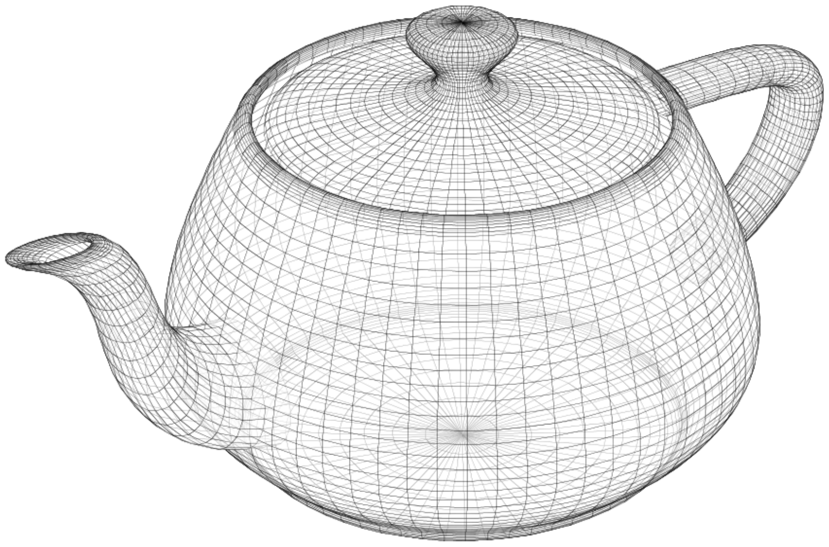


Normal Mapping

Interactive Computer Graphics

Professor Eric Shaffer

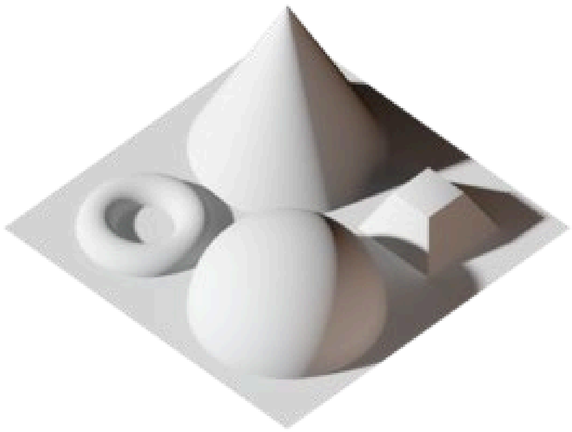
ILLINOIS

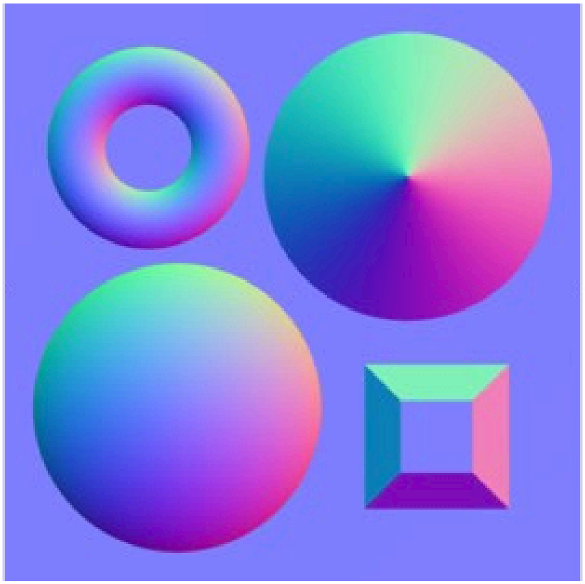
Bump Mapping and Normal Mapping

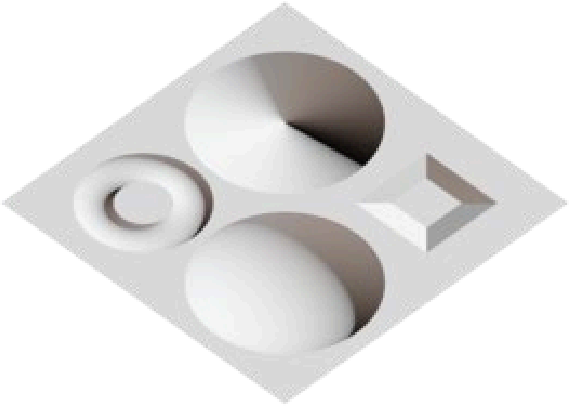
Bump mapping was proposed by Jim Blinn in 1978

Bump Mapping:
Perturbing mesh normals to create the appearance of geometric detail
Implemented using grayscale image similar to a height map
Uses less information than normal mapping

