Chapter 12

Texture Mapping

- Real life objects non-uniform in terms of color & normal
- 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 point on object surface
- Two approaches
  - Surface texture map
  - Volumetric texture

Surface texture

- Define texture pattern over \((u,v)\) domain (Image)
  - Image - 2D array of “texels”
- Assign \((u,v)\) coordinates to each point on object surface
- How: depends on surface type
- For polygons (triangle)
  - Inside - use barycentric coordinates
  - For vertices need mapping function (artist/programmer)
Texture Mapping

- Texture Mapping Example

- Texture Coordinates
  - Every triangle has object coordinates and texture coordinates
    - Object coordinates describe where triangle vertices are on the screen
    - Texture coordinates describe texel coordinates of each vertex
    - Texture coordinates are interpolated across triangle (like R,G,B,Z)
      - (well, not quite...)
    - glTexCoord2f(TYPE coords)
    - Other versions for different texture dimensions

- Fractional Texture Coordinates

- 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 of texture to tile whole surface
          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);
gVertex3d (x, y, z);

(1,0) (0,0) (0,1)

(1,1)

glTexCoord2d(4, 4);
gVertex3d (x, y, z);

(4,4) (4,0)

(0,4) (0,0)

OpenGL Details

- How to mix texture & color (replace, blend, etc...)
- Transformations: Change scale, orientation of texture on an object
- Storage: data structure + read format
  - Rule: size always power of 2
- Binding: which image to use

Texture Mapping

- Texture coordinate interpolation
  - Perspective foreshortening problem
  - Also problematic for color interpolation, etc.

Texture Coordinate Interpolation

- Perspective Correct Interpolation
  - $\alpha$, $\beta$, $\gamma$: Barycentric coordinates (2D) of point P
  - $s_0$, $s_1$, $s_2$: texture coordinates of vertices
  - $w_0$, $w_1$, $w_2$: homogenous coordinate of vertices

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

Perspective - Reminder

$$ T \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} $$

- Preserves order
- BUT distorts distances

Interpolation: Screen vs. World Space

Screen space (perspective) interpolation incorrect
- Problem ignored with shading, but artifacts more visible with texturing
Texture: Sampling & Reconstruction

How to deal with:
- pixels that are much larger than texels?
  (apply filtering, “averaging”)
- pixels that are much smaller than texels?
  (interpolate)

Reconstruction

Magnification: Interpolating Textures

- Nearest neighbor
- Bilinear
- Hermite (cubic)

Related: Upsampling pixel images

MIP-mapping

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

MIP-mapping

without

with
MIPmap storage
- Only 1/3 more space required

Texture Parameters
- In addition to color can control other material/object properties
  - Reflectance (either diffuse or specular)
  - Surface normal (bump mapping)
  - Transparency
  - 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
  - Directional

Bump Mapping
- Original surface $O(u)$
- Bump map $B(u)$
- New surface $N'(u)$

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
  - Modern GPUs allow for programming both yourself

Copyright: Alla Sheffer, UBC, 2013
Environment Mapping
- cheap way to achieve reflective effect
- generate image of surrounding
- map to object as texture

Environment 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

Cube 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

Sphere Mapping
- 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