Bump Mapping

What is Bump-Mapping?

Bump-mapping is the process of creating the illusion of 3D depth by using a manipulated surface normal in the lighting, rather than actually creating the extra surface detail.

Displacement-mapped

Bump-mapped

This is a good optimization! Displacement-mapping requires a lot of triangles, bump-mapping doesn't.

Definition of Height Fields -- Think of the Pin Box!

The Most Straightforward Type of Bump-Mapping is Height Fields
#version 330 compatibility

void main( )
{
    vST = gl_MultiTexCoord0.st;
    vMCposition = gl_Vertex .xyz;
    vECposition = ( gl_ModelViewMatrix * gl_Vertex ).xyz;
    gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
}

uniform sampler2D uHgtUnit;
uniform float uLevel1;
uniform float uLevel2;
uniform float uTol;
uniform float uDelta;
in vec3  vMCposition;
in vec3  vECposition;
in vec2 vST;
const float DELTA = 0.001;
const vec3 BLUE = vec3( 0.1, 0.1, 0.5 );
const vec3 GREEN = vec3( 0.0, 0.8, 0.0 );
const vec3 BROWN = vec3( 0.6, 0.3, 0.1 );
const vec3 WHITE = vec3( 1.0, 1.0, 1.0 );
const float LNGMIN  = -579240./2.; // in meters, same as heights
const float LNGMAX =  579240./2.;
const float LATMIN   = -419949./2.;
const float LATMAX  =  419949./2.;

vec3 stangent = vec3( 2.*DELTA*(LNGMAX-LNGMIN), 0., uExag * ( east - west ) );
vec3 ttangent = vec3( 0., 2.*DELTA*(LATMAX-LATMIN), uExag * ( north - south ) );
vec3 normal = normalize(  cross( stangent, ttangent )  );
float LightIntensity = dot( normalize( vec3(uLightX,uLightY,uLightZ) – vMCposition ), normal );
if( LightIntensity < 0.1 )
    LightIntensity = 0.1;
if( uUseColor )
{
    float here = texture2D( uHgtUnit, vST ).r;
    vec3 color = BLUE;
    if( here > 0. )
    {
        float t = smoothstep( uLevel1-uTol, uLevel1+uTol, here );
        color = mix( GREEN, BROWN, t );
    }
    if( here > uLevel1+uTol )
    {
        float t = smoothstep( uLevel2-uTol, uLevel2+uTol, here );
        color = mix( BROWN, WHITE, t );
    }
    gl_FragColor = vec4( LightIntensity*color, 1. );
}
else
{
    gl_FragColor = vec4( LightIntensity*uColor.rgb, 1. );
}

Remember that the cross product of two vectors gives you a vector that is perpendicular to both. So, the cross product of two tangent vectors gives you a good approximation to the surface normal.
Terrain Height Bump-mapping: Coloring by Height

No Exaggeration

Terrain Height Bump-mapping: Even Zooming-in Looks Good

Terrain Height Bump-Mapping on a Globe

Several textures are being mixed onto the surface of the globe

Visualization by Nick Gebbie
The Second Most Straightforward Type of Bump-Mapping is

**Height Field Equations**

This is the coordinate system we will be using. The plane is X-Y with Z pointing up.

\[
z = A \cos(2\pi B r + C) e^{-Dr}
\]

Radial ripple height equation with decay

**Normal = \text{xtangent} \times \text{ytangent}**

If we can get the two tangent vectors, then their cross product will give us the surface normal.

\[
x_{\text{tangent}} = \text{vec3}(1, 0, \frac{\partial z}{\partial x})
\]

\[
y_{\text{tangent}} = \text{vec3}(0, 1, \frac{\partial z}{\partial y})
\]

\[
\frac{\partial z}{\partial r} = -A \sin(2\pi B r + C)(2\pi B) e^{-Dr} + A \cos(2\pi B r + C)(-D) e^{-Dr}
\]

\[
2r \frac{\partial r}{\partial x} = 2x \quad 2r \frac{\partial r}{\partial y} = 2y
\]

\[
\frac{\partial x}{\partial r} = \frac{x}{r} \quad \frac{\partial y}{\partial r} = \frac{y}{r}
\]

(Note: \(x/r\) and \(y/r\) are actually the cosine and sine of the polar angle.)

You can sum the individual height field equations and get the same result as summing the height field displacements.

Combining Bump and Cube Mapping