Bump mapping

A method of "faking" the normals of a surface, so that Phong lighting gives the appearance of a bumpy.

Only works with Phong interpolation.

Displacements - in direction of normals \( \vec{n}(u,v) \), distance \( u \cdot d(u,v) \)

\[
\vec{P}^*(u,v) = \vec{P}(u,v) + d(u,v) \cdot \vec{n}(u,v) = \boxed{\vec{P}^* = \vec{P} + d\vec{n}}
\]
\[ \tilde{p}^* = \tilde{p} + d \cdot \tilde{n} \]

Perturbed normal:
\[ \tilde{n}^* = \tilde{p}_u \times \tilde{p}_v \]
\[ \frac{\partial \tilde{p}^*}{\partial u} = \tilde{p}_u = \tilde{p}_u + d_u \cdot \tilde{n} + d \cdot \tilde{n}_u = \frac{\partial \tilde{p}}{\partial u} + \frac{\partial d}{\partial u} \cdot \tilde{n} \]

\[ \tilde{p}_u \approx \tilde{p}_u + d_u \tilde{n} \]

Likewise:
\[ \tilde{p}_v \approx \tilde{p}_v + d_v \tilde{n} \]

So:
\[ \tilde{n}^* = \tilde{p}_u \times \tilde{p}_v \approx [\tilde{p}_u \times \tilde{p}_v + d_u \tilde{n} \times \tilde{p}_v + d_v \tilde{p}_u \times \tilde{n}] \]

Need to know: \( \tilde{p}_u, \tilde{p}_v, d_u, d_v, \tilde{n} \)

\( d \)-displacement - can be given in a texture map:
\( \frac{\partial d_u}{\partial u}, \frac{\partial d_v}{\partial v} \) - can compute with finite differences. Alternately: the Texture Map holds \( d_u, d_v \) directly.
Color: RGB / CMYK - not covering HSL or sRGB.

(First part of the Color Chapter).

Tri-chromatic Theory of Color

(Most) humans see colors in 3 categories: Red, Green, Blue.

Physiology: Retina has 3 kinds of cones:
- selectively sensitive to Red, Green, Blue.
- Rods - sensitive to light at low light levels, but don't discern separate colors.
Sensitivity of cones:

Red Green Blue

frequency

Opponent Theory of Vision - Hering 1878

(Most) humans see color in:

Red vs Green
Blue vs Yellow
Light vs Dark

Physiology: Your retina encodes colors into these three channels
Both theories light comes in a 3D space at standard levels of illumination.

Any color visible color "C" can be expressed as

\[ C = \alpha R + \beta G + \delta B \quad \alpha + \beta + \delta = 1 \]

Sometimes \( \alpha, \beta, \text{or} \delta \) is negative.
Additive Colors  
Red, Green, Blue  
Start with a black background,  
add red, green, blue light in mixtures.

Subtractive colors  
Silent with white & subtract out  
Cyan, Magenta or Yellow  
Example - film, paint, inks, printed matter

Printers  
CMYK  
K - black.

Professional Printers - Use  
CMYK + C, M, Y - magenta  
C, M, Y, K - green
Nominal conversion between RGB and CMY is:

\[
\begin{align*}
R &= 1 - C \\
G &= 1 - M \\
B &= 1 - Y \\
C &= 1 - R \\
M &= 1 - G \\
Y &= 1 - B
\end{align*}
\]

Color representations:

OpenGL often: floating point for RGB values.

Common representatives: 8-bits for R, G, B each
called 24-bit or 32-bit color.

"millions of colors"

Old representation: 16 bit color.

"thousands of colors"

5 bits for R, G, B, 1-bit for A.
8-bit color: 3 bits for Red
3 " " Green
2 bits for Blue

CLUT - Color look-up table
Typically a table of 256 many 24 (32) bit colors.
Each pixel in an image has a 8-bit byte specifying one of the 256 colors.
GIF's use this. (Plus they use Lempel-Ziv compression).

Rec 2020 for UHDTV
uses either 10 or 12 bits for each of Red, Green, Blue.
Next topic: Bézier curves

Control polygon

A Bézier curve

Defined for \( u \in [0, 1] \)

\[
\begin{align*}
\vec{g}(0) &= \vec{P}_0 \\
\vec{g}(1) &= \vec{P}_3 \\
\vec{g}'(0) &= 3(\vec{P}_1 - \vec{P}_0) \\
\vec{g}'(1) &= 3(\vec{P}_3 - \vec{P}_2)
\end{align*}
\]

Control points of \( g(u) \) are just 4 points \( \vec{P}_0, \vec{P}_1, \vec{P}_2, \vec{P}_3 \)
Font design: - Boundaries of letters are Bézier curves.

Manufacturing: