Assignment 3: Pipeline and Shading

Mengzhu Wang

January 27, 2024

1 Interpolation Across Triangles

We want to specify values at vertices and obtain smoothly varying values across triangles. We can interpolate texture coordinates, colors, normal vectors, ... using ***barycentric coordiantes***.

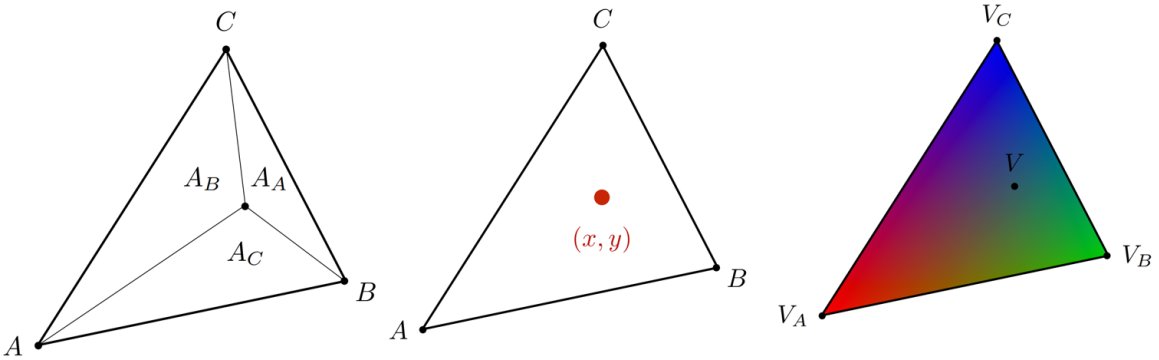
A coordinate system for triangles can be represented by $(x, y) = \alpha A + \beta B + \gamma C, \alpha + \beta + \gamma = 1$. The point is inside the triangle if all three coordinates are non-negative. The geometric viewpoint uses proportional areas,

$$\begin{aligned} \alpha &= \frac{A_A}{A_A + A_B + A_C} \\ \beta &= \frac{A_B}{A_A + A_B + A_C} \\ \gamma &= \frac{A_C}{A_A + A_B + A_C} \end{aligned} \tag{1}$$

The formulas of barycentric coordinates are,

$$\begin{aligned} \alpha &= \frac{-(x - x_B)(y_C - y_B) + (y - y_B)(x_C - x_B)}{-(x_A - x_B)(y_C - y_B) + (y_A - y_B)(x_C - x_B)} \\ \beta &= \frac{-(x - x_C)(y_A - y_C) + (y - y_C)(x_A - x_C)}{-(x_B - x_C)(y_A - y_C) + (y_B - y_C)(x_A - x_C)} \\ \gamma &= 1 - \alpha - \beta \end{aligned} \tag{2}$$

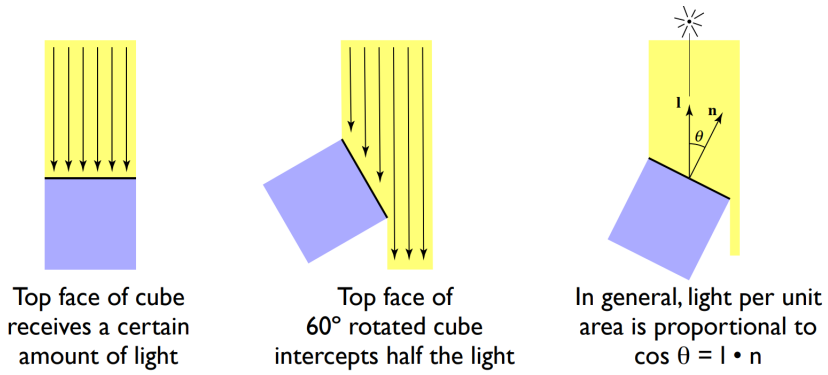
We can linearly interpolate values at vertices by $V = \alpha V_A + \beta V_B + \gamma V_C$, where V_A, V_B, V_C can be positions, texture coordinates, color, normal, depth, material attributes...