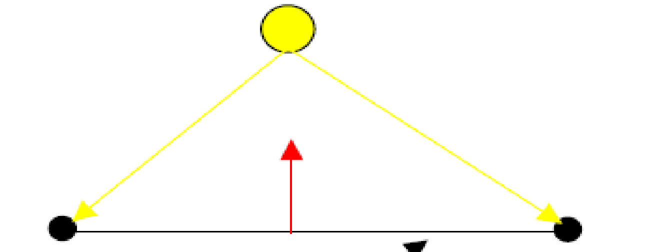
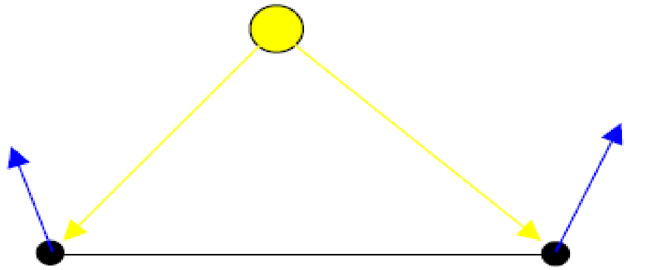
Normal Mapping:
Replaces mesh normal
Implemented by encoding normal as an RGB value

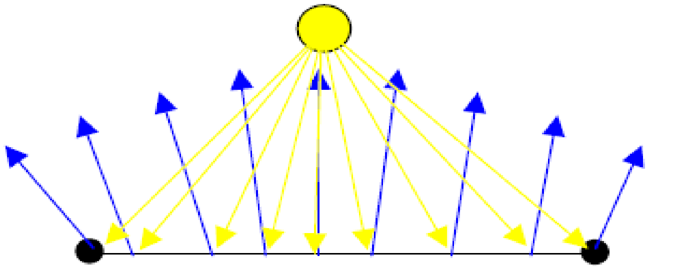
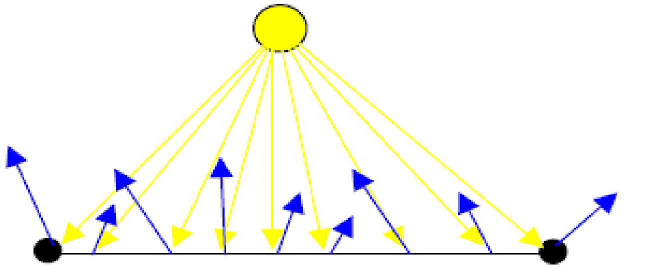




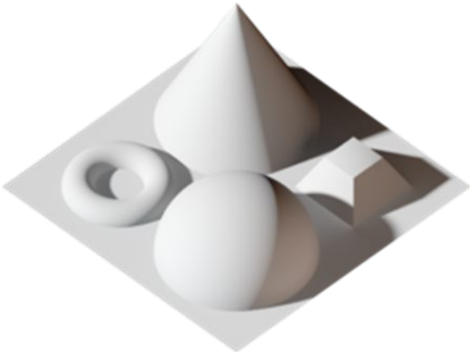


Shading

Flat shading	Goraud shading
 triangle	
Only the first normal of the triangle is used to compute lighting in the entire triangle.	The light intensity is computed at each vertex and interpolated across the surface.

Phong shading	Bump mapping
	
Normals are interpolated across the surface, and the light is computed at each fragment.	Normals are stored in a bumpmap texture, and used instead of Phong normals.

Normal Map

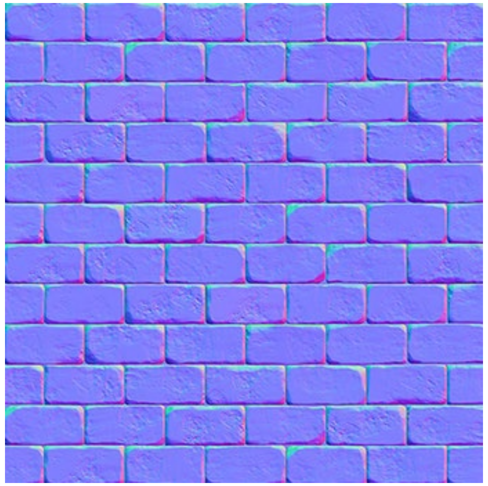


Normal vector encoded as rgb

- $[-1,1]^3 \rightarrow [0,1]^3$: rgb = n*0.5 + 0.5

RGB decoding in fragment shaders

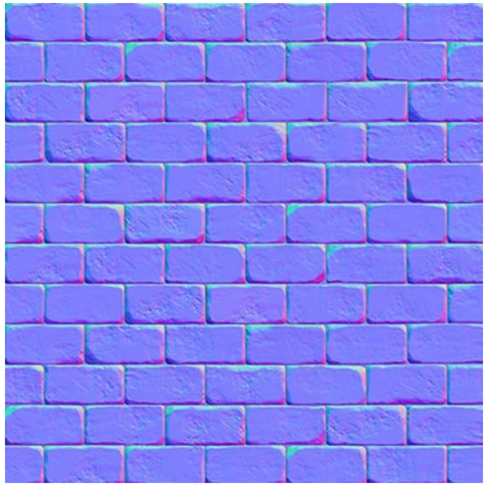
- vec3 n = texture2D(NormalMap, texcoord.st).xyz * 2.0 – 1.0

