Normal-Mapping

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

Mike Bailey
mjb@cs.oregonstate.edu

Computer Graphics
The Next Step in Bump-Mapping

What do you do if:

You don’t have height field data?

And, don’t have height field equations?

And, don’t have a regular shape to play the BTN trick with?

The next step is to try **Normal-Maps**
What is Normal-Mapping?

Normal-Mapping is a modeling technique where, in addition to you specifying the color texture, you also create a texture image that contains all of the normal vectors on the object.
What is Normal Mapping?

The three components of the normal vector \((nx, ny, nz)\) are mapped into the three color components \((red, green, blue)\) of the texture:

\[
\begin{align*}
\{nx, ny, nz\} & \text{ in the range } -1. \rightarrow 1. \text{ are placed into the texture's } \\
\{red, green, blue\} & \text{ in the range } 0. \rightarrow 1.
\end{align*}
\]

To convert the normal to a color:

\[
\begin{align*}
\{red, green, blue\} & = \frac{\{nx, ny, nz\} + \{1, 1, 1\}}{2}.
\end{align*}
\]

To convert the color back to a normal:

\[
\begin{align*}
\{nx, ny, nz\} & = 2.* \begin{pmatrix} red \\ green \\ blue \end{pmatrix} - \{1, 1, 1\}.
\end{align*}
\]
This Gets Us More Realistic Lighting Behavior, But Still Maintains the Advantages of Bump-Mapping

Ordinary Texture

Normal-Mapping
This Gets Us More Realistic Lighting Behavior, But Still Maintains the Advantages of Bump-Mapping
#version 330 compatibility

out vec3 EC_SurfacePosition;
out vec3 EC_EyePosition;
out vec3 EC_SurfaceNormal;
out vec3 EC_LightPosition;
out vec2 vST;

void main()
{
    EC_SurfacePosition = (gl_ModelViewMatrix * gl_Vertex).xyz;
    EC_EyePosition = vec3( 0., 0., 0. );
    EC_SurfaceNormal = normalize( gl_NormalMatrix * gl_Normal );
    EC_LightPosition = vec3( 0., 10., 0. );

    vST = gl_MultiTexCoord0.st;

    gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
}
#version 330 compatibility
uniform float uKa;
uniform float uKd;
uniform float uKs;
uniform float uShininess;
uniform float uFreq;
uniform sampler2D Color_Map;
uniform sampler2D Normal_Map;
in vec3 EC_SurfacePosition;
in vec3 EC_EyePosition;
in vec3 EC_SurfaceNormal;
in vec3 EC_LightPosition;
in vec2 vST;

void main( )
{
    vec3 P = EC_SurfacePosition;
    vec3 E = normalize( EC_EyePosition - EC_SurfacePosition );
    vec3 N = normalize( gl_NormalMatrix * (2.*texture( Normal_Map, uFreq*vST ).xyz - vec3(1.,1.,1.) ) );
    vec3 L = EC_LightPosition - P;
    vec3 Ambient_Color = uKa * texture( Color_Map, uFreq * vST ).gba;
    float Light_Intensity = 1.;
    L = normalize( EC_LightPosition - P );
    float Diffuse_Intensity = dot( N, L ) * Light_Intensity;
    vec3 Diffuse_Color = uKd * Diffuse_Intensity * texture( Color_Map, uFreq * vST ).rgb;
    float Specular_Intensity = pow( max( dot( reflect( -L, N ), E ), 0. ), uShininess ) * Light_Intensity;
    vec3 Specular_Color = uKs * Specular_Intensity * vec3( 1., 1., 1. );
    gl_FragColor = vec4(Ambient_Color + Diffuse_Color + Specular_Color, 1. );
}