

Texture Mapping

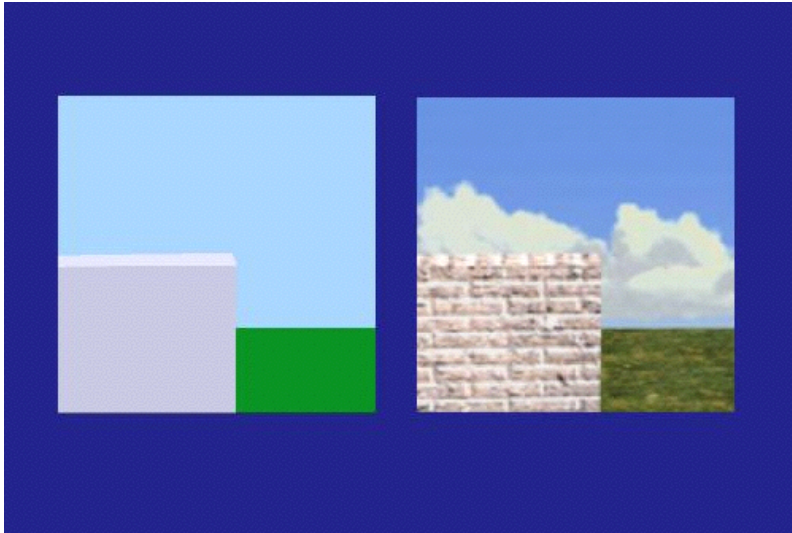
University of California Riverside

Limits of geometric modeling



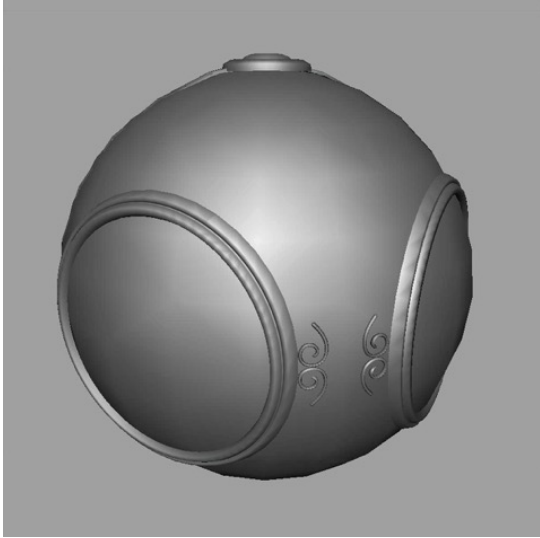
Although modern GPUs can render millions of triangles/sec, that's not enough sometimes...

Texture mapping for detail

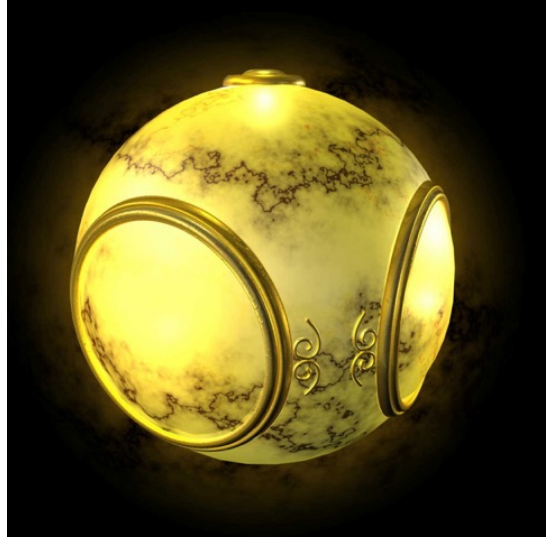


This image contains 8 polygons!

