Computer graphics

lecture 9 – Shading

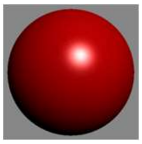
Zbigniew Szymański
zbigniew.szymanski@pjwstk.edu.pl

December 2020 - v1

1

Shading

- Goal: defining colors of pixels belonging to surfaces of objects
- Shading refers to the depiction of depth perception in 3D models or illustrations by varying the level of darkness. Shading tries to approximate local behavior of light on the object's Surface.

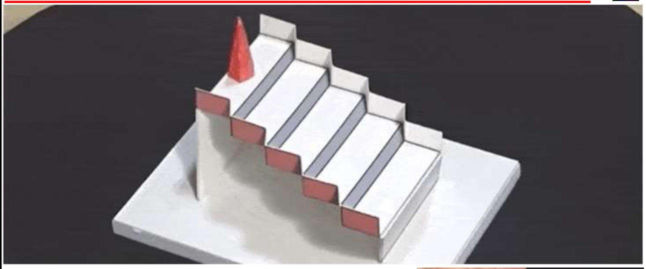


source: https://en.wikipedia.org/wiki/Shading

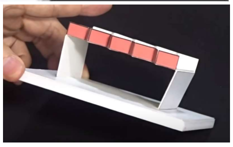
- Shading methods:
  - diffuse (flat) shading – developed in 1970s – available in most graphics libraries
  - vertex based diffuse shading (smooth) – developed in 1970s – available in most graphics libraries
  - phong shading – developed in 1970s – available in most graphics libraries
  - artistic shading

2

Shading influences perception of depth and shape



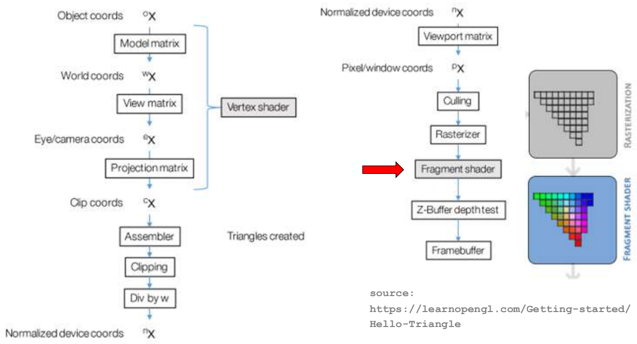
source: https://www.sciencealert.com/brain-bending-3d-staircase-wins-best-illusion-of-the-year-for-2020



source: https://youtu.be/5DYeAkx2IBo

3

Shading in graphics pipeline



source: https://learnopengl.com/Getting-started/Hello-Triangle

source: https://graphicscompendium.com/opengl/24-clipping-culling

4

Diffuse (flat) shading

- This method can be applied to objects described as "matte," indicating that the object is not at all shiny - such objects do not have a color change with a change in viewpoint.
  - Examples: paper, unfinished wood, dry, unpolished stones.

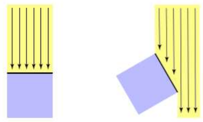
source: https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-shading/what-is-shading-light-matter-interaction

- Matte objects obey Lambert's cosine Law

5

Lambertian shading model

- Lambert's cosine law - the color c of a surface is proportional to the cosine of the angle between the surface normal and the direction to the light source



source: http://www.cs.cornell.edu/courses/cs4620/2010fa/lectures/09shadingBasics.pdf

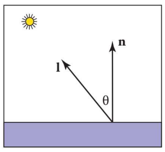


Figure 10.1. The geometry for Lambert's law. Both **n** and **l** are unit vectors.

source: Marschner, Steve; Shirley, Peter. Fundamentals of Computer Graphics (p. 234). CRC Press. Kindle Edition.

$$c \propto \cos\theta$$

$$c \propto \boldsymbol{n} \cdot \boldsymbol{l}$$

- The color on the surface will vary according to the cosine of the angle between the surface normal and the light direction.
- The vector l is typically assumed not to depend on the location of the object. That assumption is equivalent to assuming the light is "distant" relative to object size.

