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
  - $[-1,1]^3 \rightarrow [0,1]^3$: \(\text{rgb} = n \ast 0.5 + 0.5\)
- RGB decoding in fragment shaders
  - \(\text{vec3 n = texture2D(NormalMap, texcoord.st).xyz} \ast 2.0 - 1.0\)
- 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: \(n \ast l\)
  - Specular: \((n \ast h)\) shininess
- Computations done in tangent space

Maps
- Height map (Gray scale)
- Base texture (RGB)
- Normal map (normal encoded RGB)

Normal Map & Height Field

Cg Book: Normalization CubeMap
- What is it?
- Why do it?
- \((3, 1.5, 0.9) \rightarrow (0.93, 0.72, 0.63)\)
- 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 \((Z \text{ is normal direction)}\)
- What’s the normal?
  - When rendering, perturb the normal with a normal map.
  - How?

What about 2 planes?

Demo
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
  \[
  \begin{bmatrix}
  0 & 0 & 1 \\
  0 & 0 & 1 \\
  0 & -1 & 0 \\
  \end{bmatrix}
  \]
- Lights too!
  \[
  L' = L_x' \ L_y' \ L_z' = L_x - L_z \ L_y = \begin{bmatrix}
  1 & 0 & 0 \\
  0 & 1 & 0 \\
  -1 & 0 & 0 \\
  \end{bmatrix}
  \]

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

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:
    \[
    x = (M + N \cos(2 \pi t)) \cos(2 \pi s) \\
    y = (M + N \cos(2 \pi t)) \sin(2 \pi s) \\
    z = N \sin(2 \pi t)
    \]
    \[
    \frac{\partial x}{\partial s} = -2\pi (M + N \cos(2 \pi t)) \sin(2 \pi s) \\
    \frac{\partial y}{\partial s} = -2\pi N \cos(2 \pi t) \sin(2 \pi s) \\
    \frac{\partial y}{\partial t} = -2\pi N \cos(2 \pi t) \cos(2 \pi s) \\
    \frac{\partial z}{\partial s} = 0 \\
    \frac{\partial z}{\partial t} = 2\pi N \cos(2 \pi t)
    \]
Torus

\[
\begin{align*}
\frac{\partial x}{\partial s} &= -2\pi(M + N \cos(2\pi t)) \sin(2\pi s) \\
\frac{\partial y}{\partial s} &= 2\pi(M + N \cos(2\pi t)) \cos(2\pi s) \\
\frac{\partial z}{\partial s} &= 0
\end{align*}
\]

\[
\begin{align*}
\frac{\partial x}{\partial t} &= -2\pi N \sin(2\pi t) \cos(2\pi t) \\
\frac{\partial y}{\partial t} &= -2\pi N \cos(2\pi t) \sin(2\pi t) \\
\frac{\partial z}{\partial t} &= 2\pi \cos(2\pi t)
\end{align*}
\]

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

\[
N = \left\langle \cos(s) \cos(t), \sin(s) \cos(t), \sin(t) \right\rangle
\]

---

What about polygons?

- Demo

- 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 (P₀, P₁, P₂):

\[ Q - P₀ = (u - u₀)T + (v - v₀)B \]

- T = tangent vector
- B = bitangent vector
- P₀ = 1st vertex
- u₀ = s texture coordinate
- v₀ = t texture coordinate

Lengyel’s Method

set up a linear system:

\[ Q₁ = s₁T + t₁B \]
\[ Q₂ = s₂T + t₂B \]

Write in matrix form:

\[ M_{QQ₂} = M_{ST} M_{TB} \]

Multiply each side by M⁻¹\text{T}:

\[ M_{ST} = M_{ST} - M_{TB} \]

Lengyel’s Method

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

Lengyel’s Method

- Triangle: (using his notation, (s,t) = (u,v)
- Vertex attributes are defined by OpenGL:

\[ P₀ = (u₀,v₀) \]
\[ P₁ = (u₁,v₁) \]
\[ P₂ = (u₂,v₂) \]

\[ Q₁ = P₁ - P₀ \]
\[ Q₂ = P₂ - P₀ \]

So:

\[ Q₁ = s₁T + t₁B \]
\[ Q₂ = s₂T + t₂B \]

Need to solve for T and B

Lengyel’s Method

- 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:
\[
\begin{bmatrix}
  s \\
  t \\
  b
\end{bmatrix}
= \begin{bmatrix}
  B_x & B_y & B_z \\
  T_x & T_y & T_z \\
  N_x & N_y & N_z
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z
\end{bmatrix}
\]

( The 'Change Basis' case to convert the light vector to Surface Local Coordinates )

Converting from Surface Local Coordinates to Eye Coordinates:
\[
\begin{bmatrix}
  x \\
  y \\
  z
\end{bmatrix}
= \begin{bmatrix}
  B_x & T_x & N_x \\
  B_y & T_y & N_y \\
  B_z & T_z & N_z
\end{bmatrix}
\begin{bmatrix}
  s \\
  t \\
  b
\end{bmatrix}
\]

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