Texture mapping comparison



no texture



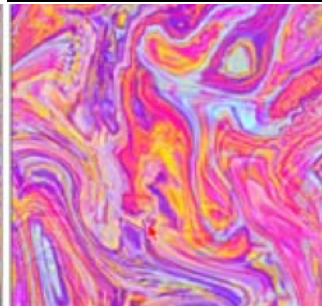
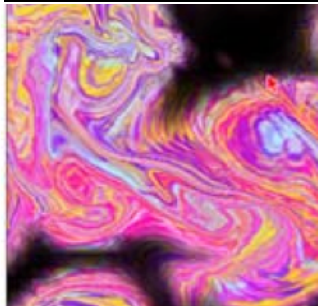
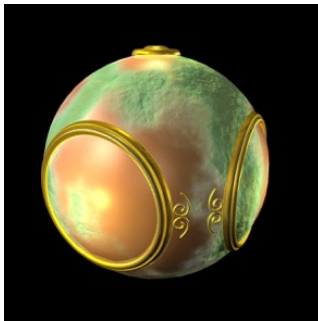
with texture



Pixar - Toy Story

Other uses of textures...

- Light maps
- Shadow maps
- Environment maps
- Bump maps
- Opacity maps
- Animation



Lookup reflectance in image

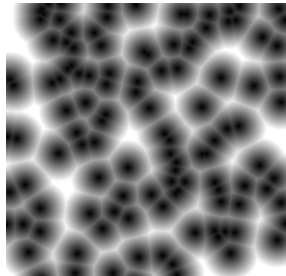
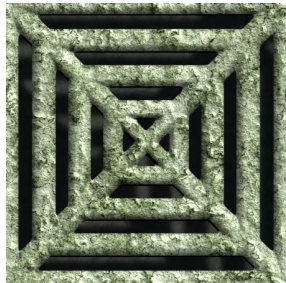
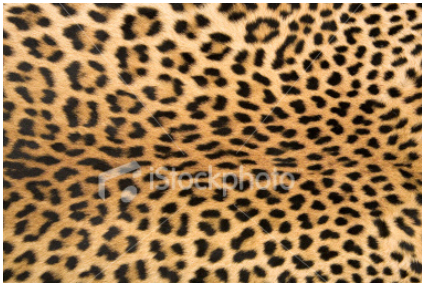
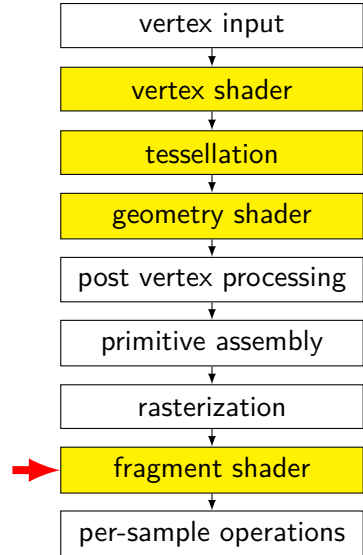


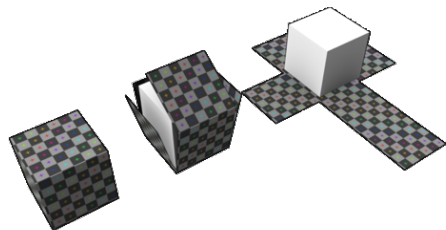
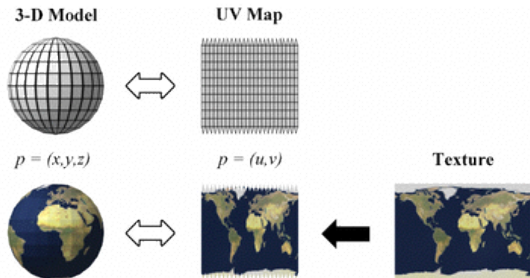
Image source: <https://commons.wikimedia.org/wiki/File:CellularTexture.png> https://commons.wikimedia.org/wiki/File:Procedural_Texture.jpg