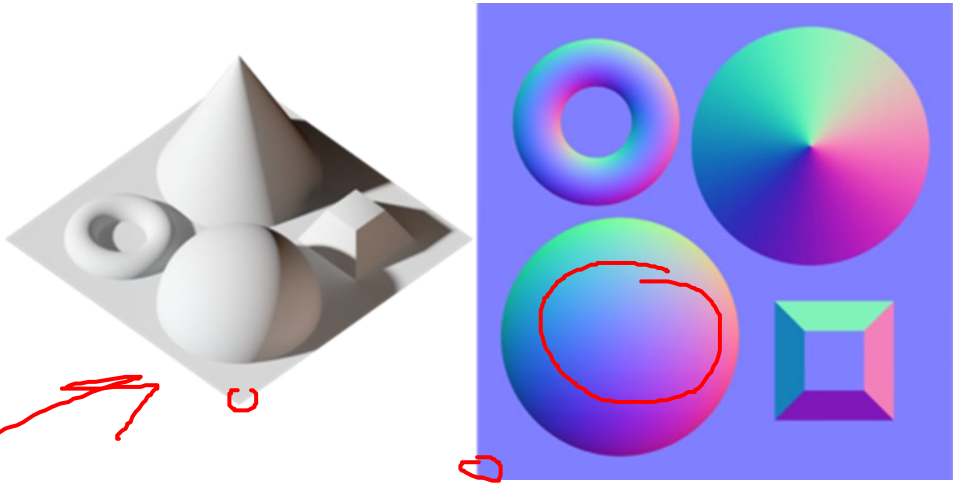


ILLINOIS

Normal Map

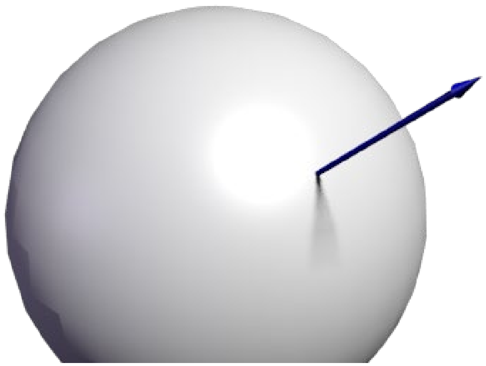
- Normal maps typically map direction out of image to +z
  - Hence RGB color for the straight up normal is (0.5, 0.5, 1.0).
  - This is why normal maps are mostly a light blue color
- Normals are then used for shading computation
  - Diffuse: n•l
  - Specular: $(n \bullet h)^{shininess}$
  - Computations done in tangent space at each fragment

