Fundamentals of Computer Graphics

Lecture 7. OpenGL lighting

Yong-Jin Liu
liuyongjin@tsinghua.edu.cn

Outline

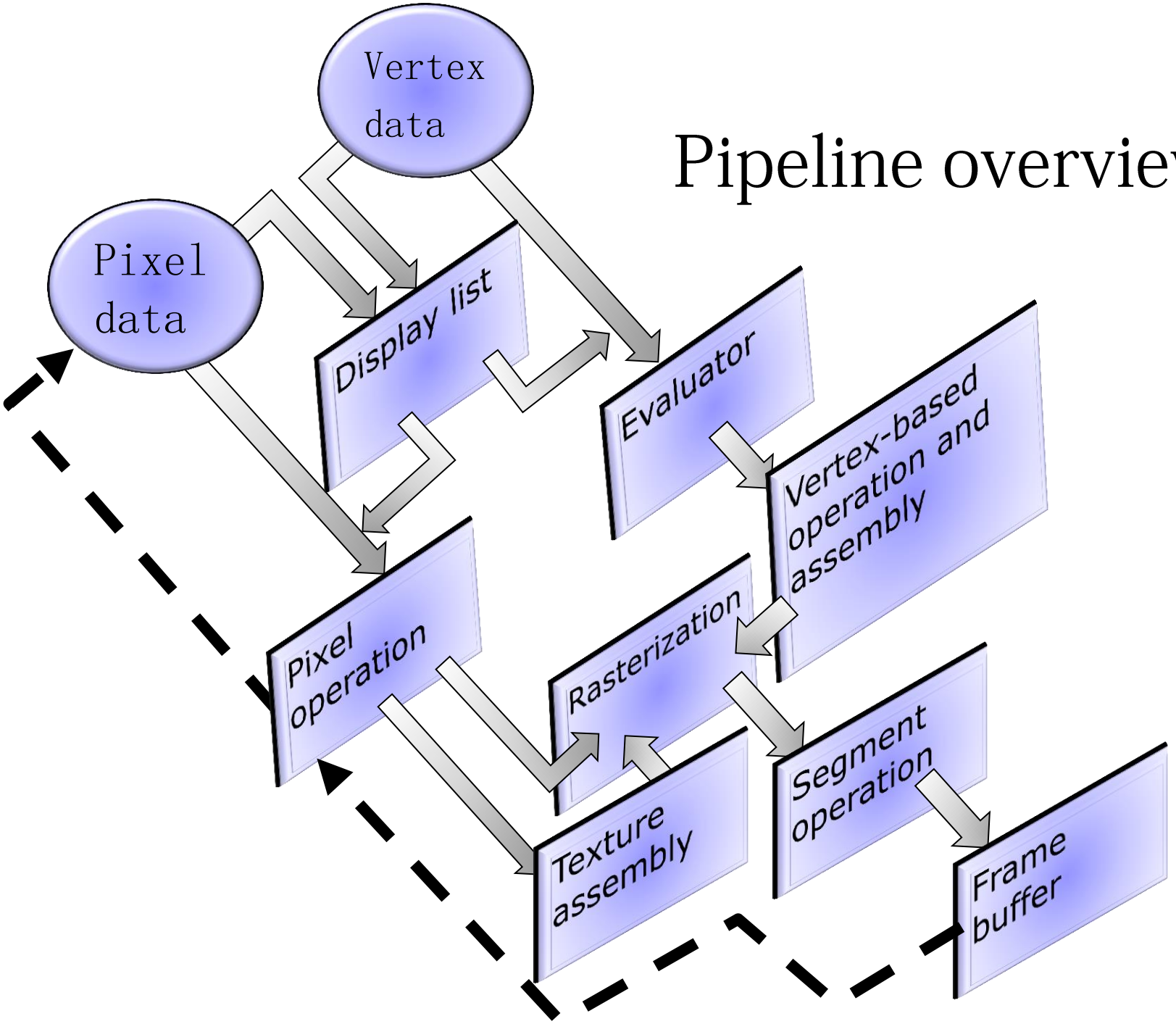
OpenGL rendering pipeline

- Geometry (object transformation)

  Learn how to carry out transformation in OpenGL

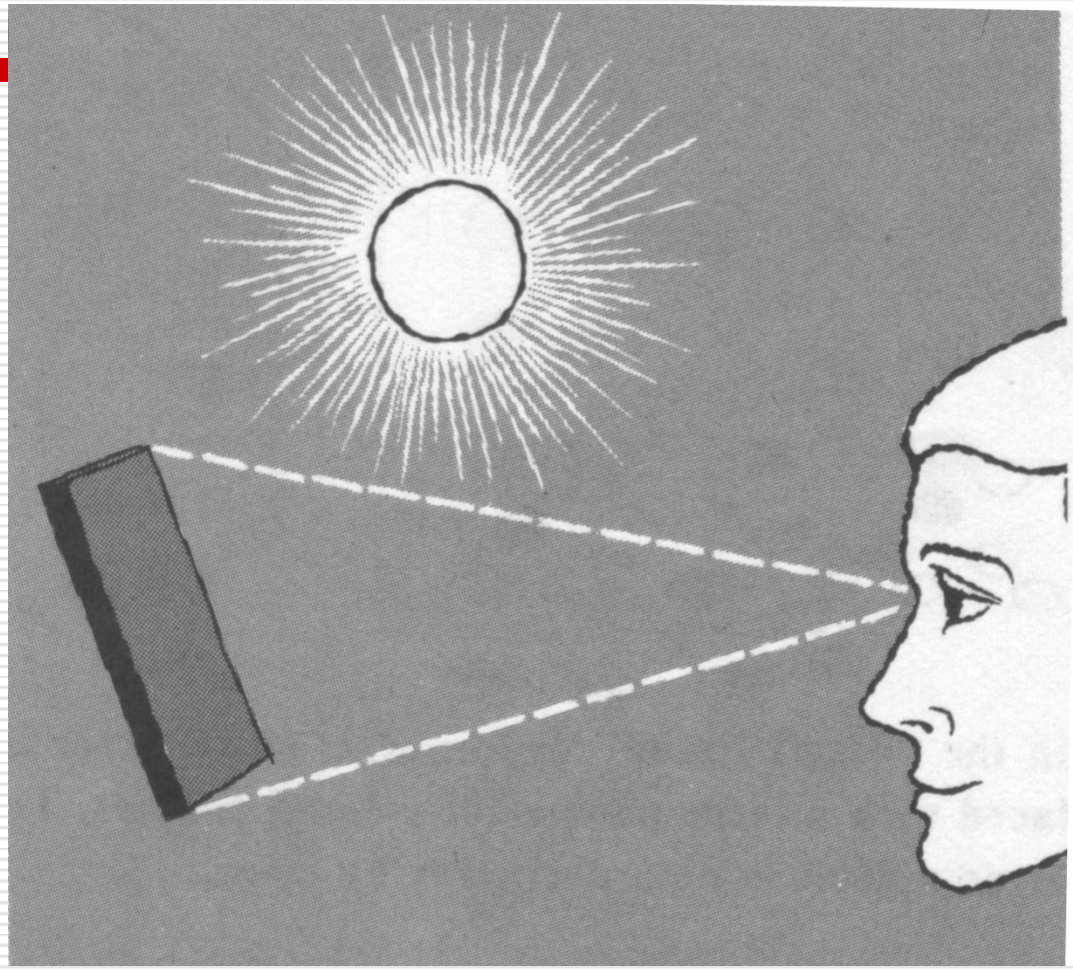
- Color (lighting, texture, etc.)

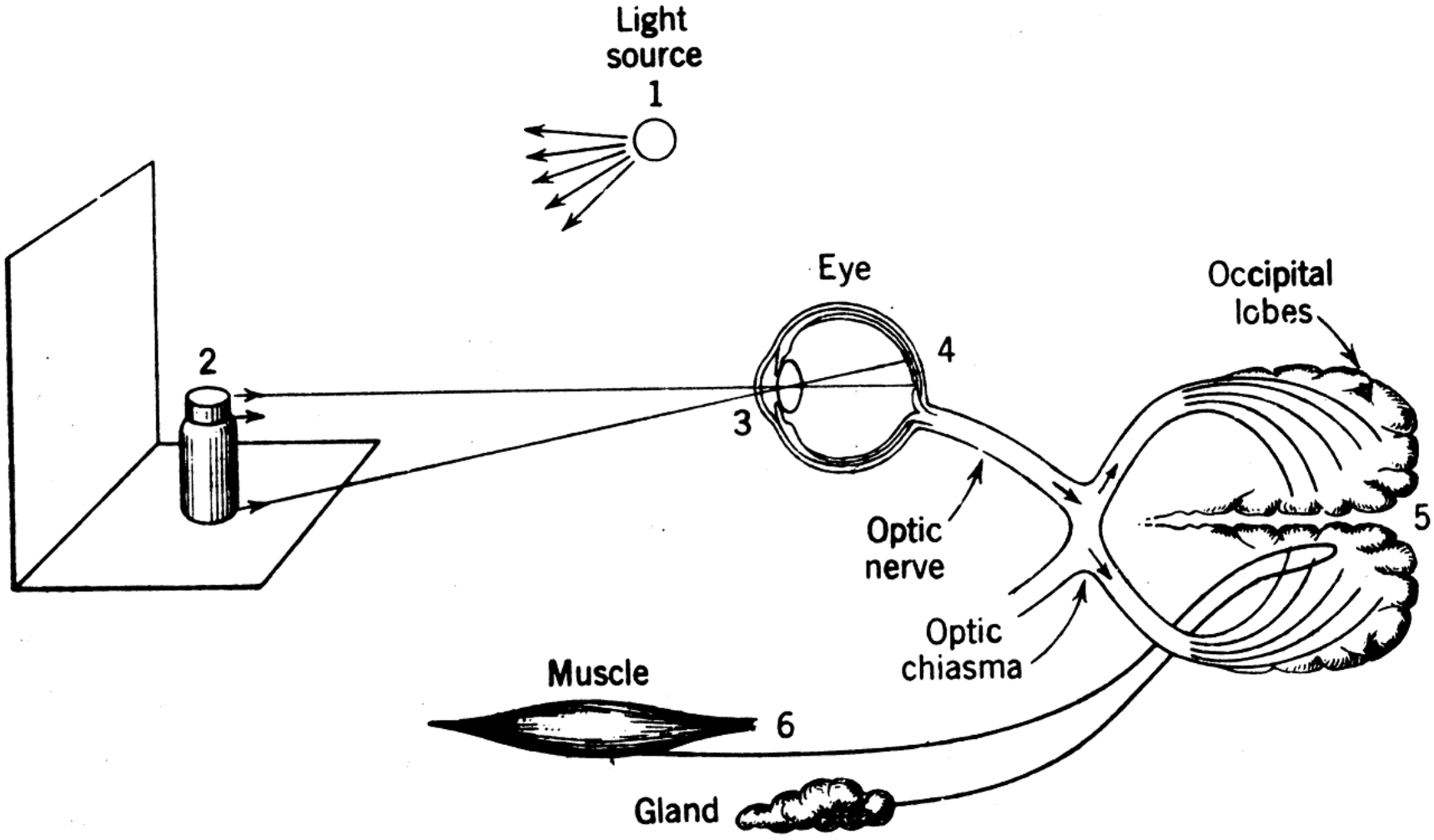
Pipeline overview



What is color

1. Physics (Physics of radiant energy)
2. Chemistry (Chemistry of material of pigments and dyes)
3. Physiology (Physiology of human vision system)
4. Psychology (Psychology of visual perception)

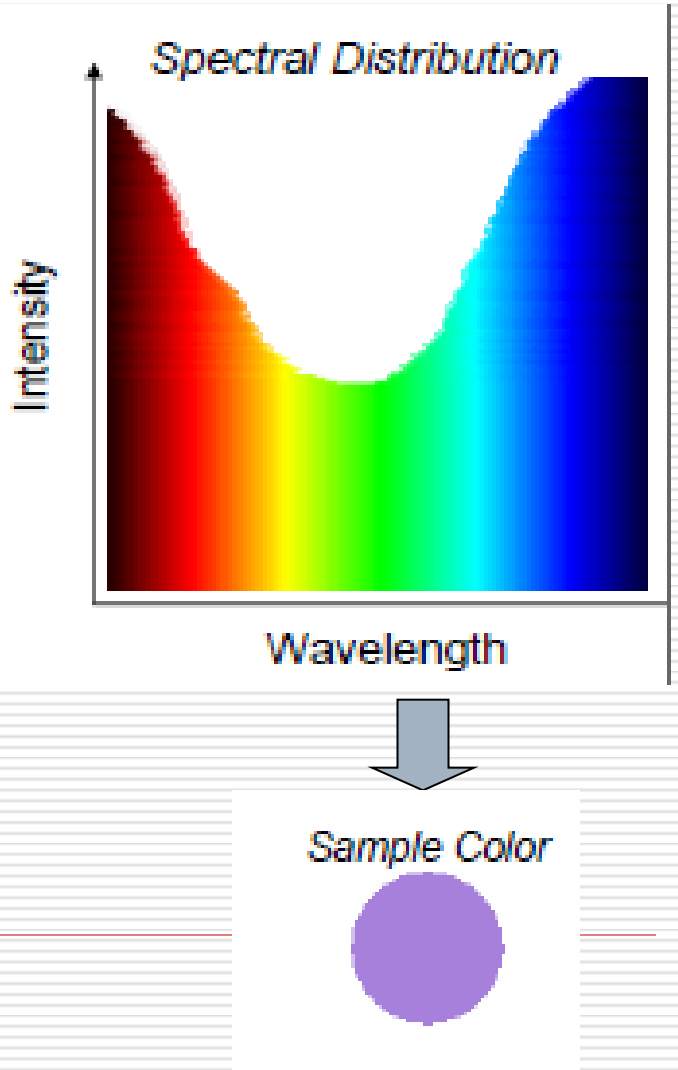


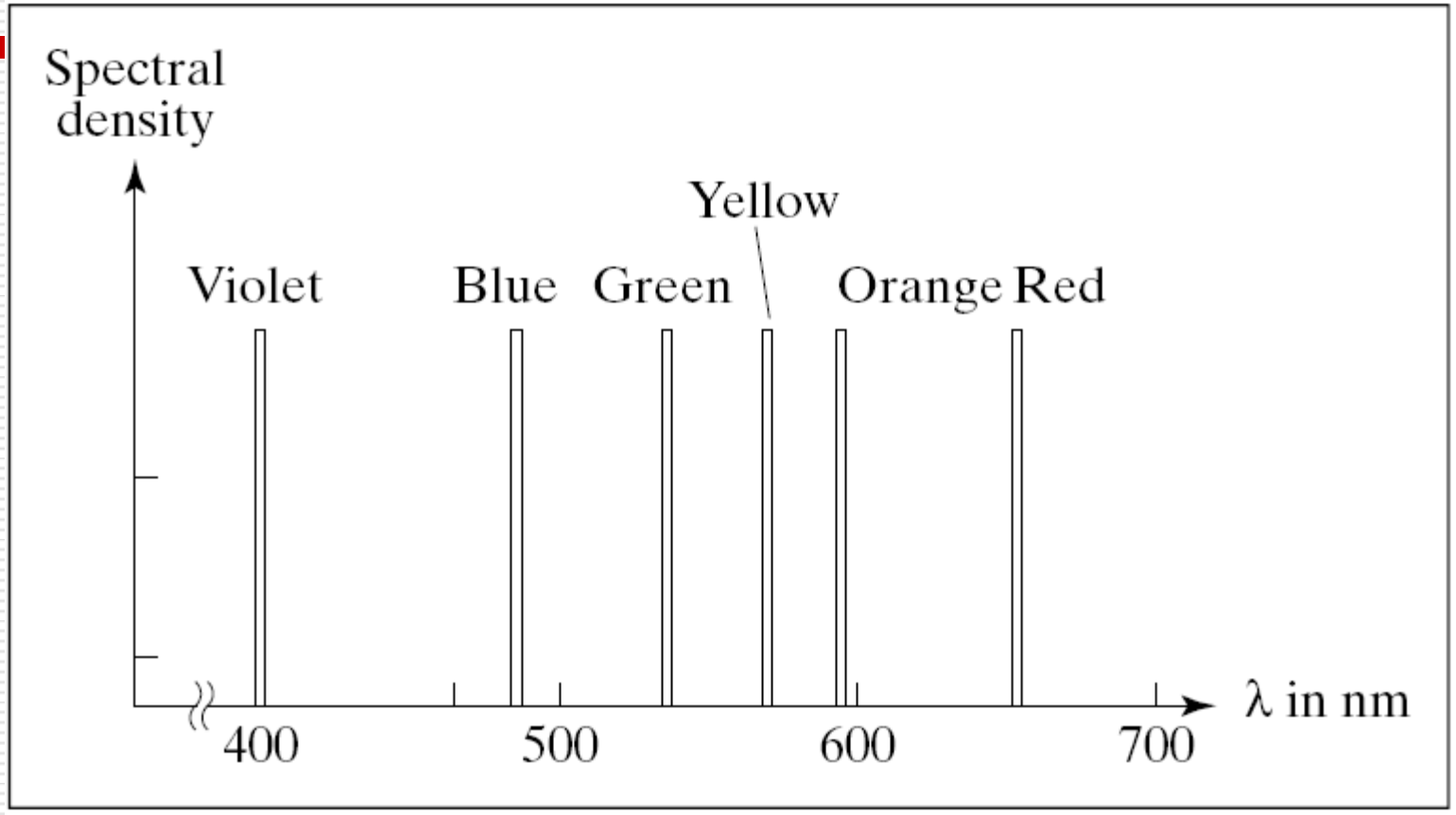

Light
source
1
Eye
Occipital
lobes
2
4
3
5
Optic
nerve
Optic
chiasma
Muscle
6
Gland

