Color Texture Mapping
- define color (RGB) for each point on object 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 text lookup at each pixel
- use value to modify a polygon’s color
  - or other surface property
- specified by programmer or artist

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

Example Texture Map
Fractional Texture Coordinates

- what if s or t is outside the interval [0...1]?
- multiple choices
  - use fractional part of texture coordinates
  - cyclic repetition of texture to tile whole surface
  - clamp every component to range [0...1]
- re-use color values from texture image border
- texturingimage

Interpolation: Screen vs. World Space

- screen space interpolation incorrect
- problem ignored with shading, but artifacts more visible with texturing

Low-Level Details

- large range of functions for controlling layout of texture data
  - state how the data in your image is arranged
  - e.g., glTexImage2D(GL_TEXTURE_2D, ..., 1) to specify parameters for use when applying the texture
  - glTexImage2D(GL_TEXTURE_2D, ..., GL_LINEAR)
  - parameter space
  - transformed parameter space

Texture Functions

- once have value from the texture map, can:
  - directly use as surface color (GL_REPLACE)
  - throw away old color, lose lighting effects
  - modulate surface color
  - multiply old color by new value, keep lighting info
  - texturing happens after lighting, not in
  - use as screen color, modulate alpha (GL_DECAL)
  - like replace, but supports texture transparency
  - blend screen color with another (GL_BLEND)
  - new value controls which of 2 colors to use
  - interpolation, new value not used directly for coloring
- specify with
- specify with glTexImage2D(GL_TEXTURE_2D, ..., GL_REPLACE)
- texture coordinate interpolation
- how many bits per pixel
- small range of functions for controlling layout of texture data
- state how the data in your image is arranged
- e.g., glTexImage2D(GL_TEXTURE_2D, ..., GL_LINEAR)
- parameter space
- transformed parameter space
- texture coordinate interpolation
- how to do with:
  - pixels that are much larger than texels?
  - apply filtering, "averaging"
  - how to do with:
  - pixels that are much smaller than texels?
  - interpolate
- perspective foreshortening problem
- perspective correct interpolation
- perspective distorted interpolation
- barycentric coordinates of a point P in a triangle
  - s0, s1, s2:
  - homogeneous coordinates of vertices
  - interpolates across triangle (like R,G,B,X)
  - well not quite!
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

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

O(u)

Original surface

B(u)

A bump map

Bump Mapping

O'(u)

Lengthening or shortening O(u) using B(u)

N(u)

The vectors to the ‘new’ surface

Displacement Mapping

- bump mapping gets silhouettes wrong
  - shadows wrong too
- change surface geometry instead
  - only recently available with realtime graphics
  - need to subdivide surface

Embossing

- at transitions
  - rotate point’s surface normal by θ or -θ