CS174A : Introduction to Computer Graphics

Kinsey 1240
MW 4-6pm

Scott Friedman, Ph.D
UCLA Institute for Digital Research and Education

UCLA

Lighting and Shading

- Now that we have moved into 3D…
  - We need to talk about lighting/shading.
  - Used to enhance the effect of dimensionality.
  - Light in the real world is rarely flat and uniform.
  - We would like to approximate how light interacts.
    - By the type of light source
    - How the light reflects off of a surface.

Lighting and Shading

- This means we need to describe
  - Various types of light sources available
  - Properties of a surface defining how light reflects

  - Model of how light interacts
  - Restrict ourselves to local interactions
  - Global lighting models are, generally, too computationally intensive for interactive graphics.
    - – However, there are techniques that attempt to approximate elements of global interactions – e.g. ambient occlusion

Lighting and Shading

- What do we mean by *global interactions*?
  - How light interacts with the material world
- Could model the physics of light.
  - Surfaces either emit, absorb or reflect light.
  - Light reflects, scatters, bleeds, etc.
    - Unfortunately…
      - There is no analytical solution fast enough – even for offline rendering – to be useful.
      - Even approximate solutions are either partial or too slow
        - » Radiosity (thermodynamics based)
        - » Ray/Path Tracing
      - GPUs have sped some of these techniques up.

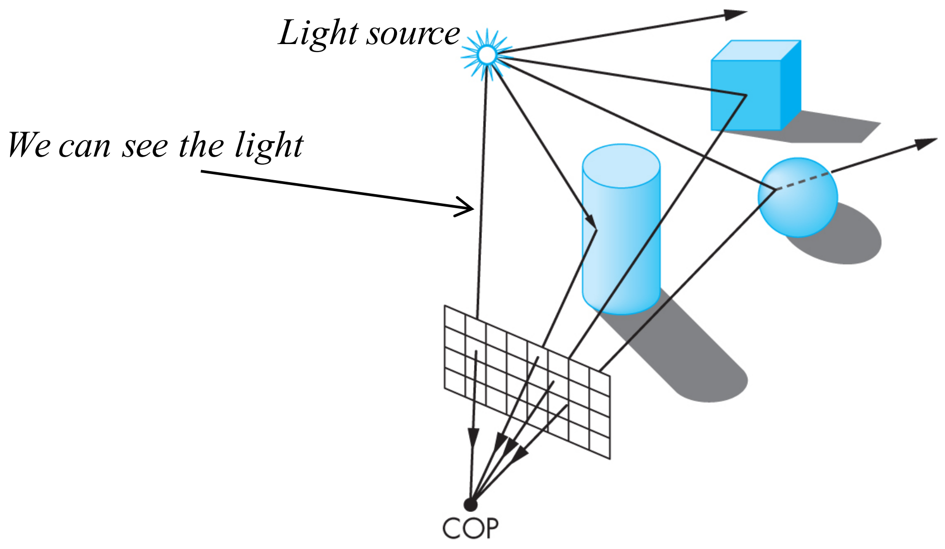
UCLA

Lighting and Shading

- Simplify things.
  - Single interactions only.
    - Between a light source and a surface.
  - Two parts to this problem
    - Model for a light source(s)
    - Model for how a surface reflects a light source.

