CIS 441/541: Introduction to Computer Graphics
Lecture 5: Math Basics, Lighting Introduction & Phong Lighting

April 19th, 2013  Hank Childs, University of Oregon
Everyone understand what to do?

My tips on debugging...

Keep in mind:

```c
for (int j = leftIdx; j <= rightIdx; j++)
{
    if (j < 0 || j >= width)
        continue;
    double proportion;
    if (rightPos != leftPos)
        proportion = (((double)leftPos) / (rightPos - leftPos));
    else
        proportion = 1.0;
    unsigned char rgb[3];
    rgb[0] = (unsigned char) ceil441(255.0*(leftRGB[0] + proportion*
        (rightRGB[0]-leftRGB[0])));
    rgb[1] = (unsigned char) ceil441(255.0*(leftRGB[1] + proportion*
        (rightRGB[1]-leftRGB[1])));
    rgb[2] = (unsigned char) ceil441(255.0*(leftRGB[2] + proportion*
        (rightRGB[2]-leftRGB[2])));
    double z = leftZ + proportion*(rightZ-leftZ);
    Assign(j, i, rgb, z);
}
Our goal with 1E: add shading
My belief about Project #1X

![Diagram showing the relationship between complexity and project concepts versus coding. The diagram includes two lines: one for concepts increasing as complexity decreases, and another for coding increasing as complexity increases. Projects 1B and 1F are labeled on the x-axis.]
Math Basics
Lighting Basics
The Phong Model
Outline

- Math Basics
- Lighting Basics
- The Phong Model
What is the norm of a vector?

- The norm of a vector is its length
  - Denoted with $\| \cdot \|$

- For a vector $A = (A.x, A.y)$,
  $$\| A \| = \sqrt{A.x \cdot A.x + A.y \cdot A.y}$$

- Physical interpretation:

- For 3D, $\| A \| = \sqrt{A.x \cdot A.x + A.y \cdot A.y + A.z \cdot A.z}$
What does it mean for a vector to be normalized?

- The vector A is normalized if \( ||A|| = 1 \).
  - This is also called a unit vector.
- To obtain a normalized vector, take \( A/||A|| \)
- Many of the operations we will discuss today will only work correctly with normalized vectors.
What is the normal of a triangle?

- A triangle coincides with a flat plane.
- A triangle’s normal is the vector perpendicular to that plane.
- If a triangle is on plane \( Ax + By + Cz = D \), then the triangle’s normal is \( (A, B, C) \)
Norm, Normal, Normalize, Oh My!

- Norm: the length of a vector ($||A||$)
- Normal: a perpendicular vector to a plane coincident with geometry
- Normalize: the operation to create a vector with length 1 ($A/||A||$)
- All 3 are important for today’s lecture
What is a dot product?

- \( A \cdot B = A.x \times B.x + A.y \times B.y \)
- **Physical interpretation:**
  - \( A \cdot B = \cos(\alpha)/(||A|| \times ||B||) \)
What is the cross product?

\[ \mathbf{A} \times \mathbf{B} = (A.y \cdot B.z - A.z \cdot B.y, B.x - A.z - A.x \cdot B.z, A.x \cdot B.y - A.y \cdot B.x) \]

- What is the physical interpretation of a cross product?
  - Finds a vector perpendicular to both \( \mathbf{A} \) and \( \mathbf{B} \).
Easy Way to Calculate Normal For a Triangle

Normal = (C-A)x(B-A)
Two ways to treat normals:

- Constant over a triangle
- Varying over a triangle

- Constant over a triangle $\leftrightarrow$ flat shading
- Varying over a triangle $\leftrightarrow$ smooth shading
Flat vs Smooth Shading
Lighting and Normals

- Two ways to treat normals:
  - Constant over a triangle
  - Varying over a triangle

- Constant over a triangle $\leftrightarrow$ flat shading
  - Take $(C-A) \times (B-A)$ as normal over whole triangle

- Varying over a triangle $\leftrightarrow$ smooth shading
  - Calculate normal at vertex, then use linear interpolation
    - How do you calculate normal at a vertex?
    - How do you linearly interpolate normals?
Algorithm:

For vertex V,
- Find all triangles $T_i$ incident to V
- $\text{Normal}(V) = \{0,0,0\}$
- NumIncident = 0
- For each $T_i$,
  - calculate $\text{Normal}(T_i)$
  - $\text{Normal}(V) += \text{Normal}(T_i)$
  - NumIncident++
- $\text{Normal}(V) /= \text{NumIncident}$

Note: our data structures don’t allow for “Find all triangles $T_i$ incident to V” very easily.

Vertex normals are precalculated for 1E
LERPing vectors

- LERP = Linear Interpolate
- Goal: interpolate vector between A and B.
