Johns Hopkins
Engineering for Professionals

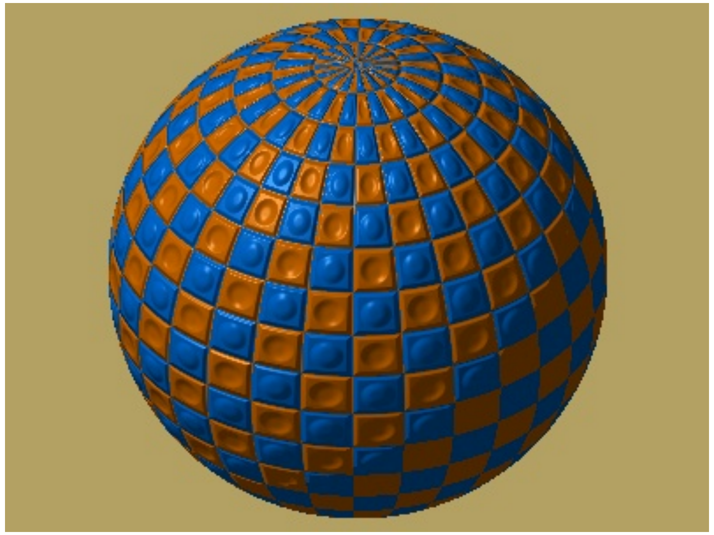
605.767 Applied Computer Graphics

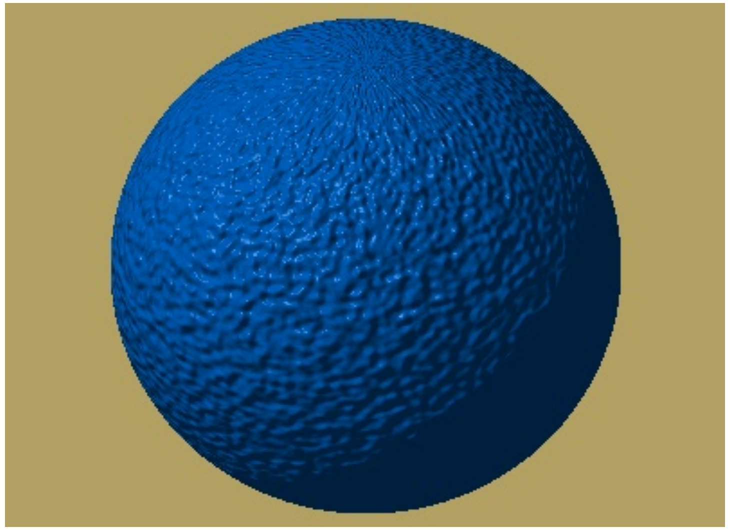
Brian Russin

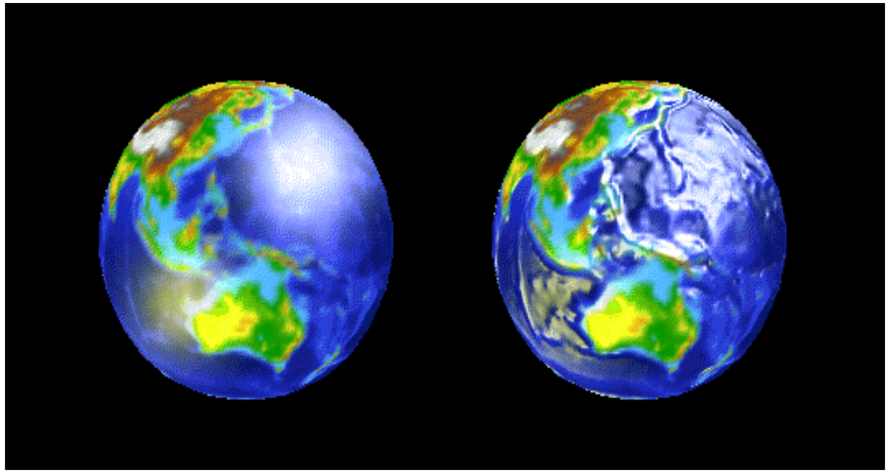
JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Module 9E
Bump Mapping

