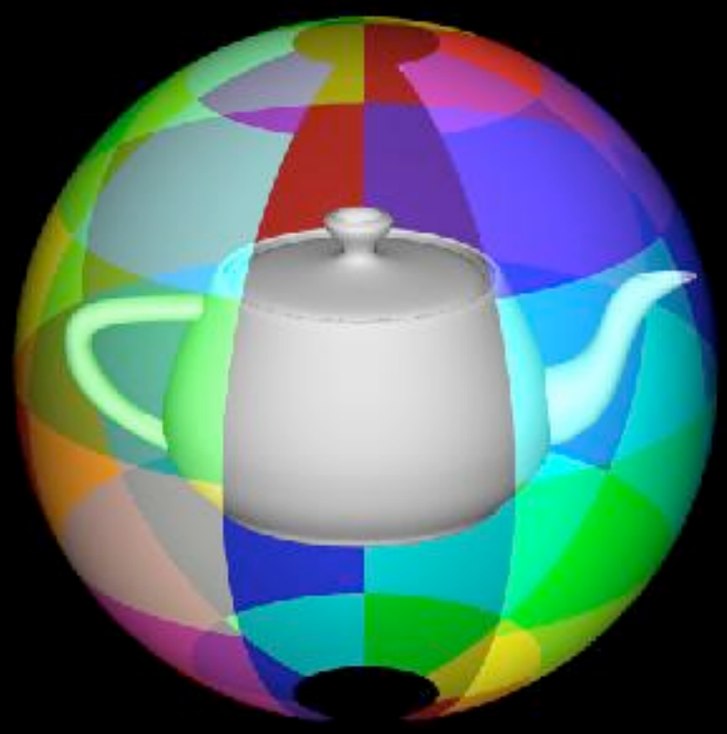




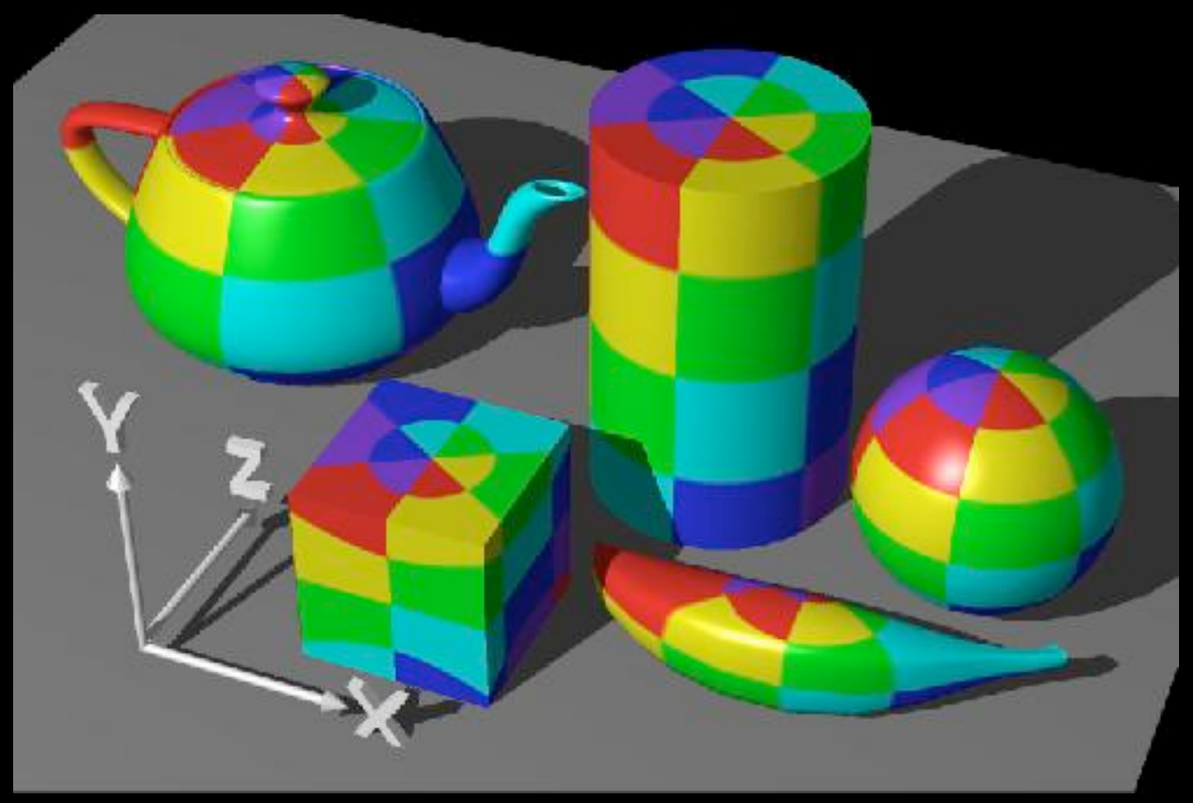
- note “pie slice” phenomena
- which coordinate axis is parallel to the cylinder axis?

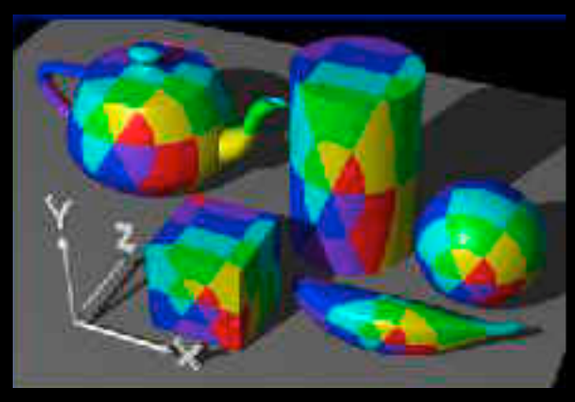
Spherical Mapping

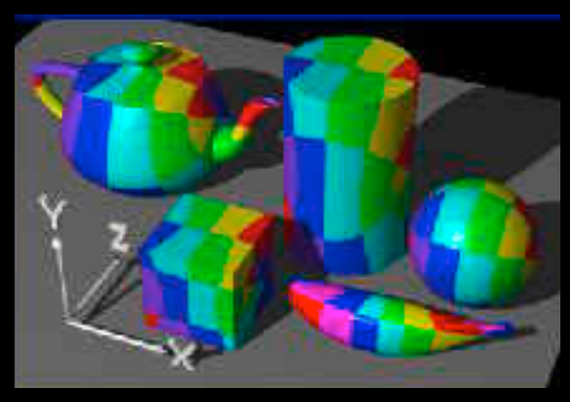
(x,y,z) -> (latitude,longitude)
-> (u,v)



[Rosalee Wolfe]

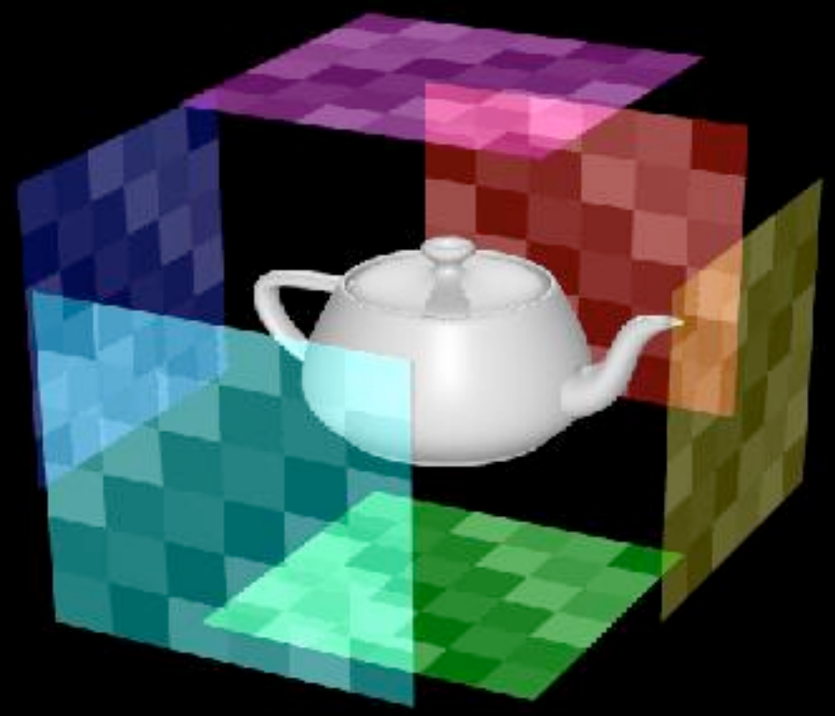




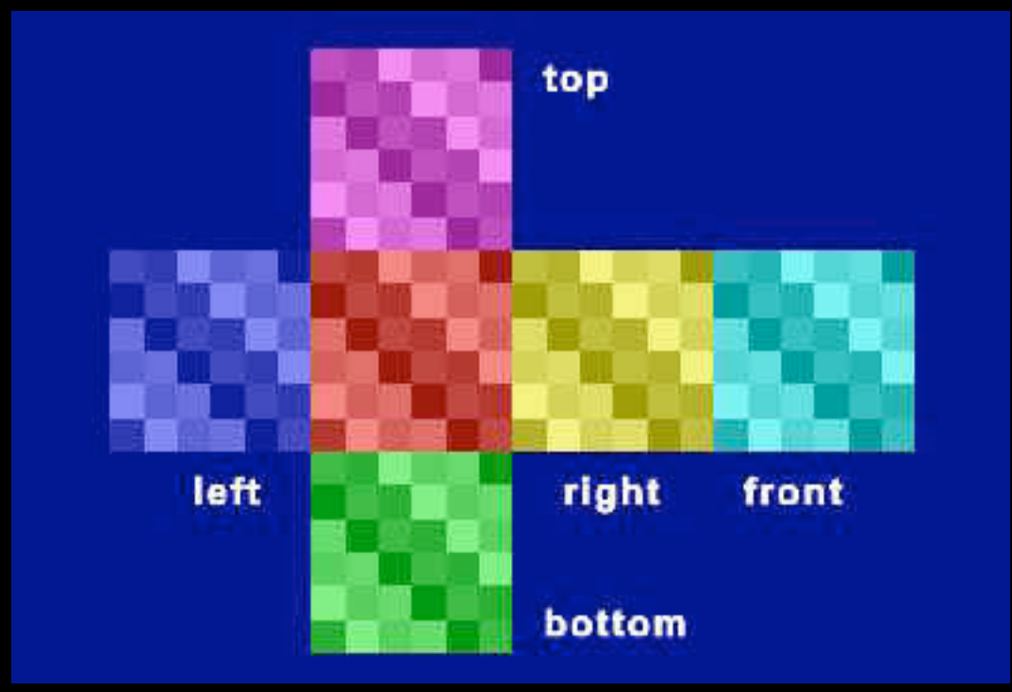


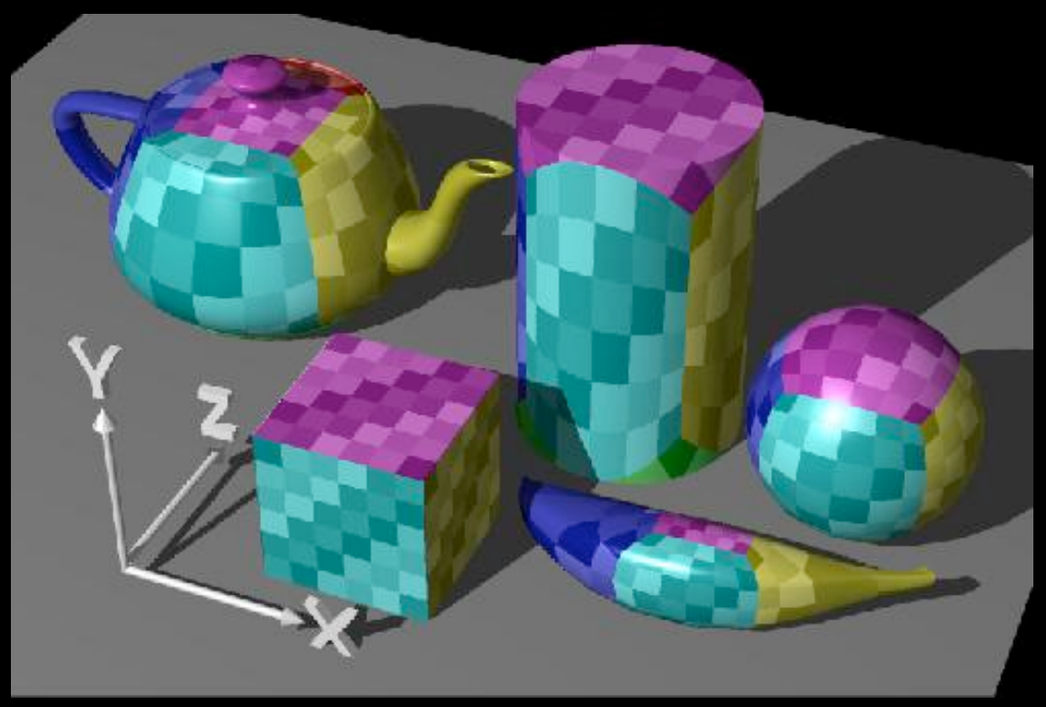
spherical map stretches squares at equator and squeezes squares at poles

Box Mapping



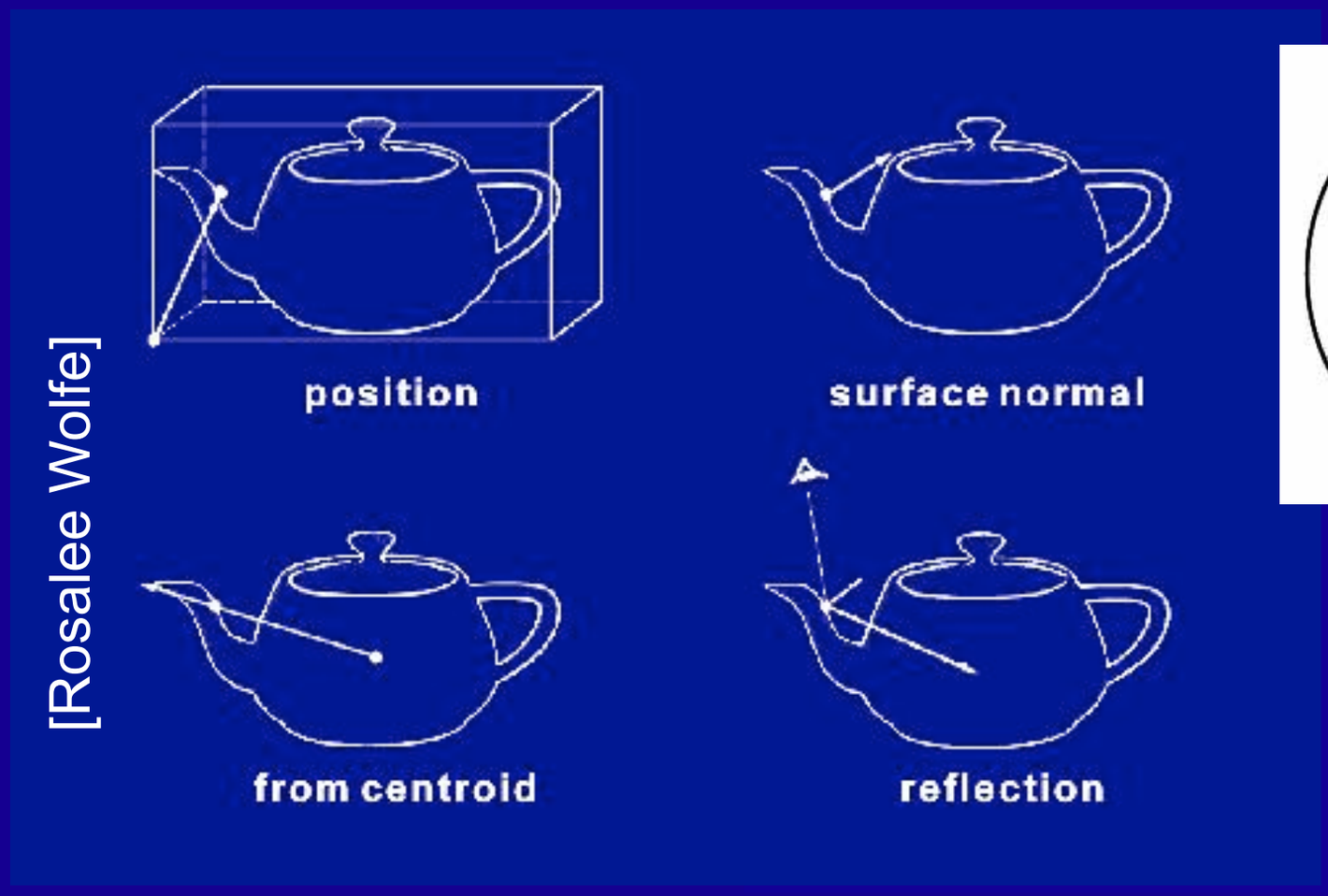
[Rosalee Wolfe]



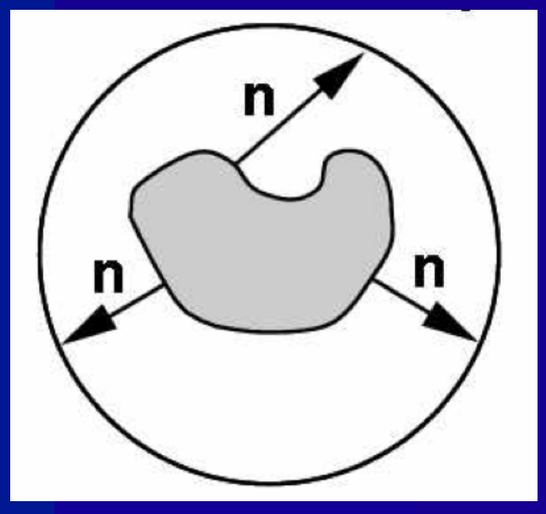


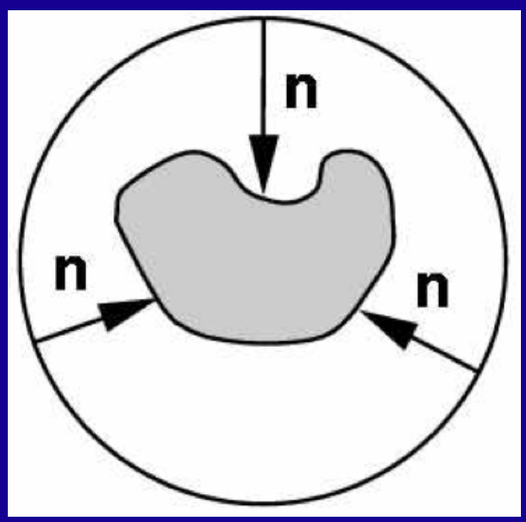
- similar to planar mapping
- planar projection -- choose which plane to project onto

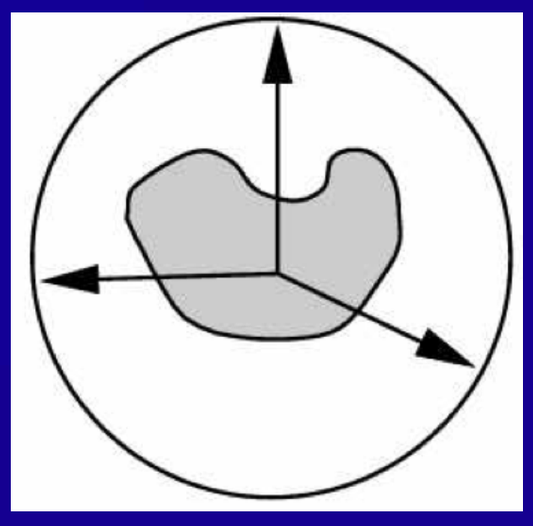
How do we map between intermediate and actual objects?



[Rosalee Wolfe]



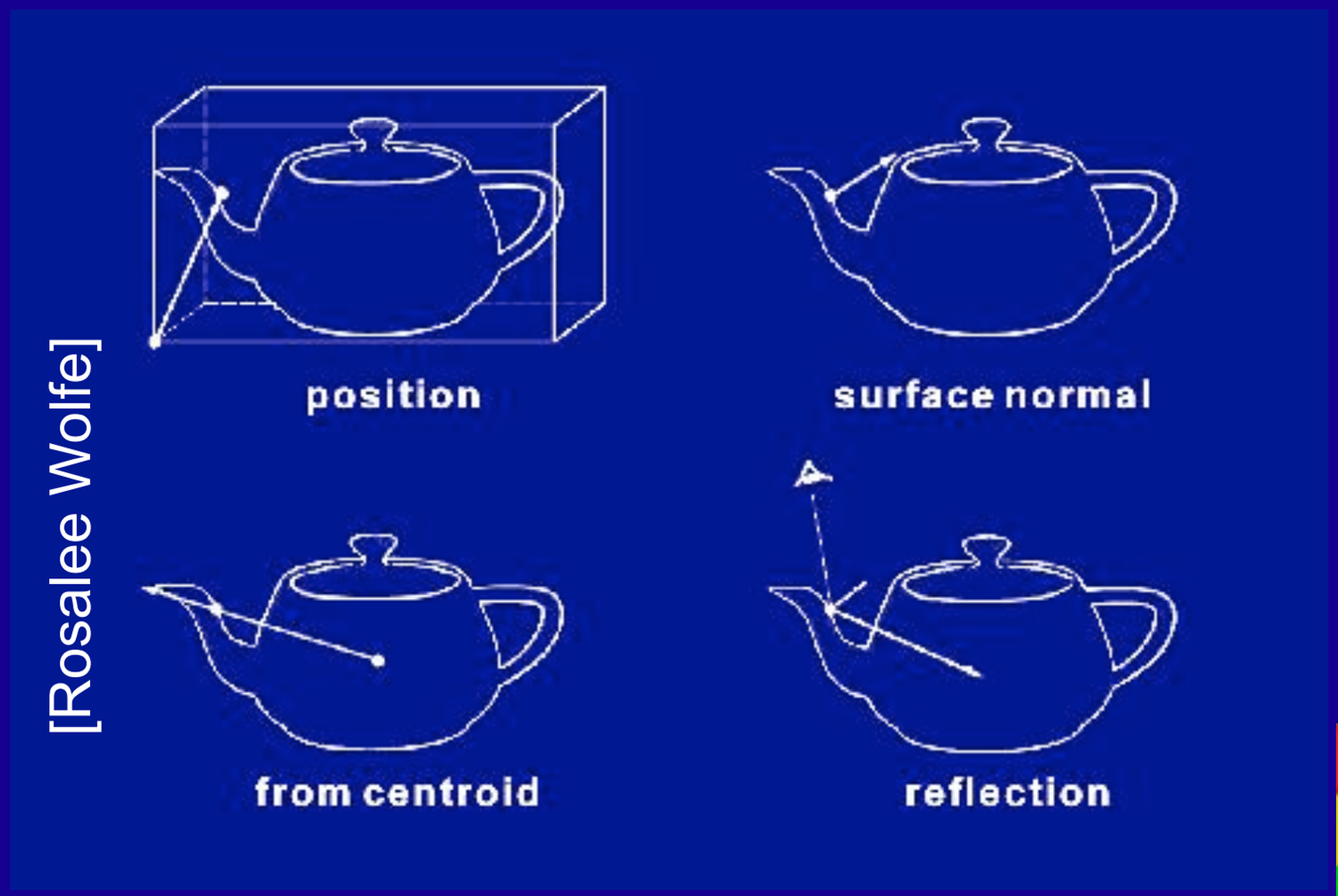


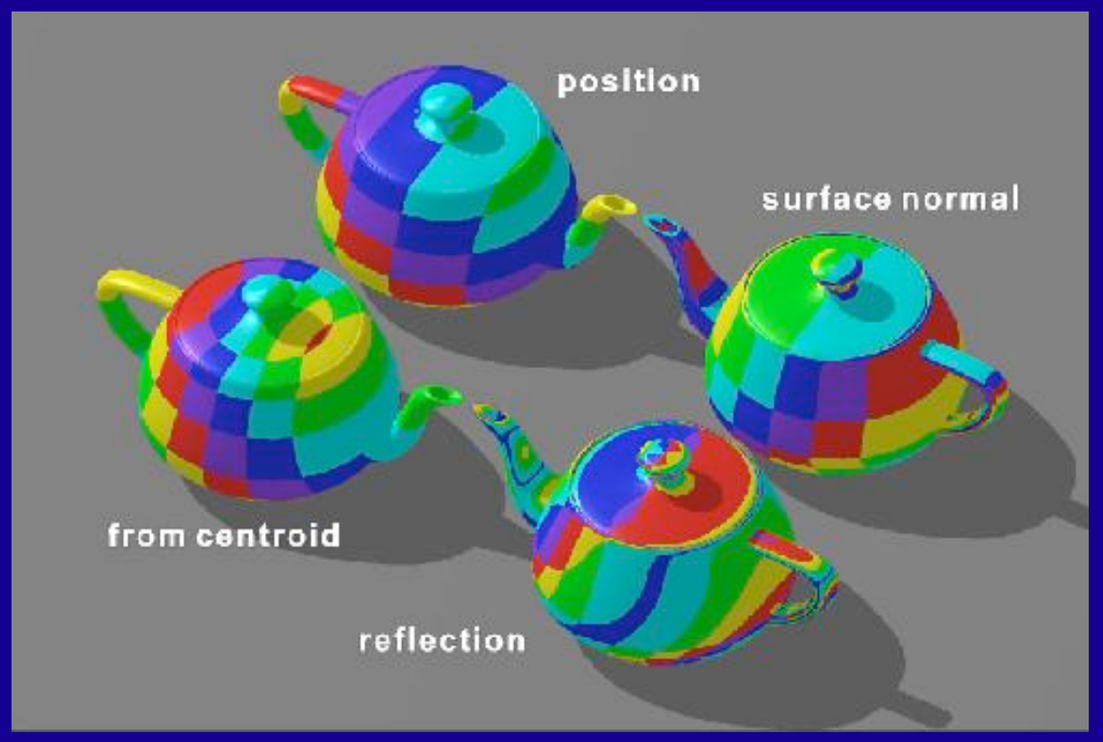


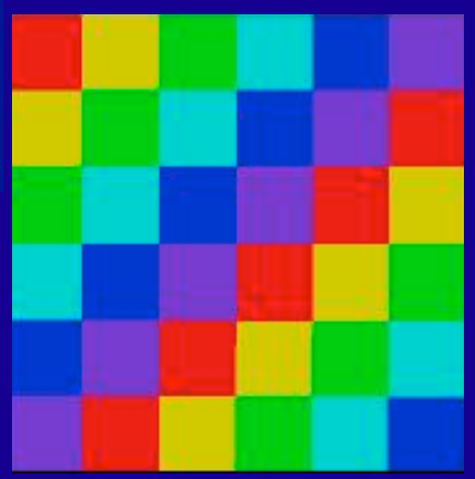
We associated (x,y,z) on the intermediate object with the texture (u,v). But which point on the actual object is this?
We choose both the **intermediate shape** and the **mapping from the actual shape to the intermediate shape**

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?

