# CSE 167: Shading

Lecturer: Jurgen Schulze

Scribed by Owen Jow on October 16, 2018

## 1 Vertex Transformation

The complete vertex transformation is defined

$$\mathbf{p}' = \mathbf{DPC^{-1}Mp}$$

for $\mathbf{p}$ (*object coordinates*), $\mathbf{M}$ the object-to-world/model matrix (*world coordinates*), $\mathbf{C}$ the camera matrix (*camera coordinates*), $\mathbf{P}$ the projection matrix (*NDC*), and $\mathbf{D}$ the viewport matrix (*window coordinates*).

$\mathbf{C^{-1}M}$ is the model/view matrix, and describes the position and orientation of the camera. Unlike the projection matrix, it is typically changeable over the course of the program's execution.

## 2 Shading

There are multiple parts to **appearance**, of which *material properties* and *lighting* are two. **Shading** gives rise to appearance by defining how surfaces interact with light. True photorealistic rendering requires physics simulation and GI (multiple light bounces). However, in this class we focus on simplified models for classical interactive applications, and only utilize local illumination (at max one light bounce), losing out on effects which require multiple bounces of light such as color bleeding.

### 2.1 Phong Lighting Model

The Phong lighting model is a sum of *diffuse*, *specular*, and *ambient* terms. It is an attempt to make things look good without any exact basis in physics.

- **Diffuse**: "light reflected equally in all directions."

  $$c_l k_d(\mathbf{n} \cdot \mathbf{L})$$

  for $c_l$ the RGB light color, $k_d$ the RGB material diffuse color, $\mathbf{n}$ the surface normal, and $\mathbf{L}$ the lighting direction. The dot product arises from Lambert's cosine law; we get maximum reflection when the light is shining directly (perpendicularly) on the surface.

- **Specular**: reflection of light in a mirror-like fashion from the light to the viewer. With ideal specular reflection, we would only see a specular highlight at the point where the angle of incidence equals the angle of reflection. However, since most materials are not perfect mirrors, we really want to model a *lobe* around the reflection direction.

We assume that a surface is made up of randomly oriented mirrors which we call microfacets. The smoother the surface, the closer the microfacet normals are to the overall surface normal and the sharper the highlight. The rougher the surface, the more the microfacet normals differ and the blurrier the highlight.

$$c_l k_s (\mathbf{R} \cdot \mathbf{e})^p$$

for $c_l$ the RGB light color, $k_s$ the RGB material specular color, $\mathbf{R}$ the reflection direction, $\mathbf{e}$ the viewing direction, and $p$ the Phong exponent (a greater value of $p$ means a smaller, sharper highlight; note that $\mathbf{R}$ and $\mathbf{e}$ are unit vectors so $\mathbf{R} \cdot \mathbf{e}$ is a cosine and its value is in $[0, 1]$).

- **Ambient**: "a minimum brightness, what's always there, a stand-in for GI."

$$c_l k_a$$

for $c_l$ the RGB light color and $k_a$ the RGB material ambient color.

The complete Phong shading model, where we sum over all lights $l_i$, is

$$\sum_i c_{l_i} (k_d (\mathbf{n} \cdot \mathbf{L_i}) + k_s (\mathbf{R} \cdot \mathbf{e})^p + k_a)$$

## 2.2 Shading Types

### 2.2.1 Per-Triangle

Here, we do a lighting computation for one point on each triangle and shade the whole triangle with the resulting color. This is fast but gives a flat, faceted appearance.

### 2.2.2 Per-Vertex

Here, we do a lighting computation at each vertex and interpolate those *colors* across the triangles. This is also fast, but utilizes inaccurate normals for some points.

Otherwise known as **Gouraud shading**.

### 2.2.3 Per-Pixel

Here, we interpolate *normals* across triangles, then do a lighting computation at each pixel. This is the slowest option, but gives the best results because it can simulate curved surfaces.

Otherwise known as **Phong shading**.