- Consider vector X, where X = B-A
- Back to normal LERP:
  - A + t*(B-A) = A+t*X
- You will need to LERP vectors for 1E
Outline

- Math Basics
- Lighting Basics
- The Phong Model
Scattering

• Light strikes A
  - Some scattered
  - Some absorbed

• Some of scattered light strikes B
  - Some scattered
  - Some absorbed

• Some of this scattered light strikes A
  and so on
Global Effects

shadow

multiple reflection

translucent surface
Local vs Global Rendering

- Correct shading requires a global calculation involving all objects and light sources
  - Incompatible with pipeline model which shades each polygon independently (local rendering)
- However, in computer graphics, especially real time graphics, we are happy if things “look right”
  - Exist many techniques for approximating global effects
Light-Material Interaction

- Light that strikes an object is partially absorbed and partially scattered (reflected)
- The amount reflected determines the color and brightness of the object
  - A surface appears red under white light because the red component of the light is reflected and the rest is absorbed
- The reflected light is scattered in a manner that depends on the smoothness and orientation of the surface
General light sources are difficult to work with because we must integrate light coming from all points on the source.
Simple Light Sources

• **Point source**
  - Model with position and color
  - Distant source = infinite distance away (parallel)

• **Spotlight**
  - Restrict light from ideal point source

• *(We will do point sources for 1E … and this class)*
Surface Types

- The smoother a surface, the more reflected light is concentrated in the direction a perfect mirror would reflect the light.
- A very rough surface scatters light in all directions.

smooth surface

rough surface
Our goal:

- For each pixel, calculate a shading factor
- Shading factor typically between 0 and 1, but sometimes >1
  - Shading >1 makes a surface more white

3 types of lighting to consider:

- Ambient
- Diffuse
- Specular

Our game plan: Calculate all 3 and combine them.
How to handle shading values greater than 1?

- Color at pixel = (1.0, 0.4, 0.8)
  - Shading value = 0.5
    - Easy!
    - Color = (0.5, 0.2, 0.4) $\rightarrow$ (128, 52, 103)

- Color at pixel = (1.0, 0.4, 0.8)
  - Shading value = 2.0
    - Color = (1.0, 0.8, 1.0) $\rightarrow$ (255, 204, 255)

- Color_R = 255*min(1, R*shading_value)

- This is how specular makes things whiter and whiter.
  - But it won’t put in colors that aren’t there.
We are almost certainly not going to make it through today’s lecture.

If you want to get started on 1E:
- Interpolate normals using LERP (normals provided in the input file)
- Add shading code from previous slide
- (That is two thirds of 1E)
Ambient Lighting

- Ambient light
  - Same amount of light everywhere in scene
  - Can model contribution of many sources and reflecting surfaces

Surface lit with ambient lighting only
Lambertian Surface

- Perfectly diffuse reflector
- Light scattered **equally** in all directions

Extreme zoom-in of part of a diffuse surface … light is scattered in all directions

(this image shows 5 of the directions)

Slide inspired by Ed Angel Computer Graphics Book
Diffuse Lighting

Surface Normal

Lambertian Surface
Diffuse Lighting

Lambertian Surface

No light reflects off the (top) surface
(Light direction and surface normal are perpendicular)
Diffuse Lighting

Lambertian Surface

Surface Normal

When the light squarely hits the surface, then that’s when the most light is reflected.
Diffuse Lighting

How much light should be reflected in this case?

A: \( \cos(\alpha) \)

And note that:

- \( \cos(0) = 1 \)
- \( \cos(90) = 0 \)
Diffuse Lighting

How much light makes it to viewer V1? Viewer V2?

A: \( \cos(\alpha) \) for both

Lambertian surfaces reflect light equally in all directions
Diffuse Lighting

- **Diffuse light**
  - Light distributed evenly in all directions, but amount of light depends on orientation of triangles with respect to light source.
  - Different for each triangle

Surface lit with diffuse lighting only
Lambertian Surface

How much light makes it to viewer V1? Viewer V2?

A: $\cos(\alpha)$ for both

Lambertian surfaces reflect light equally in all directions
What is a dot product?

- $\mathbf{A} \cdot \mathbf{B} = A.x \cdot B.x + A.y \cdot B.y$

- Physical interpretation:
  - $\mathbf{A} \cdot \mathbf{B} = \cos(\alpha)/(||\mathbf{A}|| \cdot ||\mathbf{B}||)$

\[
\begin{align*}
(A.x, B.y) & \quad (B.x, B.y) \\
\alpha & \quad \alpha
\end{align*}
\]
You can calculate the diffuse contribution by taking the dot product of L and N,
Since $L \cdot N = \cos(\alpha)$
(assuming L and N are normalized)
What about cases where $L \cdot N < 0$?
What about cases where $L \cdot N < 0$?

$L \cdot N = -1$
Non-sensical ... takes away light?
Common solution:
Diffuse light = max(0, $L \cdot N$)
But wait…

If you have an open surface, then there is a “back face”. The back face has the opposite normal.
