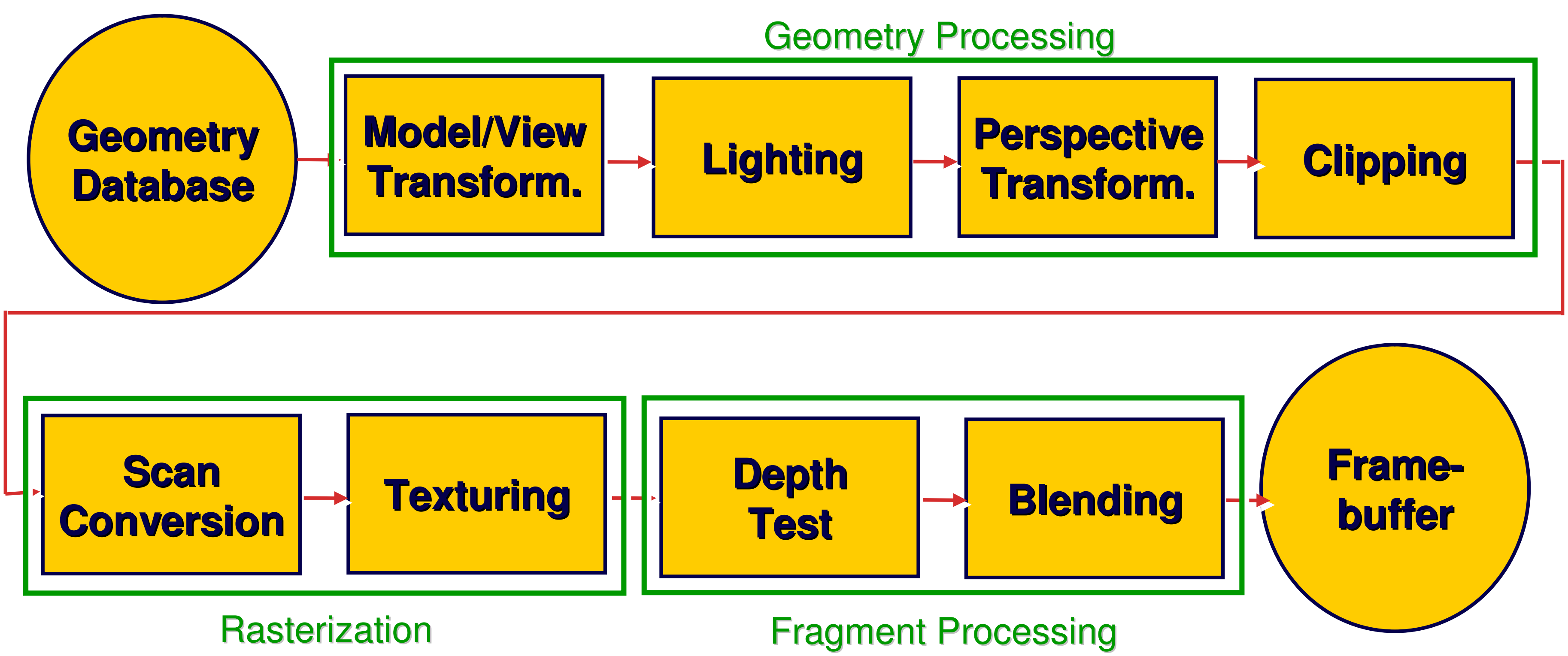
Texture Mapping and Sampling

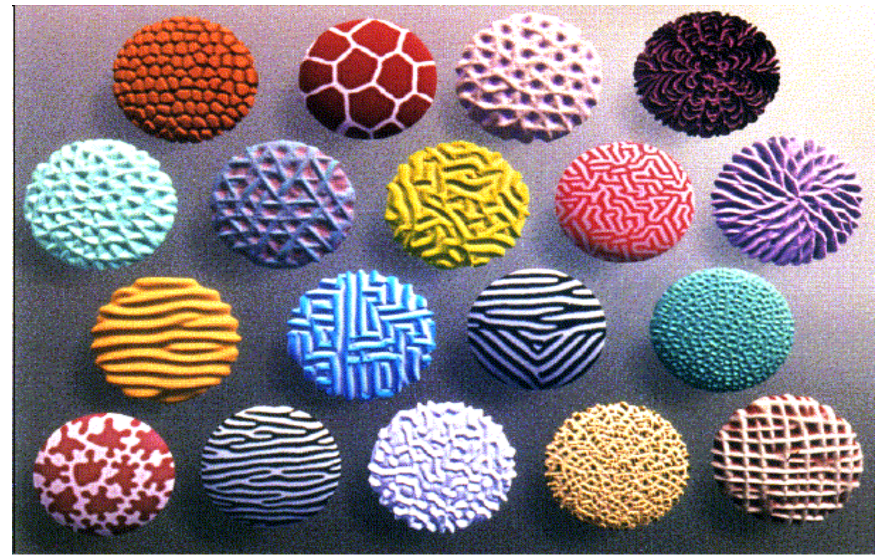
CPSC 314

The Rendering Pipeline



Texture Mapping

- Real life objects have nonuniform colors, normals
- To generate realistic objects, reproduce coloring & normal variations = **texture**
- Can often replace complex geometric details



Texture Mapping

Introduced to increase realism

- Lighting/shading models not enough

Hide geometric simplicity

- Images convey illusion of geometry
- Map a brick wall texture on a flat polygon
- Create bumpy effect on surface

Associate 2D information with 3D surface

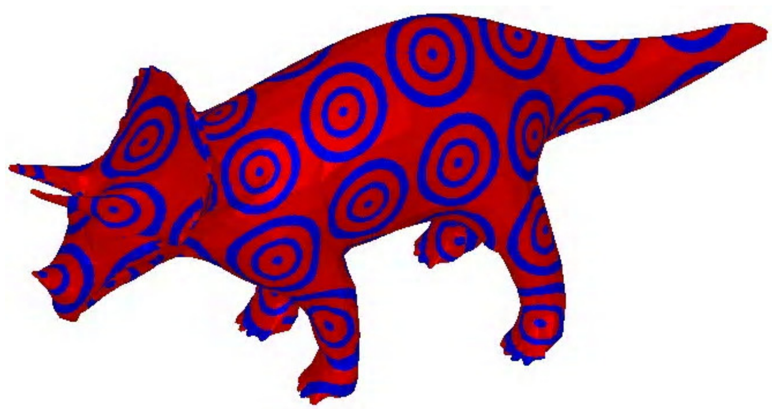
- Point on surface corresponds to a point in texture
- “Paint” image onto polygon

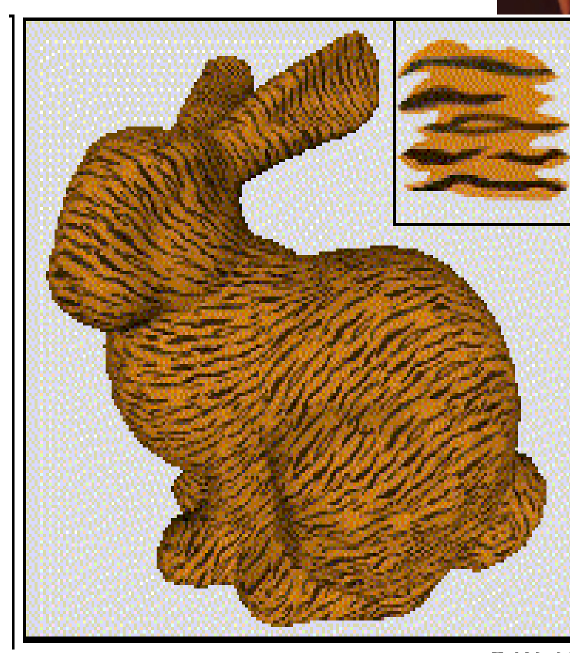
Color Texture Mapping

Define color (RGB) for each pt on surface

Two approaches

- Surface texture map
- Volumetric texture





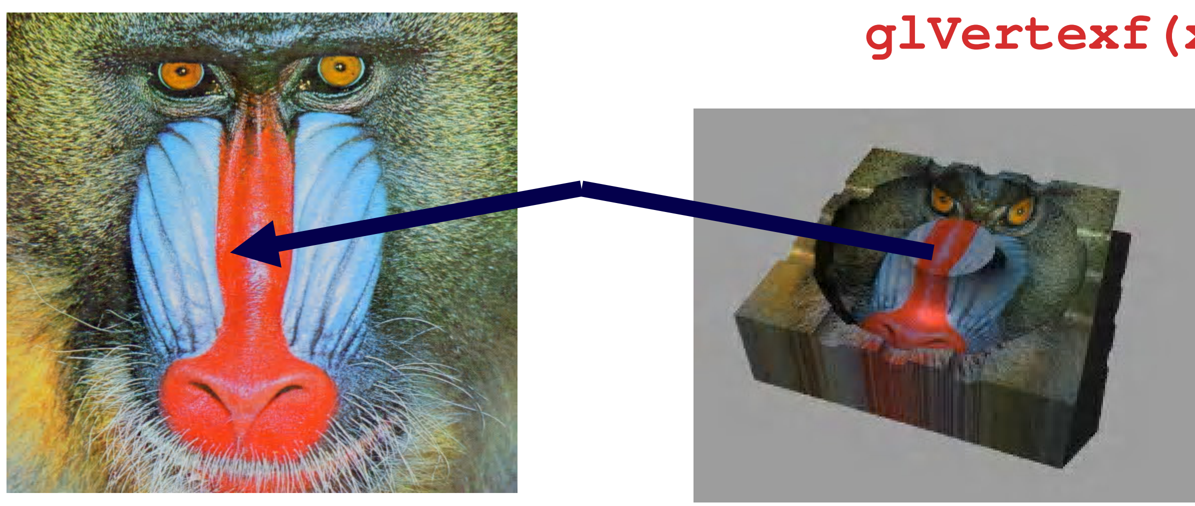
Texture Coordinates

Texture image: 2D array of color values (texels)

Assigning texture coordinates (s,t) at vertex with object coordinates (x,y,z,w)

- Use interpolated (s,t) for texel lookup at each pixel
- Use value to modify a polygon's color
  - *Or other surface property*
- Specified by programmer or artist

```
glTexCoord2f(s,t)
glVertexf(x,y,z,w)
```