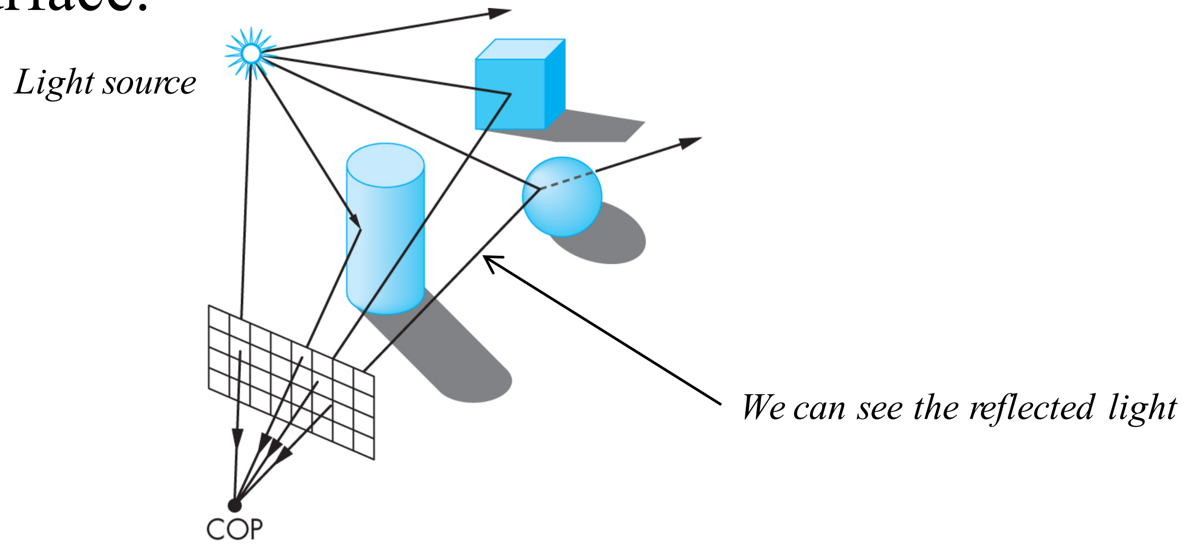
Lighting and Shading

- First, what is going on?
  - Can *see* the color of the light source itself.



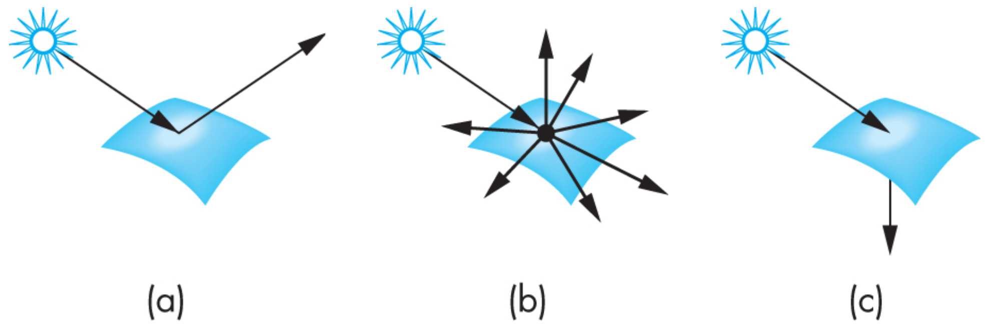
Lighting and Shading

- First, what is going on?
  - Can *see* the color of reflected light
    - Reflected light off a surface after some of the light has been absorbed and or perturbed in some way by the surface.



Lighting and Shading

- Types of surfaces
  - Specular (a)
    - Reflected in a narrow range (mirror or bright spot)
  - Diffuse (b)
    - Reflected is all directions (flat appearance or cloudy day)
  - Translucent (c)
    - Refracted, some may be reflected



Lighting and Shading

- Light source types (models)
  - Ambient
  - Point
  - Spotlight
  - Distant / Infinite
- Each has color and luminance (brightness)
  - Represented with a single RGB value.
  - Color is composed of three elements (again RGB)
    - Ambient (I_a)
    - Diffuse (I_d)
    - Specular (I_s)

Lighting and Shading

- Light sources – Ambient
  - Uniform illumination.
    - Cloudy day, indirect lighting
    - Light is scattered everywhere, directionless.
  - Model our ambient light with a luminance, $\boldsymbol{I_a}$.
    - Every fragment receives the same illumination from $\boldsymbol{I_a}$.
    - Surfaces *may* reflect ambient light differently, however.

Lighting and Shading

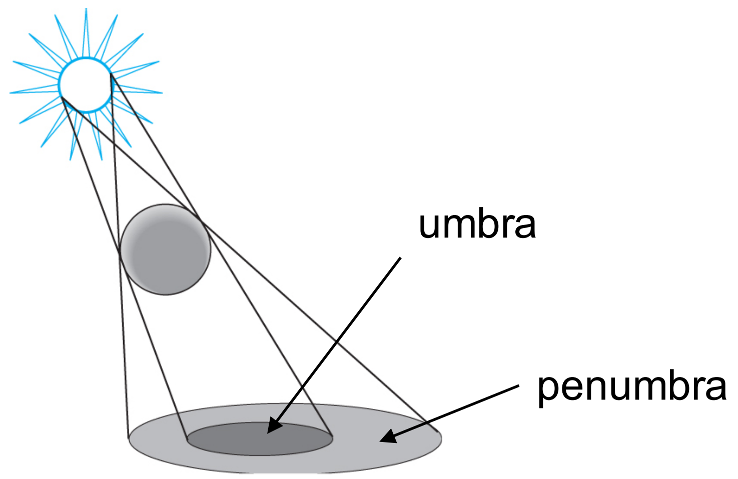
- Light sources – Point
  - Emits light equally in all directions.
    - Light bulb, sun
  - *Location, $I(p_0)$*
  - The amount of light received by a point on a surface is proportional to its distance from a point light source.
    - The terms ***a, b*** and ***c*** are used to soften the light (falloff).
    - Constant, linear and quadratic parameters.

