CS 418: Interactive Computer Graphics

Textures and Shading
Bump Mapping

Eric Shaffer

HelloTexture.html Example

- Why does it look 3D?

HelloTexture.html Example

- How could I make a “photo cube” of friends and family to show to people that aren’t really interested.
- Assume you can only use one texture.

HelloTexture.html Example

- How would you add in shading?

Texturing and Lighting in WebGL: Example Fragment Shader

```
precision mediump float;

varying vec2 vTextureCoord;
varying vec3 vTransformedNormal;
varying vec4 vPosition;

uniform bool uUseLighting;
uniform bool uUseTextures;

uniform vec3 uAmbientColor;

uniform vec3 uPointLightingLocation;
uniform vec3 uPointLightingColor;

uniform sampler2D uSampler;
```

Texturing and Lighting in WebGL: Example Fragment Shader

```
void main(void) {
    vec3 lightWeighting;
    if (!uUseLighting) {
      lightWeighting = vec3(1.0, 1.0, 1.0);
    } else {
      vec3 lightDirection = normalize(uPointLightingLocation - vPosition.xyz);

      float directionalLightWeighting =
               max(dot(normalize(vTransformedNormal), lightDirection), 0.0);
      lightWeighting = uAmbientColor +
                                uPointLightingColor * directionalLightWeighting;
    }
```

Texturing and Lighting in WebGL: Example Fragment Shader

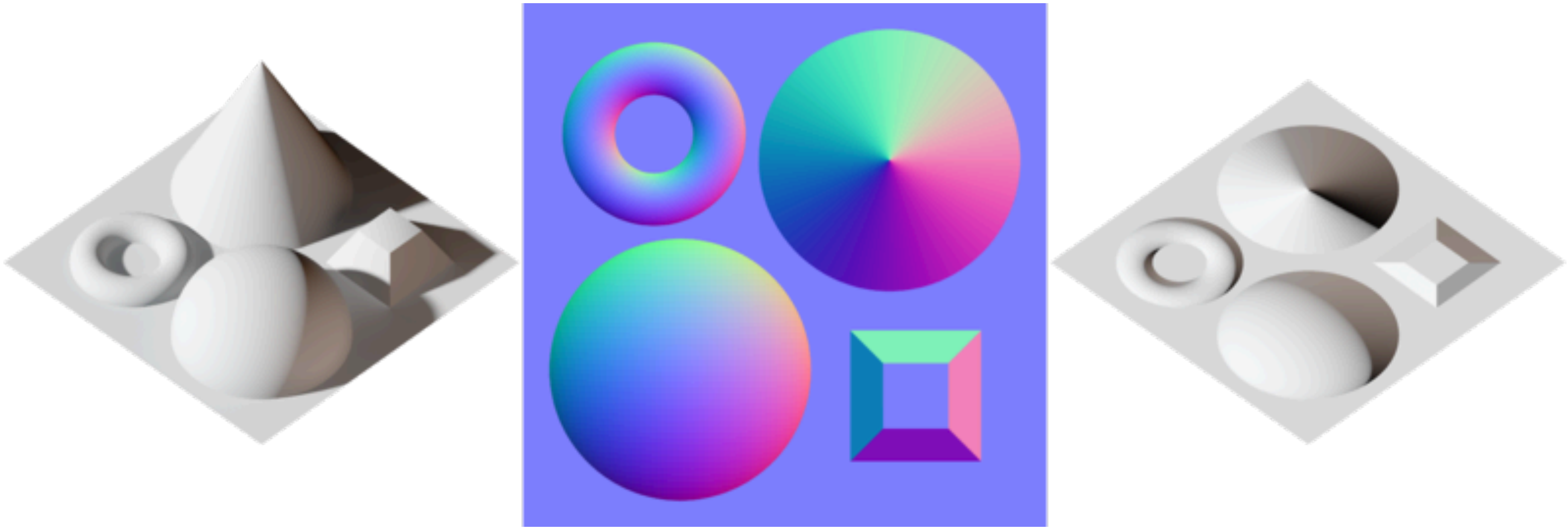
```
    vec4 fragmentColor;

    if (uUseTextures) {
      fragmentColor = texture2D(uSampler,
                                          vec2(vTextureCoord.s, vTextureCoord.t));
    } else {
      fragmentColor = vec4(1.0, 1.0, 1.0, 1.0);
    }
    gl_FragColor = vec4(fragmentColor.rgb *
lightWeighting,fragmentColor.a);
  }
```

