CS354 Computer Graphics
Texture Mapping

Qixing Huang
February 5th 2018
What adds visual realism?

Geometry Only
What adds visual realism?

Phong Shading
What adds visual realism?

Phong shading + Texture Maps
Textures Supply Rendering Detail

Without texture

With texture
Textures Make Graphics Pretty

Texture → detail, detail → immersion, immersion → fun

Microsoft Flight Simulator X
Texture Mapping

Texture mapping (Woo et al., fig. 9-1)

- Texture mapping allows you to take a simple polygon and give it the appearance of something much more complex
  - Due to Ed Catmull, PhD thesis, 1974
  - Refined by Blinn & Newell, 1976
Non-parametric Texture Mapping

• With “non-parametric texture mapping”
  – Texture size and orientation are fixed
  – They are unrelated to size and orientation of polygon
With “parametric texture mapping,” texture size and orientation are tied to the polygon
- Separate “texture space” and “screen space”
- Texture the polygon as before, but in texture space
- Deform (render) the textured polygon into screen space
- Deformation is given by parameterization

A texture can modulate just about any parameter – diffuse color, specular color, specular exponent, ...
Implementing Texture Mapping

- A texture lives in its own abstract image coordinates parameterized by \((u,v)\) in the range \([0..1], [0..1]\):

- It can be wrapped around many different surfaces:

- Computing \((u,v)\) texture coordinates in a ray tracer is fairly straightforward

- Note: if the surface moves/deforms, the texture goes with it
Mapping to Texture Image Coords

The texture is usually stored as an image. Thus, we need to convert from abstract texture coordinate:

\((u,v)\) in the range \(([0..1], [0..1])\)

to texture image coordinates:

\((u_{\text{tex}}, v_{\text{tex}})\) in the range \(([0.. w_{\text{tex}}], [0.. h_{\text{tex}}])\)

**Q:** What do you do when the texture sample you need lands between texture pixels?
Texture Resampling

• We need to resample the texture:

• A common choice is **bilinear interpolation**: 

\[
T(a,b) = T[i + \Delta_x, j + \Delta_y] \\
= (1 - \Delta_x)(1 - \Delta_y)T[i,j] + \Delta_x(1 - \Delta_y)T[i+1,j] \\
+ (1 - \Delta_x)\Delta_y T[i,j+1] + \Delta_x\Delta_y T[i+1,j+1]
\]
Texture Coordinates

- Interpolated over rasterized primitives

*parametric coordinates*

*texture coordinates*

*world coordinates*

*window coordinates*
Source of Texture Coordinates?

• Assigned ad-hoc by artist
  – Tedious!
  – Has gift wrapping problem

• Computed based on XYZ position
  – Texture coordinate generation ("texgen")
  – Hard to map to "surface space"

• From bi-variate parameterization of geometry
  – Good when geometry is generated from patches
  – So \((u,v)\) of patch maps to \((x,y,z)\) and \((s,t)\)
Texture Arrays

- **Multiple skins packed in texture array**
  - Motivation: binding to one multi-skin texture array avoids texture bind per object
Textured Polygonal Models

Key-frame model geometry + Decal skin = Result
Multiple Textures

lightmaps only


deal only

(modulate)

= combined scene

* Id Software’s Quake 2 circa 1997
Can Define Material by Program

- A ‘surface shader’ computes the color of each ray that hits the surface
- Example: Renderman surface shader

```c
surface checker(float Kd=.5, Ka=.1) {
    float smod = mod(10*s, 1);
    float tmod = mod(10*t, 1);
    if (smod < 0.5) {
        if (tmod < 0.5) Ci=Cs; else Ci=color(0,0,0);
    } else {
        if (tmod < 0.5) Ci=color(0,0,0); else Ci=Cs;
    }
    Oi = Os;
    Ci = Oi*Ci*(
        Ka*ambient() +
        Kd*diffuse(faceforward(normalize(N),I)));
}
```
Solid Textures

- **Q:** What kinds of artifacts might you see from using a marble veneer instead of real marble?

- One solution is to use **solid textures**:
  - Use model-space coordinates to index into a 3D texture
  - Like “carving” the object from the material

- One difficulty of solid texturing is coming up with the textures
Solid Textures

• Here's an example for a vase cut from a solid marble texture:

Solid marble texture by Ken Perlin, (Foley, IV-21)
Displacement and Bump Mapping

- Use surface offsets stored in texture
  - Perturb or dispute the surface
  - Shade on the resulting surface normals

\[
P(u,v) \\
S(u,v) = \frac{\partial P(u,v)}{\partial u} \quad T(u,v) = \frac{\partial P(u,v)}{\partial v} \\
N(u,v) = S \times T
\]

- **Displacement**
  \[
P'(u,v) = P(u,v) + h(u,v)N(u,v)
  \]

- **Perturbed normal**
  \[
  N'(u,v) = P'_u \times P'_v \\
  = N + h_u(T \times N) + h_v(S \times N)
  \]

From Blinn 1976
Normal Mapping

• Bump mapping via a normal map texture
  – Normal map – x,y,z components of actual normal
  – Instead of a height field 1 value per pixel
  – The normal map can be generated from the height field
  – Otherwise have to orient the normal coordinates to the surface
Displacement vs. Bump Mapping

- Rendered as displacement map over a rectangular surface
Bump mapping is much faster and consumes less resources for the same level of detail compared to displacement mapping because the geometry remains unchanged.
Bump Mapping Example

Texture #1 (diffuse color)  Texture #2 (bump map)  Rendered Image
Combing Texture Maps

- Using texture maps in combination gives even better effects
Multiple Textures
Multitexturing

\[
\begin{align*}
&\text{Diffuse} \times \text{Decal} + \text{Specular} \times \text{Gloss} = \\
&\text{Final result!}
\end{align*}
\]
Next Lecture

• Continue texture mapping

• Spatial data structure
Questions?