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.

Normal Map

- Normal vector encoded as rgb
  - $[-1,1] \rightarrow [0,1]: \text{rgb} = n \cdot 0.5 + 0.5$
- 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: $(n \cdot h)^{\text{exponent}}$
- 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?

Wall & Ceiling
Floor & Ceiling (Floor Desk) and Incororated

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

\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\mathbf{n}_0 \\
\mathbf{n}_1 \\
\mathbf{n}_2
\end{bmatrix} = \begin{bmatrix}
\mathbf{t}_0 \\
\mathbf{t}_1 \\
\mathbf{t}_2
\end{bmatrix}
\]

- Lights too!

\[
\mathbf{L}' = \begin{bmatrix}
1 & 0 & 0 \\
0 & 0 & 1 \\
0 & -1 & 0
\end{bmatrix}
\begin{bmatrix}
\mathbf{L}_0 \\
\mathbf{L}_1 \\
\mathbf{L}_2
\end{bmatrix}
\]
Tangent Space
- Tangent, Bi-tangent and Normal can form a rotation matrix too:
  $$\begin{pmatrix}
  T_x & B_x & N_x \\
  T_y & B_y & N_y \\
  T_z & B_z & N_z
  \end{pmatrix}$$
- Orthonormal matrix:
  $$B = N \times T$$
  $$N = T \times B$$
  $$T = B \times N$$

Tangent Space
- Each vertex has a Normal and a Tangent

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 $[s, t]$

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

  $$N = \left( \begin{pmatrix}
  \frac{\partial x}{\partial s} \\
  \frac{\partial y}{\partial s} \\
  \frac{\partial z}{\partial s}
  \end{pmatrix},
  \begin{pmatrix}
  \frac{\partial x}{\partial t} \\
  \frac{\partial y}{\partial t} \\
  \frac{\partial z}{\partial t}
  \end{pmatrix}\right) \times \begin{pmatrix}
  \frac{\partial x}{\partial t} \\
  \frac{\partial y}{\partial t} \\
  \frac{\partial z}{\partial t}
  \end{pmatrix}$$
  $$N = \left( \begin{pmatrix}
  \cos(s) \cos(t), \\
  \sin(s) \cos(t), \\
  \sin(t)
  \end{pmatrix} \right)$$
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 (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

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

- \( Q_1 = P_1 - P_0 \) \( (u_1,v_1) = (u_1 - u_0, v_1 - v_0) \)
- \( Q_2 = P_2 - P_0 \) \( (u_2,v_2) = (u_2 - u_0, v_2 - v_0) \)

So:
- \( Q_1 = sT + tB \)
- \( Q_2 = sT + tB \)

Need to solve for \( T \) and \( B \)

Lengyel’s Method

Write in matrix form:
\[
\begin{pmatrix} Q_1 & Q_2 \end{pmatrix} = \begin{pmatrix} sT + tB \end{pmatrix}
\]

Multiply each side by \( M^{-1}_{ST} \):
\[
\begin{pmatrix} sT + tB \end{pmatrix} = M^{-1}_{ST} \begin{pmatrix} Q_1 & Q_2 \end{pmatrix}
\]

\[
\begin{pmatrix} s \ 1 \ t \ 1 \ T \ 1 \ B \ 1 \ Q_1 \ 1 \ Q_2 \ 1 \ \end{pmatrix} = M^{-1}_{ST} \begin{pmatrix} Q_1 \ 1 \ Q_2 \ 1 \ \end{pmatrix}
\]

This is for the triangle. What wrong with that?

- This is the triangle. What wrong with that?

1. Not normalized vectors
2. Same across the triangle

What about GluSphere?

- Your next assignment