Announcements

Written Assignment 2 is due 3/8
Advanced Texturing

Bump Mapping
Displacement Mapping
Environment Mapping
Procedural Textures

Shirley Chapter 11

COMPUTER GRAPHICS
15-462
Uses for Texture Mapping (Heckbert 1986)

Use texture to affect a variety of parameters

- surface color
  (Catmull 1974)
- surface reflectance
- normal vector
- geometry
- transparency
- light source radiance
  - color (radiance) of each point on surface
  - reflectance coefficients $k_d$, $k_s$, or $n_{shiny}$
  - bump mapping (Blinn 1978)
  - displacement mapping
  - transparency mapping (clouds) (Gardener 1985)
  - environment mapping (Blinn 1978)
Fine Surface Detail

http://www.siggraph.org/education/materials/HyperGraph/mapping/bumpmap.htm

How can we model this level of detail?
Bump Mapping

- Basic texture mapping paints on to a smooth surface
- How do you make a surface look *rough*?
  - Option 1: model the surface with many small polygons
  - Option 2: perturb the normal vectors before the shading calculation
Why Does this Work?

- Which spots bulge out, and which are indented?

- The human visual system is hard-coded to expect light to come from above (the sun)
Implementing Bump Mapping

- At each point, displace the normal by some amount to change lighting
- Displacements stored in bump-map (texture image)
Bump Mapping is a Hack

• What anomalies do you see in the image below?
Displacement Mapping

- Use texture map to displace each point on the surface
  - Texture value gives amount to move in direction normal to surface

- How is this better/worse than bump mapping?
Environment Mapping

• Very shiny objects show reflections of their surroundings
  – To do this properly, we use ray tracing to calculate multiple bounces
  – That’s a lot of computation: how can we fake it?

  – Use a pre-rendered environment
  – (Won’t support inter-object reflections)
Environment Mapping (Spherical)

• Imagine that object is surrounded by an infinitely large sphere
  – Calculate reflection vector, project on to sphere
  – Need a seamless texture for the sphere
Environment Mapping (Cube Map)

• Same idea as spherical map, but with 6 textures forming a cube

• OpenGL supports both methods of environment mapping:
  – glEnable(GL_TEXTURE_CUBE_MAP_EXT);
  – glEnable(GL_TEXTURE_GEN_S); glEnable(GL_TEXTURE_GEN_T);

• What are the advantages/disadvantages of each method?
Color Mapping

- Mapping from $\mathbb{R} \to \mathbb{R}^3$, specifically from $(0,1) \to (0,1)^3$
- Allows us to convert scalar-valued functions to colors
- Option 1: Color table
  - Abrupt transitions, but dead simple to implement

- Option 2: Color spline
  - Linear/cubic/C-R/Hermite interpolation between colors
  - Can create both sharp transitions and smooth gradients
Procedural Textures

• What if we want to generate textures procedurally?
  – To save space
  – To get effects not possible with photographic textures (solid texture)
  – To animate textures
  – To get the correct texture scale

• Basic procedural textures are regular, boring
  – Built with periodic functions like $\sin()$, $\cos()$

• Need to add “interestingness” to textures
  – Disturb the regularity of basic textures
Perlin Noise

- Random perturbation function with the following characteristics:
  - Repeatable function of \(<x,y,z>\) input
  - Known range \([-1,1]\)
  - Band-limited (coherent)
  - No obvious periodicity
  - Stationary
  - Isotropic

White Noise

Perlin Noise
Perlin Noise Algorithm

- Perlin Noise is also known as gradient noise
  - Gradient value at each integer point in the 3-space lattice
  - Gradient determined by repeatable hash of floor(x), floor(y), floor(z)
  - Take dot product of each gradient with distance to lattice point
  - Weighted average using ease function: $6t^5 - 15t^4 + 10t^3$

http://mrl.nyu.edu/~perlin/noise/
Turbulence

- Noise has a range of \([-1,1]\), must convert to \([0,1]\)
- Can scale noise using \(0.5(\text{noise}+1)\)
- abs(noise) creates dark veins at zero crossings
Octaves of Noise

- Feature size of basic Perlin noise is relatively uniform
- Sum multiple calls to noise(), scaling input
- Typically, we scale the input by $2^i$
- Number of iterations = number of octaves

1-8 Octaves of Turbulence
Using Noise

- Clouds: \( \text{noise}(\text{point}+\text{offset})\times\text{intensity} \)
- Fire: \( \text{abs}(\text{noise3}(\text{point})+\text{offset}) \)
- Marble: \( \sin(\text{offset} + \text{turbulence}(\text{point})) \)
- Wood: \( \text{noise}(\text{point})\times\text{scale} - \text{int}(\text{noise}(\text{point})\times\text{scale}) \)
- Animated texture: add time-varying offset vector to point
Reaction Diffusion (Witkin & Kass 91)

- Originally developed by Alan Turing as a way to model morphogenesis
- Two (or more) morphogens whose concentration varies over a 2D grid
- Concentration affected by:
  - Decay of morphogens
  - Movement of morphogen from areas of high concentration to low (diffusion)
  - Interactions between morphogens which create and destroy them (reaction)
Reaction Diffusion: R Functions

- Difference in concentrations: giraffe markings
- Max of differences (3+ morphogens): maze-like
- Difference of abs (3+ morphogens): woven twigs
Reaction Diffusion

- Varying anisotropy by curvature or location allows for more complicated effects
  - Zebra’s stripes wrap correctly
  - Giraffe’s pattern is larger on smooth areas
Cellular Textures (Worley)

- Based on the idea that there are “feature points” in 3-space, and we care about the distance to the $n^{th}$ closest feature.
- $F_i(x,y)$ is the distance to the $i^{th}$ closest randomly distributed feature.
Cellular Texture Variations

- Cellular texture algorithm very “hackable”
- Change distribution of feature points
- Fractal sum of multiple octaves (water/tin foil)
- Linear/nonlinear combinations of $F_i$s
- Unique ID number per feature point (flagstones)
- Different distance metrics (Manhattan, super-quadratic)
Announcements

Written Assignment 2 is due 3/8