CPSC 314
TEXTURE MAPPING

COLOR TEXTURE MAPPING
• define color (RGB) for each point on object surface
• other:
  • volumetric texture
  • procedural texture

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

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)

THREE.JS
• pass texture as a uniform:
  • use it, e.g., in Fragment Shader:
  • define color (RGB) for each point on object surface
  • other:
    • volumetric texture
    • procedural texture

HOW TO USE COLOR TEXTURES
• Replace
  • Set fragment color to texture color
  • gl_FragColor = texColor;
• Modulate
  • Use texture color as reflection color in illumination equation
  • kd = texColor; ka = texColor;
  • gI_FragColor = kA*1a + kA*Id*dotProduct + ...

TEXTURE MAPPING EXAMPLE
+ =
COLOR TEXTURE MAPPING
• define color (RGB) for each point on object surface
• other:
  • volumetric texture
  • procedural texture

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 – Questions?
• 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)

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

TILED TEXTURE MAP
• Pass texture as a uniform:
  • uniforms = {
  • texture1: { type: "t", value: THREE.ImageUtils.loadTexture("textures/water.jpg") };
• var uniforms = {
texture1: { type: "t", value: ...

TEXTURE MAPPING EXAMPLE
+ =
**RECONSTRUCTION**

use "image pyramid" to precompute averaged versions of the texture.

(Images courtesy of Kiriakos Kutulakos, U Rochester)

**MIPMAPPING**

- Without MIP-mapping, use "image pyramid" to precompute averaged versions of the texture.
- Store whole pyramid in single block of memory.

**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.

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

- `texture.generateMipmaps = true`.

**TEXTURE MAPPING**

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

**OTHER USES FOR TEXTURES**

- Usually provides colour, but...
- Can also use to control other material/object properties
  - Surface normal (bump mapping)
  - Reflected color (environment mapping)

**BUMP MAPPING**

- 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: LIMITATION**

- Why don’t we modify geometry instead of modifying normals?

**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
**THE RENDERING PIPELINE**

- **Vertex Shader**
  - Vertices and attributes
- **Rasterization**
  - Scan conversion
- **Interpolation**
- **Fragment Shader**
  - Texturing:
    - Normal
  - Lighting/shading
- **Per-Sample Operations**
  - Depth test
  - Blending
- **Framebuffer**

**ENVIRONMENT MAPPING**

- used to model object that reflects surrounding textures to the eye
- example: cyborg in Terminator 2
- different approaches
  - sphere, cube most popular
  - others possible too

**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 select the pixel from the face.
  - difficulty in interpolating across faces

**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

**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

**ENVIRONMENT MAPS (EM)**

- in theory, every object should have a separate EM
- in theory, every time something moves, you should re-compute EM
- "you'll be surprised at what you can get away with"

**PERLIN NOISE: PROCEDURAL TEXTURES**

- several good explanations
  - http://www.worldsperlin.com/faq1.htm
  - http://reforg.images.net/legarde/modules/stuff/stuff.html
  - http://www.noisemachine.com/talk1

**PERLIN NOISE: TURBULENCE**

- multiple feature sizes
  - add scaled copies of noise

**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 is expensive!