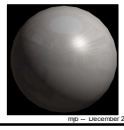
1

Using Shaders for Lighting

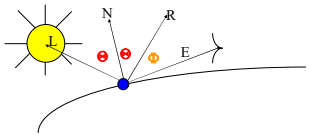
Oregon State University
Mike Bailey
mjb@cs.oregonstate.edu



2

Lighting Definitions

N = Normal vector
L = Vector from Point to the Light
R = Light reflection vector
E = Vector from the Point to the eye



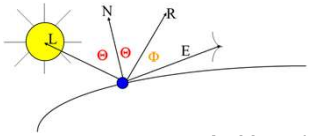
Ambient = Light intensity that is "everywhere"
Diffuse = Light intensity proportional to cos(Θ)
Specular = Light intensity proportional to $\cos^S(\Phi)$
A-D-S = Lighting model that includes Ambient, Diffuse, and Specular

Flat Interpolation = Use a single polygon normal to compute one A-D-S for the entire polygon
Smooth Interpolation = Use a normal at each vertex to compute one A-D-S at each vertex

Per-fragment lighting = Interpolate the vectors across the entire polygon and then compute A-D-S at each fragment

3

A-D-S Lighting

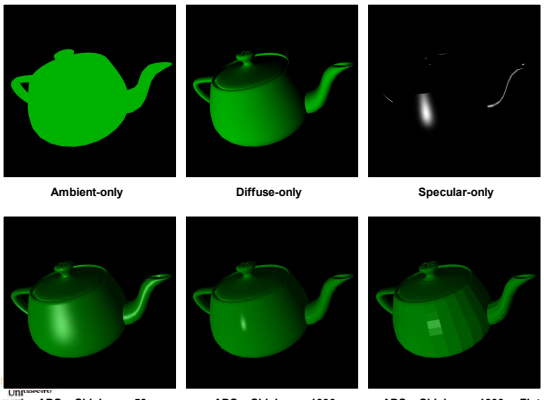


Ambient: K_a

Diffuse: K_d*cosθ

Specular: K_s*$\cos^s$φ

4

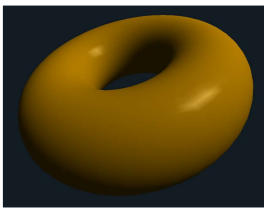


Ambient-only | Diffuse-only | Specular-only

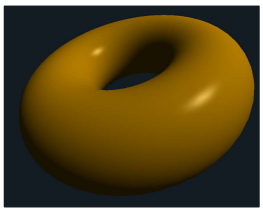
ADS – Shininess=50 | ADS – Shininess=1000 | ADS – Shininess=1000 -- Flat

5

The Difference Between Per-Vertex Lighting and Per-Fragment Lighting



Per-vertex



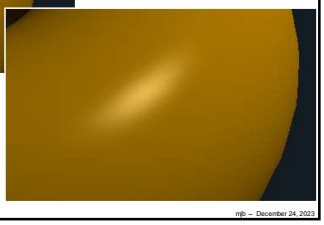
Per-fragment

6

The Difference Between Per-Vertex Lighting and Per-Fragment Lighting

Per-vertex

Per-fragment



Applying Per-Fragment Lighting

7

Vertex shader:

```
#version 330 compatibility
out vec2   vST;                    // texture coords
out vec3   vN;                     // normal vector
out vec3   vL;                     // vector from point to light
out vec3   vE;                     // vector from point to eye

const vec3 LIGHTPOSITION = vec3(  5., 5., 0. );

void
main( )
{
        vST = gl_MultiTexCoord0.st;
        vec4 ECposition = gl_ModelViewMatrix * gl_Vertex;          // eye coordinate position
        vN = normalize( gl_NormalMatrix * gl_Normal );   // normal vector
        vL = LIGHTPOSITION - ECposition.xyz;                       // vector from the point to the light position
        vE = vec3( 0., 0., 0. ) - ECposition.xyz;                  // vector from the point to the eye position
        gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
}
```