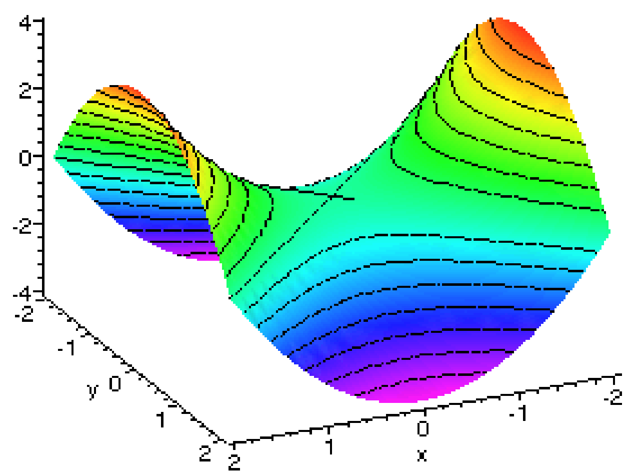
l. van de Panne

Texture Mapping

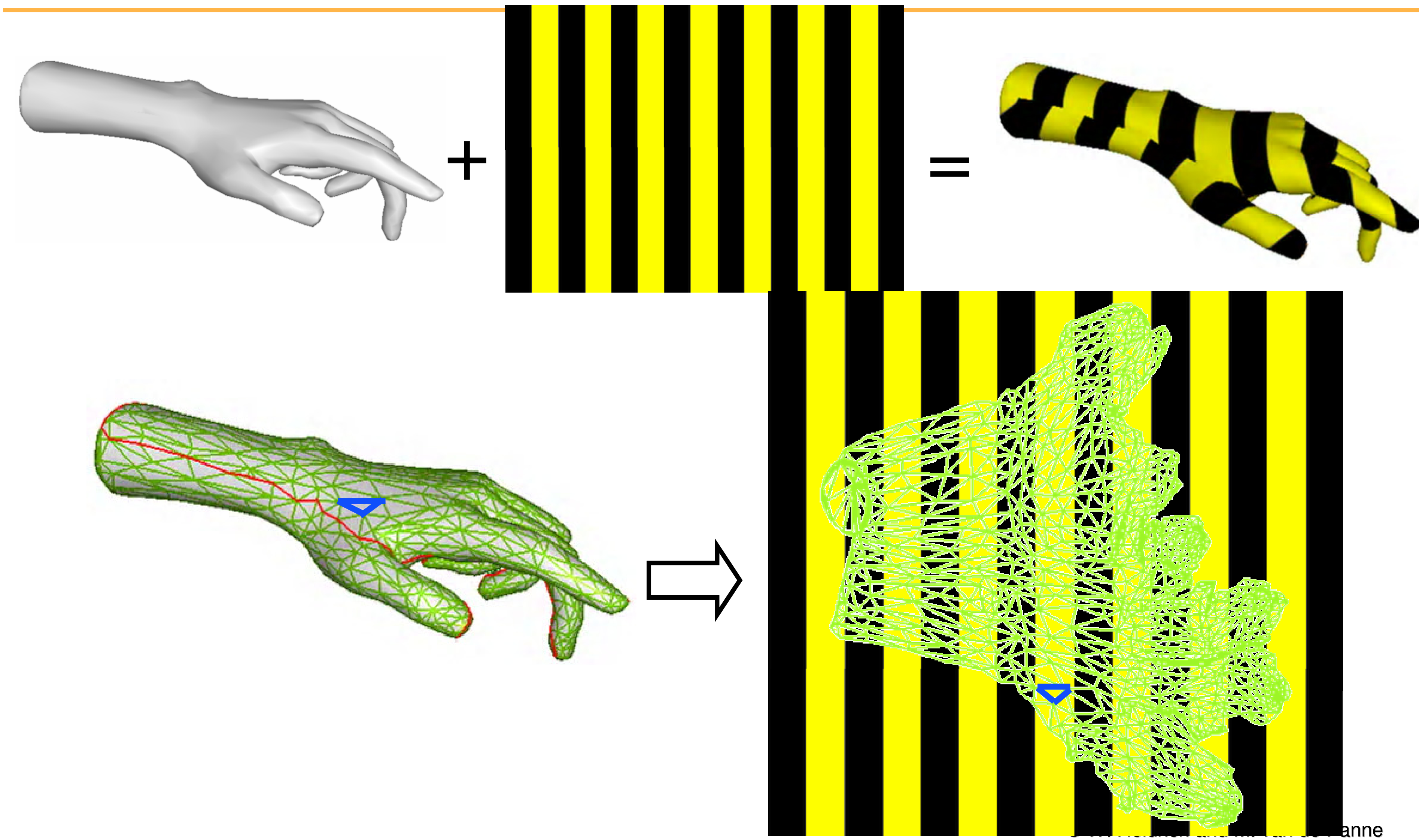
Textures of other dimensions

- 1D: represent isovalues
  - *e.g.: contour lines, temp, ...*

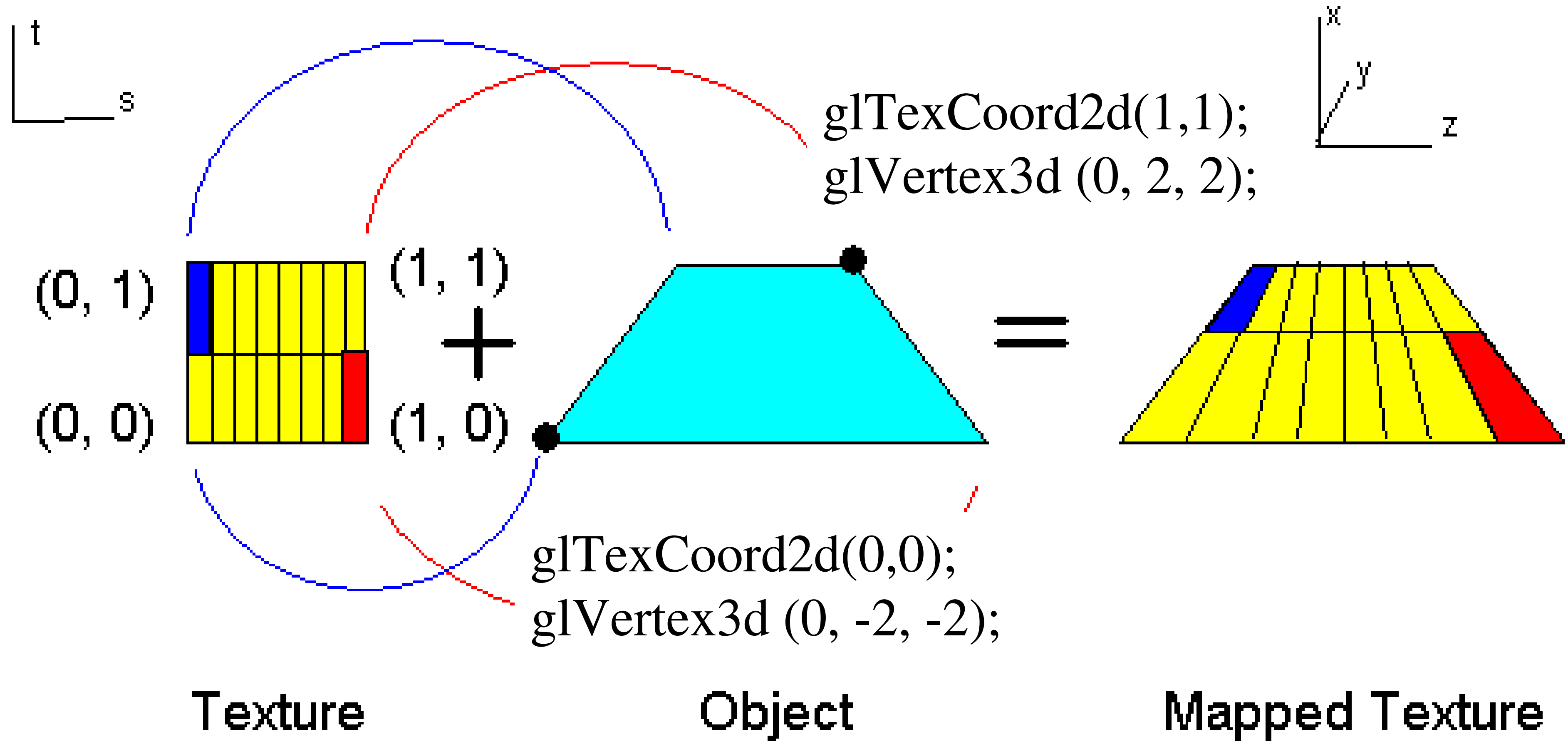
    `glTexCoord1f(s)`



Texture Mapping Example

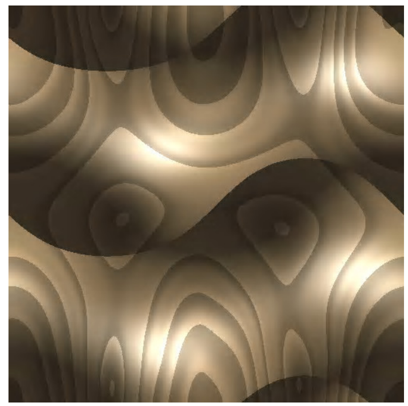


Example Texture Map

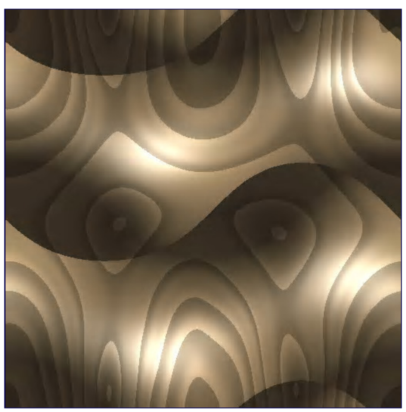


Fractional Texture Coordinates

texture image

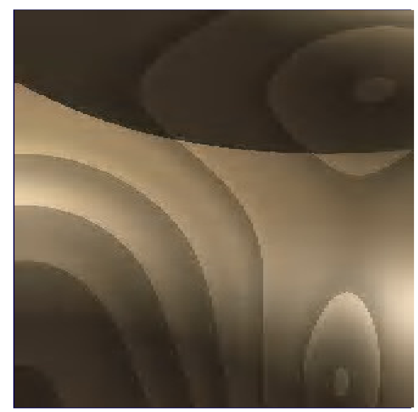


(0,1) (1,1)