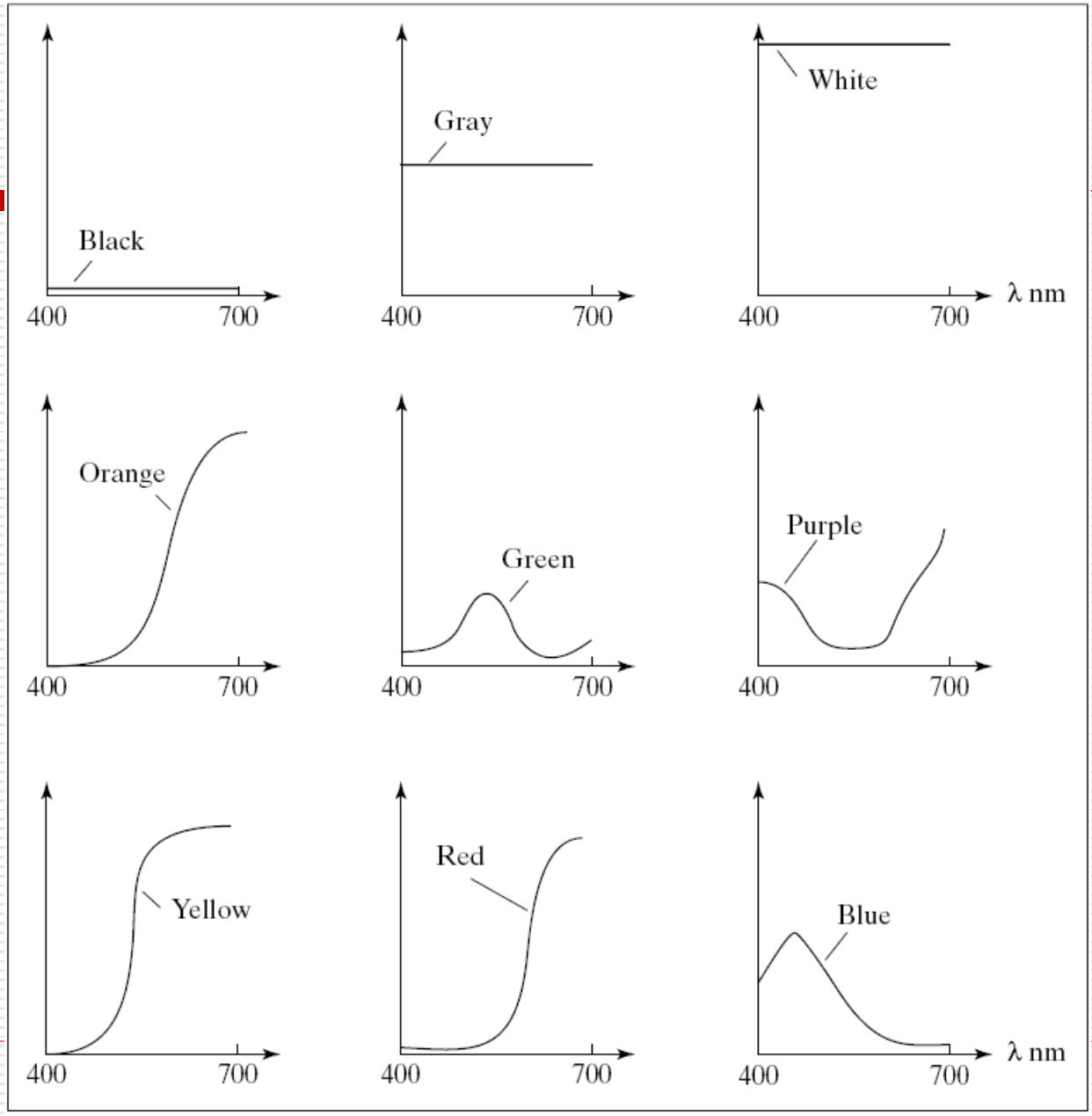
Spectral distribution of light (physics)

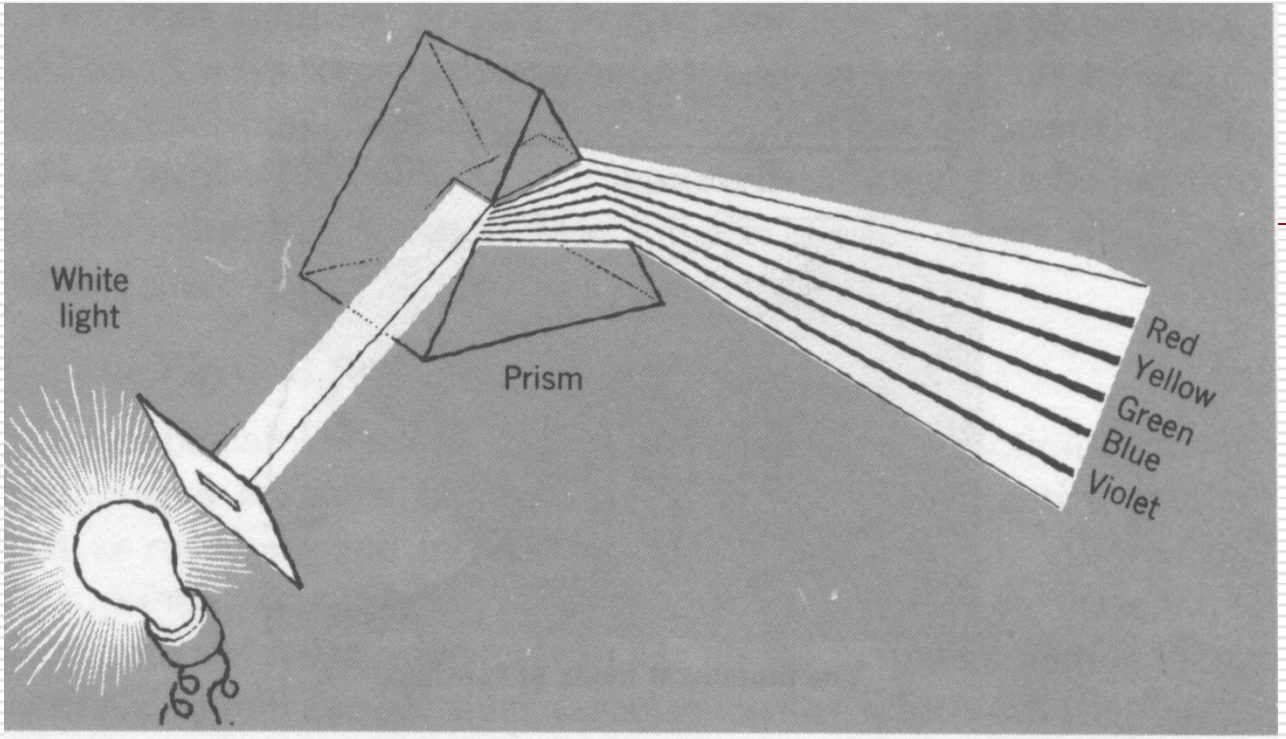
1. The usual sense of light actually contains many frequencies of electromagnetic waves

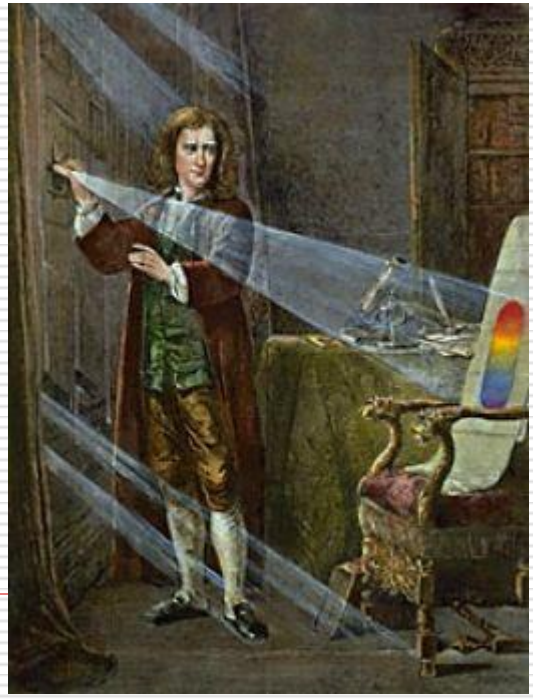
- Different frequency with different intensity
- Intensity with wavelength is the spectral distribution function
- The color of light can be expressed by this distribution function




Spectral density
Violet
Blue
Green
Yellow
Orange
Red
λ in nm
400
500
600
700

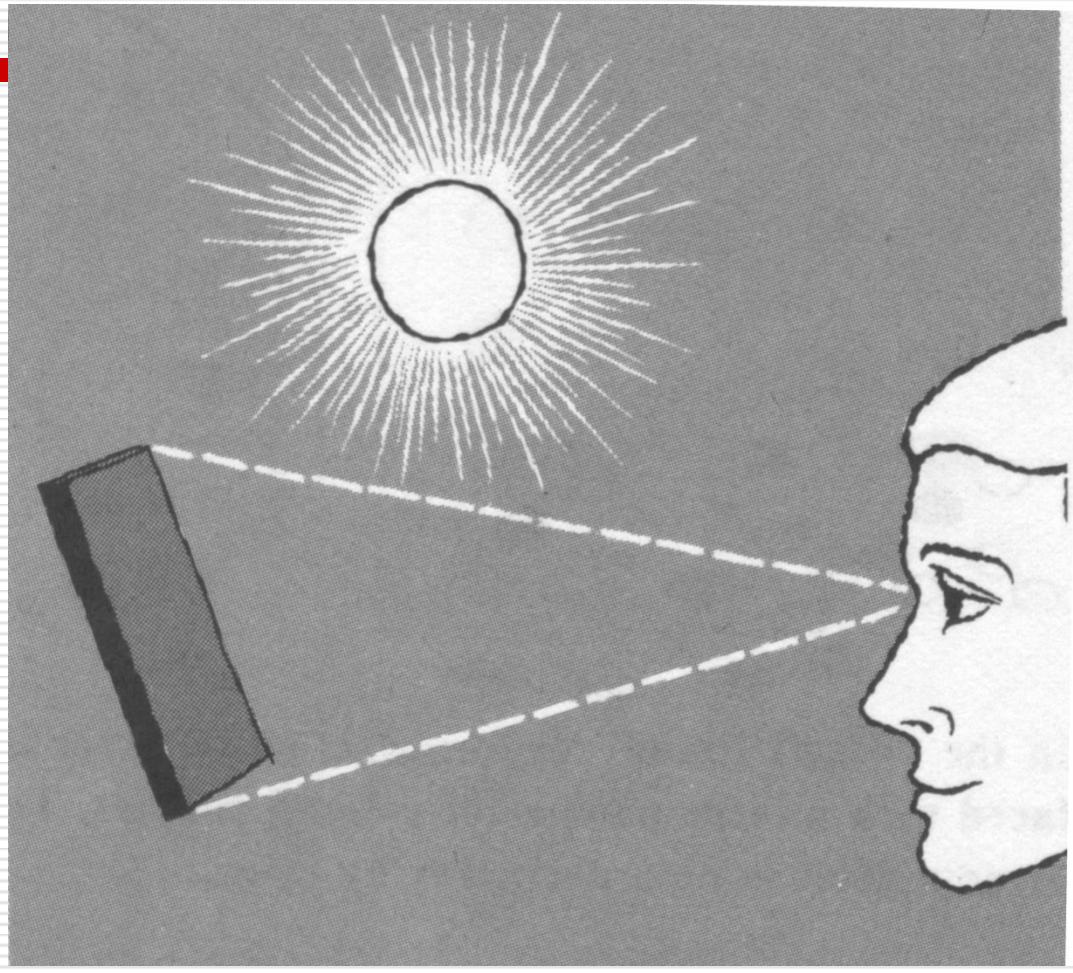

Black
400
700
Gray
400
700
White
400
700
λ nm
Orange
400
700
Green
400
700
Purple
400
700
Yellow
400
700
Red
400
700
Blue
400
700
λ nm


White
light
Prism
Red
Yellow
Green
Blue
Violet



Standard light sources

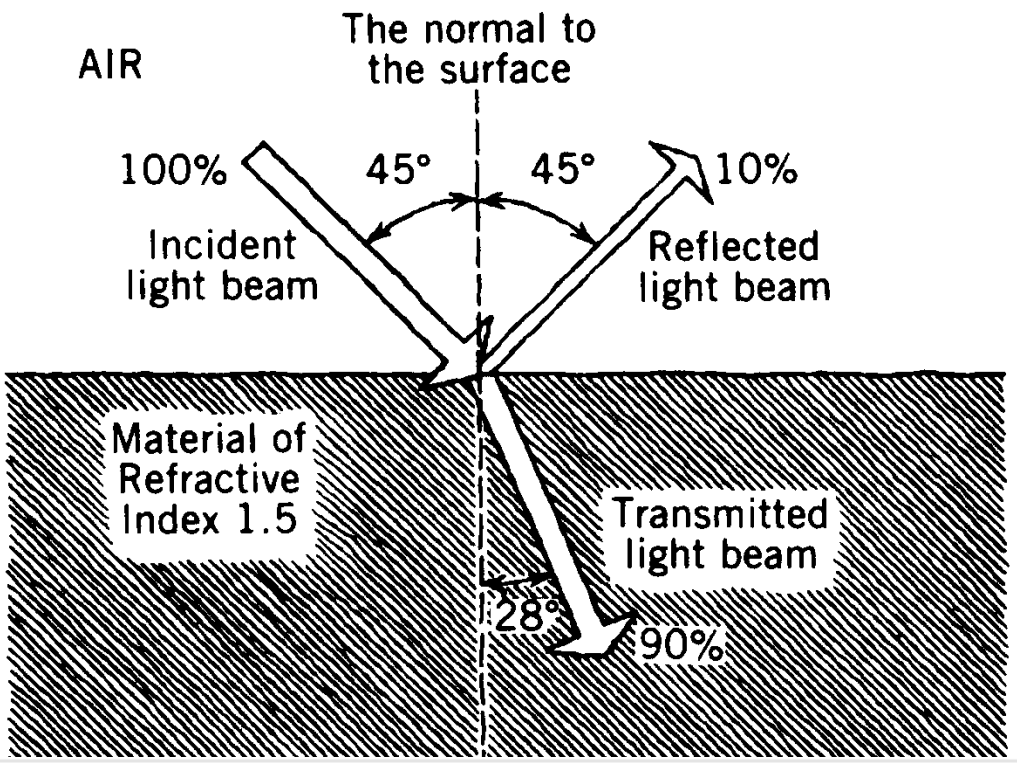
The light source which can be repeated through physics experiment in different places and time. Source is a physically realizable light, whose spectral power distribution can be experimentally determined. When the determination is made and specified, the source becomes a standard source