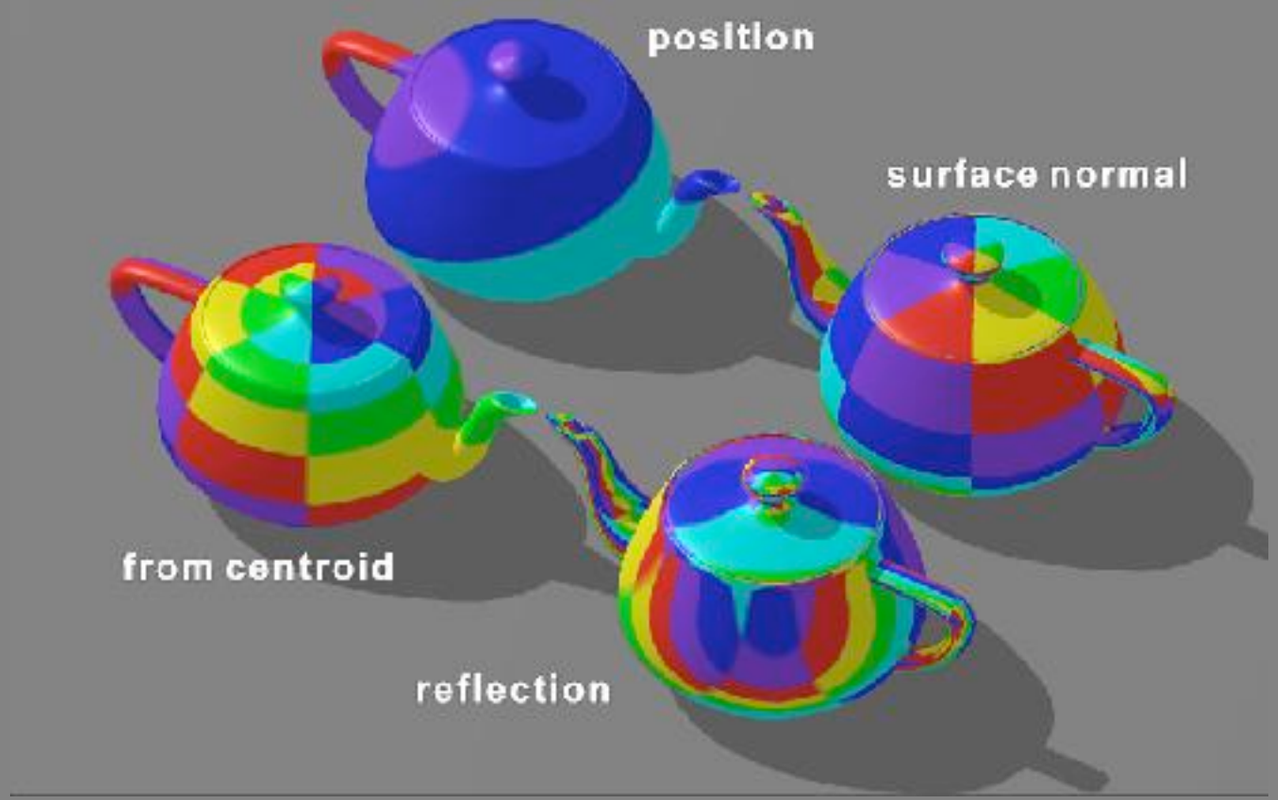


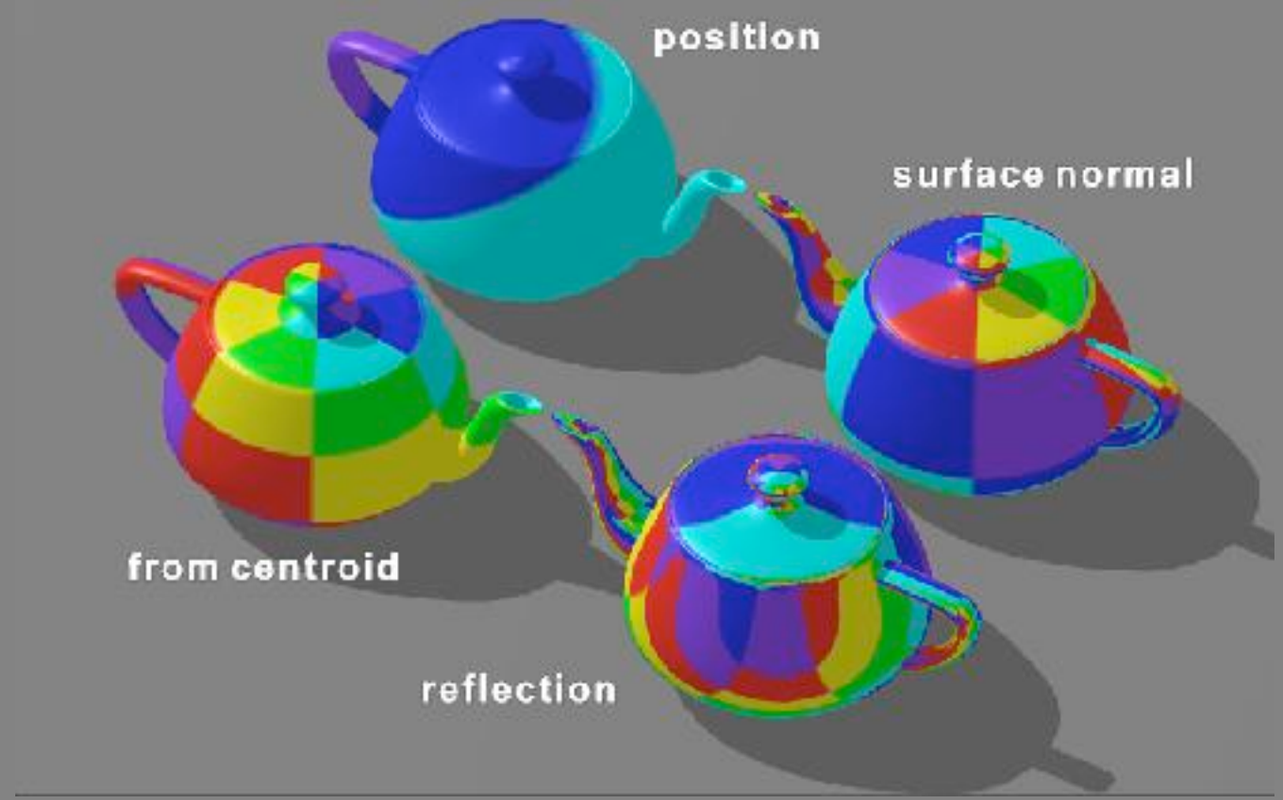


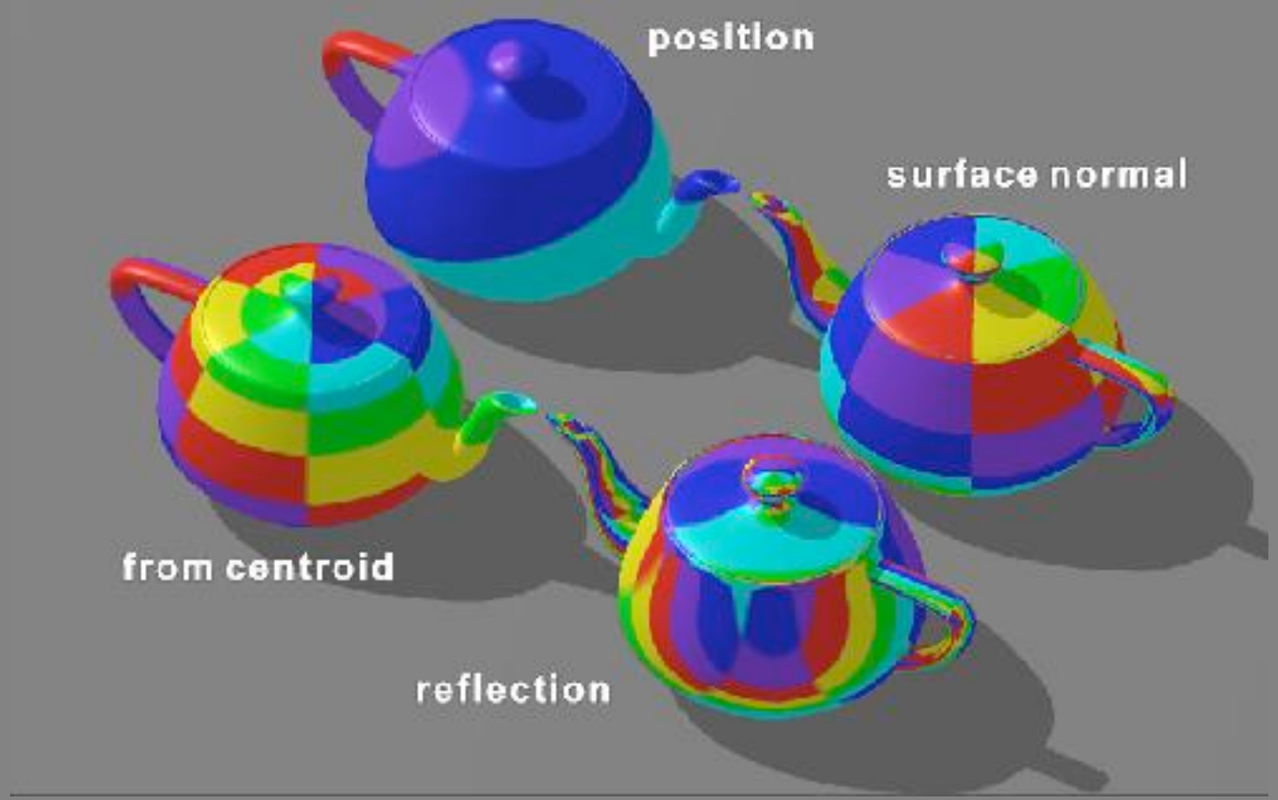


What intermediate shape was used here?

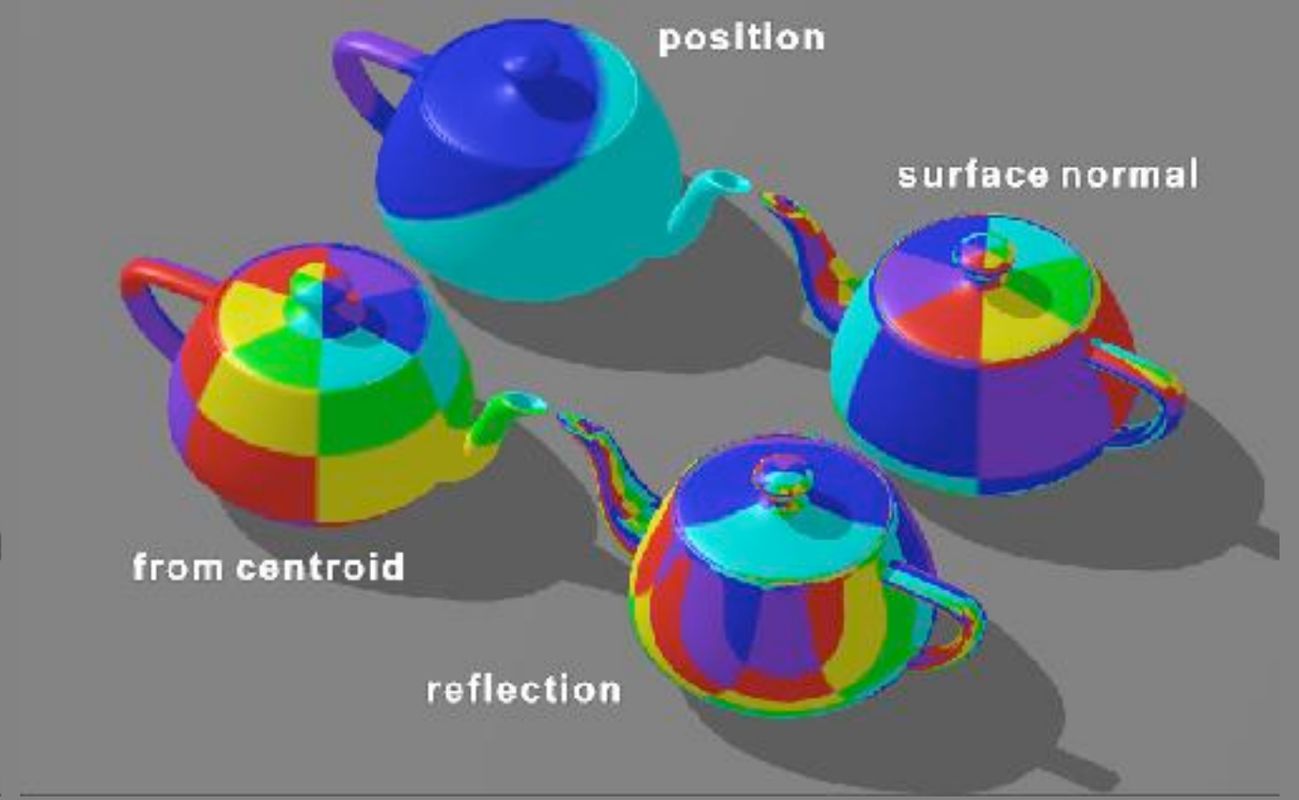
Can you tell what intermediate shape was used?
Planar map - in xy plane