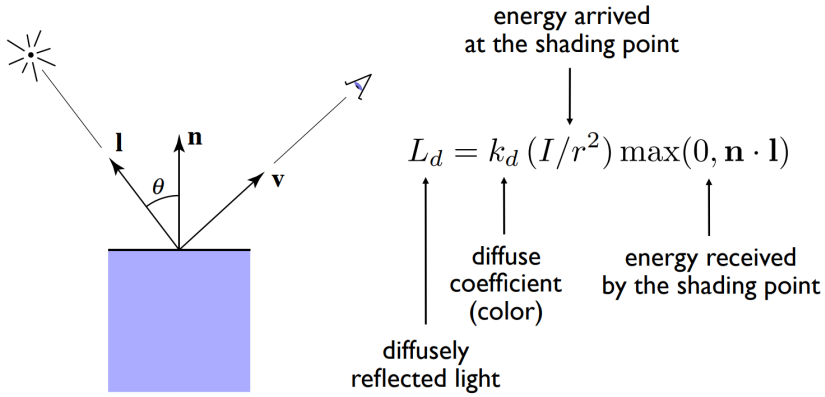


2 Blinn-Phong Reflectance Model

2.1 Lambertian (Diffuse) Term

Shading independent of view direction





2.2 Specular Term

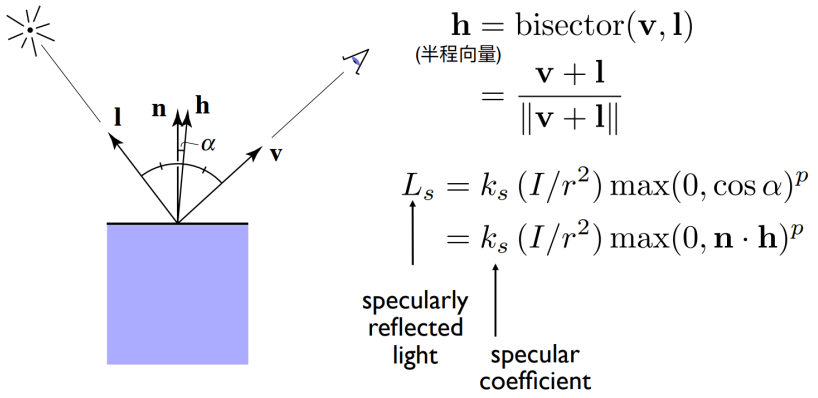
Intensity depends on view direction

- Bright near mirror reflection dirction

v close to mirror direction ⇔ half vector near normal

- Measure "near" by dot product of unit vectors

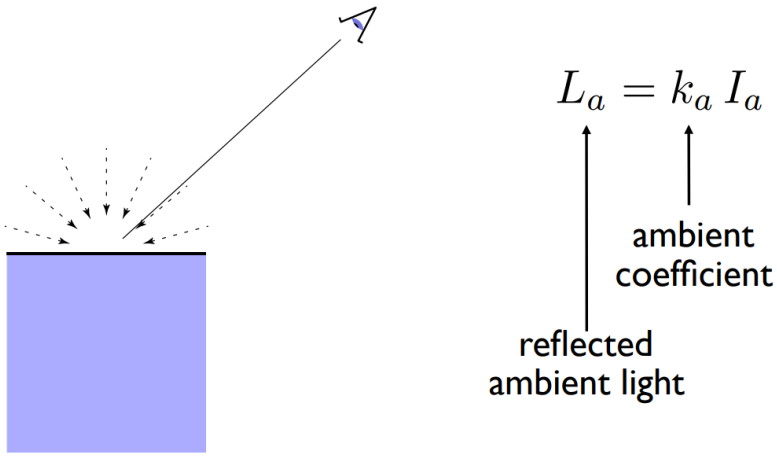
Increasing p narrows the reflection lobe



2.3 Ambient Term

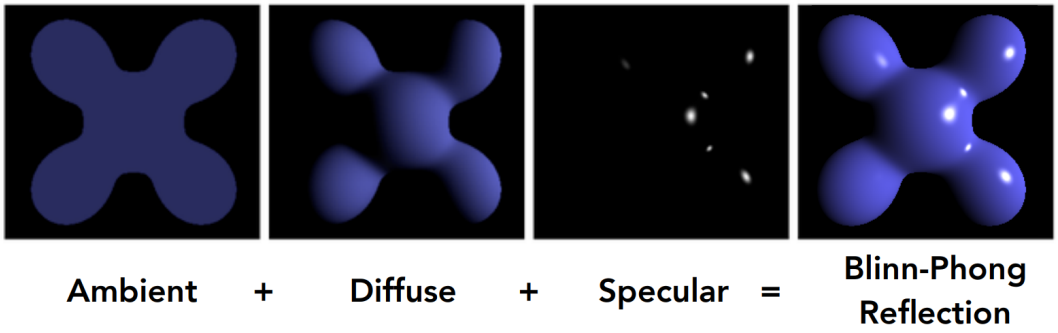
Shading that does not depend on anything

- Adding constant color to account for disregarded illumination and fill in black shadows is approximate/fake!