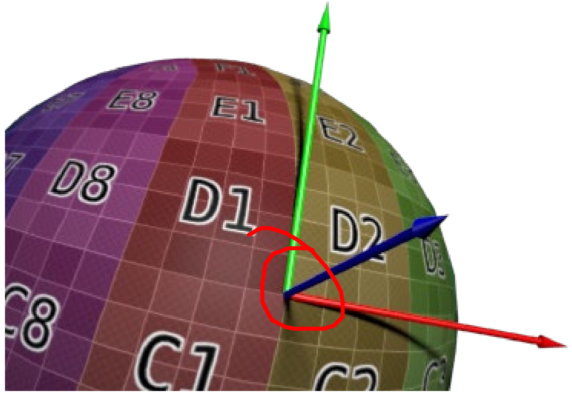


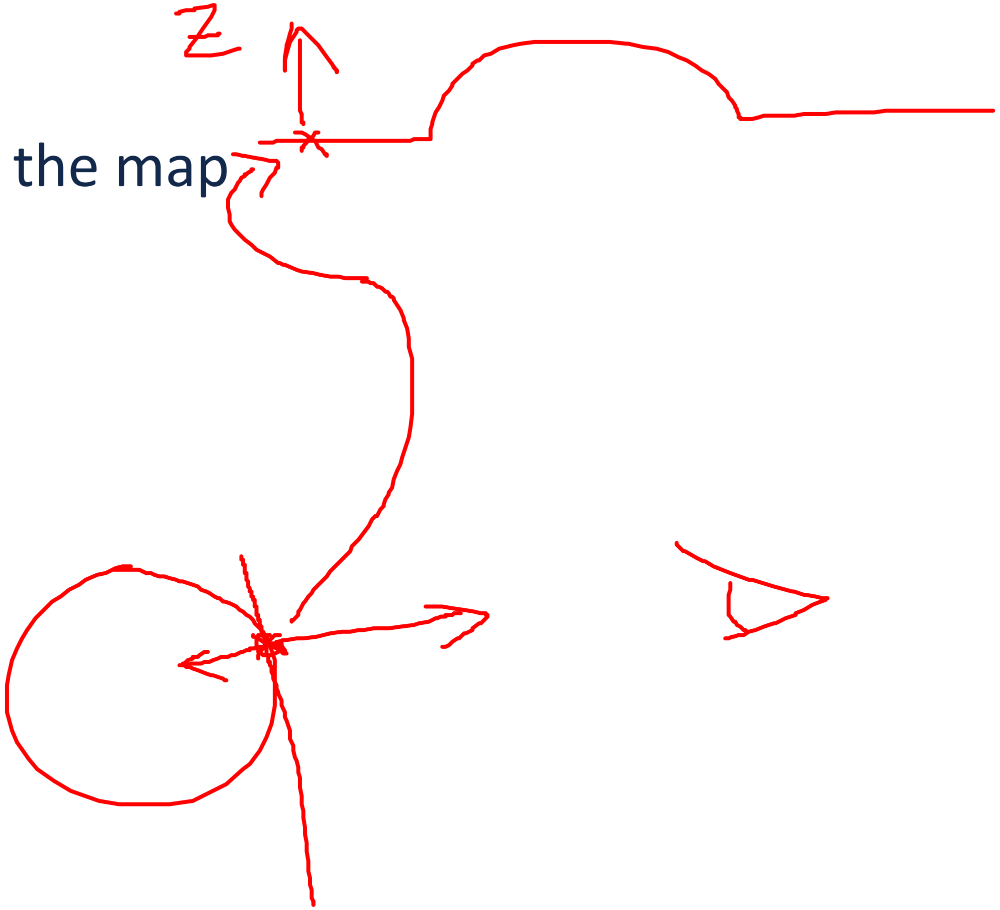


Tangent Space

- Why do we need a tangent space?
- Suppose we just set the per-fragment to a value from the map
  - Imagine the actual surface normal is (0,0,-1)
  - And the map normal is (0,0,1)
  - …we'd invert the normal and get a black spot when shading

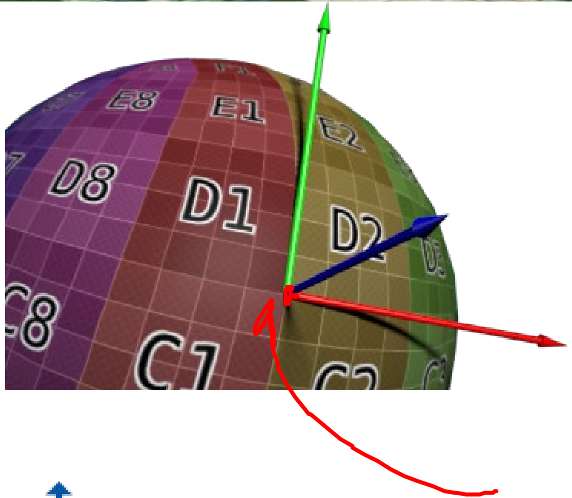




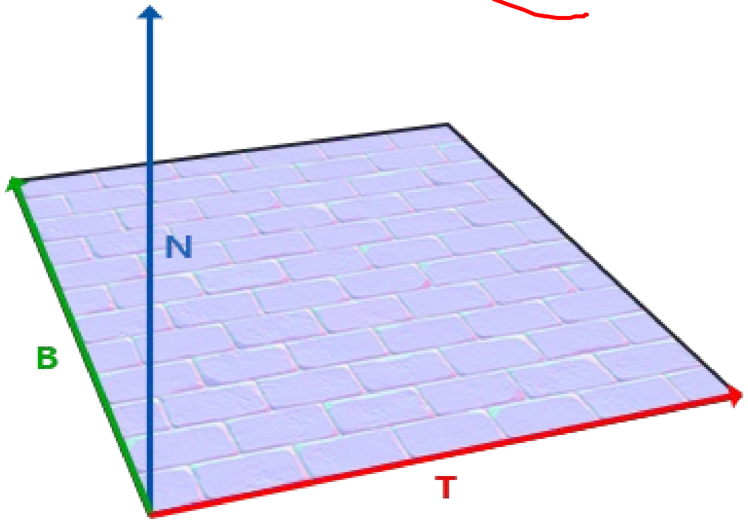


Tangent Space

- In order to build this Tangent Space,
  we need to define an orthonormal (per vertex) basis



