OpenGL Lighting

15-462 Computer Graphics
Spring 2009

Frank Palermo

“OpenGL is just a bunch of hacks.”

-Adrien Treuille

What Adrien Means...

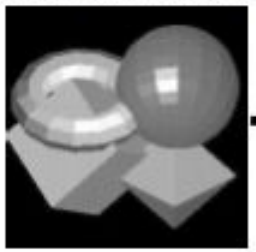
- What Adrien means is that OpenGL was designed to produce reasonable-looking 3D images quickly & simply.
- A lot of its design is not based on the way light actually behaves in the real world, or on the way we perceive a scene.
- Keep that in mind as we discuss the OpenGL pipeline and lighting model.

The OpenGL Pipeline

How does it work?

From the implementor's perspective:

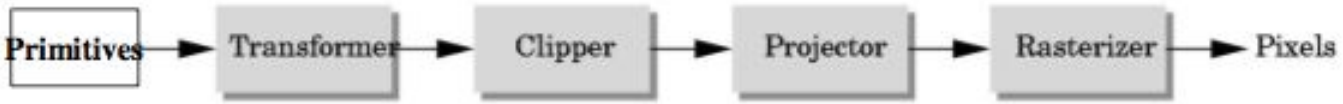
geometric objects
properties: color…
move camera and objects around



pixels

Frame buffer

graphics pipeline

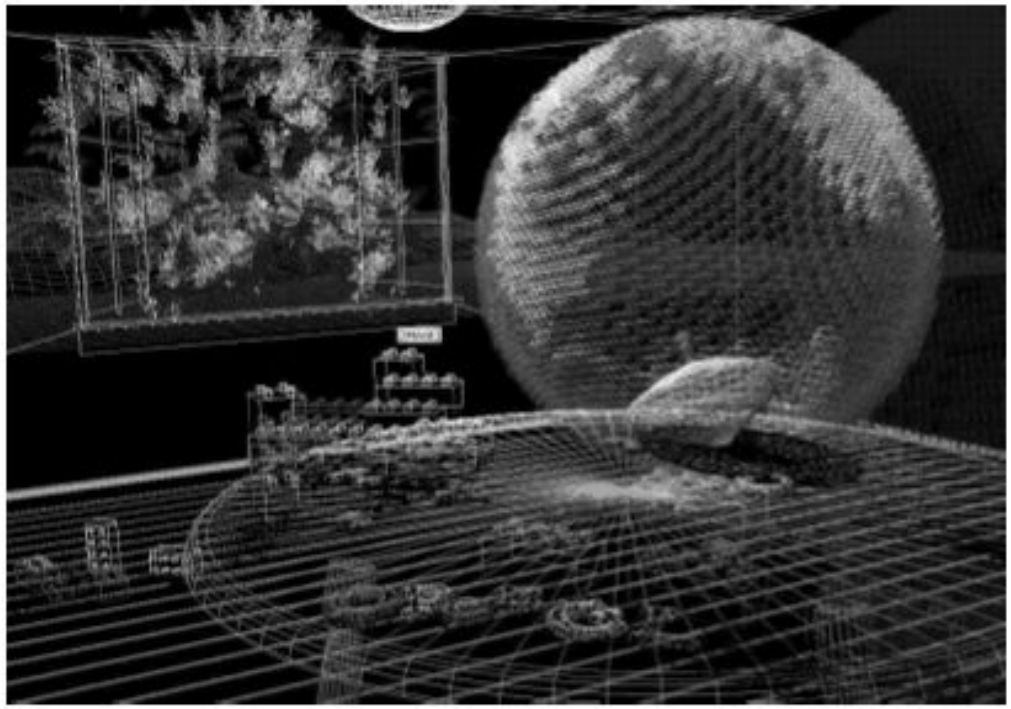


Primitives	Transformer	Clipper	Projector	Rasterizer
Primitives + material properties	**Rotate Translate Scale**	**Is it visible?**	**3D to 2D**	**Convert to pixels**

* Featuring slides shamelessly stolen from a previous teaching of 15-462!

The OpenGL Pipeline

Primitives: drawing a polygon



Build models in appropriate units (microns, meters, etc.).
From simple shapes: triangles, polygons,…

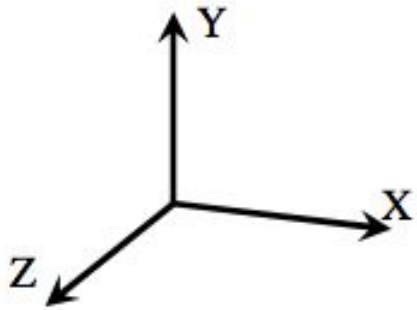
* Featuring slides shamelessly stolen from a previous teaching of 15-462!

