Texture Mapping

CS 465 Lecture 13

Texture mapping

- Objects have properties that vary across the surface
Texture Mapping

• So we make the shading parameters vary across the surface

Texture mapping

• Adds visual complexity; makes appealing images
Texture mapping

- Color is not the same everywhere on a surface
  - one solution: multiple primitives
- Want a function that assigns a color to each point
  - the surface is a 2D domain, so that is essentially an image
  - can represent using any image representation
  - raster texture images are very popular

A definition

**Texture mapping**: a technique of defining surface properties (especially shading parameters) in such a way that they vary as a function of position on the surface.

- This is very simple!
  - but it produces complex-looking effects
Examples

- Wood gym floor with smooth finish
  - diffuse color $k_D$ varies with position
  - specular properties $k_S$, $n$ are constant
- Glazed pot with finger prints
  - diffuse and specular colors $k_D$, $k_S$ are constant
  - specular exponent $n$ varies with position
- Adding dirt to painted surfaces
- Simulating stone, fabric, …
  - in many cases textures are used to approximate effects of small-scale geometry
    - they look flat but are a lot better than nothing

Mapping textures to surfaces

- Usually the texture is an image (function of $u$, $v$)
  - the big question of texture mapping: where on the surface does the image go?
  - obvious only for a flat rectangle the same shape as the image
  - otherwise more interesting
- Note that 3D textures also exist
  - texture is a function of $(u$, $v$, $w)$
  - can just evaluate texture at 3D surface point
  - good for solid materials
  - often defined procedurally
Mapping textures to surfaces

• “Putting the image on the surface”
  – this means we need a function $f$ that tells where each point on the image goes
  – this looks a lot like a parametric surface function
  – for parametric surfaces you get $f$ for free

Texture coordinate functions

• Non-parametrically defined surfaces: more to do
  – can’t assign texture coordinates as we generate the surface
  – need to have the inverse of the function $f$

  Texture coordinate fn.$\phi : S \rightarrow \mathbb{R}^2$
  – for a vtx. at $p$ get texture at $\phi(p)$
Texture coordinate functions

- Mapping from $S$ to $D$ can be many-to-one
  - that is, every surface point gets only one color assigned
  - but it is OK (and in fact useful) for multiple surface points to be mapped to the same texture point
    - e.g. repeating tiles

Texture coordinate functions

- Define texture image as a function
  \[ T : D \rightarrow C \]
  - where $C$ is the set of colors for the diffuse component
- Diffuse color (for example) at point $p$ is then
  \[ k_D(p) = T(\phi(p)) \]
Examples of coordinate functions

• A rectangle
  – image can be mapped directly, unchanged

Examples of coordinate functions

• For a sphere: latitude-longitude coordinates
  – $\phi$ maps point to its latitude and longitude
Examples of coordinate functions

• A parametric surface (e.g. spline patch)
  – surface parameterization gives mapping function directly
    (well, the inverse of the parameterization)

Examples of coordinate functions

• For non-parametric surfaces it is trickier
  – directly use world coordinates
    • need to project one out
Texture in the graphics pipeline

- Texture coordinates are another attribute
  - the application sets them to control where the texture goes
- Texturing as a fragment operation
  - because the whole point is to vary quickly across the surface
- Interpolating coordinates across triangles
  - to do texturing at fragment stage, we need interpolated \((u, v)\)
    coordinates at each fragment
  - but—sad to say—you can’t interpolate \(u\) and \(v\) linearly in
    screen space
    - not only won’t you get 0.5 at the midpoint, you’ll get
      different answers depending on the view.

Texture coordinate interp example

- Solution: interpolate \(u/w\), \(1/w\) and divide
Texture mapping demo