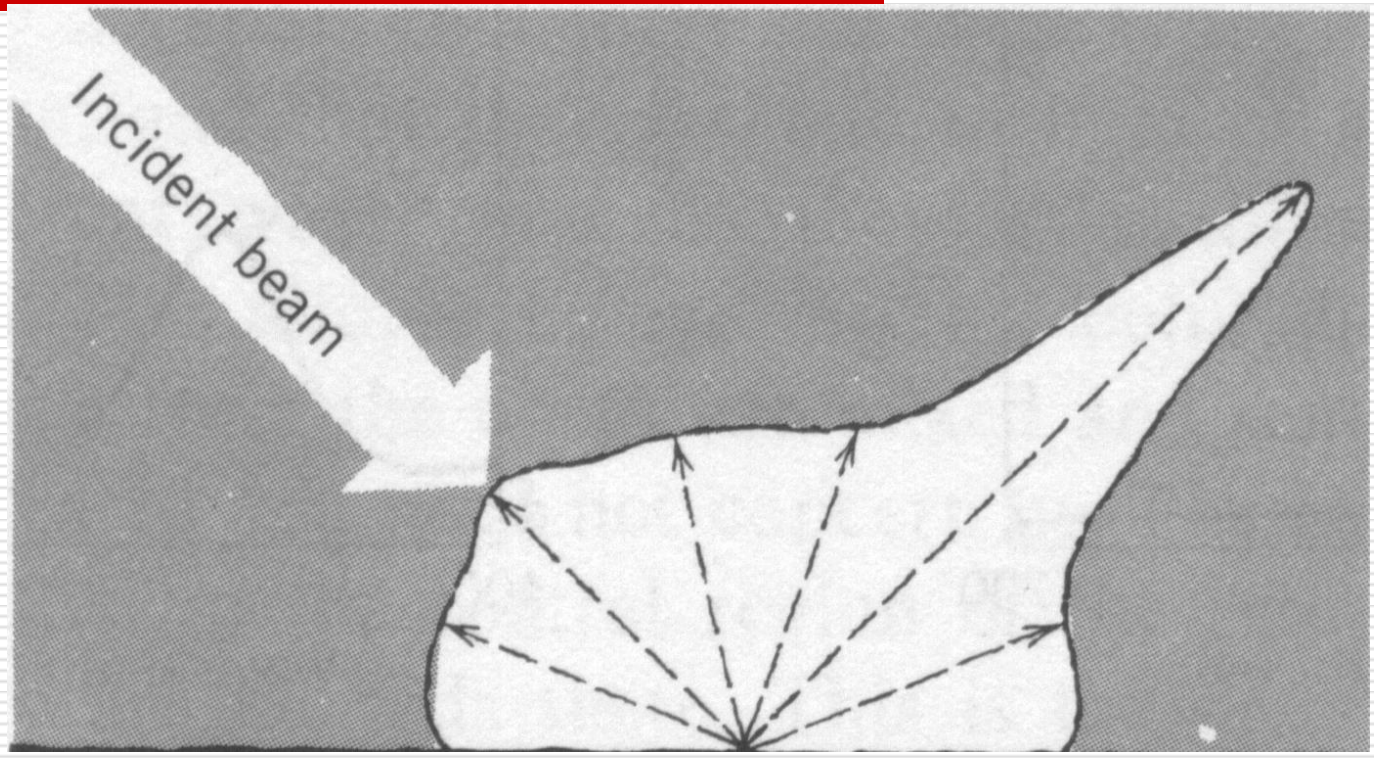


How material modify light

- Reflection
  - diffuse and ambient reflection
- Refraction

Light speed =
wave length × frequency

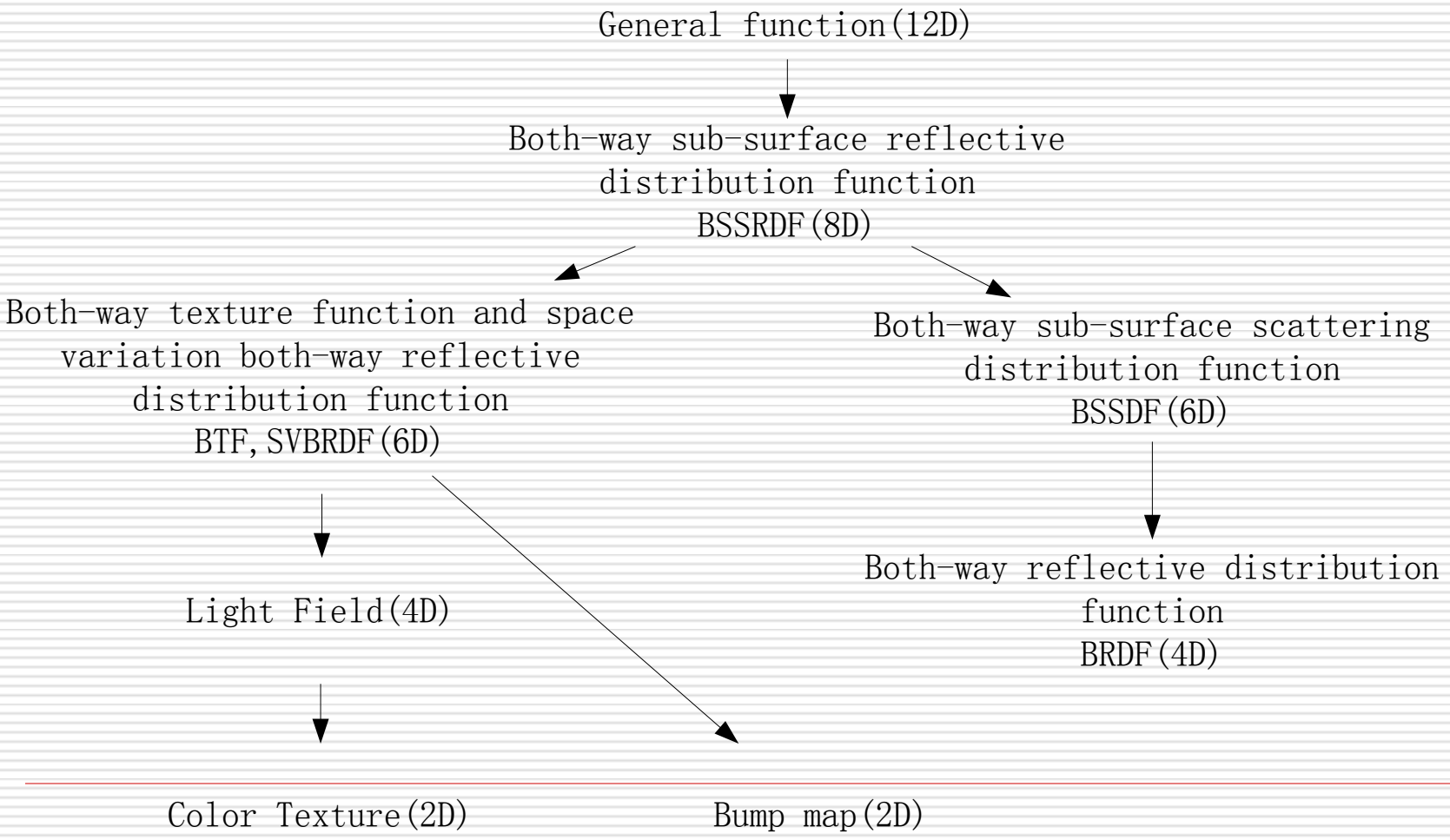


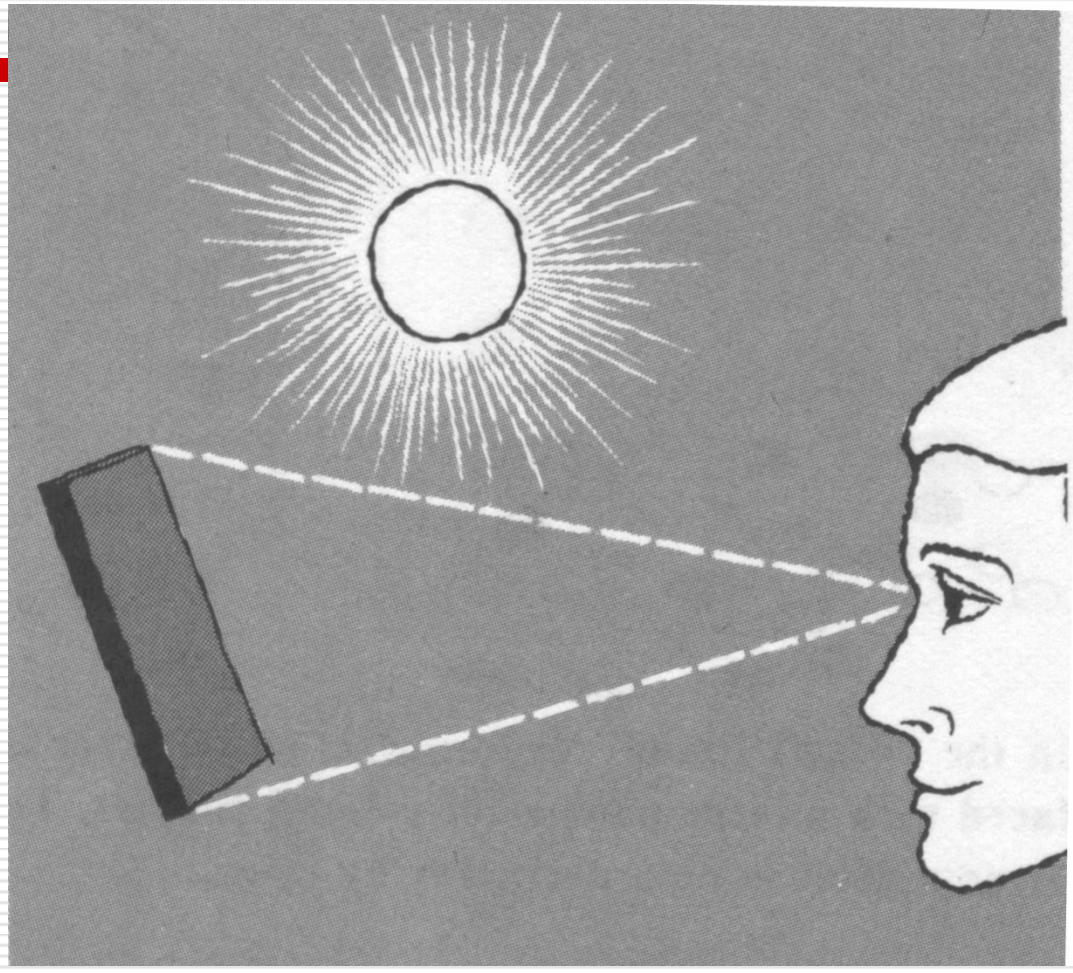

Incident beam

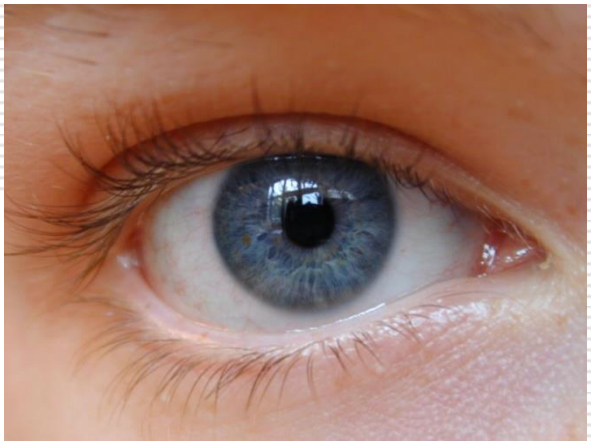
Rendering equation

- In the real world, different surfaces are with different properties, some are rough, some are smooth, some are sleek. They show different reflection effects in complex lighting conditions
- The different reflective characteristics of object surface can be used to describe the reflective properties. The complete reflective properties model can be expressed using a 12D function：incident and outgoing locations of object surface(2D), light direction(2D), time(1D), wave length(1D), incident and outgoing light(both 6D respectively), totally 12D. However, the data of 12D function is so large that it is hard to store and express. After simplification, function with lower dimension are usually be used to express reflective properties

