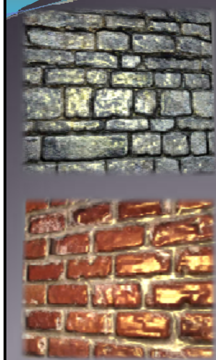
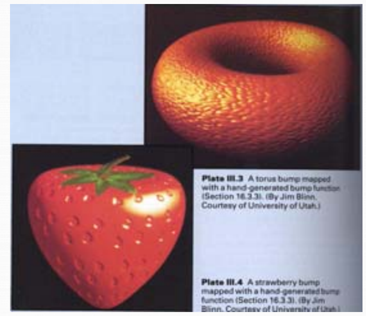
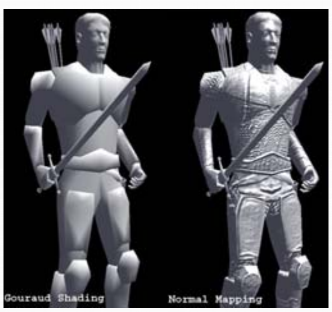


Bump Mapping

Application

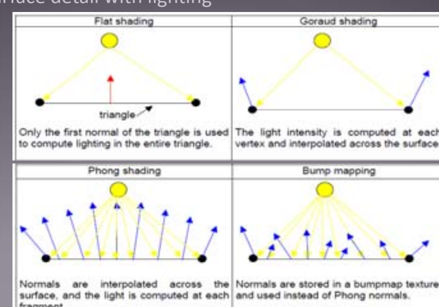


Application



Shading

Fake surface detail with lighting



Benefits

- A higher level of visual complexity in a scene, without adding more geometry.
- Simplified content authoring, because you can encode surface detail in textures as opposed to requiring artists to design highly detailed 3D models.
- The ability to apply different bump maps to different instances of the same model to give each instance a distinct surface appearance. For example, a building model could be rendered once with a brick bump map and a second time with a stucco bump map.

Look at the Mathematics

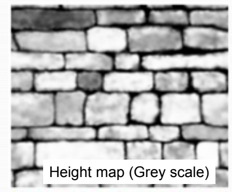
- Nvidia slides on mathematics of bump-mapping

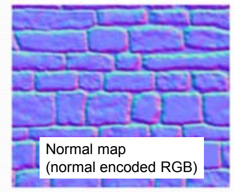
Normal Map

- Normal vector encoded as rgb
  - $[-1,1]^3 \rightarrow [0,1]^3$: rgb = n*0.5 + 0.5
- RGB decoding in fragment shaders
  - vec3 n = texture2D(NormalMap, texcoord.st).xyz * 2.0 – 1.0
- In tangent space, the default (unit) normal points in the +z direction.
  - Hence the RGB color for the straight up normal is (0.5, 0.5, 1.0). This is why normal maps are a blueish color
- Normals are then used for shading computation
  - Diffuse: n•l
  - Specular: $(n \cdot h)^{shininess}$
  - Computations done in tangent space

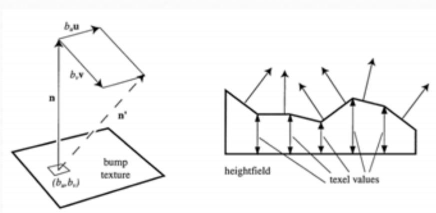
Maps


Base texture (RGB)


Height map (Grey scale)

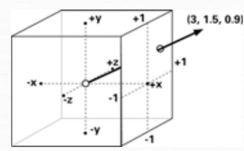

Normal map
(normal encoded RGB)

Normal Map & Height Field



Cg Book: Normalization CubeMap

- What is it?
- Why do it?



- (3, 1.5, 0.9) ->

(0.93, 0.72, 0.63)
Expand (scale/bias)
Bias=-0.5, scale =2
Bias: (0.43, 0.22, 0.13)
Scale: (0.86, 0.44, 0.26)
Approximate normalization of (3, 1.5, 0.9)

Brick Wall

- Render wall in X-Y plane (Z is normal direction)
- What's the normal?
- When rendering, perturb the normal with a normal map.
- How?

Demo

What about 2 planes?