$$d = |p - p_0|^2$$

$$i(p, p_0) = \frac{1}{a + bd + cd^2} I(p_0)$$

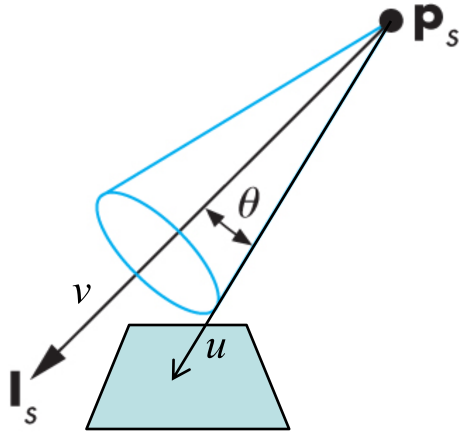
Lighting and Shading

- Light sources – Point
  - Real point sources are not points in the geometric sense
  - They have a *finite* size
    - An object's shadow casts an *umbra* and *penumbra*.
    - Point source model does not generate an umbra or penumbra.



Lighting and Shading

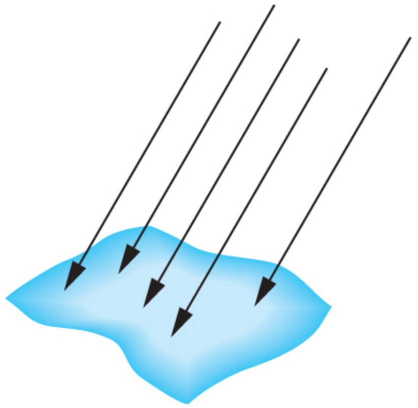
- Light sources – Spotlight
  - Location
  - *Directional.*
  - Distribution of light defined by an angle.
    - Luminance is attenuated by the magnitude of the angle between the spotlight's direction vector and the vector defined by the spotlight's location and a point on a surface. (*angular falloff*)



$$\cos\theta = u \bullet v$$

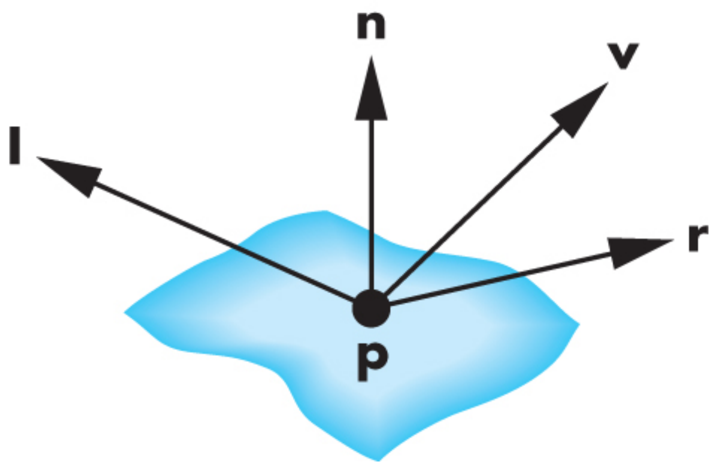
Lighting and Shading

- Light sources – Infinite
  - Light rays are effectively parallel to each other.
  - Direction vector only (i.e. no position)
  - All points on a surface use that same direction vector.
    - No need to recompute the incidence vector for each point on a surface using p_0 *as there is no* p_0



Lighting and Shading

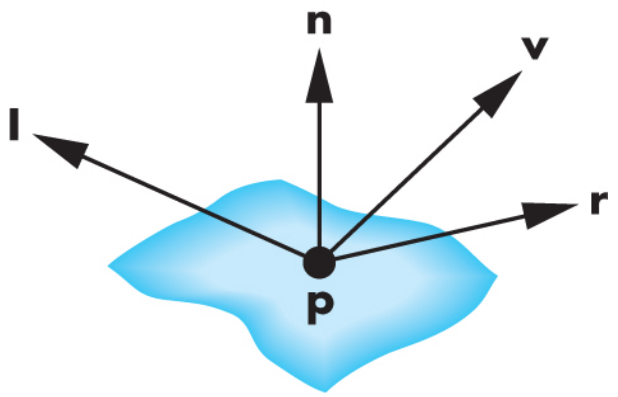
- **Shading** – The Phong Reflection Model
  - Efficient approximation of physical model
  - The Phong model defines four vectors.
    - Enable the calculation of a color value for any point on a surface.



UCLA

Lighting and Shading

- Shading – The Phong Reflection Model
  - **n**, normal at point **P** on the surface.
  - **v**, vector from point **P** to the camera location (COP).
  - ***I***, vector from point **P** to the light source (direction)
  - **r**, vector of the perfectly reflected ray from ***I***.
    - **r** is computed from **n** and ***I***.