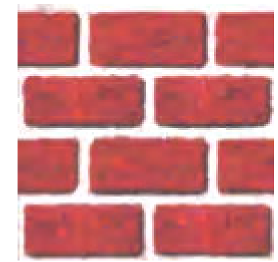
(0,0) (1,0)

(0,.5) (.25,.5)



(0,0) (.25,0)

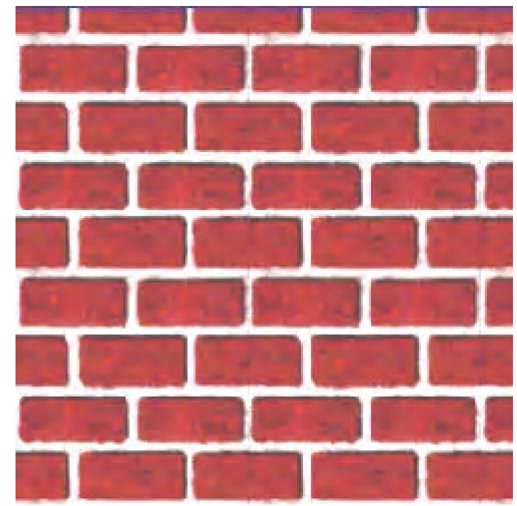
Texture Lookup: Tiling and Clamping



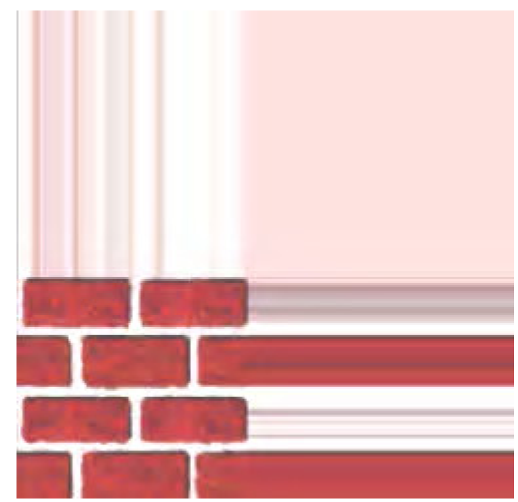
What if s or t is outside the interval [0…1]?

Multiple choices

- Use fractional part of texture coordinates
  - *Cyclic repetition*
    glTexParameteri(…, GL_TEXTURE_WRAP_S, GL_REPEAT, GL_TEXTURE_WRAP_T, GL_REPEAT, ...)

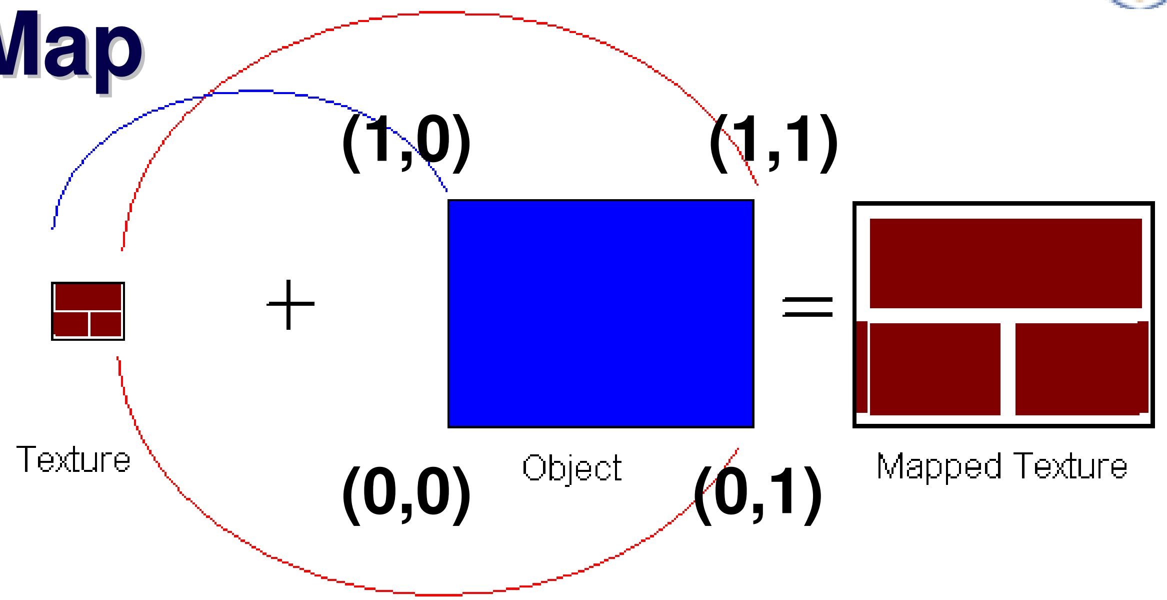


- Clamp every component to range [0…1]
  - *Re-use color values from texture image border glTexParameteri(…, GL_TEXTURE_WRAP_S, GL_CLAMP, GL_TEXTURE_WRAP_T, GL_CLAMP, ...)*

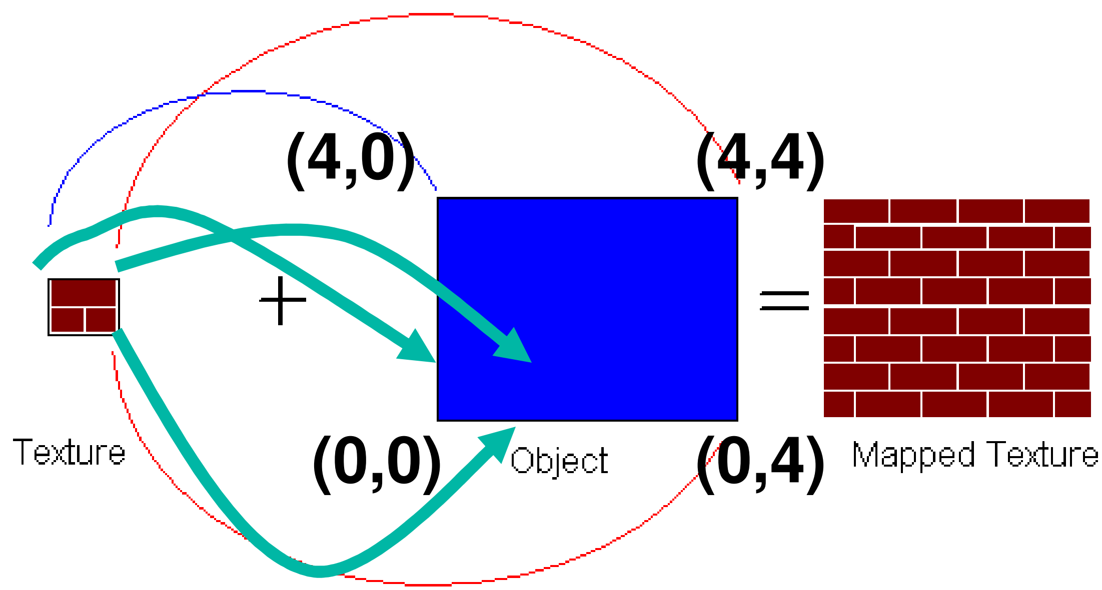


Tiled Texture Map

```
glTexCoord2d(1, 1);
glVertex3d (x, y, z);
```




```
glTexCoord2d(4, 4);
glVertex3d (x, y, z);
```



Texture Coordinate Transformation

Motivation

- Change scale, orientation of texture on an object

Approach

- *Texture matrix stack*
- Transforms specified (or generated) tex coords

```
glMatrixMode( GL_TEXTURE );
glLoadIdentity();
glRotate();
   …
```

- More flexible than changing (s,t) coordinates

Texture Functions

Ways of applying texture colour:

- `GL_REPLACE`
  - *Directly use as surface color; no lighting effects*
- `GL_MODULATE`
  - *Modulate surface color: multiply old color by new value*
  - *Texturing happens after lighting, not relit*
- `GL_DECAL`
  - *Like replace, but modulate alpha to support transparency*
- `GL_BLEND`
  - *Blend surface color with existing on-screen colour*

```
glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, <mode>)
```

Texture Pipeline

(s, t) Parameter space (0.32, 0.29) → (s', t') Transformed parameter space (0.52, 0.49) → Texel space (81, 74) → Texel color (0.9,0.8,0.7)

Object color (0.5,0.5,0.5)

Object color + Texel color → Final color (0.45,0.4,0.35)

Texture Objects and Binding

Texture object

- An OpenGL data type that keeps textures resident in memory and provides identifiers to easily access them
- Provides efficiency gains over having to repeatedly load and reload a texture
- You can prioritize textures to keep in memory
- OpenGL uses least recently used (LRU) if no priority is assigned

Texture binding

- Which texture to use right now
- Switch between preloaded textures

Basic OpenGL Texturing

Create a texture object and fill it with texture data:

- `glGenTextures(num, &indices)` to get identifiers for the objects
- `glBindTexture(GL_TEXTURE_2D, identifier)` to bind
  - *Following texture commands refer to the bound texture*
- `glTexParameteri(GL_TEXTURE_2D, …, …)` to specify parameters for use when applying the texture
- `glTexImage2D(GL_TEXTURE_2D, ….)` to specify the texture data (the image itself)

Basic OpenGLTexturing (cont.)