Tangent Space

- Do the lighting to take advantage of the normal map
- Consider a floor, normals are (0, 1, 0), normal map expects (0, 0, 1)
- Need to rotate floor normals into texture-space

$$\begin{bmatrix}0 & 0 & 1\end{bmatrix} = \begin{bmatrix}0 & 1 & 0\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0 & 0 & 1\\ 0 & -1 & 0\end{bmatrix}$$

- Lights too!

$$L' = \begin{bmatrix}L_x' & L_y' & L_z'\end{bmatrix} = \begin{bmatrix}L_x & -L_z & L_y\end{bmatrix} = \begin{bmatrix}L_x & L_y & L_z\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0 & 0 & 1\\ 0 & -1 & 0\end{bmatrix}$$

Tangent Space

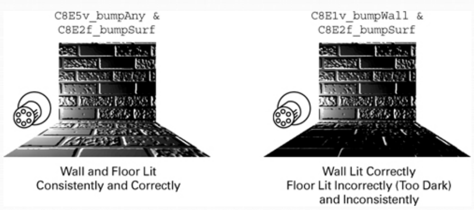
- Tangent, Bi-tangent and Normal can form a rotation matrix too:

$$\begin{bmatrix}T_x & B_x & N_x\\ T_y & B_y & N_y\\ T_z & B_z & N_z\end{bmatrix}$$

- Orthonormal matrix:

$$B = N \times T$$
$$N = T \times B$$
$$T = B \times N$$

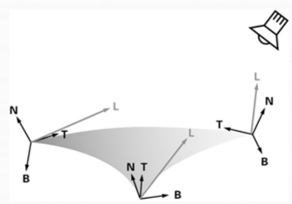
2 planes



- Demo

Tangent Space

- Each vertex has a Normal and a Tangent



Torus

- Use differential geometry to compute Tangent
- Torus:

$$x = (M + N\cos(2\pi t))\cos(2\pi s)$$
$$y = (M + N\cos(2\pi t))\sin(2\pi s)$$
$$z = N\sin(2\pi t)$$

  - M is the radius from the center of the hole to the center of the torus tube,
  - N is the radius of the tube.
  - The torus lies in the z=0 plane and is centered at the origin.
  - Parametric in [s, t]

Torus

- Use differential geometry to compute Tangent
- Torus:

$$x = (M + N\cos(2\pi t))\cos(2\pi s)$$
$$y = (M + N\cos(2\pi t))\sin(2\pi s)$$
$$z = N\sin(2\pi t)$$

$$\frac{\partial x}{\partial s} = -2\pi(M + N\cos(2\pi t))\sin(2\pi s) \qquad \frac{\partial x}{\partial t} = -2N\pi\sin(2\pi t)\cos(2\pi s)$$
$$\frac{\partial y}{\partial s} = 2\pi(M + N\cos(2\pi t))\cos(2\pi s) \qquad \frac{\partial y}{\partial t} = -2N\pi\cos(2\pi t)\sin(2\pi s)$$
$$\frac{\partial z}{\partial s} = 0 \qquad \frac{\partial z}{\partial t} = 2N\pi\cos(2\pi t)$$

Torus

$$\frac{\partial x}{\partial s} = -2\pi(M + N\cos(2\pi t))\sin(2\pi s) \qquad \frac{\partial x}{\partial t} = -2N\pi\sin(2\pi t)\cos(2\pi s)$$
$$\frac{\partial y}{\partial s} = 2\pi(M + N\cos(2\pi t))\cos(2\pi s) \qquad \frac{\partial y}{\partial t} = -2N\pi\cos(2\pi t)\sin(2\pi s)$$
$$\frac{\partial z}{\partial s} = 0 \qquad \frac{\partial z}{\partial t} = 2N\pi\cos(2\pi t)$$

$$N = \left\langle \frac{\partial x}{\partial s}, \frac{\partial y}{\partial s}, \frac{\partial z}{\partial s} \right\rangle \times \left\langle \frac{\partial x}{\partial t}, \frac{\partial y}{\partial t}, \frac{\partial z}{\partial t} \right\rangle$$

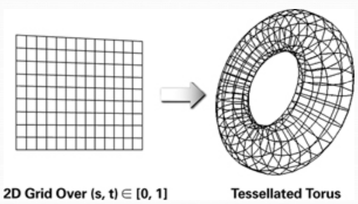
$$N = \langle \cos(s)\cos(t), \sin(s)\cos(t), \sin(t) \rangle$$