2.4 Summary

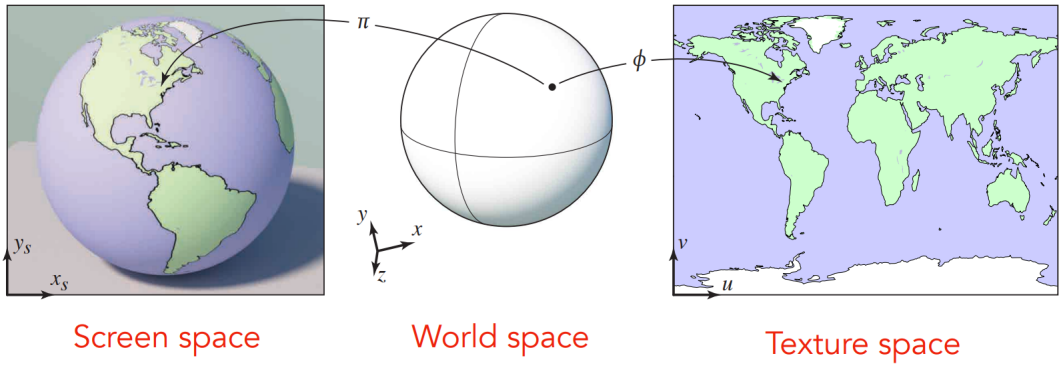
$L = L_a + L_d + L_s = k_a I_a + k_d (I/r^2) \max(0, \mathbf{n} \cdot \mathbf{l}) + k_s (I/r^2) \max(0, \mathbf{n} \cdot \mathbf{h})^p$

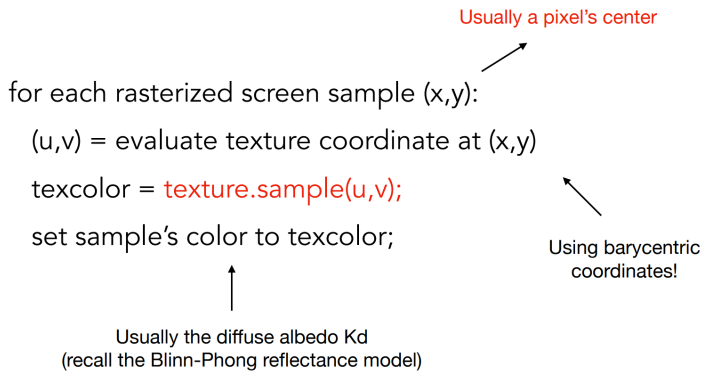


3 Texture Mapping

3.1 Simple Texture Mapping: Diffuse Color

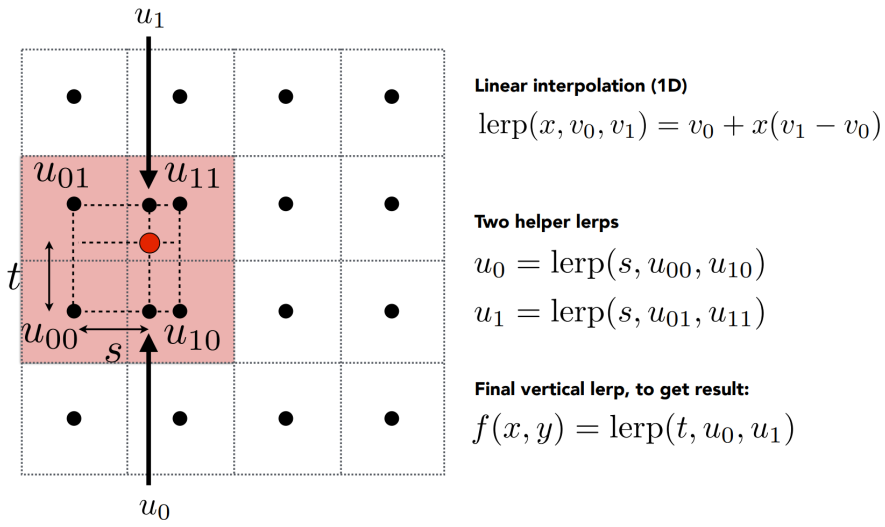
- Surfaces are 2D
- Surface lives in 3D world space
- Every 3D surface point also has a place where it goes in the 2D image (texture)





3.2 Texture Magnification

Bilinear Interpolation



4 Normal Mapping

4.1 Theory of Normal Mapping

The shading normal does not have to be the same as the geometric normal of the underlying surface. Normal mapping makes the shading normal depend on values read from a texture map. It stores the normals in a texture, with three numbers stored at every texel that are interpreted, instead of as the three components of a color, as the 3D coordinates of the normal vector.