Lighting and Shading

- Shading – The Phong Reflection Model
  - Phong supports three types of light interaction
    - Ambient (I_a)
    - Diffuse (I_d)
    - Specular (I_s)
  - These three color elements define a *material*.
    - Associate materials with surfaces (geometry)
    - Surfaces could have more than one material.

Lighting and Shading

- Shading – The Phong Reflection Model
  - Light (color) at a point on a surface
  - Sum contributions from
    - Each light source (***L***) interacting with the surface (***R***)
    - *a*: ambient, *d*: diffuse and *s*: specular terms

$$I = \sum_i (I_{a_i} + I_{d_i} + I_{s_i}) + I_a \quad \text{where}$$

$$(I_{a_i} + I_{d_i} + I_{s_i}) = (L_{a_i} R_a + L_{d_i} R_d + L_{s_i} R_s)$$

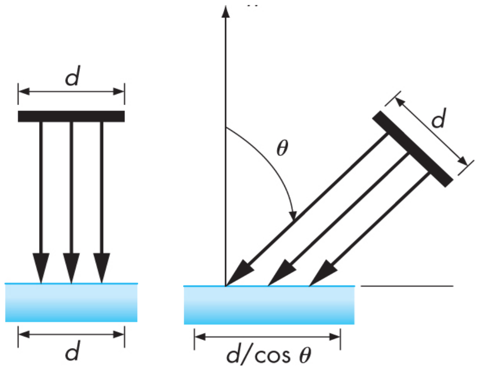
Lighting and Shading

- Shading – The Phong Reflection Model
  - Ambient Reflection
    - The luminance of ambient light at each point on a surface is the same.
    - The amount of ambient light *reflected* is determined by the scalar coefficient $\boldsymbol{k_a}$

$$I_a = k_a L_a \quad \text{where} \quad 0 \le \mathrm{k}_a \le 1$$

Lighting and Shading

- Shading – The Phong Reflection Model
  - Diffuse Reflection
    - No preferred direction of reflection (all directions)
    - Lambert's cosine law is useful to model the overall effect.
      - Only consider the vertical component of the reflected light.
      - As the angle of incidence becomes larger, reflected light gets dimmer.



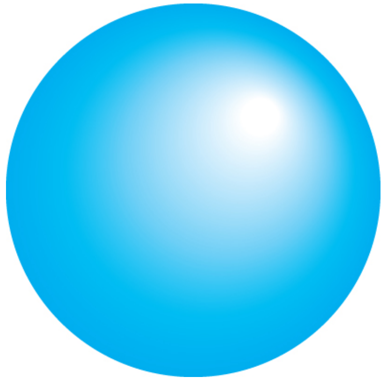
Lighting and Shading

- Shading – The Phong Reflection Model
  - Diffuse Reflection
    - The direction of the light source **l** and the surface normal **n** determine the amount of reflected light.
    - Recall: $\cos\theta = l \cdot n$
    - Reflection coefficient $\boldsymbol{k_d}$
    - Attenuation term accounts for distance (*d*) to the light source.
    - The max() is for handling when the light is below the horizon.

$$I_d = \frac{k_d}{a + bd + cd^2} \max((\mathbf{l} \bullet \mathbf{n}) L_d, 0)$$

Lighting and Shading

- Shading – The Phong Reflection Model
  - Specular Reflection
    - “Shininess” on a smooth surface.
    - Reflection of the light source itself.
    - The smoother the surface the more concentrated it is
    - A mirrored surface has a perfectly specular surface.



Lighting and Shading

- Shading – The Phong Reflection Model
  - Specular Reflection
    - Phong approximates this effect (cannot do a mirror)
    - Determined by the angle between the perfect reflection vector **r** and the direction to the viewer **v** with respect to the point on the surface.
      - k_s, is the reflection coefficient
      - α is the shininess coefficient which controls how narrow or broad the reflection (bright spot) appears.

$$I_s = k_s L_s \max((\mathbf{r} \bullet \mathbf{v})^{\alpha}, 0)$$

UCLA

Lighting and Shading

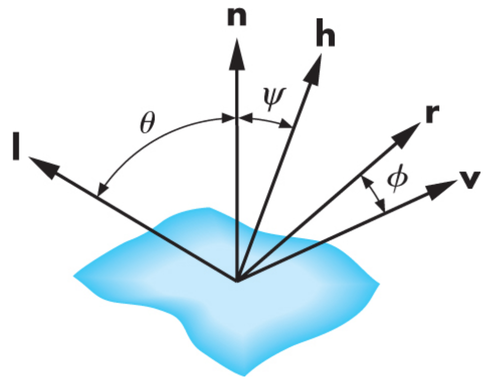
- Shading – The Phong Reflection Model
  - Full Phong Model
    - Combining the diffuse, specular and ambient terms
    - Computed for each light source
    - Could keep the falloffs separate