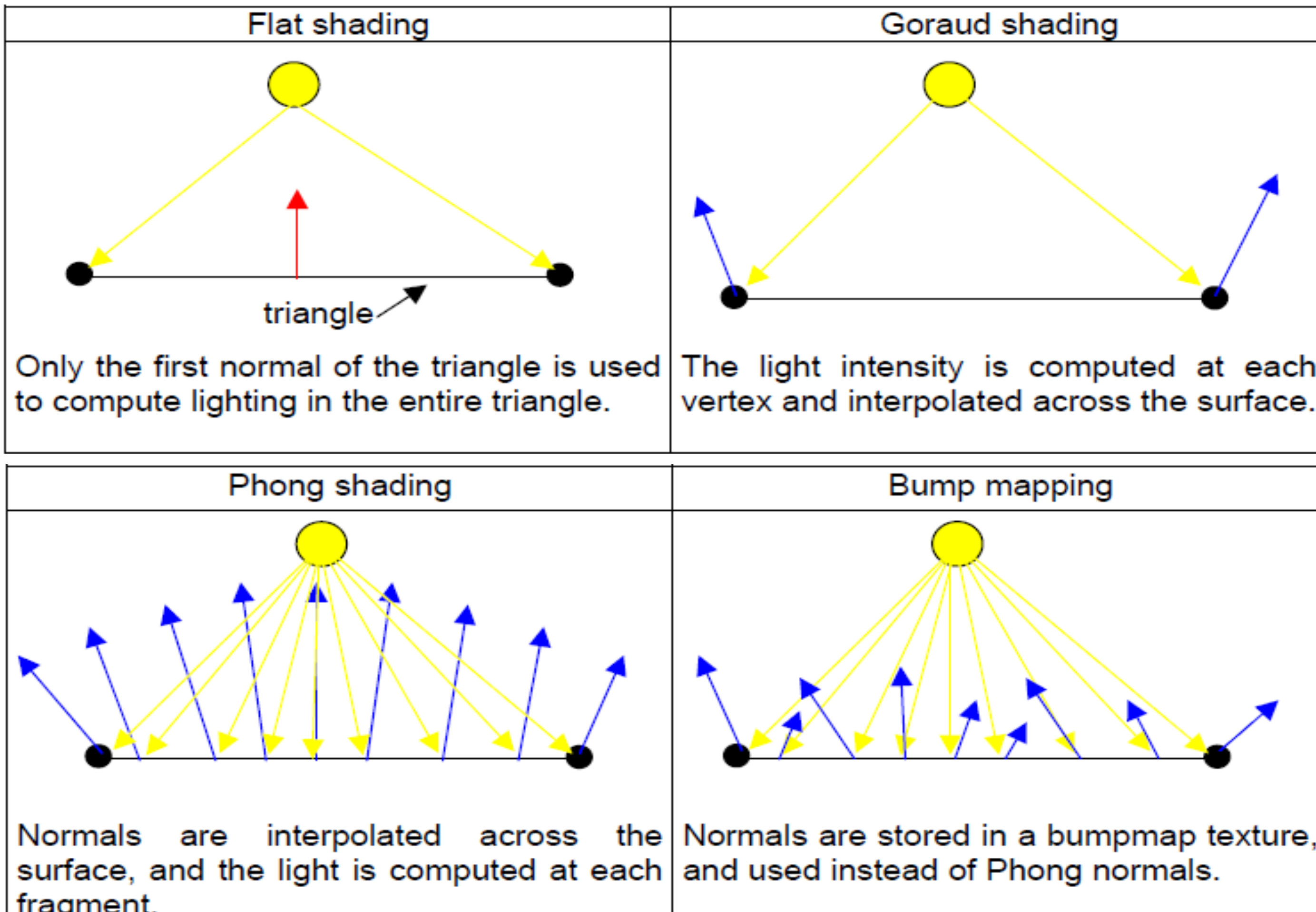
Bump Mapping and Normal Mapping

Bump Mapping:
Perturbing mesh normals to create the appearance of geometric detail

