CS230 : Computer Graphics

Lecture 2

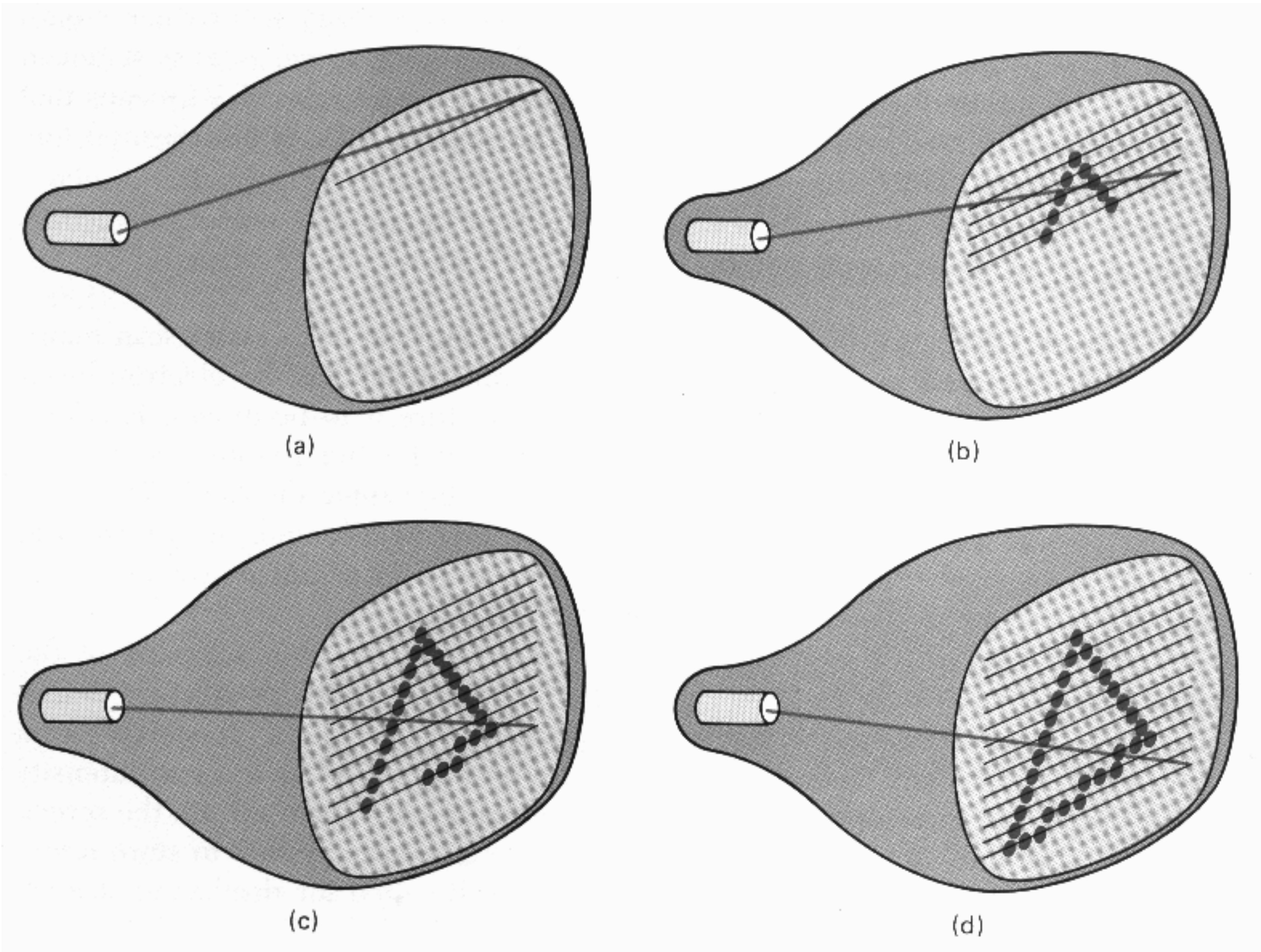
Tamar Shinar
Computer Science & Engineering
UC Riverside

Raster Devices and Images

Raster Devices

- raster displays show images as a rectangular array of pixels
- most printers are also raster devices
  - image is made by depositing ink at points on a grid
- digital cameras - have image sensors made of grid of light-sensitive pixels (2D array)
- scanner - linear array of pixels swept across page to create grid of pixels (1D array)

