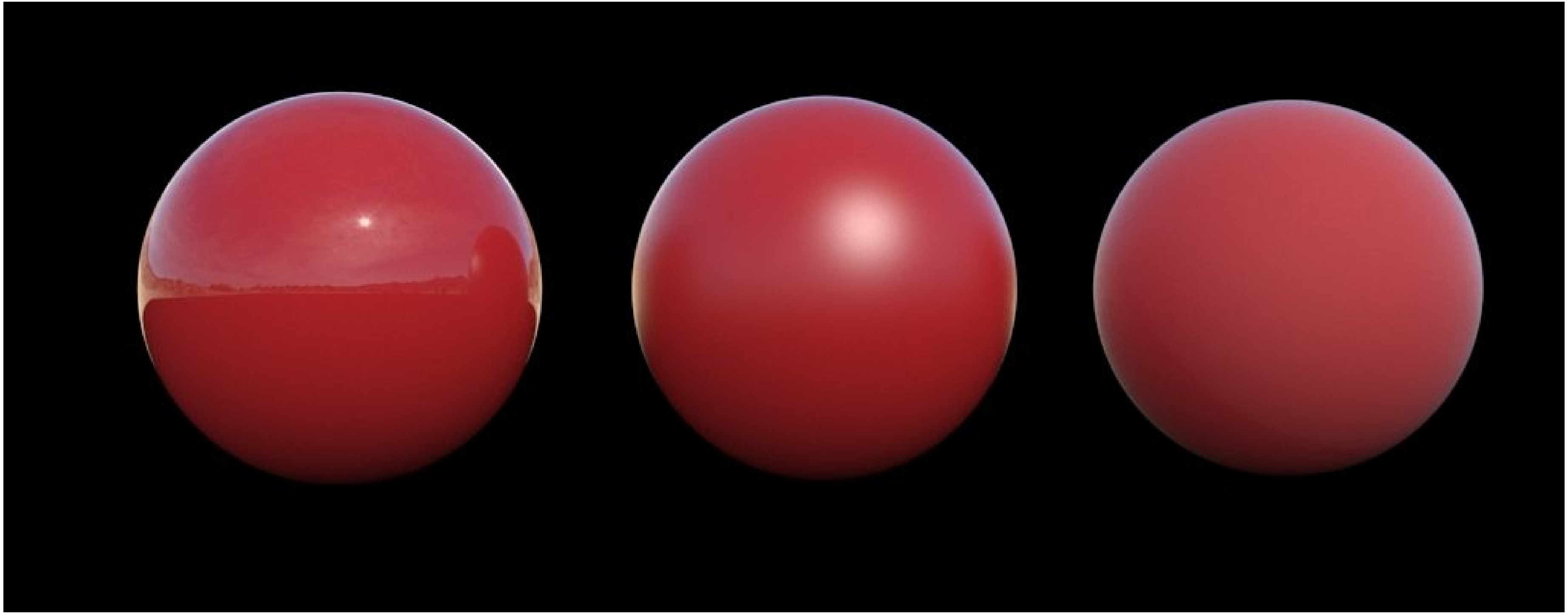
Game Architecture
Graphics IV

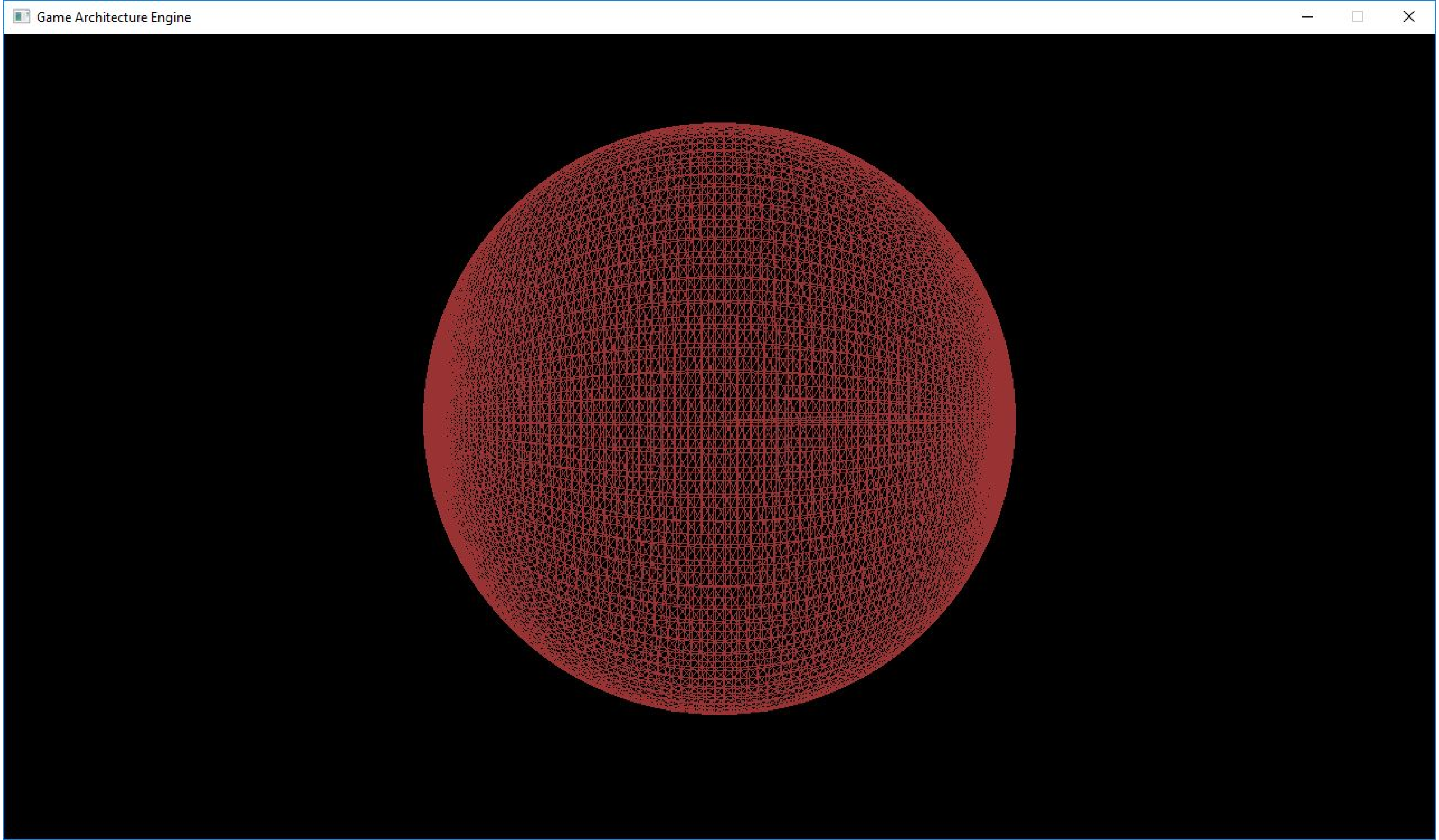
Lighting and Shading

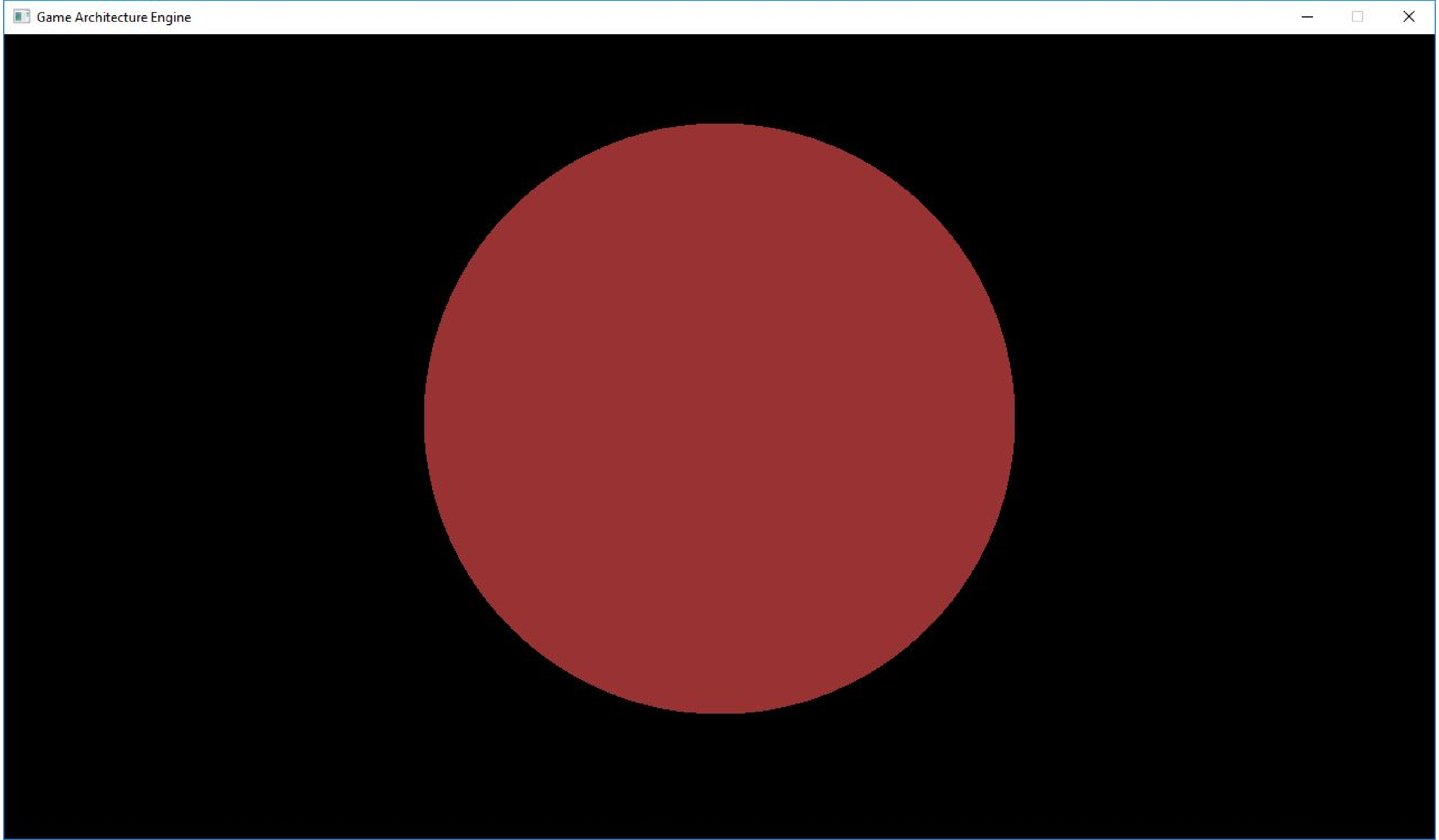
Today’s Agenda

- Background
- Lighting models
- Shading models

Example

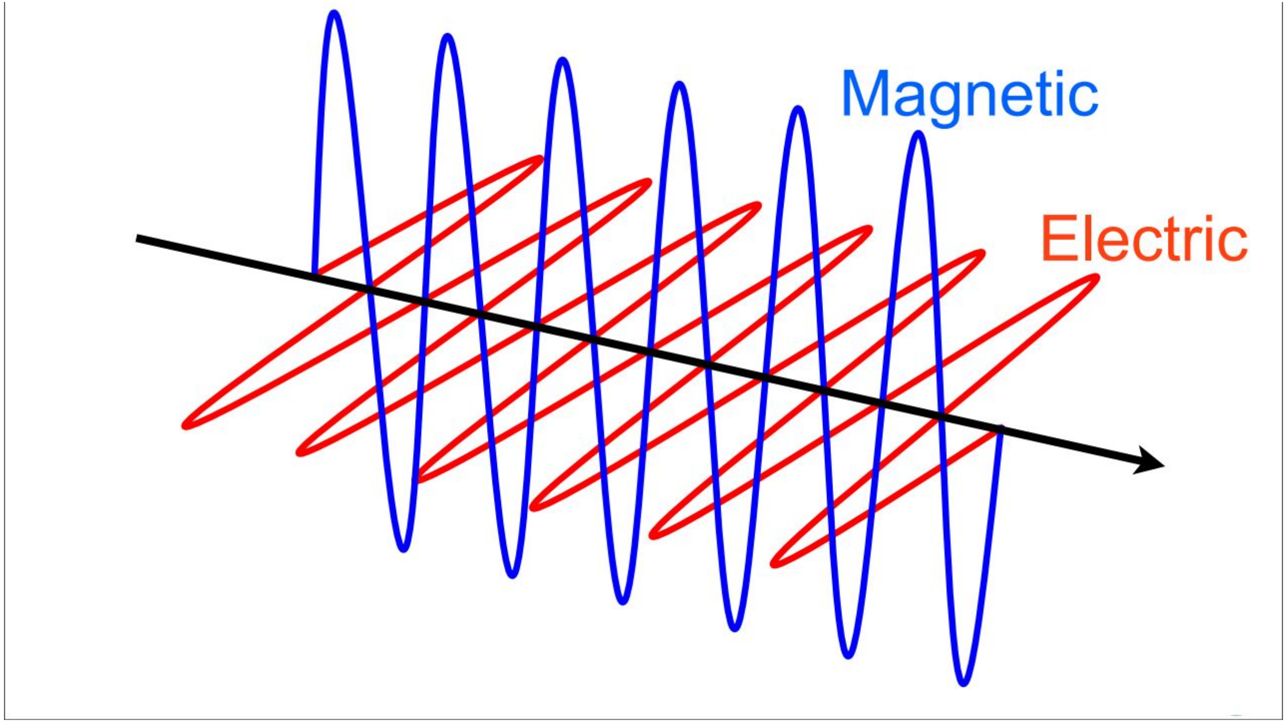



Game Architecture Engine

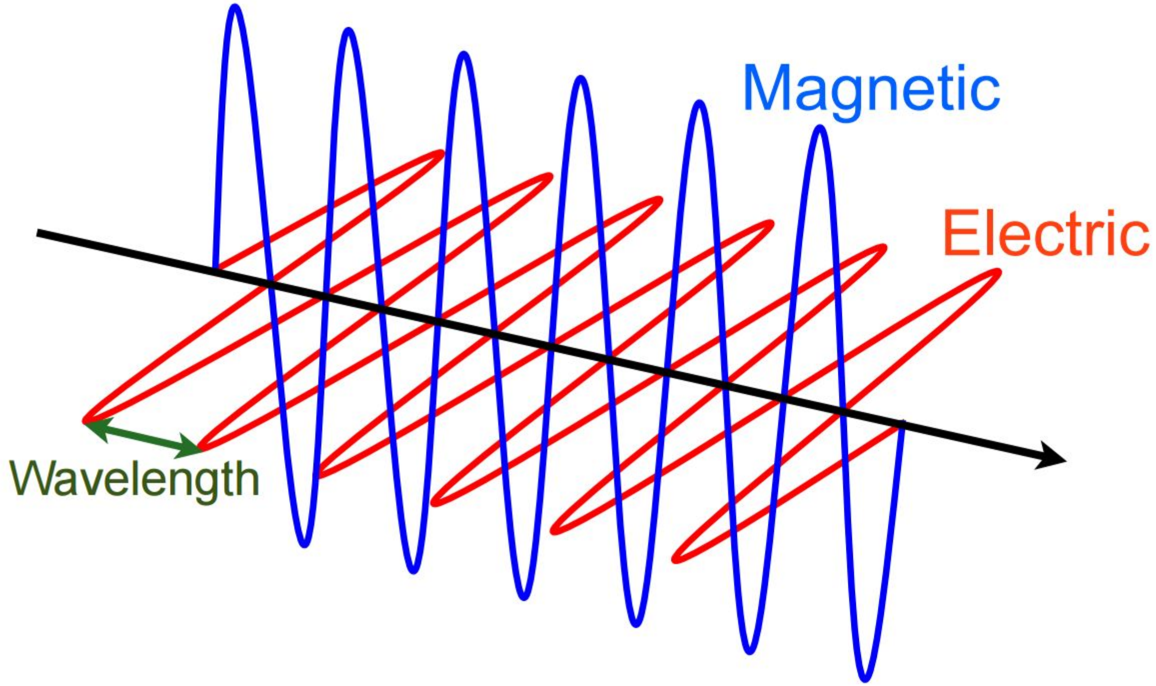

Game Architecture Engine


FORZA
MOTORSPORT
6
PEUGEOT
TOTAL
QUARTZ
BOSCH
P1
24 HEURES DU MANS
2009

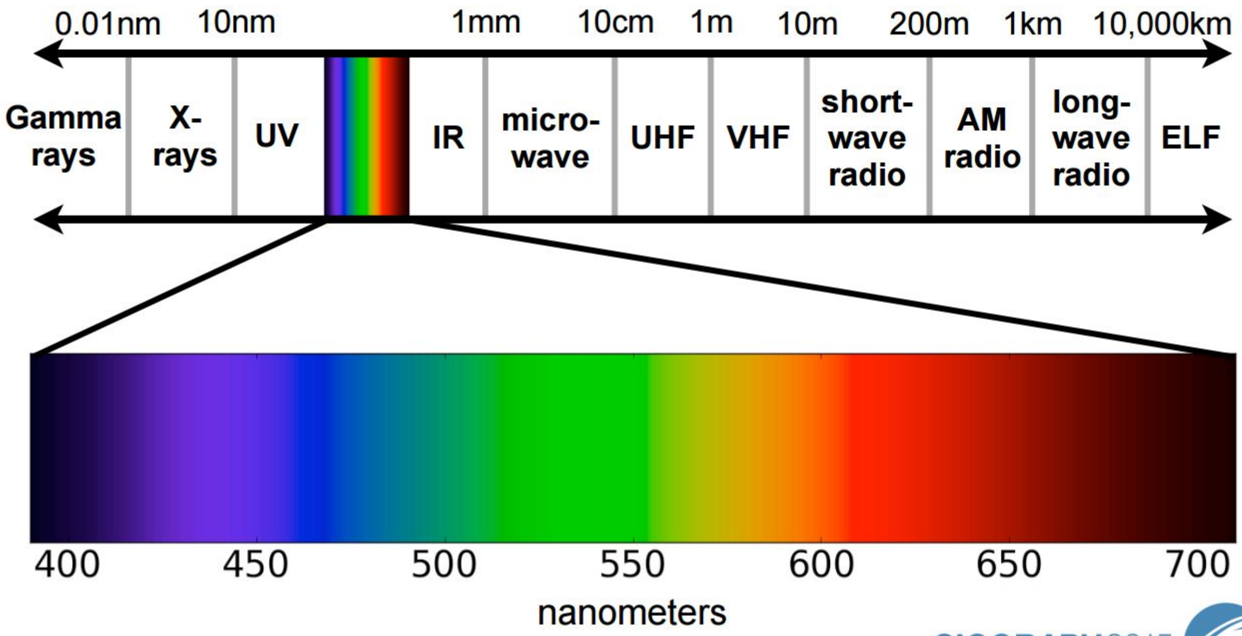
Physics of Light



Physics of Light

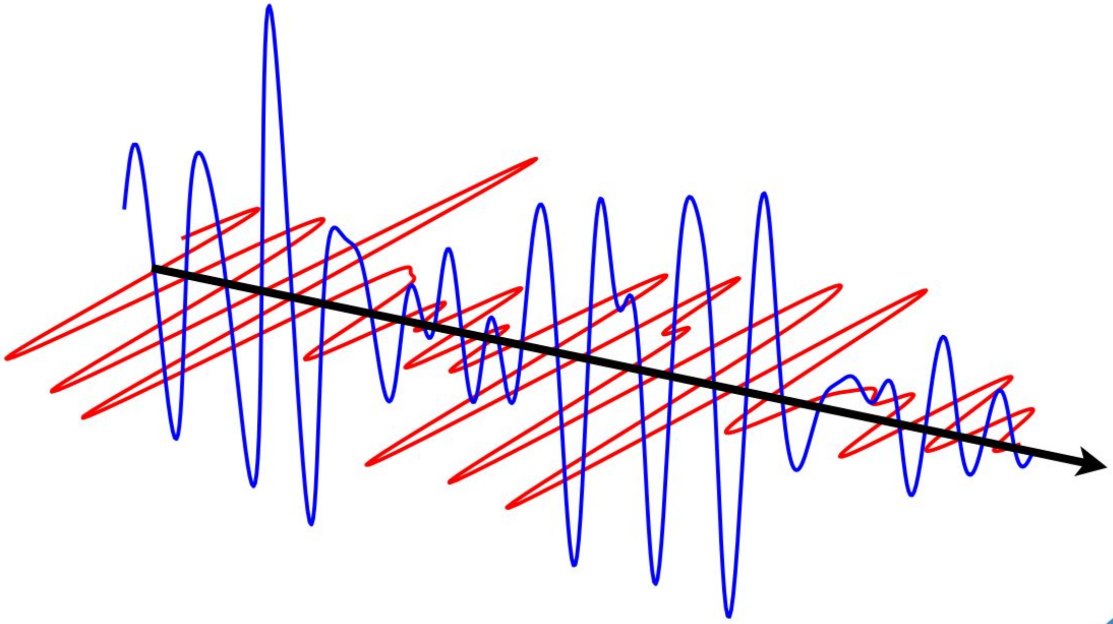


Physics of Light



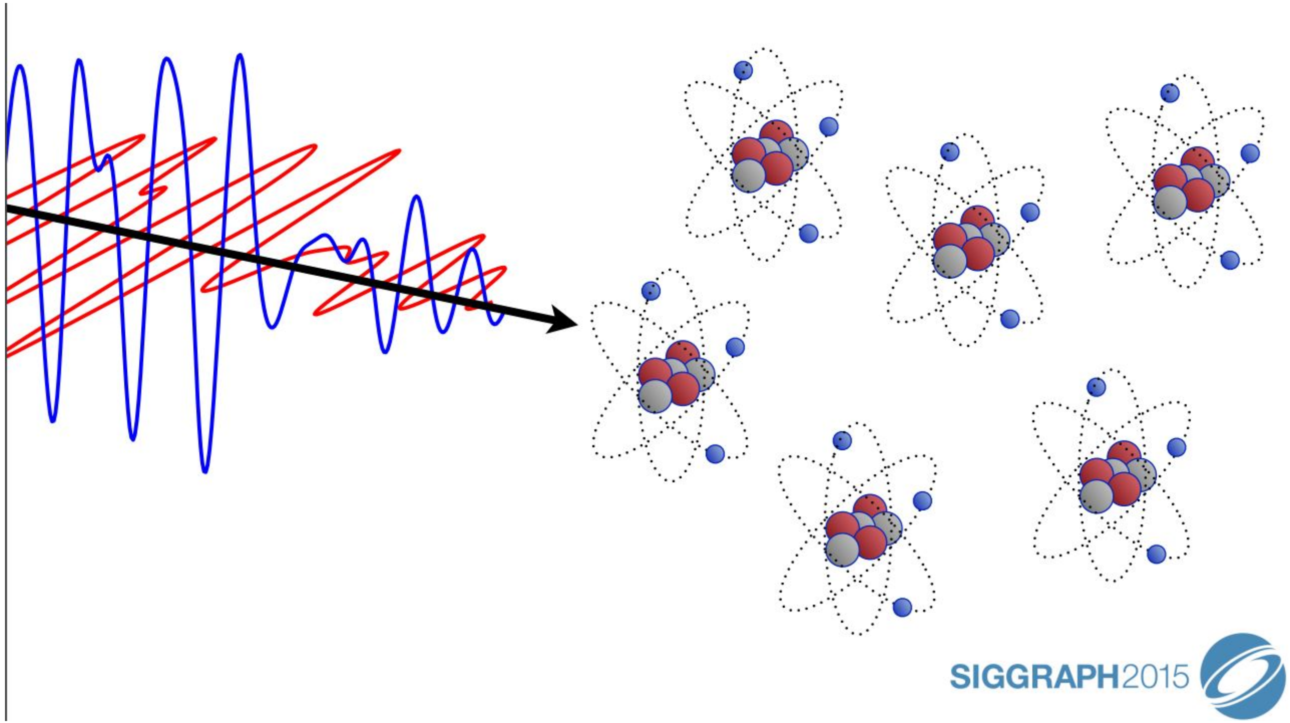