Texture mapping in the pipeline

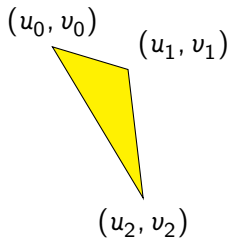
- Geometry and pixels have separate paths through pipeline
- Textures applied in fragment shader
 - End of pipeline
 - Efficient since relatively few polygons get past clipper



uv Mapping

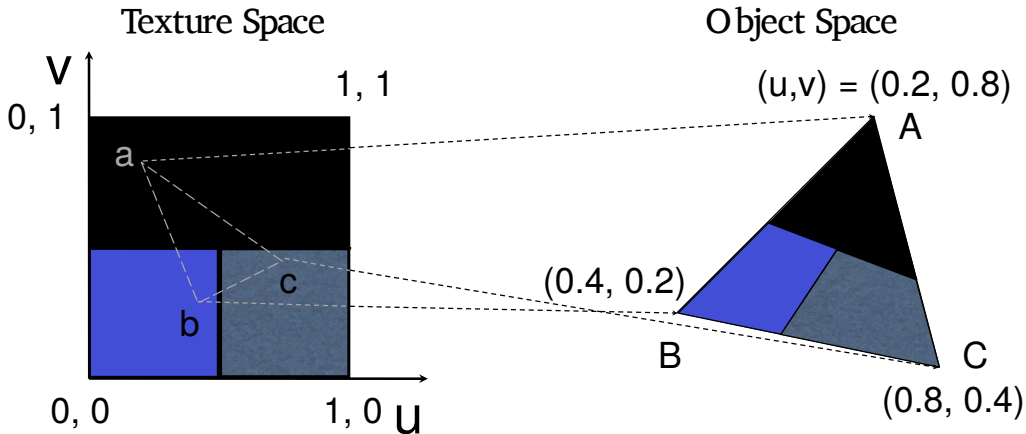


- 2D texture is parameterized by (u, v)
- Assign polygon vertices texture coordinates
- Interpolate within polygon



Texturing triangles

- Store (u, v) at each vertex
- Interpolate inside triangles using barycentric coordinates



Texturing triangles - barycentric coordinates

- Store (u, v) at each vertex
- Interpolate inside triangles using barycentric coordinates

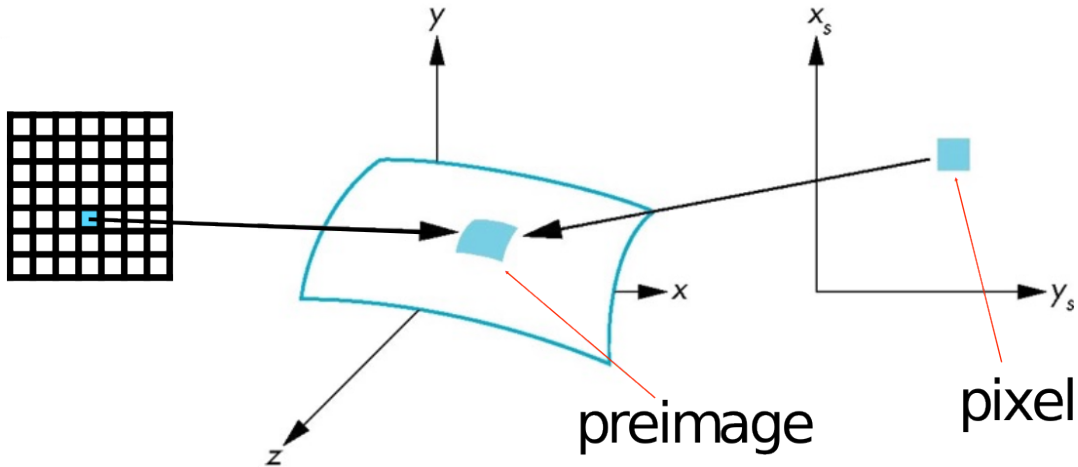
$$\mathbf{p}(\beta, \gamma) = \mathbf{p}_a + \beta(\mathbf{p}_b - \mathbf{p}_a) + \gamma(\mathbf{p}_c - \mathbf{p}_a)$$