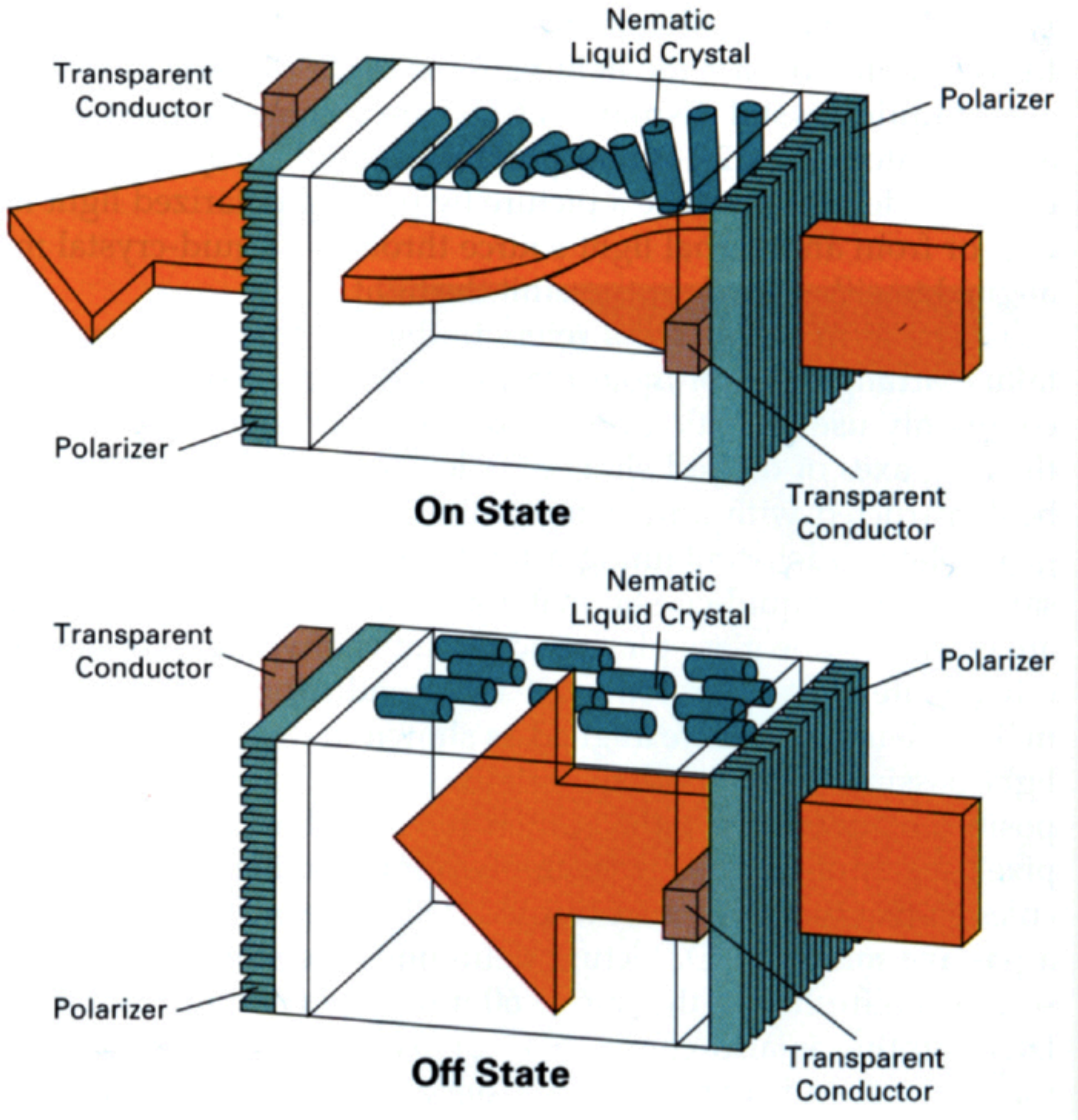
Raster Display



Hearn, Baker, Carithers

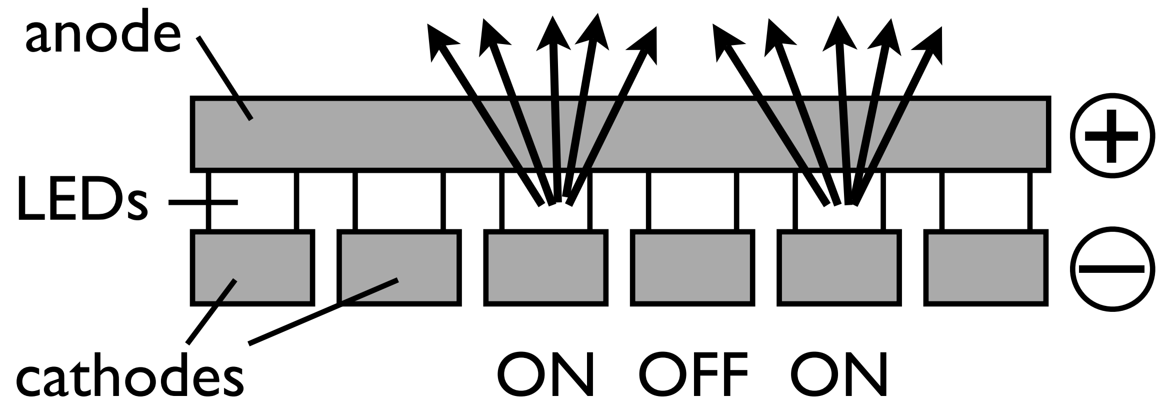
virtually all graphics system are **raster based,** meaning the image we see is a **raster of pixels**
or a rectangular array of pixels
Here a raster scan device display an image as a set of discrete points across each scanline

Transmissive vs. Emissive Display



[H&B, Fig. 2-16]

LCD

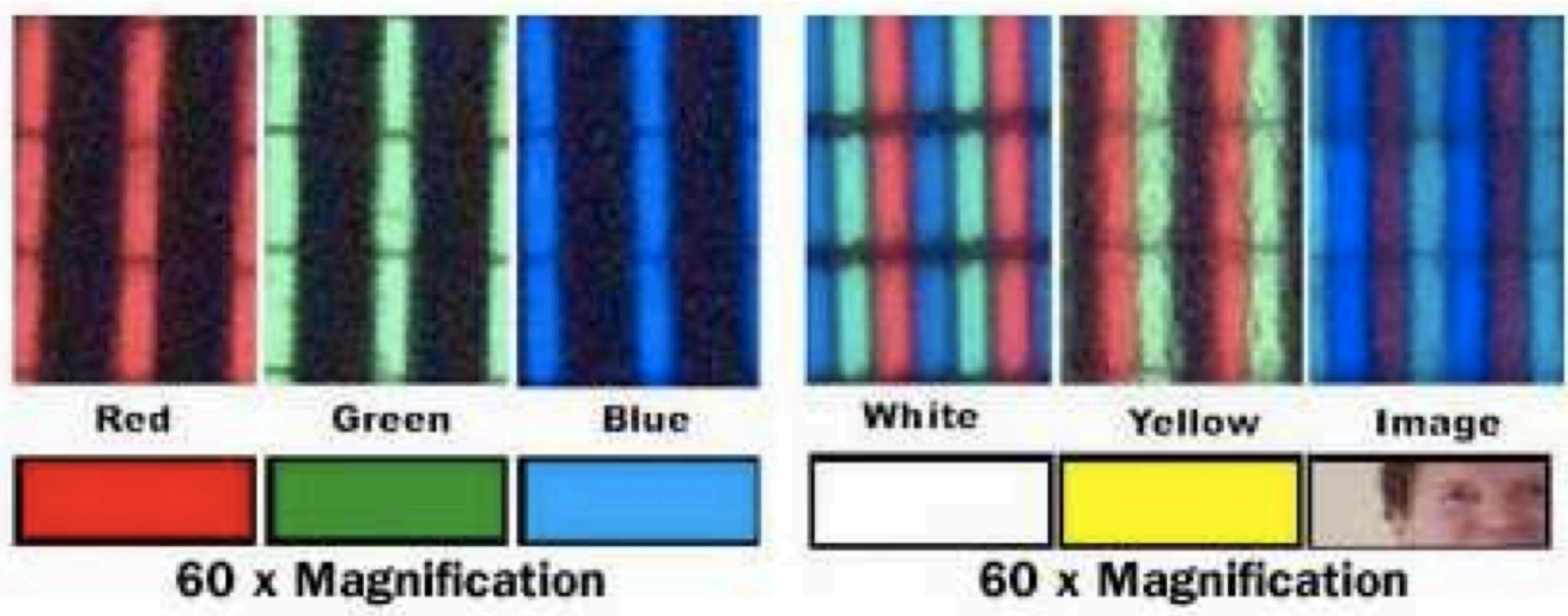


LED

Displays are either **transmissive** or **emissive**

one pixel of an LCD display:
(LEFT)In the **off state** the front polarizer blocks all the light that passes the back polarizer
in the **on state** the liquid crystal rotates the polarization of the light so it can pass through the front polarizer
the degree of rotation can be adjusted by an applied voltage
(RIGHT) LED display

Raster Display



red, green, blue subpixels

get different colors by mixing red, green, and blue
this is from an LCD monitor
printers are also raster-based. image is made out of points on a grid

What is an image?

Continuous image

$$I : R \rightarrow V$$

$$R \subset \mathbb{R}^2$$

$$V = \mathbb{R}^+ \quad \text{(grayscale)}$$

$$V = (\mathbb{R}^+)^3 \quad \text{(color)}$$

An (continuous) image is a function defined over some 2D area, that maps points to intensity level

What is an image?