SIGGRAPH2015

Physics of Light



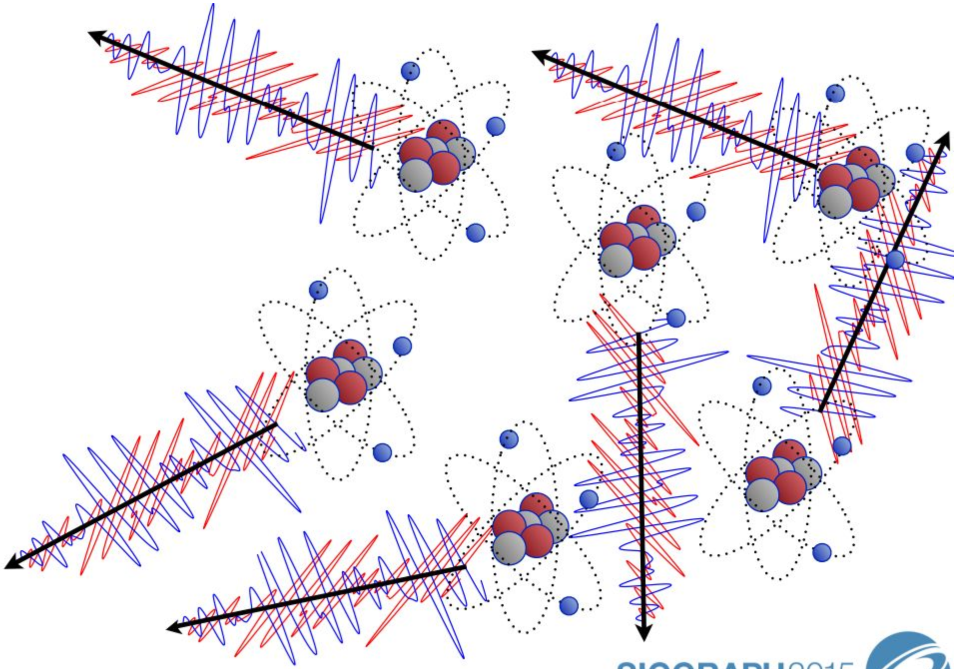
SIGGRAPH2015

Physics of Light



SIGGRAPH2015

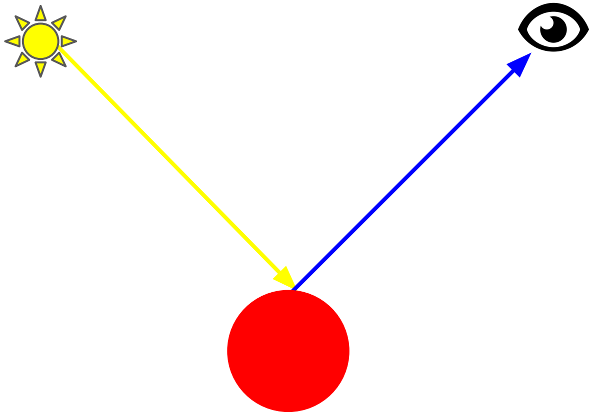
Physics of Light



SIGGRAPH2015

Lighting Model

Given the incident lighting at a point on a surface, what is the reflected light?



In trying to improve the quality of the synthetic images, we do not expect to be able to display the object exactly as it would appear in reality, with texture, overcast shadows, etc. We hope only to display an image that approximates the real object closely enough to provide a certain degree of realism.