Rendering equation







Iris

Aqueous humor

pupil

Crystalline lens

Fovea

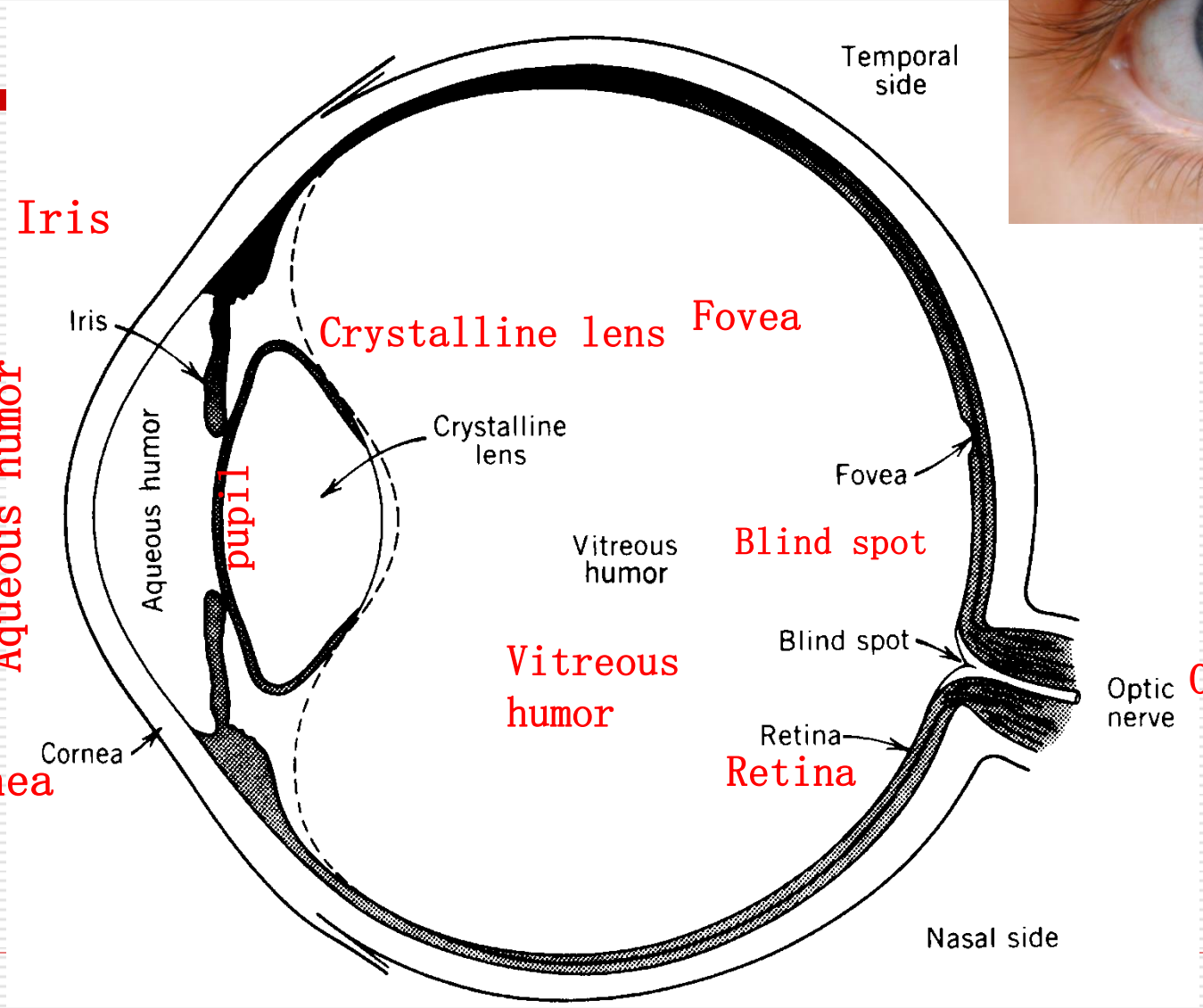
Vitreous humor

Blind spot

Optic nerve

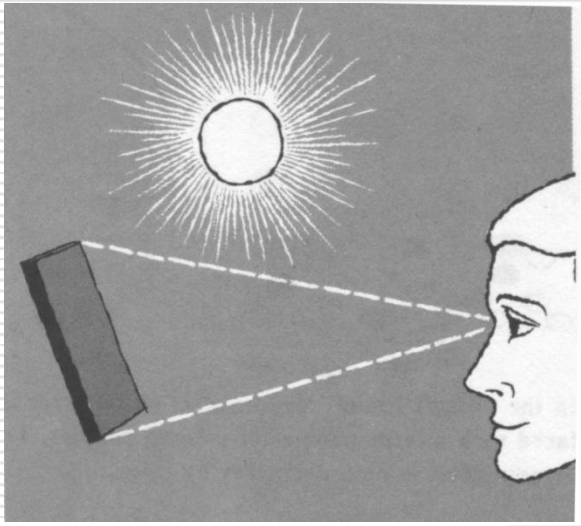
cornea

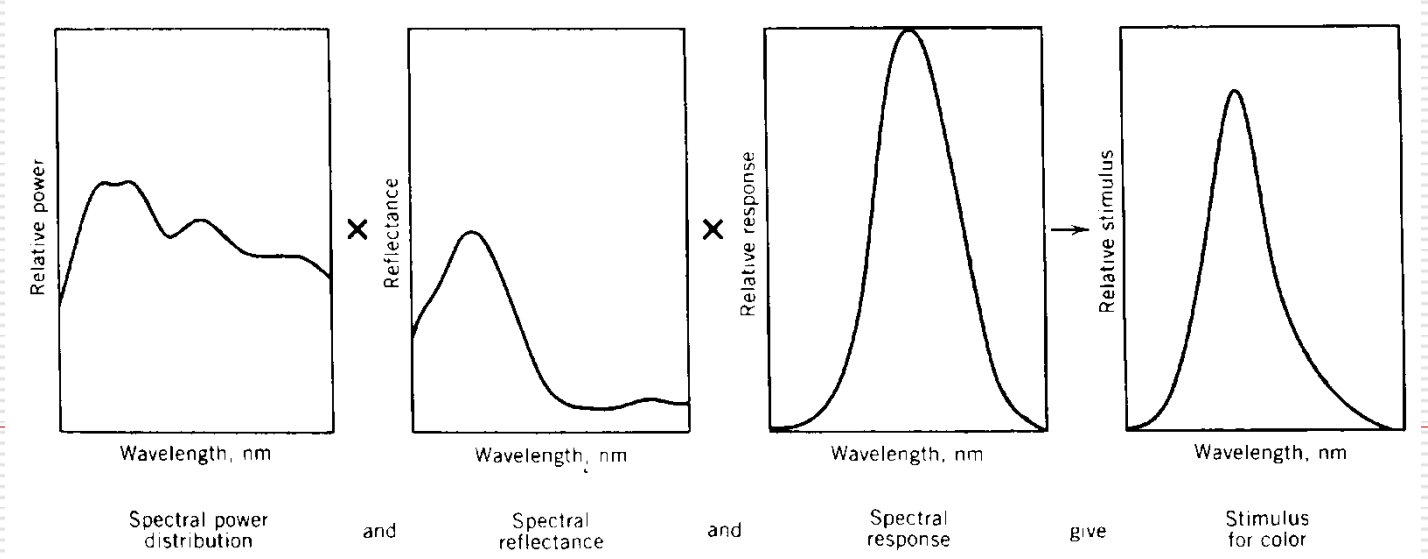
Retina



Visual Realism Requirements

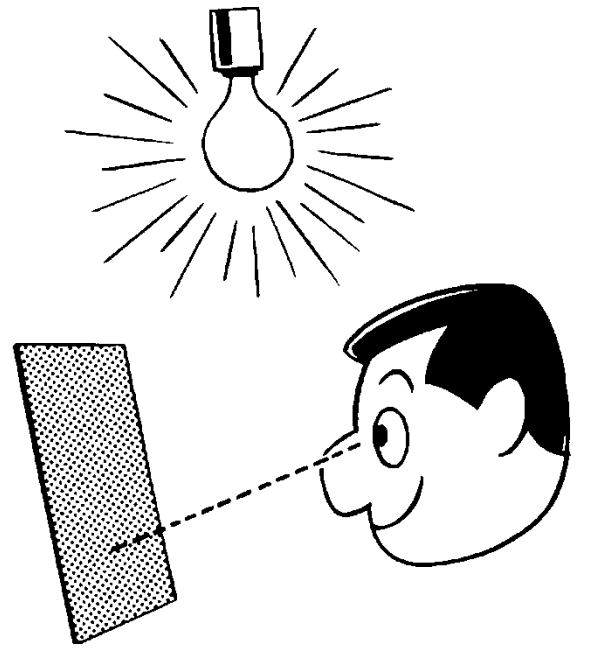
1. Light Sources
2. Materials
   (e.g., plastic, metal)
   Spectrum

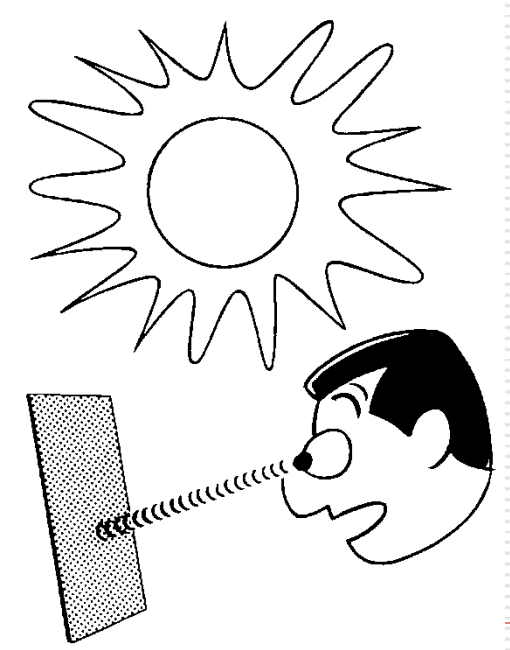




Visual Realism Requirements

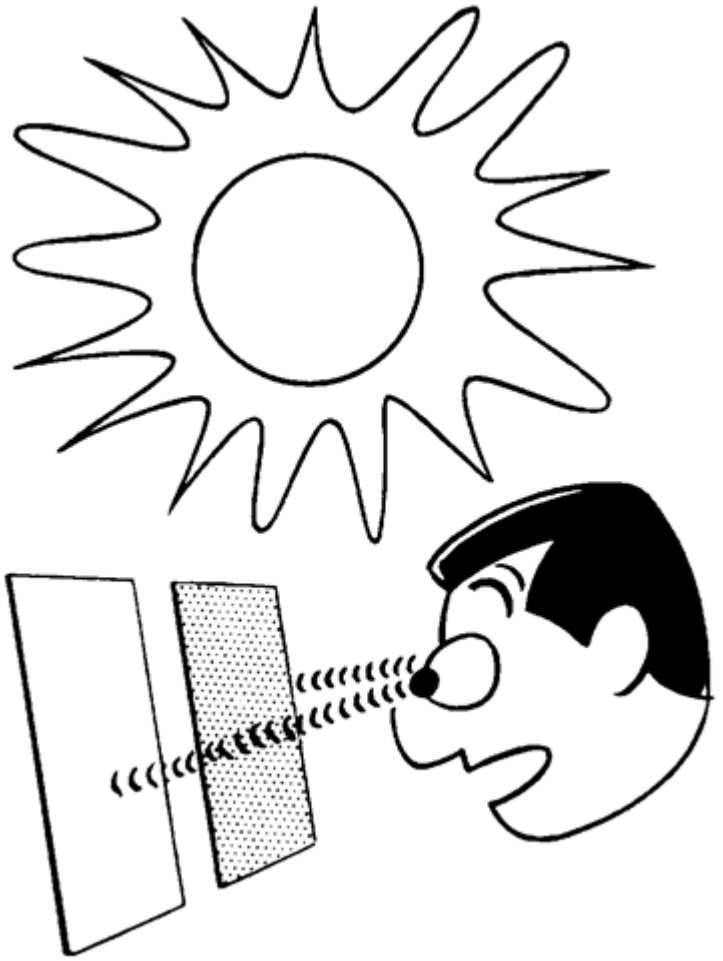
1. Light Sources: different lighting
2. Materials (e.g., plastic, metal)





Visual Realism Requirements

1. Light Sources
2. Materials:
   different material
   (e.g., plastic, metal)



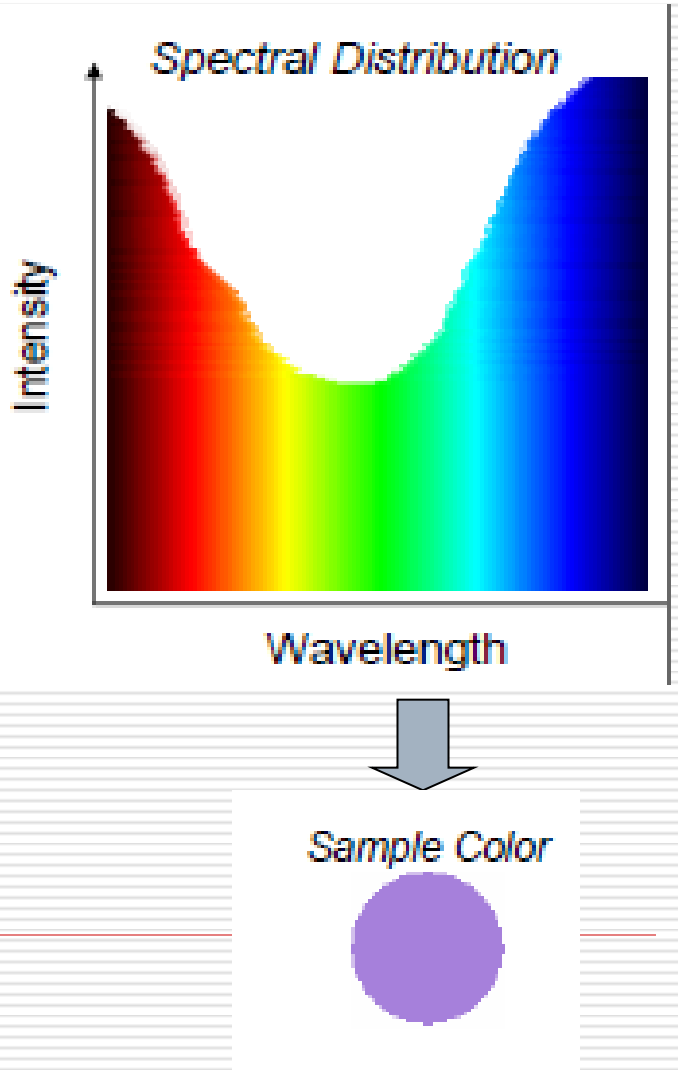
Color space

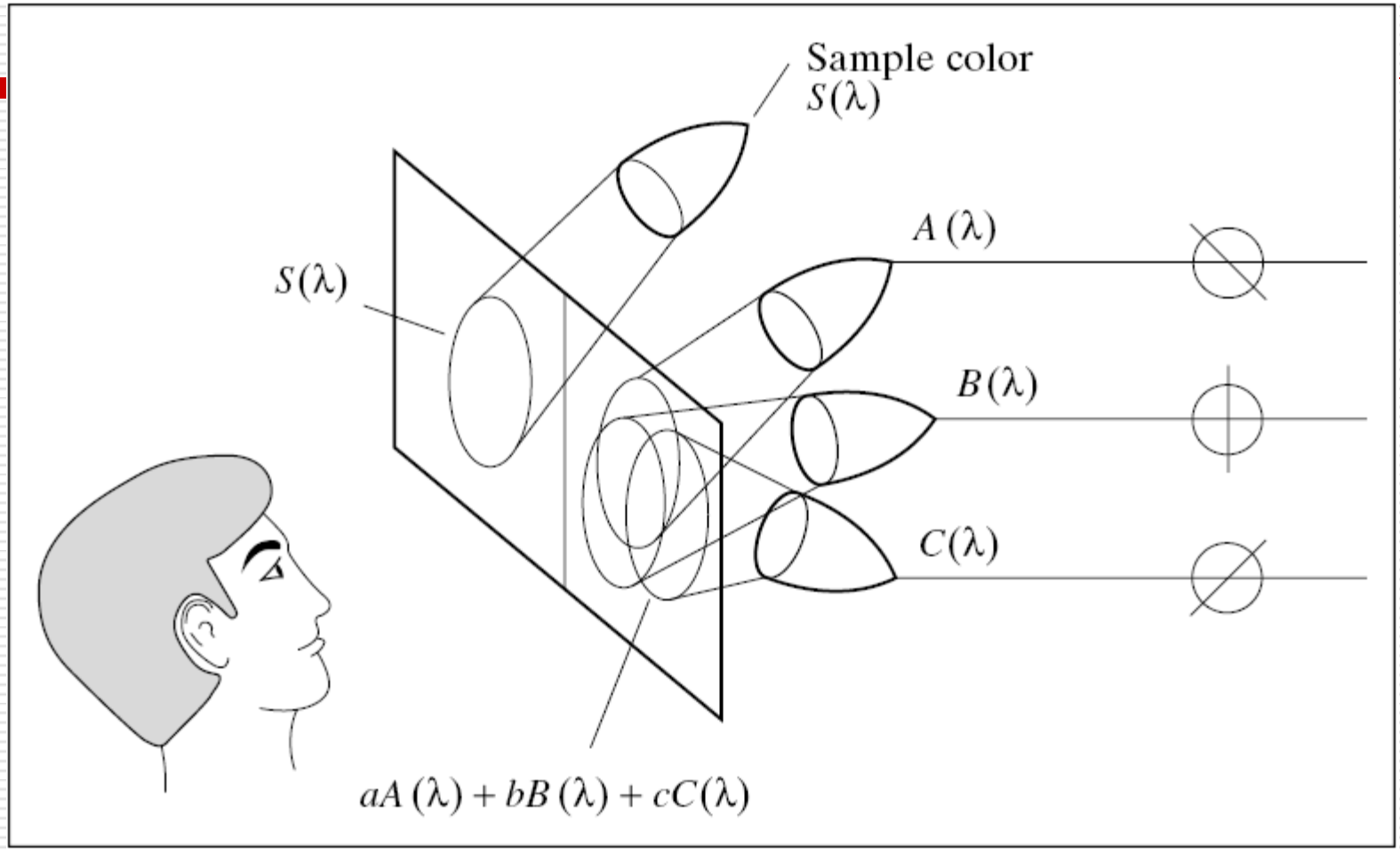
1. Using a distribution function to express a color is somewhat cumbersome and some
2. It is discovered that many colors can be expressed by three basic colors (Trichromatic theory：17, 18 centuries)

Spectral distribution of light (physics)

1. The usual sense of light actually contains many frequencies of electromagnetic waves

- Different frequency with different intensity
- Intensity with wavelength is the spectral distribution function
- The color of light can be expressed by this distribution function



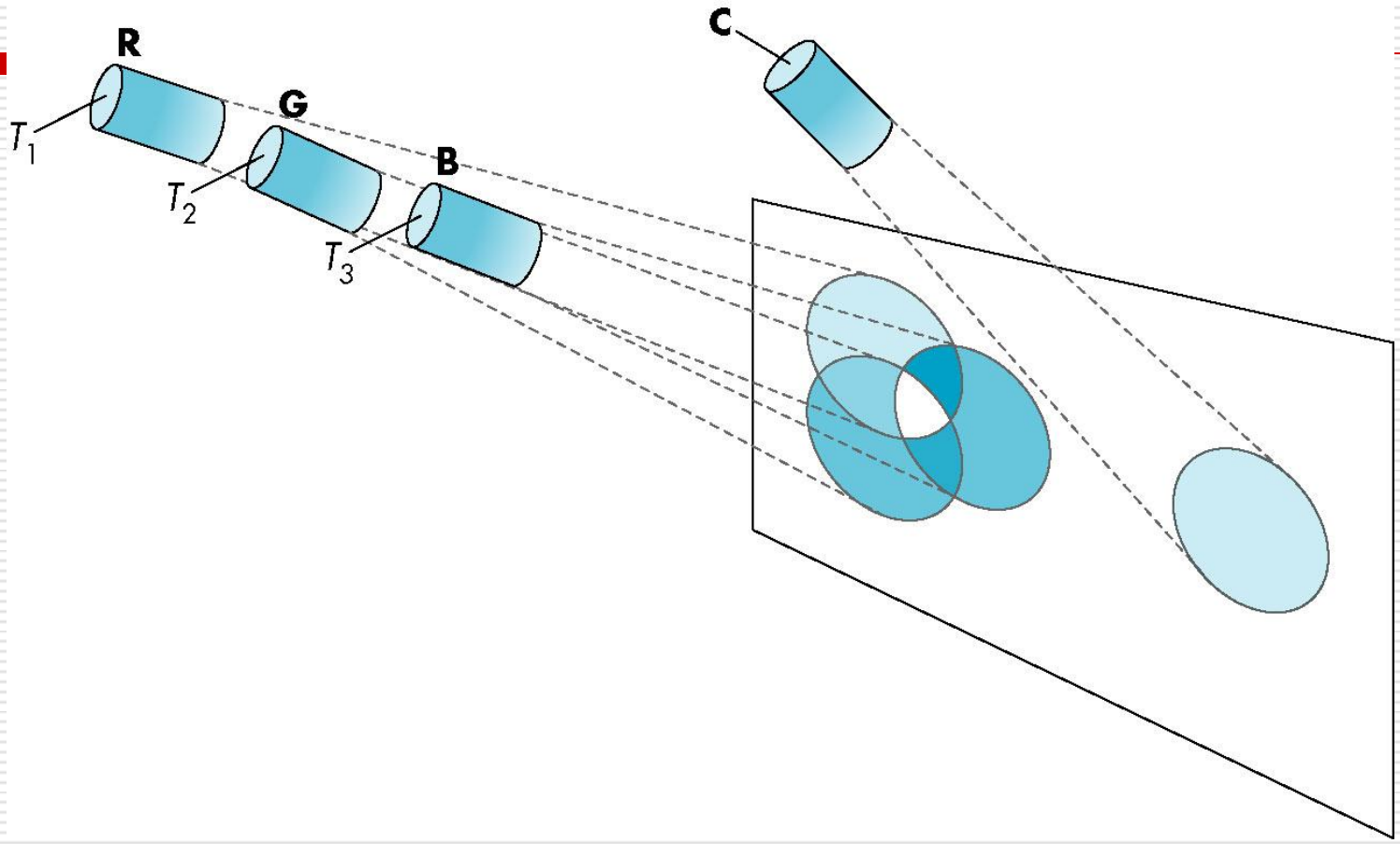

Sample color
$S(\lambda)$
$S(\lambda)$
$A(\lambda)$
$B(\lambda)$
$C(\lambda)$
$aA(\lambda) + bB(\lambda) + cC(\lambda)$