Storing normals directly in object space, in the same coordinates system used for representing the surface geometry itself, is simplest. It will need to be transformed into world space for lighting calculations, just like a normal that came with the geometry. However, if the surface is going to deform, so that the geometric normal changes, the object-space normal map can no longer be used, since it would keep providing the same normals.

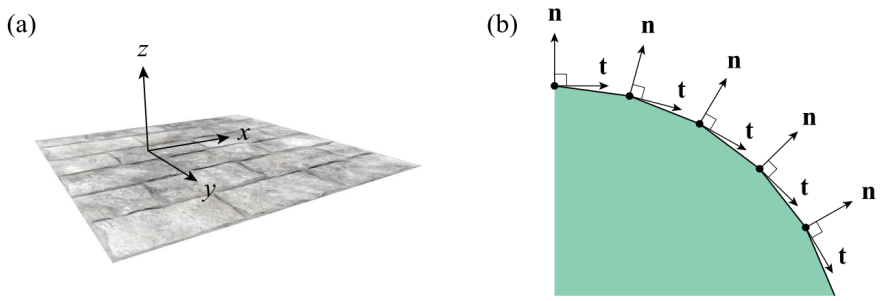
The solution is to define a coordinate system based on the ***tangent space*** of the surface: select a pair of tangent vectors and use them to define an orthonormal basis. The texture coordinate function itself provides a useful way to select a pair of tangent vectors: use the directions tangent to lines of constant u and v. These tangents are not generally orthogonal, but we can "square up" the orthonormal basis.

4.2 Tangent Space

Each vertex in a triangle mesh has a normal vector $\mathbf{n}$ and a perpendicular tangent vector $\mathbf{t}$, and both vectors form a smooth field over the entire model. A second tangent direction $\mathbf{b}$ called the bitangent vector can be calculated with a cross product. The three vectors $\mathbf{t}$, $\mathbf{b}$ and $\mathbf{n}$ form the basis of the tangent frame at each vertex, and the coordinate space in which the x, y and z axes are aligned to these directions is called tangent space. We can transform vectors from tangent space to object space using the 3×3 matrix $\mathbf{M}_{tangent}$ given by

$$\mathbf{M}_{tangent} = \begin{bmatrix} \uparrow & \uparrow & \uparrow \\ \mathbf{t} & \mathbf{b} & \mathbf{n} \\ \downarrow & \downarrow & \downarrow \end{bmatrix} \tag{3}$$

The name *TBN matrix* is often used to refer to it and its transpose/reverse.



Let $\mathbf{p}_0$, $\mathbf{p}_1$ and $\mathbf{p}_2$ be the three vertices of a triangle, wound in counterclockwise order, and let (u_i, v_i) represent the texture coordiantes associated with vertex $\mathbf{p}_i$. The values of u and v correspond to distances along the axes $\mathbf{t}$ and $\mathbf{b}$ that are aligned to the x and y directions of the texture map.

$$\mathbf{p}_i - \mathbf{p}_j = (u_i - u_j)\mathbf{t} + (v_i - v_j)\mathbf{b} \tag{4}$$

$$\begin{aligned} \mathbf{e}_1 = \mathbf{p}_1 - \mathbf{p}_0, (x_1, y_1) = (u_1 - u_0, v_1 - v_0) \\ \mathbf{e}_2 = \mathbf{p}_2 - \mathbf{p}_0, (x_2, y_2) = (u_2 - u_0, v_2 - v_0) \end{aligned} \tag{5}$$

$$\begin{aligned} \mathbf{e}_1 = x_1\mathbf{t} + y_1\mathbf{b} \\ \mathbf{e}_2 = x_2\mathbf{t} + y_2\mathbf{b} \end{aligned} \tag{6}$$