Sampled image

$$I : R \to V$$

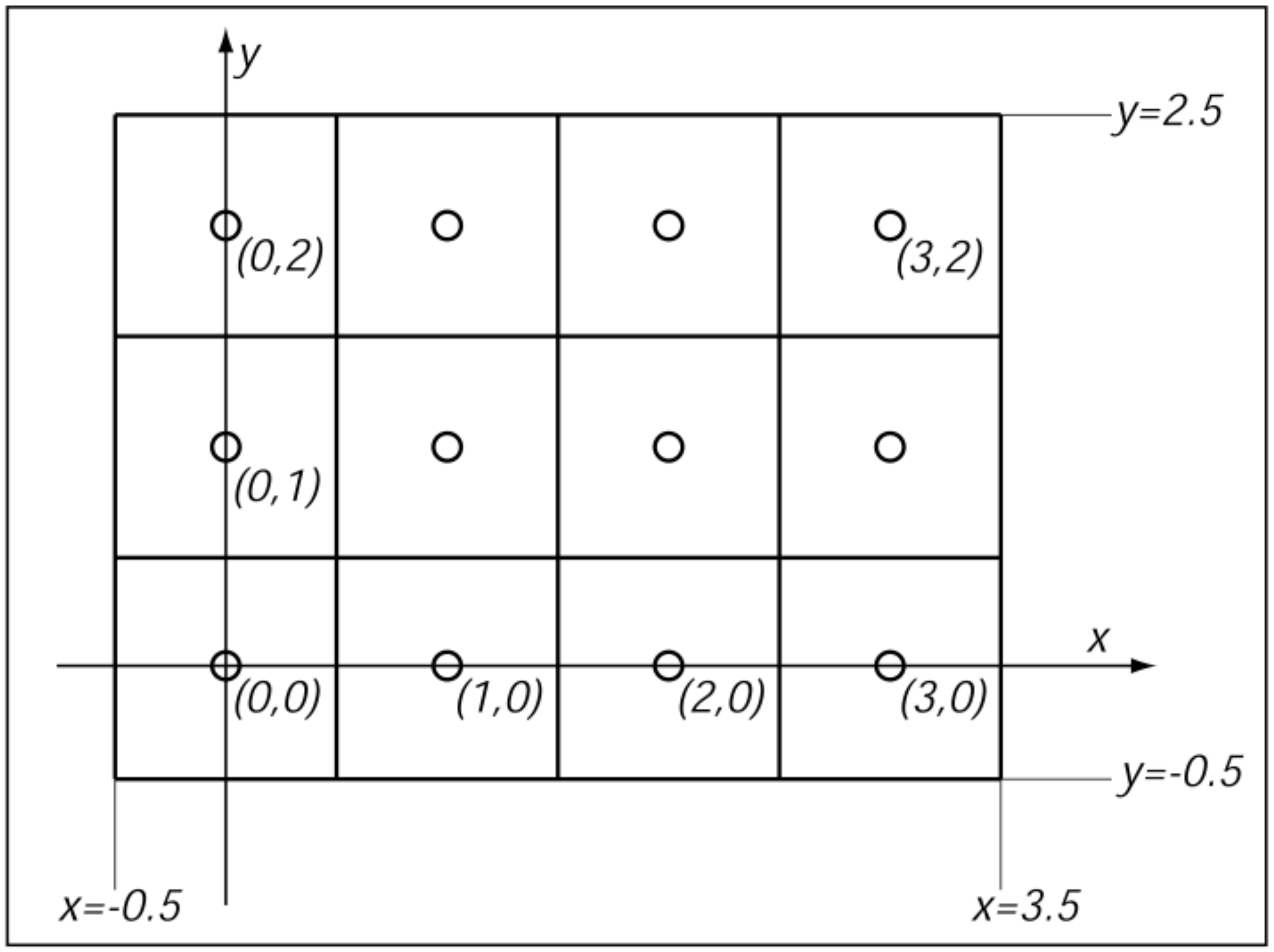
$$R \subset \mathbb{Z}^2$$

$$V = [0, 1]$$ (grayscale)

$$V = [0, 1]^3$$ (color)