Color space

1. Using a distribution function to express a color is somewhat cumbersome and some
2. It is discovered that many colors can be expressed by three basic colors (Trichromatic theory：17, 18 centuries)
3. RGB color space
   - Use three value (r, g, b) to express color

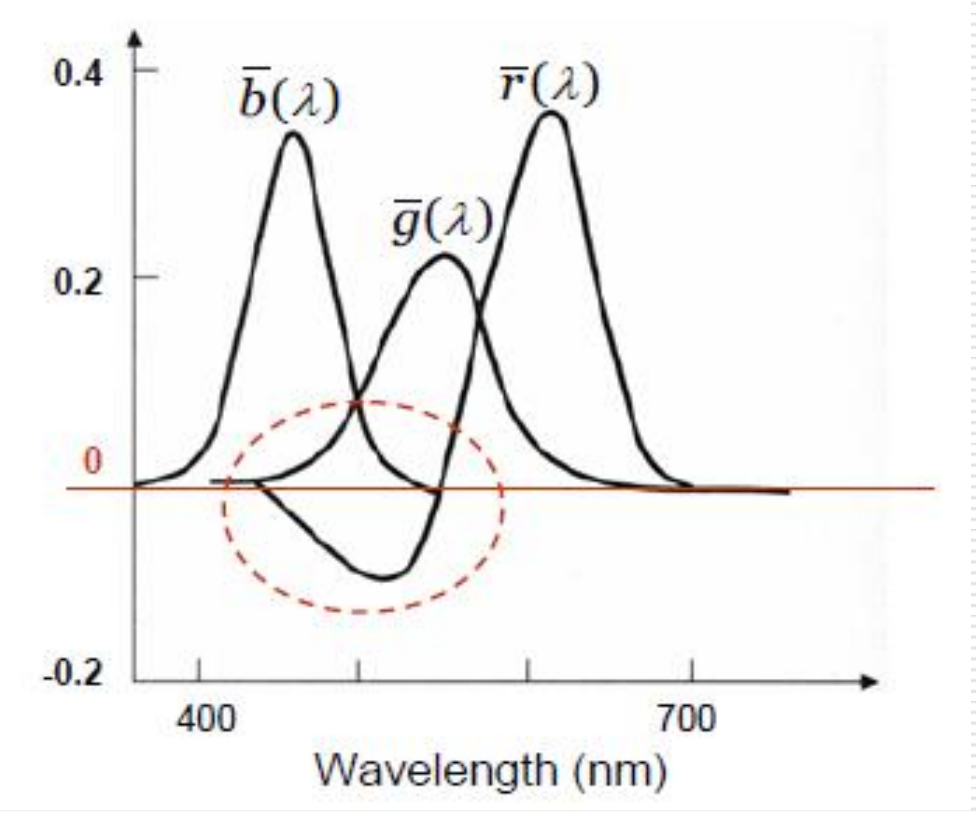
Three primary colors of human eye

1. Spectral distribution: the intensity is a function of wave length $C(\lambda)$
2. The human eye has three types of cone cells
3. The sensitivity curve of each type of cell is $S_i(\lambda)$
4. When the human eye is stimulated by visible light, the stimulation value is $A_i = \int S_i(\lambda) C(\lambda) d\lambda$
5. Therefore, the color which the brain feels is a triple (A_1, A_2, A_3)


R
G
B
C
T_1
T_2
T_3

RGB color space (2)

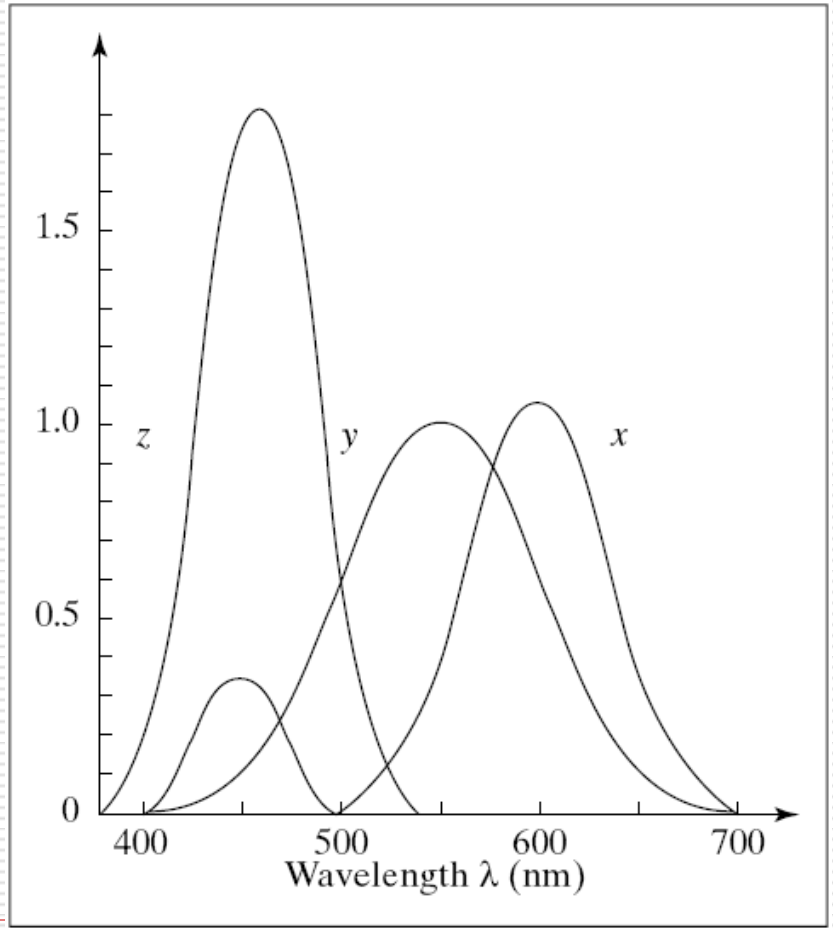
1. RGB color space is unable to express every color
2. Because the R value can not be negative in practical application



XYZ color space

The CIE model capitalises on this fact by defining Y as luminance. Z is quasi-equal to blue stimulation, or the S cone response, and X is a mix (a linear combination) of cone response curves chosen to be nonnegative.

Defining Y as luminance has the useful result that for any given Y value, the XZ plane will contain all possible chromaticities at that luminance



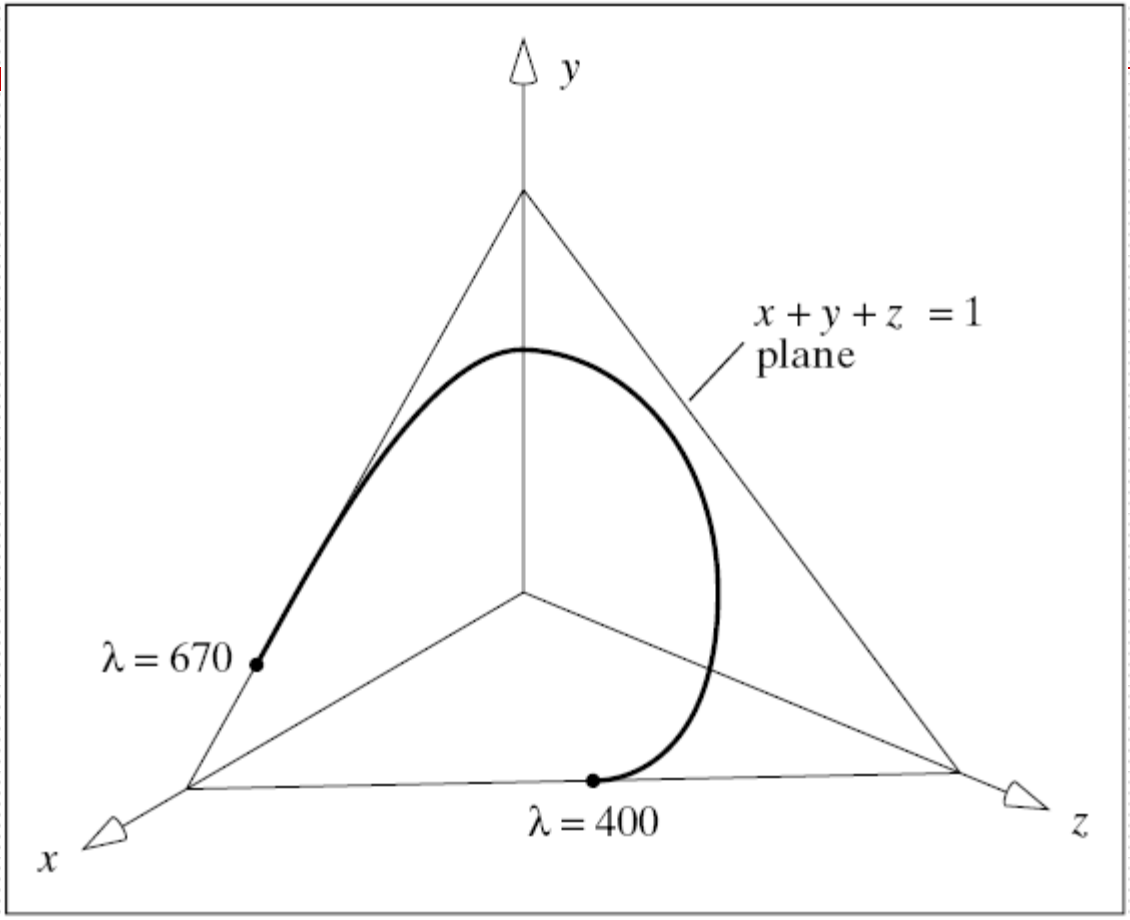
$$mono(\lambda) = x(\lambda)X + y(\lambda)Y + z(\lambda)Z$$

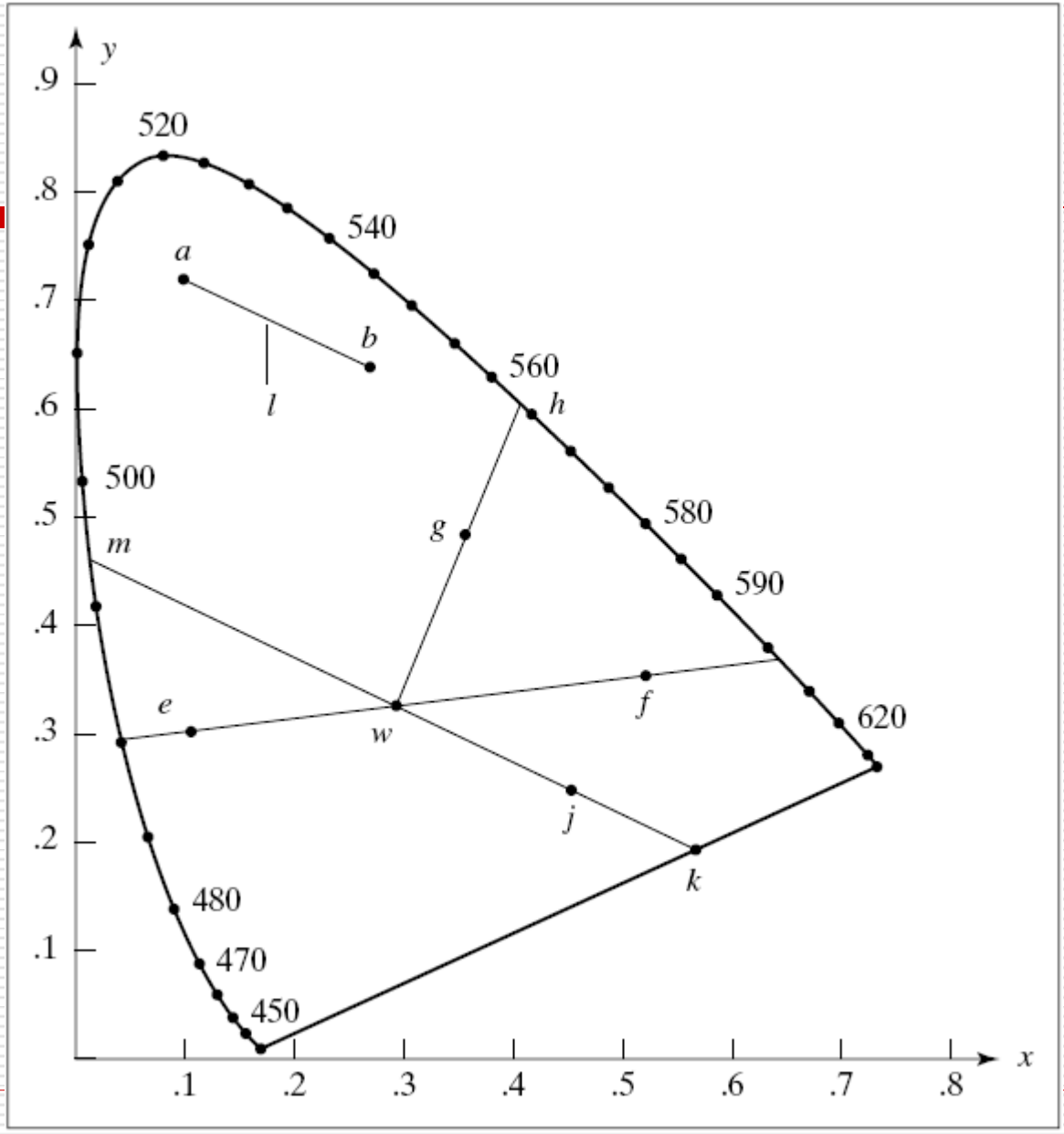
$$\bar{x}(\lambda) = \frac{x(\lambda)}{x(\lambda) + y(\lambda) + z(\lambda)}$$

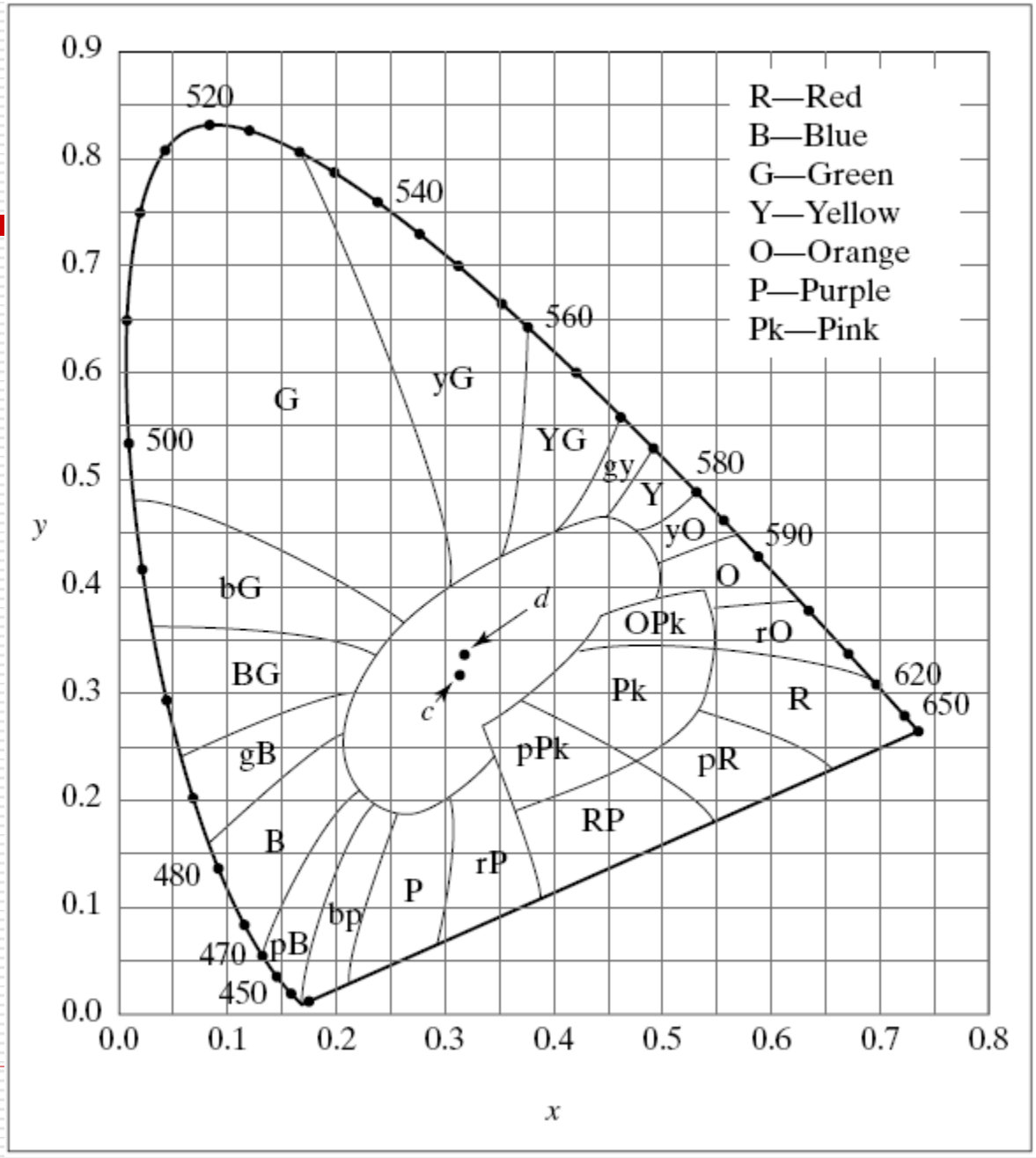
$$\bar{y}(\lambda) = \frac{y(\lambda)}{x(\lambda) + y(\lambda) + z(\lambda)}$$

