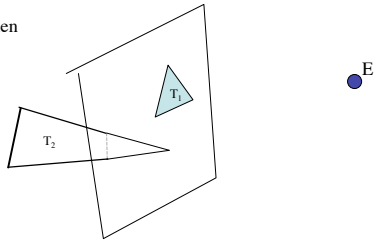
Hidden Surfaces and Shading

CS 351-50

10.22.03

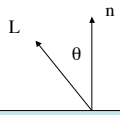
BSP Tree

if ($f_1(E) < 0$) then
  draw T_1
  draw T_2
else
  draw T_2
  draw T_1

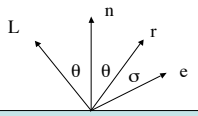


$f_1(p) = 0$ is the implicit plane equation of the plane containing T_1

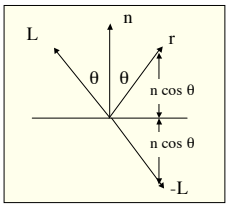
Lambert's law

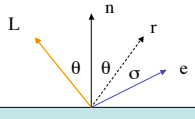


Phong Shading



Phong Shading





OpenGL: Lighting and Materials

Color Material

```
glEnable(GL_COLOR_MATERIAL);
glColorMaterial(GL_FRONT, GL_DIFFUSE);
/* now glColor* changes diffuse reflection  */
glColor3f(0.2, 0.5, 0.8);
/* draw some objects here */
glColorMaterial(GL_FRONT, GL_SPECULAR);
/* glColor* no longer changes diffuse reflection  */
/* now glColor* changes specular reflection  */
glColor3f(0.9, 0.0, 0.2);
/* draw other objects here */
glDisable(GL_COLOR_MATERIAL);
```

Coding Example

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>
void init(void) {
  GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
  GLfloat mat_shininess[] = { 50.0 };
  GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };
  glClearColor (0.0, 0.0, 0.0, 0.0);
  glShadeModel (GL_SMOOTH);
  glMaterialfv(GL_FRONT, GL_SPECULAR,
     mat_specular);
  glMaterialfv(GL_FRONT, GL_SHININESS,
     mat_shininess);
  glLightfv(GL_LIGHT0, GL_POSITION,
     light_position);
  glEnable(GL_LIGHTING);
  glEnable(GL_LIGHT0);
  glEnable(GL_DEPTH_TEST);
}
void display(void){
  glClear (GL_COLOR_BUFFER_BIT |
     GL_DEPTH_BUFFER_BIT);
  glutSolidSphere (1.0, 20, 16);
  glFlush ();
}
void reshape (int w, int h){
  glViewport (0, 0, (GLsizei) w, (GLsizei) h);
  glMatrixMode (GL_PROJECTION);
  glLoadIdentity();
  if (w <= h)
    glOrtho (-1.5, 1.5, -1.5*(GLfloat)h/(GLfloat)w,
      1.5*(GLfloat)h/(GLfloat)w, -10.0, 10.0);
  else
    glOrtho (-1.5*(GLfloat)w/(GLfloat)h,
      1.5*(GLfloat)w/(GLfloat)h, -1.5, 1.5, -10.0, 10.0);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
}
```

Coding Example (Con't.)

```
int main(int argc, char** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
  glutInitWindowSize (500, 500);
  glutInitWindowPosition (100, 100);
  glutCreateWindow (argv[0]);
  init ();
  glutDisplayFunc(display);
  glutReshapeFunc(reshape);
  glutMainLoop();
  return 0;
}
```

Adding a spotlight

```
GLfloat light1_ambient[] = { 0.2, 0.2, 0.2, 1.0 };
GLfloat light1_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat light1_specular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat light1_position[] = { -2.0, 2.0, 1.0, 1.0 };
GLfloat spot_direction[] = { -1.0, -1.0, 0.0 };

glLightfv(GL_LIGHT1, GL_AMBIENT, light1_ambient);
glLightfv(GL_LIGHT1, GL_DIFFUSE, light1_diffuse);
glLightfv(GL_LIGHT1, GL_SPECULAR, light1_specular);
glLightfv(GL_LIGHT1, GL_POSITION, light1_position);
glLightf(GL_LIGHT1, GL_CONSTANT_ATTENUATION, 1.5);
glLightf(GL_LIGHT1, GL_LINEAR_ATTENUATION, 0.5);
glLightf(GL_LIGHT1, GL_QUADRATIC_ATTENUATION, 0.2);