The OpenGL Pipeline

Transforms

- Rotate
- Translate
- Scale

- glRotate(x,y,z);
- glTranslate(x,y,z);
- draw geometry

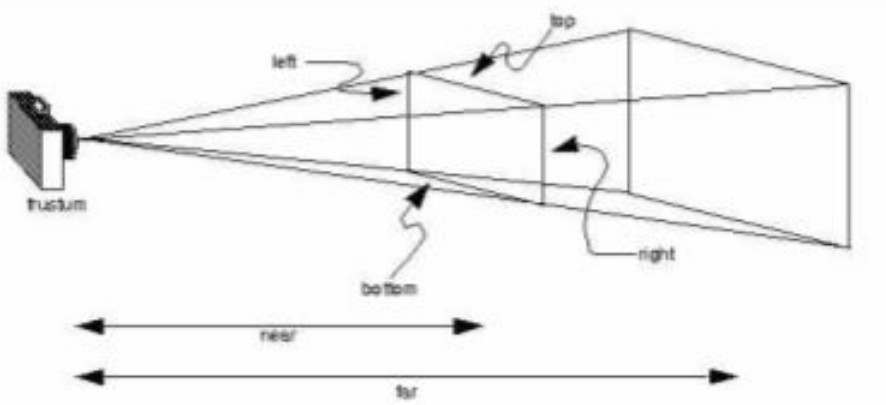


* Featuring slides shamelessly stolen from a previous teaching of 15-462!

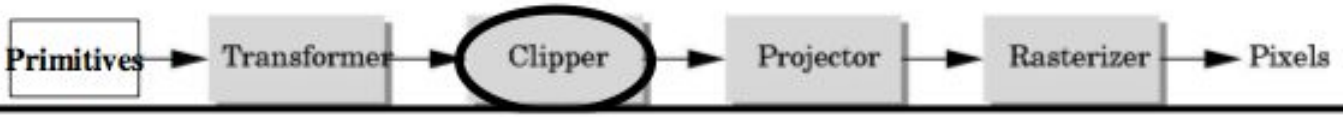
The OpenGL Pipeline

Clipping

Not everything should be visible on the screen



any vertices that lie outside of the viewing volume are clipped



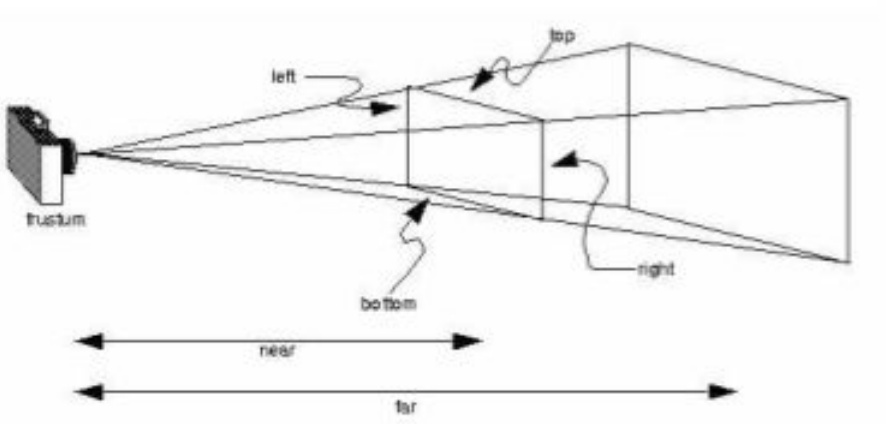
* Featuring slides shamelessly stolen from a previous teaching of 15-462!