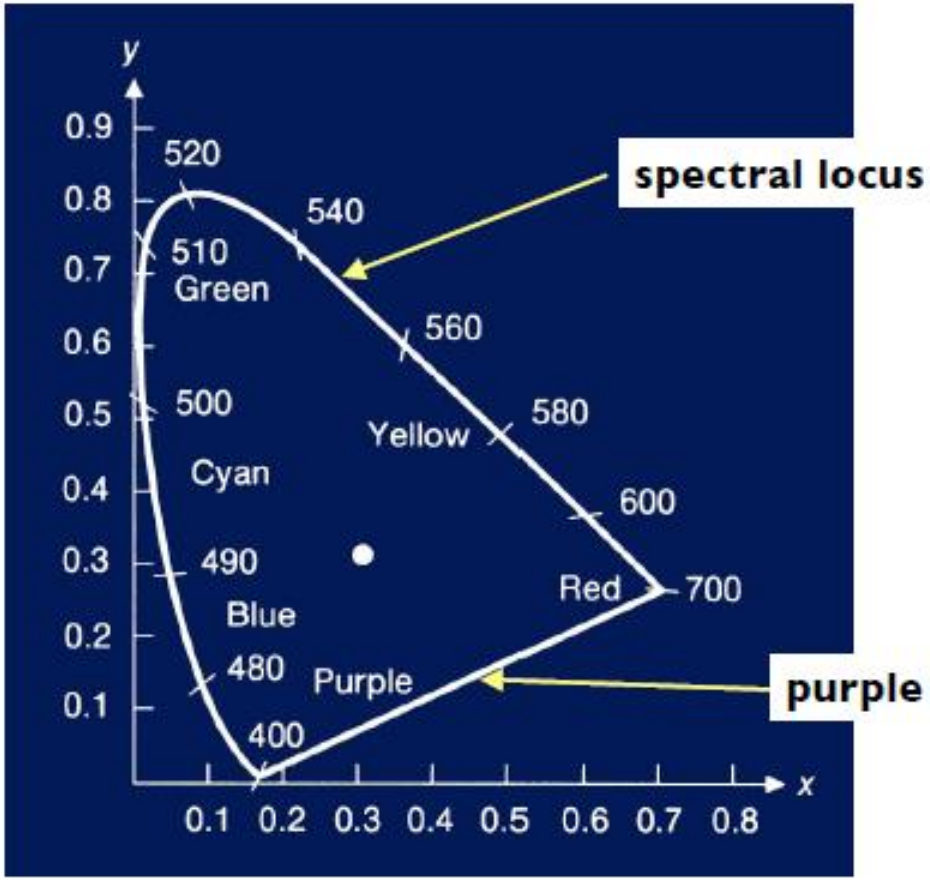
$$\bar{z}(\lambda) = \frac{z(\lambda)}{x(\lambda) + y(\lambda) + z(\lambda)}$$

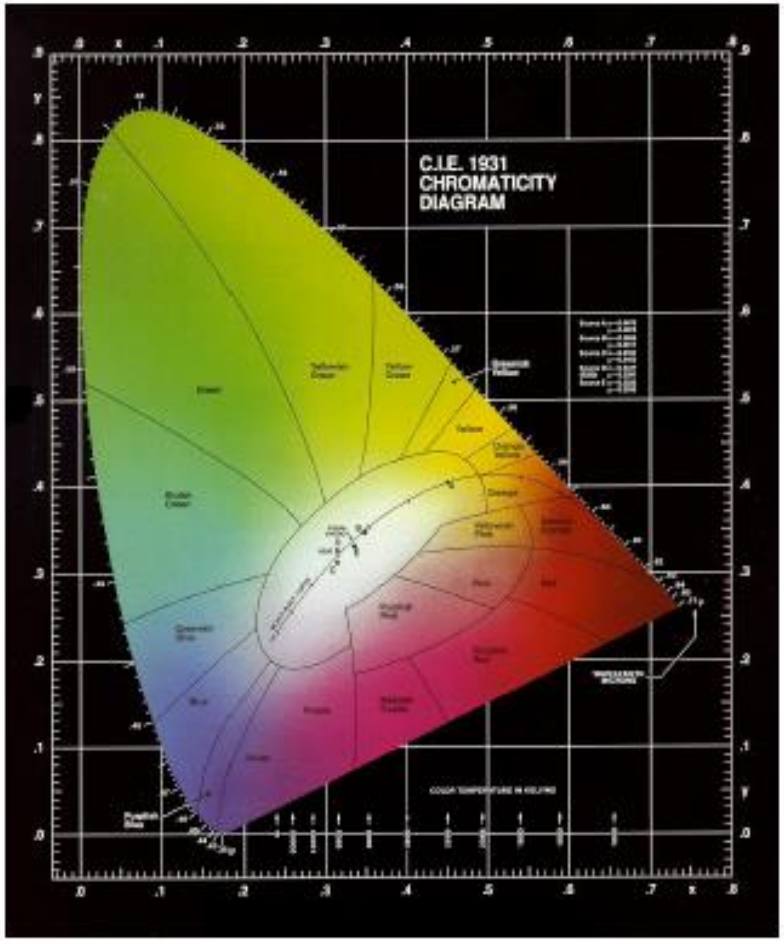
$$S(\lambda) = (\bar{x}(\lambda), \bar{y}(\lambda), \bar{z}(\lambda)) \qquad \bar{z}(\lambda) = 1 - \bar{x}(\lambda) - \bar{y}(\lambda)$$


y
x + y + z = 1
plane
λ = 670
λ = 400
x
z


y
.9
.8
.7
.6
.5
.4
.3
.2
.1
520
540
560
580
590
620
500
480
470
450
a
b
l
h
g
m
e
w
f
j
k
.1
.2
.3
.4
.5
.6
.7
.8
x


R—Red
B—Blue
G—Green
Y—Yellow
O—Orange
P—Purple
Pk—Pink
520
540
560
500
580
590
620
650
480
470
450
G
yG
YG
gy
Y
yO
O
OPk
rO
R
Pk
pR
RP
rP
P
bp
pB
B
gB
BG
bG
pPk
d
c
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
y
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
x


y
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
520
540
510
Green
560
500
580
Yellow
Cyan
600
490
Red
700
Blue
480
Purple
400
x
0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8
spectral locus
purple

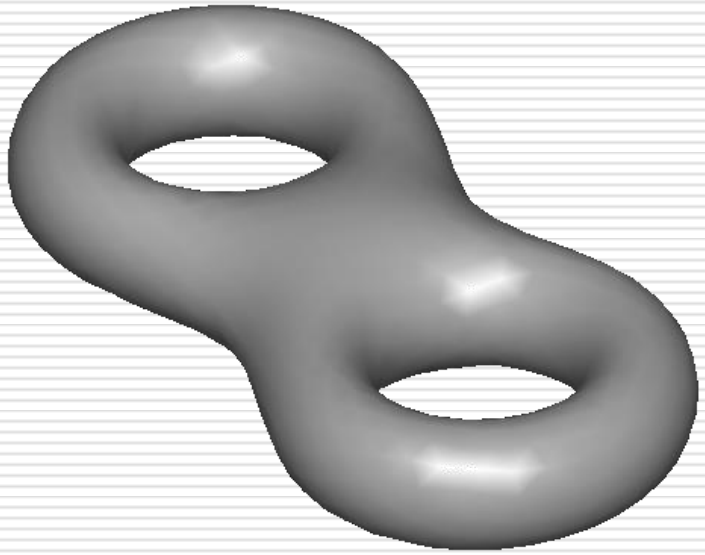

C.I.E. 1931
CHROMATICITY
DIAGRAM

Color space

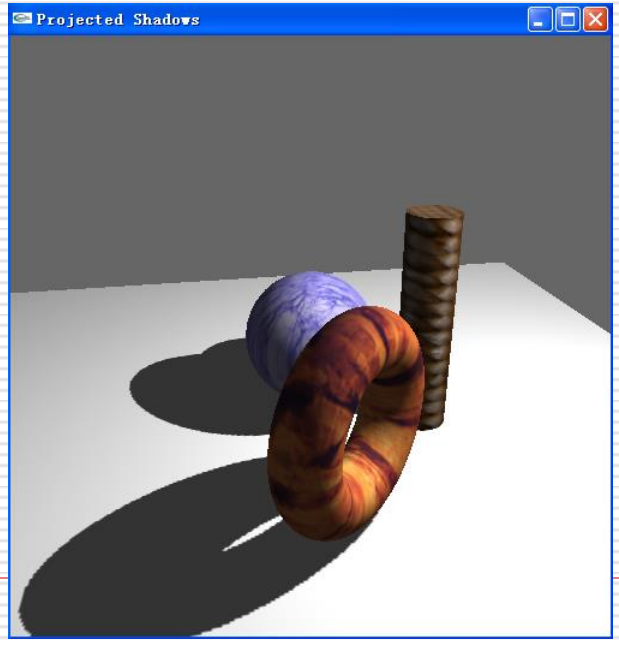
1. RGB
2. CIE XYZ
3. HSV
   - Hue, Saturation, Value
4. CMY
   - complementary color of red, green and blue
   - Cyan, Magenta and Yellow

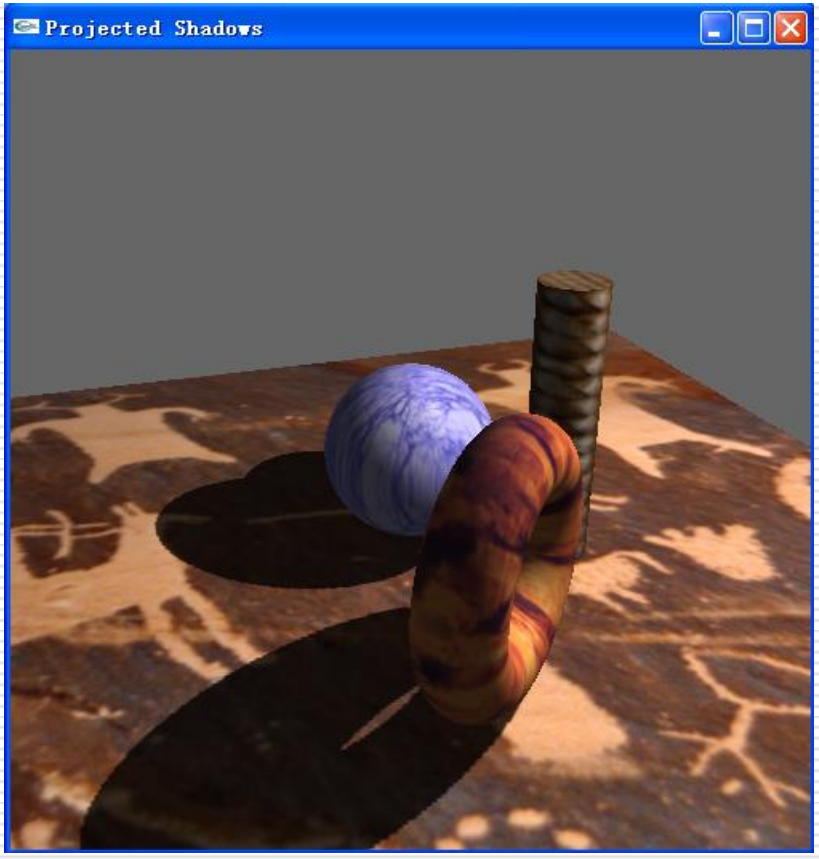
Visual Realism Requirements

1. Light Sources
2. Materials (e.g., plastic, metal)
3. Shading Models




Projected Shadows


Projected Shadows


Projected Shadows

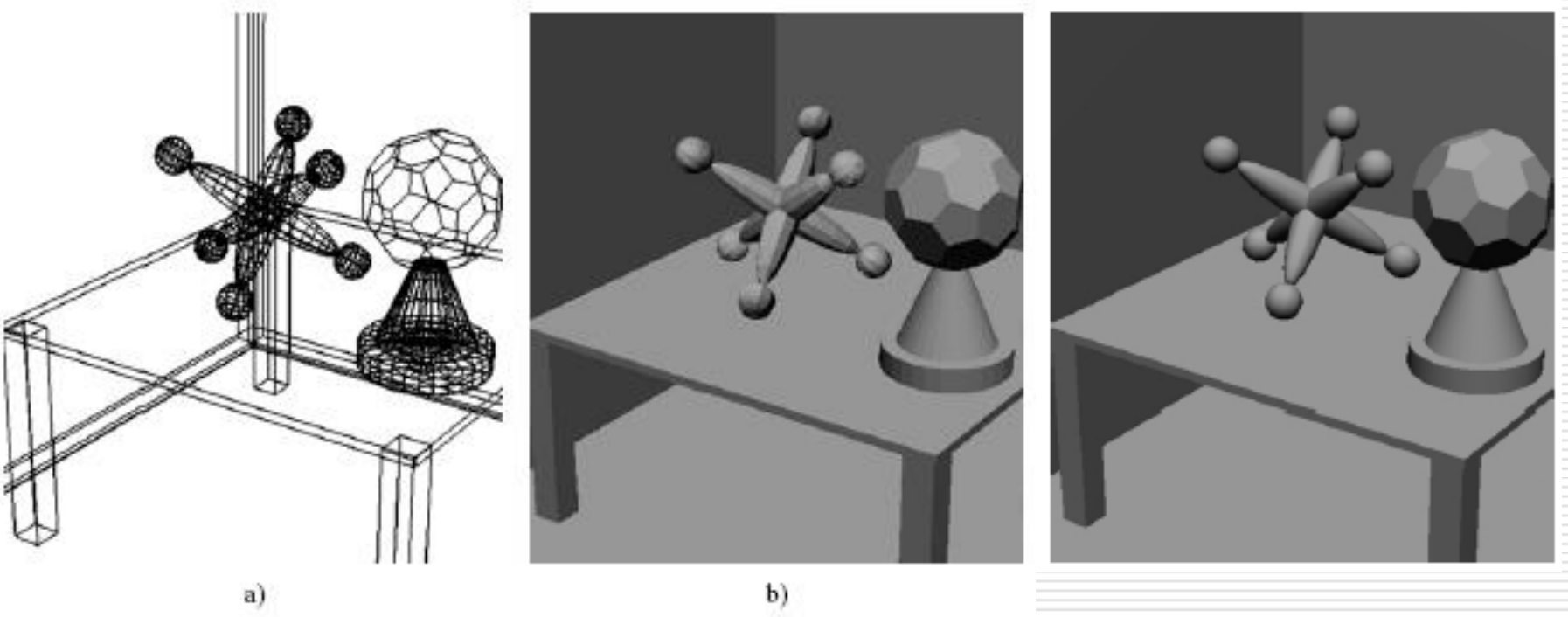
Rendering 3D Objects

Much of rendering is based on different shading models, which describe how light from light sources interacts with objects in a scene

- It is impractical to simulate all of the physical principles of light scattering and reflection
- A number of approximate models have been invented that do a good job and produce various levels of realism