$$u(\beta, \gamma) = u_a + \beta(u_b - u_a) + \gamma(u_c - u_a)$$

$$v(\beta, \gamma) = v_a + \beta(v_b - v_a) + \gamma(v_c - v_a)$$

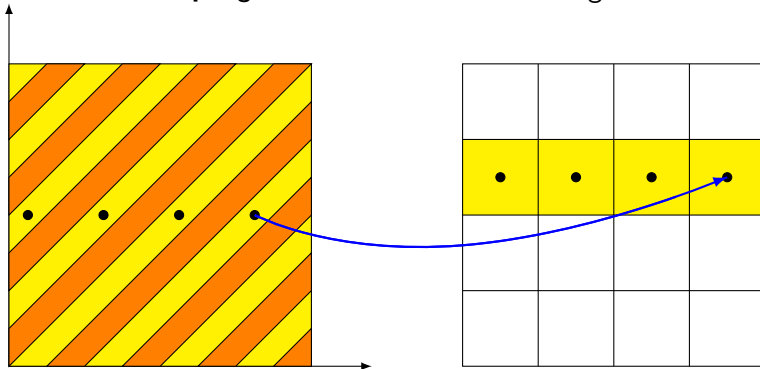
Point sampling

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

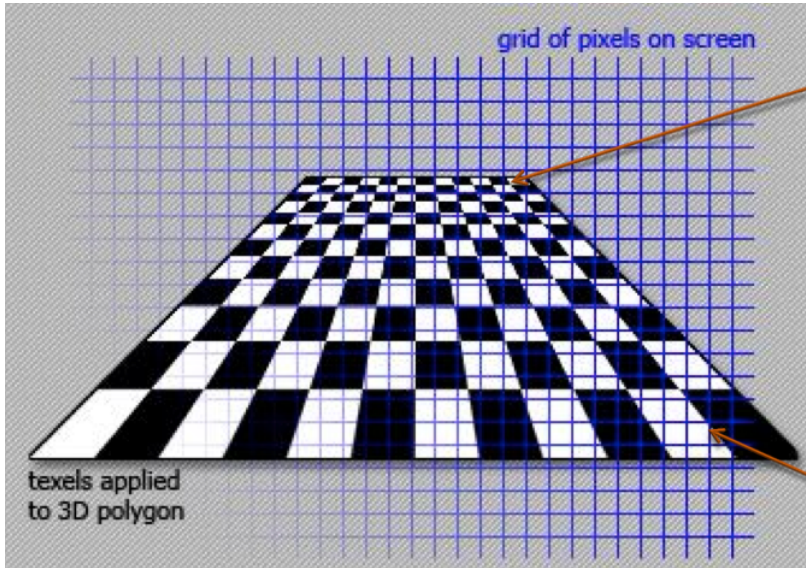


Aliasing

Point sampling textures can lead to aliasing artifacts



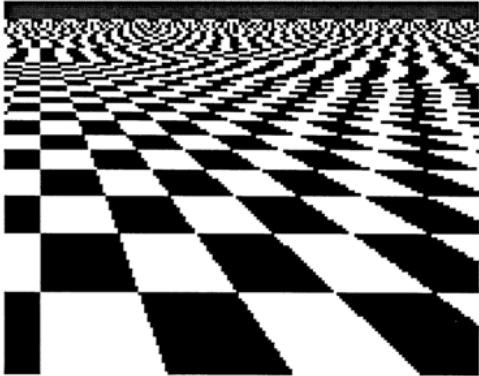
Magnification and minification



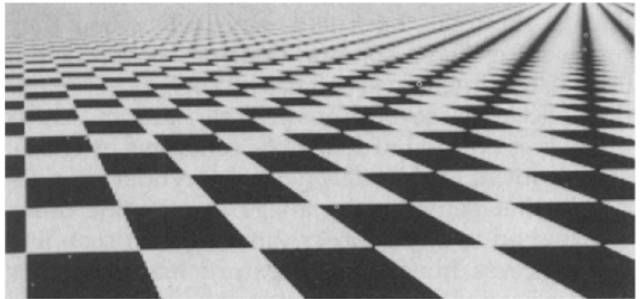
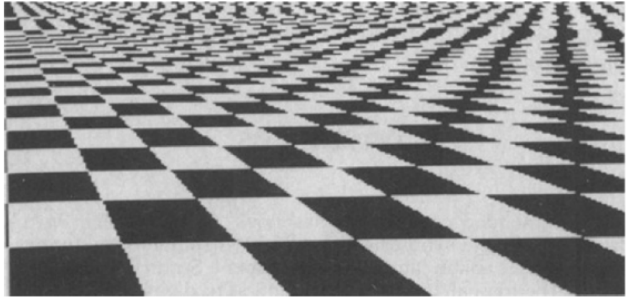
Minification