- Tangent space is composed of 3 orthogonal vectors (T, B, N)
  - Tangent (T)
  - Bitangent (B)
  - Normal (N)
- Calculate a tangent space for every vertex
  - Can then build matrix to transform vectors
  - From view space coordinates to tangent space coordinates

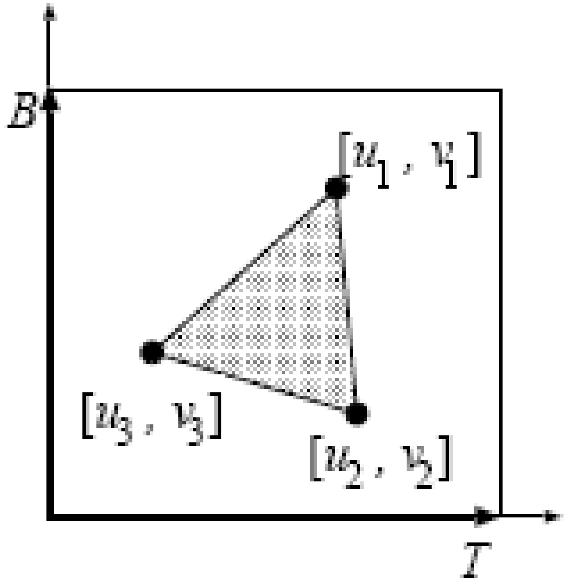


ILLINOIS

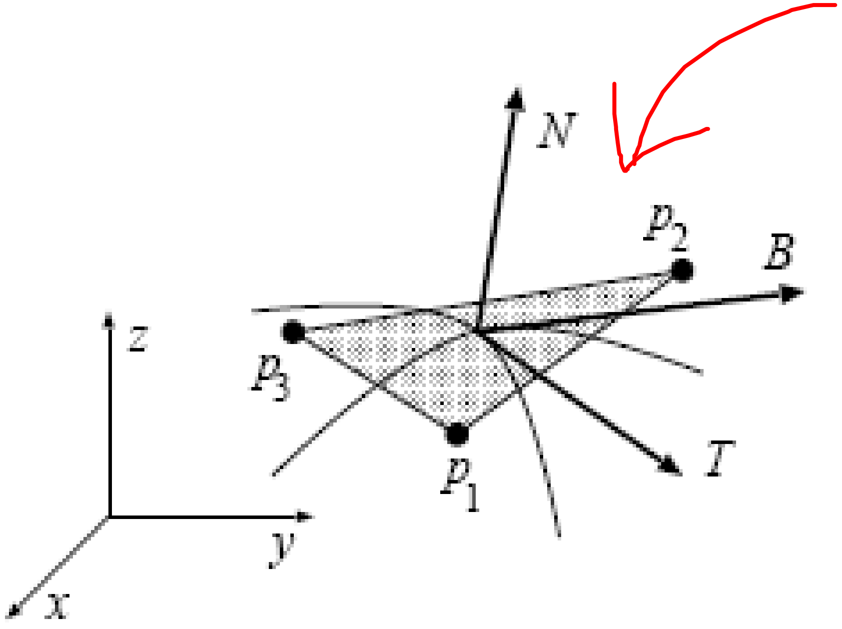
Tangent Space

- Suppose we have a vertex p_i in view coordinates
  - Texture coordinates are (u_i, v_i) are in a space tangent to p_i
  - We can use them
- The vertices p1, p2 and p3, defining the triangle :

$$p_1 = u_1.T + v_1.B$$
$$p_2 = u_2.T + v_2.B$$
$$p_3 = u_3.T + v_3.B$$



texture space



local modeling space

Tangent Space

- $p_2 - p_1 = (u_2 - u_1)T + (v_2 - v_1)B$
  $p_3 - p_1 = (u_3 - u_1)T + (v_3 - v_1)B$

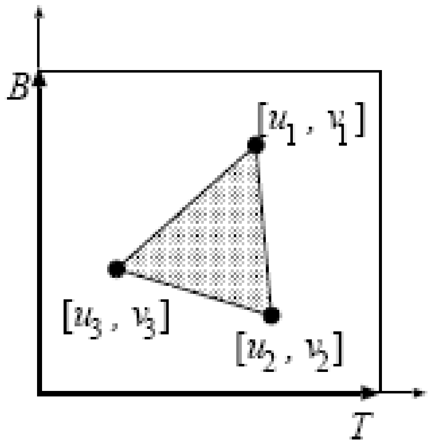
6 eqns, 6 unknowns
Why are there 6 equations?
What are the 6 unknowns

