Textures

Reading for Texture Mapping

- FCG Chap 11 Texture Mapping (except 11.7, except 11.8, 2nd ed)
- RB Chap Texture Mapping

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 Example

Color Texture Mapping

- define color (RGB) for each point on object surface
- two approaches
  - surface texture map
  - volumetric texture

Texture Coordinates

- texture image: 2D array of color values (texels)
- assigning texture coordinates (s,t) at vertex with object coordinates (x,y,z,w)
- use interpolated (s,t) for texture lookup at each pixel
- use value to modify a polygon’s color
- or other surface property
- specified by programmer or artist

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
  - clamp every component to range [0…1]
- re-use color values from texture image border
- new value controls which of 2 colors to use

Texture Functions

- once have value from the texture map, can:
  - directly use as surface color (GL_REPLACE)
  - throw away old color, lose lighting effects
  - module surface color (GL_MODULATE
  - multiply old color by new value, keep lighting info
  - texturing happens after lighting, not relit
  - use as surface color, modulate alpha (GL_DECAL
  - like replace, but supports texture transparency
  - blend surface color with another (GL_BLEND

- new value controls which of 2 colors to use
  - inddirect, new value not used directly for coloring
- specify with glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_REPLACE)

Texture Coordinate Transformation

- motivation
  - change scale, orientation of texture on an object
- approach
  - texture matrix stack
  - transforms specified (or generated) tex coords
    - glMatrixMode(GL_TEXTURE);
    - glLoadIdentity();
    - ... more flexible than changing (s,t) coordinates
- [demo]
Texture Pipeline

- Texel color
  - (0.9, 0.8, 0.7)
- Object position
  - (-2.3, 7.1, 17.7)
- Transform parameter space
  - (0.52, 0.49)
- Final color
  - (0.45, 0.4, 0.35)
- Object color
  - (0.5, 0.5, 0.5)

Texture Objects and Binding

- texture object
  - an OpenGL data type that keeps textures resident in memory and provides identifiers to easily access them
- texture coordinate interpolation
  - perspective foreshortening problem
- 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:
  - glTexImage2D(GL_TEXTURE_2D, ...)
- following texture commands refer to the bound texture
- texture coordinate interpolation
  - barycentric coordinates of a point P in a triangle
- texture coordinate interpolation
  - perspective correct interpolation
- α, β, γ : 
  - barycentric coordinates of a point P in a triangle
- 
  \[
  (s_0, t_0) = (s_2, t_2) = (x_0, y_0, z_0, w_0) \\
  (x_1, y_1, z_1, w_1) \\
  (x_2, y_2, z_2, w_2) \\
  (\alpha, \beta, \gamma) \\
  (s, t) \\
  
  (s_1, t_1) \\
  P(x, y, z) \\
  V_0(x', y') \\
  V_1(x', y') \\
  P_0(x, y, z) \\
  P_1(x, y, z) \\
  V(x', y')
  \]

Texture Coordinate Interpolation

- screen space interpolation incorrect
- problem ignored with shading, but artifacts more visible with texturing

Interpolation: Screen vs. World Space

- screen space interpolation incorrect
- perspective correct interpolation

MIPmaps

- multum in parvo – many things in a small place
- prespecify a series of prefiltered texture maps... without
- automatically constructs a family of textures from original texture size down to 1x1

Low-Level Details

- large range of functions for controlling layout of texture data
- state how the data in your image is arranged
- textures must be square and a power of 2
- 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 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

Texture Mapping

- texture coordinates
  - specified at vertices
    - glTexCoord2f(s, t);
    - glVertex3f(x, y, z);
  - interpolated across triangle (like R, G, B, Z)
  - …well not quite!

Interpolation: Screen vs. World Space

- screen space interpolation incorrect
- perspective correct interpolation

MIPmapping

- use “image pyramid” to precompute averaged versions of the texture

MIPmap storage

- only 1/3 more space required

Basic OpenGL Texturing

- create a texture object and fill it with texture data:
  - glTexImage2D(GL_TEXTURE_2D, ...)
- following texture commands refer to the bound texture
- texture coordinate interpolation
  - barycentric coordinates of a point P in a triangle
- texture coordinate interpolation
  - perspective correct interpolation
- α, β, γ : 
  - barycentric coordinates of a point P in a triangle
- 
  \[
  (s_0, t_0) = (s_2, t_2) = (x_0, y_0, z_0, w_0) \\
  (x_1, y_1, z_1, w_1) \\
  (x_2, y_2, z_2, w_2) \\
  (\alpha, \beta, \gamma) \\
  (s, t) \\
  
  (s_1, t_1) \\
  P(x, y, z) \\
  V_0(x', y') \\
  V_1(x', y') \\
  P_0(x, y, z) \\
  P_1(x, y, z) \\
  V(x', y')
  \]

Interpolation: Screen vs. World Space

- screen space interpolation incorrect
- perspective correct interpolation

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(GL_TEXTURE_2D, ...)

Low-Level Details

- large range of functions for controlling layout of texture data
- state how the data in your image is arranged
- textures must be square and a power of 2
- 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 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

Texture Mapping

- texture coordinates
  - specified at vertices
    - glTexCoord2f(s, t);
    - glVertex3f(x, y, z);
  - interpolated across triangle (like R, G, B, Z)
  - …well not quite!