Magnification

Aliasing artifacts

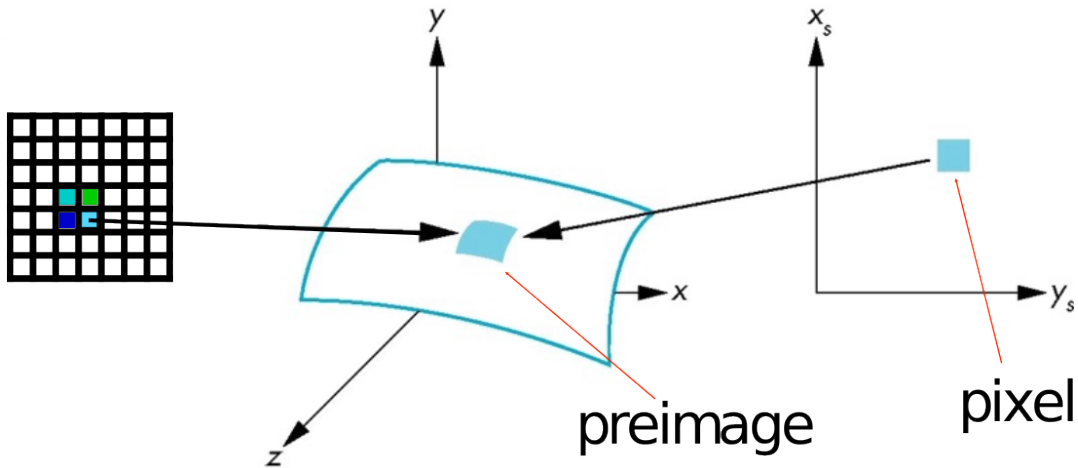


We apply **filtering** to reduce aliasing artifacts

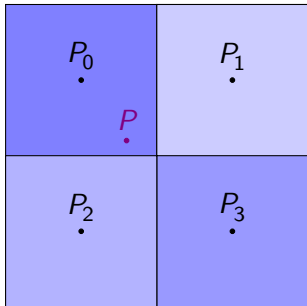


Area averaging

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



Use bilinear filtering



nearest
neighbor



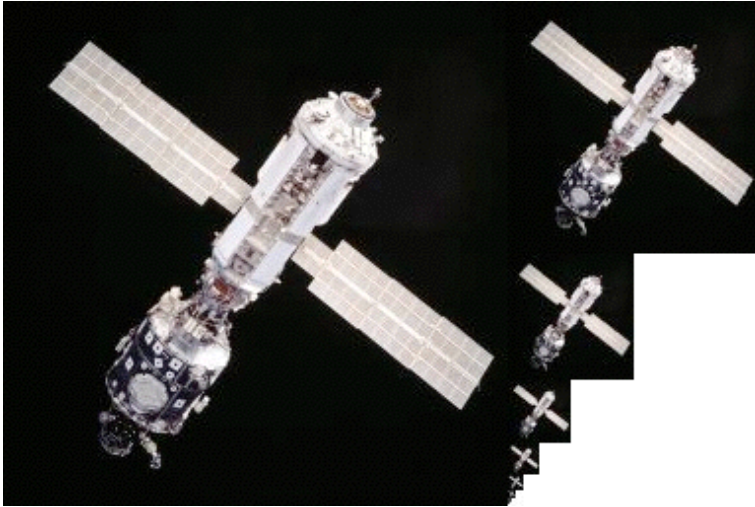
bilinear



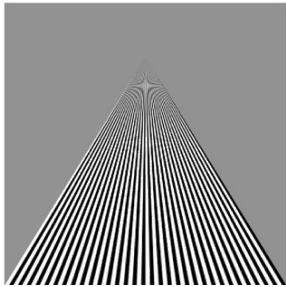
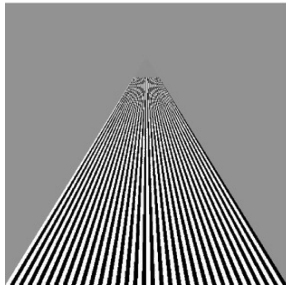
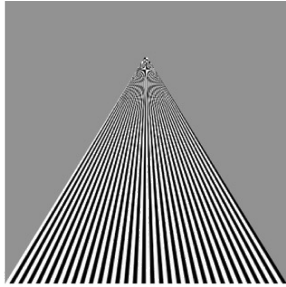
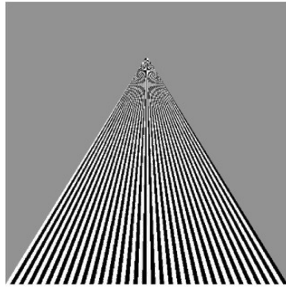
bicubic

mitigate magnification artifacts