Torus

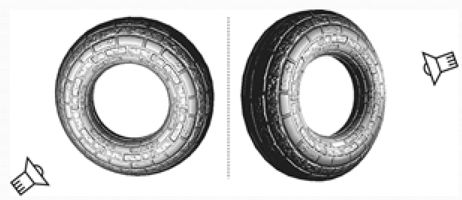
$$\frac{\partial x}{\partial s} = -2\pi(M + N\cos(2\pi t))\sin(2\pi s) \qquad \frac{\partial x}{\partial t} = -2N\pi\sin(2\pi t)\cos(2\pi s)$$
$$\frac{\partial y}{\partial s} = 2\pi(M + N\cos(2\pi t))\cos(2\pi s) \qquad \frac{\partial y}{\partial t} = -2N\pi\cos(2\pi t)\sin(2\pi s)$$
$$\frac{\partial z}{\partial s} = 0 \qquad \frac{\partial z}{\partial t} = 2N\pi\cos(2\pi t)$$

$$T = \left\langle \frac{\partial x}{\partial s}, \frac{\partial y}{\partial s}, \frac{\partial z}{\partial s} \right\rangle$$
$$B = N \times T$$
$$N = \left\langle \frac{\partial x}{\partial s}, \frac{\partial y}{\partial s}, \frac{\partial z}{\partial s} \right\rangle \times \left\langle \frac{\partial x}{\partial t}, \frac{\partial y}{\partial t}, \frac{\partial z}{\partial t} \right\rangle$$

Torus



Torus



Torus

- Demo

What about polygons?

- Don't have equations for differential geometry
- Need local tangent space frame
  - Align bump-map coordinate system with frame
    - S with Tangent and T with Bi-Tangent
- Not that hard (the Cg book is less clear than Lengyel's Method)

Lengyel's Method

- **For some point Q in a triangle (P_0, P_1, P_2):**

  $\mathbf{Q} - \mathbf{P_0} = (u - u_0)\mathbf{T} + (v - v_0)\mathbf{B}$

  T = tangent vector
  B = bitangent vector
  P_0 = 1st vertex
  u_0 = s texture coordinate
  v_0 = t texture coordinate

Lengyel's Method

- Triangle: (using his notation, (s,t) = (u,v)
  - Vertex attributes are defined by OpenGL:
    P_0 (u_0,v_0) P_1 (u_1,v_1) P_2 (u_2,v_2)

  $\mathbf{Q_1} = \mathbf{P_1} - \mathbf{P_0}$ $(s_1, t_1) = (u_1 - u_0, v_1 - v_0)$
  $\mathbf{Q_2} = \mathbf{P_2} - \mathbf{P_0}$ $(s_2, t_2) = (u_2 - u_0, v_2 - v_0)$

  So:
  $\mathbf{Q_1} = s_1\mathbf{T} + t_1\mathbf{B}$
  $\mathbf{Q_2} = s_2\mathbf{T} + t_2\mathbf{B}$

  Need to solve for **T** and **B**

Lengyel's Method

set up a linear system:

$$\mathbf{Q}_1 = s_1\mathbf{T} + t_1\mathbf{B}$$
$$\mathbf{Q}_2 = s_2\mathbf{T} + t_2\mathbf{B}$$

Write in matrix form:

$M_{Q_1Q_2} = M_{ST} \; M_{TB}$

$$\begin{bmatrix} (\mathbf{Q}_1)_x & (\mathbf{Q}_1)_y & (\mathbf{Q}_1)_z \\ (\mathbf{Q}_2)_x & (\mathbf{Q}_2)_y & (\mathbf{Q}_2)_z \end{bmatrix} = \begin{bmatrix} s_1 & t_1 \\ s_2 & t_2 \end{bmatrix} \begin{bmatrix} T_x & T_y & T_z \\ B_x & B_y & B_z \end{bmatrix}$$

Lengyel's Method

Write in matrix form:

$M_{Q_1Q_2} = M_{ST} \; M_{TB}$

$$\begin{bmatrix} (\mathbf{Q}_1)_x & (\mathbf{Q}_1)_y & (\mathbf{Q}_1)_z \\ (\mathbf{Q}_2)_x & (\mathbf{Q}_2)_y & (\mathbf{Q}_2)_z \end{bmatrix} = \begin{bmatrix} s_1 & t_1 \\ s_2 & t_2 \end{bmatrix} \begin{bmatrix} T_x & T_y & T_z \\ B_x & B_y & B_z \end{bmatrix}$$