position
surface normal
from centroid
reflection


position
surface normal
from centroid
reflection

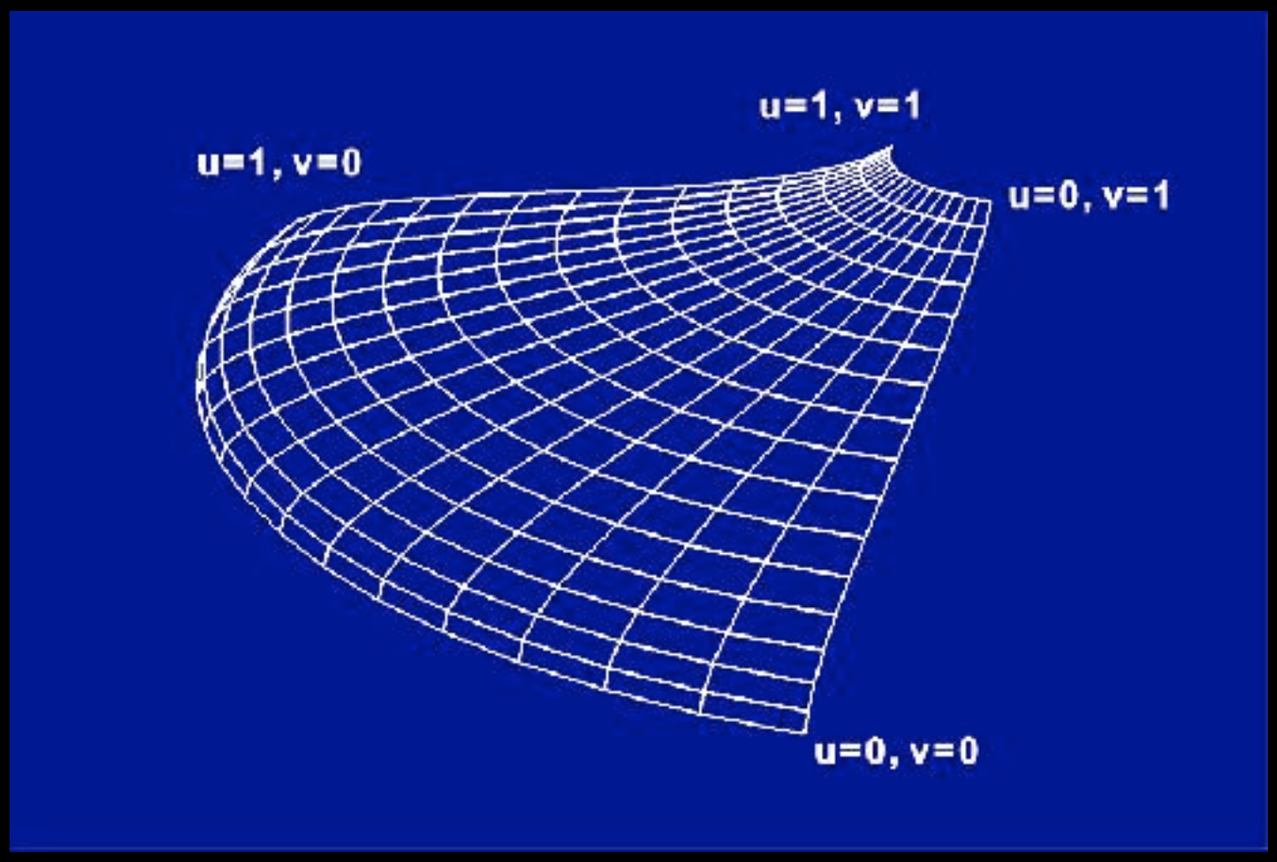


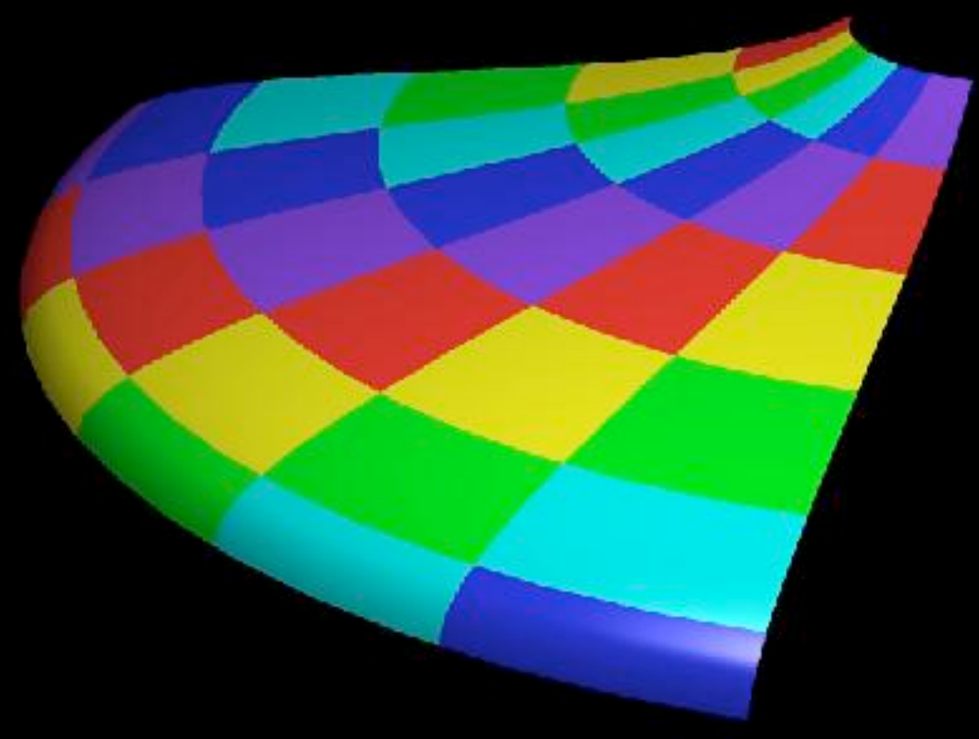
Cylindrical



Spherical

Parametric Surfaces





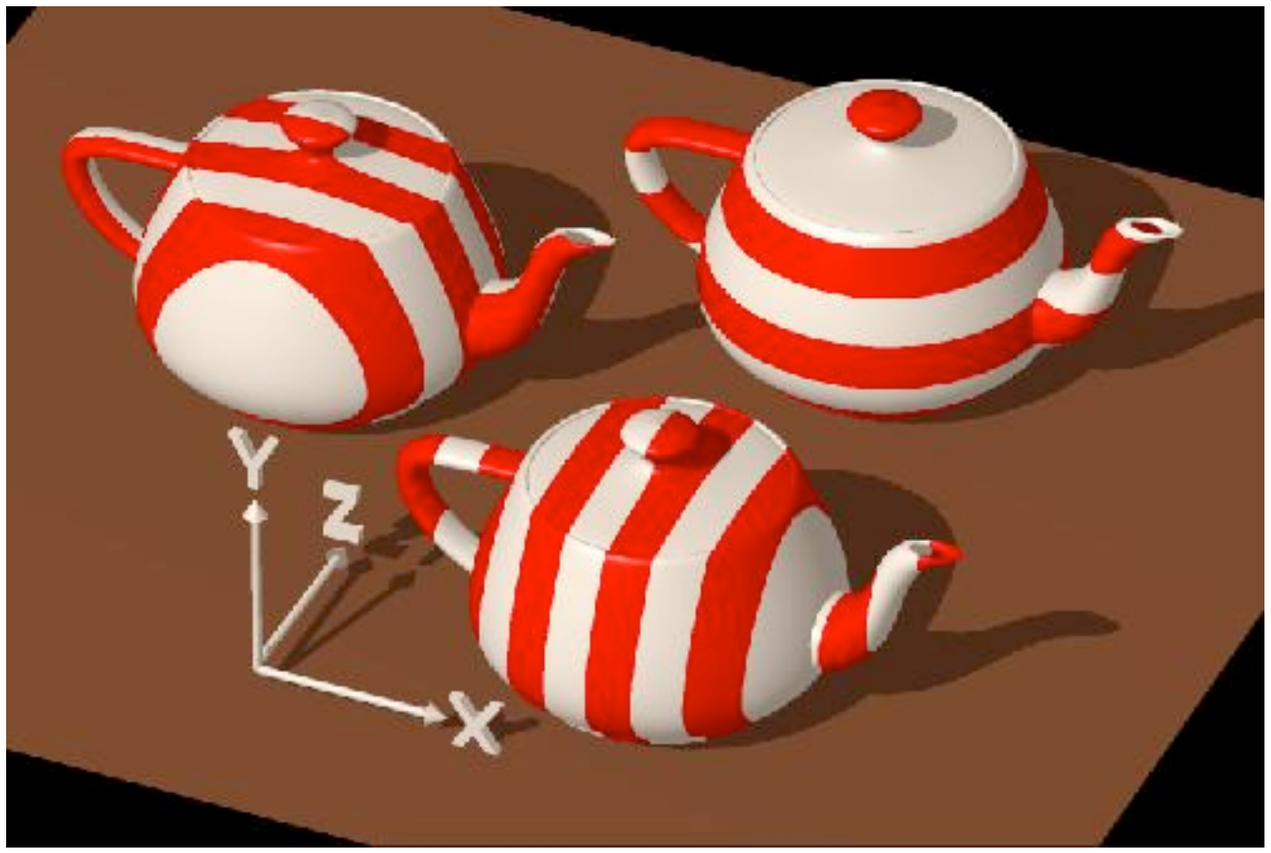
32 parametric patches

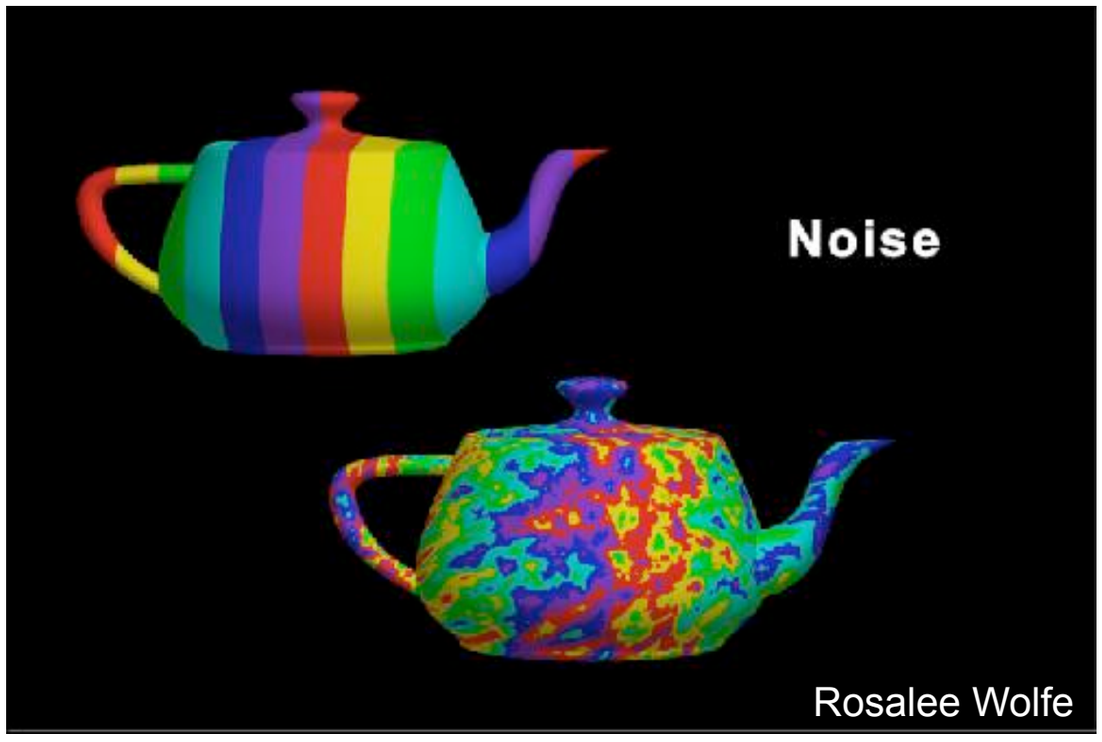
3D solid textures

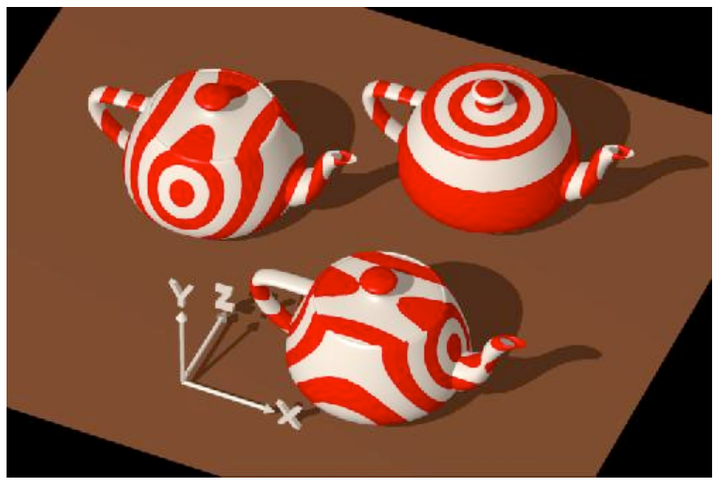
[Dong et al., 2008]

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

Procedural textures



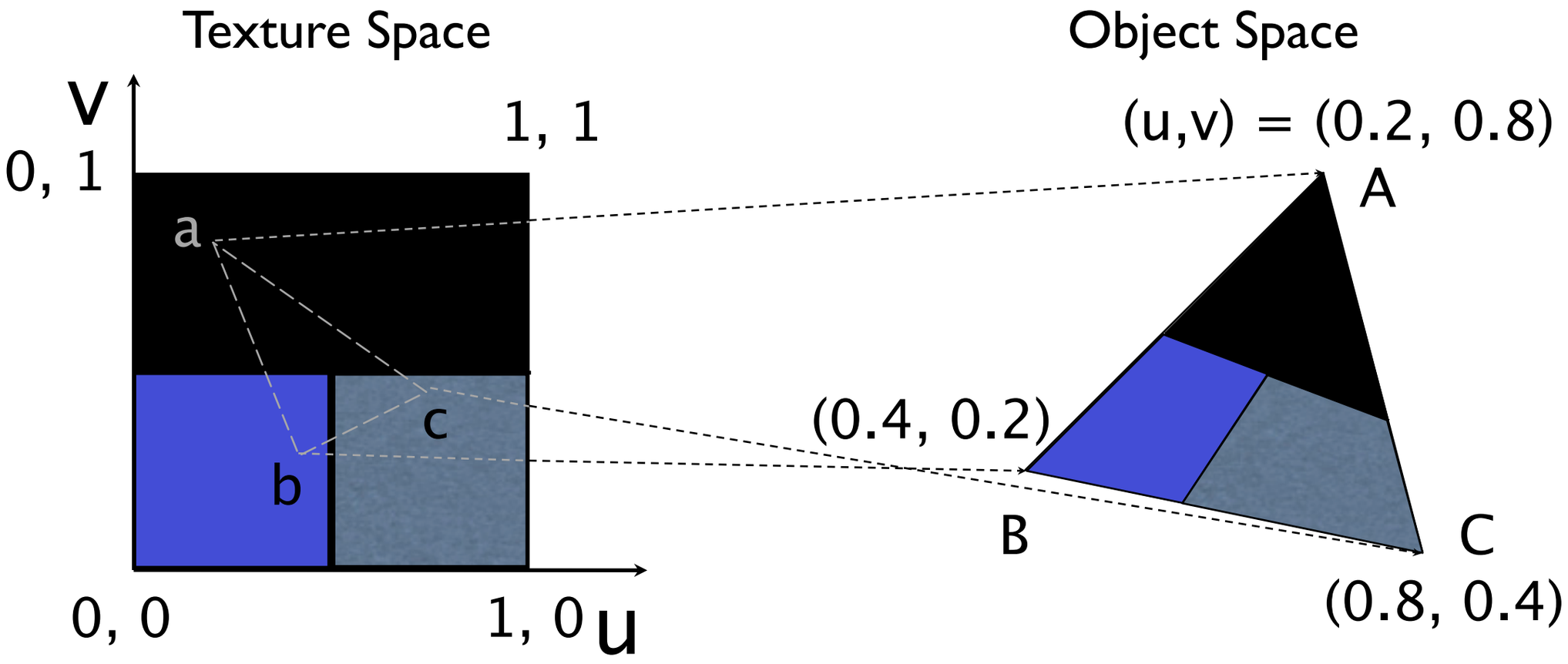




e.g., Perlin noise

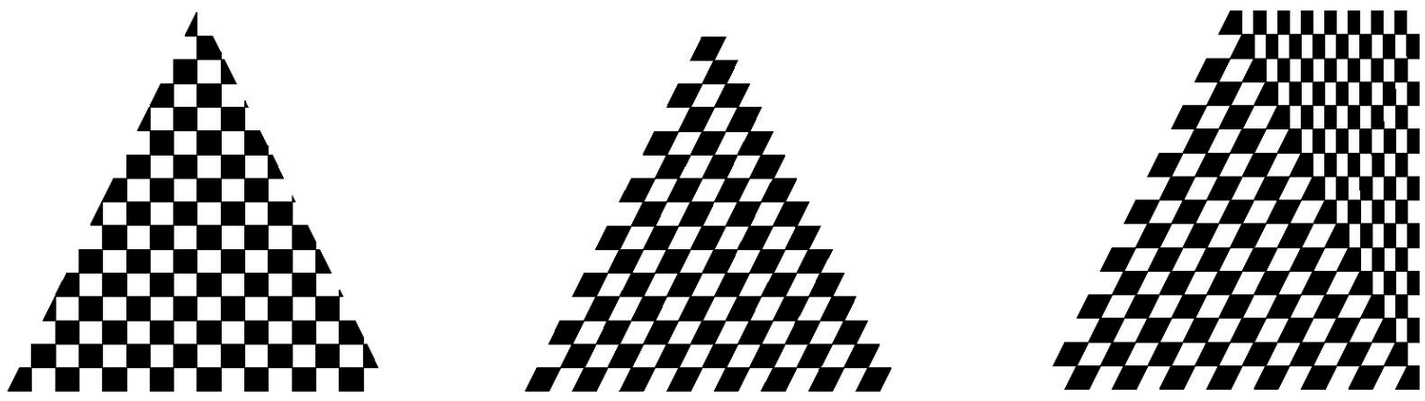
Triangles

Texturing triangles



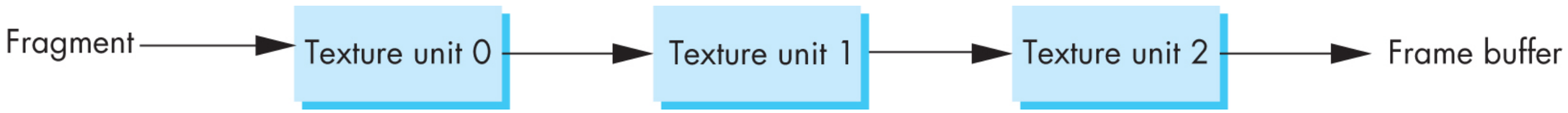
[Angel and Shreiner]

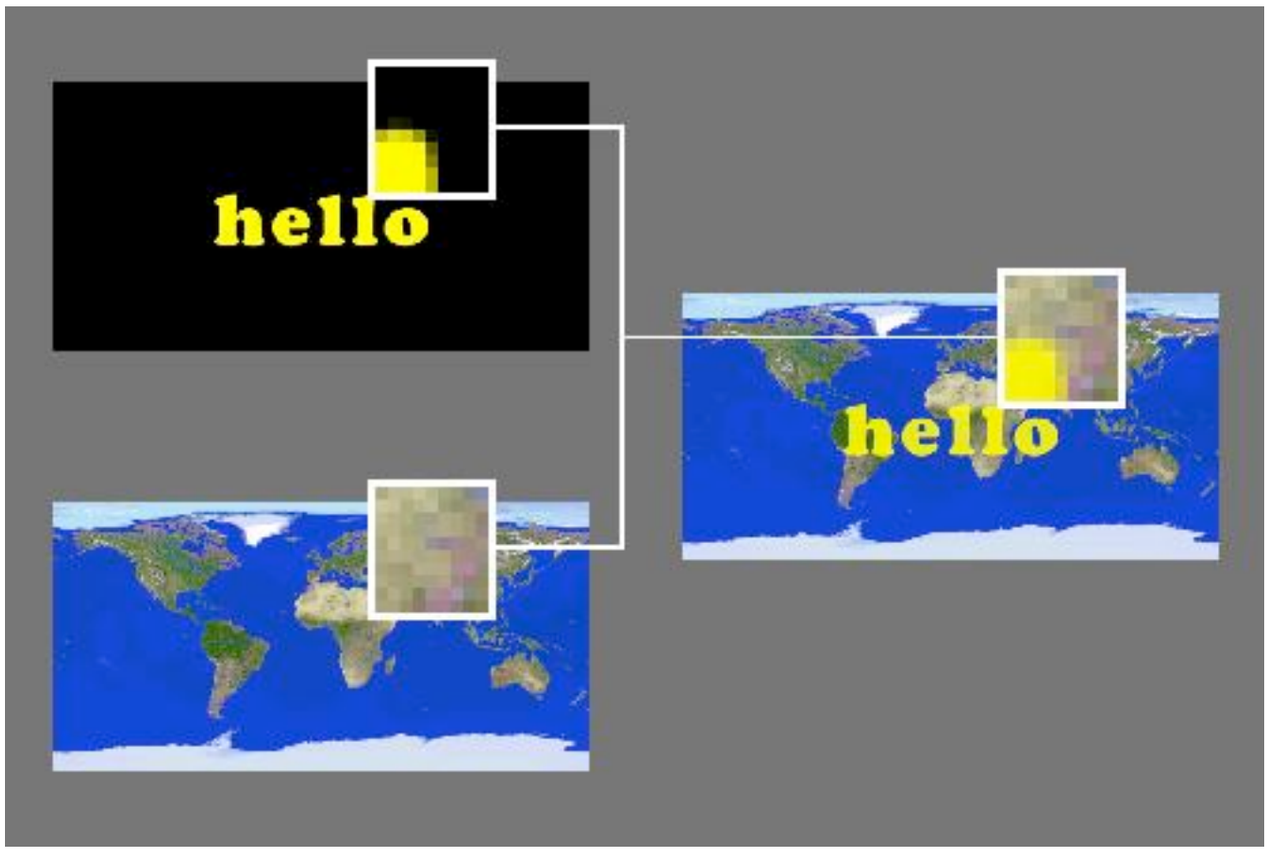
`glTexCoord*()`

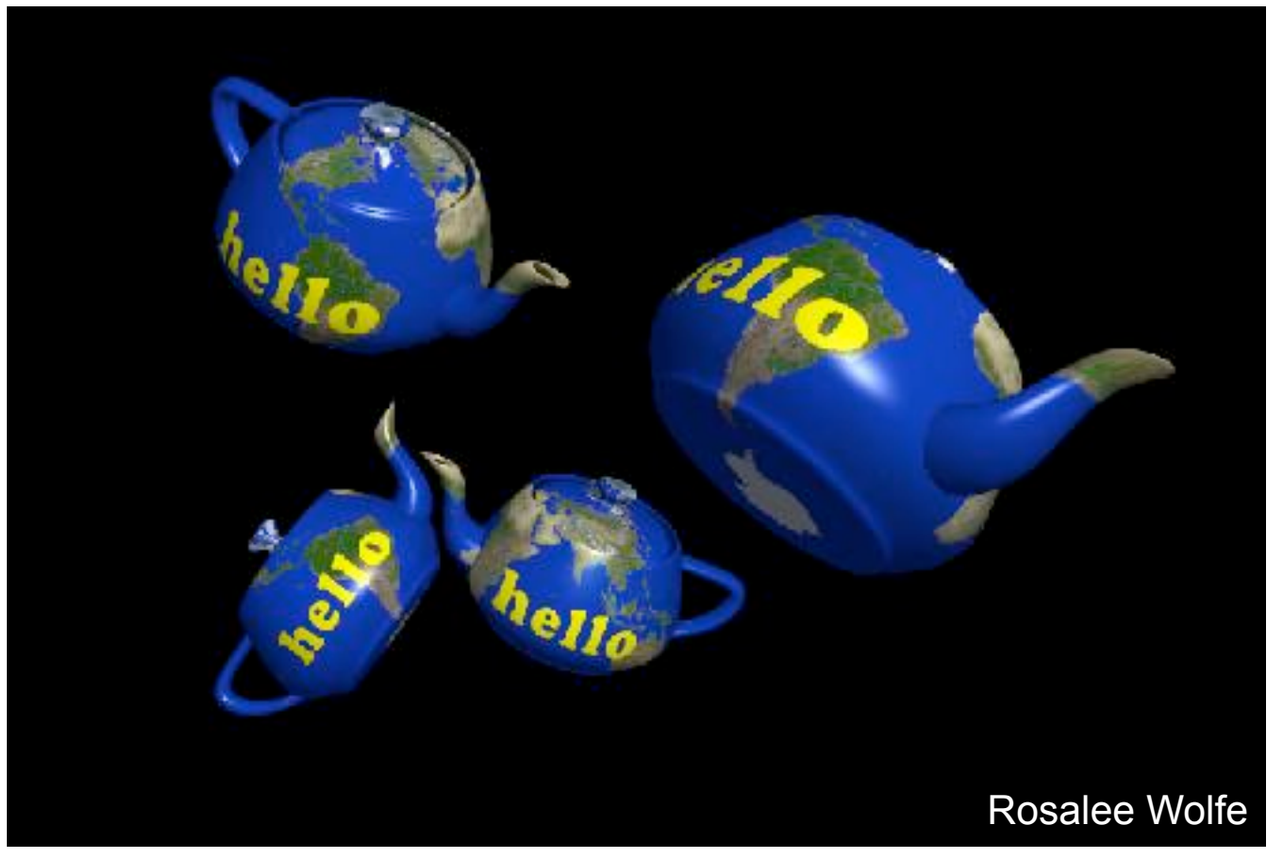


glTexCoord — set the current texture coordinates

Multitexturing

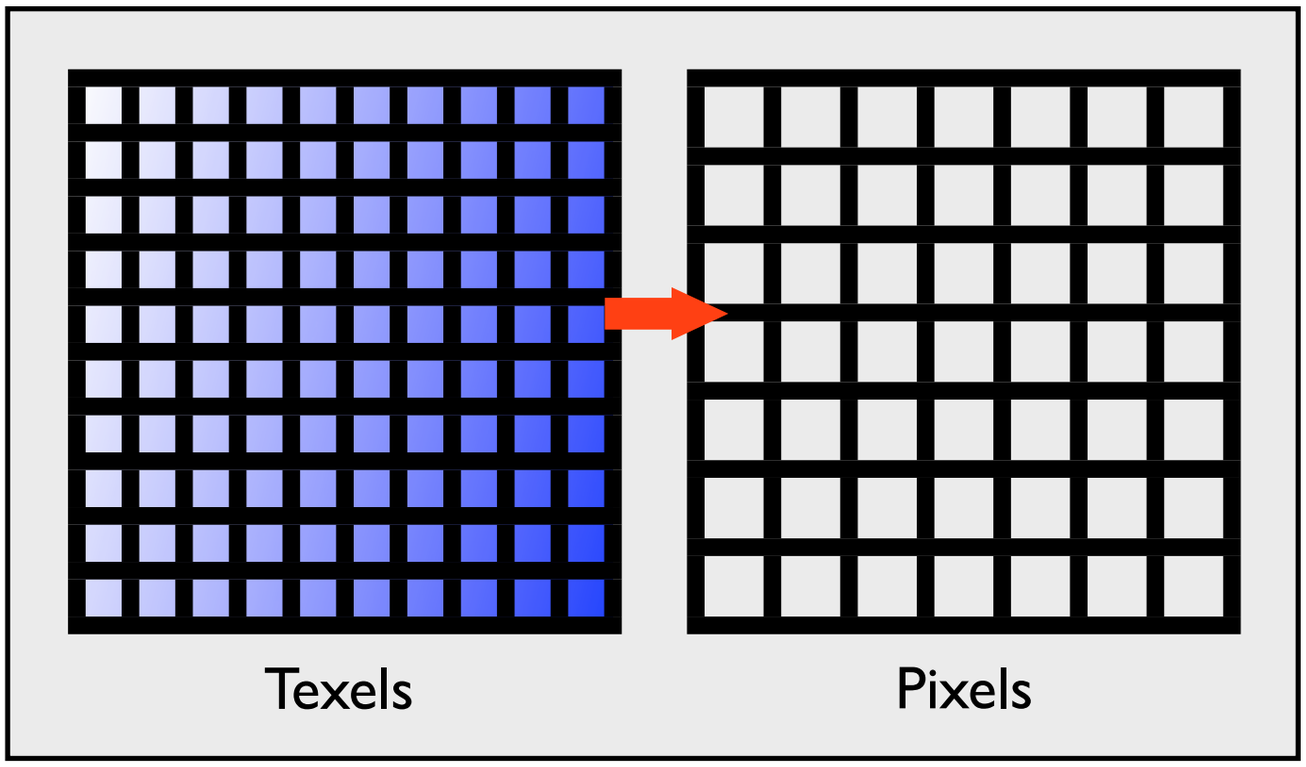


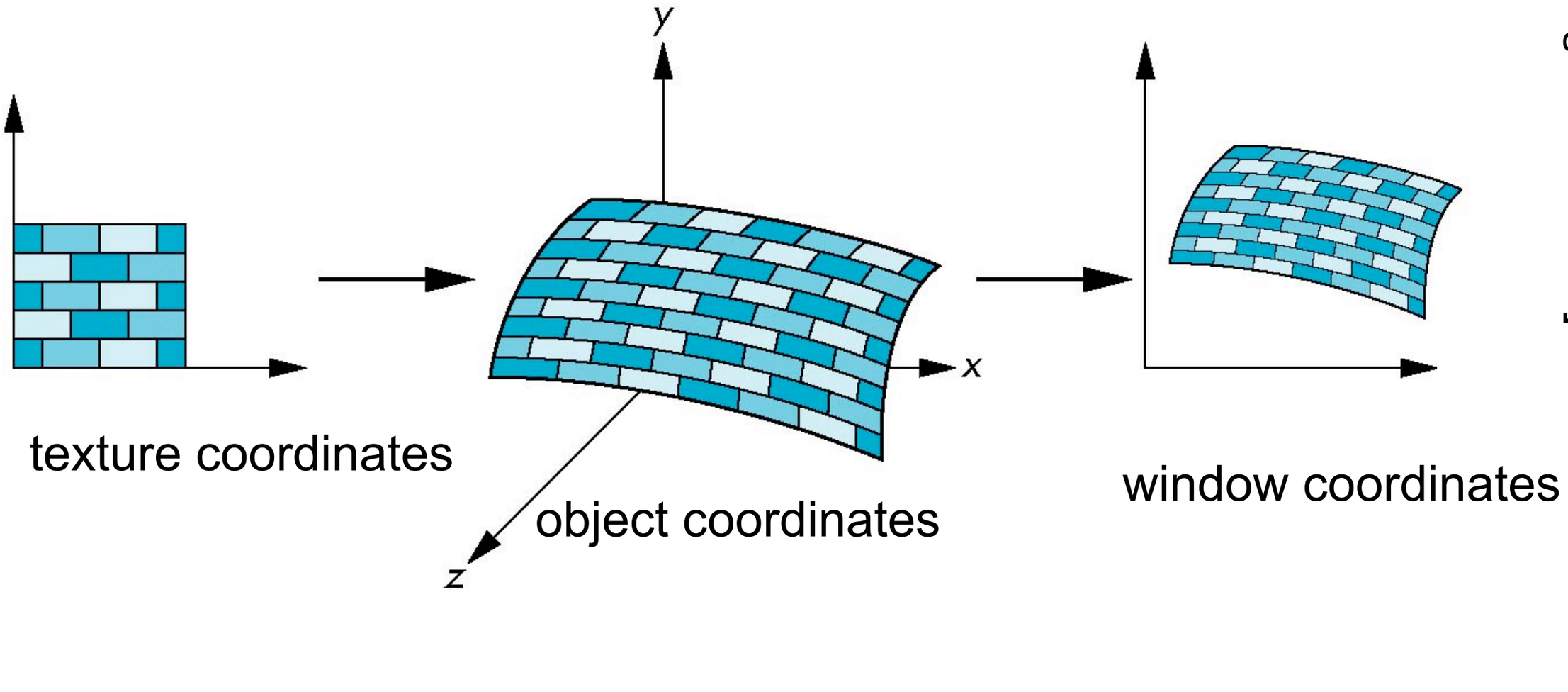




Texture Sampling

Texture Mapping

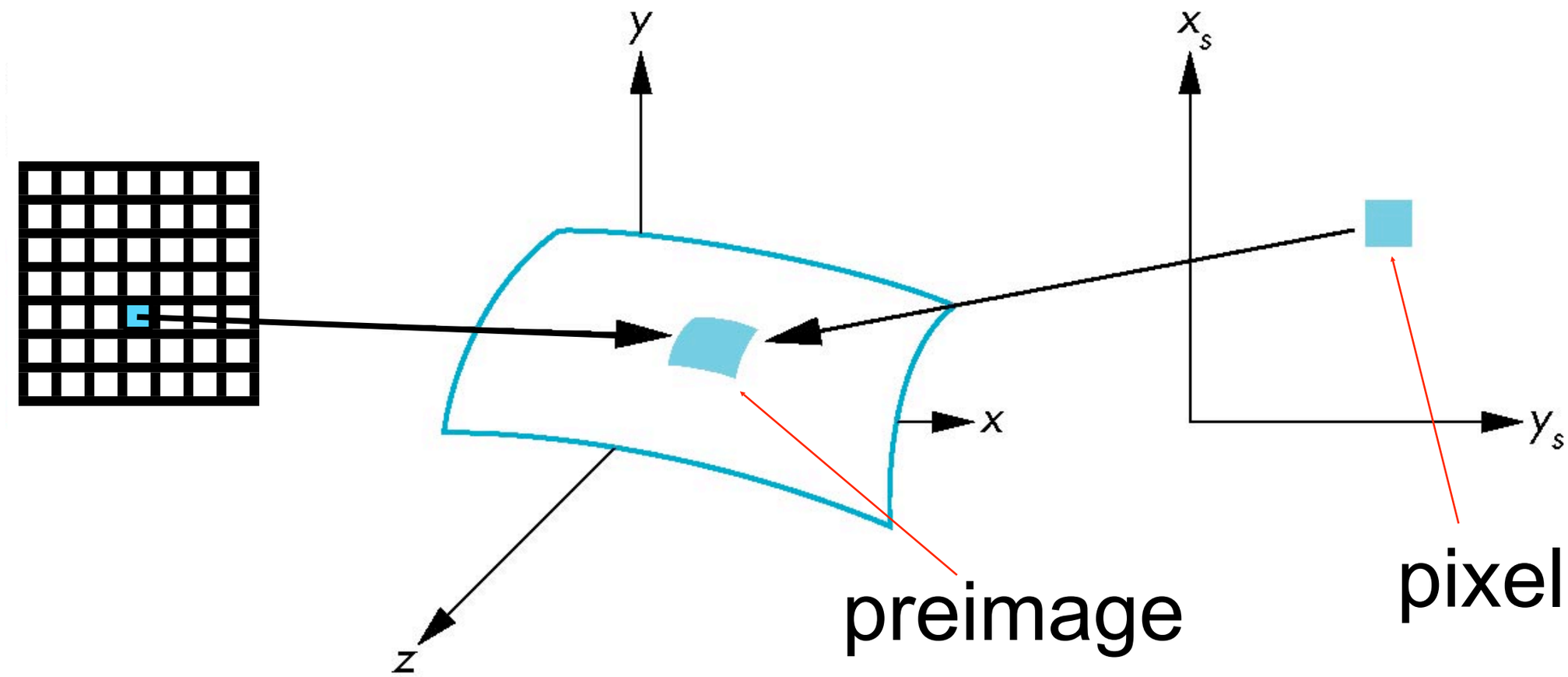




- Texture coordinates: Used to identify points in the image to be mapped
- Object Coordinates: Conceptually, where the mapping takes place
- Window Coordinates: Where the final image is really produced