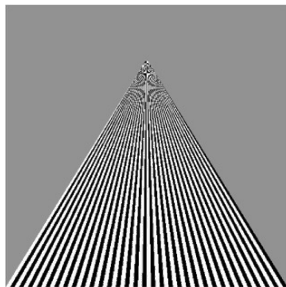
Mipmapping



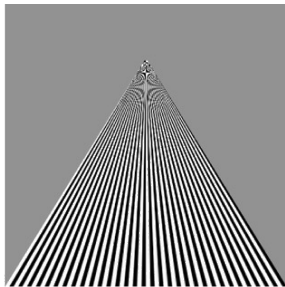
- Reduce minification artifacts
- Prefilter the texture to obtain reduced resolutions
- Requires $\frac{1}{3}$ more space
- Get a texture hierarchy indexed by level



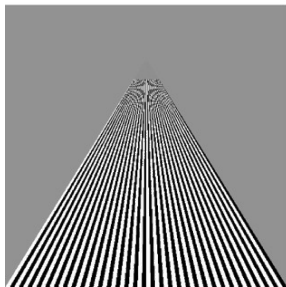
point
sampling



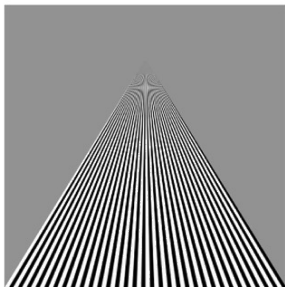
linear
filtering



mipmapped
point
sampling



mipmapped
linear
filtering

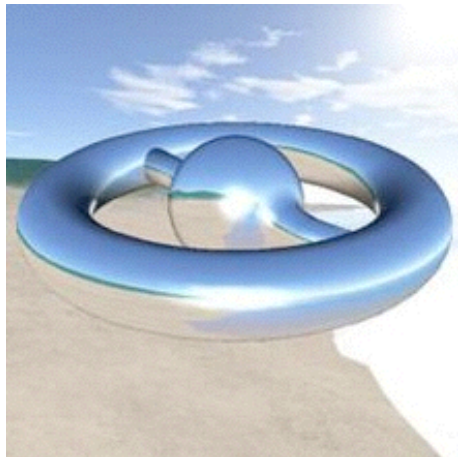
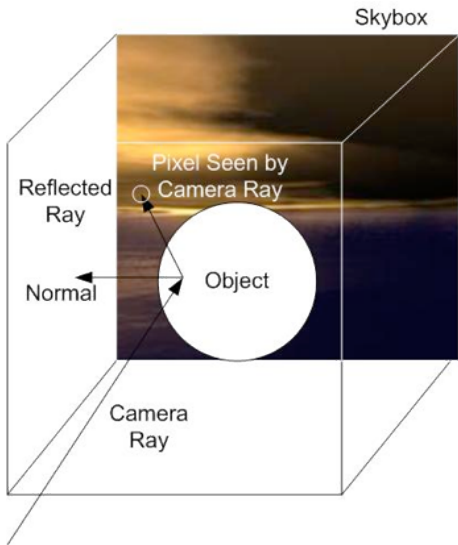