- Bui Tuong Phong, 1975

Geometric Optics

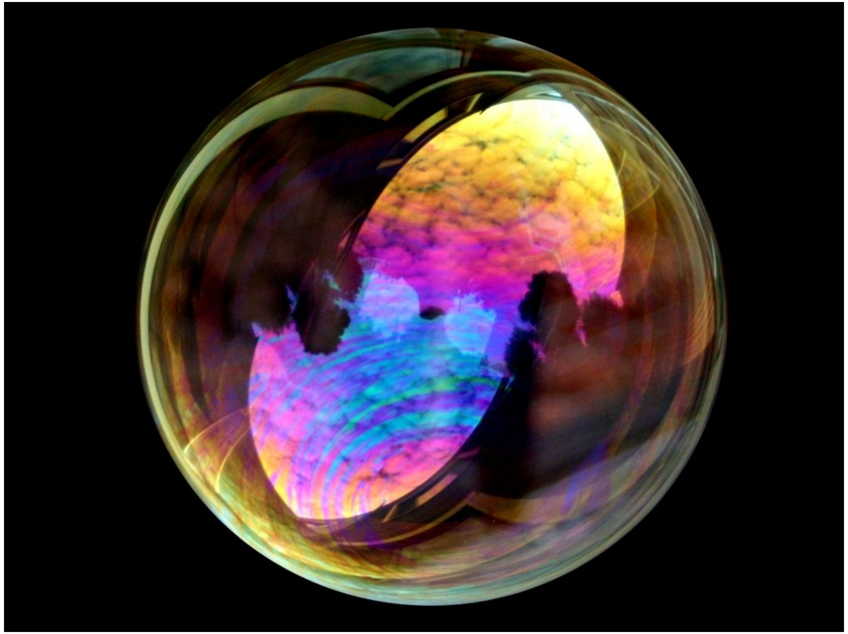
Approximate the behavior of light and its interaction with matter in terms of rays and solid uniformly dense surfaces.

Geometric Optics

Approximate the behavior of light and its interaction with matter in terms of rays and solid uniformly dense surfaces.

- Rays do not interfere with each other.

Interference

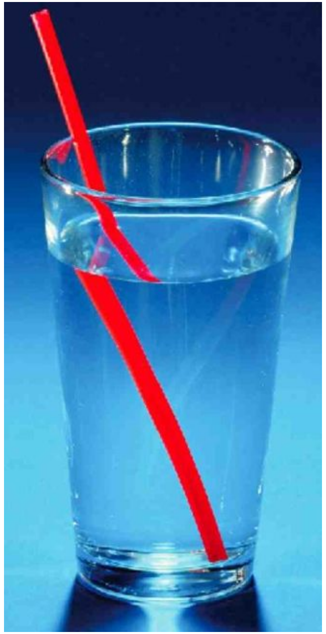


Geometric Optics

Approximate the behavior of light and its interaction with matter in terms of rays and solid uniformly dense surfaces.

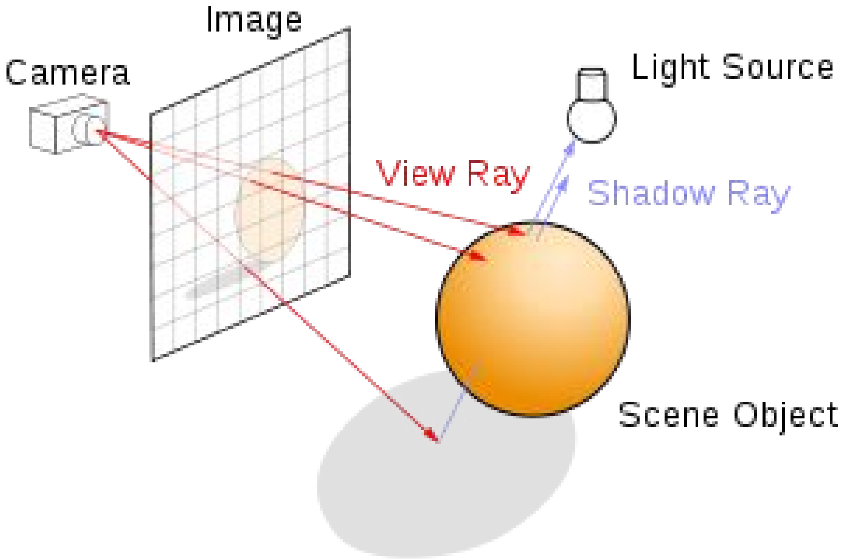
- Rays do not interfere with each other.
- Reflection and refraction only happen at surface boundaries.

Refraction



Raytracing

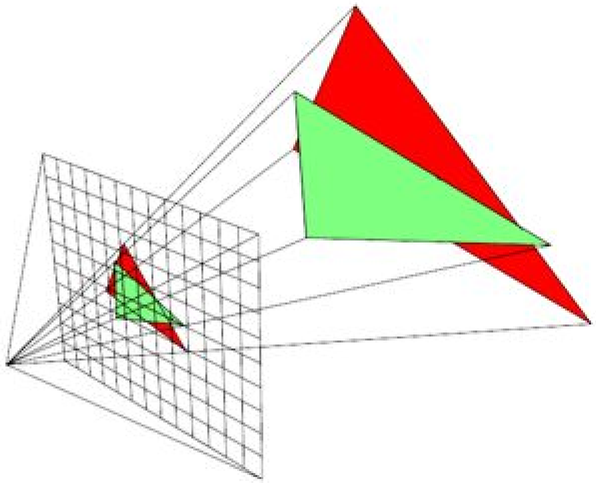
Tracing the path of light through pixels in an image plane and simulating the effects of its intersections with a surface.



Raytracing

Ray tracing is the future and ever will be.
 - Alexander Keller, NVIDIA Research

Rasterization

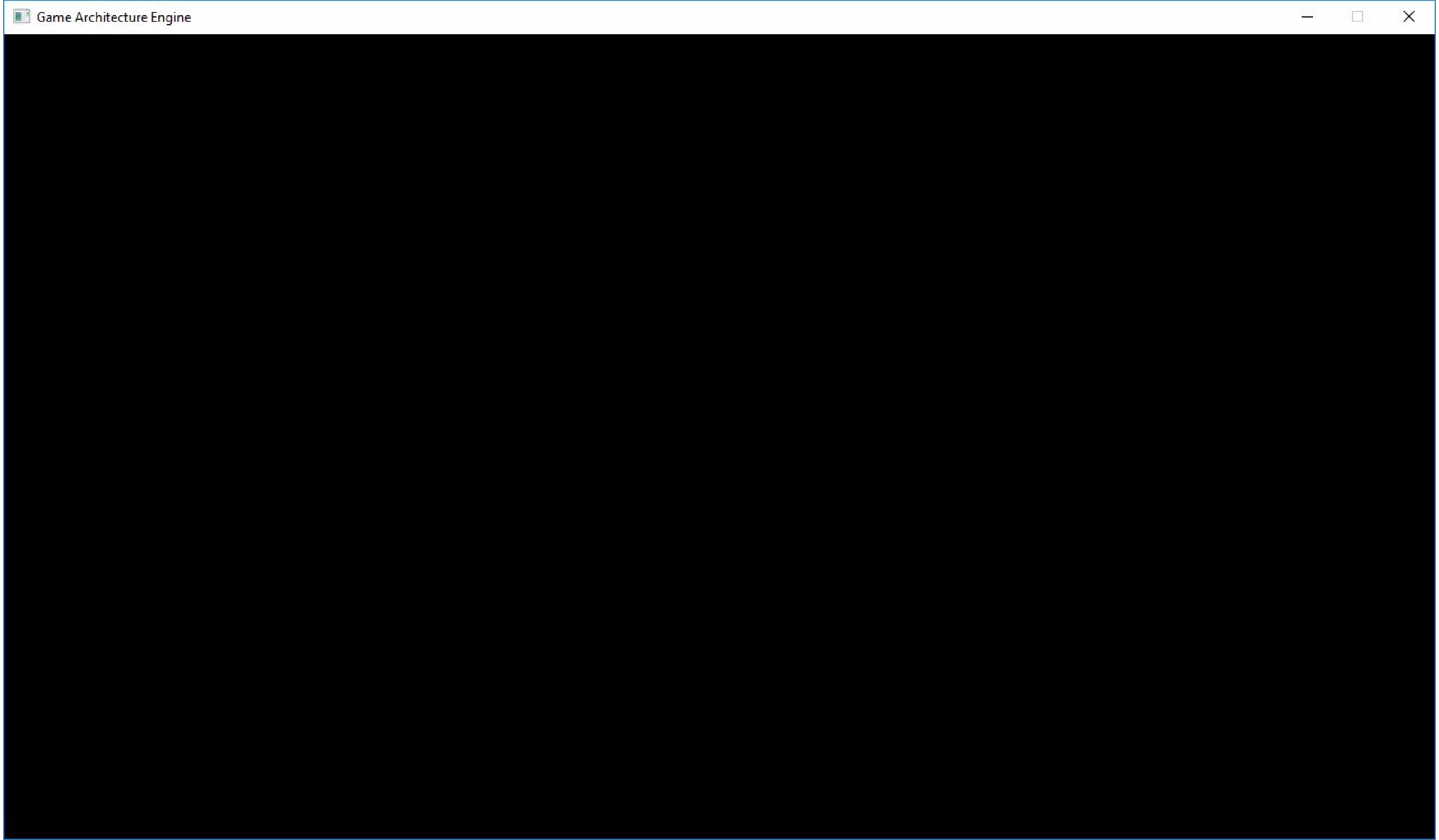


Lighting Model

Given the incident lighting at a point on a surface, what is the reflected light?

Lighting Model

```
vec3 get_color()
{
    return vec3(0.6, 0.2, 0.2);
}
```


Game Architecture Engine

Lighting Model

```
vec3 get_color()
{
    return vec3(0);
}
```

```
struct light
{
    vec3 _position;
    shape _shape;
    float _wavelength;
    float _amplitude;
};
```

Lighting Model

```
vec3 get_color()
{
    return vec3(0);
}
```

```
struct light
{
    vec3 _position;
    shape _shape;
    vec3 _color;
};
```

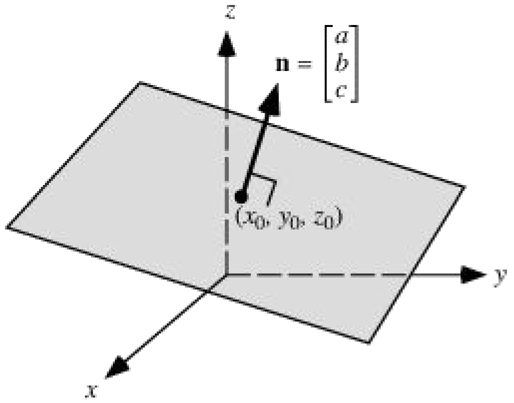
Lighting Model

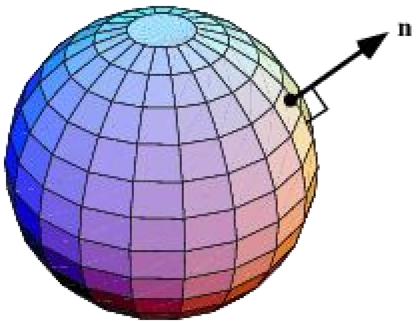
```
vec3 get_color(
    point_light light)
{
    return vec3(0);
}
```

```
struct point_light
{
    vec3 _position;
    vec3 _color;
};
```

Lighting Model

```
vec3 get_color(
    surface surface,
    point_light light)
{
    return vec3(0);
}
```






```
struct surface
{
    vec3 _position;
    vec3 _normal;
};
```

Lighting Model

```
vec3 get_color(
    surface surface,
    point_light light)
{
    return vec3(0);
}
```

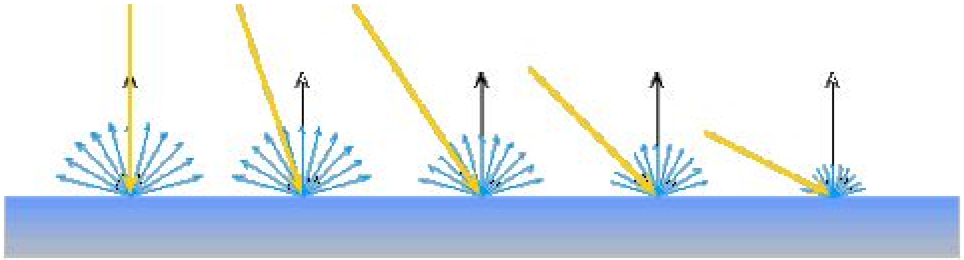
```
struct material
{
    vec3 _color;
};

struct surface
{
    vec3 _position;
    vec3 _normal;
    material _material;
};
```

Lighting Model

```
vec3 get_color(
    surface surface,
    point_light light)
{
    return vec3(0);
}
```