Point Sampling

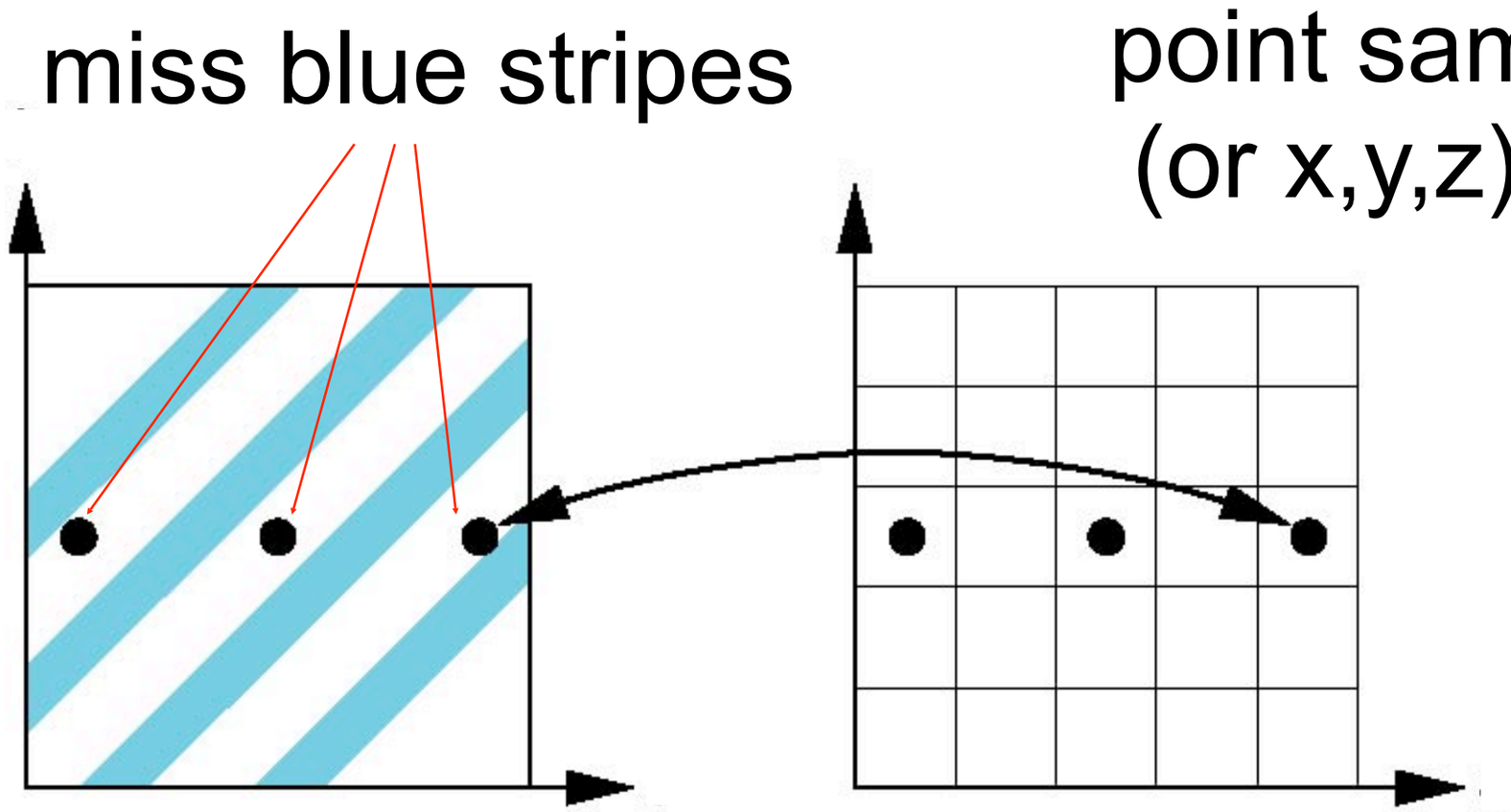
Map back to texture image and use the **nearest texel**



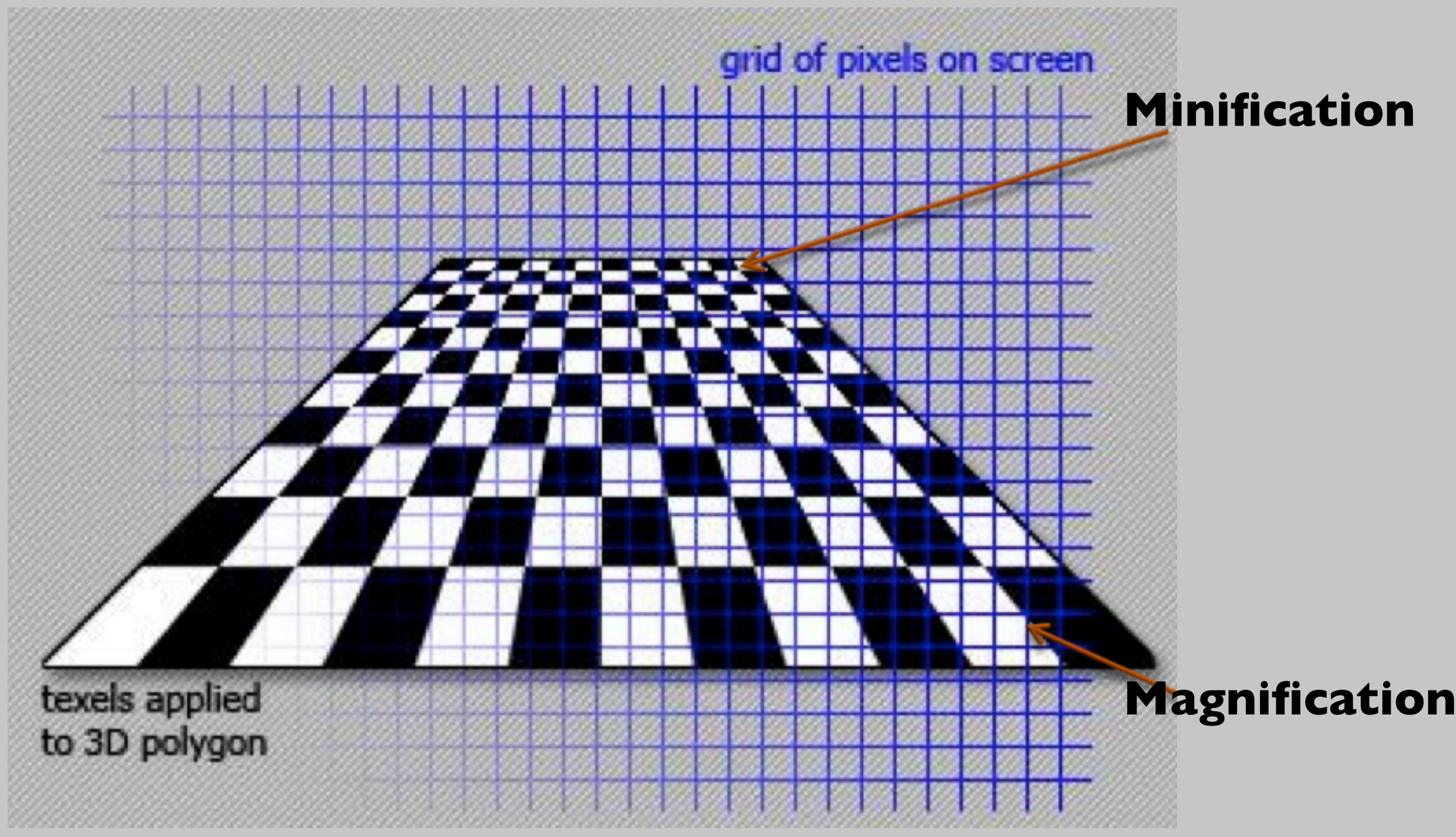
[Angel and Shreiner]

Aliasing

Point sampling of the texture can lead to aliasing artifacts



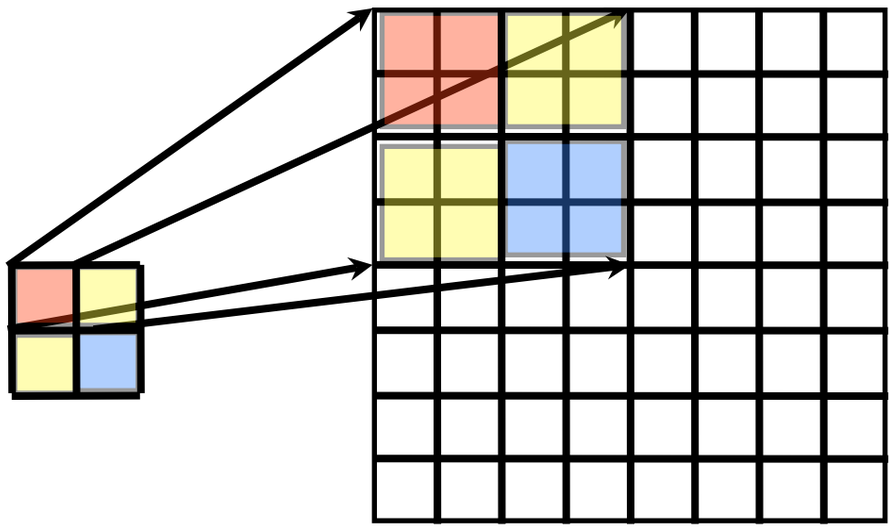
Magnification and Minification



Magnification and Minification

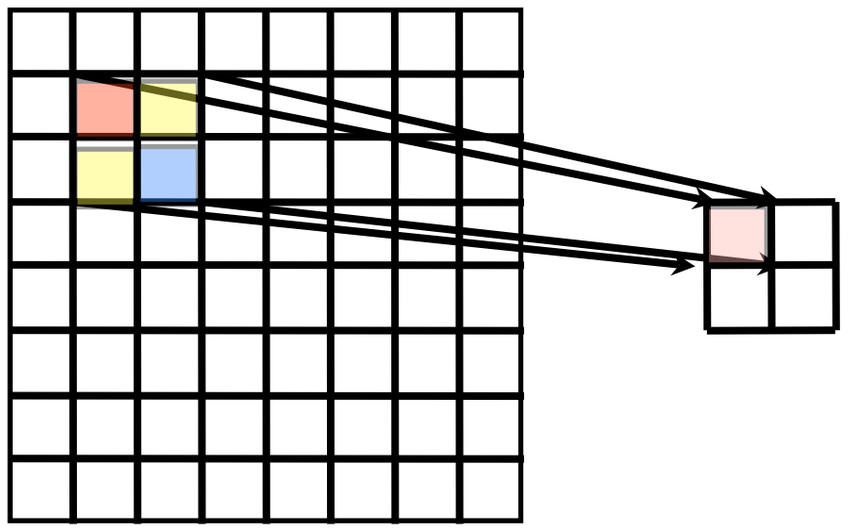
More than one texel can cover a pixel (*minification*) or
more than one pixel can cover a texel (*magnification*)