Enable texturing:

- `glEnable(GL_TEXTURE_2D)`

State how the texture will be used:

- `glTexEnvf(…)`

Specify texture coordinates for the polygon:

- Use `glTexCoord2f(s,t)` before each vertex:
  - *`glTexCoord2f(0,0);`*
    `glVertex3f(x,y,z);`

Low-Level Details

Large range of functions for controlling layout of texture data

- State how the data in your image is arranged
- e.g.: `glPixelStorei(GL_UNPACK_ALIGNMENT, 1)` tells OpenGL not to skip bytes at the end of a row
- You must state how you want the texture to be put in memory: how many bits per "pixel", which channels,…

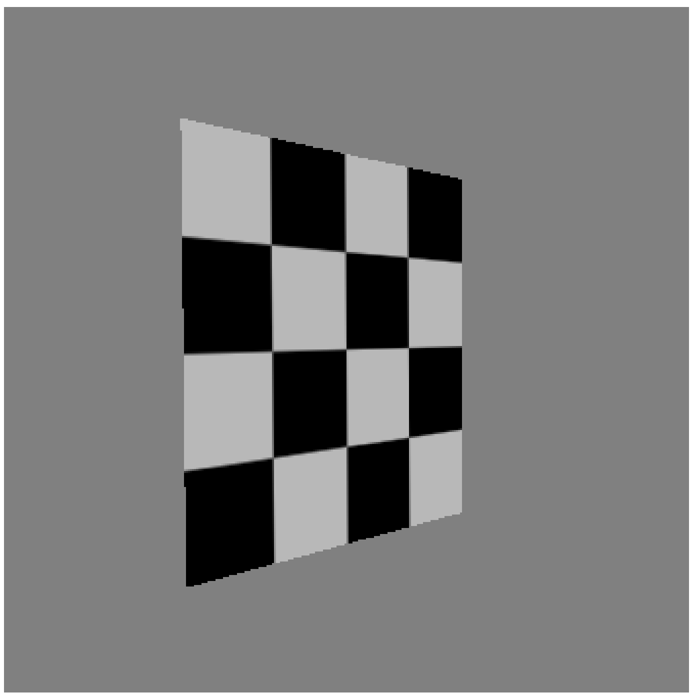
Textures must have a size of power of 2

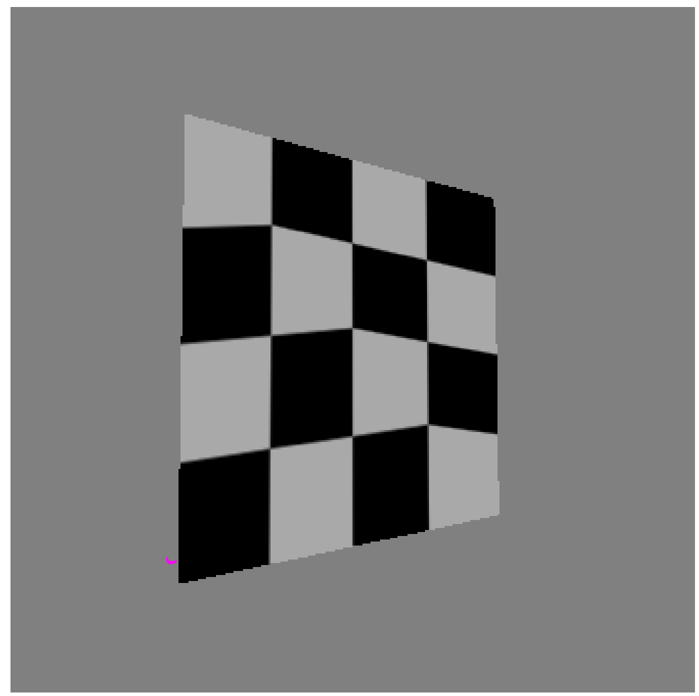
- Common sizes are 32x32, 64x64, 256x256
- But don't need to be square, i.e. 32x64 is fine
- Smaller uses less memory, and there is a finite amount of texture memory on graphics cards

Texture Mapping

Texture coordinate interpolation

- Perspective foreshortening problem

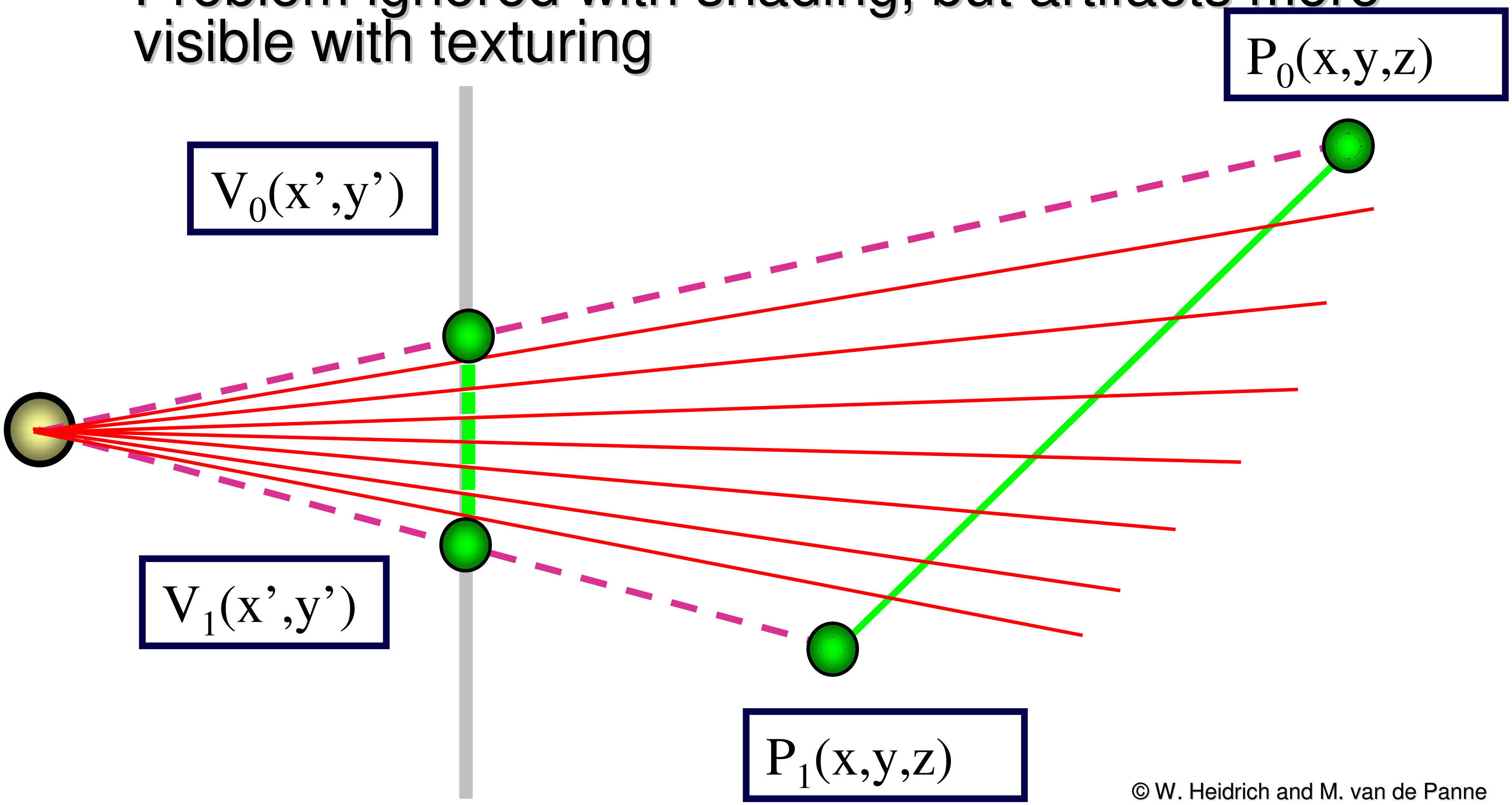




Interpolation: Screen vs. World Space

Screen space interpolation incorrect

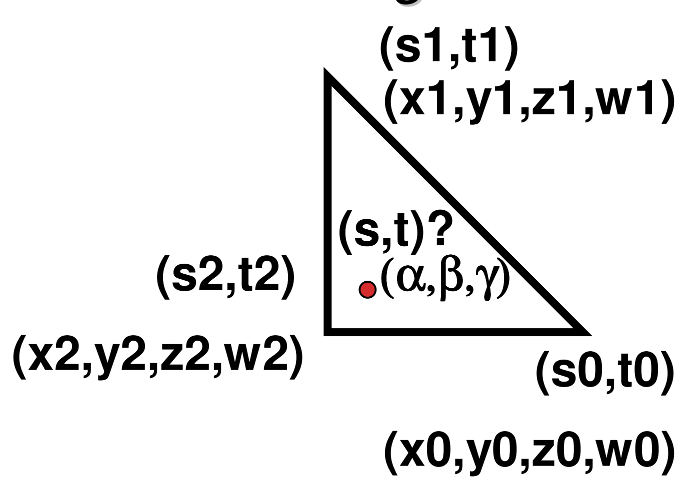
- Problem ignored with shading, but artifacts more visible with texturing