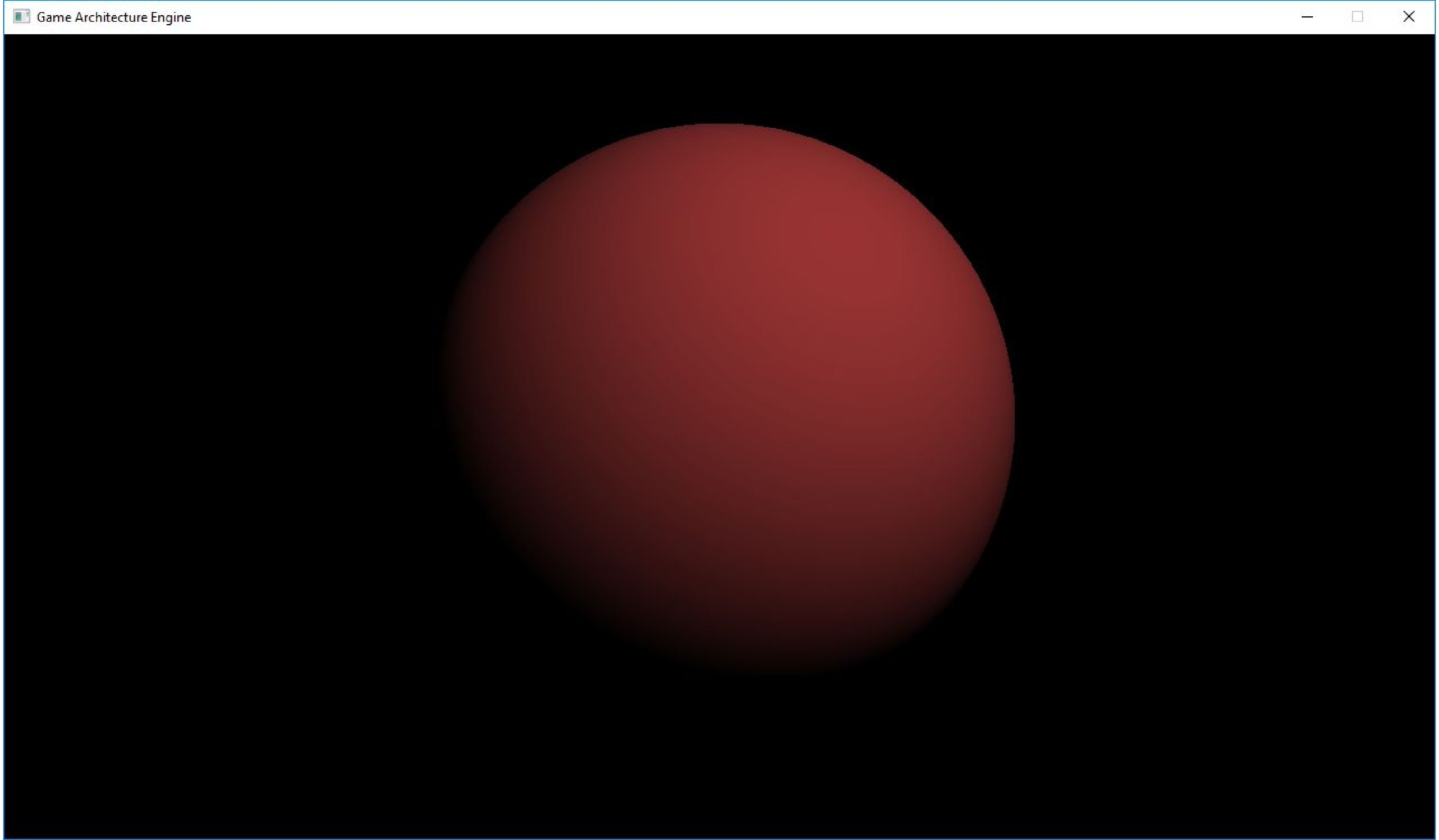
Lambert’s Cosine Law

The reflected energy from a surface in any direction is proportional to the cosine of the angle between the incident light and the surface normal.



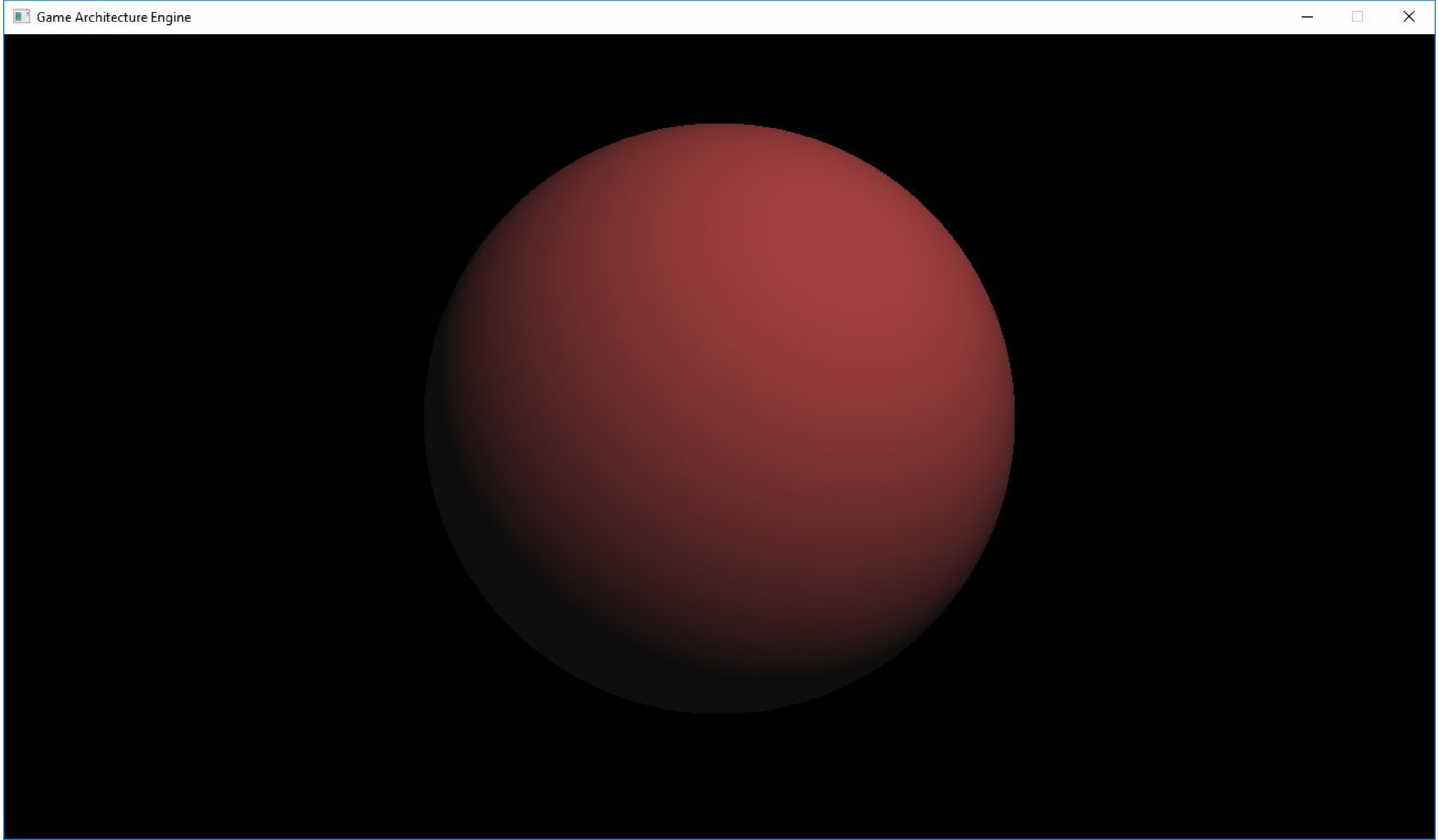
Lambert Lighting Model

```
vec3 get_color(
    surface surface,
    point_light light
)
{
    vec3 L = normalize(light._position - surface._position);
    vec3 N = normalize(surface._normal);
    return surface._material._color * light._color * dot(N, L);
}
```

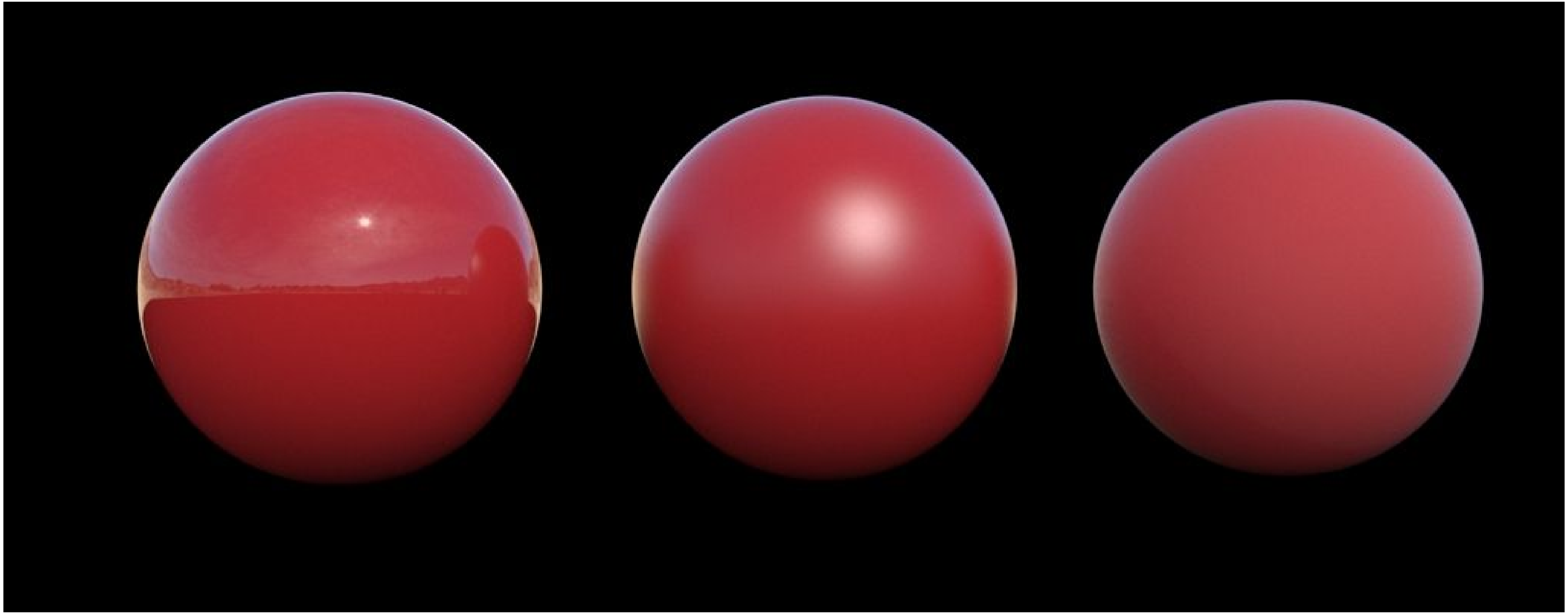

Game Architecture Engine

Lambert Lighting Model

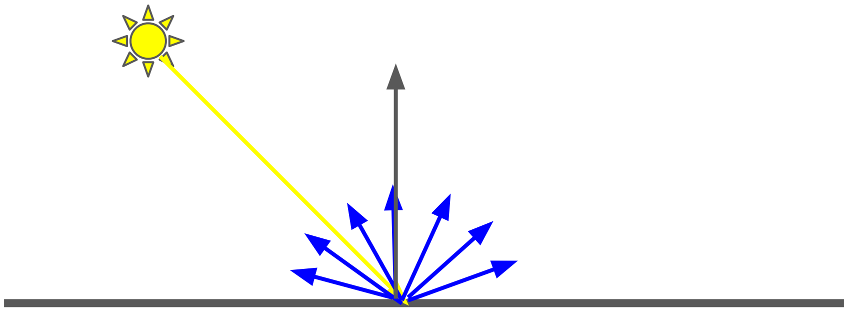
```
vec3 get_color(
    surface surface,
    point_light light
)
{
    vec3 L = normalize(light._position - surface._position);
    vec3 N = normalize(surface._normal);
    vec3 ambient_reflection = vec3(0.05);
    vec3 lambert_reflection =
        surface._material._color * light._color * dot(N, L)
    return ambient_reflection + lambert_reflection;
}
```