Interpolation: Screen vs. World Space

- screen space interpolation incorrect
- perspective correct interpolation

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(GL_TEXTURE_2D, ...)

Low-Level Details

- large range of functions for controlling layout of texture data
- state how the data in your image is arranged
- textures must be square and a power of 2
- 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 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

- at transitions
  - rotate point's surface normal by $\theta$ or $-\theta$

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 (normalized by the 3rd coordinate) selects the pixel from the face.
  - e.g., (-0.2, 0.5) gets mapped to (0.38, 0.80).
- difficulty in interpolating across faces

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

Displacement Mapping

- cheap way to achieve reflective effect
- generate image of surrounding
- map to object as texture

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

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 $p(x,y,z)$
- 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

Procedural Textures

- generate “image” on the fly, instead of loading from disk
- often saves space
- allows arbitrary level of detail

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
    - otherwise, color by objects color

Procedural Textures

- simple marble
  ```c
  function boring_marble(point)
  x = point.x;
  return marble_color(sin(x));
  // marble_color maps scalars to colors
  ```
Perlin Noise: Procedural Textures
- several good explanations
  - FCG Section 10.1

Perlin Noise: Coherency
- smooth not abrupt changes
- coherent
- white noise

Perlin Noise: Turbulence
- multiple feature sizes
  - add scaled copies of noise

Generating Coherent Noise
- just three main ideas
  - nice interpolation
  - use vector offsets to make grid irregular
  - optimization
  - sneaky use of 1D arrays instead of 2D/3D one

Interpolating Textures
- nearest neighbor
  - bilinear
  - hermite

Optimization
- save memory and time
  - conceptually: 2D or 3D grid
  - populate with random number generator
  - actually:
    - precompute two 1D arrays of size n (typical size 256)
    - random unit vectors
    - permutation of integers 0 to n-1
    - lookup
      - g(i, j, k) = G[(i + P(j + P(k)) mod n) mod n]

Perlin Marble
- use turbulence, which in turn uses noise:
  - function marble(point):
    - x = point.x + turbulence(point);
    - return marble_color(sin(x))

Perlin Noise: Turbulence
- multiple feature sizes
- add scaled copies of noise

Vector Offsets From Grid
- weighted average of gradients
  - random unit vectors

Procedural Modeling
- textures, geometry
  - nonprocedural: explicitly stored in memory
  - procedural approach
  - compute something on the fly
  - often less memory cost
  - visual richness
  - fractals, particle systems, noise

Fractal Landscapes
- fractals: not just for "showing math"
  - triangle subdivision
  - vertex displacement
  - recursive until termination condition

1D: Midpoint Displacement
- divide in half
  - randomly displace
  - scale variance by half

Fractal Dimension
- D = \log(N)/\log(r)
- N = measure, r = subdivision scale
  - Hausdorff dimension: noninteger
  - coastline of Britain: D = \log(4)/\log(3) = 1.26
  - Koch snowflake: F → FLFRRFLF
  - F: forward, R: right, L: left
  - Mariano’s Bush: F=FF-[F+F+F+F]+*[F-F-F]
  - angle 16

Language-Based Generation
- L-Systems: after Lindenmayer
  - Koch snowflake: F → FLFRRFLF
  - F: forward, R: right, L: left
  - Mariano’s Bush: F=FF-[F+F+F+F]+*[F-F-F]
  - angle 16

Self-Similarity
- infinite nesting of structure on all scales
2D: Diamond-Square
- fractal terrain with diamond-square approach
- generate a new value at midpoint
- average corner values + random displacement
- scale variance by half each time

Particle Systems
- loosely defined
- modeling, or rendering, or animation
- key criteria
- collection of particles
- random element controls attributes
  - position, velocity (speed and direction), color, lifetime, age, shape, size, transparency
  - predefined stochastic limits: bounds, variance, type of distribution

Particle System Examples
- objects changing fluidly over time
  - fire, steam, smoke, water
- objects fluid in form
  - grass, hair, dust
- physical processes
  - waterfalls, fireworks, explosions
- group dynamics: behavioral
  - birds/bats flock, fish school, human crowd, dinosaur/elephant stampede

Particle Systems Demos
- general particle systems
  - http://www.wondertouch.com
- boids: bird-like objects
  - http://www.red3d.com/cwr/boids/

Particle Life Cycle
- generation
  - randomly within "fuzzy" location
  - initial attribute values: random or fixed
- dynamics
  - attributes of each particle may vary over time
    - color darker as particle cools off after explosion
    - can also depend on other attributes
    - position: previous particle position + velocity + time
  - death
    - age and lifetime for each particle (in frames)
    - or if out of bounds, too dark to see, etc

Particle System Rendering
- expensive to render thousands of particles
- simplify: avoid hidden surface calculations
  - each particle has small graphical primitive (blob)
  - pixel color: sum of all particles mapping to it
- some effects easy
  - temporal anti-aliasing (motion blur)
  - normally expensive: supersampling over time
    - position, velocity known for each particle
    - just render as streak

Procedural Approaches Summary
- Perlin noise
- fractals
- L-systems
- particle systems
- not at all a complete list!
  - big subject: entire classes on this alone