Texture Coordinate Interpolation

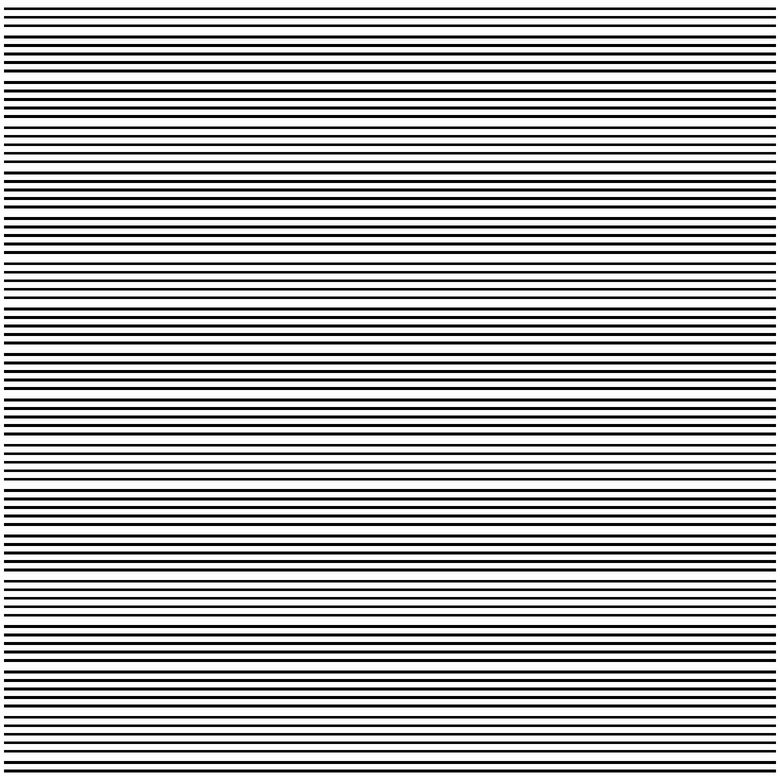
Perspective correct interpolation

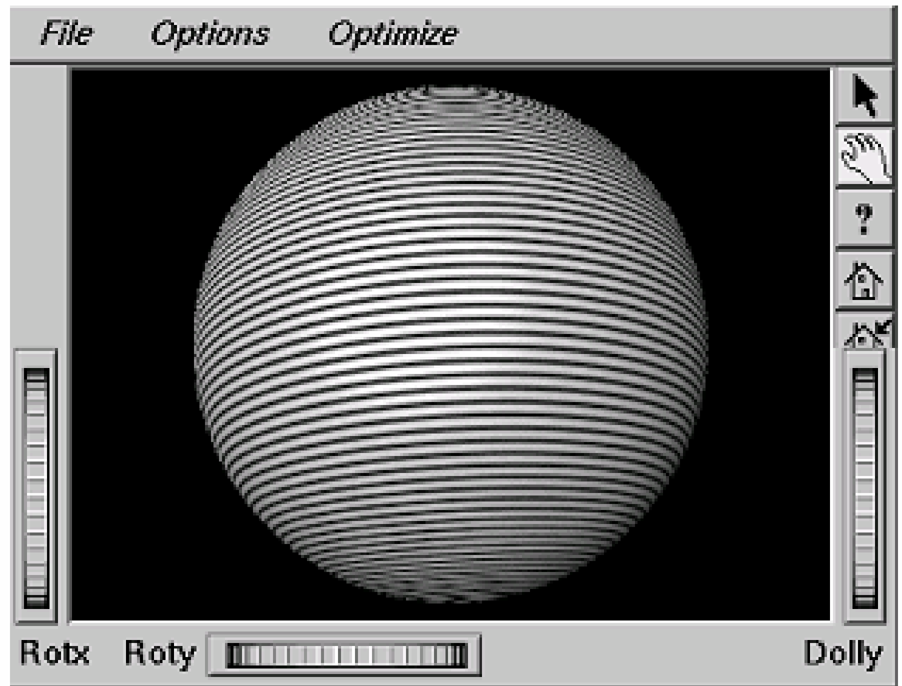
- α, β, γ :
  - *Barycentric coordinates of a point **P** in a triangle*
- *s0, s1, s2* :
  - *Texture coordinates of vertices*
- *w0, w1,w2* :
  - *Homogeneous coordinates of vertices*



$$s = \frac{\alpha \cdot s_0 / w_0 + \beta \cdot s_1 / w_1 + \gamma \cdot s_2 / w_2}{\alpha / w_0 + \beta / w_1 + \gamma / w_2}$$

Reconstruction

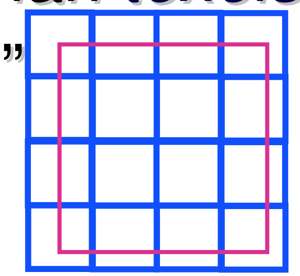




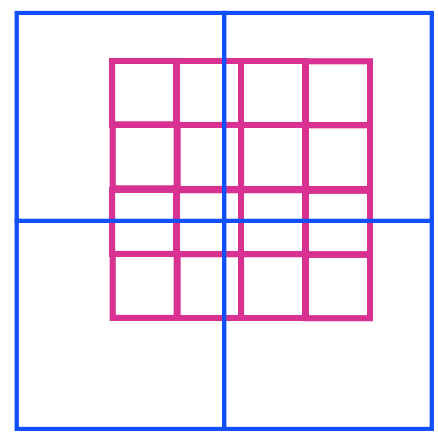
(image courtesy of Kiriakos Kutulakos, U Rochester)

Reconstruction

- How to deal with:
  - *Pixels that are much larger than texels?*
    - Apply filtering, “averaging”



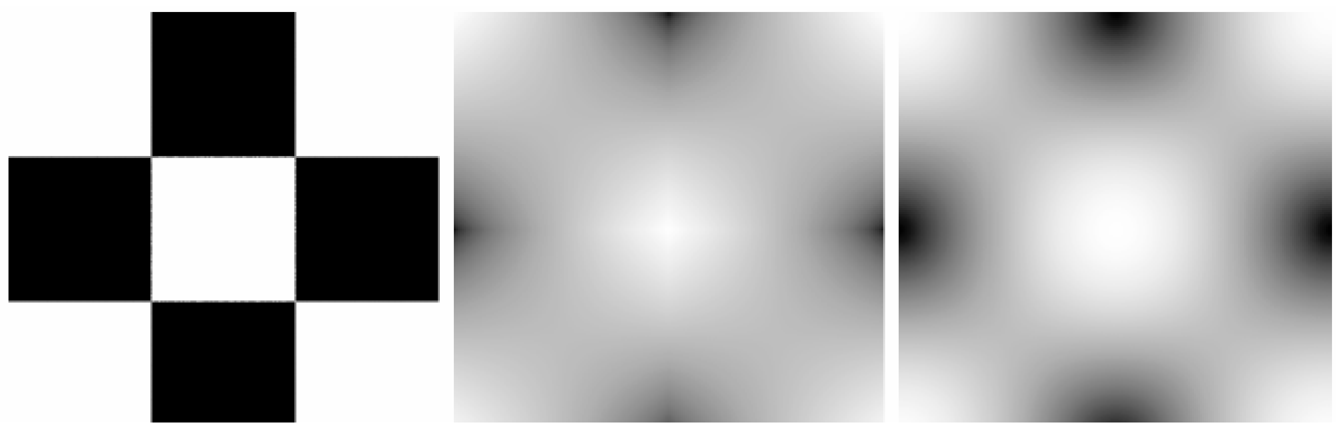
  - *Pixels that are much smaller than texels ?*
    - Interpolate



Interpolating Textures

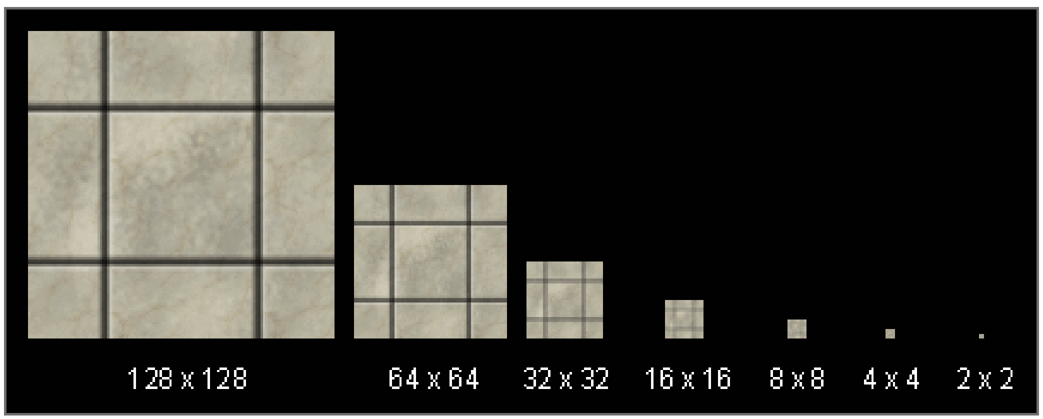
- Nearest neighbor
- Bilinear
- Hermite

1	0	1
0	1	0
1	0	1

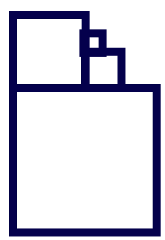


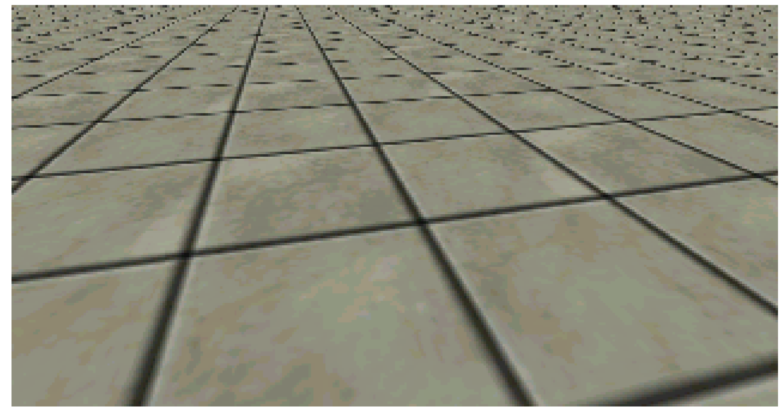
MIPmapping

use “image pyramid” to precompute averaged versions of the texture

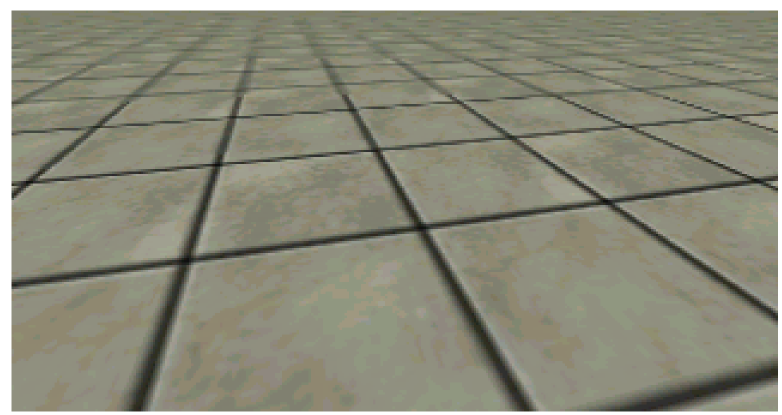


store whole pyramid in single block of memory





Without MIP-mapping



With MIP-mapping

MIPmaps

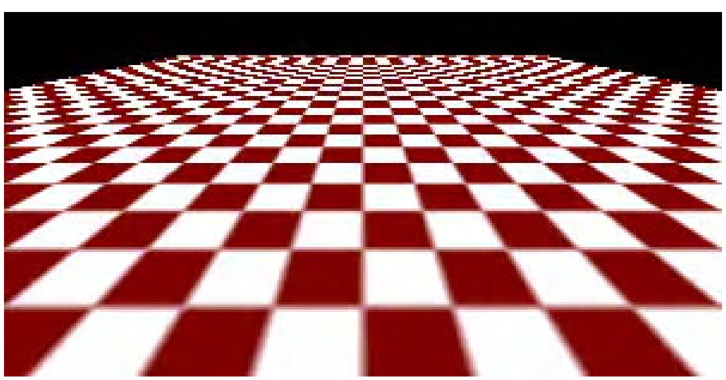
Multum in parvo ***-- many things in a small place***