6

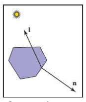
Lambertian shading model /2/

- A surface can be made lighter or darker by changing:
  - the intensity of the light source
  - the reflectance of the surface.
- The diffuse reflectance $c_r \in [0,1]$ - the fraction of light reflected by the surface.
  - This fraction will be different for different color components.
  - For example, a surface is red if it reflects a higher fraction of red incident light than blue incident light.
- Surface color is proportional to the light reflected from a surface

$$c \propto c_r(\boldsymbol{n} \cdot \boldsymbol{l})$$

- The effects of light intensity $c_l \in [0,1]$

$$c \propto c_r c_l(\boldsymbol{n} \cdot \boldsymbol{l})$$



source: Marschner, Steve; Shirley, Peter. Fundamentals of Computer Graphics (p. 234). CRC Press. Kindle Edition.

- The dot product is negative when the surface is pointing away from the light – to keep result in [0,1]: $\mathbf{c} = \boldsymbol{c}_r \circ \boldsymbol{c}_l \max(0, \boldsymbol{n} \cdot \boldsymbol{l})$

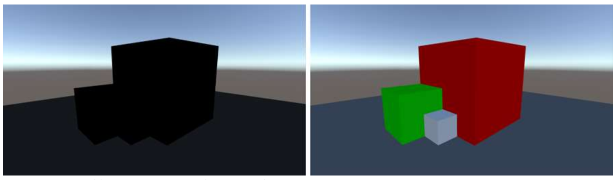
Diffuse (flat) shading - result

source: https://en.wikipedia.org/wiki/Shading

Ambient shading

- When using diffuse shading any point whose normal vector faces away from the light (or there is no light) will be black.
- Solution to this problem is to add an ambient term c_a to lighting equation: $\mathbf{c} = \boldsymbol{c}_r \circ (\boldsymbol{c}_a + \boldsymbol{c}_l \max(0, \boldsymbol{n} \cdot \boldsymbol{l}))$
- To ensure that the computed RGB color stays in the range [0, 1]

$$(\boldsymbol{c}_a + \boldsymbol{c}_l) \leq \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$



source: https://learn.unity.com/tutorial/introduction-to-lighting-and-rendering

Vertex based diffuse (Gouraud) shading

- Drawback of diffuse (flat) shading: an object made up of triangles will typically have a faceted appearance, because the triangles are an approximation of a smooth surface.
- Solution: place surface normal vectors at the vertices of the triangles. This will give a color at each triangle vertex, and this color can be interpolated using the barycentric interpolation:

$$\mathbf{c} = \boldsymbol{c}_r \circ (\boldsymbol{c}_a + \boldsymbol{c}_l \max(0, \boldsymbol{n} \cdot \boldsymbol{l}))$$

- Normal vetors at vertices can be computed by averaging the normals of the triangles that share each vertex and normalize it to unit vectors.

source: Marschner, Steve; Shirley, Peter. Fundamentals of Computer Graphics (p. 234). CRC Press. Kindle Edition.

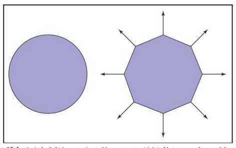
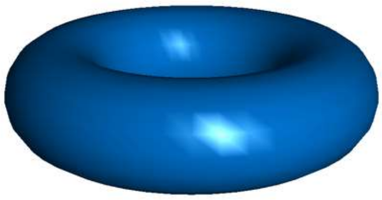


Figure 10.4. A circle (left) is approximated by an octagon (right). Vertex normals record the surface normal of the original curve.