$$\begin{bmatrix} \uparrow & \uparrow \\ \mathbf{e}_1 & \mathbf{e}_2 \\ \downarrow & \downarrow \end{bmatrix} = \begin{bmatrix} \uparrow & \uparrow \\ \mathbf{t} & \mathbf{b} \\ \downarrow & \downarrow \end{bmatrix} \begin{bmatrix} x_1 & x_2 \\ y_1 & y_2 \end{bmatrix} \tag{7}$$

$$\begin{bmatrix} \uparrow & \uparrow \\ \mathbf{t} & \mathbf{b} \\ \downarrow & \downarrow \end{bmatrix} = \frac{1}{x_1 y_2 - x_2 y_1} \begin{bmatrix} \uparrow & \uparrow \\ \mathbf{e}_1 & \mathbf{e}_2 \\ \downarrow & \downarrow \end{bmatrix} \begin{bmatrix} y_2 & -x_2 \\ -y_1 & x_1 \end{bmatrix} \tag{8}$$

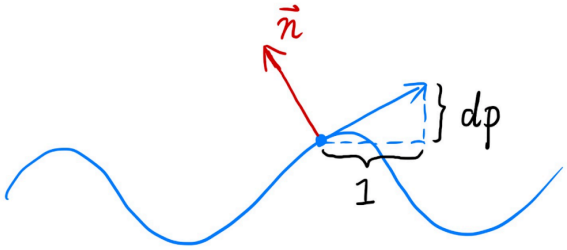
5 Bump Mapping

Adding surface detail without adding more triangles

- Perturb surface normal per pixel
- Height shift per texel defined by a texture

5.1 How to perturb the normal in flatland

- Original surface normal n(p)=(0, 1)
- Derivative at p is dp = c * [h(p+1) - h(p)]
- Perturbed normal is then n(p) = (-dp, 1).normalized()



5.2 How to perturb the normal in 3D

- Original surface normal n(p)=(0, 0, 1)
- Derivative at p are
  - dp/du = c1 * [h(u+1) - h(u)]
  - dp/dv = c2 * [h(v+1) - h(v)]
- Perturbed normal is n = (-dp/du, -dp/dv, 1).normalized()
- This is in local coordinate! Transform it to world coordinate using matrix TBN.

6 Displacement Mapping

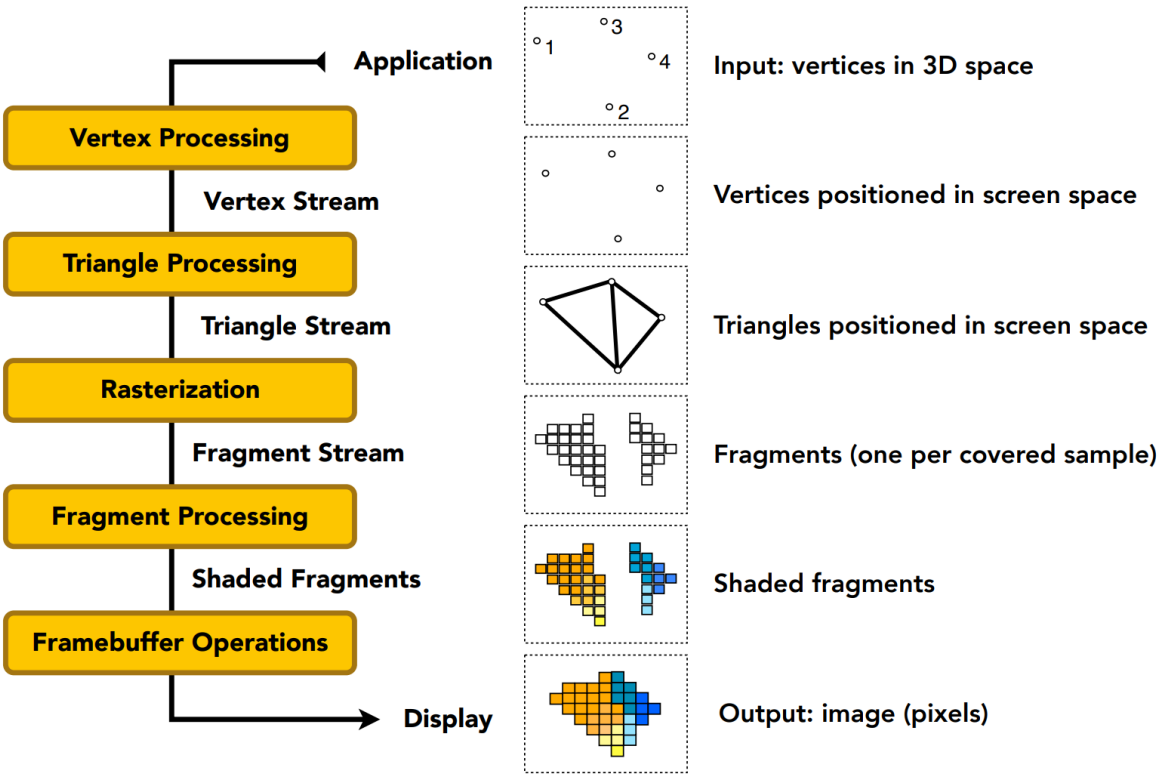
Textures can be used for more than just shading, they can be used to alter geometry. A displacement map changes the surface, moving each point along the normal of the smooth surface to a new location. The normals are roughly the same in displacement mapping and bump mapping, but the surface is different.