- $(v_3 - v_1)(p_2 - p_1) = (v_3 - v_1)(u_2 - u_1)T \cancel{+ (v_3 - v_1)(v_2 - v_1)B}$
  $- (v_2 - v_1)(p_3 - p_1) \quad - (v_2 - v_1)(u_3 - u_1)T \cancel{- (v_2 - v_1)(v_3 - v_1)B}$

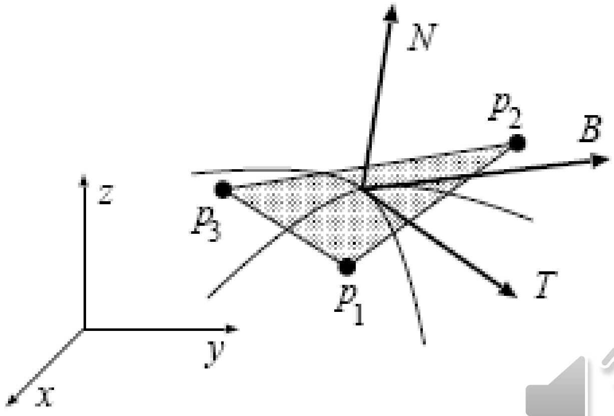
- $(u_3 - u_1)(p_2 - p_1) = \cancel{(u_3 - u_1)(u_2 - u_1)T} + (u_3 - u_1)(v_2 - v_1)B$
  $- (u_2 - u_1)(p_3 - p_1) \quad \cancel{- (u_2 - u_1)(u_3 - u_1).T} - (u_2 - u_1)(v_3 - v_1)B$

- $$T = \frac{(v_3 - v_1)(p_2 - p_1) - (v_2 - v_1)(p_3 - p_1)}{(u_2 - u_1)(v_3 - v_1) - (v_2 - v_1)(u_3 - u_1)}$$

- $$B = \frac{(u_3 - u_1)(p_2 - p_1) - (u_2 - u_1)(p_3 - p_1)}{(v_2 - v_1)(u_3 - u_1) - (u_2 - u_1)(v_3 - v_1)}$$



texture space



local modeling space

TBN Matrix Per Vertex

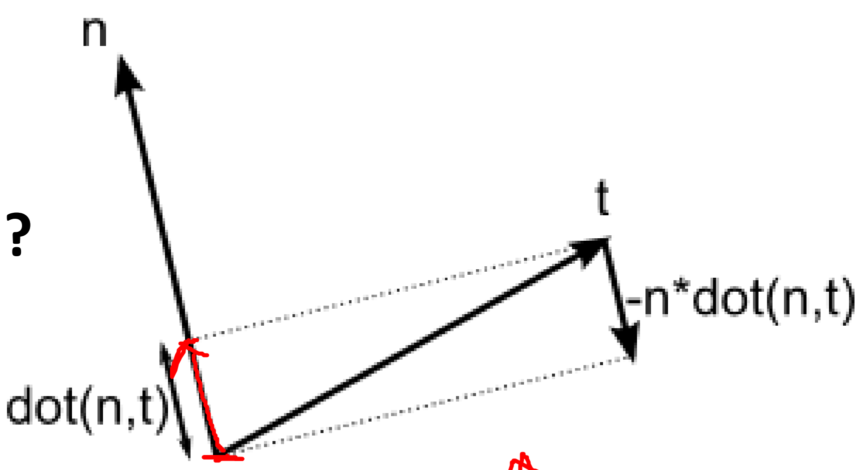
- For each triangle compute N, T, B
- For each vertex:
  - Use the averaged face normal as the vertex normal
  - Do the same for tangent and bitangent vectors
- Note that the T, B vectors might not be orthogonal to N vector
  - Use Gram-Schmidt to make sure they are orthogonal
  - Normalize them
- …you now have per vertex NTB which you can use to
  - convert shading calculations to tangent space
  - use the normal map normal instead of the geometric normal

ILLINOIS

Gram-Schmidt Orthogonalization

Assume N,T, and B are unit length

How could the equations below be simplified?



$$T = T - N\frac{(N \cdot T)}{(N \cdot N)}$$

$$B = B - N\frac{(N \cdot B)}{(N \cdot N)} - T\frac{(T \cdot B)}{(T \cdot T)}$$

ILLINOIS

Coordinate Transformation

Tangent space to view space

$$\begin{bmatrix} {}^{o}v_x \\ {}^{o}v_y \\ {}^{o}v_z \end{bmatrix} = \begin{bmatrix} T_x & B_x & N_x \\ T_y & B_y & N_y \\ T_z & B_z & N_z \end{bmatrix} \begin{bmatrix} {}^{T}v_x \\ {}^{T}v_y \\ {}^{T}v_z \end{bmatrix}$$

