Texture Mapping

• Objects have properties that 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.

- $\phi : S \rightarrow \mathbb{R}^2$
  - 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

• 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)

• For non-parametric surfaces it is trickier
  – directly use world coordinates
    • need to project one out
Examples of coordinate functions

- Triangles
  - specify \((u,v)\) for each vertex
  - define \((u,v)\) for interior by linear interpolation

Barycentric coordinates (will see again)

- A coordinate system for triangles (will see this again)
  - interior point as convex affine combination of vertices
    \[ p = \alpha a + \beta b + \gamma c \]
    \[ \alpha = 1 - \beta - \gamma \]
    \[ \alpha + \beta + \gamma = 1 \]
  - Geometric viewpoint: areas

Barycentric coordinates

- A coordinate system for triangles
  - geometric viewpoint: distance ratios perpendicular to edges

Texture coordinates on meshes

- Texture coordinates become per-vertex data like vertex positions
  - can think of them as a second position: each vertex has a position in 3D space and in 2D texture space
- How to come up with vertex \((u,v)\)s?
  - use any or all of the methods just discussed
    - in practice this is how you implement those for curved surfaces approximated with triangles
  - use some kind of optimization
    - try to choose vertex \((u,v)\)s to result in a smooth, low distortion map
Cool Trick: “Advect” Texture Coords

Jos Stam, “Stable Fluids,” SIGGRAPH 99