Bump Mapping







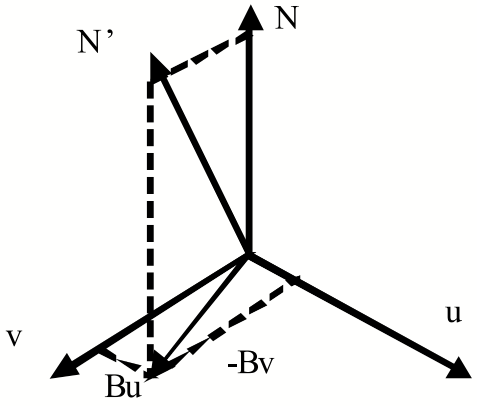
http://msdn.microsoft.com

Bump Mapping

- Texture mapping affects surface shading but the surface continues to appear geometrically smooth
  - Even if texture is an image of a rough surface it may not look right
    - Direction of light source in the image may be different
- Bump mapping enables a surface to appear bumpy or dimpled
  - Without modeling these surface impressions geometrically
    - Developed by Blinn in 1978
- Surface normal is perturbed according to information in a two dimensional bump map
  - Texture is used to modulate the surface normal rather than change a color in the illumination equation
  - Local reflection model produces shading variations on the smooth surface - **implies Phong shading**
- **Silhouette edge** does not show the expected cross section
  - Since the underlying model does not have the imperfections

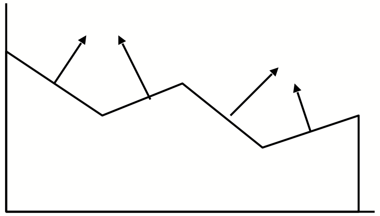
Blinn’s Method - Bump Mapping

- Blinn’s original method (1978) stored 2 signed values **Bu** and **Bv**
  - ***offset vector bump map***
  - Method uses gradients of bump map texture to define amount in u,v axes to perturb the normal
    - u,v are perpendicular to N
    - N’ = N + D where D = $B_u u - B_v v$
    - See Figure 6.33 (6.27 in 3rd Edition)
- Can define a bump map as a height field and use its derivatives at (u,v) to calculate D
  - Difference in neighboring columns to get slope in u
  - Difference in neighboring rows to get slope in v



A = N x Pv

B = N x Pu



Height Field

Normal Mapping

- Preferred method for bump mapping in today's hardware is to use **normal maps** within fragment shaders
  - Store the perturbed normal
  - Encodes x,y,z to [-1,1] ranges within an 8 bit value
    - 0 represents -1.0 normal component value (1 -> 1.0)
    - (128, 128, 255) texel value -> (0.0, 0.0, 1.0) normal
- Normals and lighting vectors must be in the same space when used within the shaders
  - Object/local, world, tangent space (relative to the surface)
- Normal maps are usually stored in tangent space
  - Storing in object space has issues if the object undergoes deformation (non-uniform scaling)
    - Light vector would need to be transformed to object space

Using Tangent Space

- Tangent space is sometimes called **surface local coordinate space**
    - Varies over the surface
        - Each point is at (0, 0, 0) – origin of tangent space
        - Normal at that point is (0, 0, 1)
    - Need to transform light and viewer directions into this space
    - A tangent space basis vector is passed in as a vertex attribute
        - Form 3 vectors: tangent vector T, normal vector N, and **bitangent** vector B
        - B = N X T
        - Often referred to as **binormal** vector
    - These form the basis matrix of the tangent space
        - Transforms light directions from world to tangent space

$$L' = \begin{bmatrix} T_x & T_y & T_z & 0 \\ B_x & B_y & B_z & 0 \\ N_x & N_y & N_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} L$$

Finding Tangents

- In tangent space x corresponds to s dimension in the bump map
  - y corresponds to t
- http://www.fabiensanglard.net/bumpMapping/index.php
- Also: **Mathematics for 3D Game Programming and Computer Graphics** by Eric Lengyel

```
generateTangent(float3 v1, float3 v2, text2 st1, text2 st2) {
  float coef = 1/ (st1.u * st2.v - st2.u * st1.v);
  float3 tangent;
  tangent.x = coef * ((v1.x * st2.v) + (v2.x * -st1.v));
  tangent.y = coef * ((v1.y * st2.v) + (v2.y * -st1.v));
  tangent.z = coef * ((v1.z * st2.v) + (v2.z * -st1.v));
}
```