Environment mapping



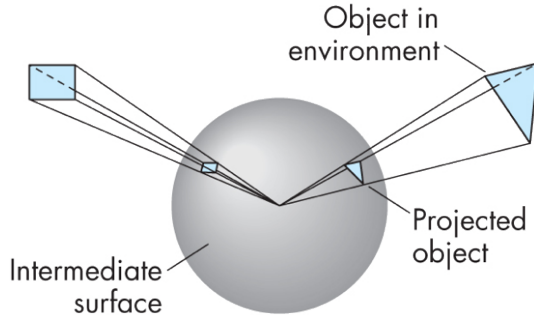
Environment mapping

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



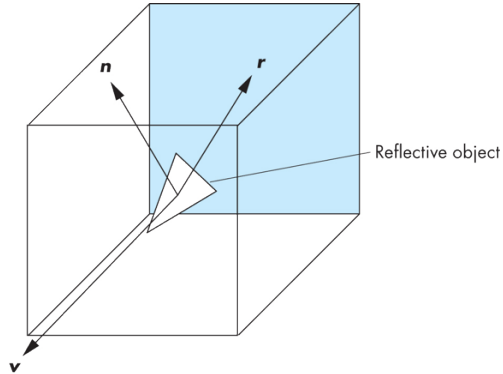
Sphere mapping

- Project objects in the environment onto sphere centered at eye
- Unwrap and store as texture
- Use reflection direction to look up texture value



Cube mapping

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

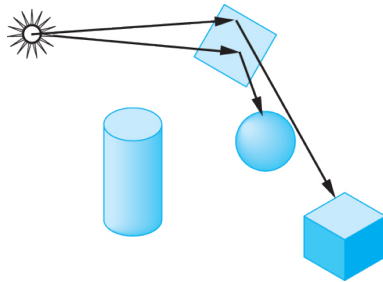


Environment mapping

Create the effect of a mirror with two-pass rendering

First pass: render the scene from the perspective of the mirror

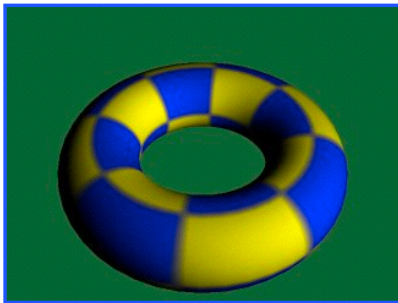
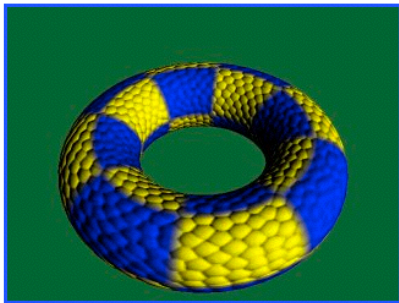
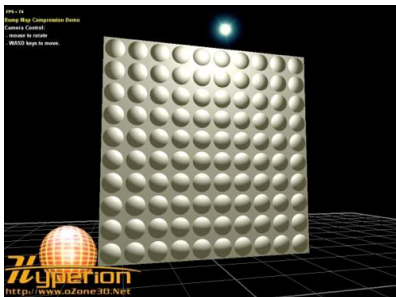
Second pass: render from original pov; use the first image as a texture for the mirror