Normal Mapping:
A way of implementing bump mapping



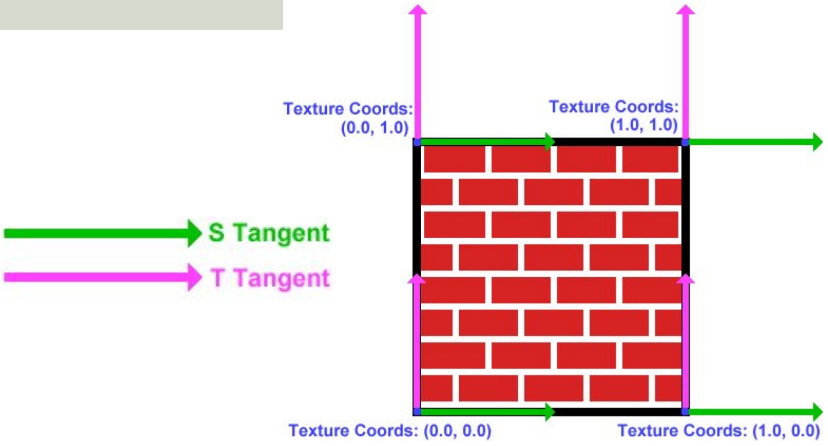
Shading


Flat shading
triangle
Only the first normal of the triangle is used to compute lighting in the entire triangle.
Goraud shading
The light intensity is computed at each vertex and interpolated across the surface.
Phong shading
Normals are interpolated across the surface, and the light is computed at each fragment.
Bump mapping
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
- We define a tangent space, a local frame at every surface point
  - The frame 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: $n \bullet l$
  - Specular: $(n \bullet h)^{shininess}$
  - Computations done in tangent space

Tangent Space

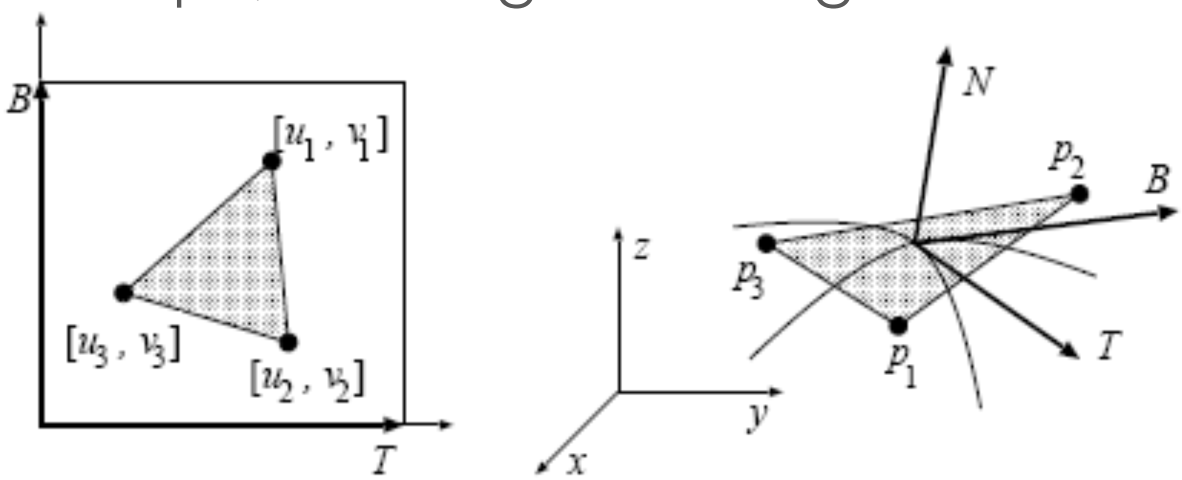


- In order to build this Tangent Space, we need to define an orthonormal (per vertex) basis, which will define our tangent space.
- Tangent space is composed of 3 orthogonal vectors (T, B, N)
  - Tangent (S Tangent)
  - Bitangent (T Tangent)
  - Normal
- One has to calculate a tangent space matrix for every vertex

Tangent Space

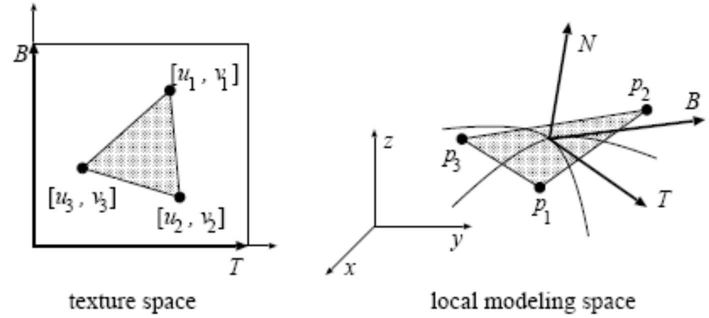
- Suppose we have a point p_i in world coordinates
  - texture coordinates are (u_i, v_i) are in a space tanget to p_i
  - We can use them
- The points p1, p2 and p3, defining the triangle :