n_x = number of columns

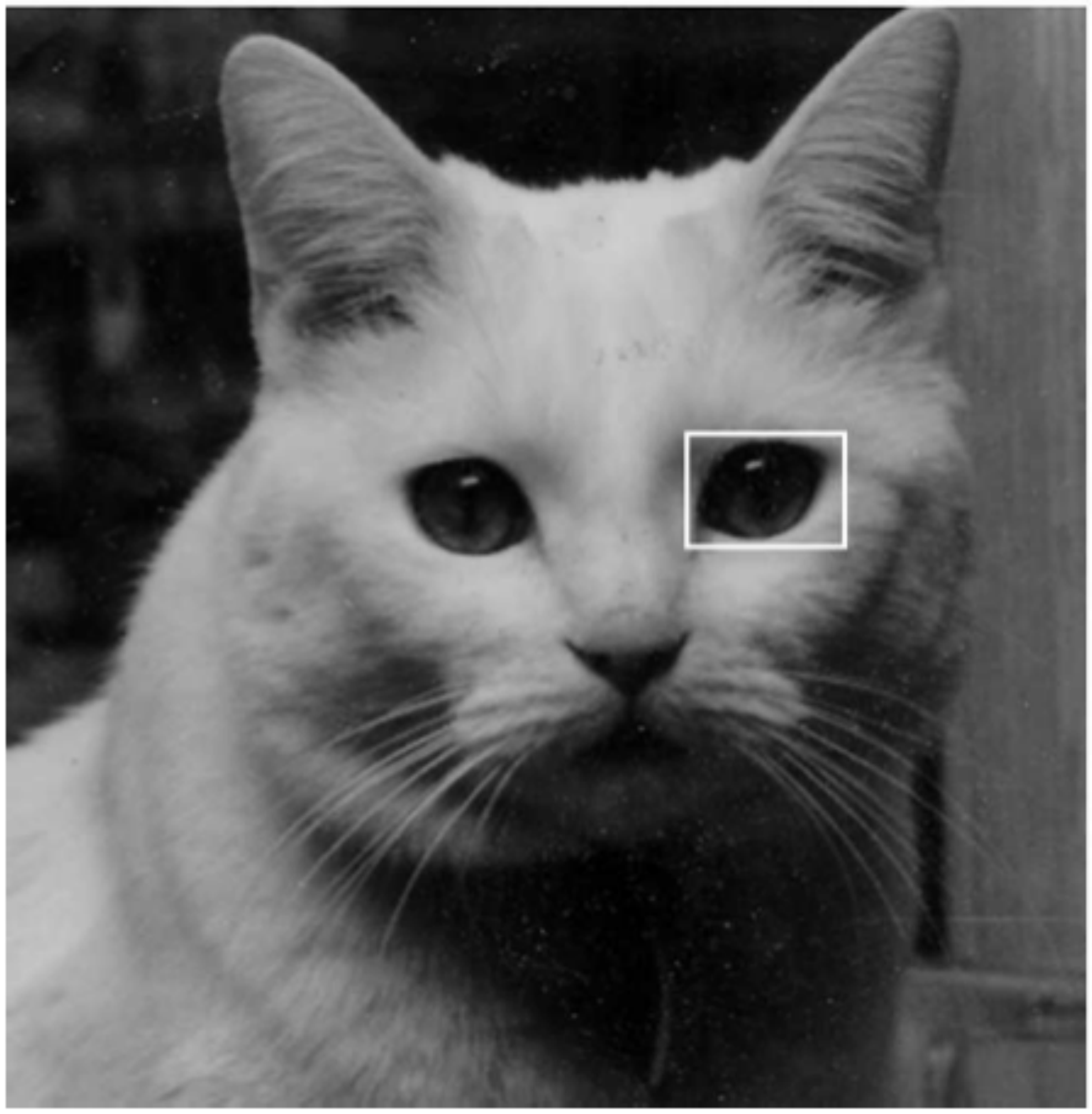
n_y = number of rows

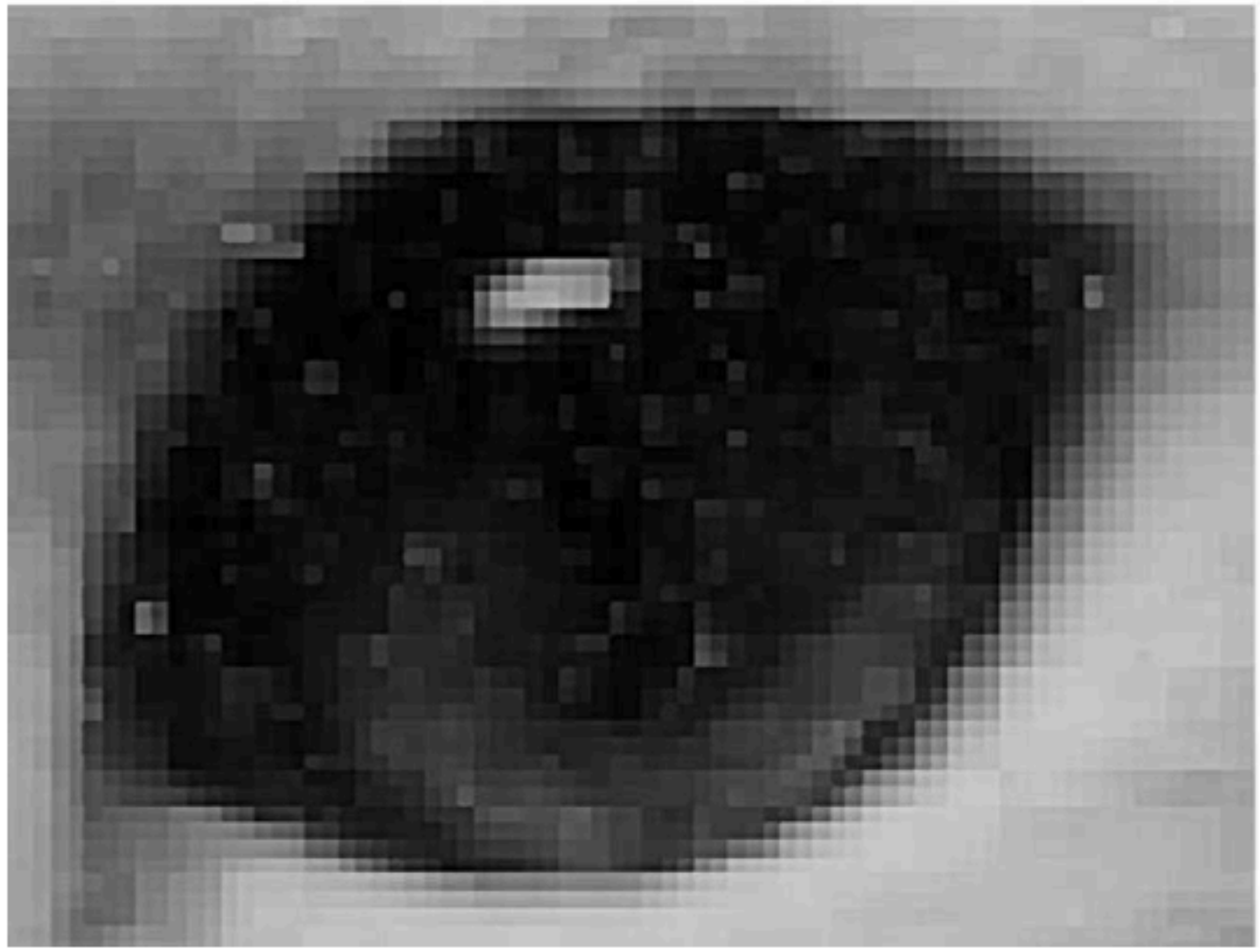


$$[-0.5, n_x - 0.5] \times [-0.5, n_y - 0.5]$$

each pixel value represents the **average color** of the image over that pixel's area.

Raster Image





A **raster image** is 2D array storing pixel values at each pixel (picture element)
3 numbers for color
alternative: **vector image** -- essentially a set of instructions for rendering an image

Bit depth - defined by device standards

Bit-Depth	Number of Colors
1	2 (monochrome)
2	4 (CGA)
4	16 (EGA)
8	256 (VGA)
16	65,536 (High Color, XGA)
24	16,777,216 (True Color, SVGA)
32	16,777,216 (True Color + Alpha Channel)

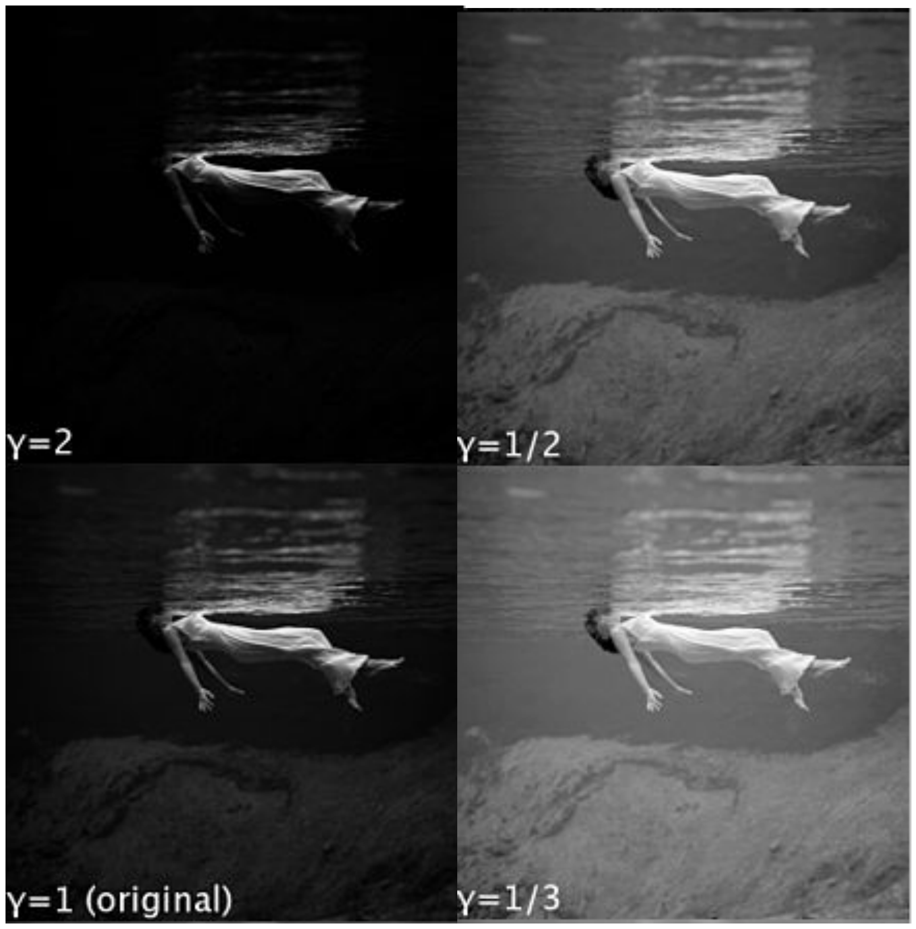
(Note alpha)

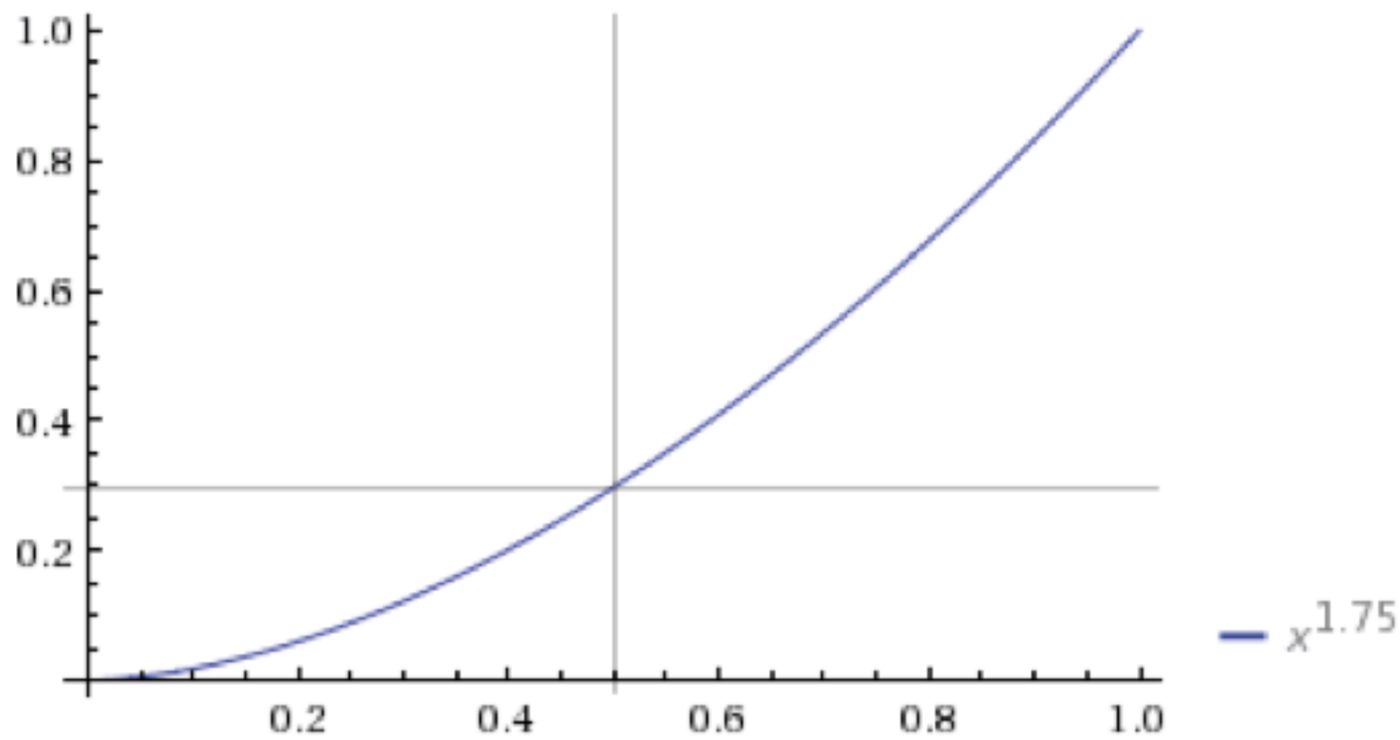
(Humans can perceive ~10,000,000 colors)

in practice, it is sufficient for pixels to have a bounded range e.g., [0,1]
They are represented in integers