glLightf(GL_LIGHT1, GL_SPOT_CUTOFF, 45.0);
glLightfv(GL_LIGHT1, GL_SPOT_DIRECTION, spot_direction);
glLightf(GL_LIGHT1, GL_SPOT_EXPONENT, 2.0);

glEnable(GL_LIGHT1);
```

OpenGL
Example 5-4 : Stationary Light Source

```
glViewport (0, 0, (GLsizei) w, (GLsizei) h);
glMatrixMode (GL_PROJECTION);
glLoadIdentity();
if (w <= h)
   glOrtho (-1.5, 1.5, -1.5*h/w, 1.5*h/w, -10.0, 10.0);
else
   glOrtho (-1.5*w/h, 1.5*w/h, -1.5, 1.5, -10.0, 10.0);
glMatrixMode (GL_MODELVIEW);
glLoadIdentity();

/* later in init() */
GLfloat light_position[] = { 1.0, 1.0, 1.0, 1.0 };
glLightfv(GL_LIGHT0, GL_POSITION, position);
```

OpenGL
Example 5-5 : Independently Moving Light Source

```
static GLdouble spin;

void display(void)
{
   GLfloat light_position[] = { 0.0, 0.0, 1.5, 1.0 };
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

   glPushMatrix();
      gluLookAt (0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
      glPushMatrix();
         glRotated(spin, 1.0, 0.0, 0.0);
         glLightfv(GL_LIGHT0, GL_POSITION, light_position);
      glPopMatrix();
      glutSolidTorus (0.275, 0.85, 8, 15);
   glPopMatrix();
   glFlush();
}
```

OpenGL Lighting Equation

vertex color = $\text{emission}_{\text{material}}$ +

$\text{ambient}_{\text{light model}} * \text{ambient}_{\text{material}}$ +

$$\sum_{i=0}^{n-1} (1/(k_c + k_l * d + k_q d^2)) * (\text{spotlight effect})_i *$$

[$\text{ambient}_{\text{light}} * \text{ambient}_{\text{material}}$ +

