TEXTURE MAPPING

- real life objects have nonuniform colors, normals
- to generate realistic objects, reproduce coloring & normal variations = texture
- can often replace complex geometric details

BUMP MAPS

2D texture maps that are used to model the appearance of surface bumps, by adding small perturbations to the surface normals. The rendered geometry does not actually have bumps, i.e., it is smooth!!

threejs.org: materials/bumpmap
VOLUMETRIC TEXTURES
• model r,g,b for every point in a volume
• often computed using procedural function

BASIC TEXTURE MAP
interpolate (u,v) from vertices using barycentric coordinates
3D model: u,v texture coordinates are assigned to vertices by artist or program.
rendered image

ENVIRONMENT MAP
2 of 6 images for a cube map; as a viewer, you are inside this cube!

There is an invisible corner seam in this image!

TEXTURE MAPPING EXAMPLE
2D texture map: Image
Pixels here are called "texels"
FRACTIONAL TEXTURE COORDINATES

(0.0) (0.0) (0.0) (0.25,0)
(0.0) (0.5) (0.0) (0.25,0.5)
(0.0) (0.1) (0.1) (1.0)
(0.5) (0.25,0.5)

THREE.JS

• pass texture as a uniform:

```javascript
var uniforms = {
  texture: { type: "t", value: THREE.ImageUtils.loadTexture( "texture.jpg" ) }};
var material = new THREE.ShaderMaterial( { uniforms, ... });
```

• uv will be passed on to the vertex shader (no need to write this):

```javascript
attribute vec2 uv;
```

• use it, e.g., in Fragment Shader:

```javascript
uniform sampler2D texture1;
varying vec2 texCoord;
vec4 texColor = texture2D(texture1, texCoord);
```

HOW TO USE COLOR TEXTURES

• Replace
  • Set fragment color to texture color

```javascript
{gl_FragColor = texColor;}
```

• Modulate
  • Use texture color as reflection color in illumination equation

```javascript
kd = texColor; ka = texColor;
gl_FragColor = ka*ia + kd*id*dotProduct + ...;
```

TEXTURE LOOKUP: TILING AND CLAMPING

• What if s or t is outside [0…1]?
  • Multiple choices
    • Use fractional part of texture coordinates
    • Cyclic repetition (repeat)
    • Clamp every component to range [0…1]
    • Re-use color values from texture image border
IN THREE.JS

```javascript
var texture = THREE.ImageUtils.loadTexture( "textures/water.jpg" );
texture.wrapS = THREE.RepeatWrapping;
texture.wrapT = THREE.ClampToEdgeWrapping;
texture.repeat.set( 4, 4 );
```

RECONSTRUCTION

• how to deal with:
  • pixels that are much larger than texels?
    • minification
  • pixels that are much smaller than texels?
    • magnification

(Image courtesy of Kiriakos Kutulakos, U Rochester)
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
  • texture.generateMipmaps = true
    • automatically constructs a family of textures from original texture size down to 1x1
  • texture.mipmaps[...]

MIPMAP STORAGE

• only 1/3 more space required

HOW TO INTERPOLATE S,T?
TEXTURE COORDINATE INTERPOLATION

\[ u = \frac{\alpha \cdot u_1 + \beta \cdot u_2}{\alpha h_1 + \beta h_2} + \frac{\gamma \cdot u_3}{\gamma h_3} \]

Perspective correct interpolation (see Scan Conversion notes)

\[ u = \alpha \cdot u_1 + \beta \cdot u_2 + \gamma \cdot u_3 \]

Linear interpolation i.e., using barycentric coordinates

INTERPOLATION: SCREEN VS. WORLD SPACE

- Screen space interpolation incorrect under perspective
- Problem ignored with shading, but artifacts more visible with texturing

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

Virtual surface created with the bump map

Normals corresponding to this virtual surface
Normal/Bump mapping

- BUMP MAPPING: LIMITATION
  - 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

- generate image of surrounding or reflection
  - sphere map or cube map

---

CUBE MAP

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

• Cube map: direction of vector 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 select the pixel from the face.

SPHERE MAP

- texture is distorted fish-eye view
- point camera at mirrored sphere
- spherical texture mapping creates texture coordinates that correctly index into this texture map

• note: viewpoint is always at the center!

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
    • e.g., ShaderToy
  • computing is cheap, memory access not as much
PROCEDURAL TEXTURE EFFECTS: BOMBING

- randomly drop bombs of various shapes, sizes and orientation into texture space (store data in table)
  - for point P search table and determine if inside shape
    - if so, color by shape's color
    - otherwise, color by object's color

PROCEDURAL TEXTURES: PERLIN NOISE

- several good explanations
  - http://www.noisemachine.com/talk1
  - http://freespace.virgin.net/hugo.elias/models/m_perlin.htm

THE RENDERING PIPELINE

1. Vertices and attributes
   - Vertex Shader
     - Modelview transform
     - Per-vertex attributes
   - Rasterization
     - Scan conversion
     - Interpolation
   - Per-Sample Operations
     - Depth test
     - Blending
2. Vertex Post-Processing
   - Viewport transform
   - Clipping
3. Fragment Shader
   - Texturing/...
   - Lighting/shading
4. Framebuffer