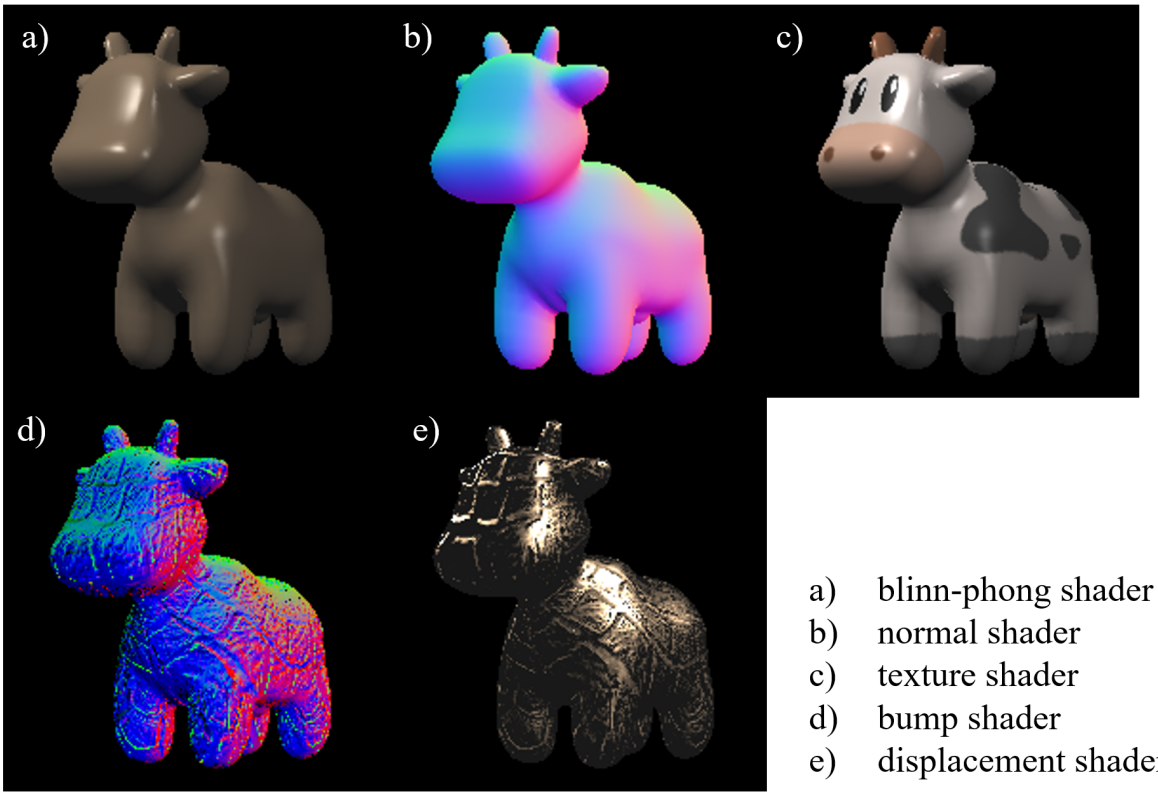
The most common way is to tessellate the smooth surface with a larger numer of small triangles and then displace the vertices of the resulting mesh using the displacement map.

- Uses the same texture as in bumping mapping
- Moves the vertices

7 Graphics Pipeline

- Model, View, Projection transforms
- Sampling triangle coverage
- Z-Buffer Visibility Tests
- Shading, Texture mapping





a) blinn-phong shader
b) normal shader
c) texture shader
d) bump shader
e) displacement shader