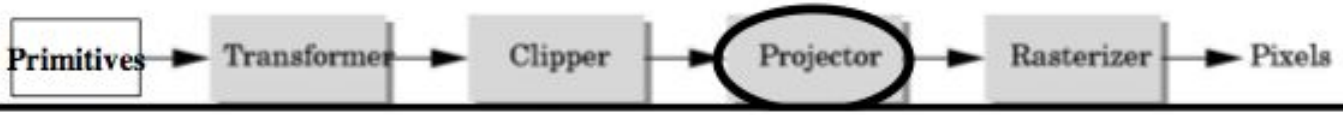
The OpenGL Pipeline

Position it relative to the camera

Perspective projection

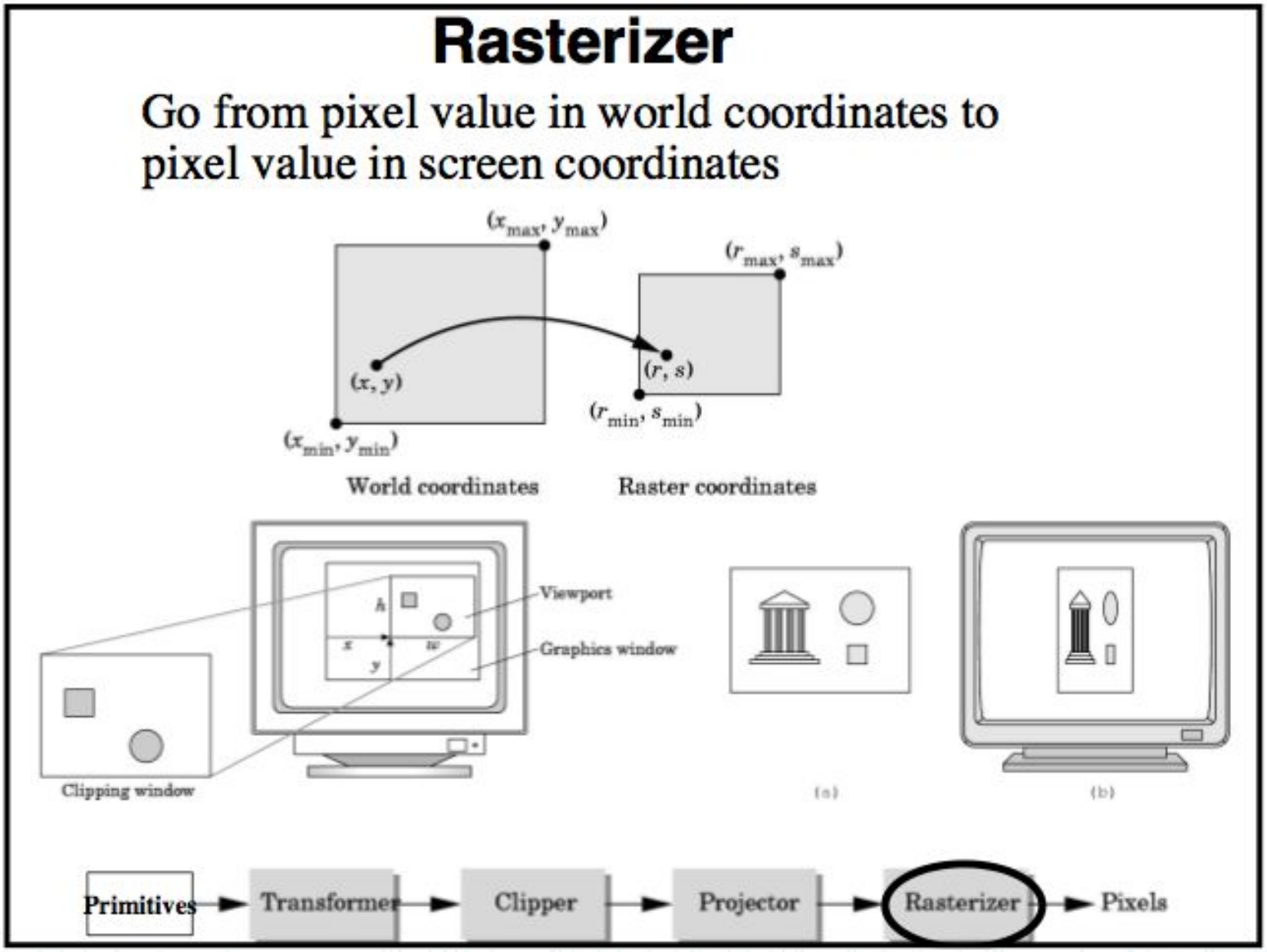
glFrustum (left, right, bottom, top, near, far);





* Featuring slides shamelessly stolen from a previous teaching of 15-462!

The OpenGL Pipeline



* Featuring slides shamelessly stolen from a previous teaching of 15-462!

OpenGL Lighting

- By default, OpenGL's fixed-function pipeline implements the Blinn-Phong Shading Model.
  - Originally from Bui Tuong Phong's Ph.D. work at the University of Utah (1973).
  - Modified by James F. Blinn (1977).
- Light is modeled in three categories: ambient, diffuse, and specular.