Game Architecture Engine

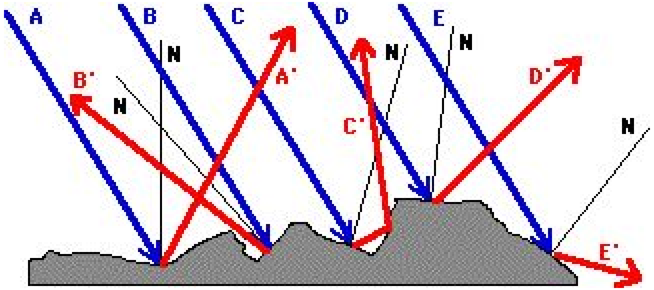
Example



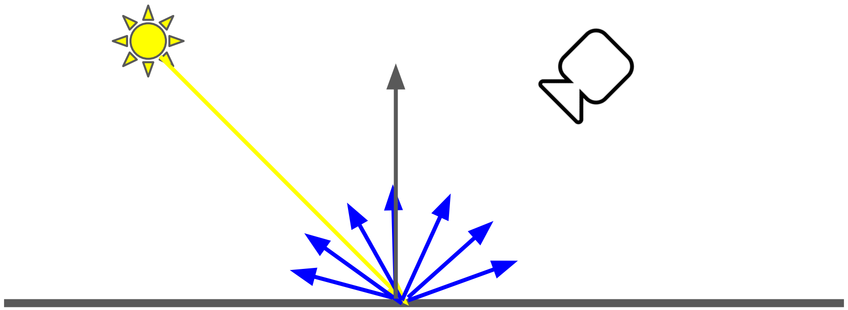
Diffuse Reflectance



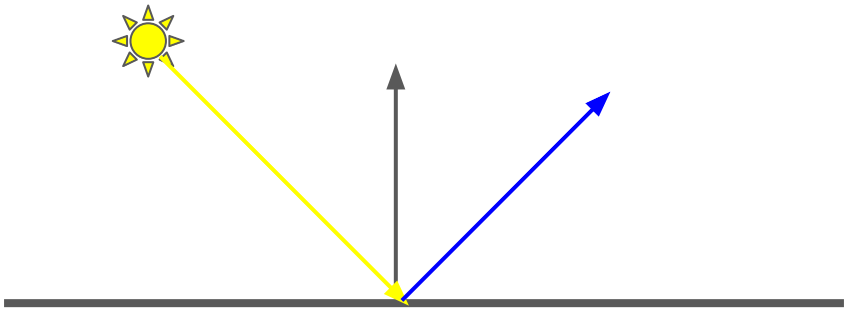
Diffuse Reflectance



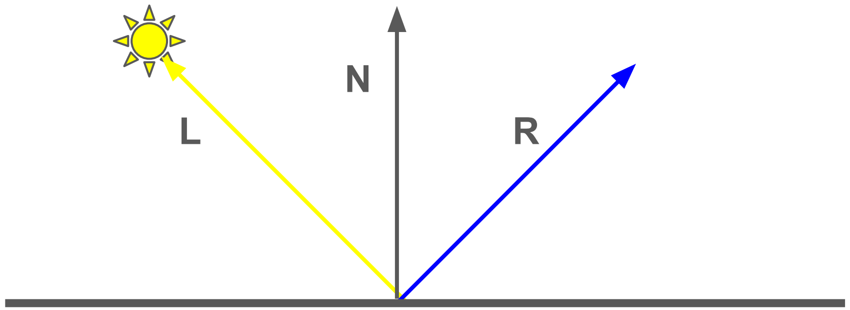
Diffuse Reflectance



Specular Reflectance

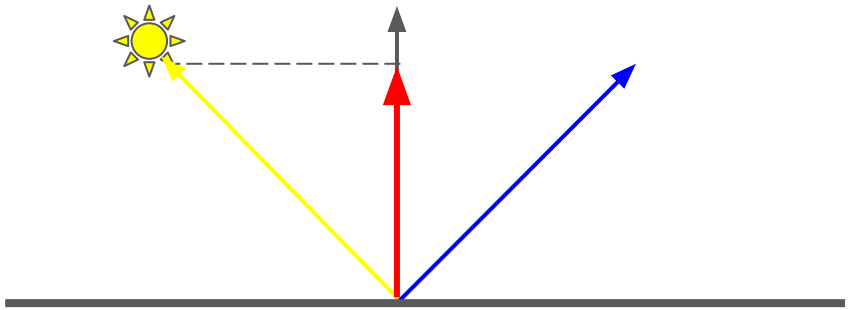


Specular Reflectance



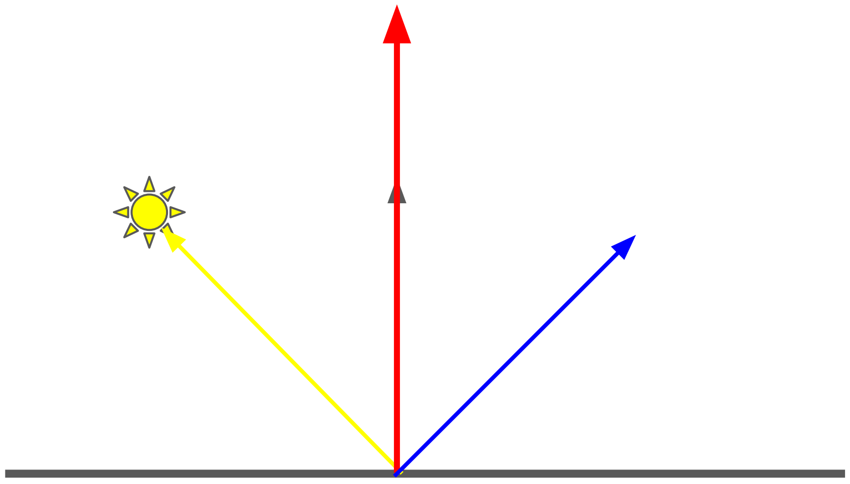
Specular Reflectance

$\mathbf{R} = \mathrm{proj}_N(L)$



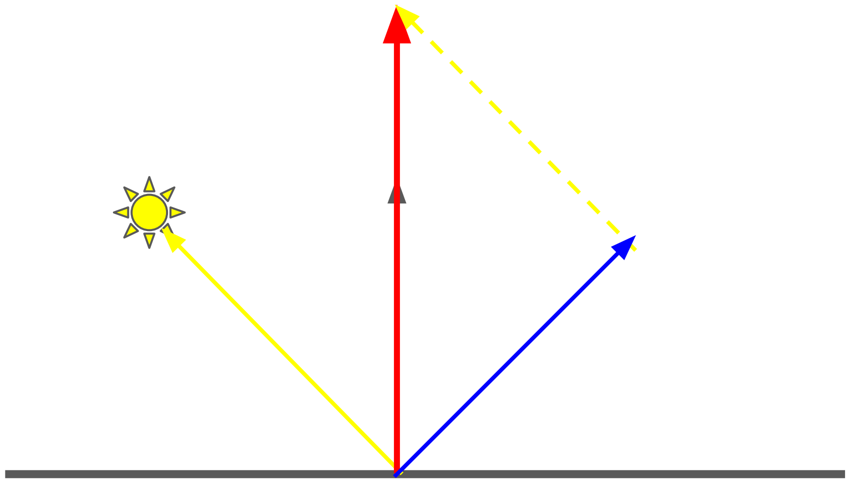
Specular Reflectance

$\mathbf{R} = 2\mathrm{proj}_{\mathbf{N}}(\mathbf{L})$



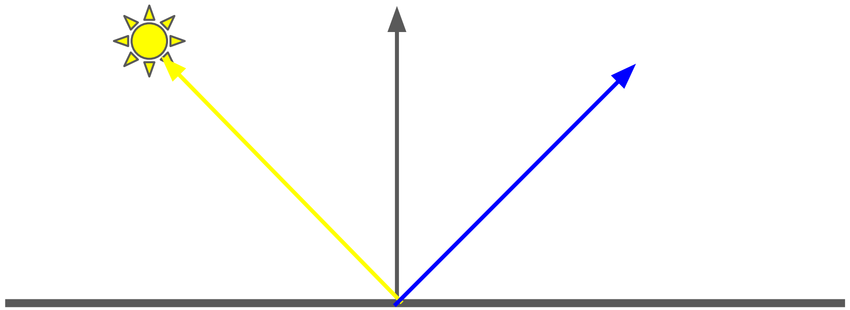
Specular Reflectance

$\mathbf{R} = 2\text{proj}_{\mathbf{N}}(\mathbf{L}) - \mathbf{L}$

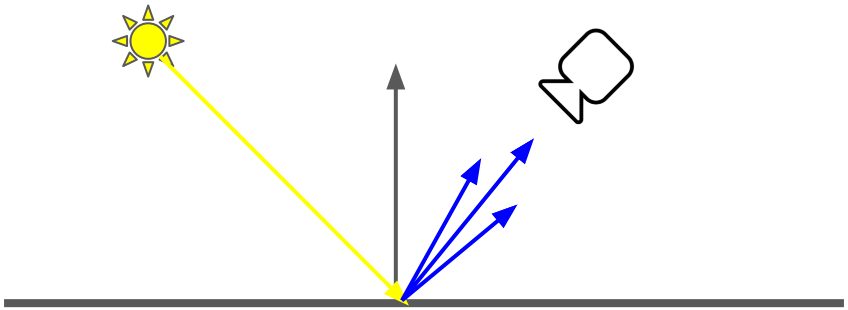


Specular Reflectance

$$\mathbf{R} = 2(\mathbf{N}\cdot\mathbf{L})\mathbf{N} - \mathbf{L}$$



Specular Reflectance



Specular Reflection

```
vec3 get_specular_color(
    surface surface,
    point_light light,
    camera camera)
{
}
```

```
struct camera
{
    vec3 _position;
};
```

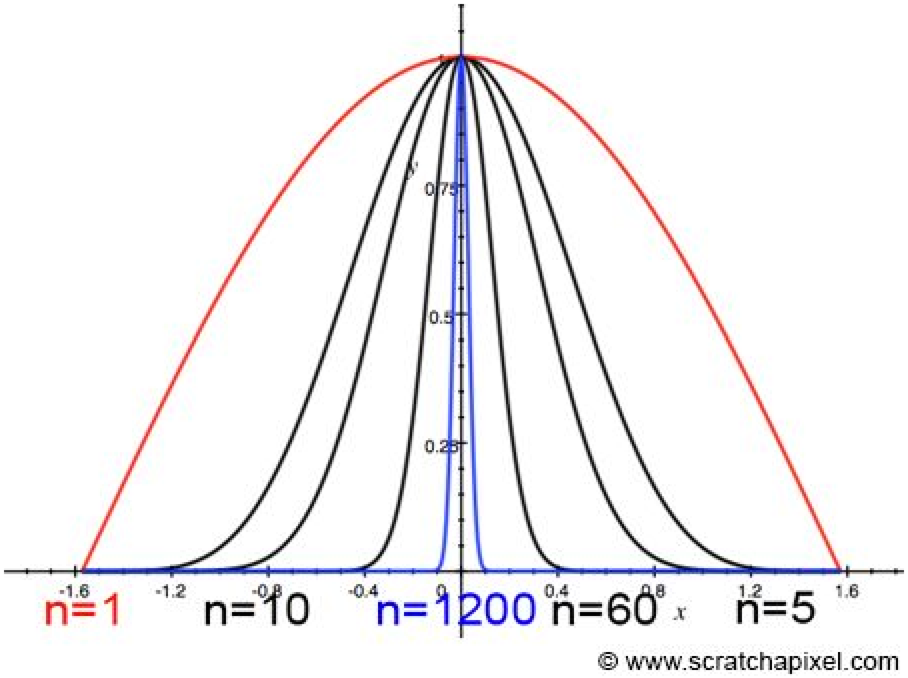
Specular Reflection

```
vec3 get_specular_color(
    surface surface,
    point_light light,
    camera camera)
{
    vec3 L = normalize(light._position - surface._position);
    vec3 N = normalize(surface._normal);
    vec3 R = reflect(-L, N);
}
```

Phong Lighting Model

```
vec3 get_specular_color(
    surface surface,
    point_light light,
    camera camera)
{
    vec3 L = normalize(light._position - surface._position);
    vec3 N = normalize(surface._normal);
    vec3 R = reflect(-L, N);
    vec3 V = normalize(camera._position - surface._position);
    float specular_intensity = dot(R, V);
    return vec3(specular_intensity);
}
```