Multiply each side by M^{-1}_{ST}

$M^{-1ST} * M_{Q_1Q_2} = M^{-1}_{ST} * M_{ST} \; M_{TB}$

$M_{TB} = M^{-1ST} * M_{Q_1Q_2}$

$$\begin{bmatrix} T_x & T_y & T_z \\ B_x & B_y & B_z \end{bmatrix} = \frac{1}{s_1t_2 - s_2t_1} \begin{bmatrix} t_2 & -t_1 \\ -s_2 & s_1 \end{bmatrix} \begin{bmatrix} (\mathbf{Q}_1)_x & (\mathbf{Q}_1)_y & (\mathbf{Q}_1)_z \\ (\mathbf{Q}_2)_x & (\mathbf{Q}_2)_y & (\mathbf{Q}_2)_z \end{bmatrix}.$$

Lengyel's Method

$$\begin{bmatrix} T_x & T_y & T_z \\ B_x & B_y & B_z \end{bmatrix} = \frac{1}{s_1t_2 - s_2t_1} \begin{bmatrix} t_2 & -t_1 \\ -s_2 & s_1 \end{bmatrix} \begin{bmatrix} (\mathbf{Q}_1)_x & (\mathbf{Q}_1)_y & (\mathbf{Q}_1)_z \\ (\mathbf{Q}_2)_x & (\mathbf{Q}_2)_y & (\mathbf{Q}_2)_z \end{bmatrix}.$$

- This is for the triangle. What wrong with that?

Lengyel's Method

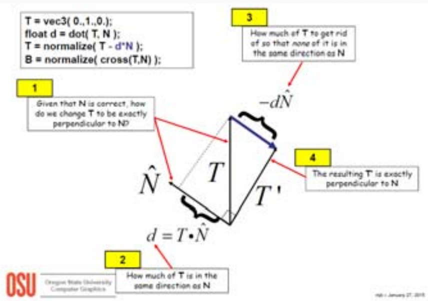
$$\begin{bmatrix} T_x & T_y & T_z \\ B_x & B_y & B_z \end{bmatrix} = \frac{1}{s_1t_2 - s_2t_1} \begin{bmatrix} t_2 & -t_1 \\ -s_2 & s_1 \end{bmatrix} \begin{bmatrix} (\mathbf{Q}_1)_x & (\mathbf{Q}_1)_y & (\mathbf{Q}_1)_z \\ (\mathbf{Q}_2)_x & (\mathbf{Q}_2)_y & (\mathbf{Q}_2)_z \end{bmatrix}.$$

- This is for the triangle. What wrong with that?
  1. Not normalized vectors
  2. Same across the triangle

Another way to establish Tangent

- Pick a general rule, e.g. 'tangent = up'
  - 1) use Gram-Schmidt to correctly orthogonalize it WRT the Normal
    - Next slide
  - 2) use two cross-products to correctly orthogonalize it WRT the Normal
    - T= vec3(0.0, 1.0, 0.0);
    - B= normalize(cross(T,N));
    - T= normalize(cross(N,B));

Gram-Schmidt Orthogonalization



Converting Between Coordinate Systems

Converting from Eye Coordinates to Surface Local Coordinates:

$$\begin{Bmatrix} s \\ t \\ h \end{Bmatrix} = \begin{bmatrix} B_x & B_y & B_z \\ T_x & T_y & T_z \\ N_x & N_y & N_z \end{bmatrix} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}$$

(The "Orange Book" uses this to convert the light vector to Surface Local Coordinates.)

Converting from Surface Local Coordinates to Eye Coordinates:

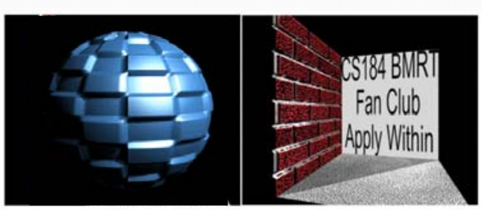
$$\begin{Bmatrix} x \\ y \\ z \end{Bmatrix} = \begin{bmatrix} B_x & T_x & N_x \\ B_y & T_y & N_y \\ B_z & T_z & N_z \end{bmatrix} \begin{Bmatrix} s \\ t \\ h \end{Bmatrix}$$

Displacement Mapping

- Bump mapping
  - can be at pixel level
  - has no geometry/shape change
- Displacement Mapping
  - Actually modify the surface geometry (vertices)
  - re-calculate the normals
  - Can include bump mapping

Displacement Mapping

- Bump mapped normals are inconsistent with actual geometry. No shadow.
- Displacement mapping affects the surface geometry



Mark Kilgard's GDC explanation

- http://www.slideshare.net/Mark_Kilgard/geometryshaderbasedbumpmappingsetup