Ambient Lighting

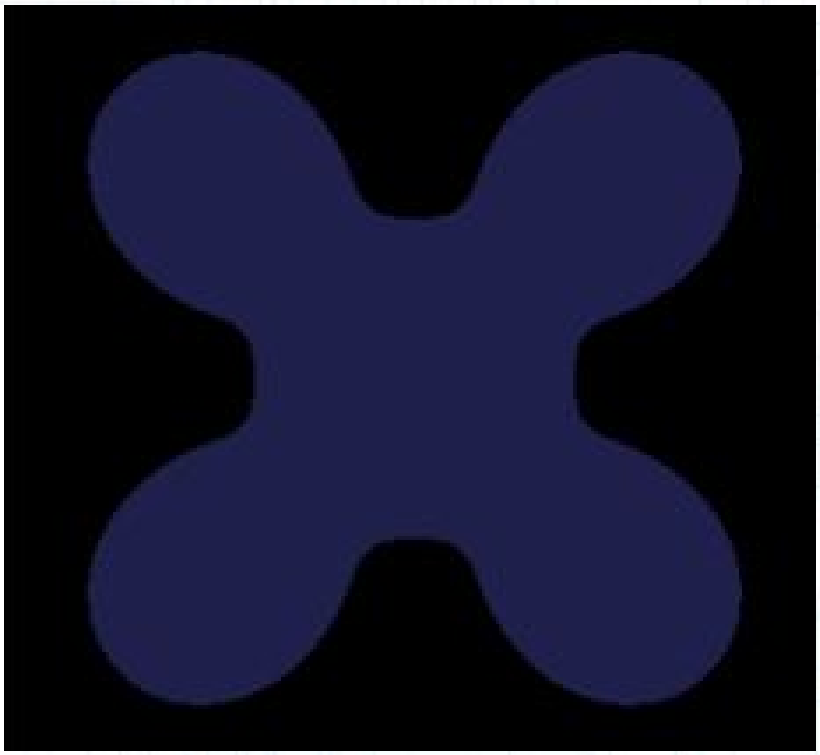


Image: Brad Smith

- Approximates the low level of light that is normally present everywhere in a scene (scattered by many objects before reaching the eye).
- Constant term; applies equally to all points on the object.

Diffuse Lighting

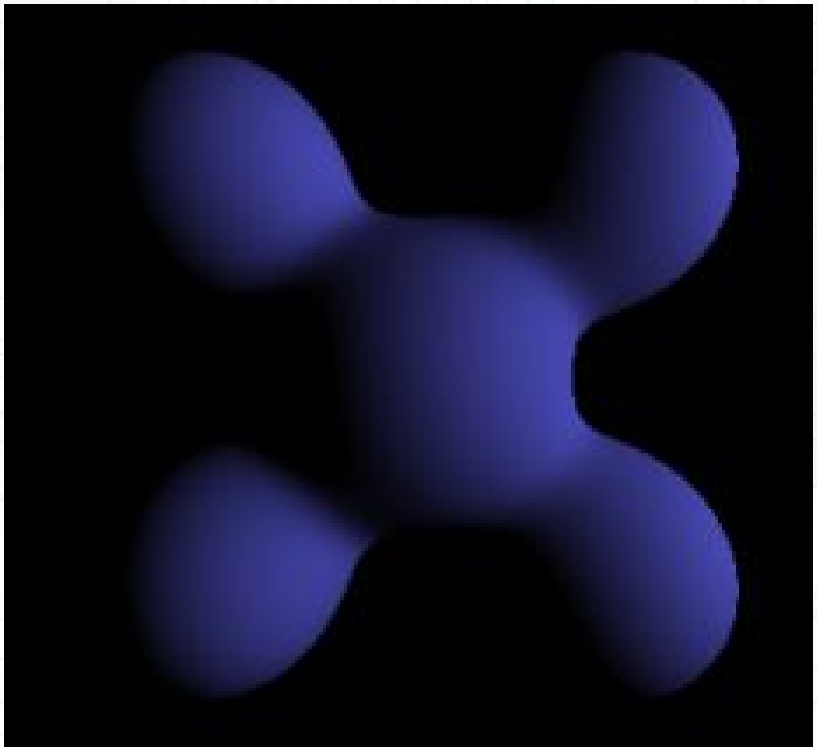


Image: Brad Smith

- Approximates light scattered by objects with rough surfaces.
- Its intensity depends on the angle between the light source and the surface normal (*not* the direction to the viewer).