Oregon State University Computer Graphics

Applying Per-Fragment Lighting

8

Fragment shader:

```
#version 330 compatibility
uniform float   uKa, uKd, uKs;                    // coefficients of each type of lighting
uniform float   uShininess;                       // specular exponent
in vec2   vST;                                    // texture cords
in vec3   vN;                                     // normal vector
in vec3   vL;                                     // vector from point to light
in vec3   vE;                                     // vector from point to eye
void main( )
{
        vec3 Normal = normalize(vN);
        vec3 Light  = normalize(vL);
        vec3 Eye    = normalize(vE);

        vec3 myColor =            vec3( 1.0, 0.5, 0.0 );      // default color
        vec3 mySpecularColor = vec3( 1.0, 1.0, 1.0 );         // specular highlight color

        << possibly change myColor >>

        vec3 ambient = uKa * myColor;
        float d = 0.;
        float s = 0.
        if( dot(Normal,Light) > 0. )      // only do specular if the light can see the point
        {
                d = dot(Normal,Light);
                vec3 ref = normalize(  reflect( -Light, Normal ) );          // reflection vector
                s = pow( max( dot(Eye,ref),0. ), uShininess );
        }
        vec3 diffuse    = uKd * d * myColor;
        vec3 specular = uKs * s * mySpecularColor;
        gl_FragColor = vec4( ambient + diffuse + specular,  1. );
}
```

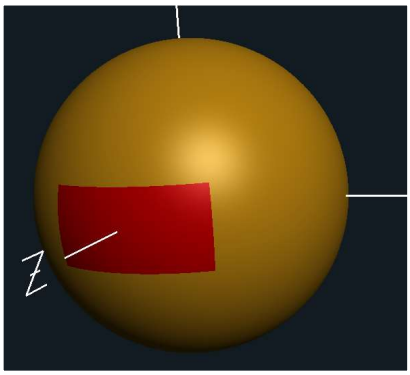
Here's where we figure out what color this fragment will be

Here's where we apply lighting to that color

Oregon State University Computer Graphics

Applying Per-Fragment Lighting

9



Oregon State University Computer Graphics

Per-fragment A-D-S Lighting with Flat Interpolation

10

Each polygon has a single lighting value applied to every pixel within it.

Oregon State University Computer Graphics

Per-fragment A-D-S Lighting with Flat Interpolation

11

Vertex shader:

```
. . .

flat out vec2   vST;          // texture coords
flat out  vec3  vN;           // normal vector
flat out  vec3  vL;           // vector from point to light
flat out  vec3  vE;           // vector from point to eye

. . .
```

Fragment shader:

```
. . .

flat in vec2   vST;           // texture cords
flat in  vec3  vN;            // normal vector
flat in  vec3  vL;            // vector from point to light
flat in  vec3  vE;            // vector from point to eye
. . .
```

Oregon State University Computer Graphics

12

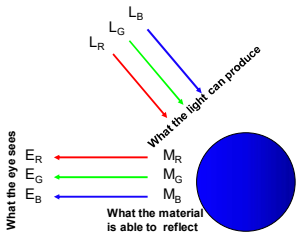
Flat Shading

Smooth Shading

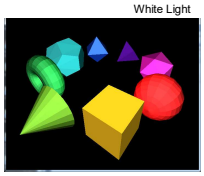
Oregon State University Computer Graphics

What you see depends on the light color and the material color

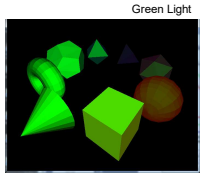
13



White Light



Green Light



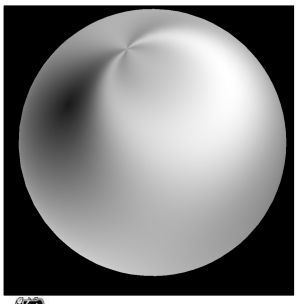
$E_R = L_R * M_R$
$E_G = L_G * M_G$
$E_B = L_B * M_B$

This is how you implement subtractive coloring.

Oregon State University Computer Graphics

A-D-S Anisotropic Lighting with Normal Interpolation

14



Note: The bright spot is not circular because the material has different properties in different directions. Materials such as fur, hair, and brushed metal behave this way.

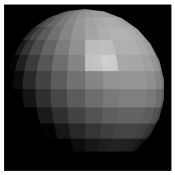
James Kajiya and Timothy Kay, "Rendering Fur with Three Dimensional Textures", *Proceedings of SIGGRAPH 1989*, Volume 23, Number 3, July 1989, pp. 271-280.

Oregon State University Computer Graphics

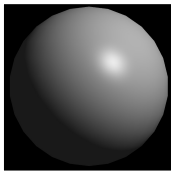
Summary

15

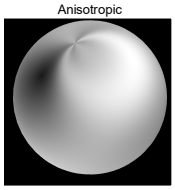
Flat



Smooth



Anisotropic



Oregon State University Computer Graphics

Besides Hair, Other Real World Examples of Anisotropic Lighting Behavior

16

Information Desk

Elevator Doors

Oregon State University Computer Graphics