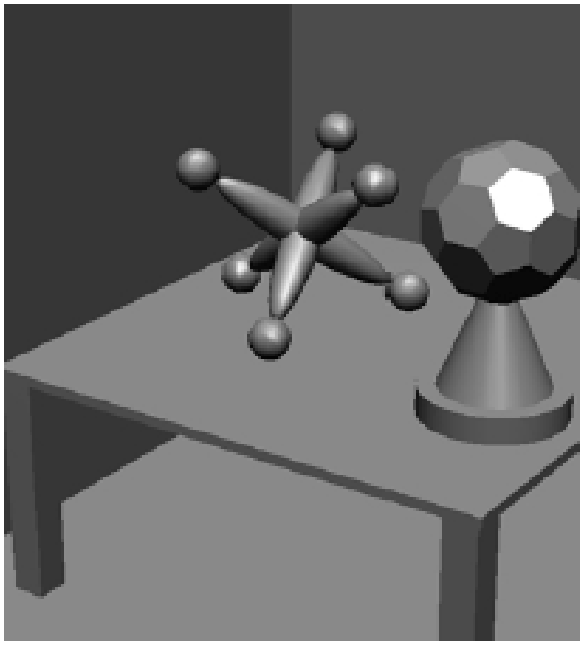
OpenGL rendering example

wire-frame flat shading smooth shading

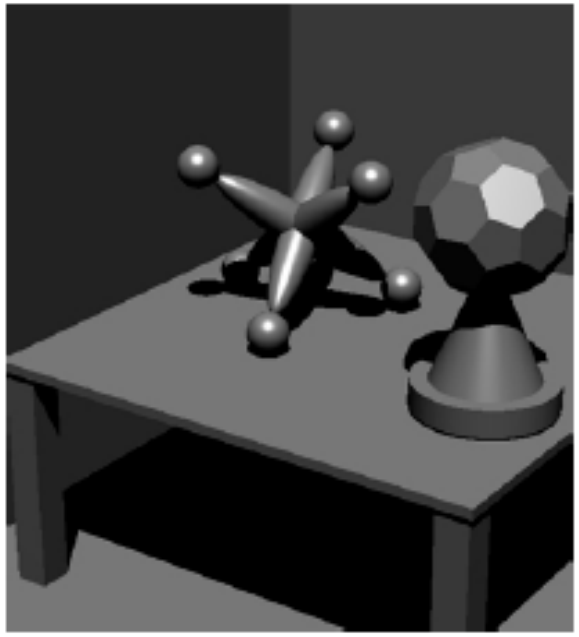


OpenGL rendering example

specular highlights,



shadows,

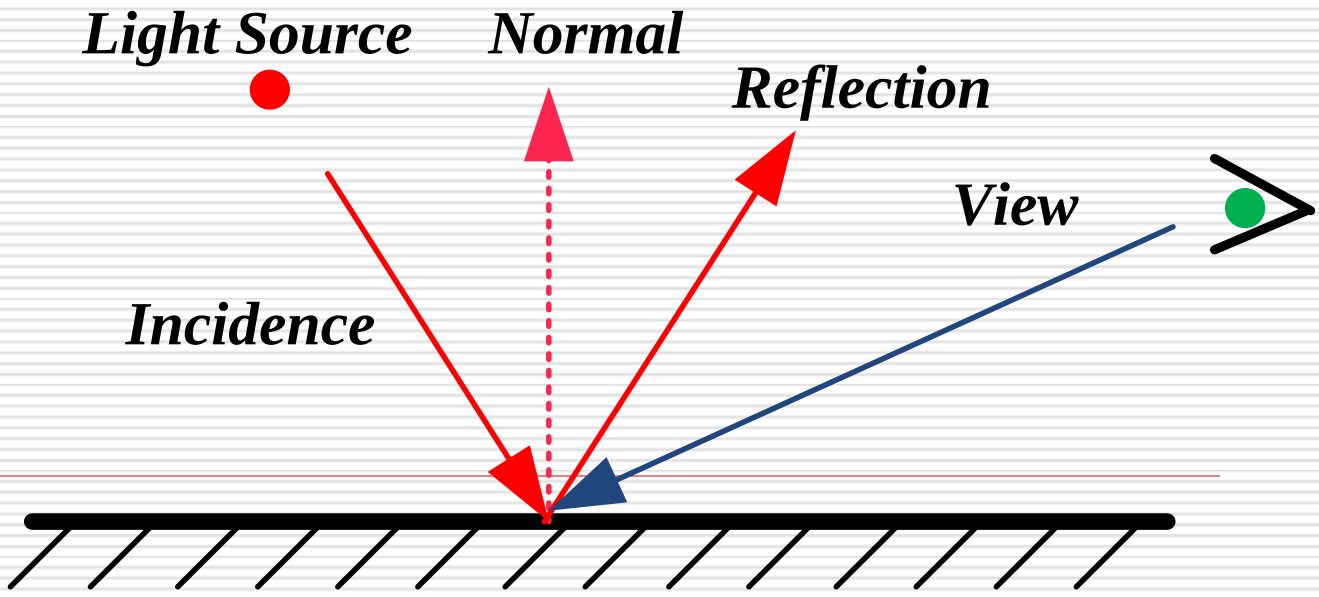


textures

A simple model of light propagation

Reflection

- The ideal angle of incidence equals the angle of reflection
- Incident, normal and reflective lights are in the same plane



A simple model of light propagation

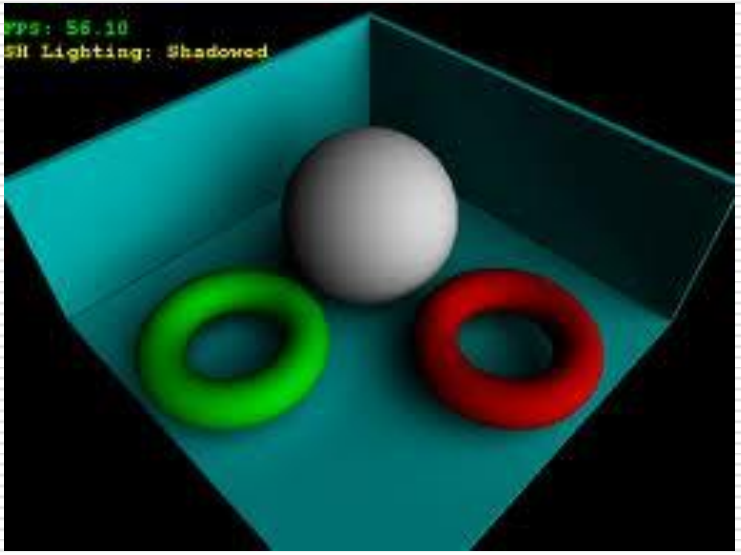
Conservation of energy

$$I_i = I_d + I_s + I_t + I_v$$

- $\mathbf{I_i}$: energy of incident light
- $\mathbf{I_d}$: energy of diffuse light
- $\mathbf{I_s}$: energy of specular light
- $\mathbf{I_t}$: energy of refraction
- $\mathbf{I_v}$: energy absorbed

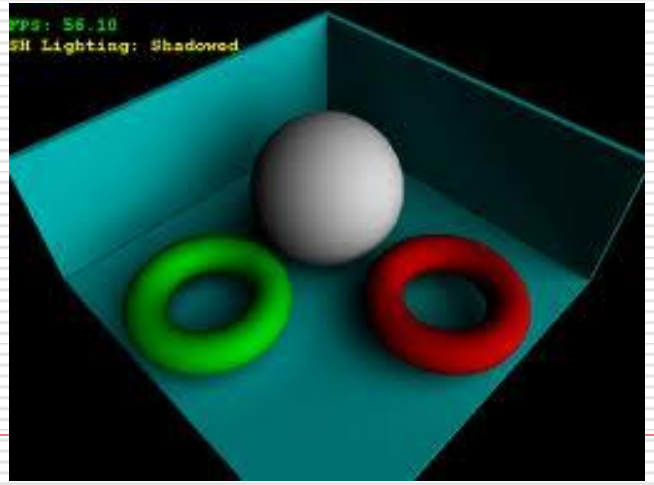
OpenGL light model

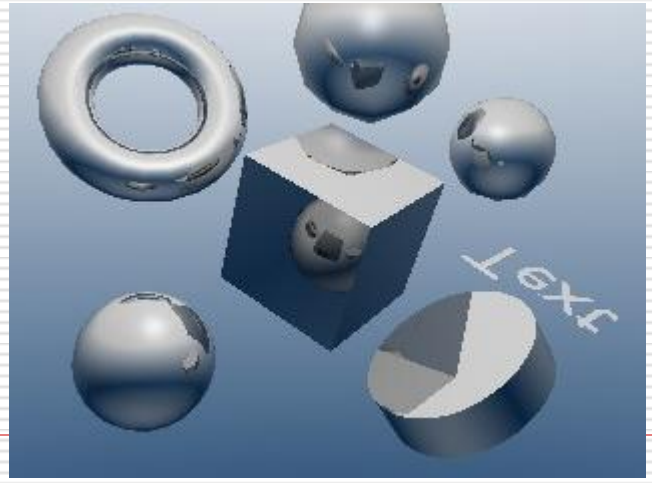
- Light source
  - Point light
  - Direction light
- Three components
  - Diffuse light
  - Specular light
  - Ambient light (approximate the effect of global light roughly)



OpenGL light model

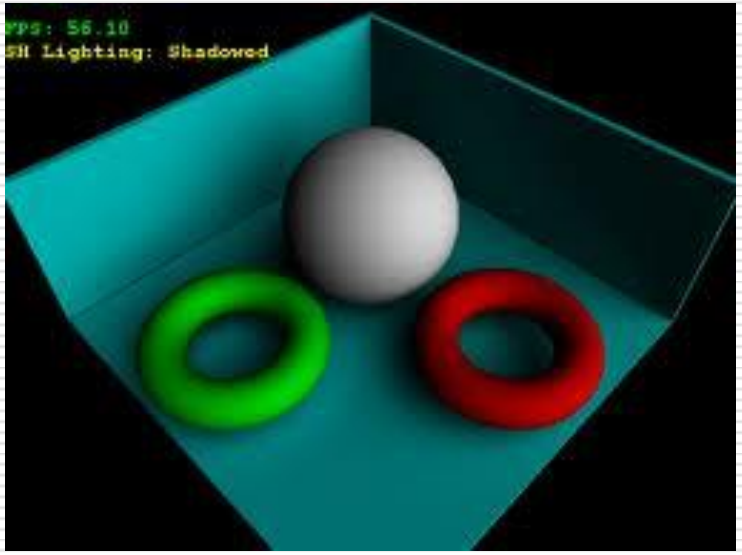
- Model is not completely physically correct
- But it provides fast and relatively good results on the screen

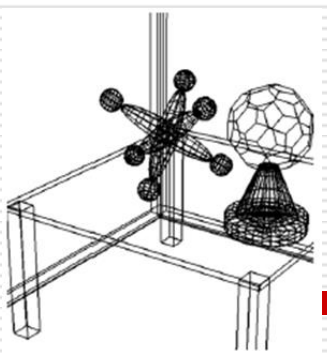


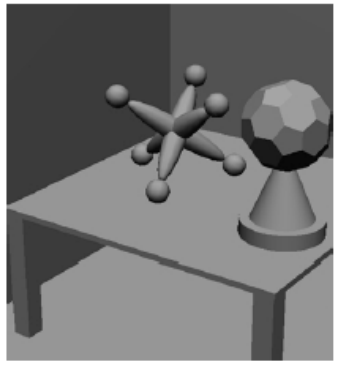


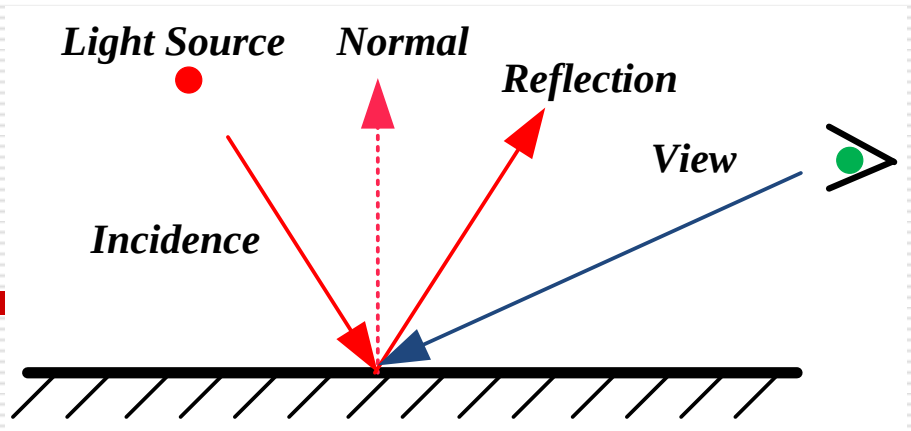
OpenGL light model

- Light source
  - Point light
  - Direction light
- Three components
  - Diffuse light
  - Specular light
  - Ambient light (approximate the effect of global light roughly)





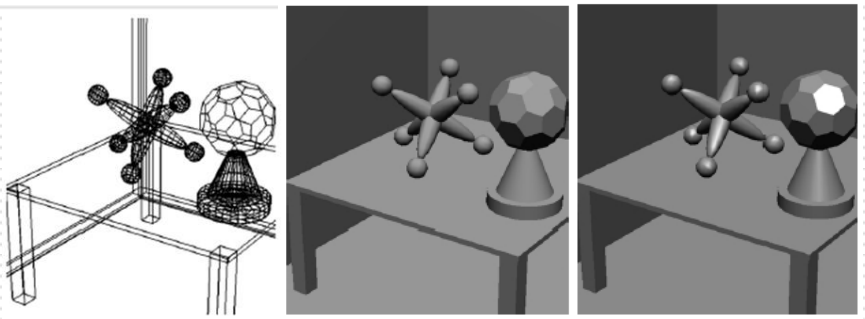


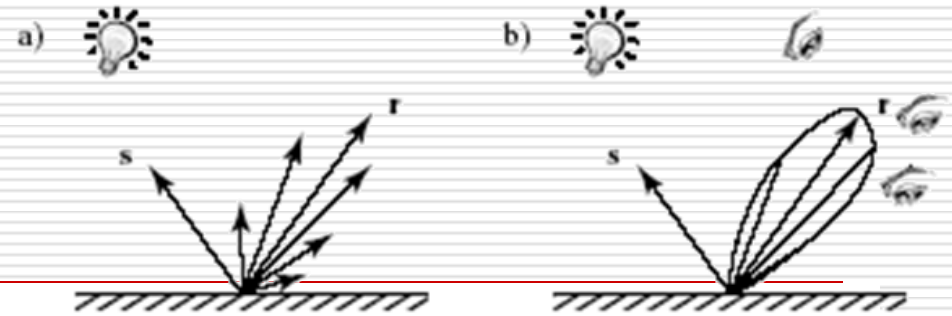


- Diffuse component
  - Diffuse light evenly spreads in all directions, has nothing to do with view point
  - According Lambert cosine law, diffuse light intensity is

$$I_d = I_i K_d * (L \cdot N)$$

  - K_d is the diffuse reflection coefficient of object





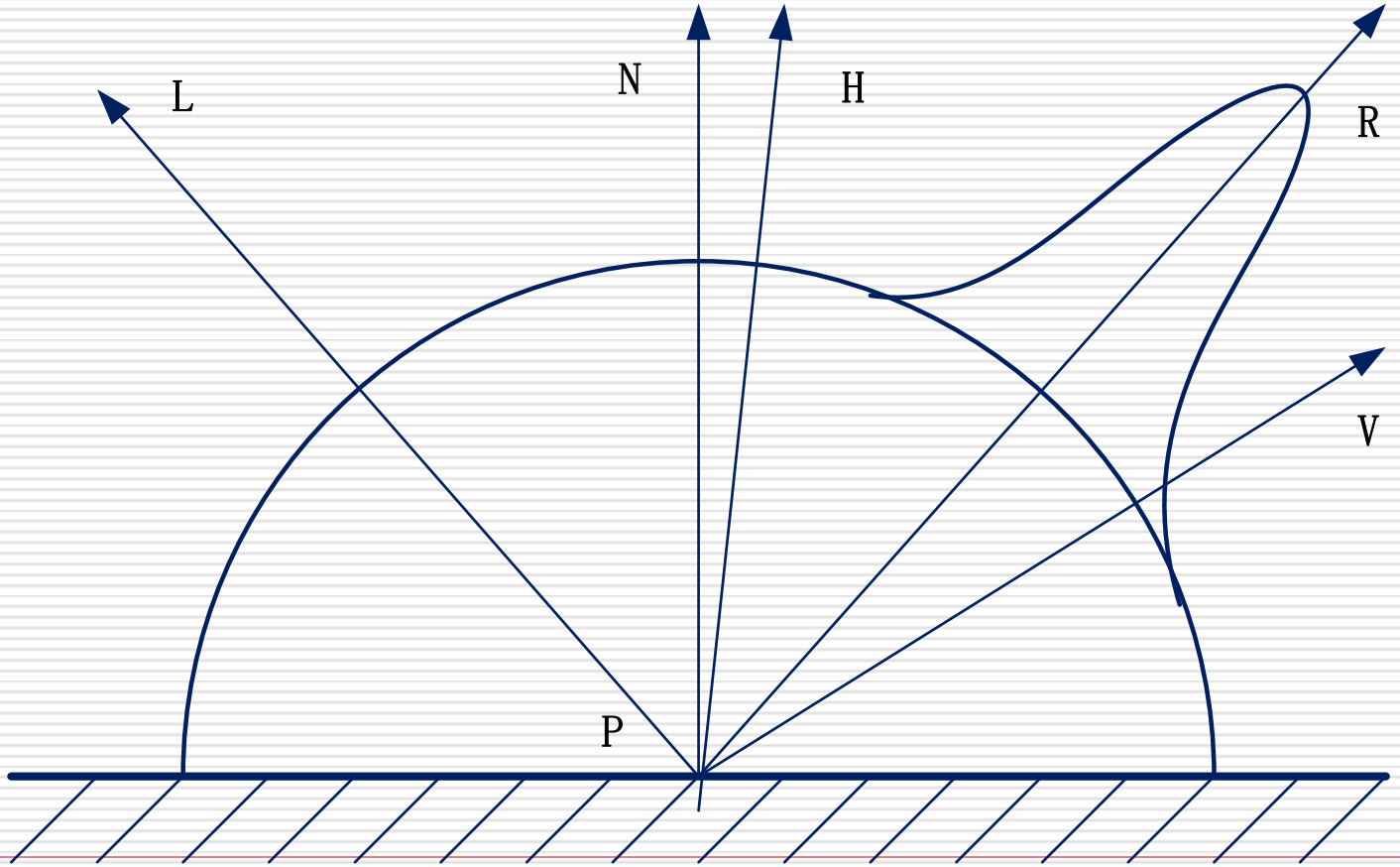
Specular component

- For general smooth surface, light is reflected in a limited range, and the intensity along the reflected light direction decided by the law of reflection is maximum
- The intensity of specular light is

$$I_s = I_i K_s * (R \cdot V)^n$$

- K_s is the specular coefficient of object, n is reflection index, representing the gloss level of object surface, the greater the number, the smoother the surface