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



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

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

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

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

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

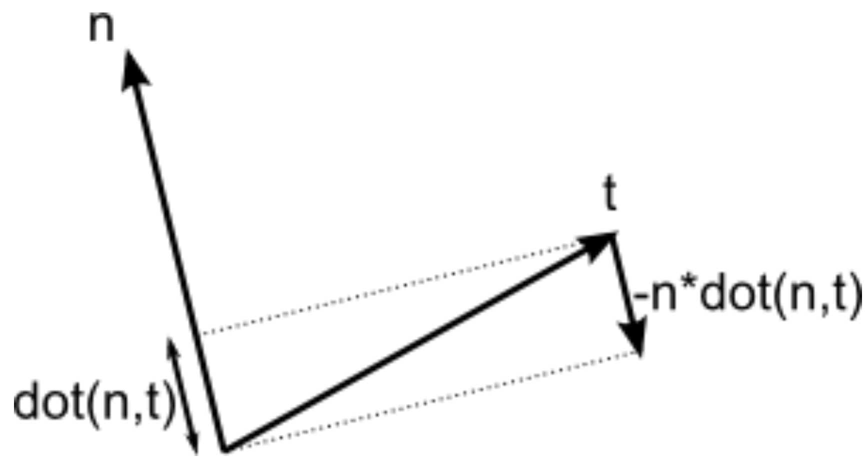
TBN Matrix Per Vertex

- 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 world shading calculations to tangent space
  - use the bump normal instead of the geometric normal

Orthogonalization

$$\mathbf{t} = \mathbf{t} - \mathbf{n}(\mathbf{n} \cdot \mathbf{t})$$

$$\mathbf{b} = \mathbf{b} - (\mathbf{n} \cdot \mathbf{b})\mathbf{n} - (\mathbf{t} \cdot \mathbf{b})\mathbf{t}$$



Coordinate Transformation

Tangent space to object space

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

Object space to tangent space

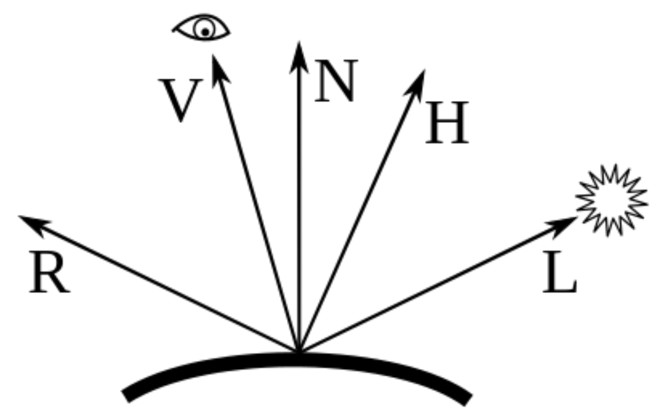
$$\begin{bmatrix} {}^{T}v_x \\ {}^{T}v_y \\ {}^{T}v_z \end{bmatrix} = \begin{bmatrix} T_x & B_x & N_x \\ T_y & B_y & N_y \\ T_z & B_z & N_z \end{bmatrix}^{-1} \begin{bmatrix} {}^{o}v_x \\ {}^{o}v_y \\ {}^{o}v_z \end{bmatrix} = \begin{bmatrix} T_x & T_y & T_z \\ B_x & B_y & B_z \\ N_x & N_y & N_z \end{bmatrix}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