Specular Lighting

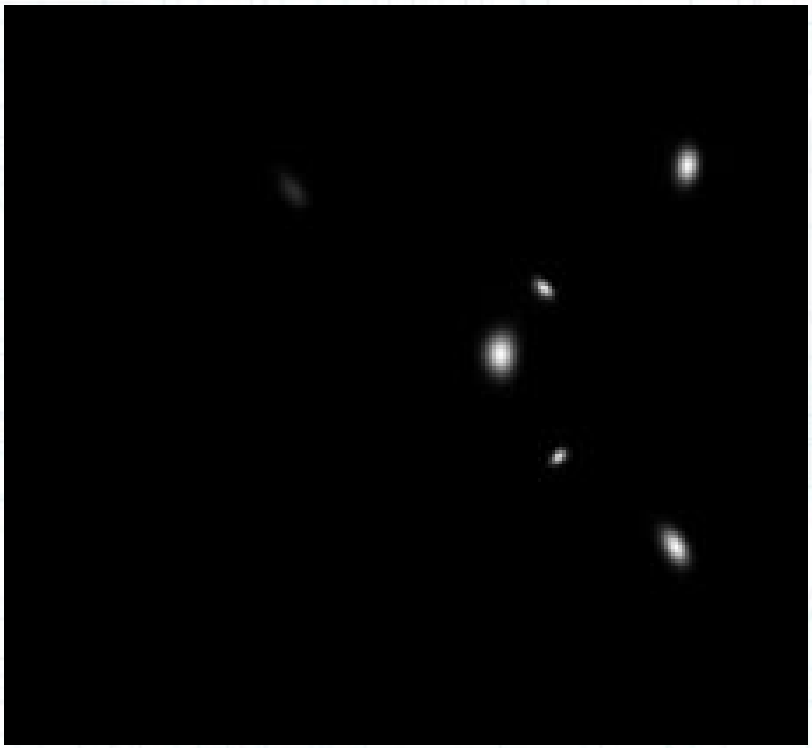


Image: Brad Smith

- Approximates light reflected by “shiny” objects with smooth surfaces.
- Its intensity depends on the angle between the viewer and the direction of a ray reflected from the light source.

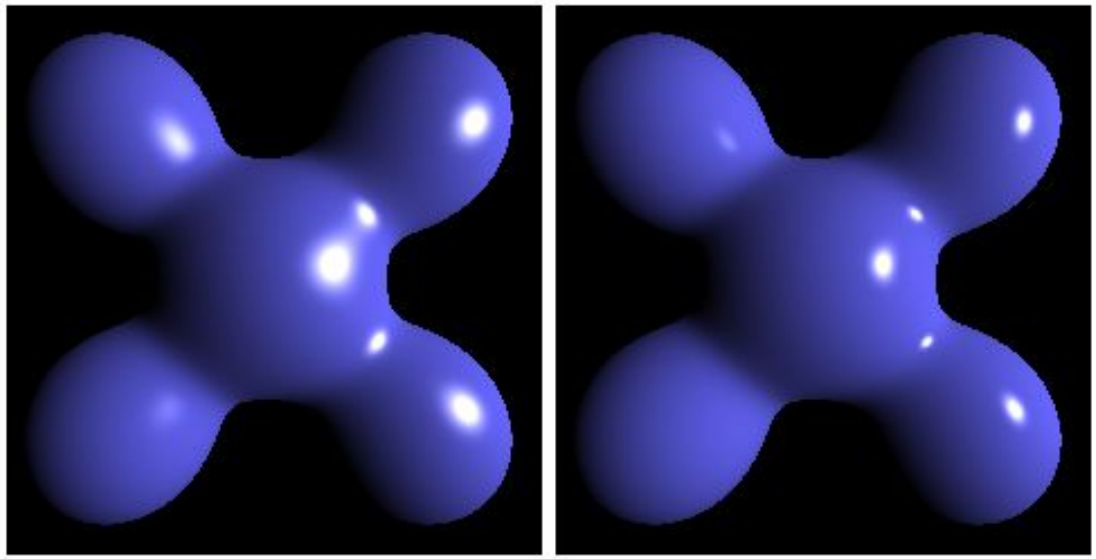
The Equation

$$I_p = k_a i_a + \sum_{\text{lights}} (k_d (L \cdot N) i_d + k_s (R \cdot V)^{\alpha} i_s).$$

- Ka, Kd, Ks = Ambient, Diffuse, and Specular Color (Set for Materials , i.e. Objects)
- Ia, Id, Is = Ambient, Diffuse, and Specular Intensity (Set for Lights)
- Dot Products: Provide the dependence on the light-surface and reflection-viewer angles discussed earlier.