Specular component

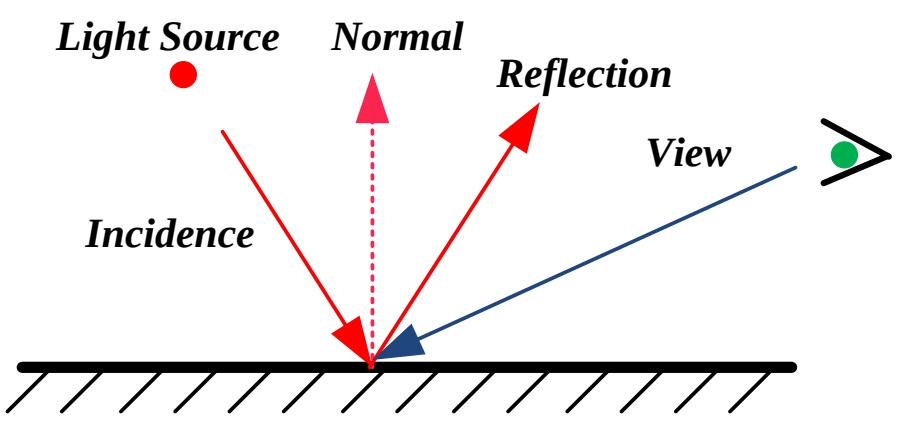


OpenGL light model

- Ambient component
  - Approximate global light effect roughly
  - Define by

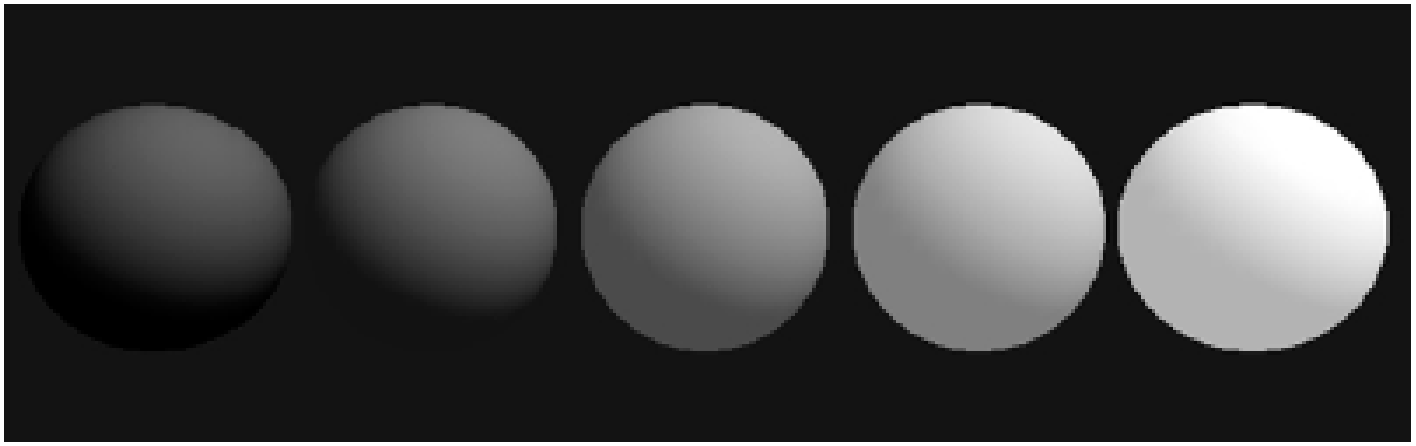
$$I_a = I_i K_a$$

  - K_a is ambient coefficient



OpenGL light model

- Adding ambient component to diffuse component
  - Different ambient coefficient K_a
  - Modest ambient light softens shadows; too much ambient light washes out shadows



OpenGL light model

- Phong light model in OpenGL is the sum of the three component above

$$I = I_i K_a + I_i K_s * (R \cdot V)^n + I_i K_d * (L \cdot N)$$

Phong light model

Diffuse

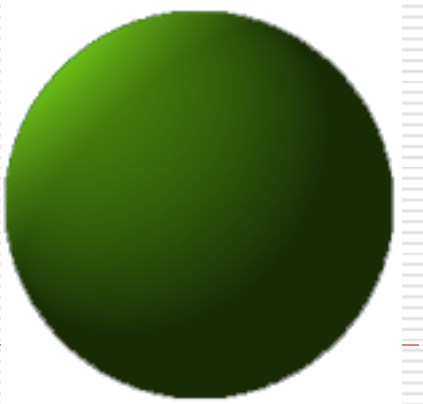
+

Ambient

+

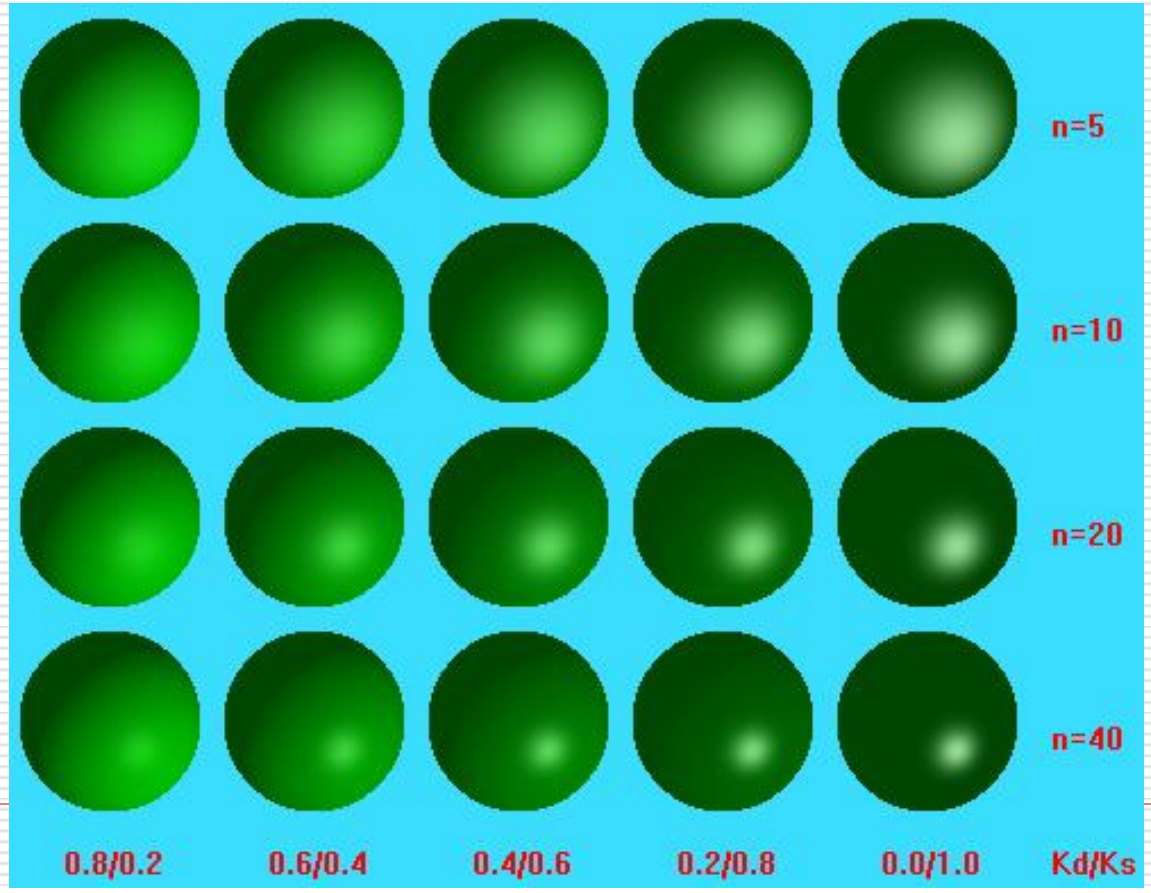
Specular

=



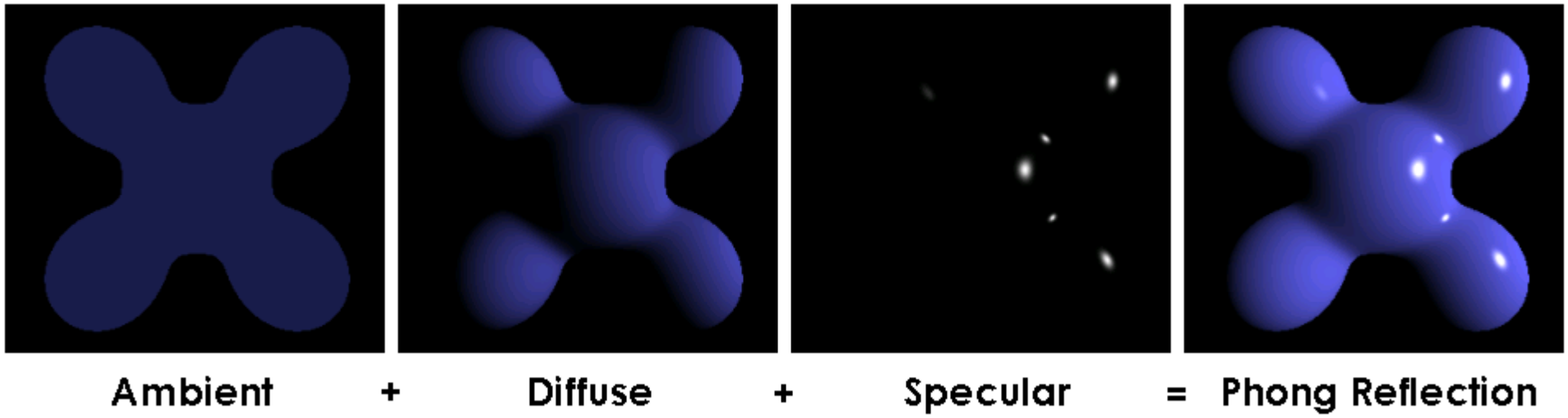
Phong light model

- Different parameters



Phong light model

- LocalColor =
  $I * (ka + kd*(N \cdot L) + ks*(N \cdot H)^n)$



Use light source in OpenGL

- Light sources are given a number in [0, 7]: GL_LIGHT_0, GL_LIGHT_1, etc.
- Each light has a position specified in homogeneous coordinates using a GLfloat array named, for example, litePos
- The light is created using glLightfv (GL_LIGHT_0, GL_POSITION, litePos)
- If the position is a vector (4th component = 0), the source is infinitely remote (like the sun)

Use light source in OpenGL

- The light color is specified by a 4-component array [R, G, B, A] of GLfloat, named (e.g.) amb0. The A value can be set to 1.0 for now.
- The light color is specified by glLightfv (GL_LIGHT_0, GL_AMBIENT, amb0); Similar statements specify GL_DIFFUSE and GL_SPECULAR

Use light source in OpenGL

- Lights do not work unless you turn them on.
  - In your main program, add the statements glEnable (GL_LIGHTING); glEnable (GL_LIGHT_0);
  - If you are using other lights, you will need to enable them also
- To turn off a light, glDisable(GL_LIGHT_0);
- To turn them all off, glDisable (GL_LIGHTING)

Creating an Entire Light

```
// define some colors
GLfloat amb0[ ] = {0.2, 0.4, 0.6, 1.0};
GLfloat diff0[ ] = {0.8, 0.9, 0.5, 1.0};
GLfloat spec0[ ] = { 1.0, 0.8, 1.0, 1.0};
// attach them to LIGHT0
glLightfv(GL_LIGHT0, GL_AMBIENT, amb0);
glLightfv(GL_LIGHT0, GL_DIFFUSE, diff0);
glLightfv(GL_LIGHT0, GL_SPECULAR, spec0);
```

Example

- Suppose a sphere has ambient and diffuse reflection coefficients (0.8, 0.2, 0.1), so it appears mostly red when bathed in white light.
- We illuminate it with a greenish light I_s = (0.15, 0.7, 0.15)
- The reflected light is then given by (0.12, 0.14, 0.015), which is a fairly even mix of red and green, and would appear yellowish.
  0.12 = 0.8 x 0.15
  0.14 = 0.2 x 0.7
  0.015 = 0.1 x 0.15

Example

- If the color of a sphere is 30% red, 45% green, and 25% blue, it makes sense to set its ambient and diffuse reflection coefficients to $(0.3K, 0.45K, 0.25K)$ respectively, where K is some scaling value that determines the overall fraction of incident light that is reflected from the sphere.
- Now if it is bathed in white light having equal amounts of red, green, and blue ($I_{sr} = I_{sg} = I_{sb} = I$) the individual diffuse components have intensities $I_r = 0.3\ K\ I$, $I_g = 0.45\ K\ I$, $I_b = 0.25\ K\ I$, so as expected we see a color that is 30% red, 45% green, and 25% blue

Use light source in OpenGL

- Global ambient light is present even if no lights are created. Its default color is {0.2, 0.2, 0.2, 1.0}.
- To change this value, create a GLfloat array of values newambient and use the statement glLightModelfv (GL_LIGHT_MODEL_AMBIENT, newambient);

```
GLfloat amb[ ] = {0.2, 0.3, 0.1, 1.0};
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, amb);
```

Moving Light Sources in OpenGL

- To move a light source independently of the camera:
  1. Set its position array
  2. Clear the color and depth buffers
  3. Set up the modelview matrix to use for everything except the light source and push it
  4. Move the light source and set its position
  5. Pop the matrix
  6. Set up the camera, and draw the objects

Code: Independent Motion of Light

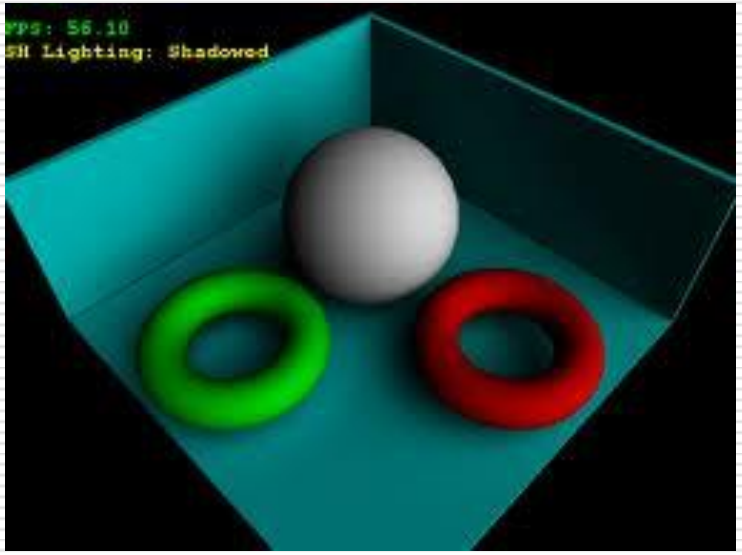
```
void display()
{  GLfloat position[ ] = {2, 1, 3, 1}; //initial light position
   // clear color and depth buffers
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
   glPushMatrix();
   glRotated(...);      glTranslated(...);    // move the light
   glLightfv(GL_LIGHT0, GL_POSITION, position);
   glPopMatrix();
   gluLookAt(...); // set the camera position
   <.. draw the object ..>
glutSwapBuffers(); }
```

Code: Light Moves with Camera

```
GLfloat pos[ ] = {0,0,0,1};
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glLightfv(GL_LIGHT0, GL_POSITION, position);
// light at (0,0,0)
gluLookAt(...); // move light and camera
// draw the object
```

OpenGL light model

- Light source
  - Point light
  - Direction light
- Three components
  - Diffuse light
  - Specular light
  - Ambient light (approximate the effect of global light roughly)



End.
Thanks