**Back-Face Culling**

- Most objects in scene are typically "solid".
  - Specifically: orientable closed manifolds.
  - Cannot self-intersect.
  - Sphere is orientable since has two sides, 'inside' and 'outside'.
  - Möbius strip or a Klein bottle is not orientable.
  - Closed: cannot "walk" from one side to the other.
  - Sphere is closed manifold.
  - Plane is not closed.

**Back-Face Culling: Manifolds**

- Examples of manifold objects:
  - Sphere
  - Torus
  - Well-formed CAD part
- Examples of non-manifold objects:
  - A single polygon
  - A terrain or height field
  - Polyhedron w/ missing face
  - Anything with cracks or holes in boundary
  - One-polygon thick lampshade

**Back-Face Culling: NDCS**

- Most objects in scene are typically "solid".
- Specifically: orientable closed manifolds.
  - Must have two distinct sides.
  - Cannot self-intersect.
  - A sphere is orientable since has two sides, 'inside' and 'outside'.
  - Möbius strip or a Klein bottle is not orientable.
  - Closed: cannot "walk" from one side to the other.
  - Sphere is closed manifold.
  - Plane is not.

**Back-Face Culling: VCS**

- First idea: cull if \( N_z < 0 \)
- Sometimes misses polygons that should be culled.

**Back-Face Culling: Manifolds**

- Most objects in scene are typically "solid".
- Specifically: orientable closed manifolds.
  - Manifold: local neighborhood of all points isomorphic to disc.
  - Boundary partitions space into interior \& exterior.

**Back-Face Culling**

- Not rendering back-facing polygons improves performance.
  - By how much?
    - Reduces by about half the number of polygons to be considered for each pixel.
    - Optimization when appropriate.

**Clarification/Review: Depth Test Precision**

- Reminder: projective transformation maps eye-space \( z \) to generic \( z \)-range (NDC).

\[
\begin{align*}
\frac{x}{w} &= \frac{x - 2 w}{f - w} \\
\frac{y}{w} &= \frac{y - 2 w}{f - w} \\
\frac{z}{w} &= \frac{z}{w} \\
\frac{w}{w} &= \frac{1}{w}
\end{align*}
\]

- Thus \( z = \frac{1}{2} \).

- \( z_x = \frac{z(+w)}{f-w} \), \( z_y = -\frac{2}{f-w} \), \( z_w = -z_x \), \( w_w = -z_x \).

**Invisible Primitives**

- Why might a polygon be invisible?
  - Polygon outside the field of view / frustum.
  - Solved by clipping.
  - Polygon is back-facing.
  - Polygon is occluded by object(s) nearer the viewpoint.
  - Solved by hidden surface removal.

**Correction: HSV and RGB**

- HSV/HSI conversion from RGB.
  - Not expressible in matrix.

\[
I = \frac{R + G + B}{3} \quad S = 1 - \min(R, G, B)
\]

\[
H = \cos^{-1} \left( \frac{1}{2} \left[ (R - G) + (R - B) \right] \right)
\]
Texture Functions
- Once have value from the texture map, can:
  - Directly use as surface color: GL_REPLACE
  - Directly use as surface color: GL_REPLACE
  - Use interpolated (s,t) for texel lookup at each pixel
  - Use value to modify a polygon's color
  - Or other surface property
  - Specify by programmer or artist

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 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
- Specify texture coordinates for the polygon

Texture Coordinate Transformation
- Motivation
  - Change scale, orientation of texture on an object
  - Approach
  - Texture matrix stack
  - Transforms specified (or generated) to texture
  - GL_TEXTURE_2D, ...)
  - To specify the texture data (the image itself)
  - Enable texture: glEnable(GL_TEXTURE_2D)
  - State how the texture will be used
  - Texture object (and fill it with texture data)
  - GL_TEXTURE_2D
  - GL_TEXTURE_2D
- Following texture commands refer to the bound texture
- glTexParameter( GL_TEXTURE_2D, ... )
- Specify parameters for use when applying the texture
- glTexImage2D(GL_TEXTURE_2D, ...)
  - To specify the texture data (the image itself)
  - No priority is assigned
  - Switch between preloaded textures
Low-Level Details
- large range of functions for controlling layout of texture data
- state how the data in your image is arranged
  - e.g.: glPixelStore(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 be square and size a power of 2
- common sizes are 32x32, 64x64, 256x256
- smaller uses less memory, and there is a finite amount of texture memory on graphics cards
- ok to use texture template sample code for project 4

Texture Mapping
- texture coordinates
  - specified at vertices
    - glTexImage2D(s, t);
    - glVertexf(x, y, z);
  - interpolated across triangle (like R,G,B,Z)
  - ...well not quite!
- 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
        - (x1, y1, z1, w1)
        - (x2, y2, z2, w2)
      - (x0, y0, z0, w0)
      - (x2, y2, z2, w2)
      - α = x1/w1 + β x2/w2 + γ x3/w3
      - β = x1/w1 + β x2/w2 + γ x3/w3
      - γ = x1/w1 + β x2/w2 + γ x3/w3

Reconstruction
- how to deal with:
  - pixels that are much larger than texels?
    - apply filtering, "averaging"
  - pixels that are much smaller than texels?
    - interpolate
- pixels that are much larger than texels
- pixels that are much smaller than texels
- how many bits per pixel
- OpenGL not to skip bytes at the end of a row
- shadows wrong too
- change surface geometry instead
- only recently available with realtime graphics
- need to subdivide surface

MIPmapping
- "image pyramid" to precompute averaged versions of the texture
- only 1/3 more space required
- "image pyramid" image

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
- glTexImage2D
- automatically constructs a family of textures from original texture size down to 1x1
- use "image pyramid": will only store whole pyramid in single block of memory

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
- lengthening or shortening
  - rotate point’s surface normal by _ or _
  - at transitions

Embossing
- bump mapping gets silhouettes wrong
- shadows wrong too
- change surface geometry instead
- only recently available with realtime graphics
- need to subdivide 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
Environment Mapping
- cheap way to achieve reflective effect
- generate image of surrounding
- map to object as texture

Sphere Mapping
- used to model object that reflects surrounding textures to the eye
- movie example: cyborg in Terminator 2
- different approaches
  - sphere, cube most popular
    - OpenGL support
      - GL_SPHERE_MAP, GL_CUBE_MAP
  - others possible too

Cube Mapping
- 6 planar textures, sides of cube
- point camera in 6 different directions, facing out from origin

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

Review: Texture Objects and Binding
- texture objects
- texture management: switch with bind, not reloading
- can prioritize textures to keep in memory
- Q: what happens to textures kicked out of memory?
  - A: resident memory (on graphics card) vs. nonresident (on CPU)

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 Texture Principles
- 3D function \( p \)
  \[ \forall p = p(x,y,z) \]
- texture space – 3D space that holds the texture (discrete or continuous)
- rendering: for each rendered point \( P(x,y,z) \) compute \( p(x,y,z) \)
- volumetric texture mapping function/space transformed with objects