Texture Mapping
Texture Mapping

- To enhance the visual effect of an object, trade photo-realism for efficiency
  - Texture Mapping: texture wrapped on the object
- To add pseudo-realism to shiny animated objects by causing their surrounding environment to be reflected in them
  - Environment mapping: texture moved as the objects moved
Various Mapping Techniques in CG

- 2D texture mapping
- Bump mapping
- Light maps
- Environment or reflection mapping

*Store information in a domain → for the (later) rendering*

- Texture Mapping is cheap, while global illumination computation is totally different and much more expensive
Texture Properties

What can be texture? What can be modulated with texture mapping?

- **Color**

- **Specular Color**
  - for environment reflection mapping

- **Normal Vector Perturbation**
  - Bump mapping

- **Displacement along surface normal**
  - Displacement mapping

- **Transparency**
  - Etched glass where a shiny surface is roughened (to cause opacity) with some decorative pattern
Mapping Among Different Spaces

Texture Space
(u,v)

Surface Parameterization

Geometry Space
(x,y,z)

Projection

Screen Space
(x,y)
Intermediate Mapping Methods

- Based on projection
  - Project both the model and the texture image onto an intermediate surface

Examples of two-part texture mapping. The intermediate surfaces are (left) a plane; (middle) a cylinder; and (right) a sphere.
Plane Projection
Cylinder Projection
Cube Projection
When will the projection method fail?

Solution: Intrinsic Parameterization Methods (later)
Bump Mapping and Normal Mapping

- Bump Mapping: to enable a (low resolution) surface to
  - Appear as if it were wrinkled or dimpled
  - Without the need to model these depressions geometrically
    → modify normal according to info in 2D bump map
  - Problem: geometry doesn’t change, silhouette follows original geometry

- Normal mapping: use “normal” to enrich details
The Nature of Light and Perception

- Perception: what you can see isn’t based on the objects that you are viewing but on the rays of light cast from a light source and reflected from those objects.
  - your eyes don’t directly see objects as there is no physical correlation between your eyes and those objects.
- the light rays originate from an energy source (e.g. sun, lamp) →
  - you visually perceive an object = it is the rays of light reflected or scattered off that object that your eyes absorb.
Four types of lights in OpenGL

- Lights from the light source:
  1) AMBIENT LIGHT
      light scattered, averagely brightens up the whole room
  2) DIFFUSE LIGHT
      directional light cast by a light source
  3) SPECULAR LIGHT
      reflects off the surface in a sharp and uniform way

- Lights from the object:
  4) EMISSIVE LIGHT:
      being emitted by an object

Is OpenGL Lighting Enabled?
+ glEnable(GL_LIGHTING);
+ glDisable(GL_LIGHTING);

NO

Final polygon colour is determined by: glColor*(..);

YES

Are Colour-driven Materials Enabled?
+ glEnable(GL_COLOR_MATERIAL);
+ glDisable(GL_COLOR_MATERIAL);

NO

Final polygon colours for Ambient, Diffuse, Specular and Emission components are determined by: glMaterial*(..);

YES

Which colour components are set by glColor* and which by glMaterial? 
+ glColorMaterial( .. )

Final polygon colours for components specified by glColorMaterial are set by glColor*(); - the remaining components are set by glMaterial*();
How does the normal work?

- About the lighting:
  - $B = N \cdot L$
  - (brightness $\leftarrow$ normal dot product light vector)

- Gouraud shading: (in most real-time video game models, and openGL)
  - only compute the normal and lighting on vertices
  - linear interpolate the lighting on interior pixels

- With normal texture:
  - per-pixel lighting (on each pixel, we have normal now)
How does the normal work?

original mesh

simplified mesh

normal as the texture

texture mapped simplified mesh
**Geometry as texture**

- **Geometry Image** [Gu, Gortler, and Hoppe, SIGGraph02]
  - Store the geometry \((x,y,z)\) of each vertex \(\rightarrow (r,g,b)\) in the texture space
  - Work for general surfaces, but need a cutting preprocess
    - Good cutting \(\rightarrow\) less distortion
    - An intuitive topological method to generate the cut
Environment Mapping

--Some surfaces texture should reflect the surrounding (example: Movie “ Terminator”)

• also called “reflection mapping”
  → a shortcut to rendering shiny objects that reflect its surrounding environment
  • Ray tracing process → map construction (offline) + indexing (online)
  • Nearly every 3D computer game today uses this form of texture mapping

• Not a single image wrapped onto the surface:
  • when the viewer position changes, or the object moves → the reflection changes
  ➢ should map surface points to an appropriate reflected direction in the 360 degree environment surrounding the object
Environment Mapping (cont.)

- Common Mapping Techniques for Environment Mapping:
  - Sphere Mapping
  - Cubic Mapping

- Environment texture
  - Pre-computed and stored by projections
  - indexed by a 3D direction vector

- Problems:
  - Geometrically correct when objects is small w.r.t. the environment (o/w lighting might change)
  - Only reflect the environment - not itself → wrong for concave objects (why?)
  - Separate maps are required for different objects in one scene (why?)
Light Mapping

- To enable lighting to be pre-calculated and stored as a two-dimensional texture map
  - Pre-compute:
    - Vertex brightness using distance from each vertex to a light
    - Pixel brightness using multitexture when texture map is also used
  - Shading → Indexing
  - Can stored separately from other texture maps with lower resolution

Example: Quake
(a first-person shooter video game)

Moving objects? → multiple maps + interpolation
Light Mapping (cont.)

X

=
Difficulty in 2D Texture Mapping

Need low-distorted mapping, sometimes not easy to compute!
3D Texture

- Challenges for wrapping texture images onto surfaces (for 2D texture mapping):
  - Distortion control could be non-trivial
  - Topological discontinuity could be awkward
- Procedural texture
  - Define a continuous texture function over the whole $\mathbb{R}^3$ space
  - Spatial Efficiency: functions instead of 3D texture images

...Check the paper: [Perlin Noise 1985], google “Perlin Noise”
– using noise function to simulate turbulence
3D Texture

- Challenges for wrapping texture images onto surfaces (for 2D texture mapping):
  - Distortion control could be non-trivial
  - Topological discontinuity could be awkward
- Procedural texture
  - Define a continuous texture function over the whole $\mathbb{R}^3$ space
  - Spatial Efficiency: functions instead of 3D texture images
- Texture synthesis
  - Not a mapping problem any more
  - Less texture patterns (less resources)
3D Texture

- Challenges for wrapping texture images onto surfaces (for 2D texture mapping):
  - Distortion control could be non-trivial
  - Topological discontinuity could be awkward
- Procedural texture
  - Define a continuous texture function over the whole $\mathbb{R}^3$ space
  - Spatial Efficiency
- Texture synthesis
  - Not a mapping problem any more
  - Less texture patterns (less resources) compared to 2D texture
3D Texture (cont.)

Big visual difference:

-- Texture moving with the object
Next:

The surface mapping (parameterization) problem:

How to control the distortion? (What distortion?)
How about other issues - boundary continuity, poles?