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



Texture Pixels

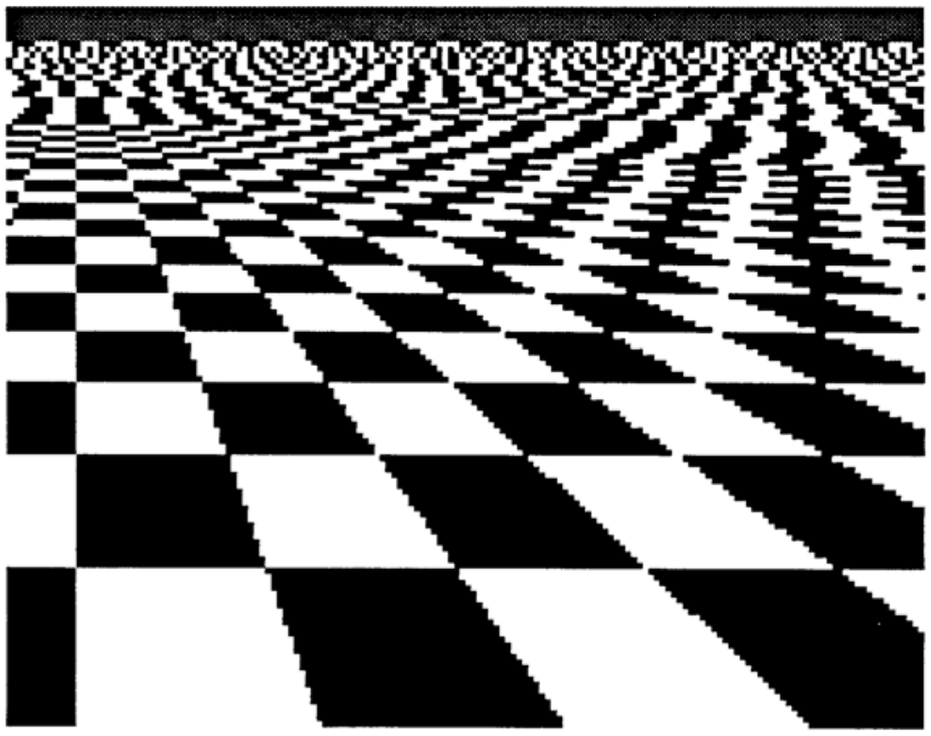
Magnification

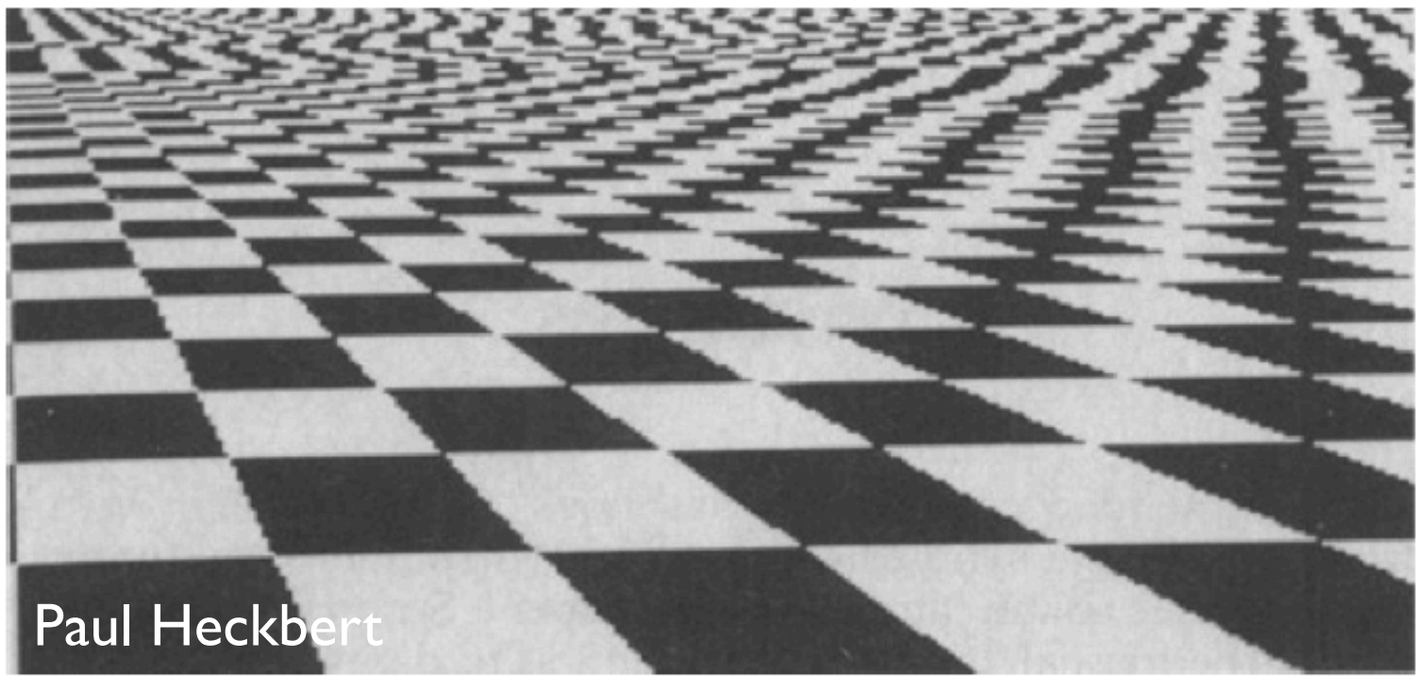


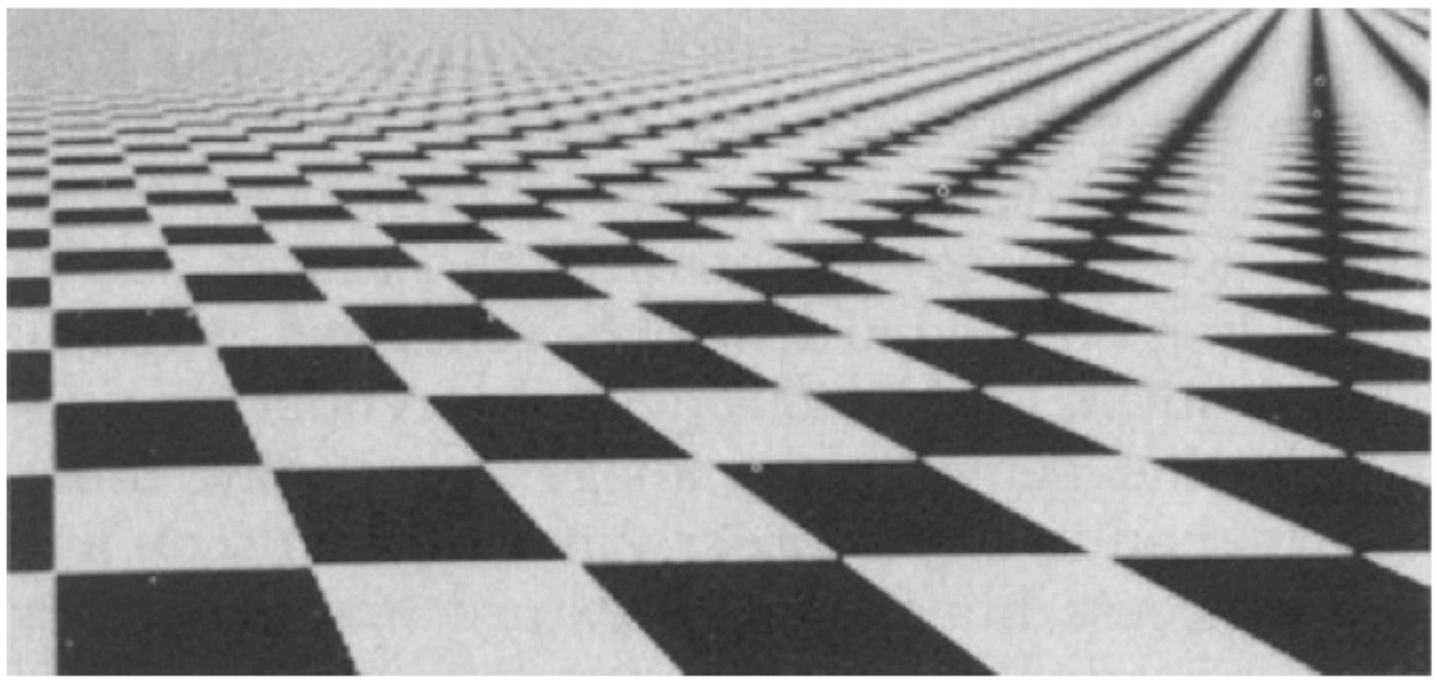
Texture Pixels

Minification

Aliasing artifacts



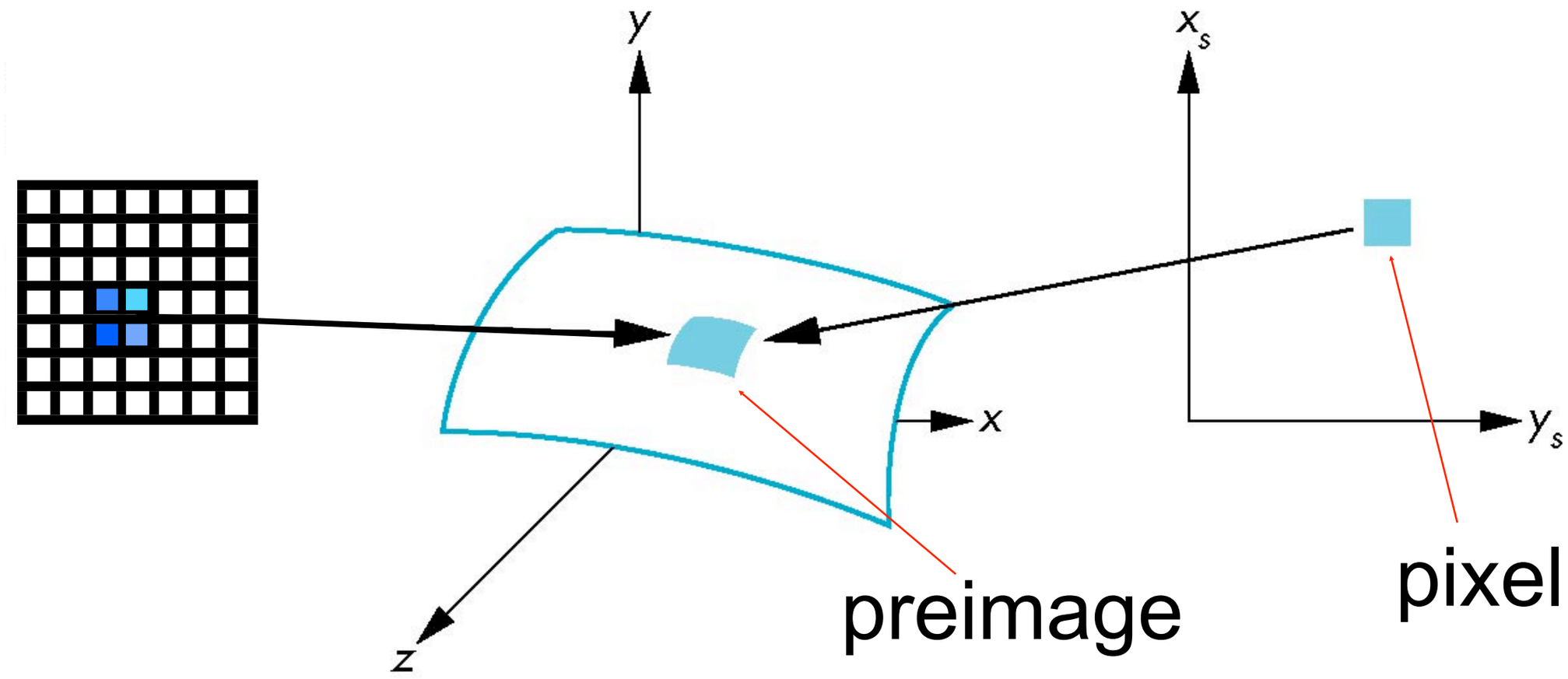




We apply **filtering** to reduce aliasing artifacts

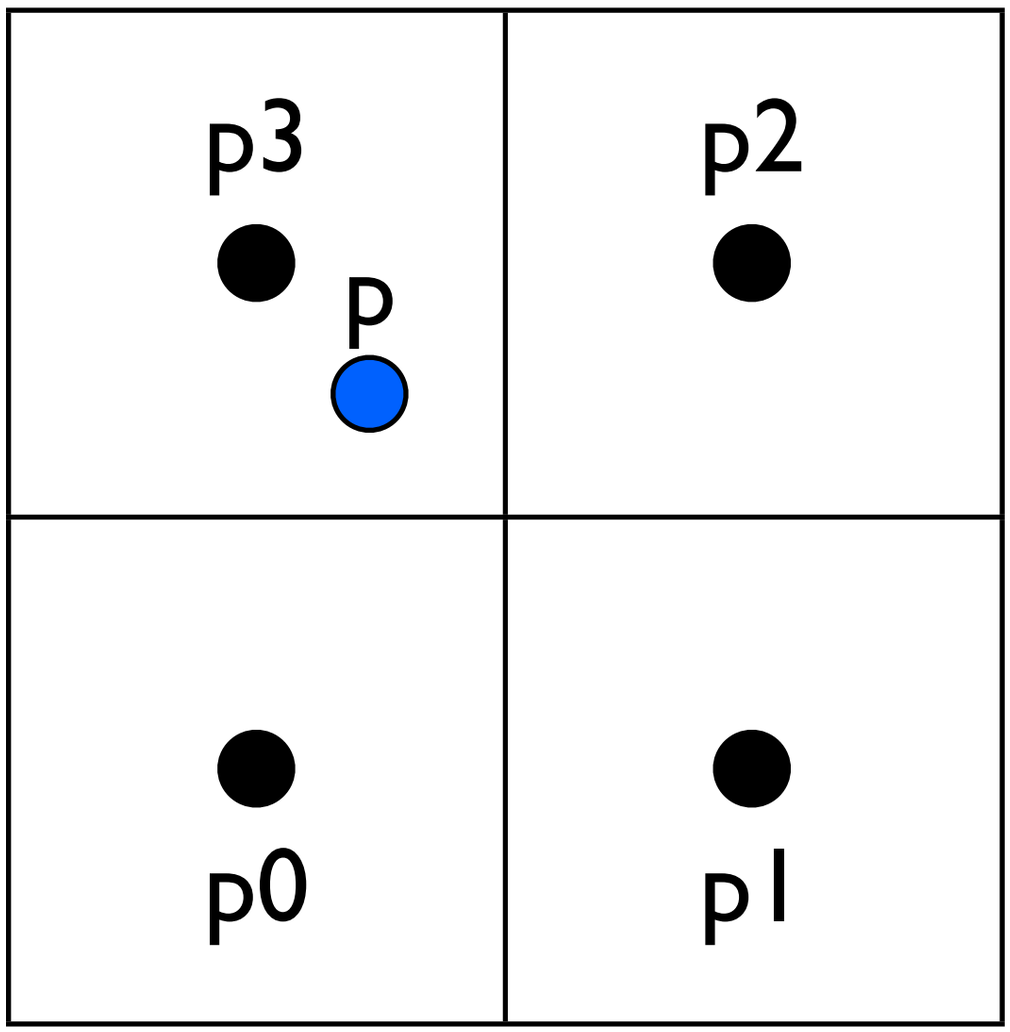
Area Averaging

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



[Angel and Shreiner]

Use bilinear filtering



p = ?

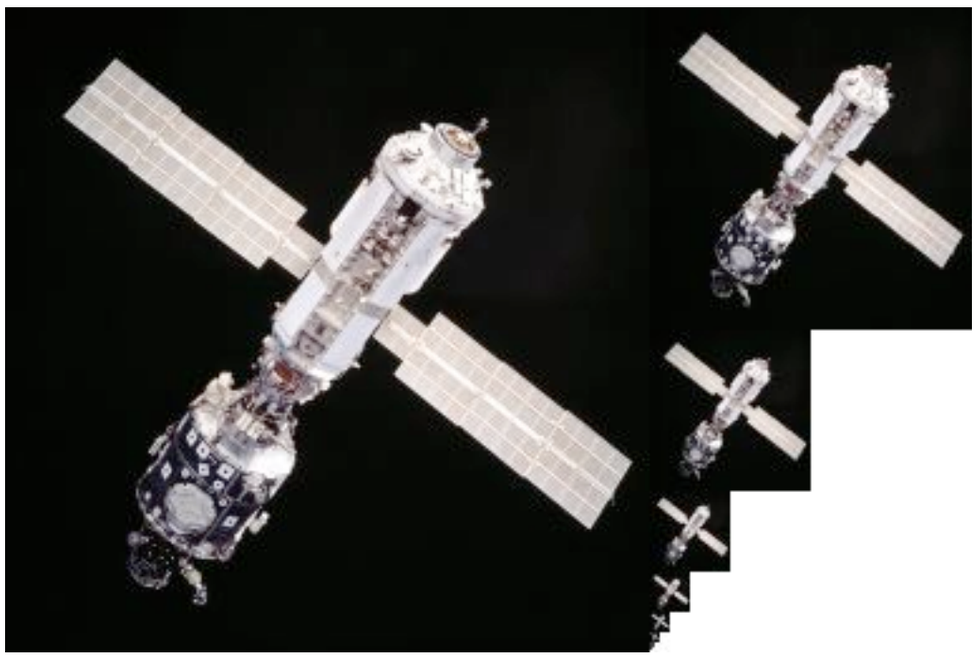
nearest neighbor **bilinear** **bicubic**

Wikipedia

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

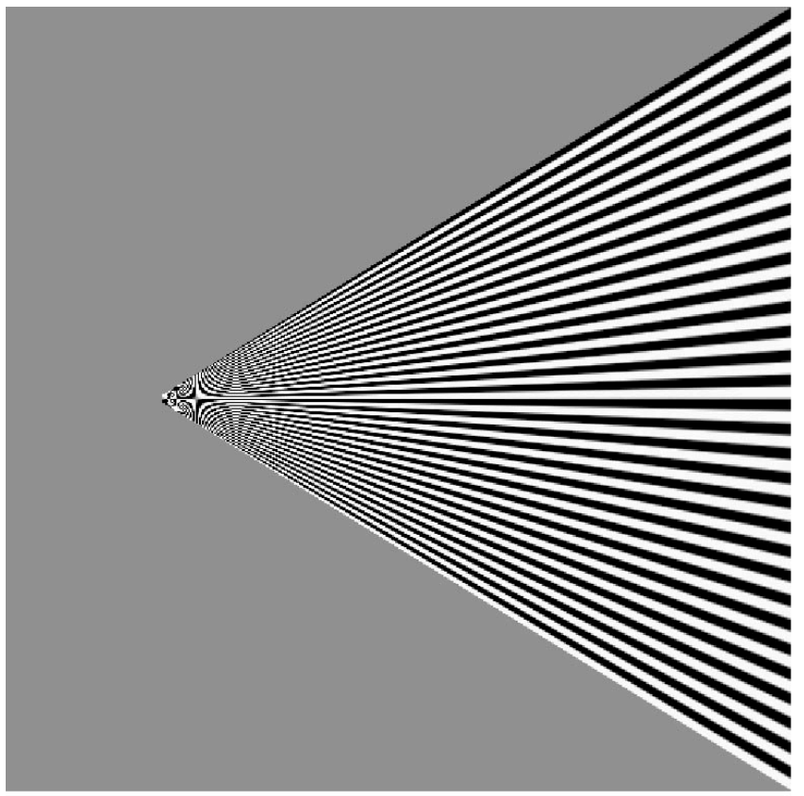
Reduce minification artifacts

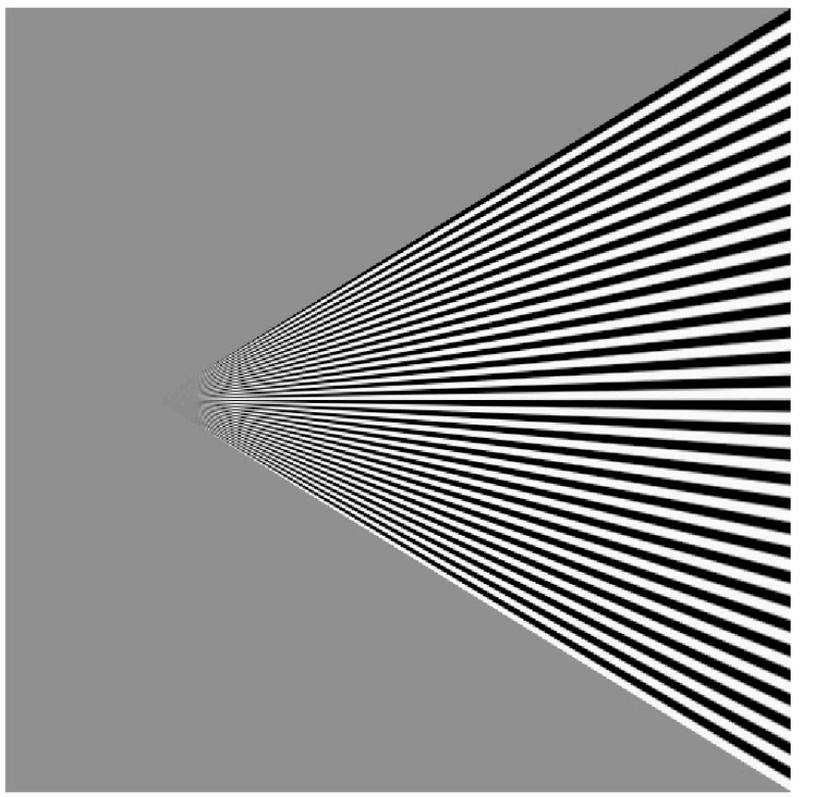
Prefilter the texture to obtain reduced resolutions

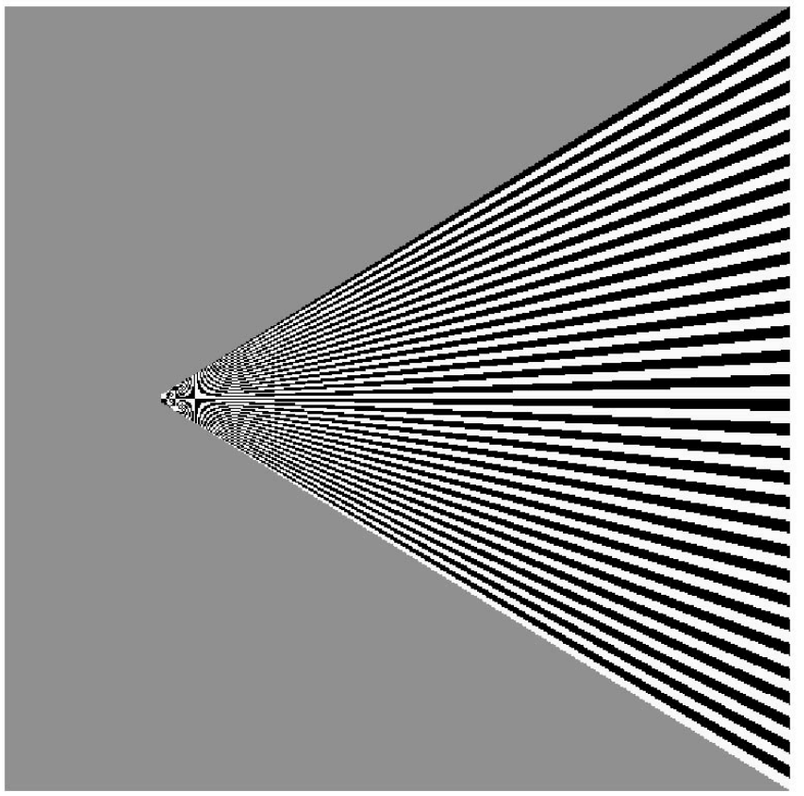
Requires 1/3 more space

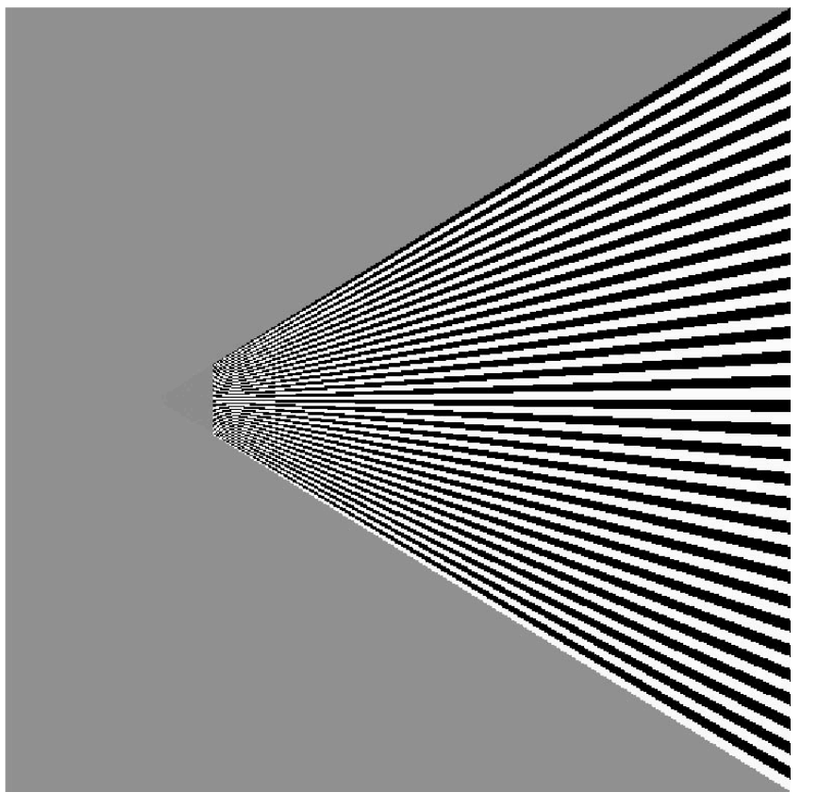
Get a texture hierarchy indexed by level

[Angel and Shreiner]

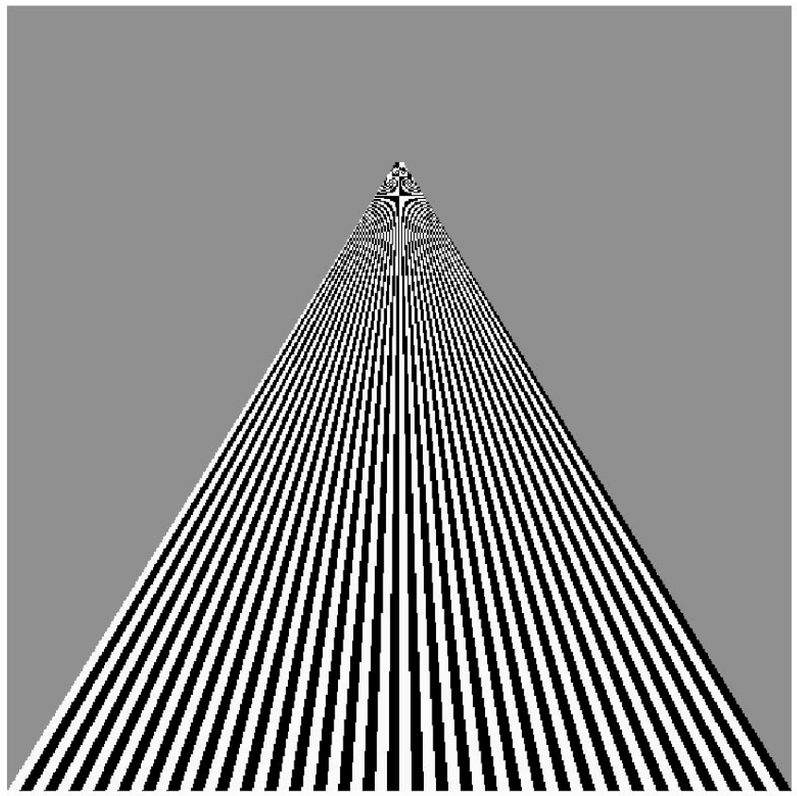




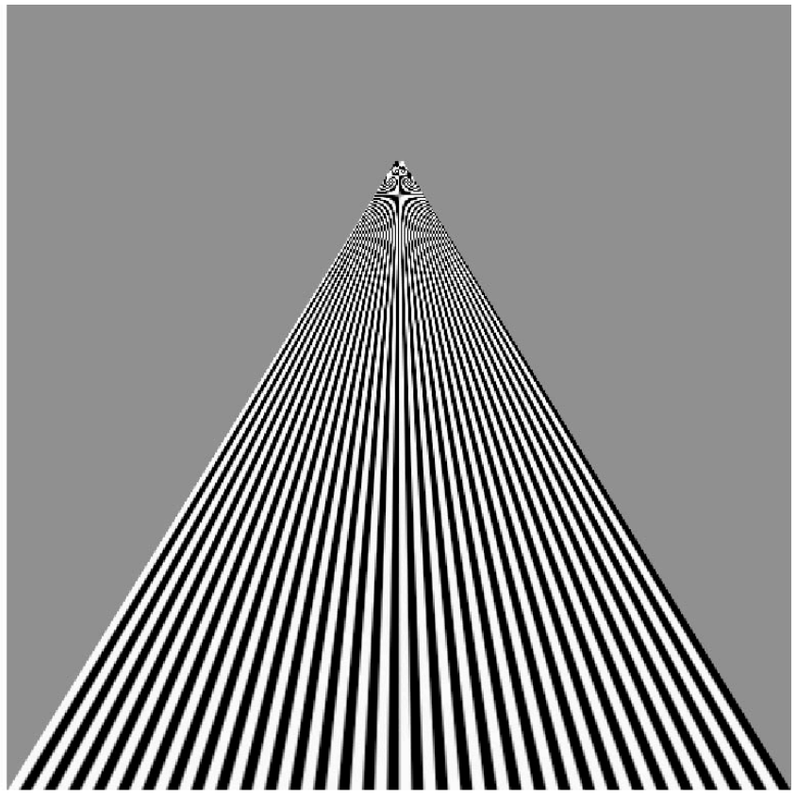




point sampling

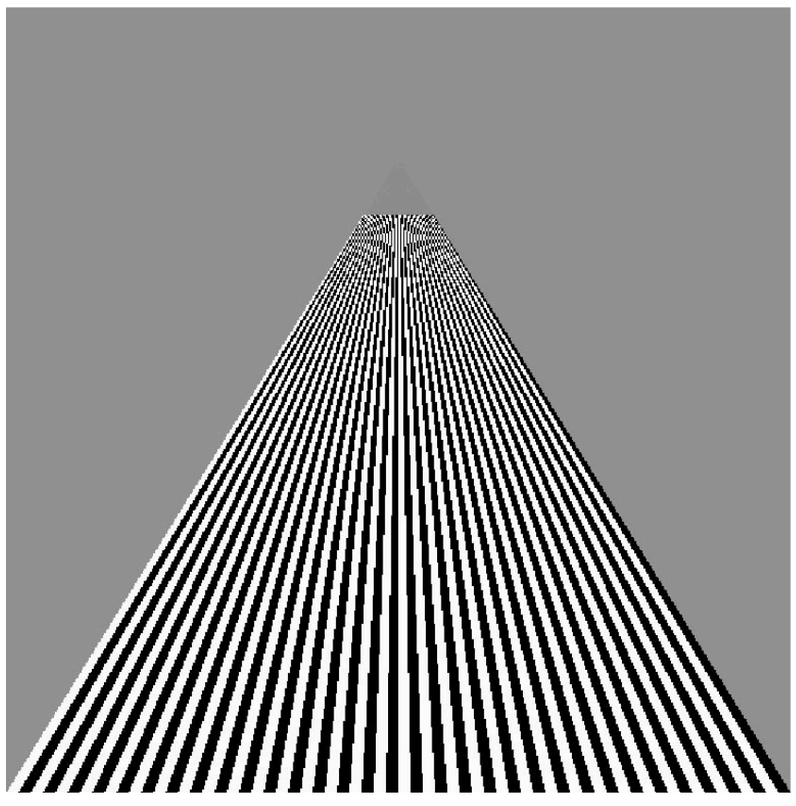


linear filtering

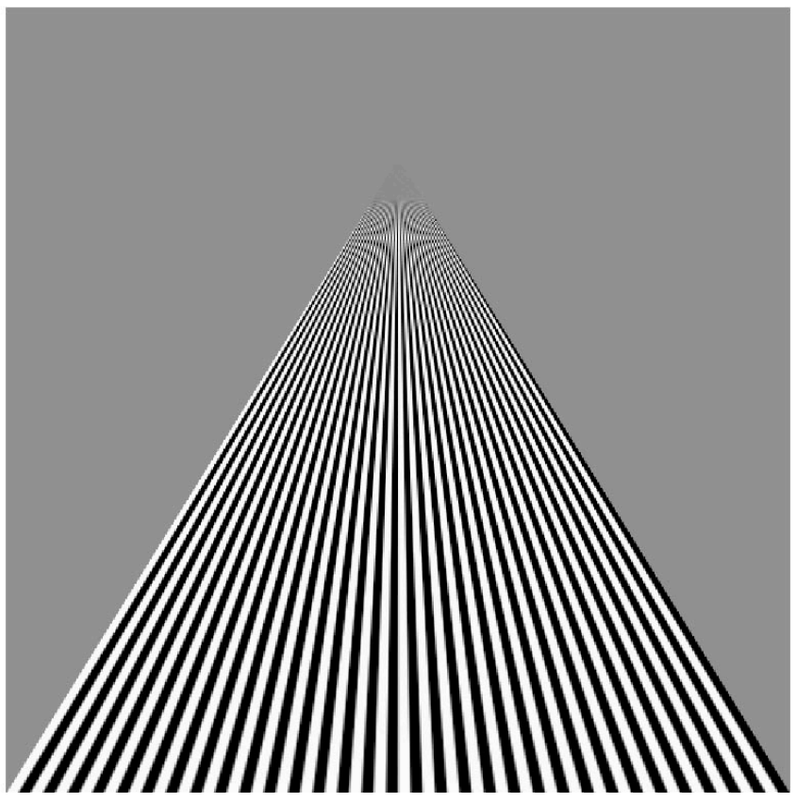


[Angel and Shreiner]

