Bump Mapping

Which one of these two images has a better visual effect?
Bump Mapping

Or, these two?
Bump Mapping

- Many textures are the result of small perturbations in the surface geometry.
- Modeling these changes would result in an explosion in the number of geometric primitives.
- Bump mapping attempts to alter the lighting across a polygon to provide the illusion of texture. How?
Bump Mapping

Perturb normals of a surface - creates a bump-like effect

*Does not change silhouette edges*

*Introduced by James Blinn in 1978*
Bump Mapping

The sphere looks bumpy
But the underlying surface
is not changed
Bump Mapping — methods

Approach 1: for each visible point:

1. Look up the height in the heightmap that corresponds to the position on the surface.
2. Calculate the surface normal of the heightmap, using the finite difference method.
3. Combine the surface normal from step two with the true ("geometric") surface normal so that the combined normal points in a new direction.
4. Calculate the interaction of the new "bumpy" surface with lights in the scene using the Phong reflection model.
Bump Mapping — Heightmap (Heightfield)

Heightmap:
- A raster image containing one channel interpreted as a distance of displacement or “height” from the “floor” of a surface.
- Black representing minimum height and white representing maximum height.
- Used in bump mapping to calculate where this 3D data would create shadow in a material.
- In displacement mapping to displace the actual geometric position of points over the textured surface.
Bump Mapping — heightmap (heightfield)

The same heightmap converted to a 3D mesh

Heightmap of Earth's surface in equirectangular projection, normalized as 8-bit grayscale
Bump Mapping — methods

Approach 2: for each visible point:

1. Look for the specified normal for that point in a normal map that contains the modified normal for each point on the surface.
2. Use the specified normal to calculate the interaction of the new "bumpy" surface with lights in the scene using the Phong reflection model.

Leads to more predictable results

Easier for artists to work with

most common method of bump mapping today
Bump Mapping

- This modifies the surface normals using approach 2 (left) and approach 1 (right)
Bump Mapping — approach 2
Bump Mapping – approach 2
Bump Mapping – approach 2
Displacement Mapping

Use texture to **displace** the surface geometry

Bump mapping only affects the normals,
Displacement mapping changes the entire surface (including the silhouette)
Displacement Mapping

- Consider the lighting for a modeled surface.
Displacement Mapping

- We can model this as deviations from some base surface.
- The question is then how these deviations change the lighting.
Displacement Mapping

- Step 1: Putting everything into the same coordinate frame as B(u,v).
  - $x(u,v), y(u,v), z(u,v)$ – this is given for parametric surfaces, but easy to derive for other analytical surfaces.
  - Or $O(u,v)$
Displacement Mapping

- Define the tangent plane to the surface at a point \((u,v)\) by using the two vectors \(O_u\) and \(O_v\).
- The normal is then given by:
  - \(N = O_u \times O_v\)
The new surface positions are then given by:

\[ \mathbf{O}'(u,v) = \mathbf{O}(u,v) + \mathbf{B}(u,v) \mathbf{N} \]

Where, \( \mathbf{N} = \mathbf{N} / |\mathbf{N}| \)

Differentiating leads to:

\[ \mathbf{O}'_u = \mathbf{O}_u + \mathbf{B}_u \mathbf{N} + \mathbf{B} (\mathbf{N})_u \approx \mathbf{O}'_u = \mathbf{O}_u + \mathbf{B}_u \mathbf{N} \]

\[ \mathbf{O}'_v = \mathbf{O}_v + \mathbf{B}_v \mathbf{N} + \mathbf{B} (\mathbf{N})_v \approx \mathbf{O}'_v = \mathbf{O}_v + \mathbf{B}_v \mathbf{N} \]

If \( \mathbf{B} \) is small.
Displacement Mapping

- This leads to a new normal:

\[ N'(u,v) = O_u \times O_v - B_u(N \times O_v) + B_v(N \times O_u) + B_u B_v(N \times N) \]

- \[ = N - B_u(N \times O_v) + B_v(N \times O_u) \]

- \[ = N + D \]
Displacement Mapping

- For efficiency, can store $B_u$ and $B_v$ in a 2-component texture map.
- The cross products are geometric terms only.
- $N'$ will of course need to be normalized after the calculation and before lighting.
  - This floating point square root and division makes it difficult to be embedded into hardware.
Displacement Mapping

Do you see the difference?
3D Textures

Use a 3D mapping

\((x_o, y_o, z_o) \rightarrow (r, s, t)\)

Usually stored procedurally

Can simulate an object carved from a material
3D Textures - Noise and Turbulence

- Useful tools when we want to add random variations to our texture, but in a controllable way.
- A noise function is a continuous function that varies throughout space at a uniform frequency.
- To create a simple noise function, consider a 3D lattice, with a random value assigned to each triple of integer coordinates.
Create a 3D integer-aligned lattice of random numbers

For any 3DPoint \( p \), noise is defined as:

\[
\text{noise}(\text{nDPoint } p) = \begin{align*}
\text{Find 2 neighbors of } p \\
\text{Linearly interpolate neighbors' table values} \\
\text{Return interpolated value}
\end{align*}
\]
Noise by itself looks pretty ugly. We can use noise to make a more interesting function called turbulence. A turbulence functions can be computed by summing many different frequencies of noise functions.
3D Textures - Noise and Turbulence

Noise with self-similarity

Add together many octaves of noise

\[ turbulence(x) = \sum_{i=0}^{k} \frac{1}{2^i} \left| \text{noise}(2^i x) \right| \]
Marble example

- However, even turbulence is rarely used all by itself. We can use turbulence to build more fancy 3D textures, such as 3D marble textures [Ken Perlin].

- Basic idea:
  - use a \textit{sin} wave function to fill space with black and white stripes
  - use \textit{turbulence} to distort those planes
    \[ \text{Marble} = \sin(f \times (x + A \times \text{Turb}(x, y, z))) \]
  - control the thickness of the veins by varying the \textit{frequency} of the \textit{sin} function and control the distortion of the veins by varying the \textit{amplitude} of the turbulence function

\textit{Slide Courtesy of Leonard McMillan & Jovan Popovic, MIT}
3D Textures - Noise and Turbulence

Simplified versions

\[ \text{turbulence}(x) \quad \text{sin}(x + \text{turbulence}(x)) \]
Animating Turbulence

*Use an extra dimension as time*
End of Texture-mapping II