$$I = \frac{1}{a + bd + cd^2}(k_d L_d \max(\mathbf{l} \bullet \mathbf{n}, 0) + k_s L_s \max((\mathbf{r} \bullet \mathbf{v})^\alpha, 0)) + k_a L_a$$

$$\mathbf{l} = 2(\mathbf{l} \bullet \mathbf{n})\mathbf{n} - 1.$$

Lighting and Shading

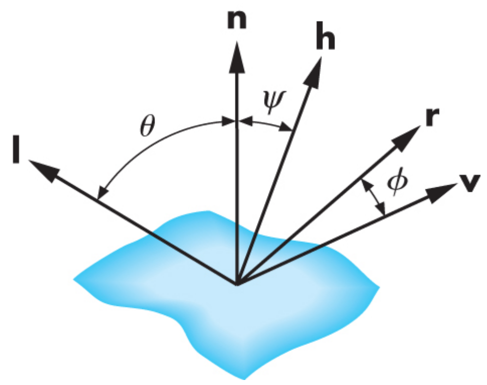
- Shading – The Phong Reflection Model
  - Blinn-Phong Model
    - Phong requires we compute **r**, the reflection vector, for every point on the surface. *(would rather not have do this)*
    - When **l, n, r** and **v** are in the *same plane* the half-way vector **h** can be used to avoid computing **r**.
    - **h** is halfway between **l** and **v, which we know**.



$$\mathbf{h} = \frac{\mathbf{l}+\mathbf{v}}{|\mathbf{l}+\mathbf{v}|}$$

Lighting and Shading

- Shading – The Phong Reflection Model
  - Blinn-Phong Model
    - The angle between **n** and **h** (ψ) is *smaller* than the angle between **r** and **v** (ϕ).
    - The shininess coefficient (α) has to be raised to get approximately the same result as pure Phong.



$$\phi = 2\psi$$

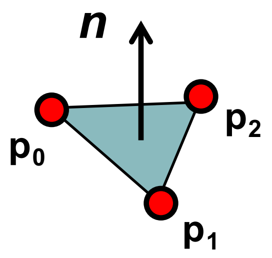
UCLA

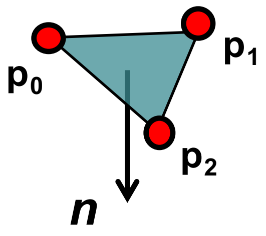
Lighting and Shading

- Shading – The Normal
  - A normal can be computed from three non-collinear points located on a plane. (a triangle?)

$$\mathbf{n} = (\mathbf{p}_2 - \mathbf{p}_0) \times (\mathbf{p}_1 - \mathbf{p}_0).$$

  - The order of the points *matters.*
  - That order we consider the points matters as that *determines* the direction of the normal **n**.





Lighting and Shading

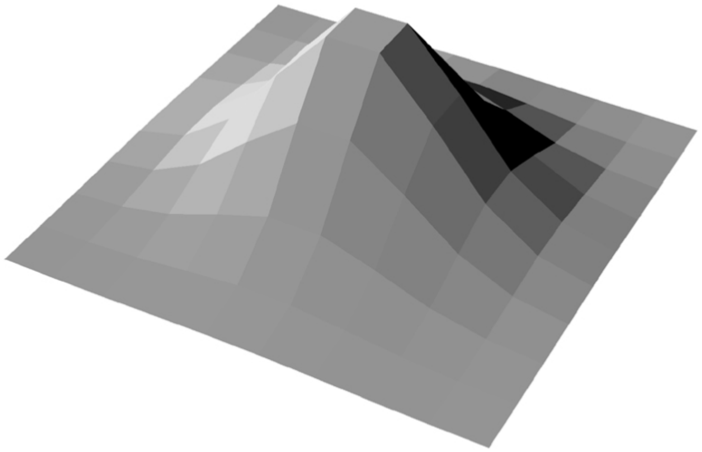
- Shading – The Normal
    - For curved surfaces, how the normal is computed depends on how the surface is represented.
    - Take the unit sphere at the origin, for example.
    - The normal at any point on the sphere is simply: (why?)

$$\mathbf{n} = \mathbf{p}$$

    - Often, in WebGL we are approximating the normal using nearby points and planes (triangles).
    - This is because we can only represent objects, including surfaces, with an approximation using triangle primitives.