But wait... If you have an open surface, then there is a “back face”. The back face has the opposite normal. How can we deal with this case?
But wait…

If you have an open surface, then there is a “back face”. The back face has the opposite normal.

How can we deal with this case?

Idea #1: encode all triangles twice, with different normals
Idea #2: modify diffuse lighting model
If you have an open surface, then there is a “back face”. The back face has the opposite normal.

How can we deal with this case?

Idea #1: encode all triangles twice, with different normals
Idea #2: modify diffuse lighting model

Diffuse light = abs(L \cdot N)

This is called two-sided lighting
We will use two-sided lighting for project 1E, since we have open surfaces.

Note that Ed Angel book assumes closed surfaces and recommends one-sided lighting.
One-sided lighting with open surfaces is disappointing
The most valuable thing I learned in Freshman Physics

- “angle in = angle out”
The most valuable thing I learned in Freshman Physics

- “angle in = angle out”
Specular Lighting

Light reflects in all directions.
But the surface is smooth, not Lambertian, so amount of reflected light varies.
So how much light??
How much light reflects with specular lighting?

Smooth Surface

Consider V located along reflection ray.

Answer: most possible
Call this “1”
How much light reflects with specular lighting?

Consider V located along perpendicular ray.

*Answer:* none of it
*Call this “0”*
How much light reflects with specular lighting?

Smooth Surface

How much light gets to point V?

Highest proportion of light reflecting
How much light reflects with specular lighting?

Smooth Surface

How much light gets to point V?

A: proportional to \( \cos(\alpha) \)
How much light reflects with specular lighting?

Smooth Surface

How much light gets to point V?

A: proportional to $\cos(\alpha)$
(Shininess strength) * $\cos(\alpha)^\text{(shininess coefficient)}$
Values of $\gamma$ between 100 and 200 correspond to metals.

Values between 5 and 10 give a surface that looks like plastic.
How much light reflects with specular lighting?

How much light gets to point V?

A: proportional to \( \cos(\alpha) \)
(Shininess strength) \( \times \) \( \cos(\alpha)^{\text{(shininess coefficient)}} \)
How much light reflects with specular lighting?

Smooth Surface

Great!
We know that \( \cos(\alpha) = V \cdot R \).
How much light reflects with specular lighting?

Smooth Surface

Great!
We know that \( \cos(\alpha) \) is \( V \cdot R \).
But what is \( R \)?
It is a formula: \( R = 2 \cdot (L \cdot N) \cdot N - L \).
For specular lighting, we will use one-sided lighting for project 1E

- It just looks better

- Diffuse: abs(L·N)
- Specular: max(0, S*(R·V)^η)
Outline

- Math Basics
- Lighting Basics
- The Phong Model
Phong Model

- Combine three lighting effects: ambient, diffuse, specular
Phong Model

- Simple version: 1 light, with “full intensity” (i.e., don’t add an intensity term)

- Phong model
  - \[ \text{Shading\_Amount} = K_a + K_d \times \text{Diffuse} + K_s \times \text{Specular} \]

- Signature:
  - double CalculatePhongShading(LightingParameters &, double *viewDirection, double *normal)
  - For us, viewDirection = (0,0, +1)
struct LightingParameters
{
    LightingParameters(void)
    {
        lightDir[0] = -0.6;
        lightDir[1] = 0;
        lightDir[2] = -0.8;
        Ka = 0.3;
        Kd = 0.7;
        Ks = 5.3;
        alpha = 7.5
    }
};

double lightDir[3]; // The direction of the light source
double Ka;          // The coefficient for ambient lighting.
double Kd;          // The coefficient for diffuse lighting.
double Ks;          // The coefficient for specular lighting.
double alpha;       // The exponent term for specular lighting.
};
Goal: add Phong shading

Extend your project1D code

File proj1e_geometry.vtk available on web (9MB)

File “reader1e.cxx” has code to read triangles from file.

No Cmake, project1e.cxx
Changes to data structures

class Triangle
{
    public:
        double X[3], Y[3], Z[3];
        double colors[3][3];
        double normals[3][3];
};

→ reader1e.cxx will not compile until you make these changes
→ reader1e.cxx will initialize normals at each vertex
More comments

- New: more data to help debug
  - I will make the shading value for each pixel available.
  - I will also make it available for ambient, diffuse, specular.
- Don’t forget to do two-sided lighting
- This project in a nutshell:
  - LERP normal to a pixel
    - You all are great at this now!!
  - Add method called “CalculateShading”.
    - My version of CalculateShading is about ten lines of code.
  - Modify RGB calculation to use shading.
Where Hank spent his debugging time...

Convex surface

Concave surface

Lighting direction

Convex surface

Lighting direction
What to do if you run into trouble?

- OH: today 2:15-4, Monday 12-2, Thurs 3:30-6
- Piazza
- Email me