Monitor Gamma

displayed intensity = (max intensity) a^γ



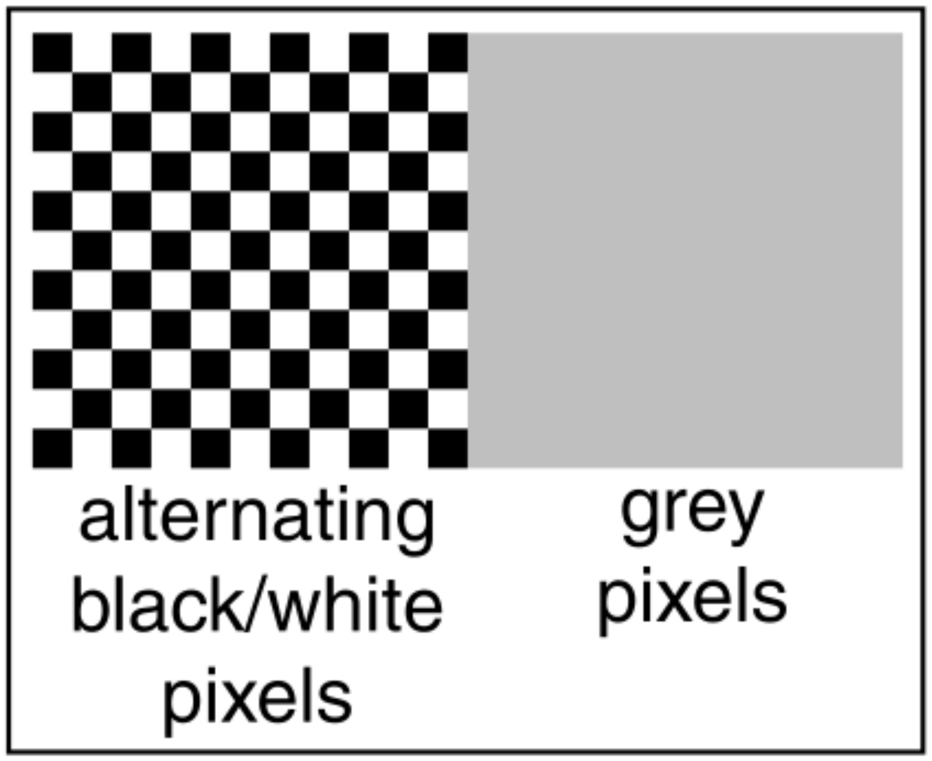


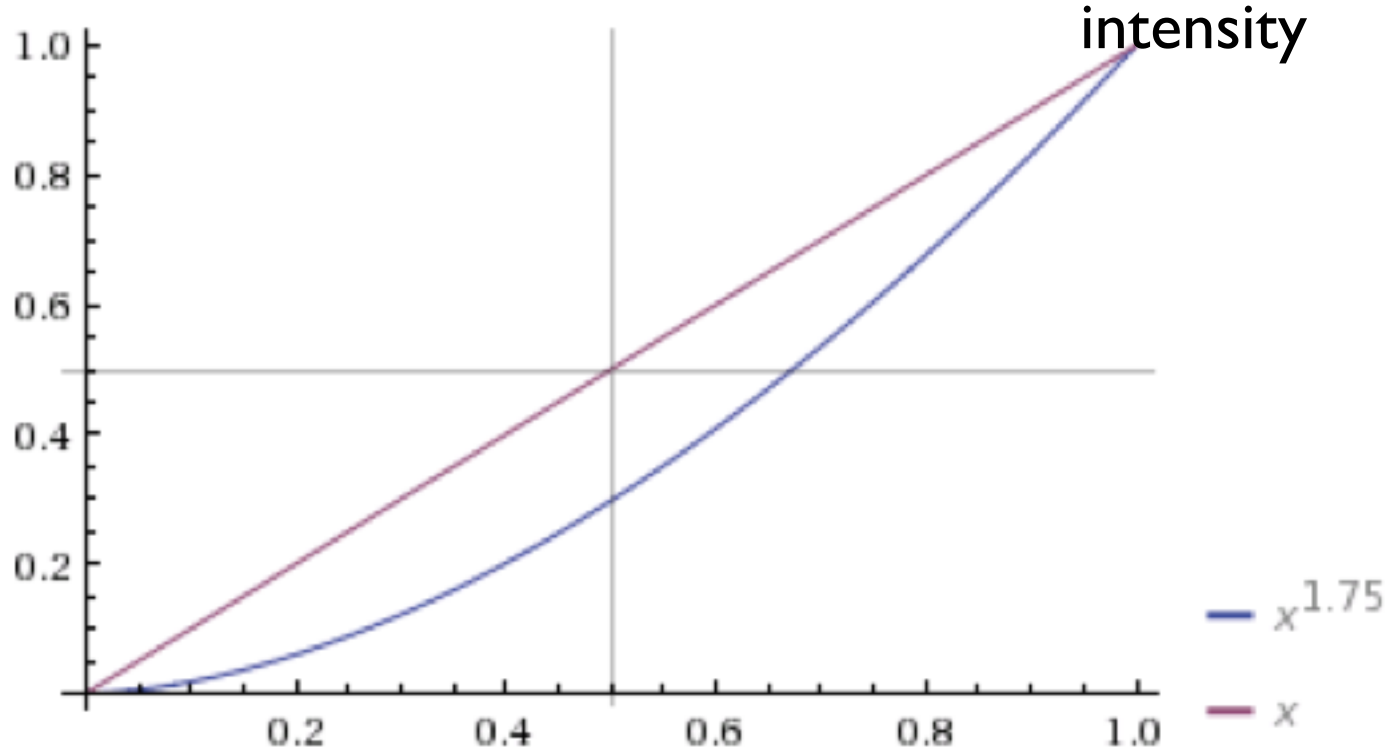
monitors convert pixel values, a, into displayed intensities
monitors are nonlinear with respect to input

Gamma Correction

displayed intensity = (max intensity)$(\underbrace{a^{\frac{1}{\gamma}}})^{\gamma}$