- Prespecify a series of prefiltered texture maps of decreasing resolutions
- Requires more texture storage
- Avoid shimmering and flashing as objects move

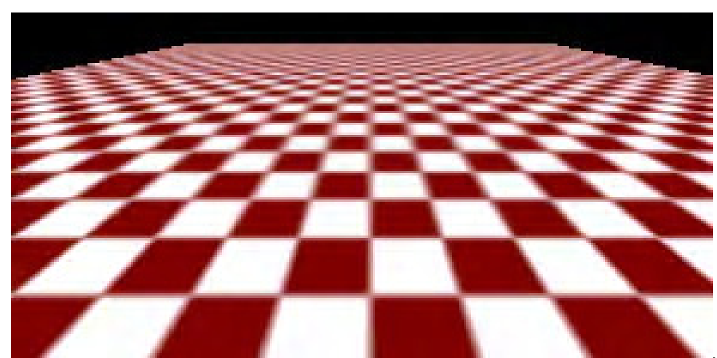
gluBuild2DMipmaps

- Automatically constructs a family of textures from original texture size down to 1x1

without

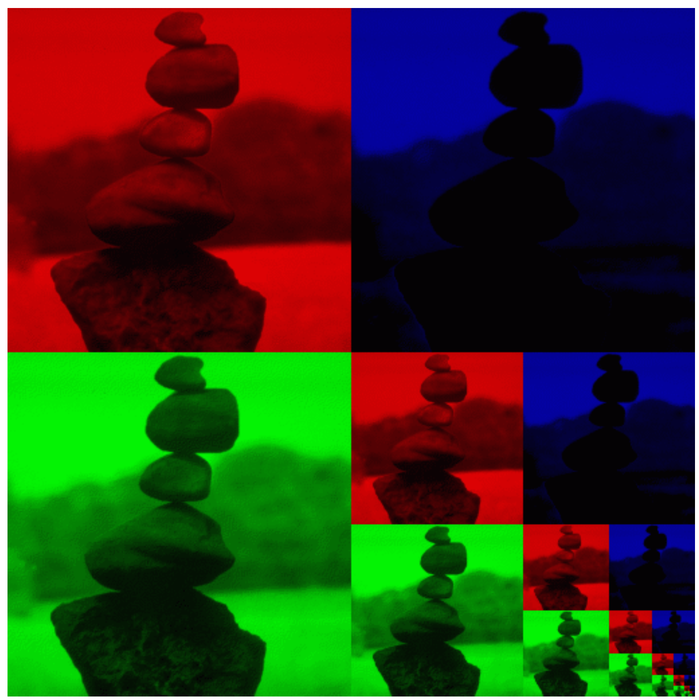


with



MIPmap storage

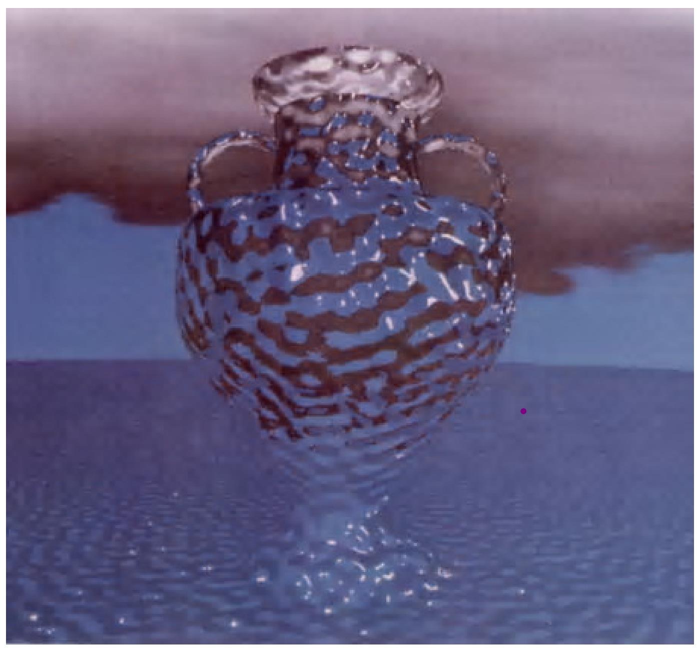
only 1/3 more space required



Texture Parameters

In addition to color can control other material/object properties

- Surface normal (bump mapping)
- Reflected color (environment mapping)

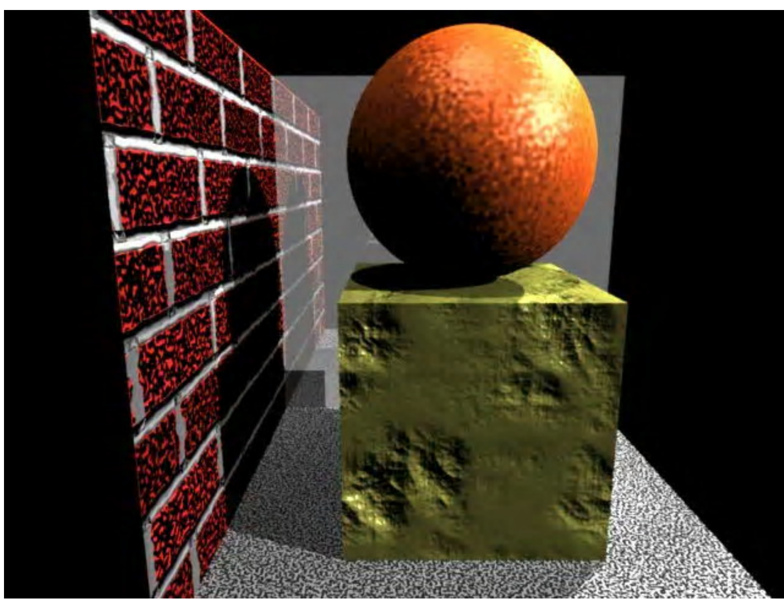
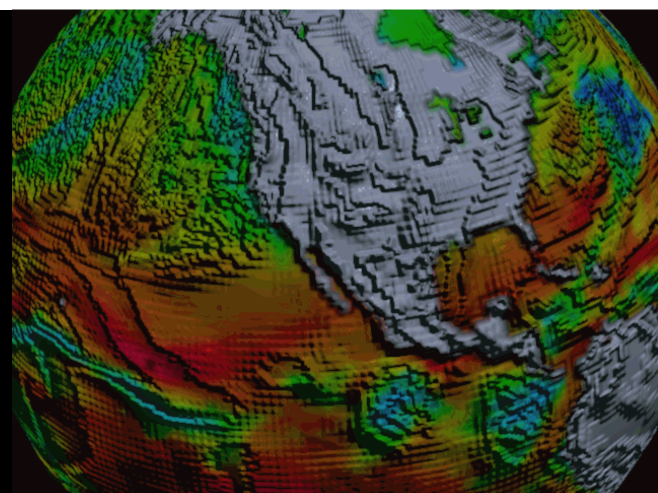
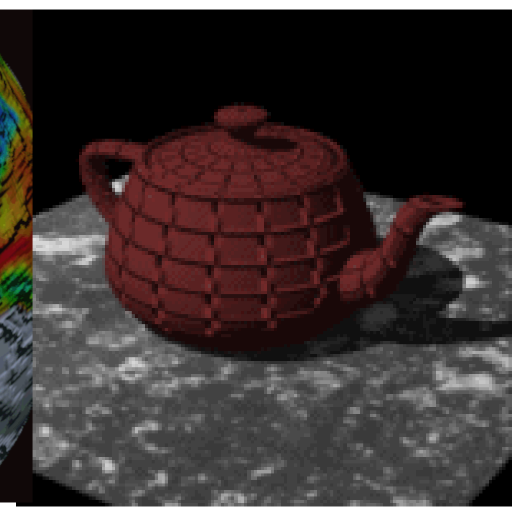


Bump Mapping: Normals As Texture

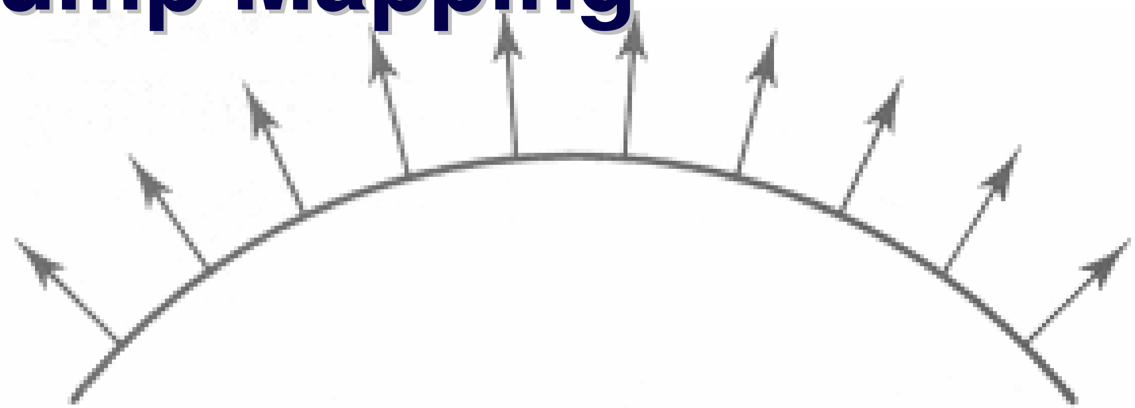
Object surface often not smooth – to recreate correctly need complex geometry model