View space to tangent space

$$\begin{bmatrix} {}^{T}v_x \\ {}^{T}v_y \\ {}^{T}v_z \end{bmatrix} = \begin{bmatrix} T_x & B_x & N_x \\ T_y & B_y & N_y \\ T_z & B_z & N_z \end{bmatrix}^{-1} \begin{bmatrix} {}^{o}v_x \\ {}^{o}v_y \\ {}^{o}v_z \end{bmatrix} = \begin{bmatrix} T_x & T_y & T_z \\ B_x & B_y & B_z \\ N_x & N_y & N_z \end{bmatrix} \begin{bmatrix} {}^{o}v_x \\ {}^{o}v_y \\ {}^{o}v_z \end{bmatrix}$$

Shading in the Tangent Space

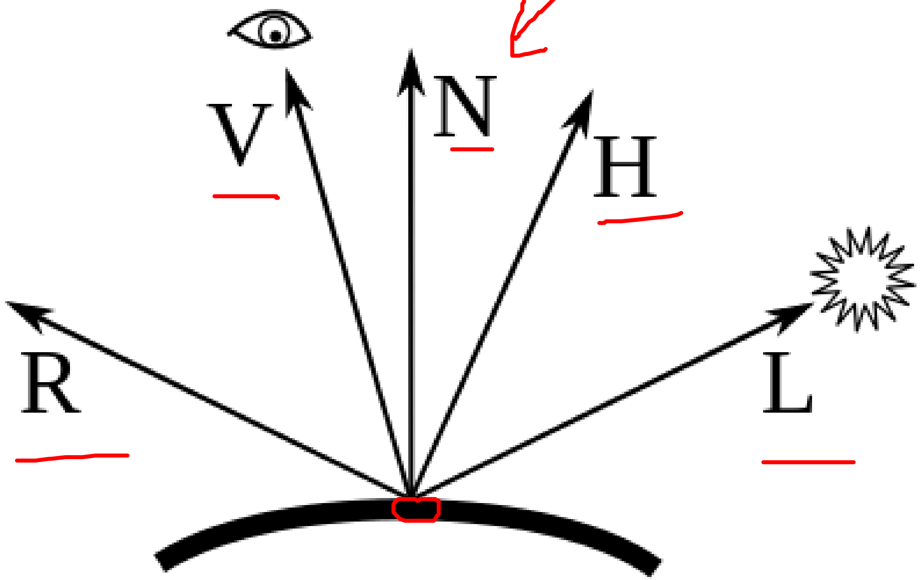
We only need to convert
- The light direction L
- the eye direction V

into the tangent space

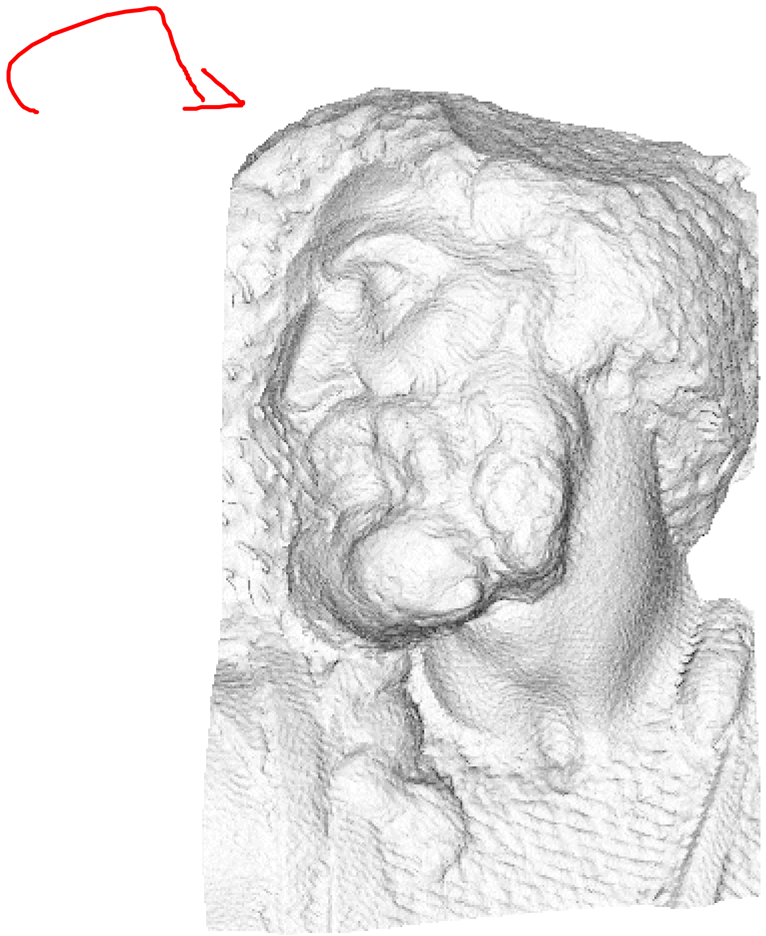
Multiply each of the above by a matrix and we can find the vectors V and L and then R or H in the tangent space