gamma-corrected intensity





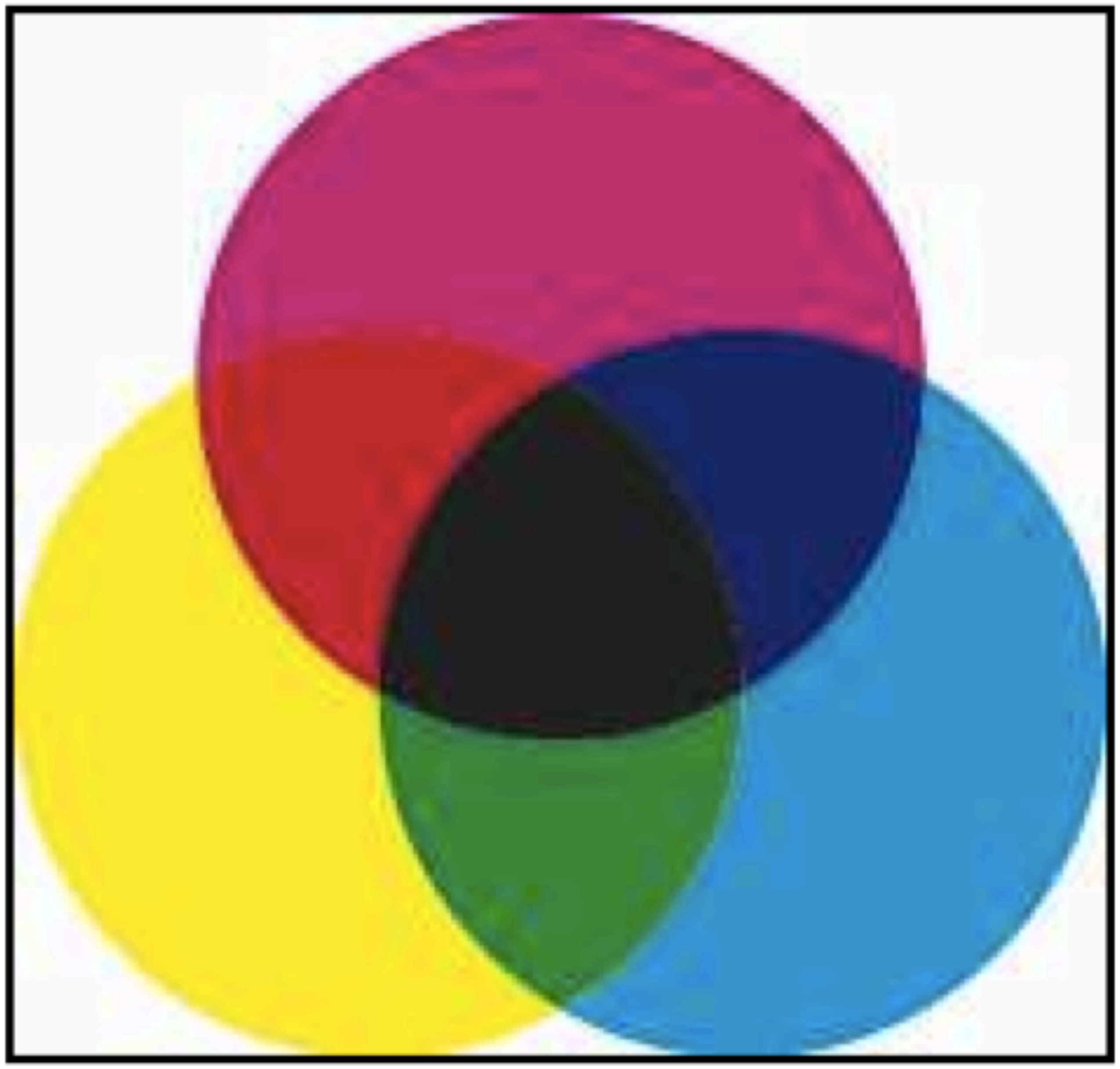
find gamma using, e.g., checkboard

then gamma-correct the input

find gamma, so that you can give the monitor a^{1/\gamma}
- find a such that a^{\gamma} = .5 through checkboard test and solve for gamma

Color representation

additive

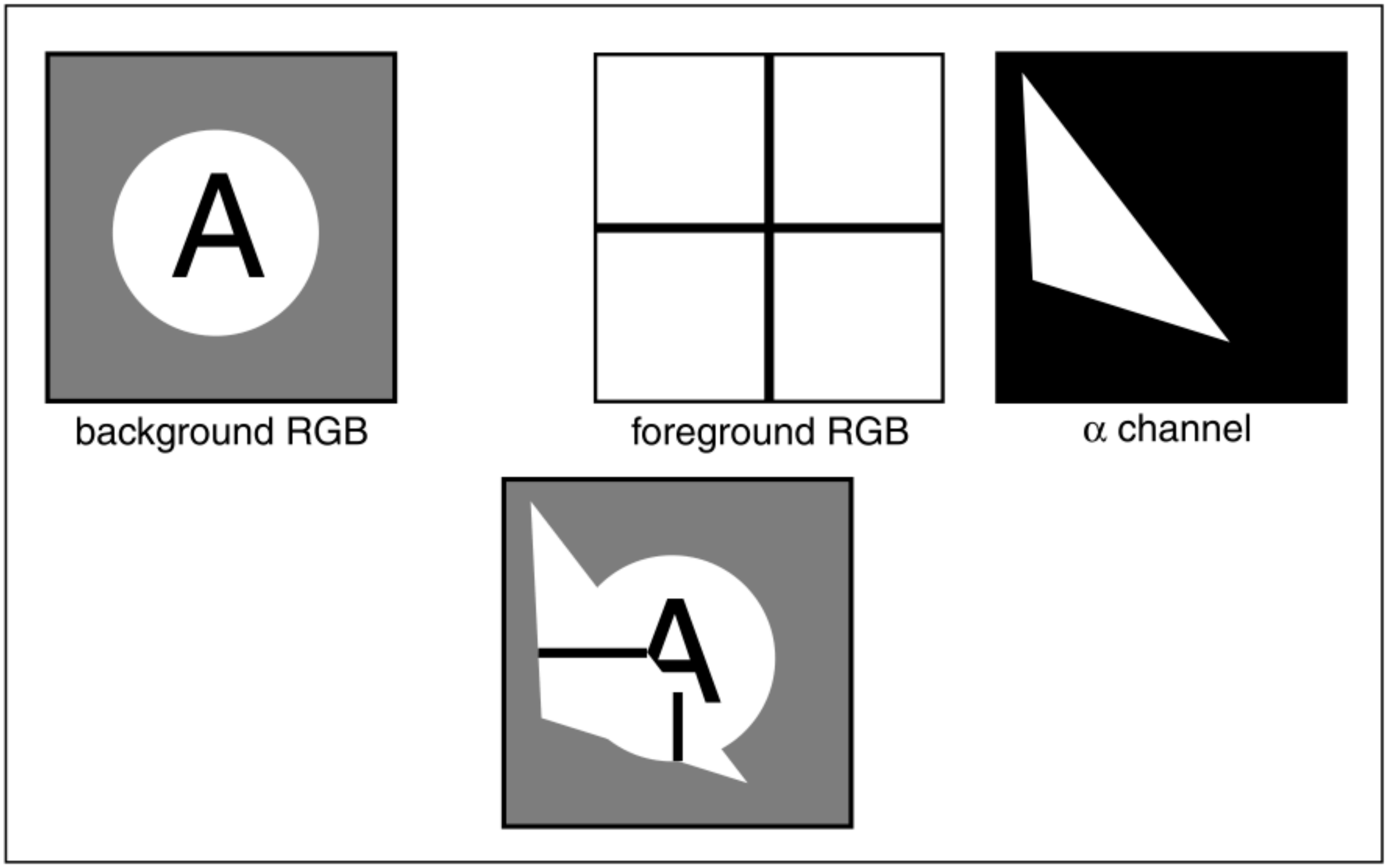


subtractive

additive color – Primary colors are red, green, blue. form a color by adding these. CRTs, projectors, LCD displays, positive film
subtractive color – form a color by filtering white light with cyan, magenta, and yellow filters printing, negative film