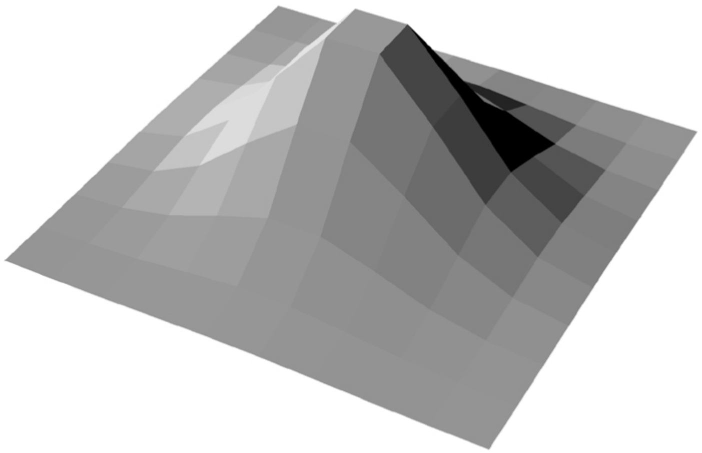
Lighting and Shading

- Shading – Flat
  - Vectors **l, n** and **v** vary as points (**p**) on a surface are evaluated.
  - If the surface is flat, and the light source (**l**) and viewer (**v**) are far away these vectors can be held constant across a surface primitive.
  - ***Flat shading*** where every point on a **surface primitive** receives the same shading (color).
  - We can compute the shading ***once*** per primitive.



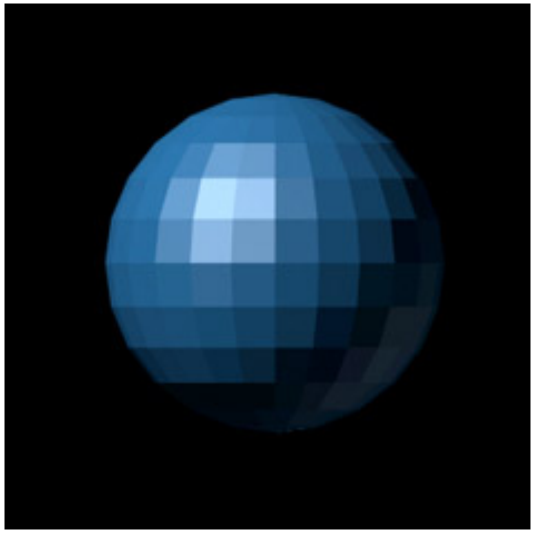
Lighting and Shading

- Shading – Flat
  - Looks unrealistic for a non-flat surface or if the light source or viewer are not far away.
  - Depending on how small the elements are that make up the approximation of the surface, it may not be as visible.
  - Squint your eyes and look at this surface and it looks smoother.



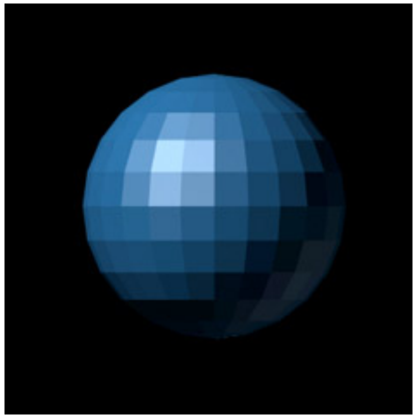
Lighting and Shading

- Shading – Flat
  - Your brain has an amazing ability to fill in what *should* be there – including interpolating color.
  - Flat shading is called ***per primitive*** in WebGL since we are applying a single color value to the entire primitive.



Lighting and Shading

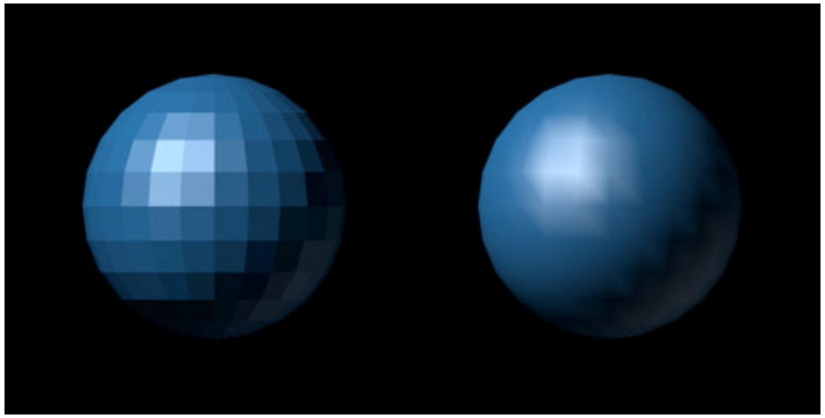
- Shading – Smooth
  - By providing color and normal information ***per vertex*** we can achieve a much smoother shading result.
  - Here we let the rasterizer interpolate the color values across the surface of the primitive. (looks the same)
  - This requires us to think about how to compute the normal for a surface. (sphere in this case, why is the result the same?)