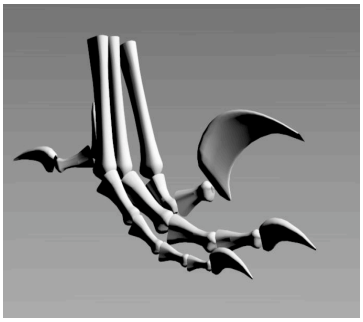
(max { L · n , 0}) * $\text{diffuse}_{\text{light}} * \text{diffuse}_{\text{material}}$ +

(max { s · n , 0})shininess * $\text{specular}_{\text{light}} * \text{specular}_{\text{material}}$]$_i$

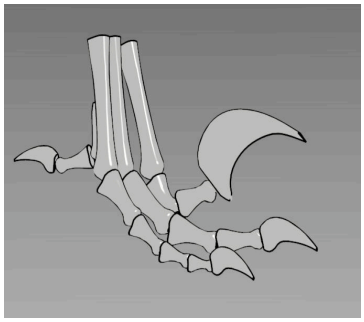
Artistic Shading

Diffuse shaded model

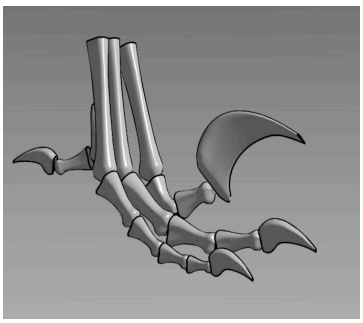
$I = c_r(c_a + c_l \max(0, L \cdot n))$ with $c_r = c_l = 1$ and $c_a = 0$.

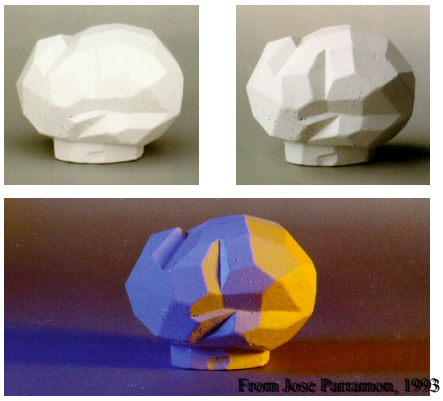


Just Highlights and Edge Lines



Hand-tuned Phong shading





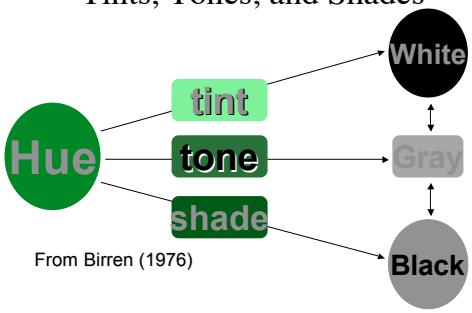
Shading used by Artists

Complementary Shading

Final Image

From "The Book of Color" by Jose Parramon, 1993

Tints, Tones, and Shades

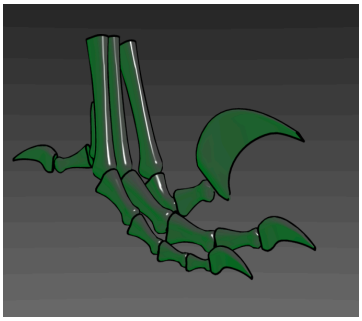


From Birren (1976)

Creating Tones

Green to Gray (tone)

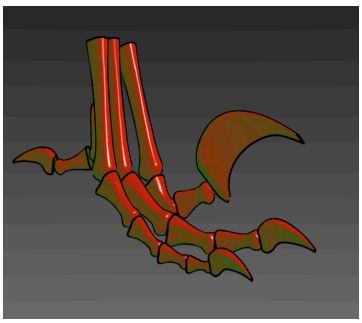
Model Shaded using Tones



Using Color Temperature

Warm to Cool Hue Shift

Constant Luminance Tone Rendering

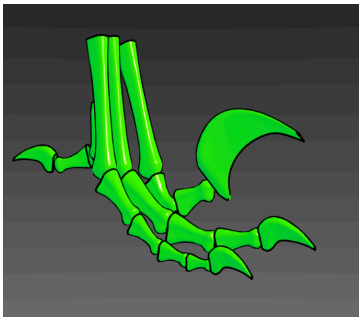


Creating Undertones

Warm to Cool Hue Shift

Green with Warm to Cool Hue Shift

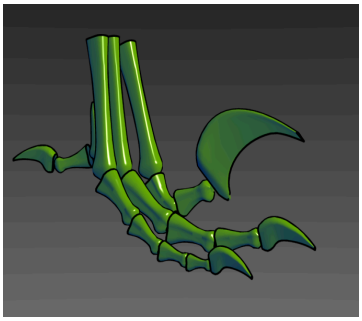
Model tone shaded with cool to warm undertones

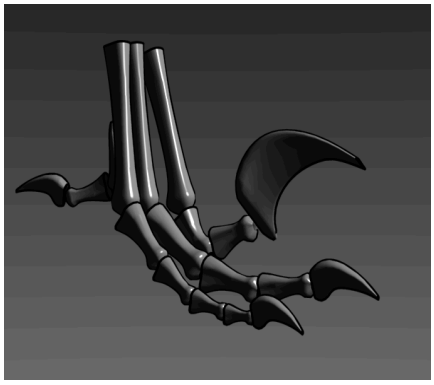


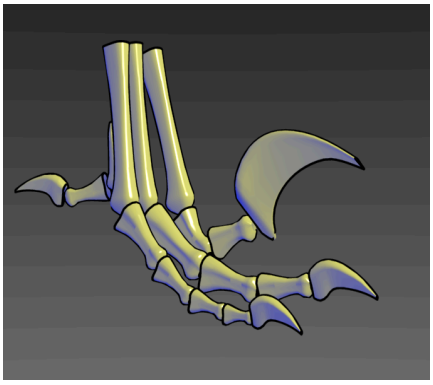
Combining Tones with Undertones

Green with Tone and Undertone

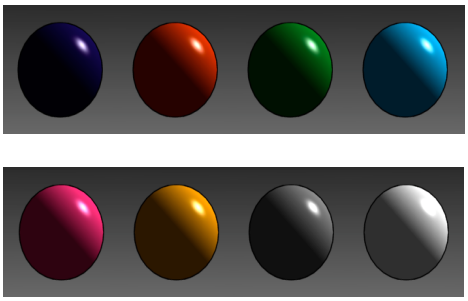
