Texture Mapping
Texture Mapping: Motivation

- Scenes created with diffuse lighting look convincingly three-dimensional, but are flat, chalky, and “cartoonish”
- Phong lighting lets us simulate materials like plastic and (to a lesser extent) metal, but scenes still seem very cartoonish and unreal
- Big problem: polygons are too coarse-grained to usefully model fine surface detail
- Solution: Texture mapping
Texture Mapping: Motivation

- **Adding surface detail** helps keep CG images from looking simple and sterile
- Explicitly modeling this detail in geometry can be very expensive
  - Zebra stripes, wood grain, writing on a whiteboard
- **Texture mapping** pastes images onto the surfaces in the scene, adding realistic fine detail without exploding the geometry
Texture Mapping: Examples
The Problem:

• We don't want to represent all this detail with geometry
The Quest for Visual Realism

Model

Model + Shading

Model + Shading + Textures

At what point do things start looking real?

For more info on the computer artwork of Jeremy Birn, see [http://www.3drender.com/jbirn/productions.html](http://www.3drender.com/jbirn/productions.html)
Texture Mapping

- In short: it is impractical to explicitly model fine surface detail with geometry
- Solution: use images to capture the “texture” of surfaces
- Texture maps can modulate many factors that affect the rendering of a surface
  - Color or reflectance (diffuse, ambient, specular)
  - Transparency (smoke effects)
  - What else?
Texture Mapping: Fundamentals

- A texture is typically a 2-D image
  - Image elements are called **texels**
  - Value stored at a texel affects surface appearance in some way
    - Example: diffuse reflectance, shininess, transparency…
  - The **mapping** of the texture to the surface determines the correspondence, i.e., how the texture lies on the surface
    - Mapping a texture to a triangle is easy (**why?**)
    - Mapping a texture to an arbitrary 3-D shape is more complicated (**why?**)
Texture Mapping

- Increase the apparent complexity of simple geometry
- Like wallpapering or gift-wrapping with stretchy paper
- Curved surfaces require extra stretching or even cutting
Photo-textures

For each triangle in the model establish a corresponding region in the phototexture.

During rasterization interpolate the coordinate indices into the texture map.
Texture Mapping Difficulties

- Tedious to specify texture coordinates
- Acquiring textures is surprisingly difficult
  - Photographs have projective distortions
  - Variations in reflectance and illumination
- Tiling problems
Common Texture Coordinate Mappings

- Orthogonal
- Cylindrical
- Spherical
- Perspective Projection
- Texture Chart
Texture Mapping in OpenGL (checker.c)

#include <GL/glut.h>
#include <stdlib.h>
#include <stdio.h>

/* Create checkerboard texture */
#define checkImageWidth 64
#define checkImageHeight 64
static GLubyte checkImage[checkImageHeight][checkImageWidth][4];

static GLuint texName;

void makeCheckImage(void)
{
 int i, j, c;

 for (i = 0; i < checkImageHeight; i++) {
 for (j = 0; j < checkImageWidth; j++) {
  c = (((i&0x8)==0)^((j&0x8))==0))*255;
  checkImage[i][j][0] = (GLubyte) c;
  checkImage[i][j][1] = (GLubyte) c;
  checkImage[i][j][2] = (GLubyte) c;
  checkImage[i][j][3] = (GLubyte) 255;
 }
 }
}
void init(void)
{
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glShadeModel(GL_FLAT);
    glEnable(GL_DEPTH_TEST);

    makeCheckImage();
    glPixelStorei(GL_UNPACK_ALIGNMENT, 1);

    glGenTextures(1, &texName);
    glBindTexture(GL_TEXTURE_2D, texName);

    glTexParameter(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameter(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexParameter(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
    glTexParameter(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);

    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, checkImageWidth, checkImageHeight,
                 0, GL_RGBA, GL_UNSIGNED_BYTE, checkImage);
}
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glEnable(GL_TEXTURE_2D);
    glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);
    glBindTexture(GL_TEXTURE_2D, texName);
    glBegin(GL_QUADS);
    glTexCoord2f(0.0, 0.0); glVertex3f(-2.0, -1.0, 0.0);
    glTexCoord2f(0.0, 1.0); glVertex3f(-2.0, 1.0, 0.0);
    glTexCoord2f(1.0, 1.0); glVertex3f(0.0, 1.0, 0.0);
    glTexCoord2f(1.0, 0.0); glVertex3f(0.0, -1.0, 0.0);
    glTexCoord2f(0.0, 0.0); glVertex3f(1.0, -1.0, 0.0);
    glTexCoord2f(0.0, 1.0); glVertex3f(1.0, 1.0, 0.0);
    glTexCoord2f(1.0, 1.0); glVertex3f(2.41421, 1.0, -1.41421);
    glTexCoord2f(1.0, 0.0); glVertex3f(2.41421, -1.0, -1.41421);
    glEnd();
    glFlush();
    glDisable(GL_TEXTURE_2D);
}
void reshape(int w, int h)
{
    glViewport(0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60.0, (GLfloat) w/(GLfloat) h, 1.0, 30.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -3.6);
}

void keyboard (unsigned char key, int x, int y)
{
    switch (key) {
    case 27:
        exit(0);
        break;
    default:
        break;
    }
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(250, 250);
    glutInitWindowPosition(100, 100);
    glutCreateWindow(argv[0]);
    init();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard);
    glutMainLoop();
    return 0;
}
Output (checker.c)
Texture Mapping & Illumination

- Texture mapping can be used to alter some or all of the constants in the illumination equation:
  - pixel color, diffuse color, alter the normal, ....

\[
I_{\text{total}} = k_a I_{\text{ambient}} + \sum_{i=1}^{\text{lights}} I_i \left( k_d \left( \hat{N} \cdot \hat{L} \right) + k_s \left( \hat{V} \cdot \hat{R} \right)^{n_{\text{shiny}}} \right)
\]

Phong's Illumination Model

- Constant Diffuse Color
- Diffuse Texture Color
- Texture used as Label
- Texture used as Diffuse Color
Texture Chart

- Pack triangles into a single image
Procedural Textures

\[ f(x, y, z) \rightarrow \text{color} \]

Image by Turner Whitted
Procedural Textures

- **Advantages:**
  - easy to implement in ray tracer
  - more compact than texture maps (especially for solid textures)
  - infinite resolution

- **Disadvantages**
  - non-intuitive
  - difficult to match existing texture
Examples of Procedural Textures

Ken Perlin

Justin Legakis

Demo: http://graphics.lcs.mit.edu/classes/6.837/F98/Lecture22/Slide27.html
What's Missing?

- What's the difference between a real brick wall and a photograph of the wall texture-mapped onto a plane?

- What happens if we change the lighting or the camera position?
Remember Gouraud Shading?

- Instead of shading with the normal of the triangle, shade the vertices with the average normal and interpolate the color across each face.

*Illusion* of a smooth surface with smoothly varying normals.
Normal Interpolation

- Interpolate the average vertex normals across the face and compute *per-pixel shading*

Must be renormalized
Bump Mapping

- Use textures to alter the surface normal
  - Does not change the actual shape of the surface
  - Just shaded as if it were a different shape
Bump Mapping

- Treat the texture as a single-valued height function
- Compute the normal from the partial derivatives in the texture
Another Bump Map Example

Cylinder w/Diffuse Texture Map

Bump Map

Cylinder w/Texture Map & Bump Map

Demo CrazyMap
What's Missing?

- There are no bumps on the silhouette of a bump-mapped object.

- Bump maps don’t allow self-occlusion or self-shadowing.
Displacement Mapping

- Use the texture map to actually move the surface point
- The geometry must be displaced before visibility is determined
Displacement Mapping

Image from: 

Geometry Caching for Ray-Tracing Displacement Maps

by Matt Pharr and Pat Hanrahan.

note the detailed shadows cast by the stones
Displacement Mapping

Ken Musgrave
Environment Maps

- We can simulate reflections by using the direction of the reflected ray to index a spherical texture map at "infinity".
- Assumes that all reflected rays begin from the same point.
What's the Best Chart?

Box Map

Latitude Map

GL Map
Environment Mapping Example

Terminator II
Environment mapping example

Image by Henrik Wann Jensen
Environment map by Paul Debevec
Illumination Maps

- Quake introduced *illumination maps* or *light maps* to capture lighting effects in video games.

Texture map:

*Light map*

Texture map + light map:
Illumination Maps

- Also called "Light Maps"
Illumination Maps

• Illumination maps differ from texture maps in that they:
  – Usually apply to only a single surface
  – Are usually fairly low resolution
  – Usually capture just intensity (1 value) rather than color (3 values)

• Illumination maps can be:
  – Painted by hand: Disney’s Aladdin ride
  – Calculated by a global illumination process
Other Texture Applications

- Lots of other interesting applications of the texture-map concept (we’ll return to some):
  - Shadow maps
  - 3-D textures (marble, wood, clouds)
  - Procedural textures
  - Environment maps & cube maps

- For a neat explanation of the first three (with cool applets, as usual) check out:
MIP-maps

- For a texture of $2^n \times 2^n$ pixels, compute $n-1$ textures, each at $\frac{1}{2}$ the resolution of previous:

  - This multiresolution texture is called a **MIP-map**
Generating MIP-maps

• Generating a MIP-map from a texture is easy
  - For each texel in level $i$, average the values of the four corresponding texels in level $i-1$

• If a texture requires $n$ bytes of storage, how much storage will a MIP-map require?