Diffuse vertex based shading - result



source: https://en.wikipedia.org/wiki/Shading

Blinn-Phong shading

- Some surfaces are essentially like matte surfaces, but they have highlights.
  - Examples: polished tile floors, gloss paint, and whiteboards.
- Highlights move across a surface as the viewpoint moves. This means that we must add a unit vector **e** toward the eye into our equations.
- If you look carefully at highlights, you will see that they are really reflections of the light; sometimes these reflections are blurred.
- The color of these highlights is the color of the light—the surface color seems to have little effect.

source: http://www.mdavid.com.au/photography/specularhighlights.shtml

Computing highlight color based on reflection vector

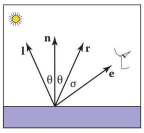


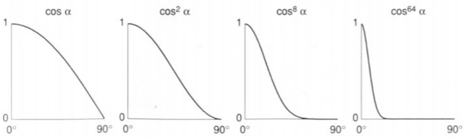
Figure 10.5. The geometry for the Phong illumination model. The eye should see a highlight if σ is small.

source: Marschner, Steve; Shirley, Peter. Fundamentals of Computer Graphics (p. 234). CRC Press. Kindle Edition.

- When r is given, we'd like a heuristic function that is bright when **e** equals **r** and falls off gradually when **e** moves away from **r** (cosine)

$$\mathbf{c} = \boldsymbol{c_l} \max(0, \boldsymbol{e} \cdot \boldsymbol{r})^p$$

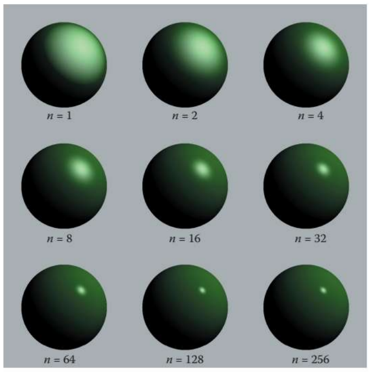
  - $\mathbf{c}_l$ – rgb light color, each component in the range [0, 1]
  - **e** – direction to the eye (unit vector)
  - **l** – direction to the light source (unit vector)
  - **r** – reflection vector (unit vector)
  - **n** – normal vector (unit vector)
  - p – Phong exponent



source: http://www.cs.cornell.edu/courses/cs4620/2010fa/lectures/09shadingBasics.pdf

13

Influence of Phong exponent on the highlight



source: Marschner, Steve; Shirley, Peter. Fundamentals of Computer Graphics (p. 234). CRC Press. Kindle Edition.

14

Computation of the reflection and bisector vectors

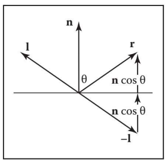


Figure 10.7. The geometry for calculating the vector **r**.

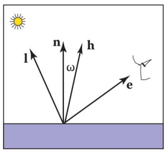


Figure 10.8. The unit vector **h** is halfway between **l** and **e**.

- Geometric calculation of **r**

$$\mathbf{r} = -\boldsymbol{l} + 2(\boldsymbol{l} \cdot \boldsymbol{n})\boldsymbol{n}$$

  - $\cos\theta$ may be negative - need to check for negative values, because they corrupt further calculations
- Heuristic - calculation of bisector **h** - the unit vector halfway between **l** and **e**

$$\mathbf{h} = \frac{\boldsymbol{e} + \boldsymbol{l}}{\|\boldsymbol{e} + \boldsymbol{l}\|}$$

  - when ω is small → **e** is close to **r** (look at fig. 10.7 and imagine **e** at the same place as **r**)
  - instead of using $\boldsymbol{e} \cdot \boldsymbol{r}$ we may also use $\boldsymbol{h} \cdot \boldsymbol{n}$
  - cos ω is positive for eye and light above the plane – no need to check for negative values

source: Marschner, Steve; Shirley, Peter. Fundamentals of Computer Graphics (p. 234). CRC Press. Kindle Edition.

15

Computing highlight color based on bisector vector

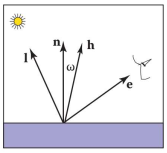


Figure 10.8. The unit vector **h** is halfway between **l** and **e**.

source: Marschner, Steve; Shirley, Peter. Fundamentals of Computer Graphics (p. 234). CRC Press. Kindle Edition.