Shading in the Tangent Space

We only need to convert

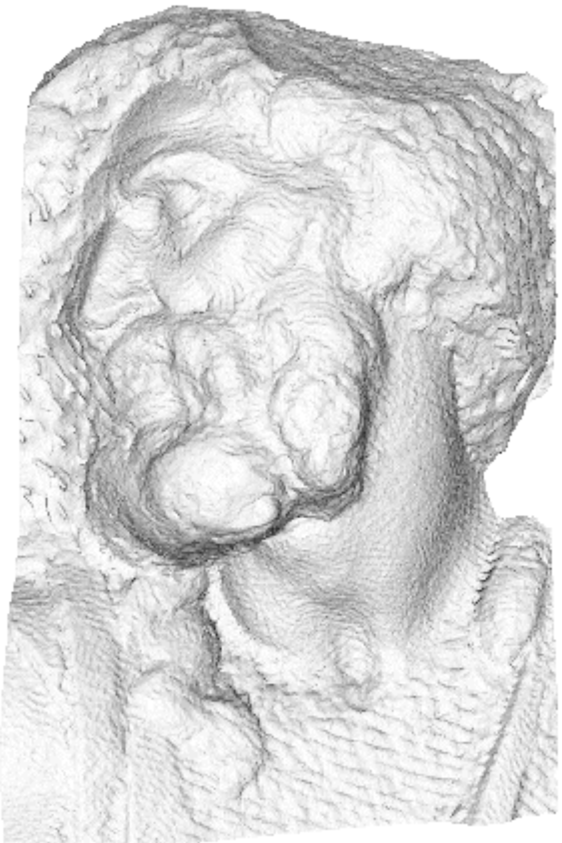
- the vertex
- the eyepoint
- the light position

into the tangent space

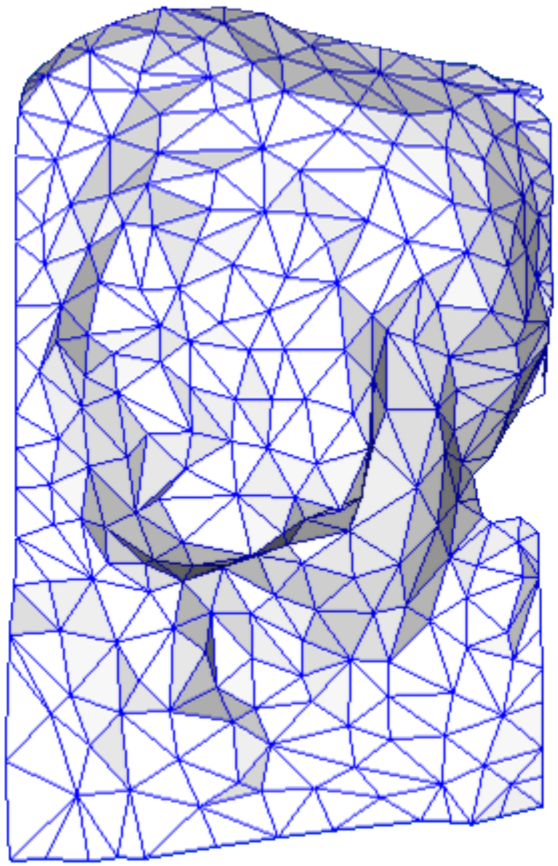


Multiply each of the above points by matrix [TBN] and we can find the vectors V and L and R or H in the tangent space

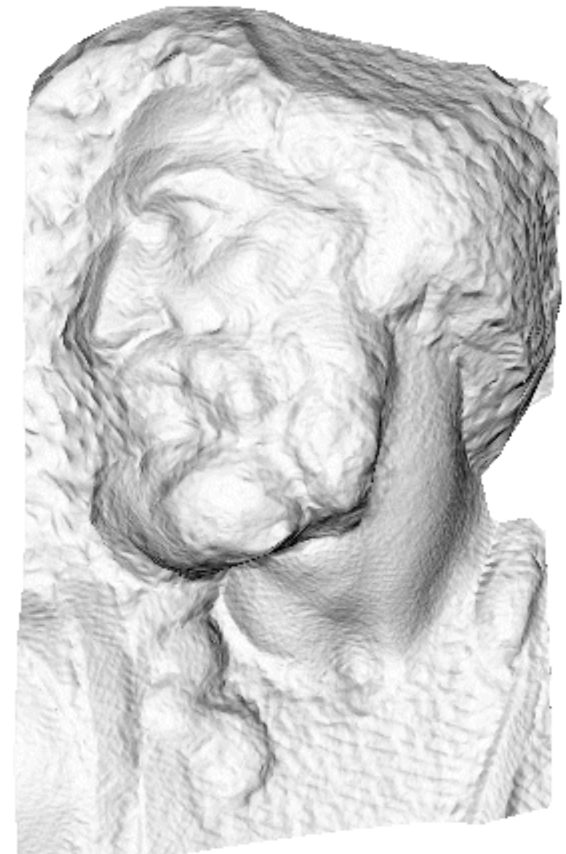
We then compute Phong reflectance model in the tangent space to generate a color



original mesh
4M triangles



simplified mesh
500 triangles



simplified mesh
and normal mapping
500 triangles