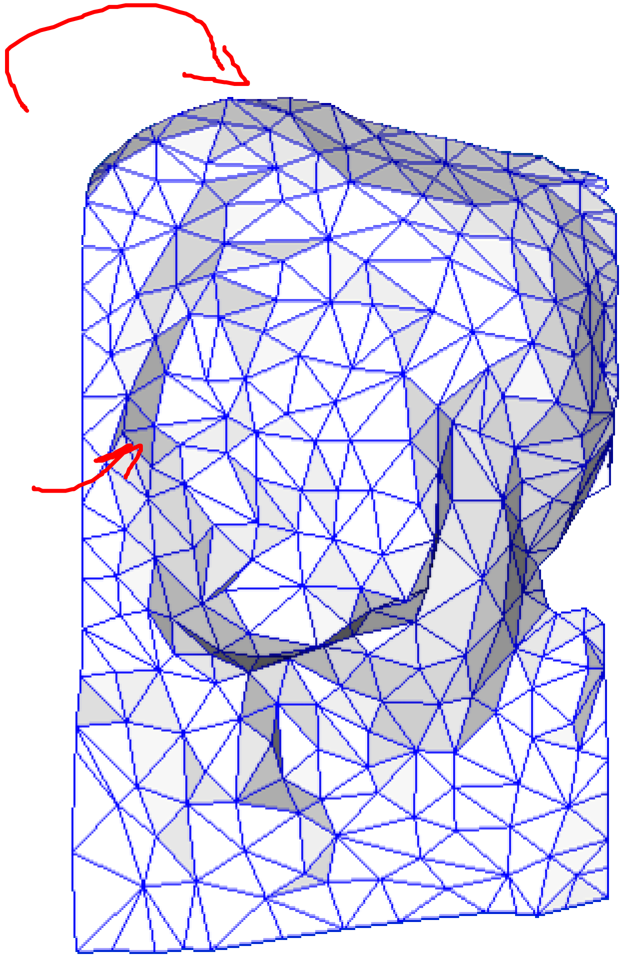
We then compute Phong reflectance model in the tangent space using the normal from the map to generate a color

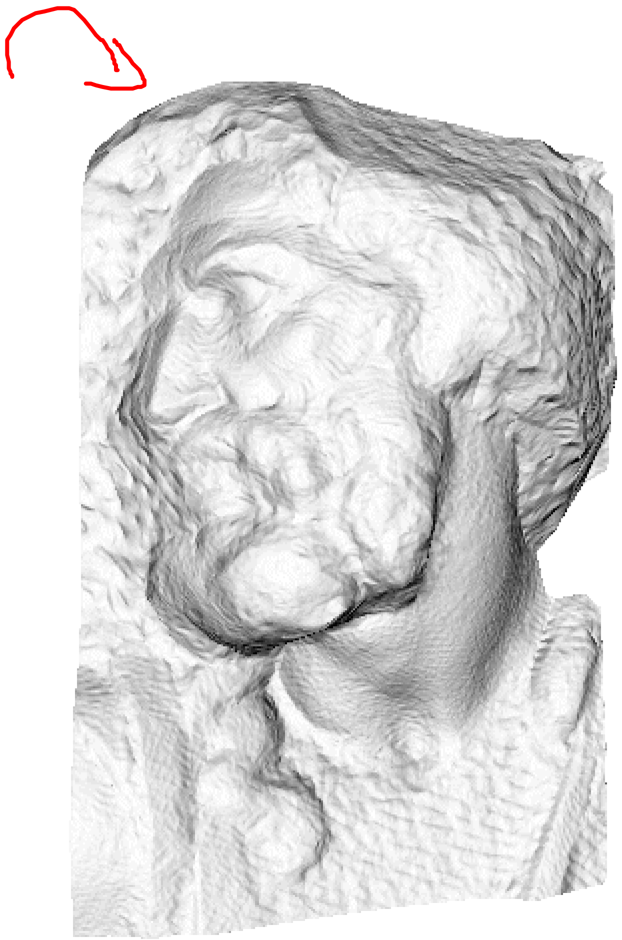
from map



You could build T and B in model coordinates and send them down to the vertex shader to be converted to view coordinate with the ModelView matrix…would be more efficient than building in view coordinates…







original mesh
4M triangles

simplified mesh
500 triangles

simplified mesh
and normal mapping
500 triangles