Phong Lighting Model

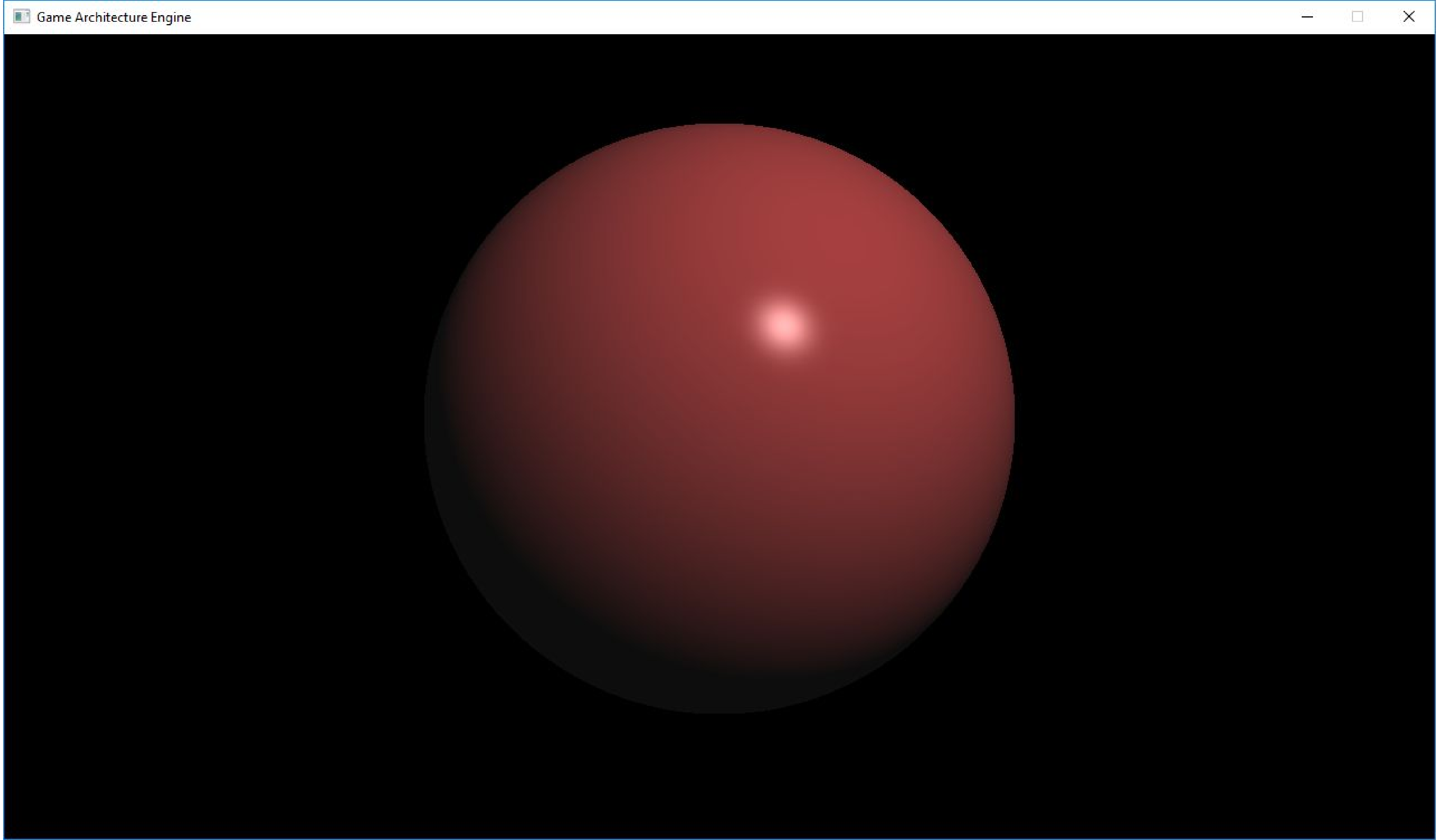


Specular Reflection

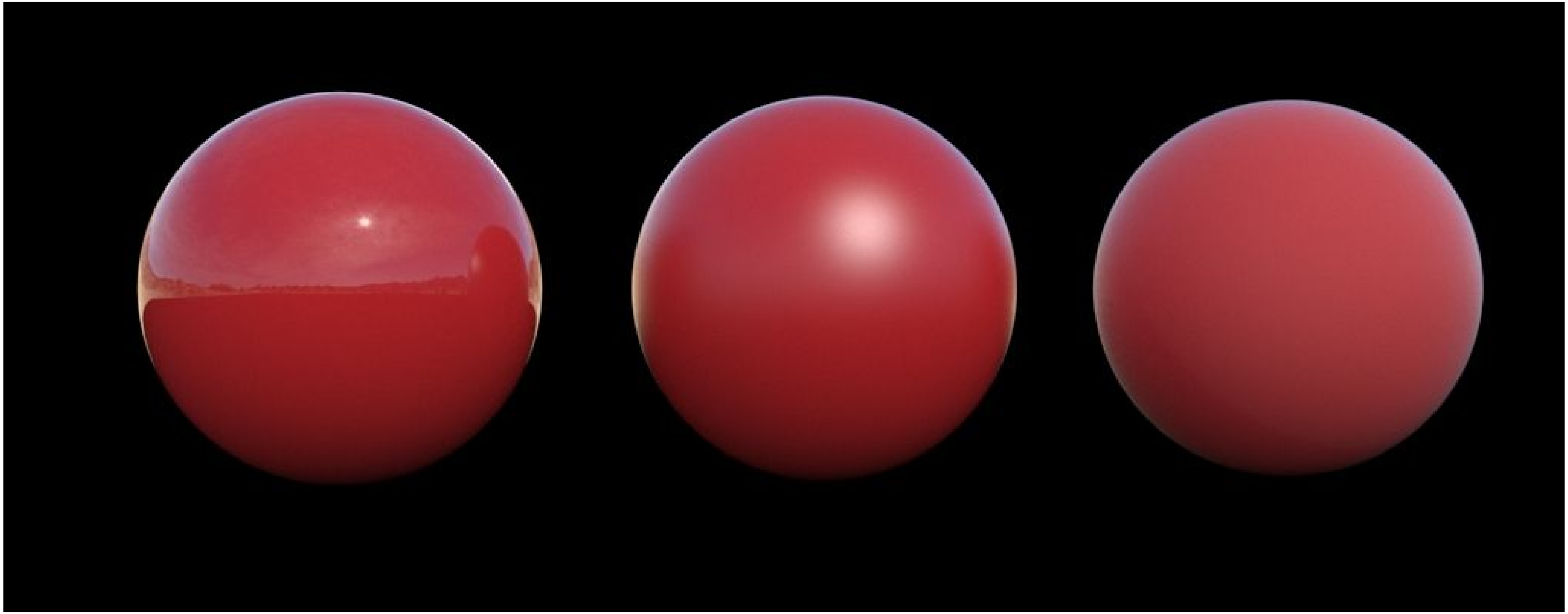
```
struct material
{
    vec3 _color;
    float _specular_power;
};
```

Phong Lighting Model

```
vec3 get_specular_color(
    surface surface,
    point_light light,
    camera camera)
{
    vec3 L = normalize(light._position - surface._position);
    vec3 N = normalize(surface._normal);
    vec3 R = reflect(-L, N);
    vec3 V = normalize(camera._position - surface._position);
    float specular_intensity =
        pow(dot(R, V), surface._material.specular_power);
    return vec3(specular_intensity);
}
```


Game Architecture Engine

Example



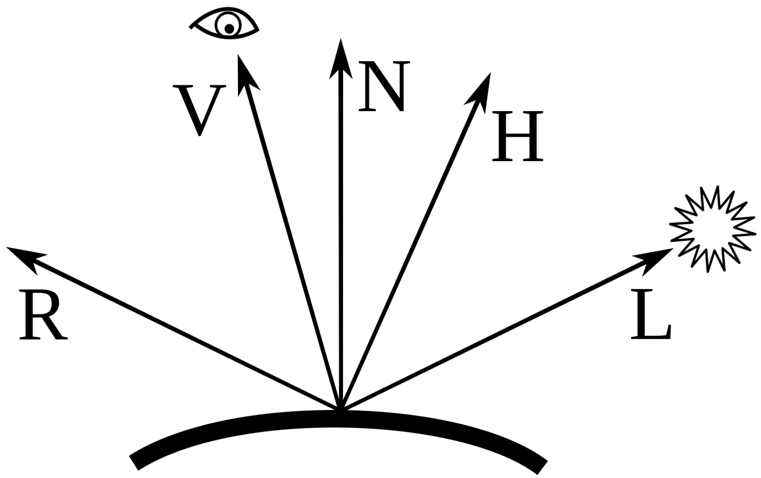
Physically Based Rendering

- Lambert and Phong are both empirical models.
- Neither attempt to realistically model the physical interaction of light with a surface.

Torrance-Sparrow Lighting Model

Based on a more complete model of reflection called the BRDF (Bidirectional Reflectance Distribution Function).

$$f(\mathbf{l}, \mathbf{v}) = \frac{F(\mathbf{l}, \mathbf{h}) G(\mathbf{l}, \mathbf{v}, \mathbf{h}) D(\mathbf{h})}{4(\mathbf{n} \cdot \mathbf{l})(\mathbf{n} \cdot \mathbf{v})}$$



Torrance-Sparrow Lighting Model

$$f(\mathbf{l}, \mathbf{v}) = \frac{\boxed{F(\mathbf{l}, \mathbf{h})} G(\mathbf{l}, \mathbf{v}, \mathbf{h}) D(\mathbf{h})}{4(\mathbf{n} \cdot \mathbf{l})(\mathbf{n} \cdot \mathbf{v})}$$

Fresnel reflectance

The Fresnel reflectance is the ratio of incoming light that is reflected vs refracted from a given surface.

Fresnel Reflectance

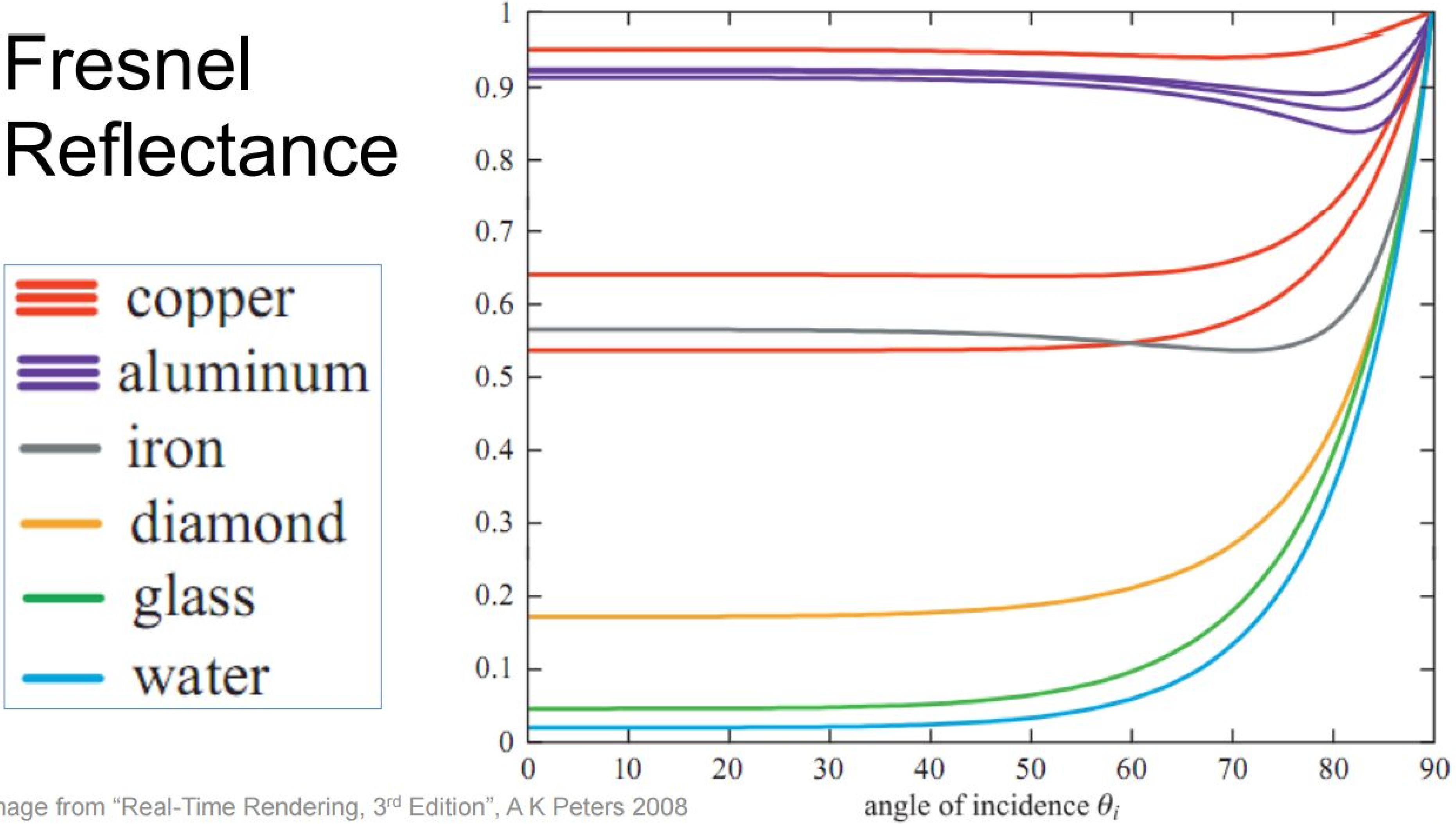
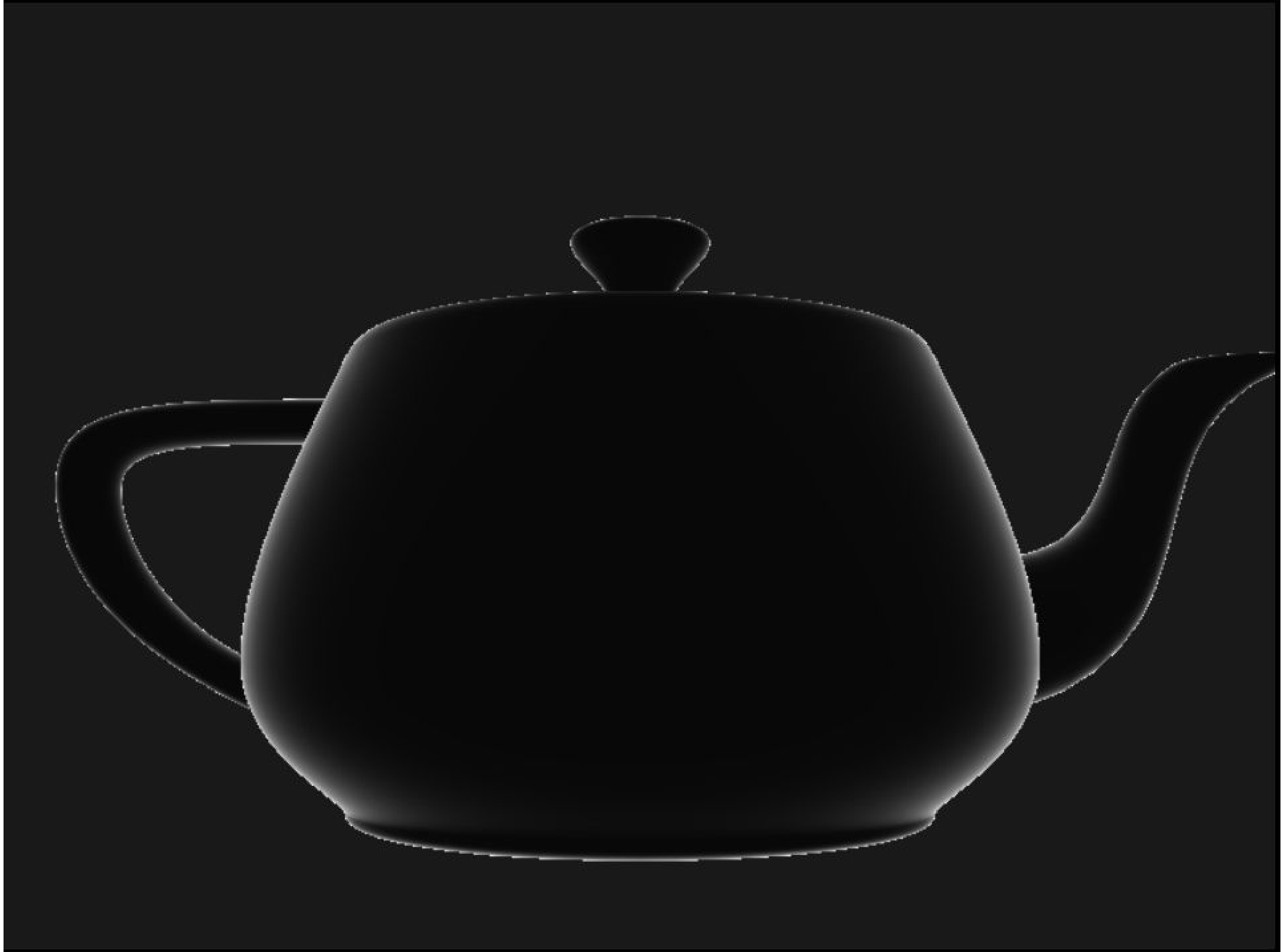