- The formula describing the highlight

$$\mathbf{c} = \boldsymbol{c_l}(\boldsymbol{h} \cdot \boldsymbol{n})^p$$

  - $\mathbf{c}_l$ – rgb light color, each component in the range [0, 1]
  - **h** – halfway vector between **l** and **e** (unit vector)
  - **l** – direction to the light source (unit vector)
  - **e** – direction to the viewer (unit vector)
  - **n** – normal vector (unit vector)
  - p – Phong exponent
- Final lighting equation including diffuse and specular component

$$\mathbf{c} = \boldsymbol{c_r} \circ (\boldsymbol{c_a} + \boldsymbol{c_l} \max(0, \boldsymbol{n} \cdot \boldsymbol{l})) + \boldsymbol{c_l}(\boldsymbol{h} \cdot \boldsymbol{n})^p$$

source: http://www.cs.cornell.edu/courses/cs4620/2010fa/lectures/09shadingBasics.pdf

16

Surface normal vector interpolation

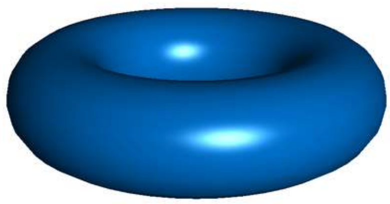
- On smooth surfaces with highlights the color may change very quickly.
- To avoid artifacts the triangles should be very small (when a single color is assigned to one triangle) - the performance of graphics pipeline would be very low.
- Another solution is to interpolate normal vectors for each pixel and then applying Phong shading at each pixel.

$$\mathbf{c} = \boldsymbol{c_r} \circ (\boldsymbol{c_a} + \boldsymbol{c_l} \max(0, \boldsymbol{n} \cdot \boldsymbol{l})) + \boldsymbol{c_l}(\boldsymbol{h} \cdot \boldsymbol{n})^p$$

$$\mathbf{n} = \alpha \boldsymbol{n_0} + \beta \boldsymbol{n_1} + \gamma \boldsymbol{n_2}$$

17

Phong shading - result



source: https://en.wikipedia.org/wiki/Shading

18

Example fragment shader /1/

```
#version 330 core
in vec3 vecColor;
in vec2 TexCoord;
in vec3 Normal;
in vec3 FragPos;

out vec4 color;

uniform sampler2D Texture0;
uniform sampler2D Texture1;

uniform vec3 lightColor;
uniform vec3 lightPos;
uniform vec3 viewPos;

void main()
{
...
```

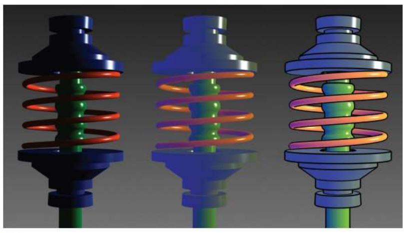

19

Example fragment shader /2/

```
...
void main()
{
float ambientStrength = 0.1;
vec3 ambient = ambientStrength * lightColor;

vec3 norm = normalize(Normal);
vec3 lightDir = normalize(lightPos - FragPos);
float diff = max(dot(norm, lightDir), 0.0);
vec3 diffuse = diff * lightColor;

float specularStrength = 0.5;
vec3 viewDir = normalize(viewPos - FragPos);
vec3 reflectDir = reflect(-lightDir, norm);
float spec = pow(max(dot(viewDir, reflectDir), 0.0), 32);
vec3 specular = specularStrength * spec * lightColor;

color = vec4(ambient+ diffuse+specular,1.0)* texture(Texture0,
TexCoord);
}
```


20

Artistic shading



source: Marschner, Steve; Shirley, Peter. Fundamentals of Computer Graphics (p. 234). CRC Press. Kindle Edition.

Figure 10.9. Left: a Phong-illuminated image. Middle: cool-to-warm shading is not useful without silhouettes. Right: cool-to-warm shading plus silhouettes. *Image courtesy Amy Gooch.*

- Drawing silhouettes - draw an edge as a silhouette when one of the two triangles sharing an edge faces toward the viewer, and the other triangle faces away from the viewer (**e** – vector towards the viewer)

$$(\boldsymbol{e} \cdot \boldsymbol{n}_0)(\boldsymbol{e} \cdot \boldsymbol{n}_1) \leq 0$$

21