The Blinn-Phong Model In OpenGL

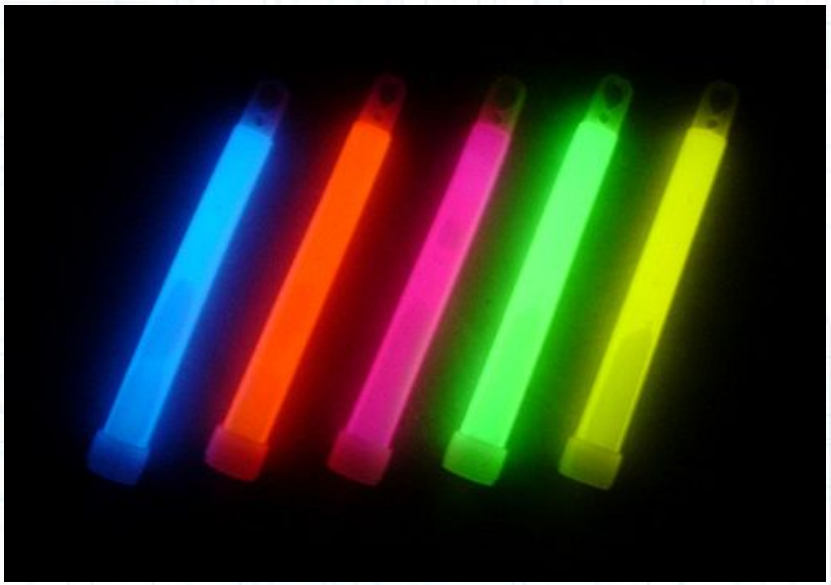
- Phong: The calculation is done over the entire surface.
- Blinn-Phong: The calculation is done only at vertices; the remainder of the surface is interpolated from surrounding vertices.



Blinn-Phong Phong

Image: Brad Smith

Emissive Lighting



- OpenGL adds emissive lighting, which allows an object to “glow” with its own light.
- Is this everything we need to model all possible lighting phenomena?

The Blinn-Phong Model In OpenGL

- OpenGL evaluates the equation for you. You just need to set the material parameters and light intensities (and provide surface normals!).
- Chapter 5 of the OpenGL Programming Guide describes the commands you will need and gives examples of their use.

Some OpenGL Commands

- Setting light intensities:

  glLightfv(GL_LIGHT0, GL_DIFFUSE, Id);

  glLightfv(GL_LIGHT0, GL_SPECULAR, Is);

- Setting material colors:

  glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, Ka);

  glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, Kd);

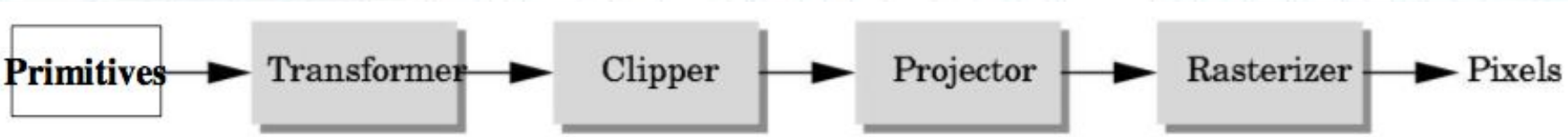
  glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, Ks);

- Setting surface normals:

  glVertex3dv(&vertices[0].x);

  glNormal3dv(&normals[1].x);

Limitations of OpenGL Lighting



- The pipeline is essentially one-way (left to right on the illustration).
- Once a polygon has been rendered, it's forgotten and can't be used as part of any other lighting calculations.
- No mirrors, glow sticks made with emissive light won't actually light up anything else in the scene, etc.
- Can only capture light source → object → viewer (called “direct illumination”).

Direct vs. Global Illumination

- "Fast Separation of Direct and Global Components of a Scene using High Frequency Illumination" by Nayar et. al. (SIGGRAPH 2006).
- OpenGL could render the image at center, but it would miss all of the information in the image at right.

Solutions?

- Use a completely different algorithm such as raytracing or photon mapping which supports global illumination.
  - Projects 3 & 4.
- Program the pipeline to behave differently using GLSL.
  - Project 2.
- Be happy with Blinn-Phong.
  - Project 1 (and lecture next Tuesday!)