Image from "Real-Time Rendering, 3rd Edition", A K Peters 2008



Fresnel Reflectance

barely changes

changes somewhat

goes rapidly to 1

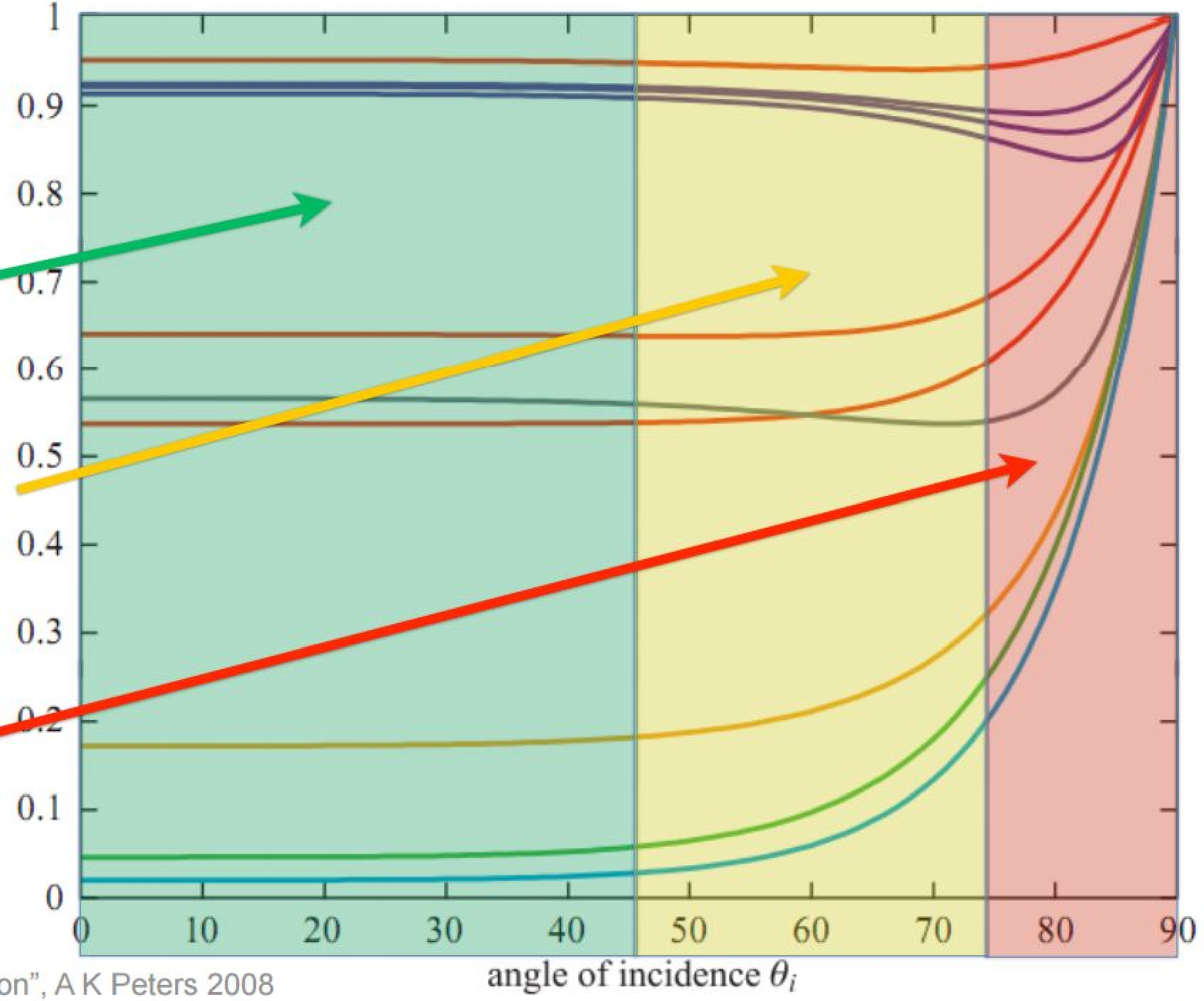
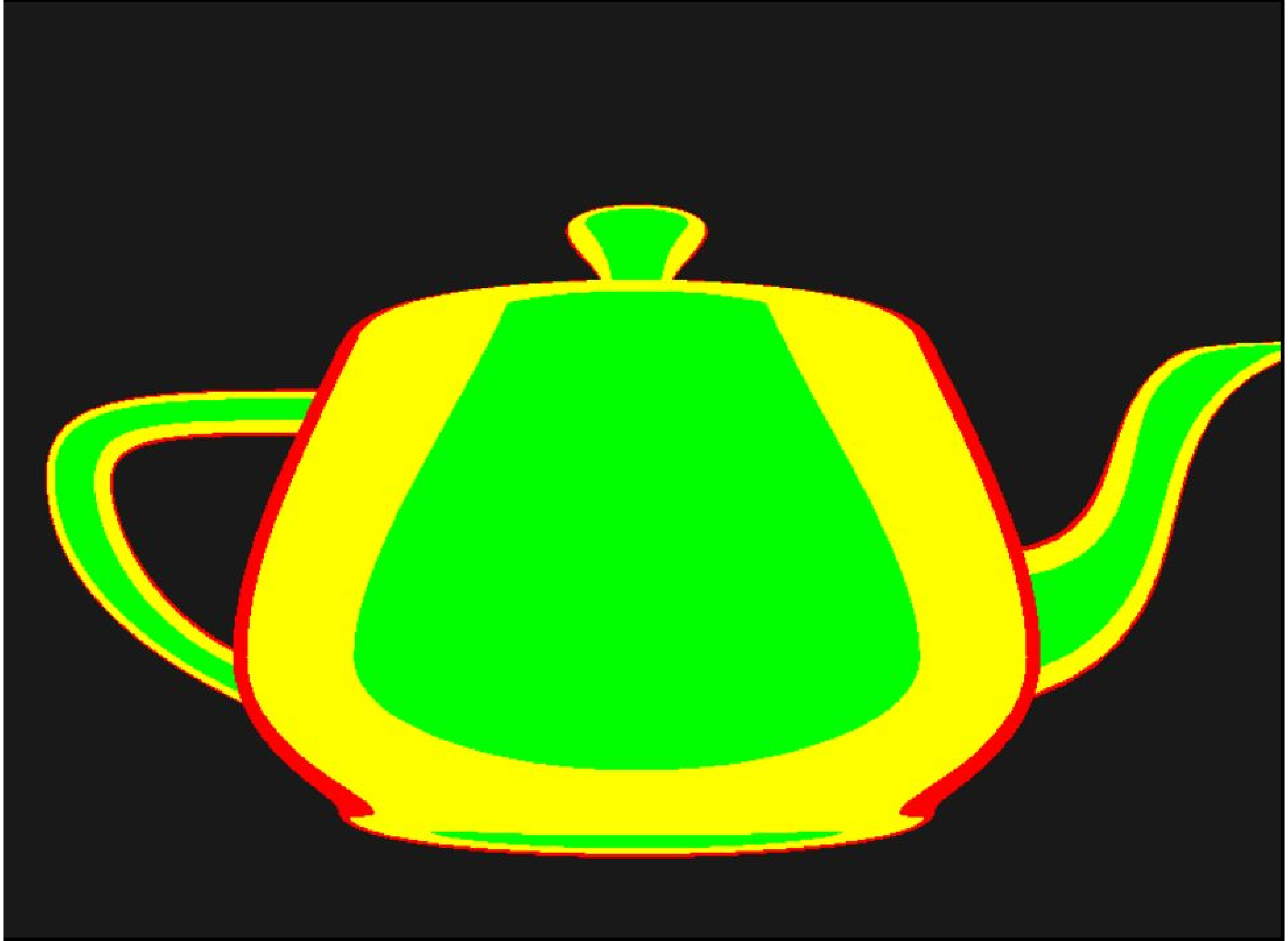


Image from “Real-Time Rendering, 3rd Edition”, A K Peters 2008



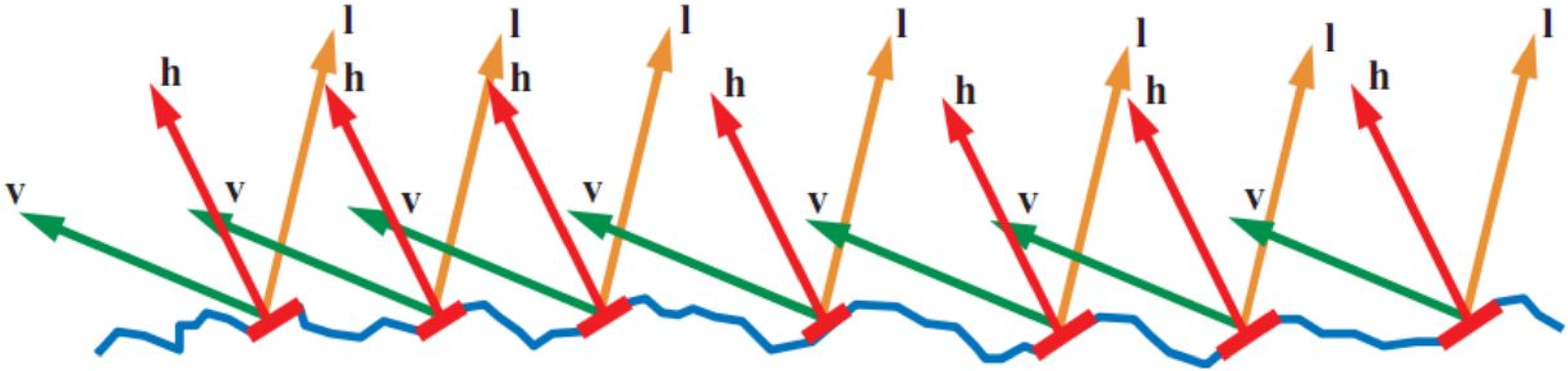
Torrance-Sparrow Lighting Model

$$f(\mathbf{l}, \mathbf{v}) = \frac{F(\mathbf{l}, \mathbf{h})G(\mathbf{l}, \mathbf{v}, \mathbf{h})\boxed{D(\mathbf{h})}}{4(\mathbf{n} \cdot \mathbf{l})(\mathbf{n} \cdot \mathbf{v})}$$

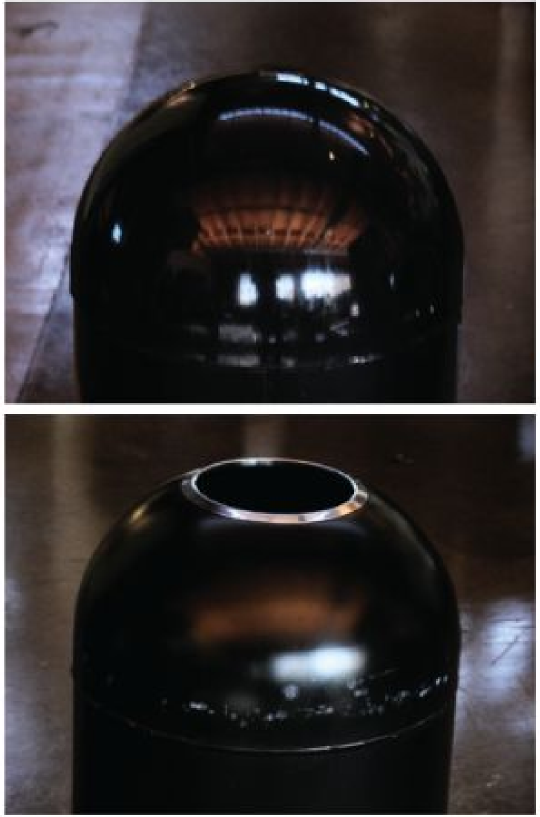
Normal Distribution Function

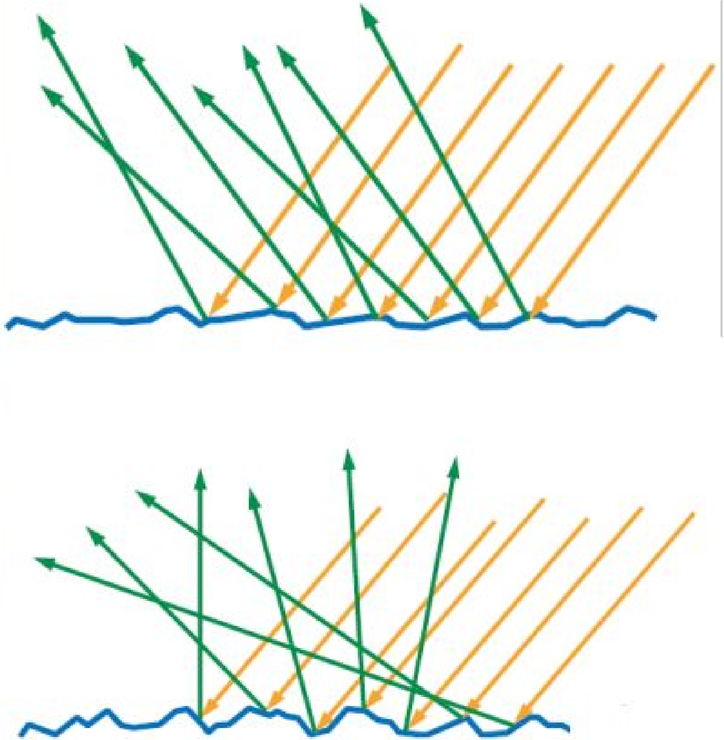
A probability density function giving the likelihood of a microfacet normal being aligned in a particular direction (the half-angle direction).