Alpha Channel

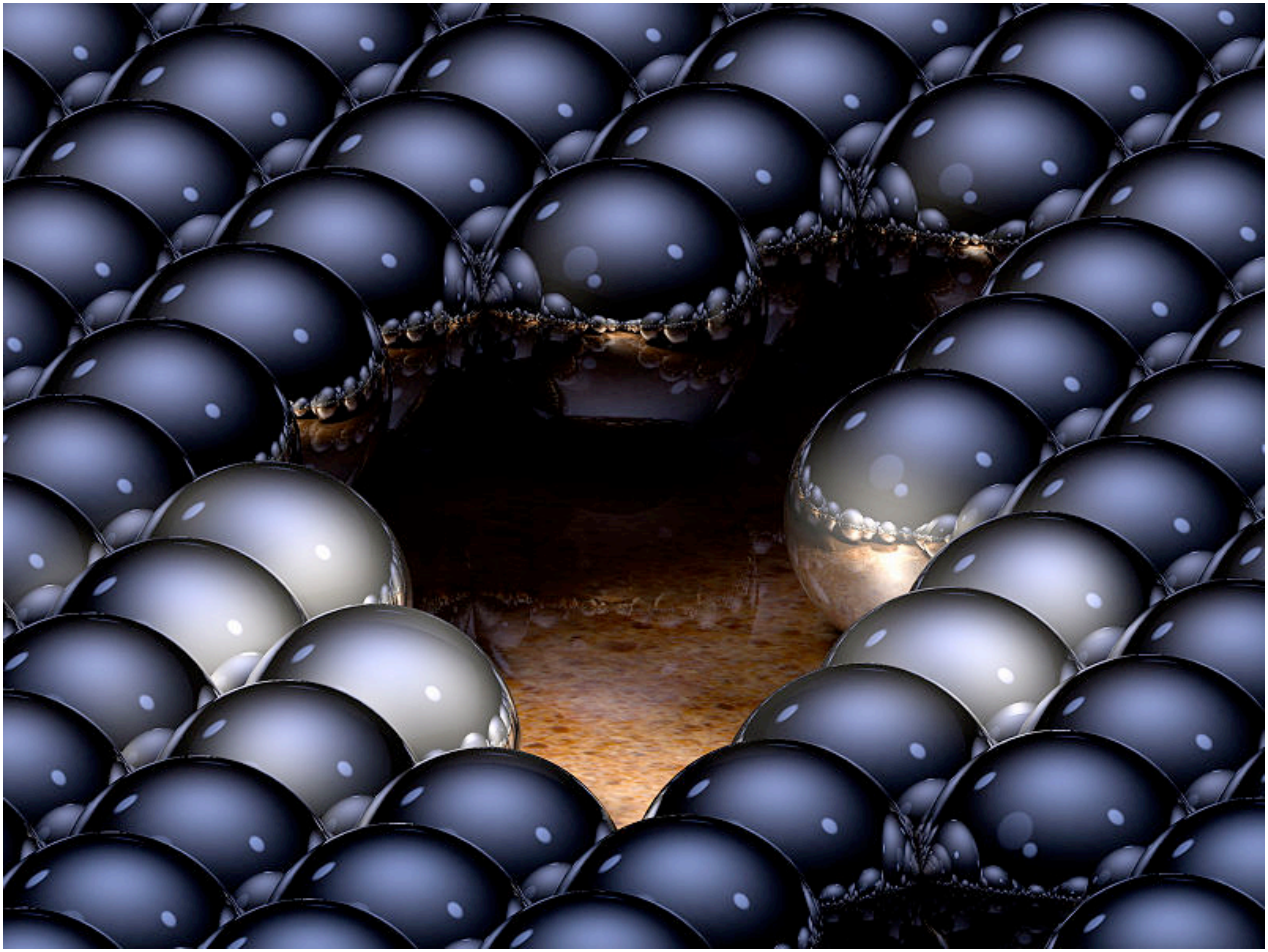
$$\mathbf{c} = \alpha \mathbf{c}_f + (1 - \alpha)\mathbf{c}_b$$



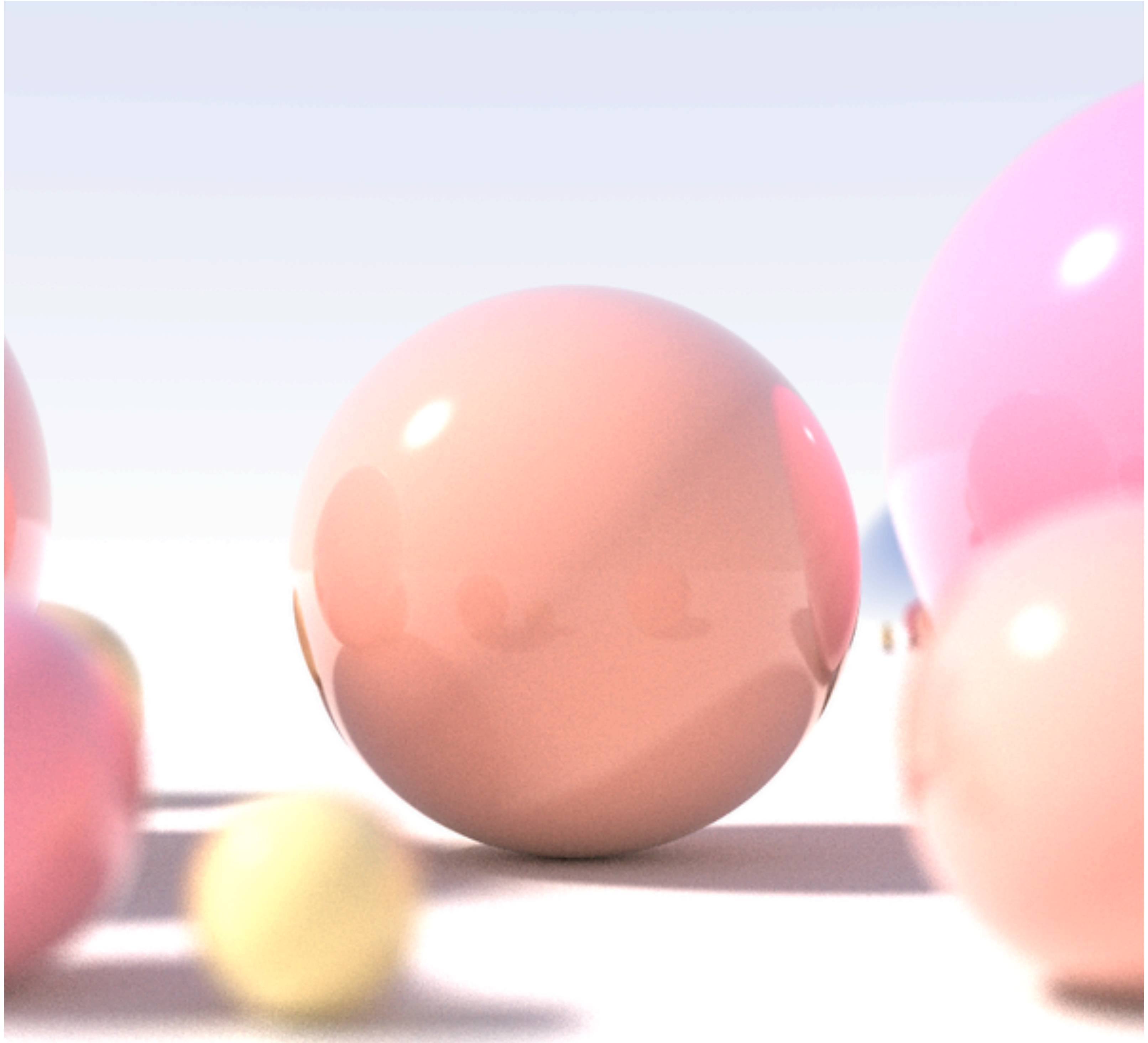
Compositing: two different interpretations: **pixel coverage** (fraction of pixel covered) and **blending**