Model shaded with tones and undertones



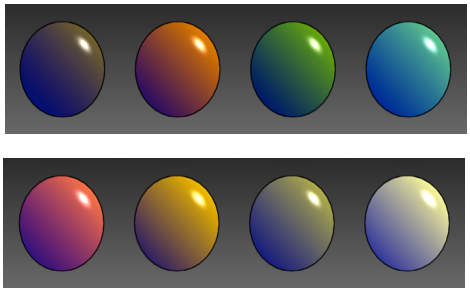




Phong Shaded Spheres



Spheres with New Shading



Phong Shading Formula

$$c = c_r (c_a + c_l \max(0, L \cdot n)) + c_l c_p \cos(h \cdot n)^n$$

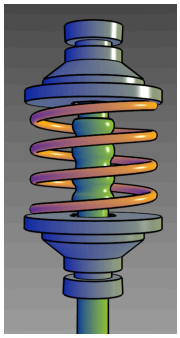
New Shading Formula

$$I = k_w c_{warm} + (1 - k_w) c_{cool}$$

where

$$k_w = (1 + (L \cdot n))*.5)$$

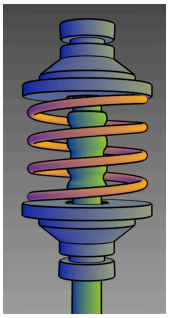
New Shading



OpenGL Approximation

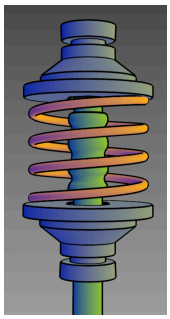
Without Highlights

Light RGB Intensities
L_1 = (0.5, 0.5, 0.0)
L_2 = (-0.5, -0.5, 0)

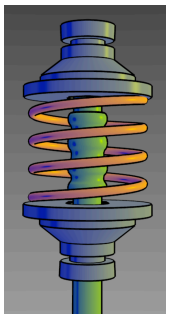


OpenGL Approximation

Without Highlights

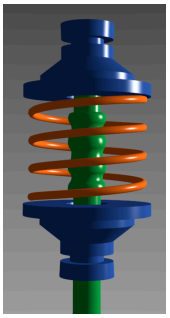


With highlights

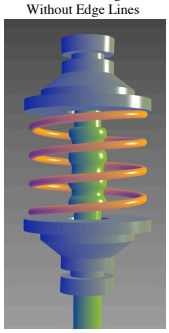


Warm to Cool Shading

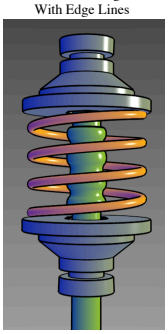
Phong Shaded



New Shading
Without Edge Lines



New Shading
With Edge Lines

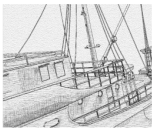


Toon Shading

A regular 3D version of Olaf

A regular 3D version of a boat

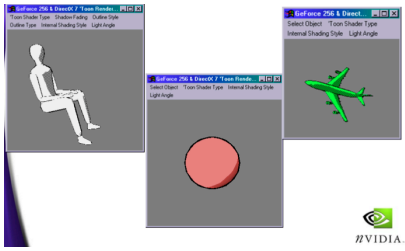
A Toon version of Olaf



A Toon pencil sketch version of the boat (Boat images courtesy of Discreet)

Intel: http://www.intel.com/labs/media/3dsoftware/nonphoto.htm

Toon Shading



Nvidia: developer.nvidia.com/object/Toon_Shading.html

Toon Shading

Blender: w3imagis.imag.fr/Membres/Jean-Dominique.Gascuel/DEAIVR/ Cours2002/17%20janvier/Blender-tutorial80.pdf