CS4620/5620: Lecture 20

Texture Mapping

Announcements

- Extra office hours

Texture mapping

- Objects have properties that vary across the surface

Texture Mapping

- Cannot model every single change using primitives
- Instead we make the shading parameters (and other properties) vary across the surface

Texture mapping

- Textures increase apparent visual complexity of geometry and material
  - Diffuse material properties
  - Specular properties
  - Normals
  - Positions
  - Lighting....

- Increases realism

Texture Mapping: applications

- Surprisingly simple idea but with big results
  - Again uses memory
Texture mapping

- Material properties are not the same everywhere on a surface
- Want a function that assigns a color (or some other material/geometry) to each point
  - the surface is a 2D domain
  - can represent using any image representation
  - raster texture images are very popular

A definition

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

* actually, surface, normal, geometry, lighting,...

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

Examples

- Wood floor with smooth finish
  - diffuse color $k_D$ varies with position (specular properties $k_S$, $n$ are constant)
- Glazed pot with finger prints
  - specular exponent $n$ varies with position (diffuse and specular colors $k_D$, $k_S$ are constant)
- Adding dirt to painted surfaces
- Simulating stone, fabric, ... 
  - 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

Aisde: Types of Textures

- 3D Textures
- 2D Textures
  - The most common
- Procedural texturing
  - Write a piece of code

Texture mapping using images
Texture Mapping using Images

- Most common form of texturing
- Map an image onto a surface
- Assume (u,v) coordinates in texture
- Mapping $M_{\text{Tex}}^{-1}(x,y,z) \rightarrow (u,v)$
  - Between object space and texture space

How does it work?

Texture Pipeline

Projector Functions

- Planes, cylinders, spheres

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

Projector 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 $\phi(p)$
Example: texture mapping for diffuse color

- 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 projector functions

- A square/rectangle
  - image can be mapped directly, unchanged
- An arbitrary plane
  - simple affine transformation (rotate, scale, translate)
- A triangle

Examples of projector functions

- For a sphere: latitude-longitude coordinates
  - \( \phi \) maps point to its latitude and longitude

Examples of projector functions

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

Cylinder

\[ u = \frac{\theta}{2\pi}, \quad v = (1 + \sqrt{2}/2) \]

\[ \theta = \arccos(2v - 1) \]

Arbitrary Meshes
**Projector Function: Arbitrary Surfaces**
- Non-parametric surfaces: project to parametric surface

**Cylindrical**

**Spherical**

**Projector Functions: User-Specified**
- Distortion in direction perpendicular to projection
- Approach
  - Unwrap mesh
    - Set of planar projections
    - Minimize the distortion
  - Smaller textures for each of the projections
  - Pack it into a larger texture
**Examples of projector functions**

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

**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