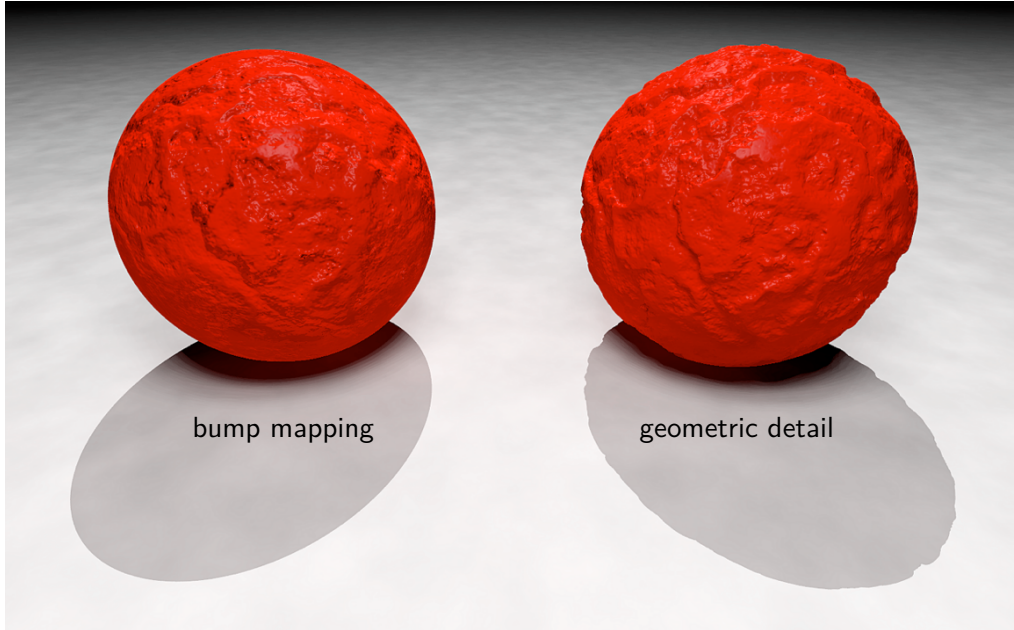
Bump mapping



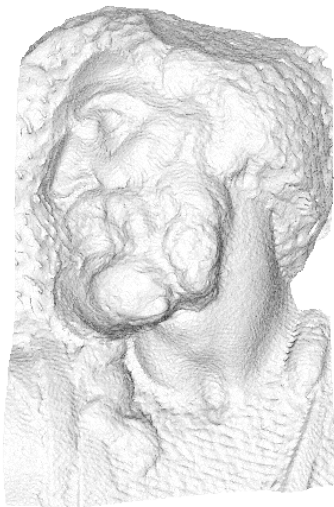
Bump mapping examples



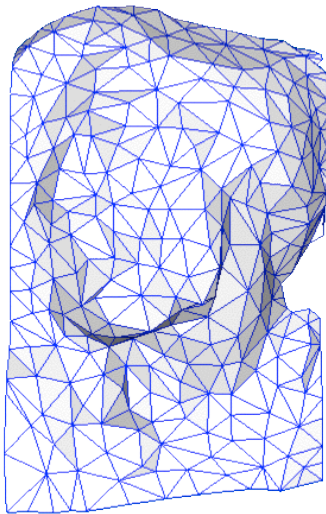
Bump mapping - limitations



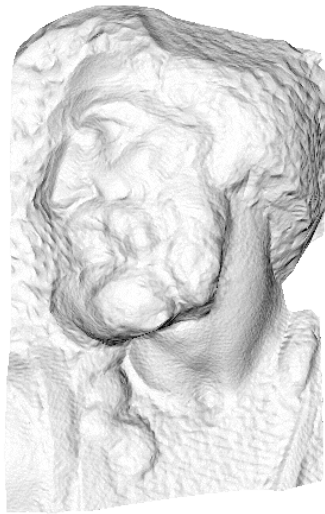
Normal mapping



original mesh
4M triangles



simplified mesh
500 triangles



simplified mesh
and normal mapping
500 triangles