Microfacet Theory



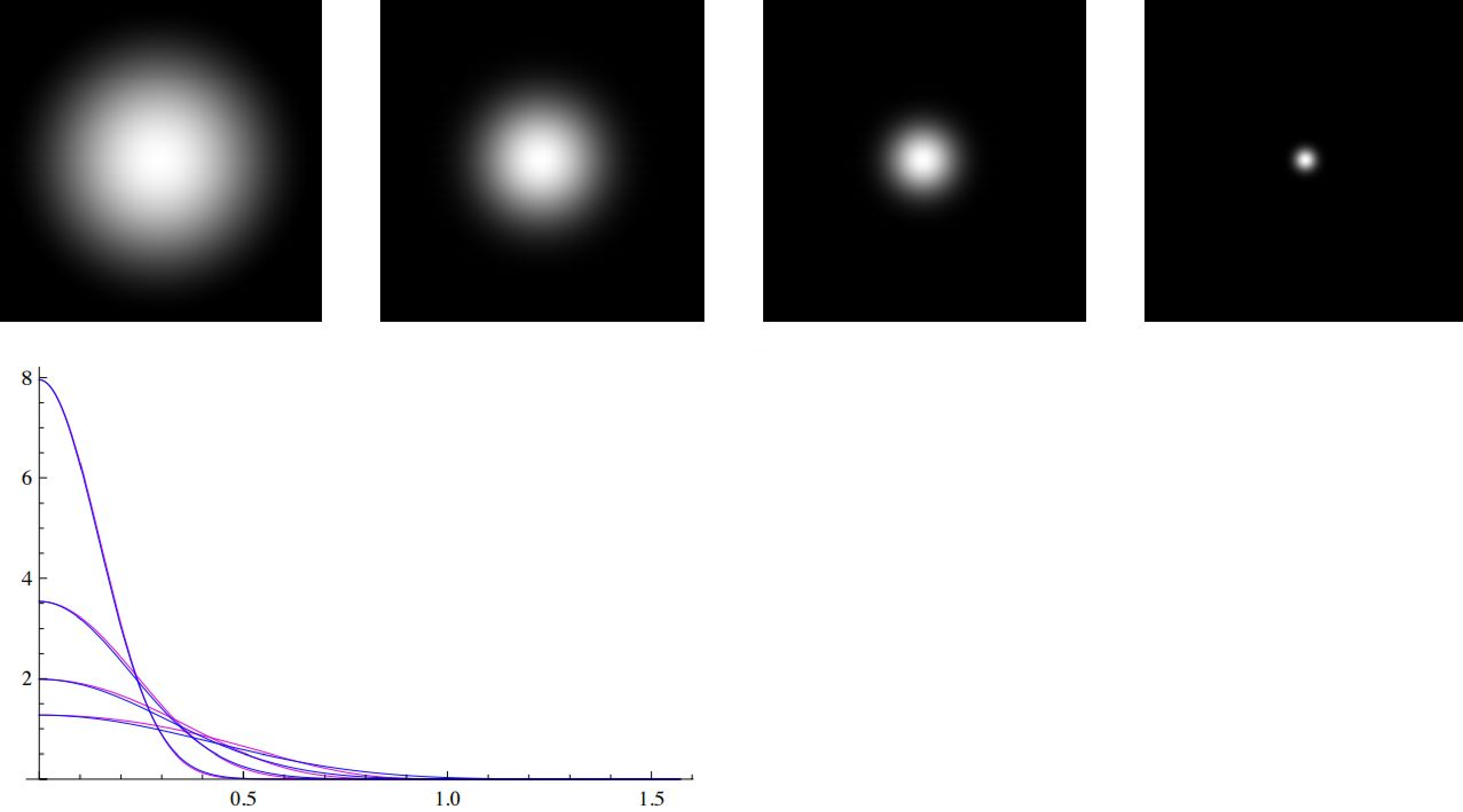
Rougher = Blurrier Reflections

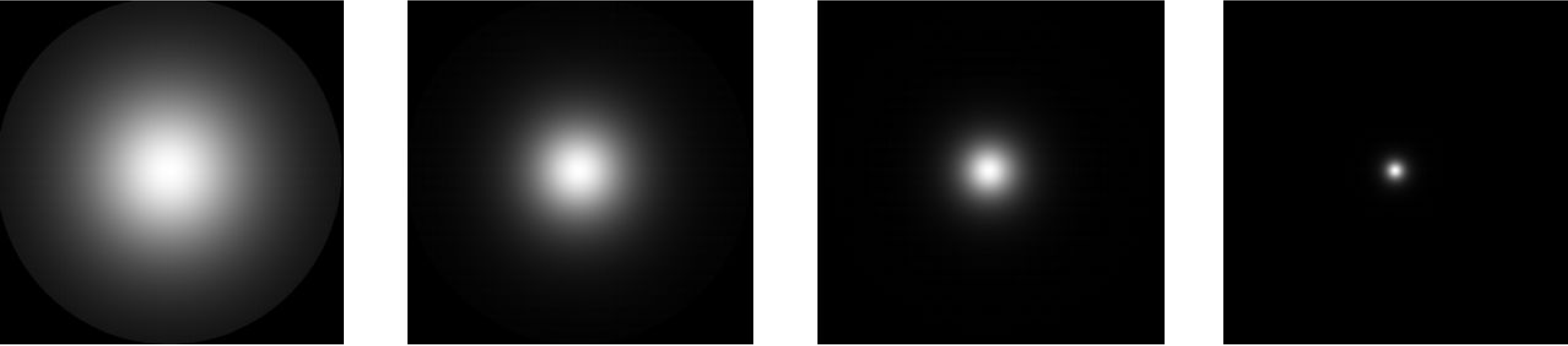


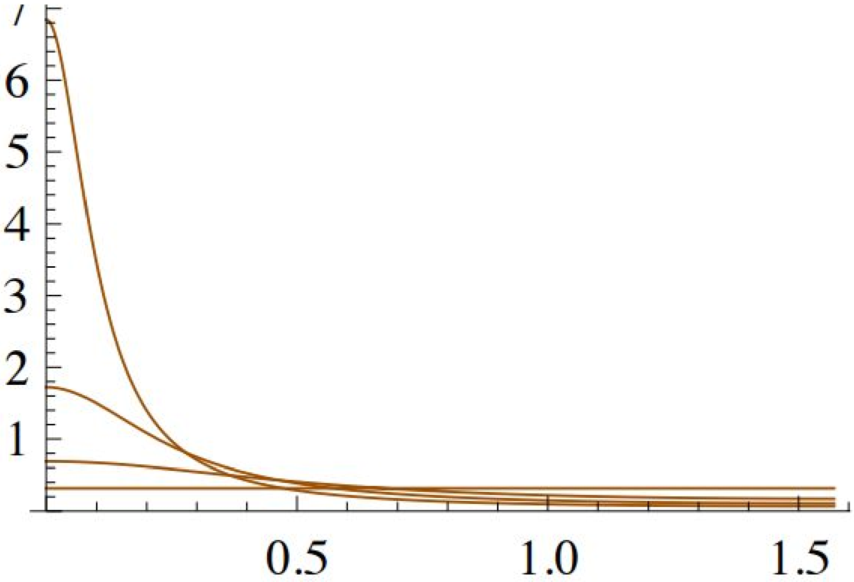


Images from "Real-Time Rendering, 3rd Edition", A K Peters 2008

SIGGRAPH2015


8
6
4
2
0.5
1.0
1.5





7
6
5
4
3
2
1
0.5
1.0
1.5

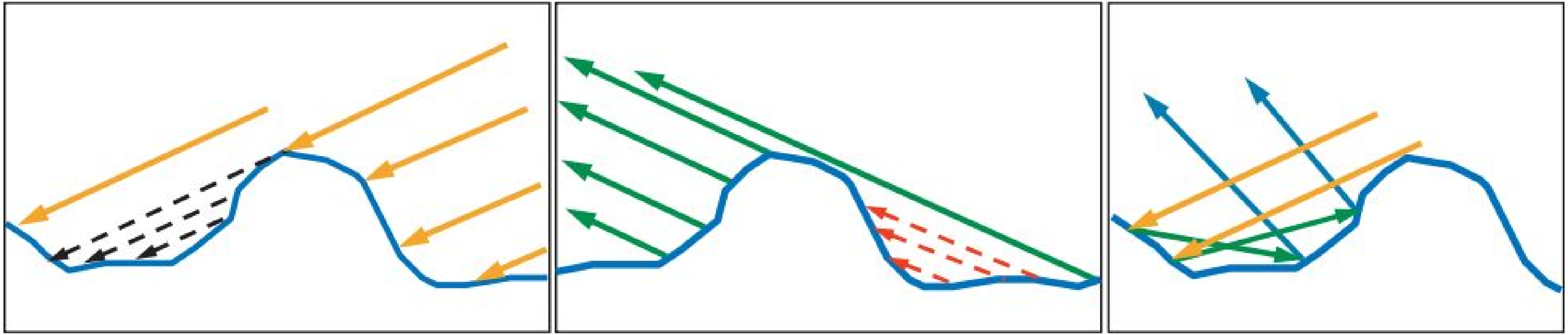
Torrance-Sparrow Lighting Model

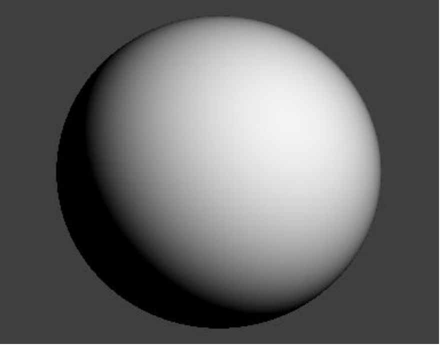
$$f(\mathbf{l}, \mathbf{v}) = \frac{F(\mathbf{l}, \mathbf{h})G(\mathbf{l}, \mathbf{v}, \mathbf{h})D(\mathbf{h})}{4(\mathbf{n} \cdot \mathbf{l})(\mathbf{n} \cdot \mathbf{v})}$$

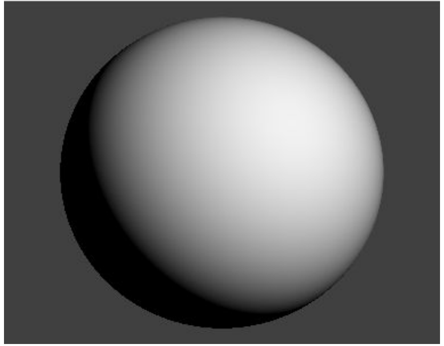
Geometry Function

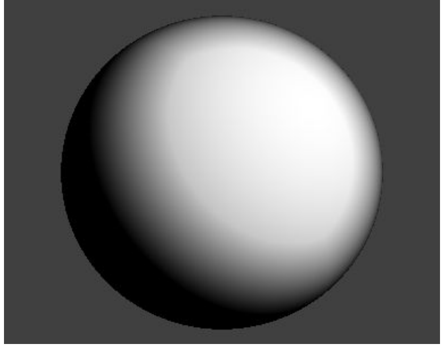
Describing the probability that reflection from a microfacet is blocked by other microfacets.

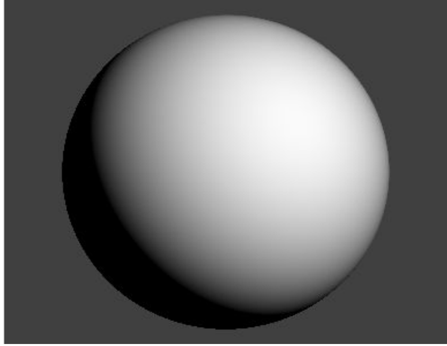
Shadowing and Masking

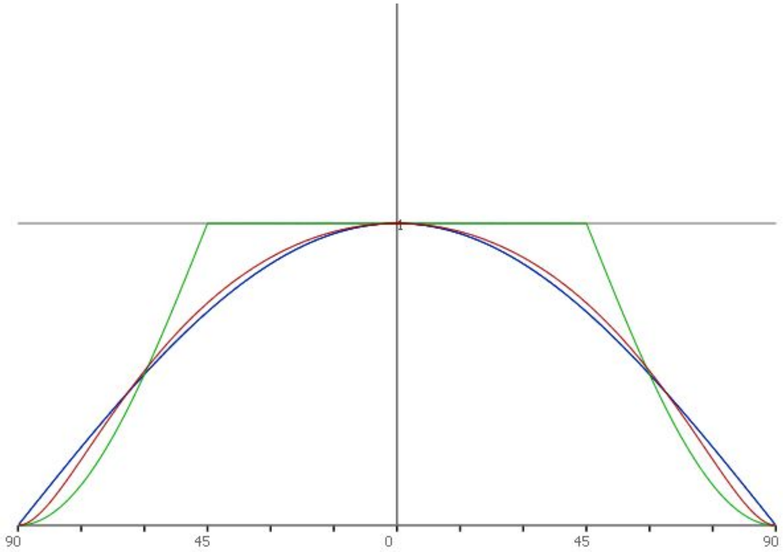











1
90
45
0
45
90

Torrance-Sparrow Lighting Model

```
vec3 get_specular_color(
    surface surface,
    point_light light,
    camera camera)
{
    vec3 L = normalize(light._position - surface._position);
    vec3 V = normalize(camera._position - surface._position);
    return vec3(brdf(L, V));
}
```

Cornell Box

Shading

- Answers the question of how to apply color to each fragment (or pixel) to rasterized surface..
- Closely related but not the same thing as lighting!

- Flat
- Gouraud
- Phong


AKIRA
SARAH
45"00
ROUND 1
STAGE 2
1' 19"76

Shading

- Answers the question of how to apply color to each fragment (or pixel) to rasterized surface..
- Closely related but not the same thing as lighting!

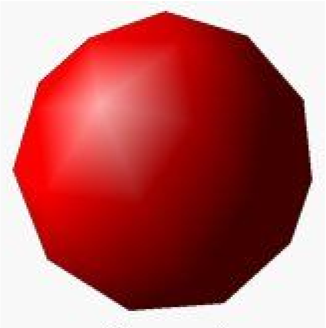
- **Flat**

- Gouraud
- Phong

Shading

- Answers the question of how to apply color to each fragment (or pixel) to rasterized surface..
- Closely related but not the same thing as lighting!

- Flat
- **Gouraud**

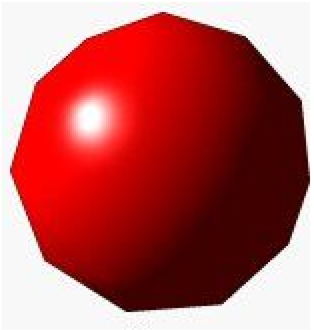


- Phong

Shading

- Answers the question of how to apply color to each fragment (or pixel) to rasterized surface..
- Closely related but not the same thing as lighting!

- Flat
- Gouraud
- **Phong**



Shadows

Global Illumination