  - 2 edges vectors between vertices and their texture coordinates
  - Will give tangent vector for single triangle
  - Need to average tangent vectors at a vertex
    - Similar to how normal vectors are averaged
  - Sphere, torus, other geometric shapes have well defined tangent vectors

Vertex Attributes

- Generic vertex attributes can be passed as vertex arrays

```
glVertexAttribPointer(GLuint index, GLint size, GLenum type,
                      GLboolean normalized, GLsizei stride,
                      glEnableVertexAttribArray(GLuint
index)const GLvoid* pointer)
```

  - normalized indicates whether values stored as int are converted to [-1,1] range

```
glEnableVertexAttribArray(GLuint index)
```

```
glDisableVertexAttribArray(GLuint index)
```

- To access within shader, bind the index to the shader program

```
glBindAttribLocation(GLhandle program, GLuint index,
                     const GLchar* name)
```

- See: https://www.khronos.org/opengl/wiki/GLAPI/glVertexAttribPointer

Vertex Shader for Bump-Mapping

```
// from OpenGL Shading Language 3rd Edition by Randi Rost, Bill Licea-Kane

#version 140

uniform vec3 LightPosition;
uniform mat4 MVMatrix;
uniform mat4 MVPMatrix;
uniform mat4 NormalMatrix;
in vec4 MCVertex;
in vec3 MCNormal;
in vec3 MCTangent;
in vec2 TexCoord0;
out vec3 LightDir;
out vec3 EyeDir;
out vec2 TexCoord;
void main()
{
    gl_Position = MVPMatrix * MCVertex;
    EyeDir      = vec3(MVMatrix * MCVertex);
    TexCoord    = TexCoord0.st;

    vec3 n = normalize(NormalMatrix * MCNormal);
    vec3 t = normalize(NormalMatrix * MCTangent);
    vec3 b = cross(n, t);

    vec3 v;
    v.x = dot(LightPosition, t);
    v.y = dot(LightPosition, b);
    v.z = dot(LightPosition, n);
    LightDir = normalize(v);

    v.x = dot(EyeDir, t);
    v.y = dot(EyeDir, b);
    v.z = dot(EyeDir, n);
    EyeDir = normalize(v);
}
```

Creating Normal Maps

- Tools to create normal maps from color maps are available
  - Photoshop or PaintShop Pro plugin
    - https://developer.nvidia.com/nvidia-texture-tools-exporter
    - GIMP
      - https://docs.gimp.org/en/gimp-filter-normal-map.html
- http://download.nvidia.com/developer/SDK/Individual_Samples/samples.html
  - Detailed normal maps sample has code to convert bump maps to normal maps

Parallax Mapping and Relief Mapping

- With normal mapping bumps never block each other
  - As you look along a real brick wall at some angle the mortar between bricks is occluded
- Parallax mapping was introduced in 2001 by Kaneko
  - Stores a height field texture
  - Used to shift texture coordinates to retrieve a different part of the surface
    - Amount of shift depends on the height and angle of the eye to the surface
- **Parallax Occlusion Mapping**, **Relief Mapping**, and **Steep Parallax Mapping** try to find where the view vector first intersects a height field
  - Independently developed at the same time
  - See Figure 6.39 (6.33 in 3rd Edition) – problem is to find the intersection of ray with the height field
  - See Figures 6.40 - 6.42 (6.34 - 6.36 in 3rd Edition) for more
    - Note silhouette edge problem
- https://casual-effects.com/research/McGuire2005Parallax/index.html

Displacement Mapping

- Can model bumps as true geometry in a fine mesh
  - Vertex shader can use height from texture to modify vertex location
  - Often used for terrain rendering
  - Water / wave simulations
- Displacement mapping can make collision detection more challenging
  - Base surface is unperturbed

References

- http://www.ozone3d.net/tutorials/bump_mapping.php
- http://www.3dkingdoms.com/tutorial.htm#space
- http://www.fabiensanglard.net/bumpMapping/index.php
- http://www.paulsprojects.net/tutorials/simplebump/simplebump.html
  - Fixed function pipeline with extensions