UCLA

Lighting and Shading

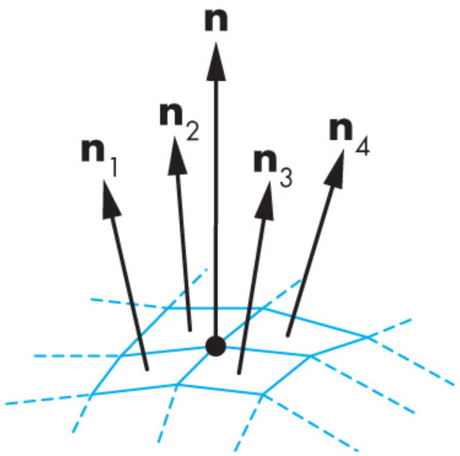
- Shading – Smooth
  - *Gouraud* shading
    - Associate normals per vertex rather than per primitive
    - Normal is the average of the primitives adjacent to the vertex

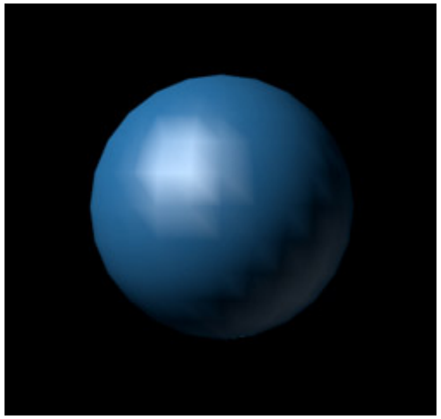


Lighting and Shading

- Shading – Smooth
  - For a vertex that is shared by four surfaces we would compute the vertex normal as an average

$$\mathbf{n} = \frac{\mathbf{n}_1 + \mathbf{n}_2 + \mathbf{n}_3 + \mathbf{n}_4}{|\mathbf{n}_1 + \mathbf{n}_2 + \mathbf{n}_3 + \mathbf{n}_4|}.$$