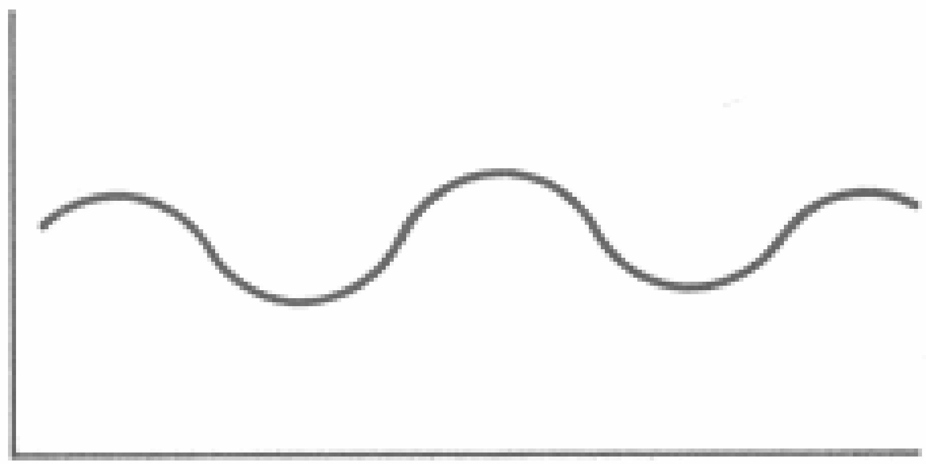
Can control shape "effect" by locally perturbing surface normal

- Random perturbation
- Directional change over region

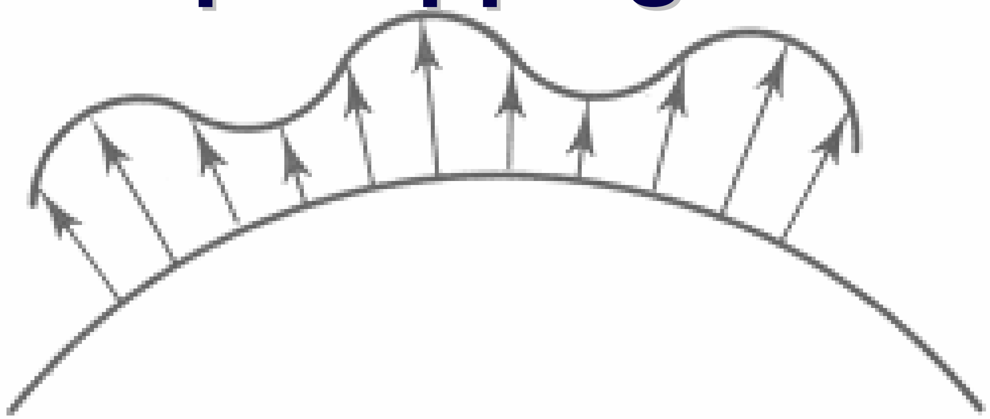




Bump Mapping



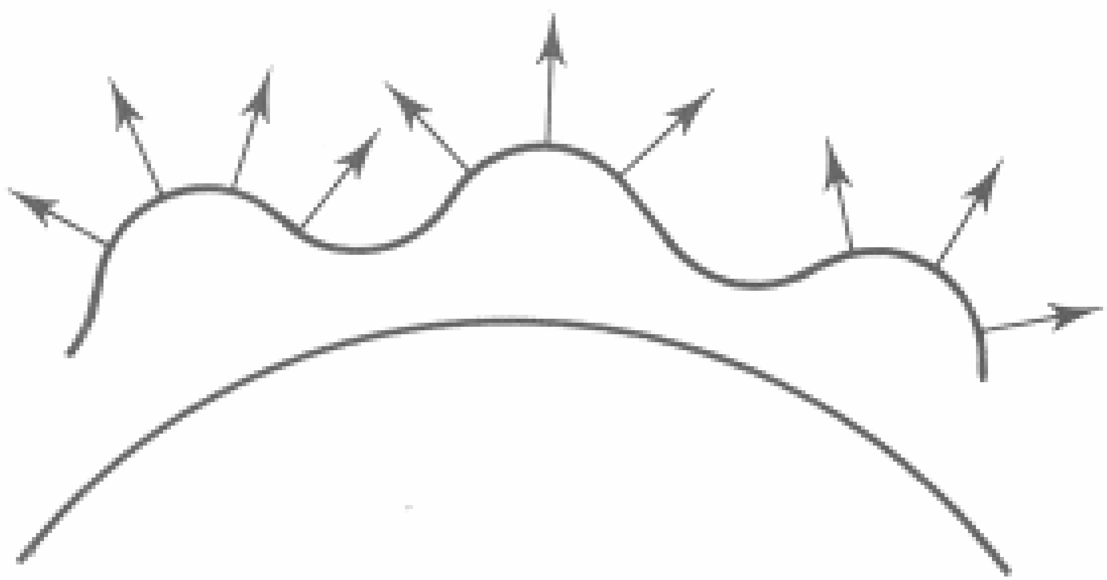


Bump Mapping



$O'(u)$

Lengthening or shortening $O(u)$ using $\boldsymbol{B}(u)$



$N'(u)$

The vectors to the 'new' surface

Displacement Mapping

Bump mapping gets silhouettes wrong

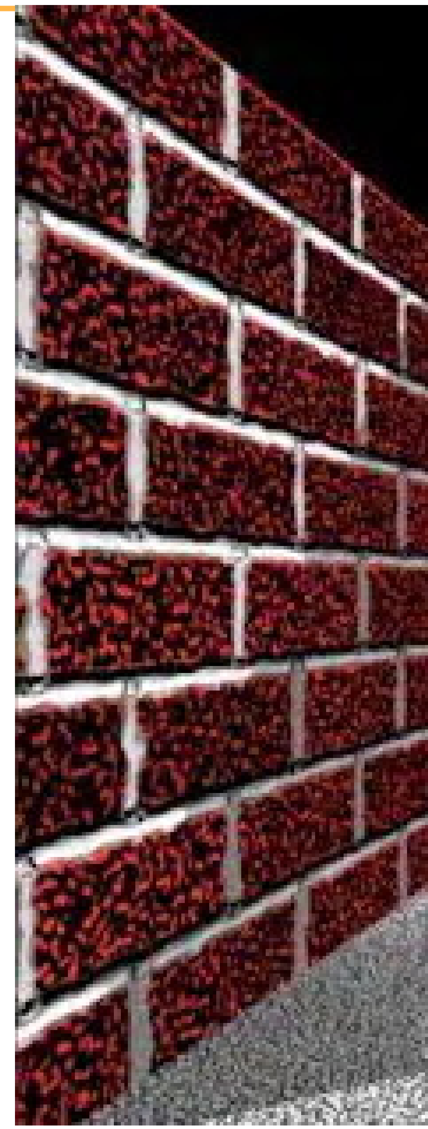
- Shadows wrong too

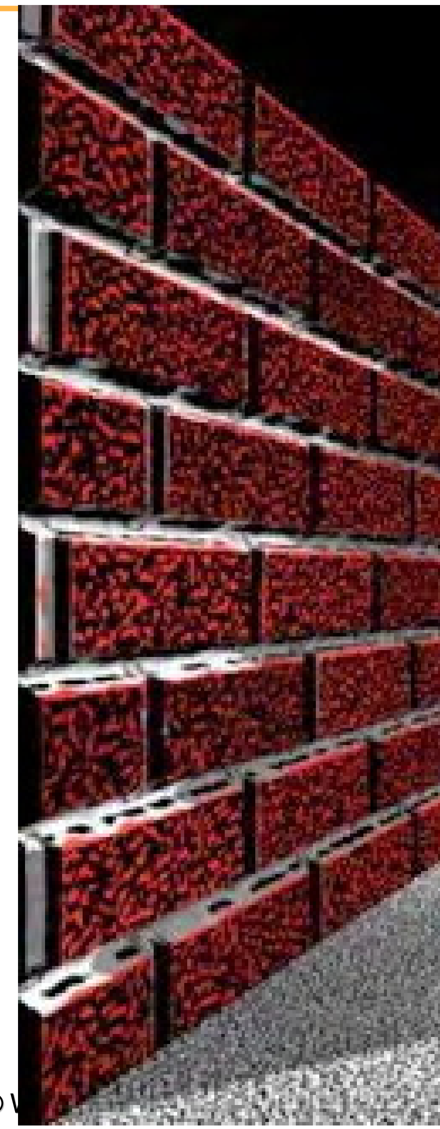
Change surface geometry instead

- Need to subdivide surface

GPU support

- Bump and displacement mapping not directly supported: require per-pixel lighting
- However: modern GPUs allow for programming both yourself