mipmapped point sampling



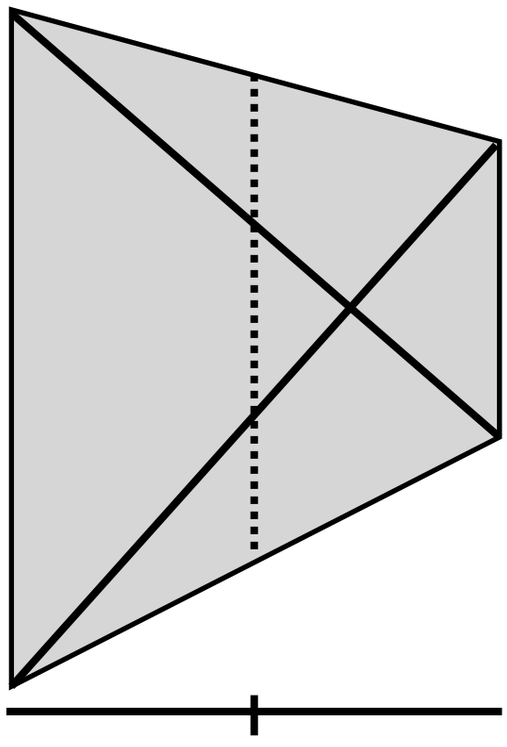
mipmapped linear filtering

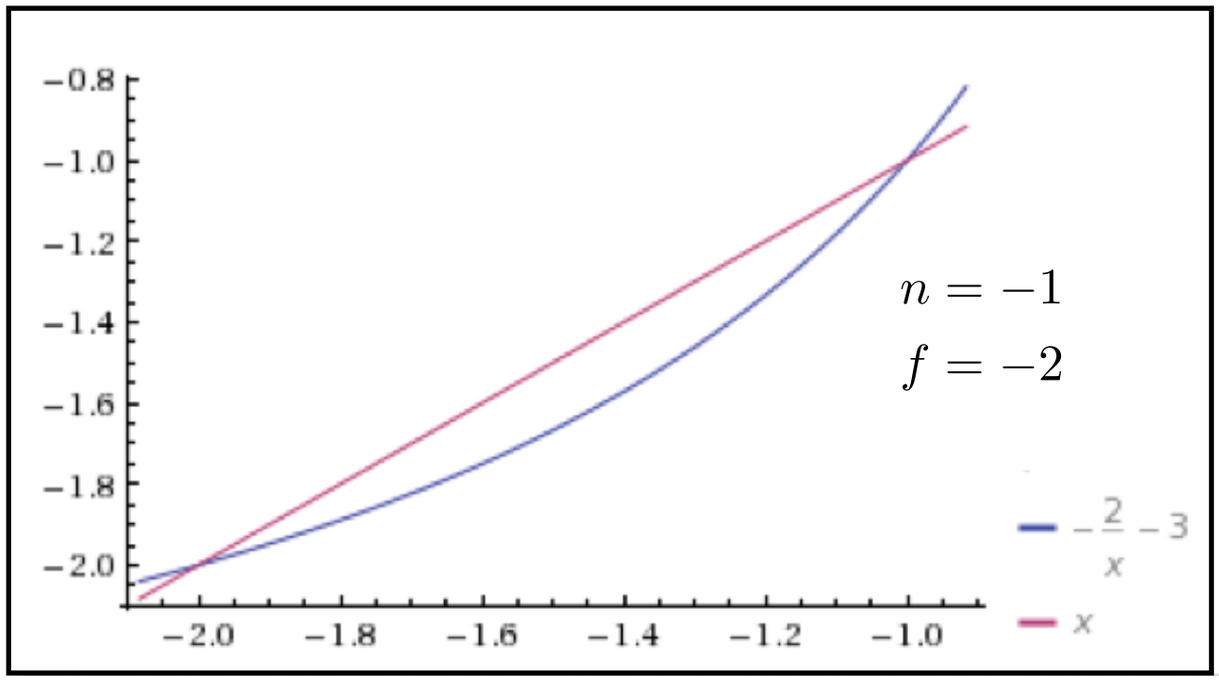


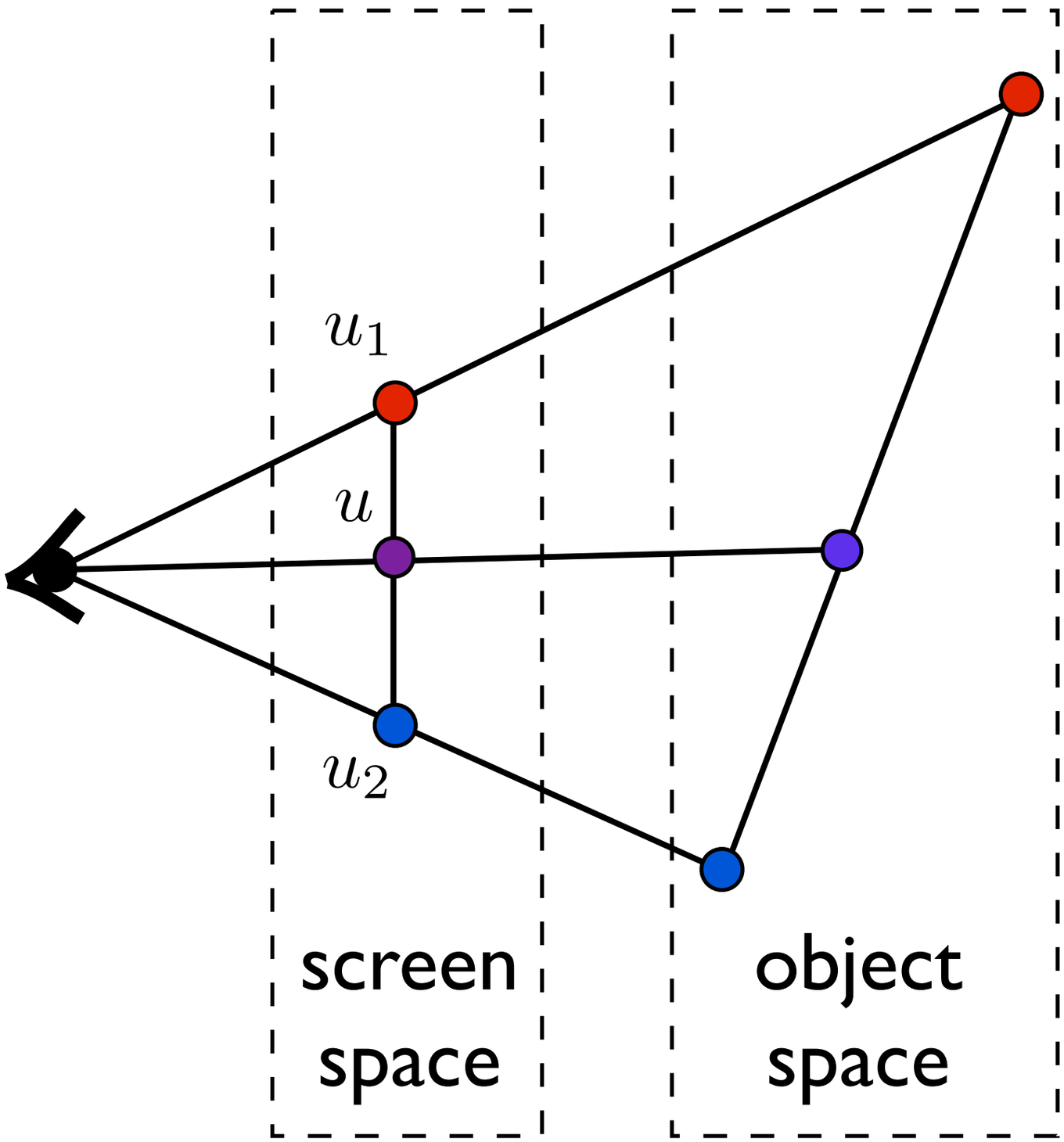
Perspective correct interpolation

Perspective correct interpolation

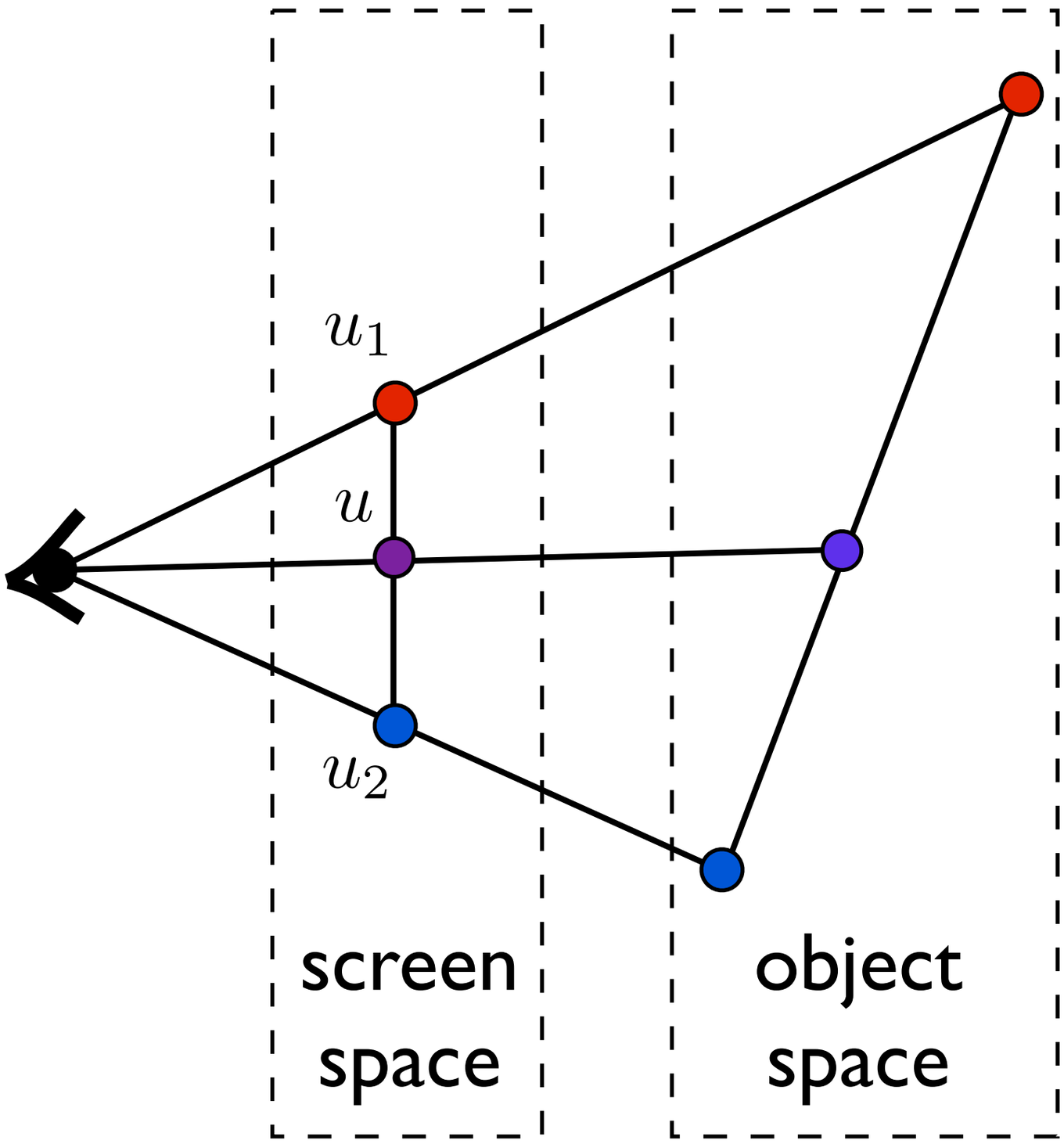
- In triangle rasterization algorithm, we found barycentric coordinates in 2D screen space
  - but not the correct object space barycentric coords
  - these coordinates are *okay* for z-buffer test







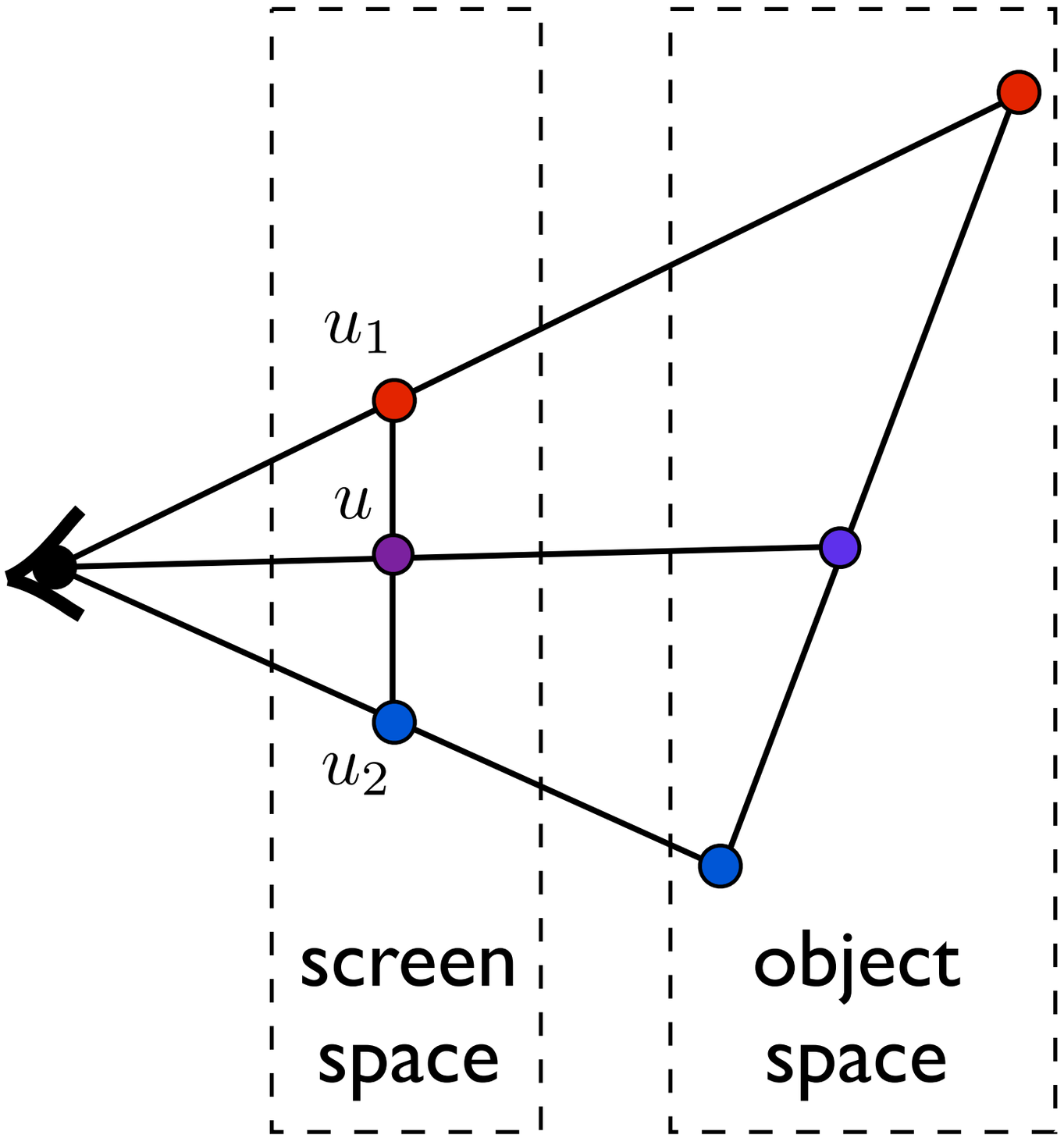
$$u = \frac{1}{2}u_1 + \frac{1}{2}u_2$$



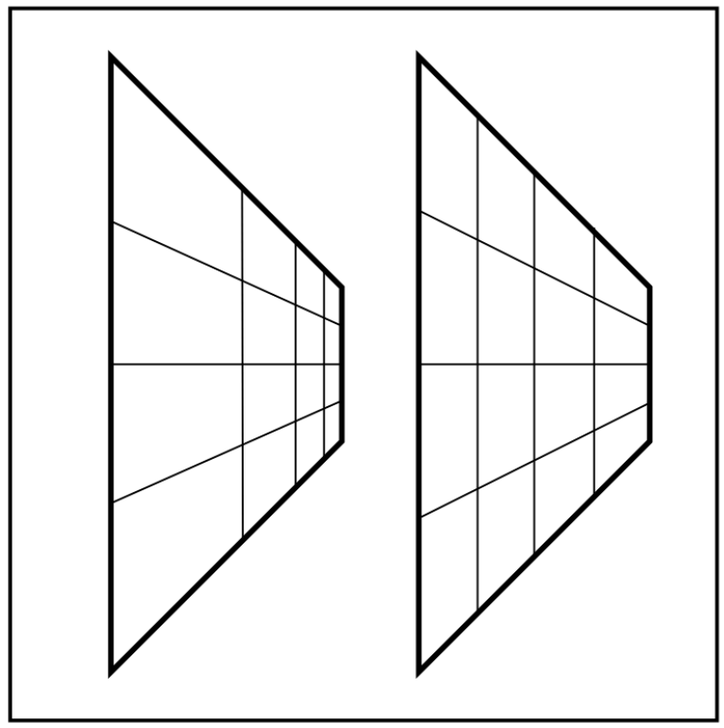
$$u = \frac{1}{2}u_1 + \frac{1}{2}u_2$$

Issue: to shade a fragment which is part of a textured triangle
we need the barycentric coordinates of the fragment
These will be the weights for the weighted average of the vertex texture coordinates.
However, after a perspective transformation, the relative distances inside the triangle have been distorted due to foreshortening.
I need to get my weights based on object or world space coordinates.

Interpolation with screen space weights is incorrect

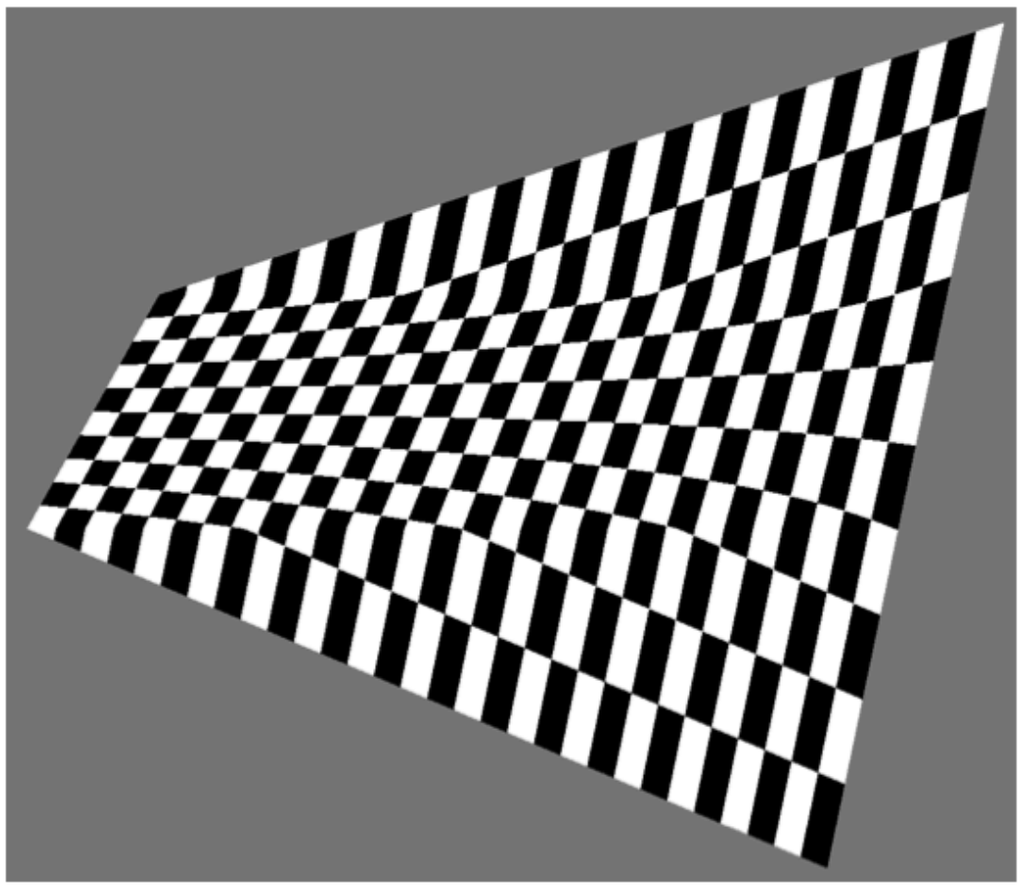


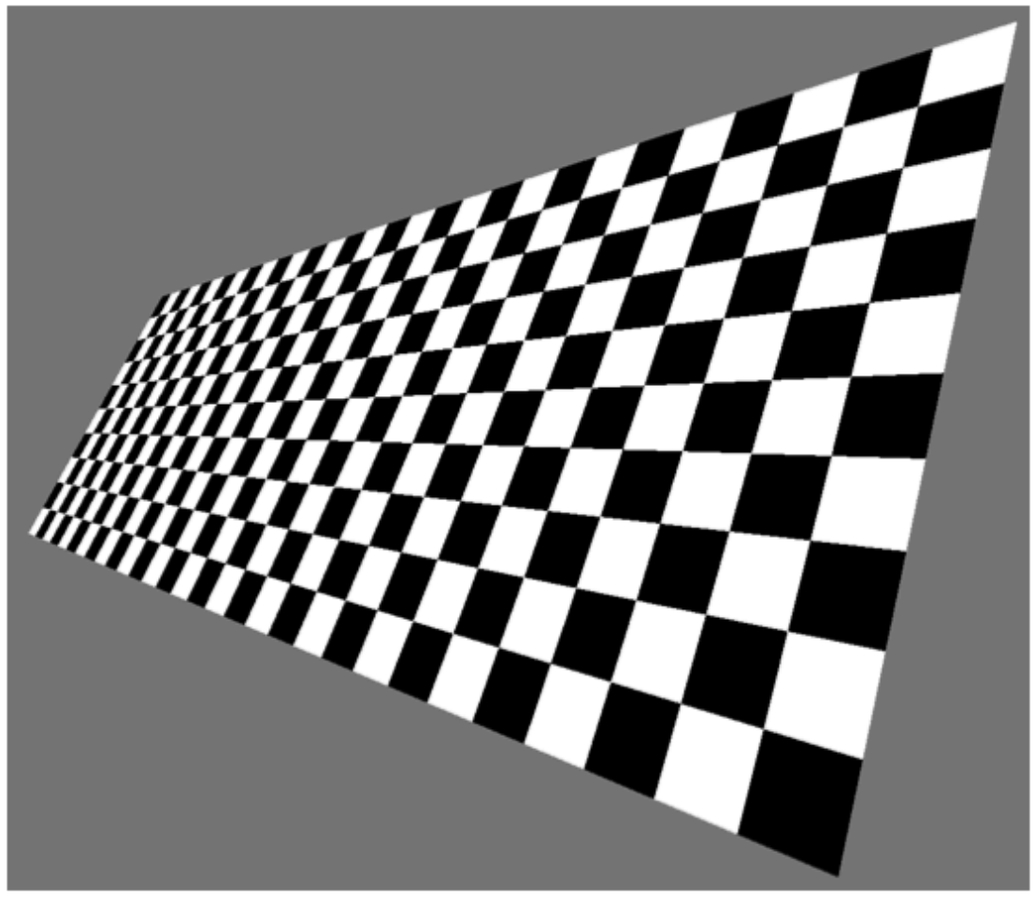
$$u = \frac{1}{2}u_1 + \frac{1}{2}u_2$$



Perspective correct interpolation

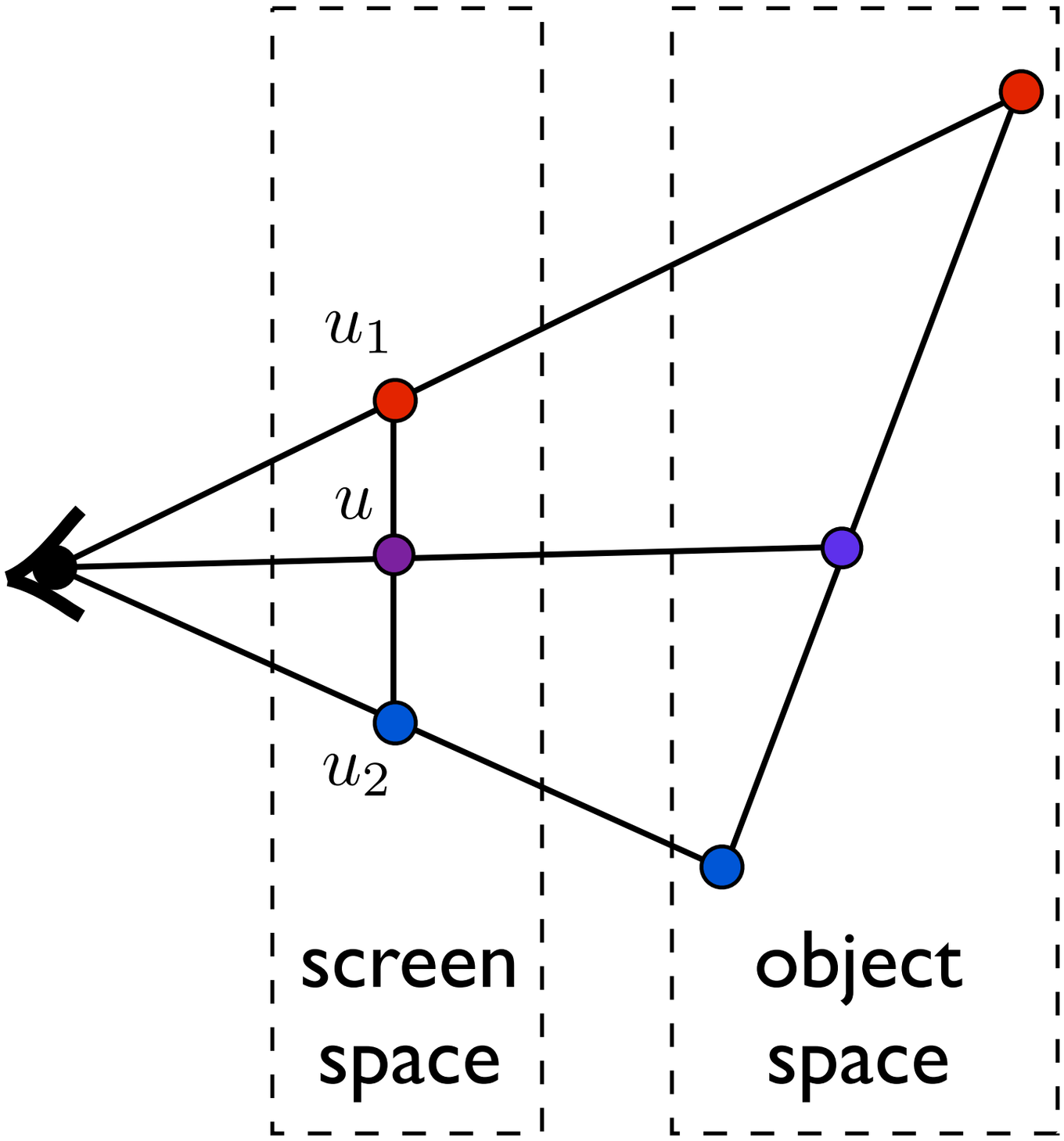
Using screen space weights looks wrong for textures