• Answer: $4n/3$
Representing MIP-maps

Trivia: MIP = *Multum In Parvo* (many things in a small place)
Using MIP-maps

- Each level of the MIP-map represents a pre-blurred version of multiple texels
  - A texel at level $n$ represents $2^n$ original texels
- When rendering:
  1. Figure out the texture coverage of the pixel (i.e., the size of the pixel in texels of the original map)
  2. Find the level of the MIP map in which texels average approximately that many original texels
  3. Interpolate the value of the four nearest texels
Using MIP-maps

- **Even better:**
  1. Likely, the coverage of the pixel will fall somewhere between the coverage of texels in two adjacent levels of the MIP map
  2. Find the pixel’s value in each of the two textures using two bilinear interpolations
  3. Using a **third interpolation**, find a value in between these two values, based on the coverage of the pixel versus each of the MIP-map levels
  4. This is (misleadingly?) called **trilinear interpolation**
Using MIP-maps

- How many interpolations does a texture lookup using trilinear interpolation in a MIP-mapped texture involve?

- How many texel values from the MIP-map must be fetched for such a lookup?
MIP-map Example

- No filtering:

- MIP-map texturing:
Textures can Alias

- **Aliasing** is the under-sampling of a signal, and it's especially noticeable during animation.
Textures can Alias

- Small details may "pop" in and out of view

[Images showing nearest neighbor and mipmaps & linear interpolation]
Texture Map Aliasing

- Naive texture mapping looks **blocky, pixelated**
  - Problem: using a single texel to color each pixel:
    
    ```
    int uval = (int) (u * denom + 0.5f);
    int vval = (int) (v * denom + 0.5f);
    int pix = texture.getPixel(uval, vval);
    ```

- Actually, **each pixel maps to a region in texture**
  - If the pixel is larger than a texel, we should **average** the contribution from multiple texels somehow
  - If the pixel is smaller than a texel, we should **interpolate** between texel values somehow
  - Even if pixel size \(\approx\) texel size, a pixel will in general **fall between four texels**

- An example of the general problem called **aliasing**
Texture Map Antialiasing

- Use *bilinear interpolation* to average nearby texel values into a single pixel value
  - Find 4 nearest texture samples
    - Round $u$ & $v$ up and down
  - Interpolate texel values in $u$
  - Interpolate resulting values in $v$

- Also addresses the problem of many pixels projecting to a single texel (*Why?*)
Moiré Patterns

- A moiré pattern is the combination of two or more patterns viewed at the same time.

http://www.sandlotscience.com/Moire/Moire_frm.htm
Texture Map Antialiasing

- What if a single pixel covers many texels?
  - Problem: sampling those texels at a single point (the center of the pixel):
    - Produces Moire patterns in coherent texture (checkers)
    - Leads to flicker or *texture crawling* as the texture moves
  - Approach: blur the image under the pixel, averaging the contributions of the covered texels
    - But calculating which texels every pixel covers is way too expensive, especially as the texture is compressed
  - Solution: pre-calculate lower-resolution versions of the texture that incorporate this averaging (Mipmapping)
Can We Do Better?

- *What assumption does MIP-mapping implicitly make?*
- A: The pixel covers a square region of the texture
  - More exactly, the compression or oversampling rate is the same in $u$ and $v$
- *Is this a valid assumption? Why or why not?*
MIP-maps and Signal Processing

- An aside: aliasing and antialiasing are topics in *sampling theory*
- Nyquist theorem, convolution and reconstruction, filters and filter widths
- Textures are particularly difficult because a tiled texture can easily generate infinite frequencies
  - E.g., a checkered plane receding to an infinite horizon
- Using a MIP-map amounts to *prefiltering* the texture image to reduce artifacts caused by sampling at too low a rate
Texture Synthesis

Basic problem
- You have a sample of some texture
- You want to be able to generate more of the same

Applications
- Create new textures for objects
- Infill missing regions
Sample Replication
Multiresolution Analysis

Jeremy De Bonet, SIGGRAPH ‘97,

Multiresolution Sampling Procedure for Analysis and Synthesis of Texture Images

Basic Concept

Analyze sample to generate global conditional distribution function

Generate new samples from that function
Multiresolution Synthesis

- Analyze sample
  - joint features
  - multiple resolutions
- Synthesize texture with similar joint features
Multiresolution synthesis
Trivia

- Ultima Underworld is THE forefather of modern continuous-movement first-person texture-mapped gaming.
- Reportedly it was a demonstration of the in-development Underworld technology that prompted John Carmack to write the Wolfenstein 3D engine.
- Wolf 3D made it to the shelves only a short time after Underworld.