WHY IS TEXTURE IMPORTANT?
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

• hide geometric simplicity
  • images convey illusion of geometry
  • map a brick wall texture on a flat polygon
  • create bumpy effect on surface

• usually:
  associate 2D information with a surface in 3D
  • point on surface $\leftrightarrow$ point in texture
  • “paint” image onto polygon
COLOR TEXTURE MAPPING

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

$(u_0, v_0)$

$(u_1, v_1)$

$(u_2, v_2)$

$(u, v)$ parameterization
sometimes called $(s, t)$
TEXTURE MAPPING – Questions?
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 EXAMPLE
TEXTURE MAPPING EXAMPLE
Pause .... --> Math Example
THREE.JS

• pass texture as a uniform:

```javascript
var uniforms = {
  texture1: { 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

  \[ \text{gl\_FragColor} = \text{texColor}; \]

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

  \[ \text{kd} = \text{texColor}; \text{ ka} = \text{texColor}; \]
  \[ \text{gl\_FragColor} = \text{ka}\times\text{ia} + \text{kd}\times\text{id}\times\text{dotProduct} + \ldots; \]
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);
```
TILED TEXTURE MAP

(1,0) + (1,1) = (0,0) + (0,1) = Mapped Texture

(4,0) + (4,4) = (0,0) + (0,4) = Mapped Texture
RECONSTRUCTION

(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
MIP MAPS

- 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[...]`

without

with
MIPMAP STORAGE

• only 1/3 more space required
HOW TO INTERPOLATE S,T?
Texture coordinate interpolation

- Perspective foreshortening problem
- Also problematic for color interpolation, etc.
OTHER USES FOR TEXTURES
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: 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

Original surface

A bump map
BUMP MAPPING

\[ O'(u) \]
Lengthening or shortening \( O(u) \) using \( B(u) \)

\[ N'(u) \]
The vectors to the 'new' surface
EMBOSSING

• at transitions
  • rotate point's surface normal by $\theta$ or $-\theta$
BUMP MAPPING: LIMITATION
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
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
  • others possible too
**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
CUBE MAPPING

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

how to calculate?

• 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
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”
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!
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
PERLIN NOISE: PROCEDURAL TEXTURES

• several good explanations
  • http://www.noisemachine.com/talk1
  • http://freespace.virgin.net/hugo.elias/models/m_perlin.htm
  • http://www.robo-murito.net/code/perlin-noise-math-faq.html

http://mrl.nyu.edu/~perlin/planet/
PERLIN NOISE: TURBULENCE

• multiple feature sizes
  • add scaled copies of noise

Sum of Noise Functions = (Perlin Noise)
PERLIN NOISE: TURBULENCE

- multiple feature sizes
  - add scaled copies of noise
THE RENDERING PIPELINE

Vertices and attributes

- **Vertex Shader**
  - Modelview transform
  - Per-vertex attributes

- **Rasterization**
  - Scan conversion
  - Interpolation

- **Per-Sample Operations**
  - Depth test
  - Blending

- **Fragment Shader**
  - Texturing/...
  - Lighting/shading

- **Vertex Post-Processing**
  - Viewport transform
  - Clipping

- **Framebuffer**