[Heckbert and Morton, 1990]

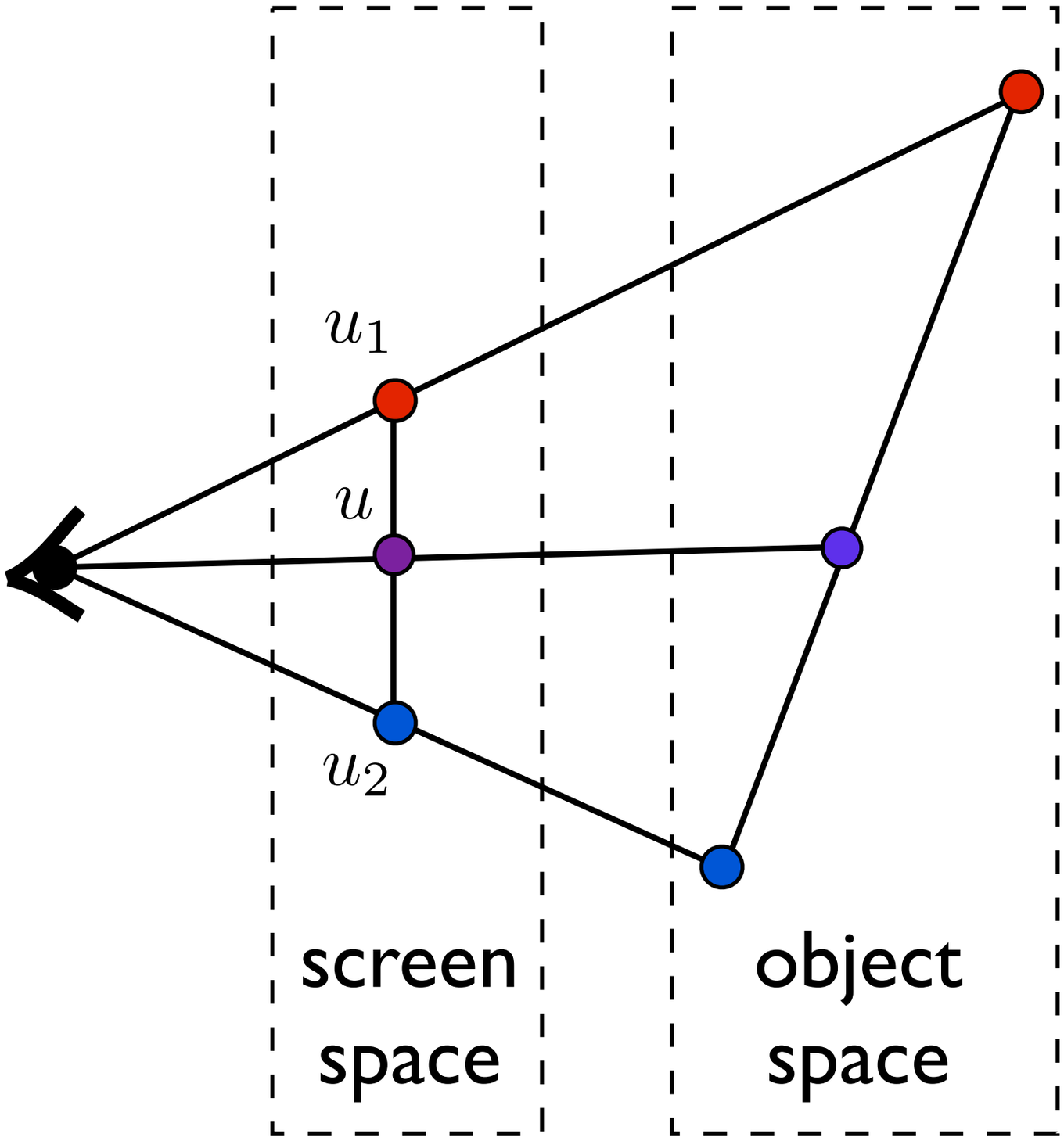
http://en.wikipedia.org/wiki/Texture_mapping#Perspective_correctness



$$u = \frac{1}{2}u_1 + \frac{1}{2}u_2$$

Do we need to transform back to object space?

$$\mathbf{v}_{\mathrm{sc}} = M_{\mathrm{vp}} M_{\mathrm{pers}} M_{\mathrm{cam}} \mathbf{v}$$



$$u = \frac{1}{2}u_1 + \frac{1}{2}u_2$$

Do we need to transform back to object space?

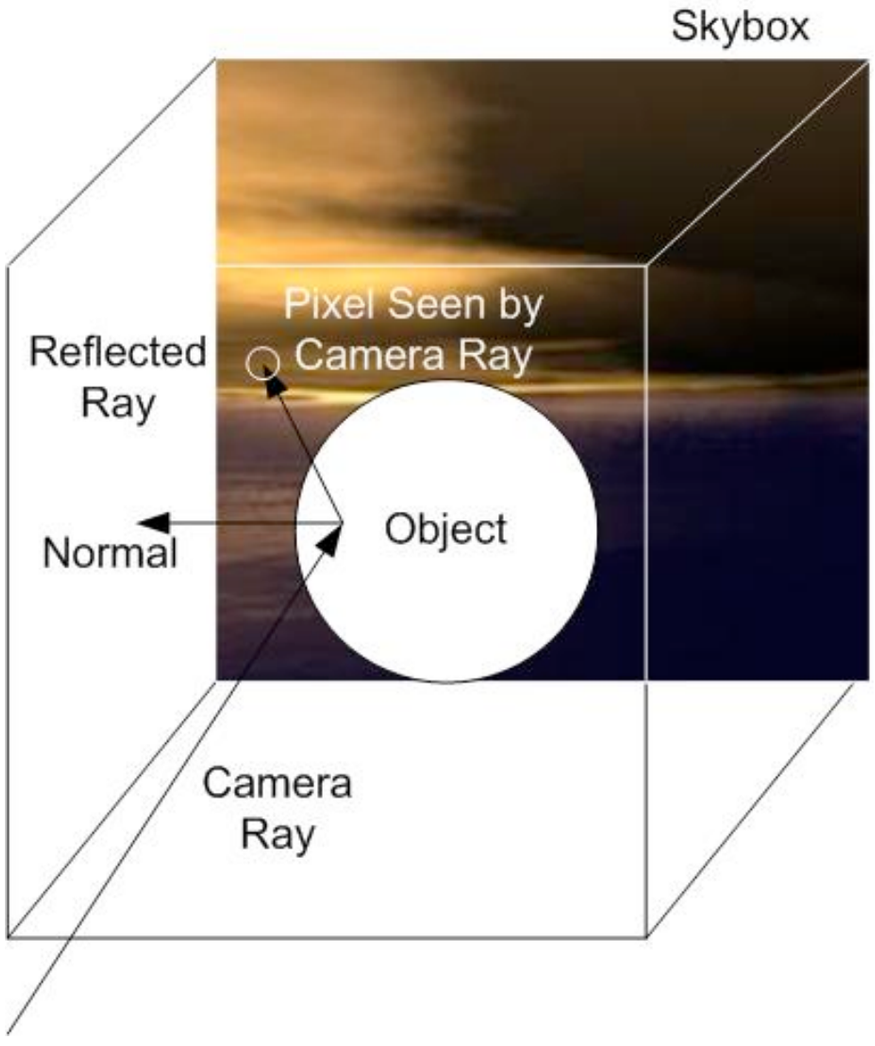
NO!

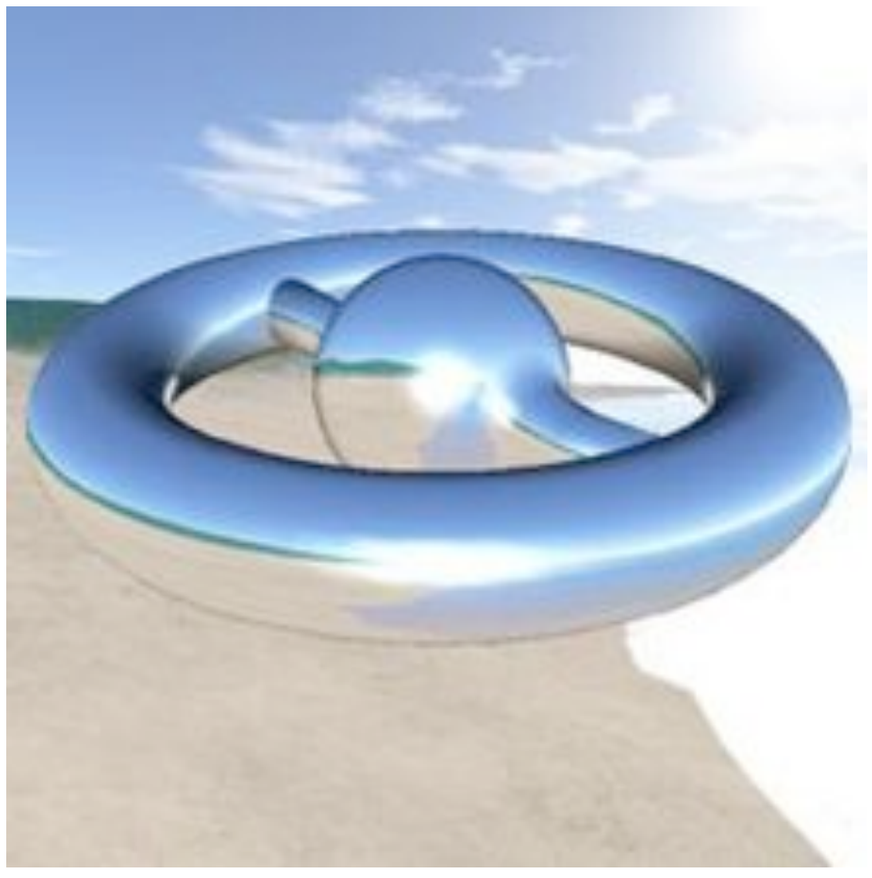
<whiteboard>

Environment mapping

Environment Mapping

Use a texture for the distant environment simulate the effect of ray tracing more cheaply

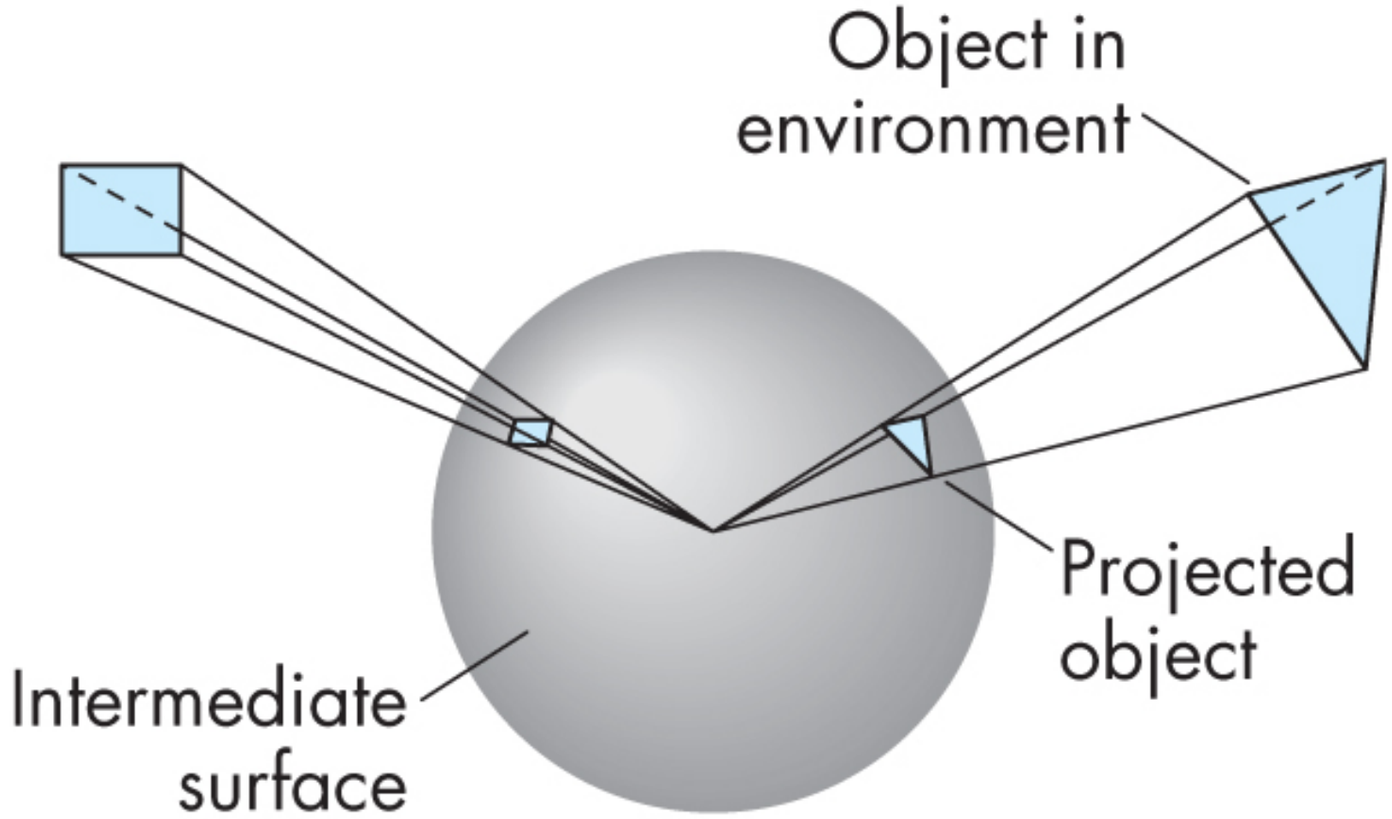




Wikimedia Commons

Sphere Mapping

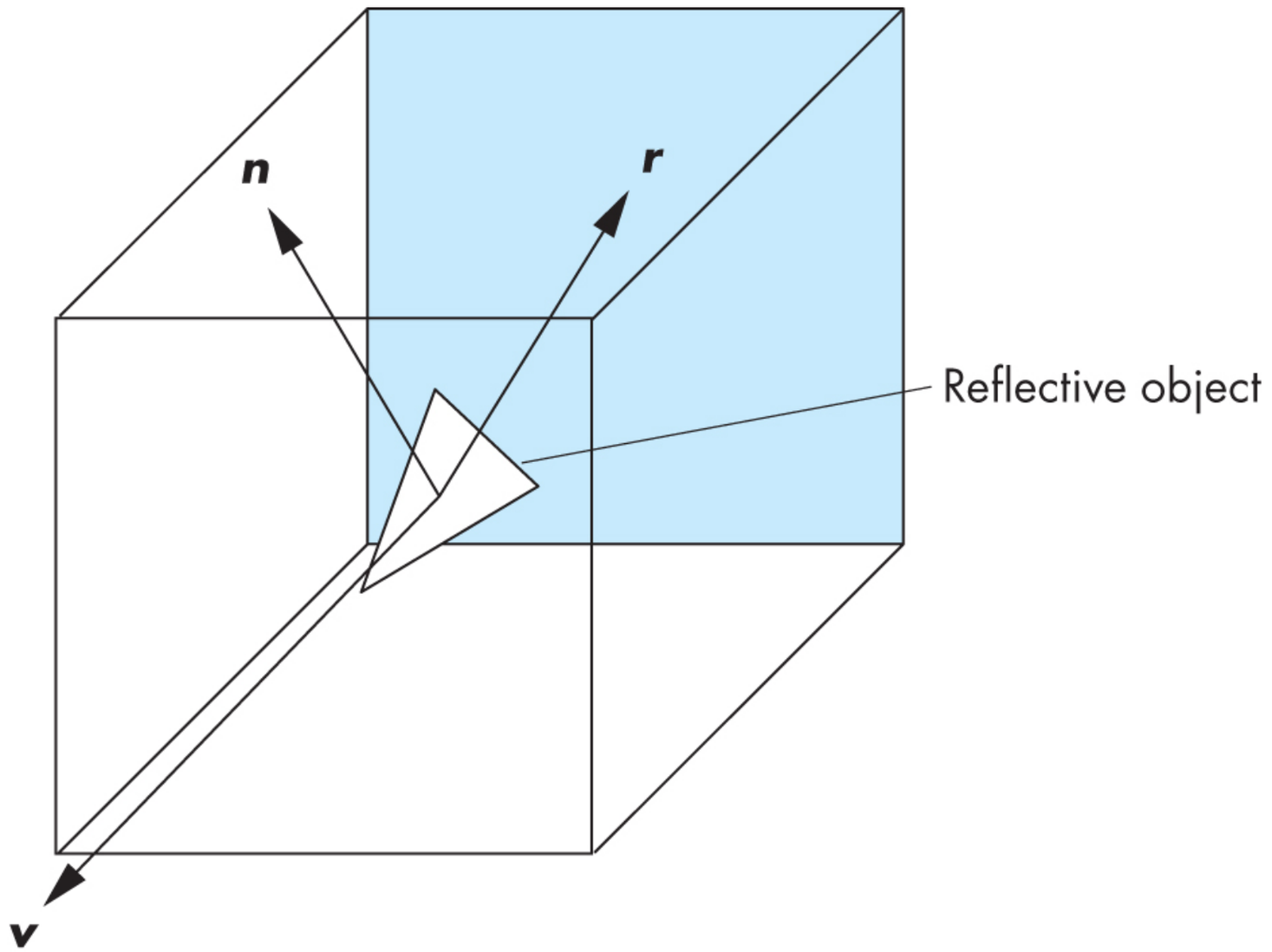
- Project objects in the environment onto sphere centered at eye
- unwrap and store as texture
- use reflection direction to lookup texture value



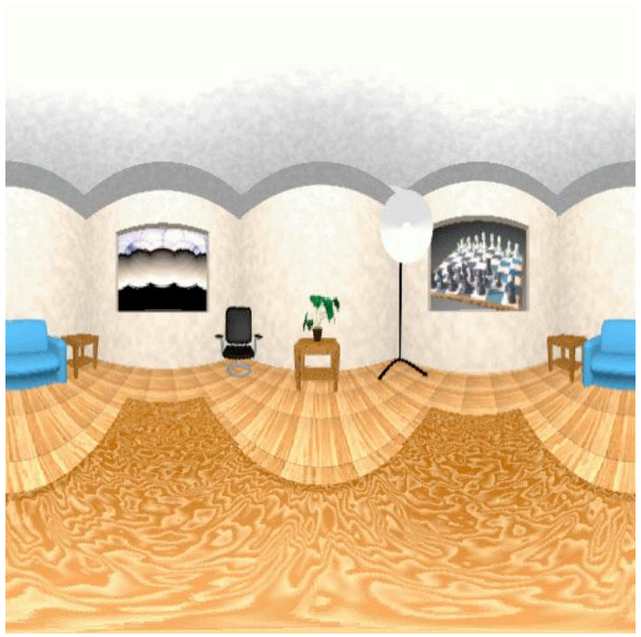
How is environment mapping different from ray tracing?
 - typically only the direction of the reflection vector is used to look up the texture value- this doesn't reproduce the true intersection of the reflected ray the the object it hits
- Note: realism of environment map degrades as model is displaced from where the textures were generated

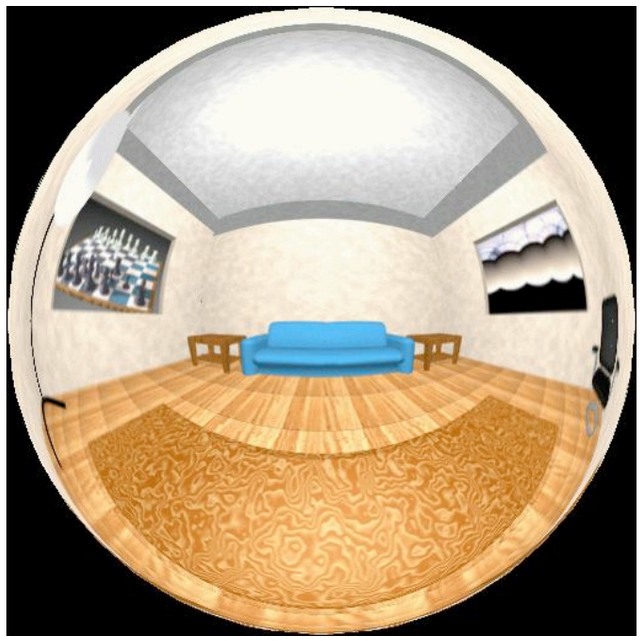
Cube Mapping

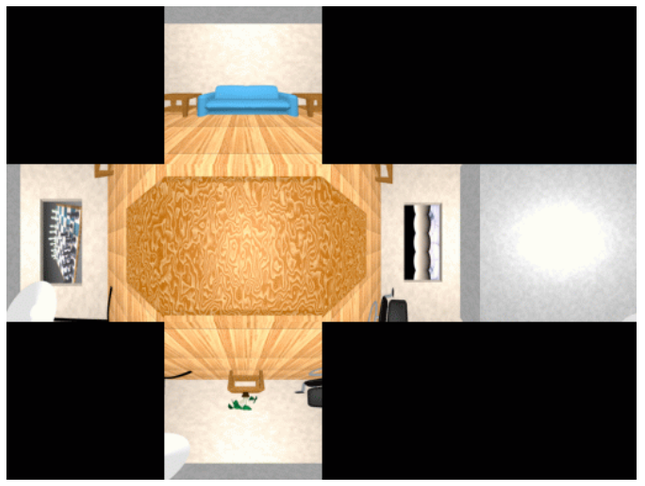
- Compute six projections, one for each wall
- store as texture
- use reflection direction to lookup texture value