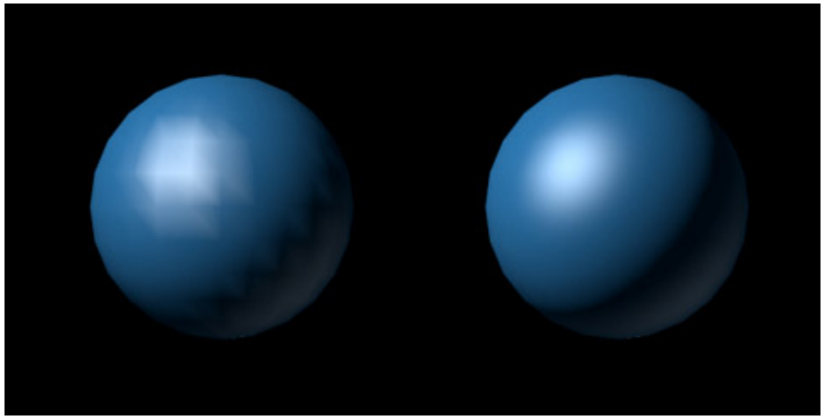




UCLA

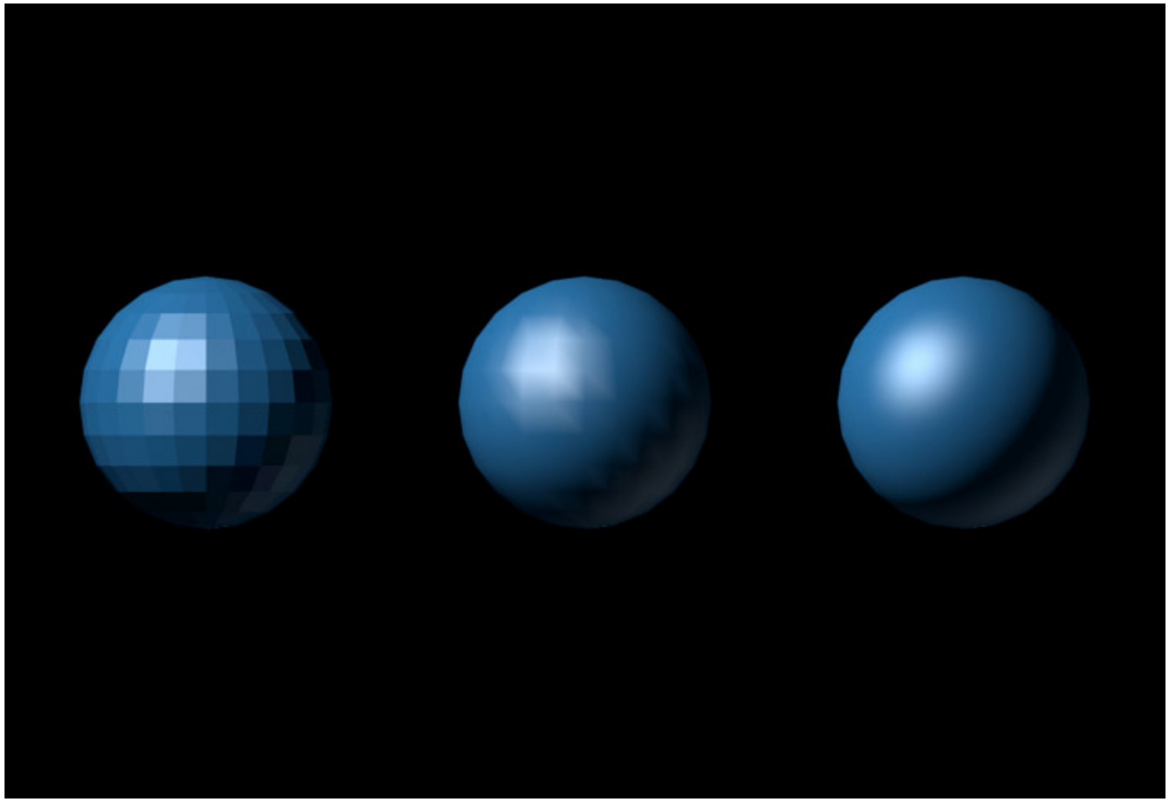
Lighting and Shading

- Shading – Smooth
  - ***Phong shading*** takes Gouraud shading one step further.
  - Interpolate the normals over the surface.
  - Phong shading requires us to compute lighting at the ***fragment*** level. Big difference though (GPU to the rescue)
  - Don’t be confused by Phong Lighting and Phong Shading!
    - He was a busy guy in the early 70’s



Lighting and Shading

- Shading – Flat, Gourand, Phong (Phong/Blinn)



Lighting and Shading

- Coding Lighting
  - You will need a new ‘attribute’ – normals
    - *n(i, j, k)*
    - These are vectors of floats
  - Like vertices you can derive them:
    - By hand
    - Algorithmically
  - How you derive them will depend on the lighting you wish to perform
    - Per primitive (triangle)
    - Per vertex

Lighting and Shading

- Coding Lighting
  - Make sure your vectors are pointing in the direction they need to be.
    - Directional lights often trip people up
      - We define them in the direction 'light travels'
      - Use them from surface to light (reversed)
  - Normalize your vectors
    - Can lead to problems that are hard to understand

Lighting and Shading

- Coding Lighting
  - Normals have to be transformed
    - need to reflect associated object transformations
  - Can't I use the model view matrix?
    - Maybe sort of
    - Model view matrix is 4x4, normals are not homogeneous
      - Add a zero? Add a one?
    - This isn't the issue, really

Lighting and Shading

- Coding Lighting
  - If the model view matrix has a shear (asymmetric scale)
    - The normal vector will get skewed and will no longer be orthogonal to the tangent of the surface
  - Even if there is a symmetric scale, problem
    - Now every normal has to be re-normalized
  - We also do not need the translation embedded into the model view matrix
    - With an orthogonal model view matrix it would work but you are doing a lot of extra arithmetic

UCLA

Lighting and Shading

- Coding Lighting
  - What always works?
    - This does: $G = (M^{-1})^T$
      - M = upper 3x3 of model view matrix
      - G = will properly transform normals
- When you look on the internet you will find as many ways of coding lighting as you will find examples. Unless you try to understand what is going on it will be hard to make sense of any of it – and you will be randomly trying things until something works.
  - That's just frustrating and boring ☹

Lighting and Shading

- Coding Lighting
  - How do you get: $G = (M^{-1})^{\mathrm{T}}$
      - M = upper 3x3 of model view matrix
    - $N \cdot T = 0$
    - $N' \cdot T' = (GN) \cdot (MT) = 0$
    - $(GN) \cdot (MT) = (GN)^{\mathrm{T}} * (MT)$
    - $(GN)^{\mathrm{T}} \cdot (MT) = N^{\mathrm{T}} G^{\mathrm{T}} MT$
    - If $G^{\mathrm{T}} M = I$, then $\mathrm{N}' \cdot T' = N \cdot T = 0$
    - $G^{\mathrm{T}} M = I \Leftrightarrow G = (M^{-1})^{\mathrm{T}}$
  - *If M is orthogonal we can use it directly*
    - $M^{-1} = M^{\mathrm{T}} \Rightarrow G = M$