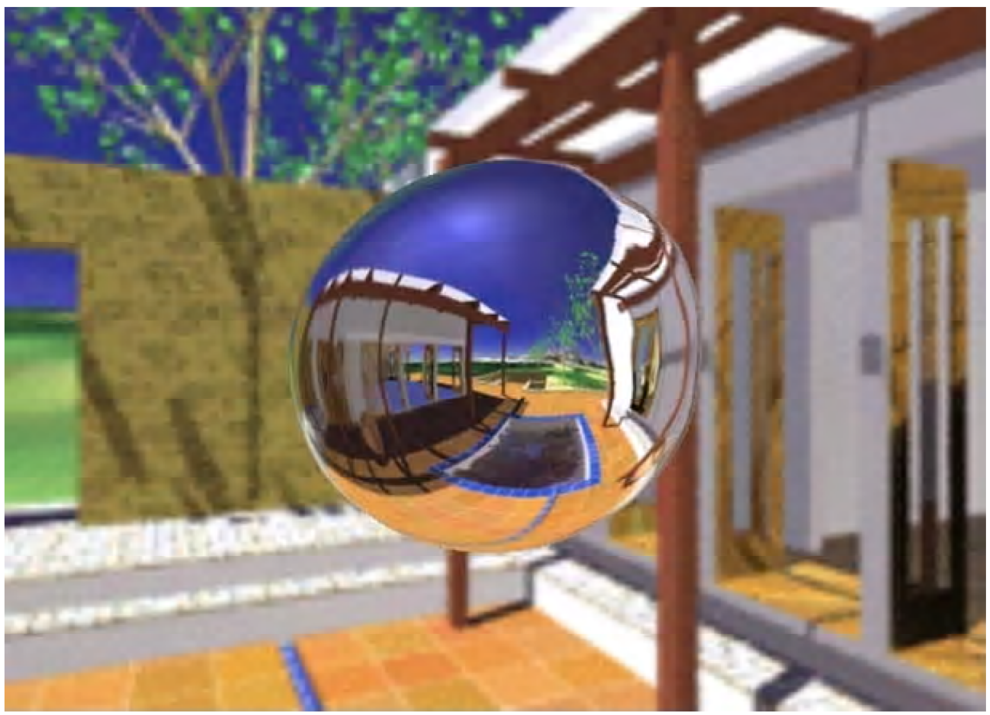


©

Environment Mapping

Cheap way to achieve reflective effect

- Generate image of surrounding
- Map to object as texture



Sphere Mapping

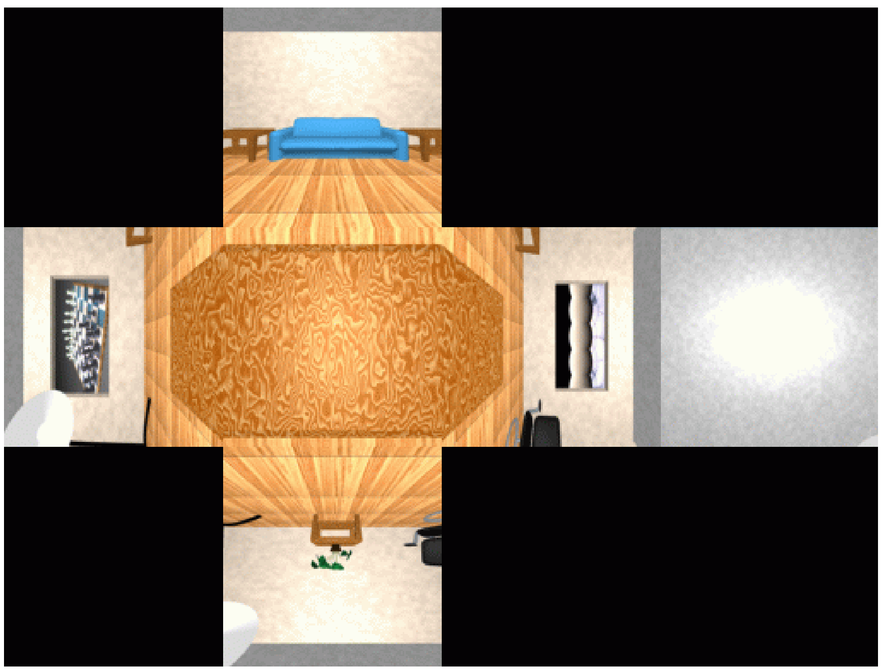
Texture is distorted fish-eye view

- Point camera at mirrored sphere
- Spherical texture mapping creates texture coordinates that correctly index into this texture map

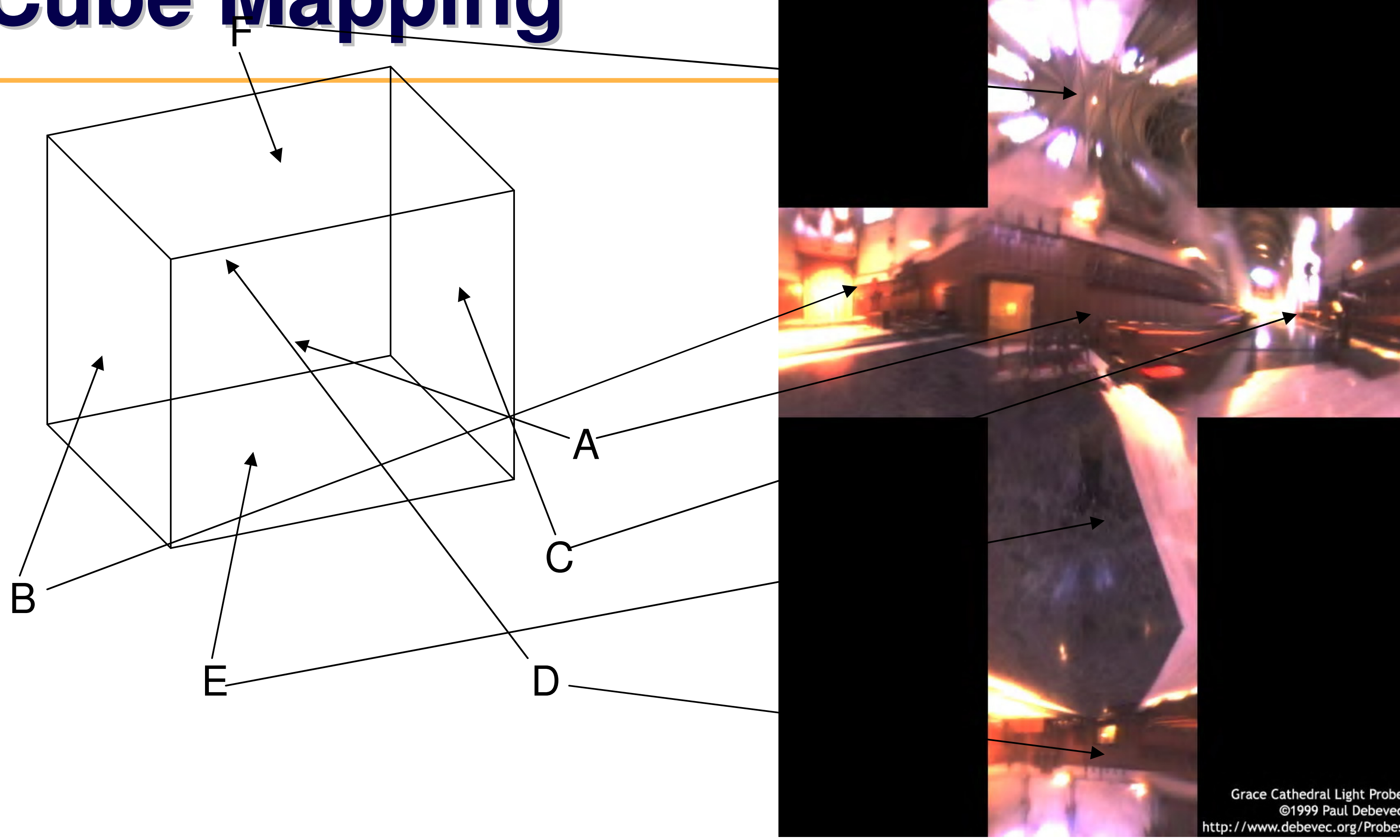
Cube Mapping

6 planar textures, sides of cube

- Point camera in 6 different directions, facing out from origin



Cube Mapping



Cube Mapping

Direction of reflection vector* r *selects the face of the cube to be indexed

- Co-ordinate with largest magnitude
  - *e.g., the vector (-0.2, 0.5, -0.84) selects the –Z face*

- Remaining two coordinates (normalized by the 3rd coordinate) selects the pixel from the face.
  - *E.g., (-0.2, 0.5) gets mapped to (0.38, 0.80).*

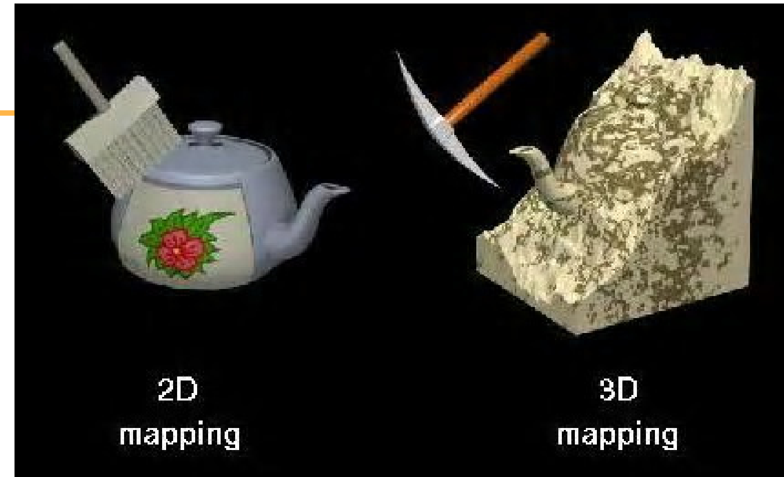
Difficulty in interpolating across faces

Volumetric Texture

Define texture pattern over 3D domain - 3D space containing the object

- Texture function can be digitized or procedural
- For each point on object compute texture from point location in space

Common for natural material/irregular textures (stone, wood,etc…)



Volumetric Bump Mapping

Marble

Bump

Procedural Textures

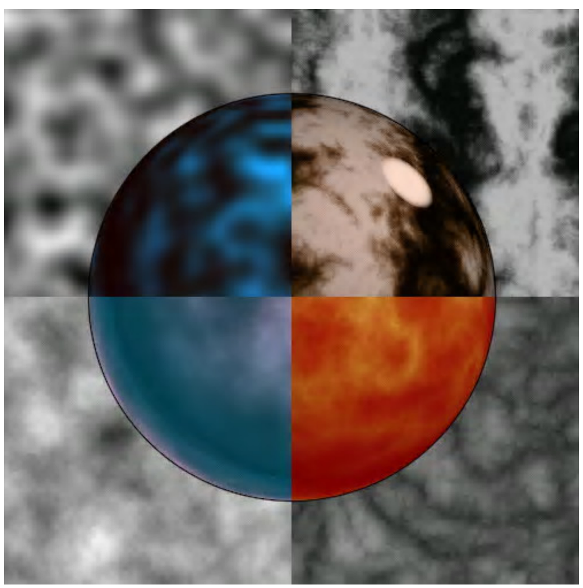
Generate "image" on the fly, instead of loading from disk

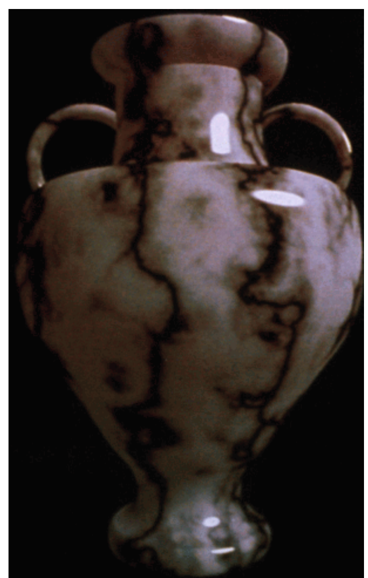
- Often saves space
- Allows arbitrary level of detail

Procedural Textures

Several good explanations

- Text book Section 10.1
- http://www.noisemachine.com/talk1
- http://freespace.virgin.net/hugo.elias/models/m_perlin.htm
- http://www.robo-murito.net/code/perlin-noise-math-faq.html





http://mrl.nyu.edu/~perlin/planet/