Different environment maps







www.reindelsoftware.com

Blinn/Newell latitude mapping

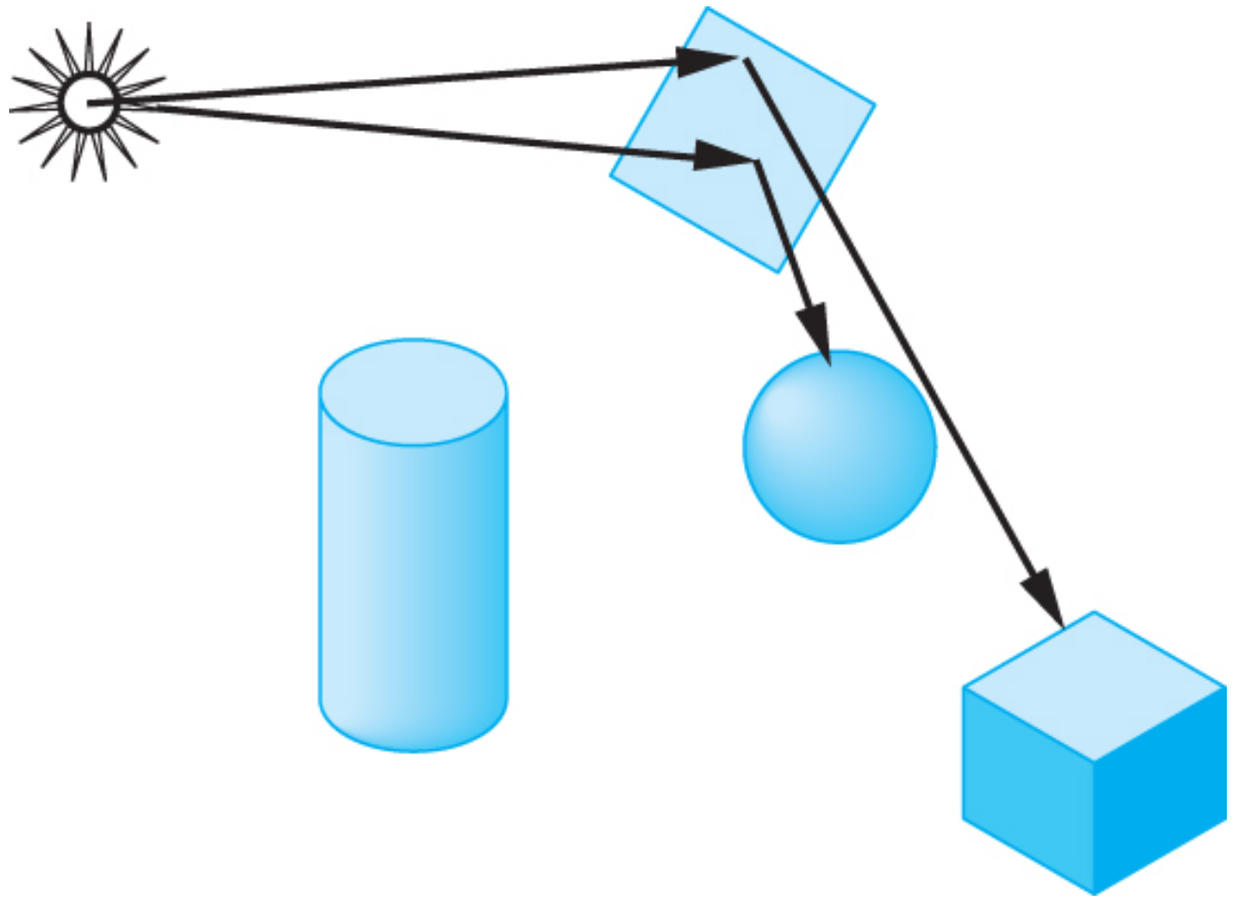
OpenGL spherical mapping

Cube mapping

Environment Mapping

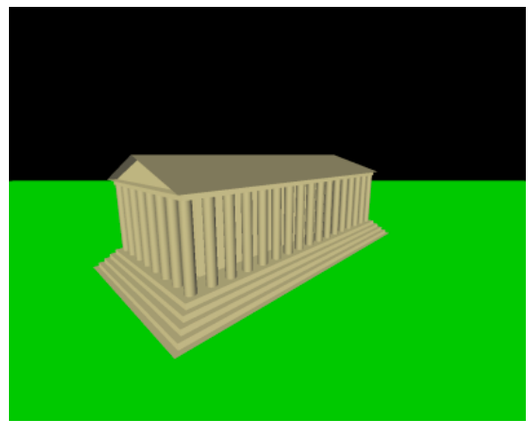
Create the effect of a mirror with two-pass rendering

1. First pass: render the scene from the perspective of the mirror
2. Second pass: render from original pov; use the first image as a texture for the mirror



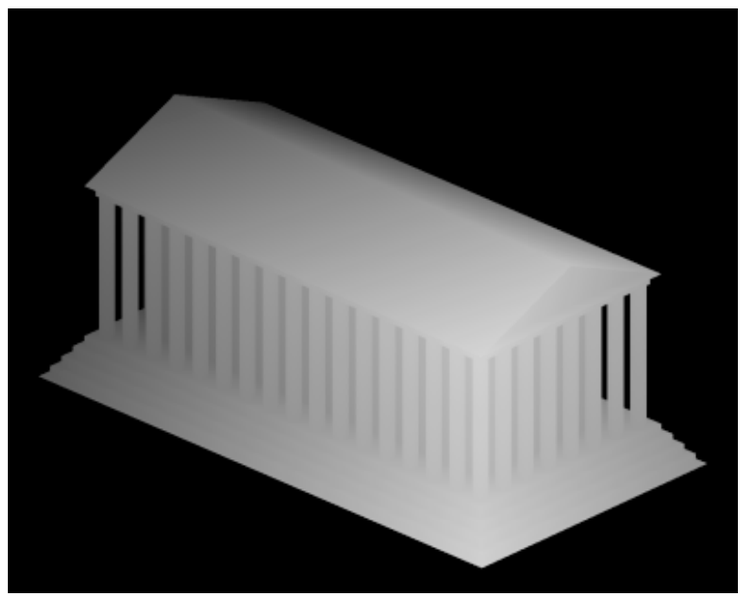
Shadow Mapping

first pass from light's perspective

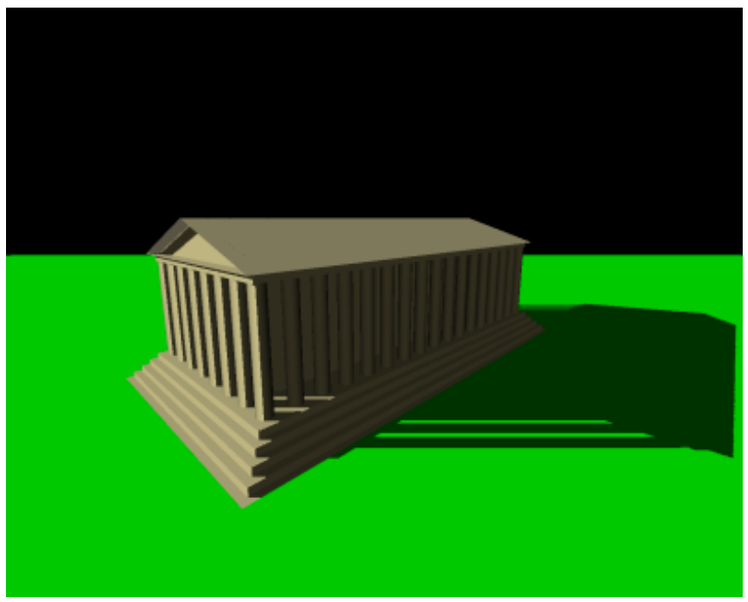


Wikimedia Commons

1. render scene from pov of light and store z-buffer in a texture



2. when rendering scene from desired pov, also render from light pov and test pixel against stored texture

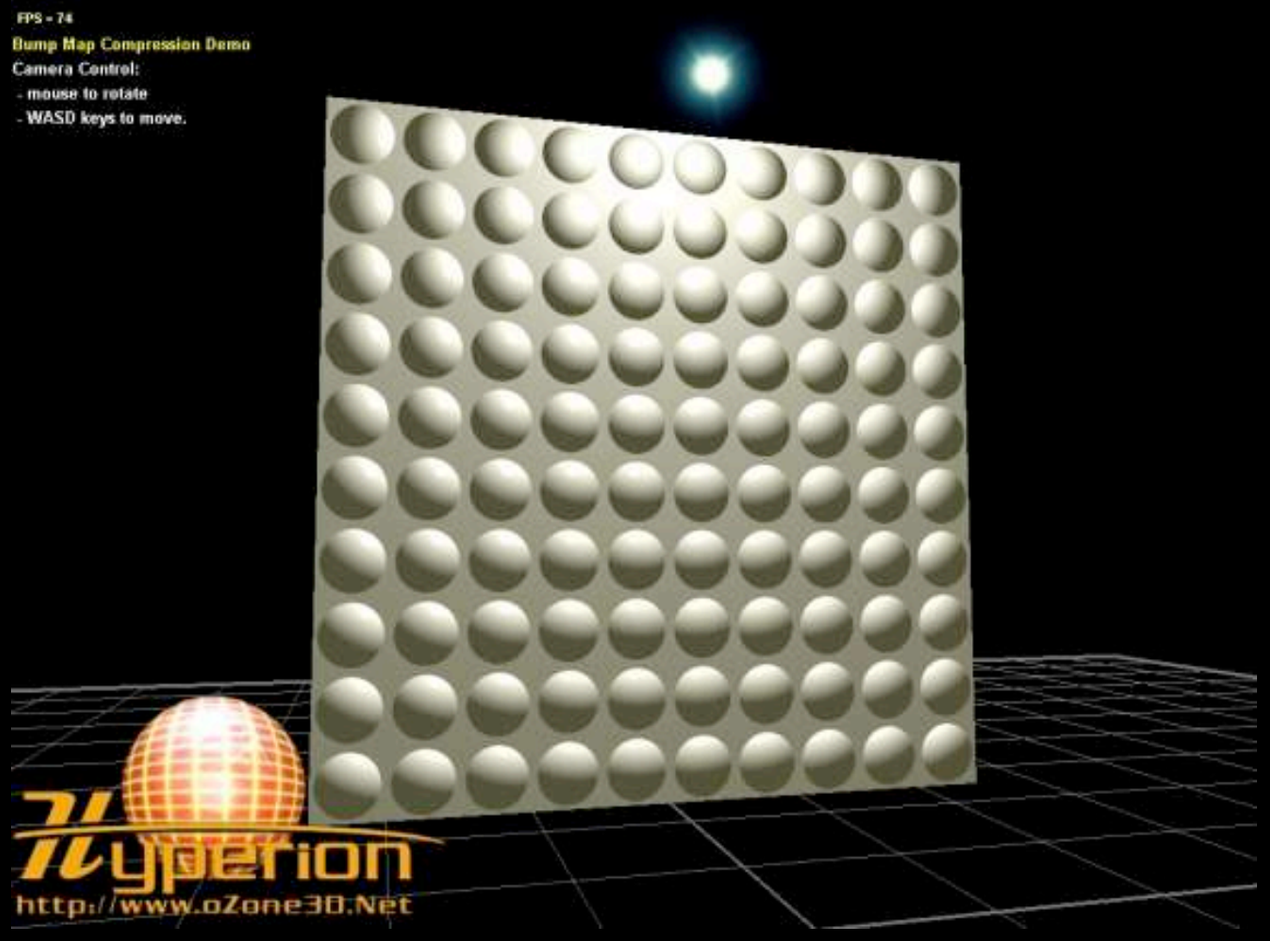


Bump Mapping

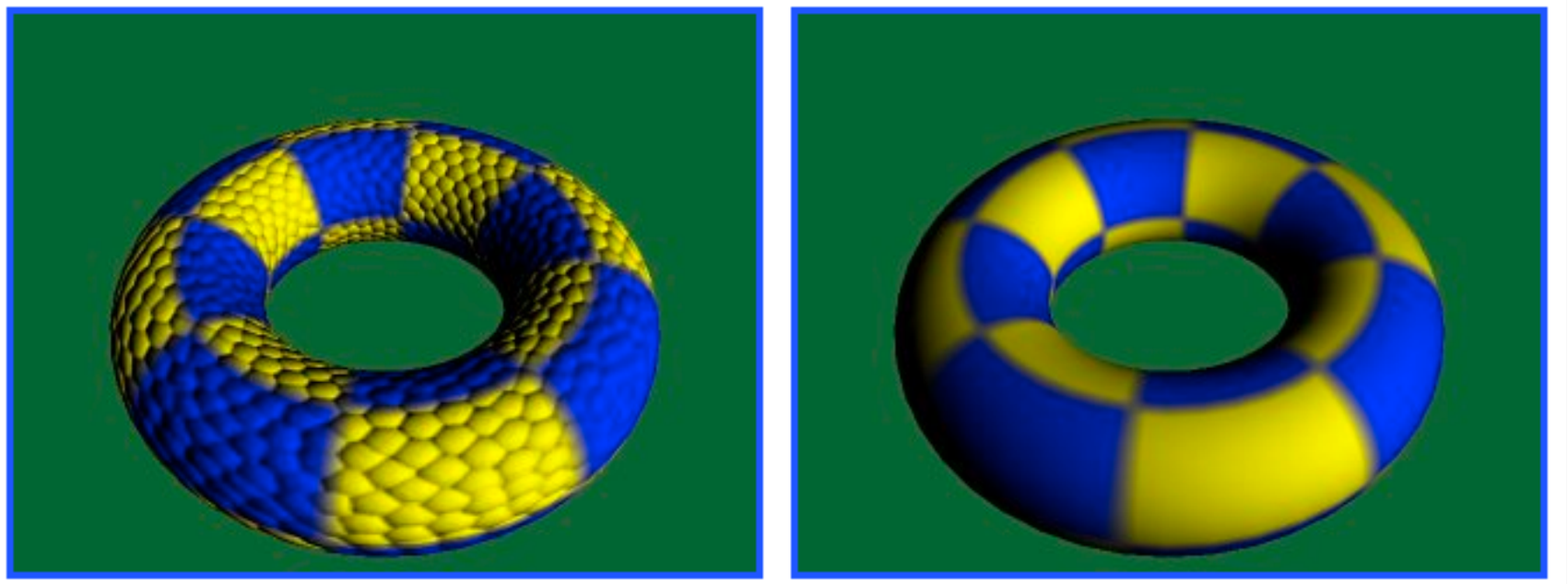
perturb
normal
vectors

doesn't
affect
silhouette

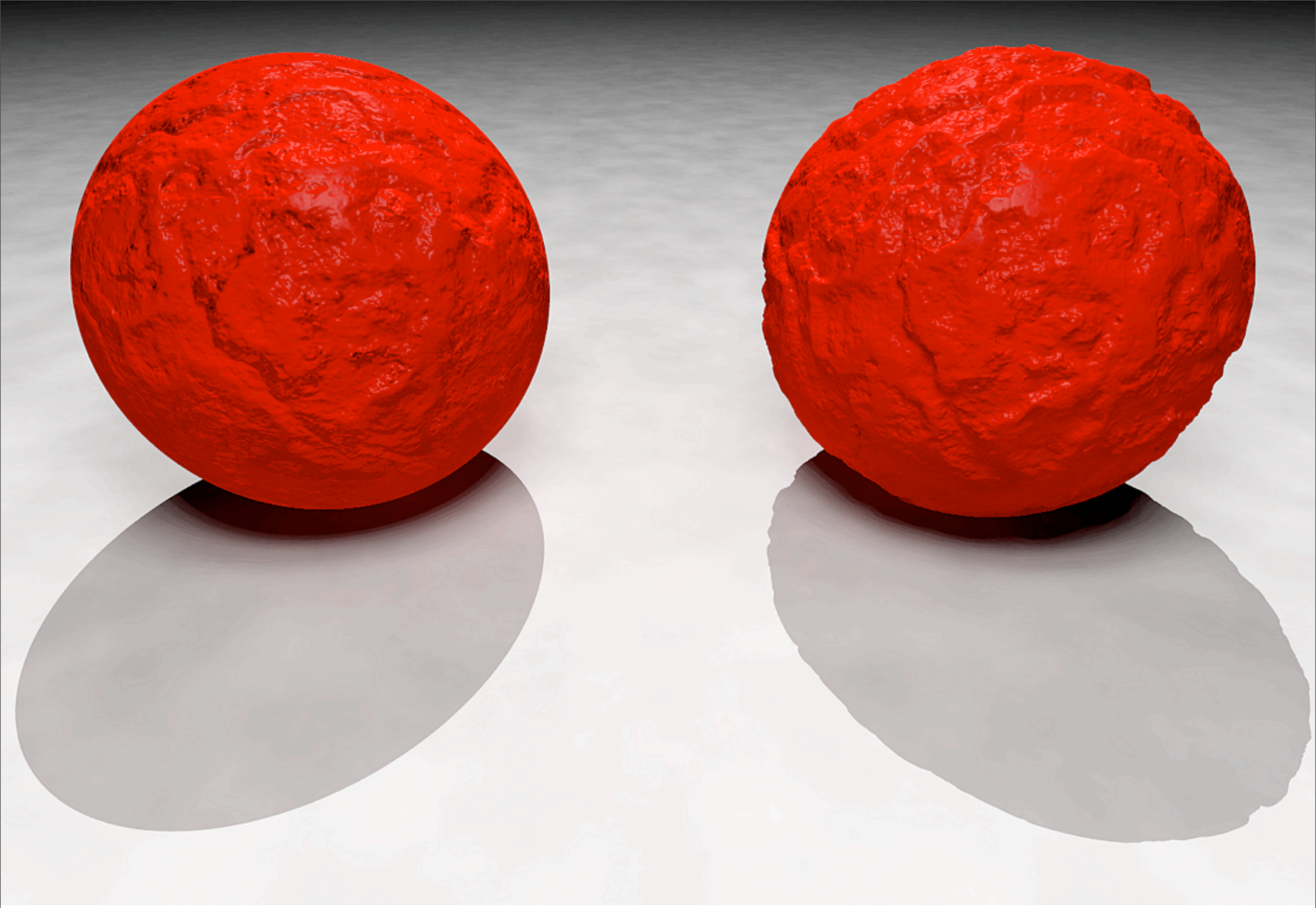
[DirectXTutorial.com]



http://www.lg.clanhost.cz

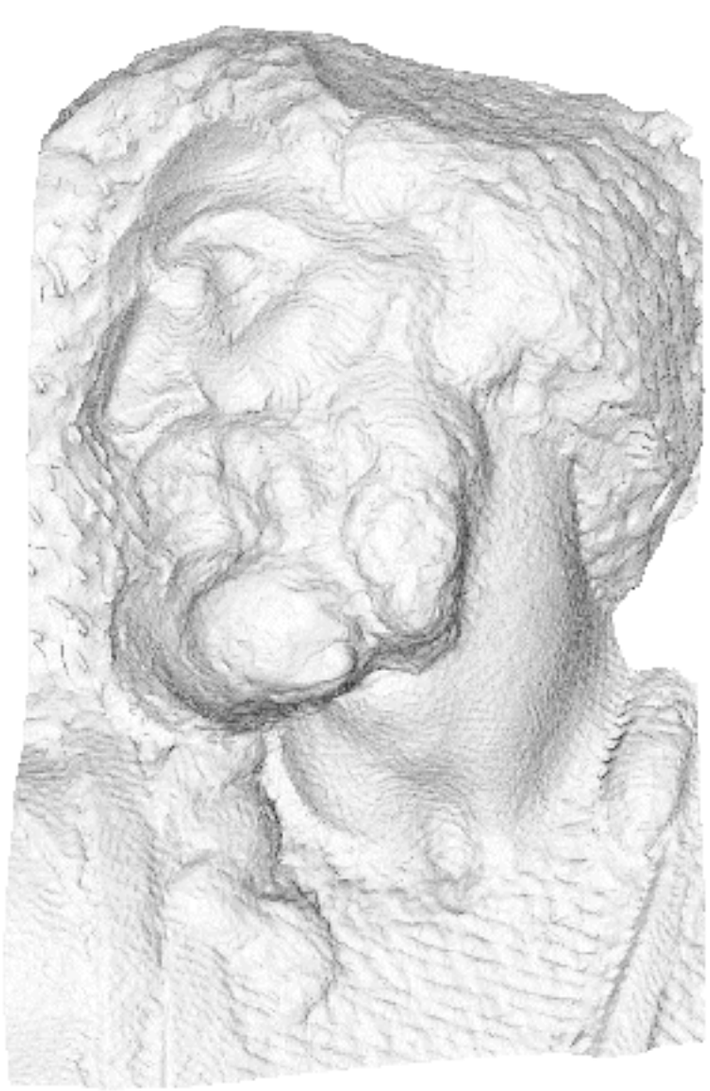


http://www.paulsprojects.net/tutorials/simplebump/simplebump.html

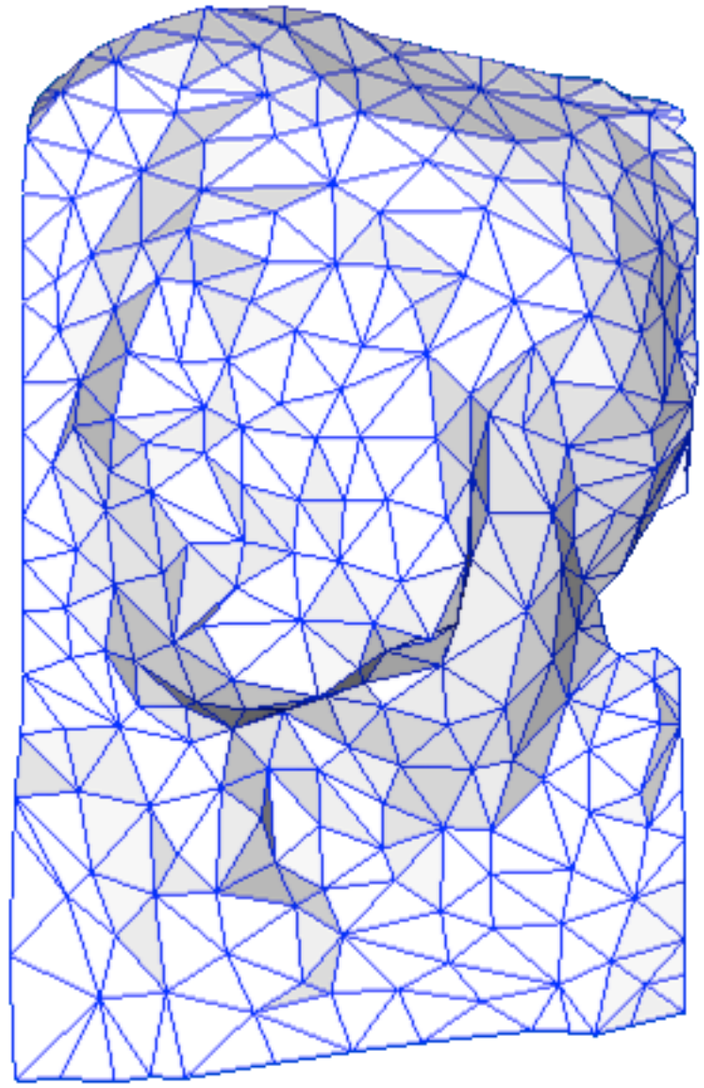


Wikimedia Commons

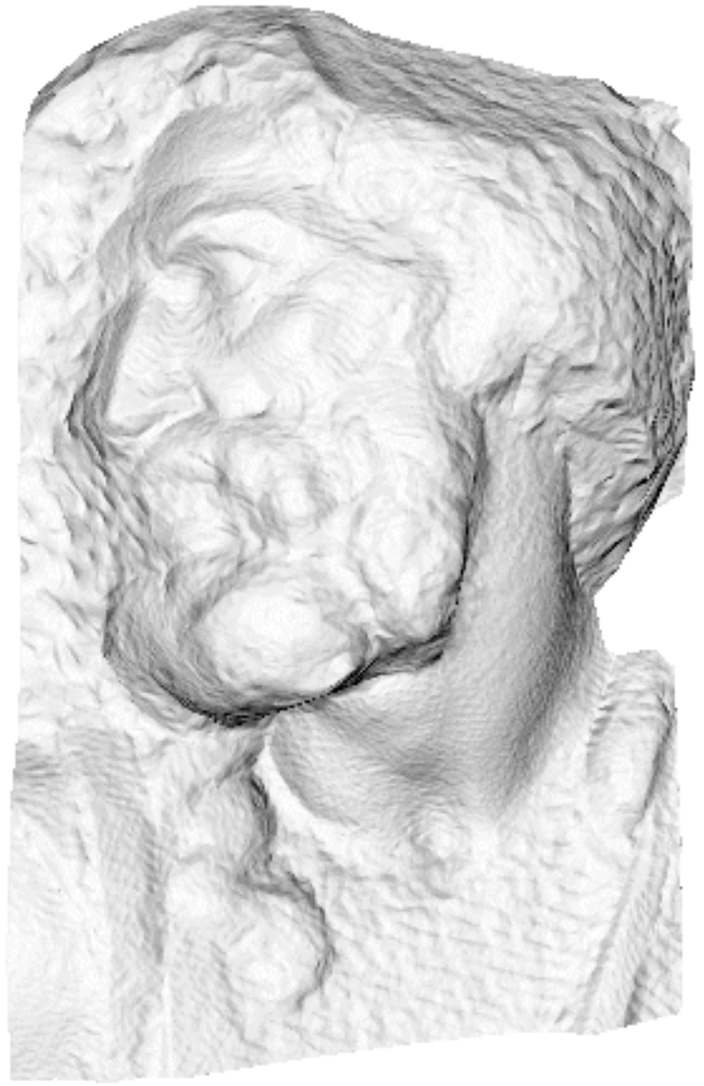
Normal Mapping



original mesh
4M triangles



simplified mesh
500 triangles



simplified mesh
and normal mapping
500 triangles

Wikimedia Commons