Ray Tracing


Wikimedia Commons


Greg L., Wikimedia Commons

up to 16 reflections per ray



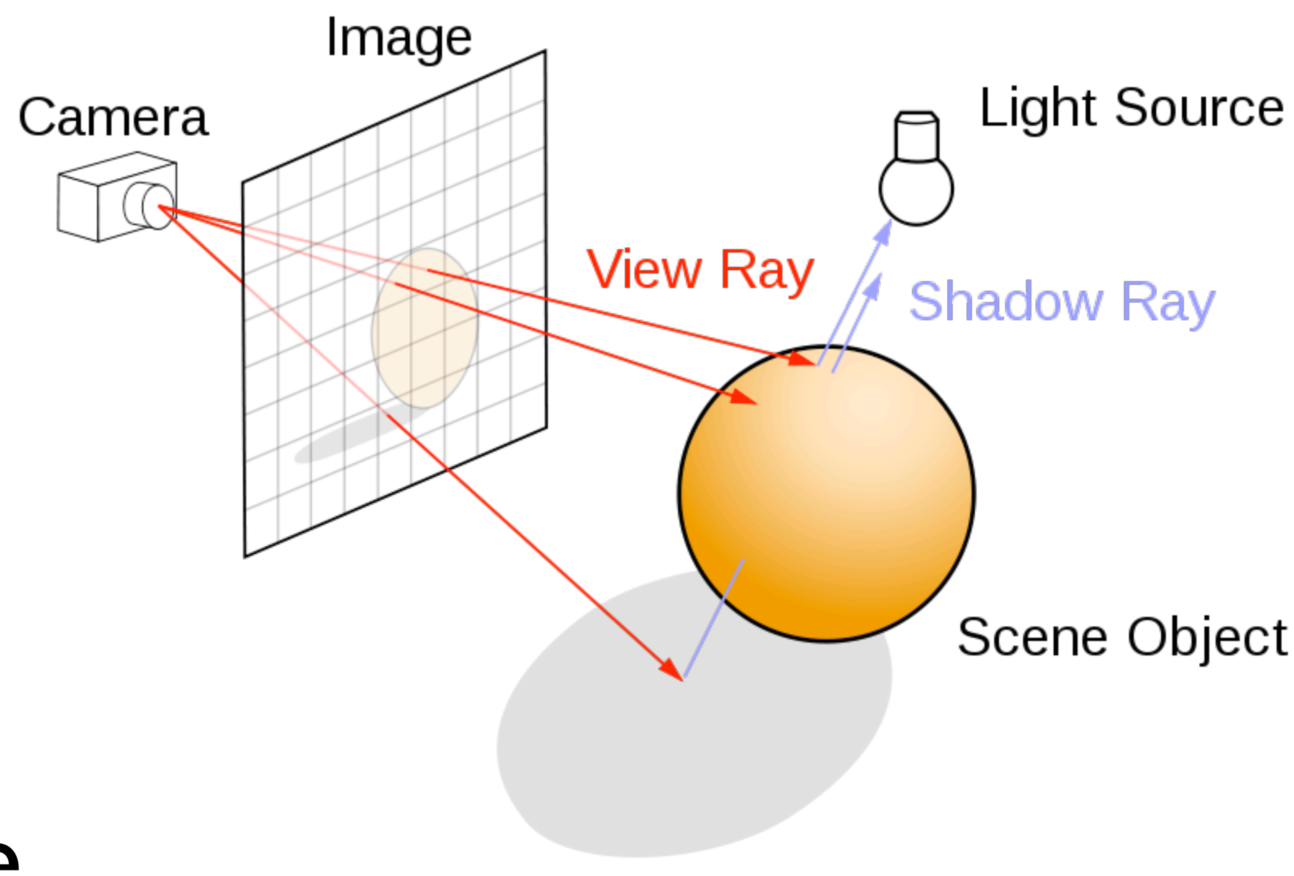
Wikimedia Commons

shallow depth of field, area light sources, diffuse interreflection

Basic Algorithm

for each pixel

1. **cast view ray**: compute view ray from camera through pixel into scene
2. **intersect**: find intersection of ray with closest object
3. **shade**: compute the color of the intersection point

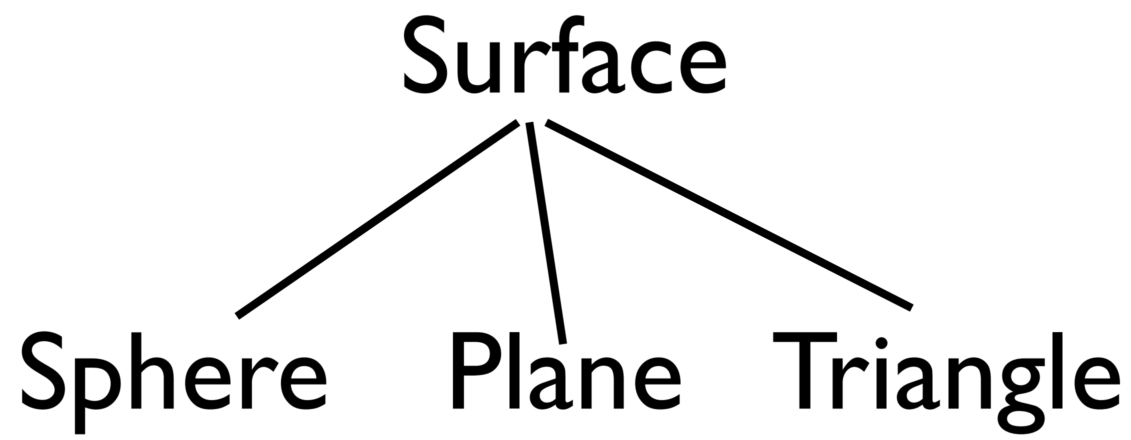


Ray Tracing Program

```
for each pixel do
    compute viewing ray
    if ( ray hits an object with t in [0, inf] ) then
        compute n
        evaluate shading model and set pixel to that color
    else
        set pixel color to the background color
```

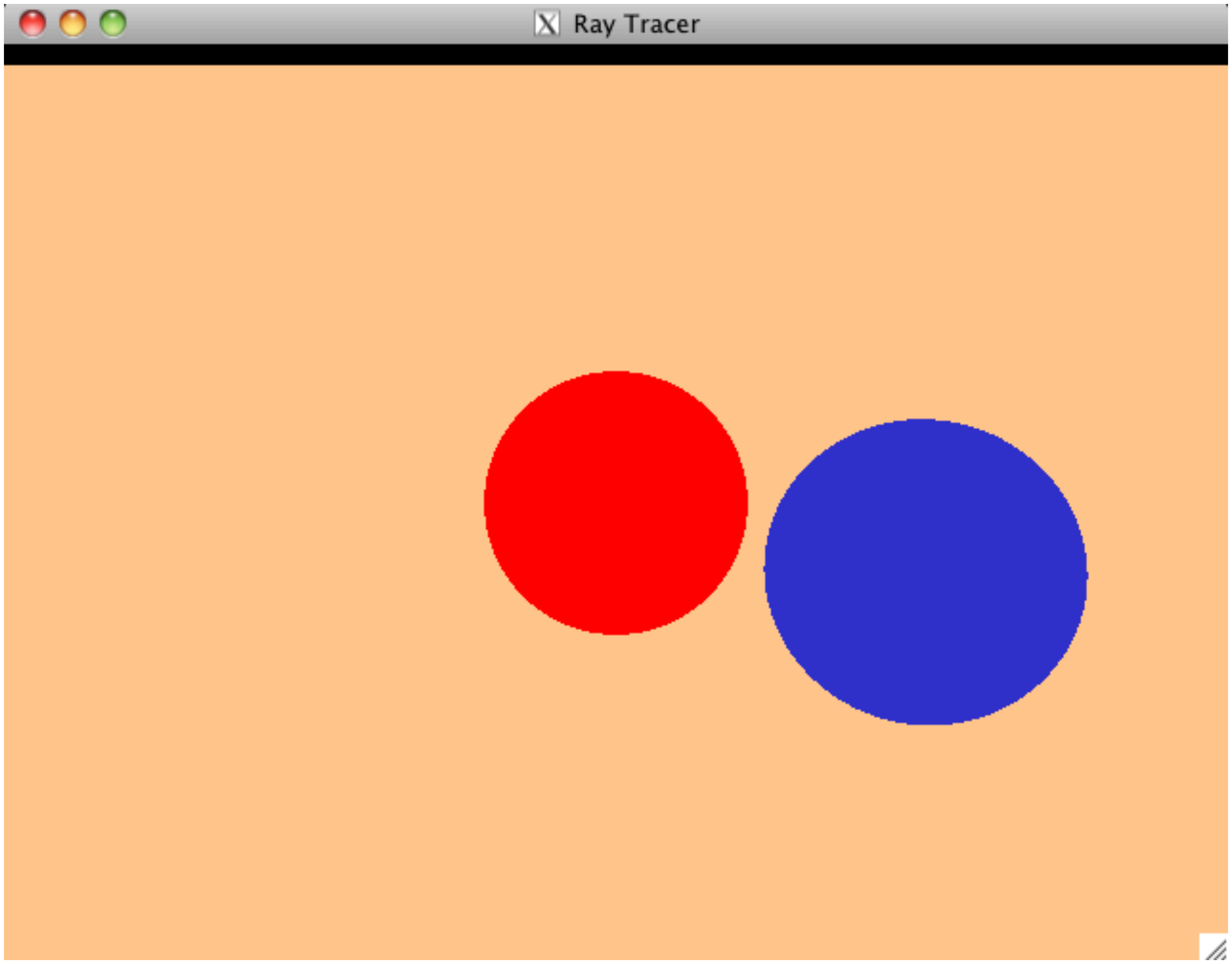
Object-oriented design

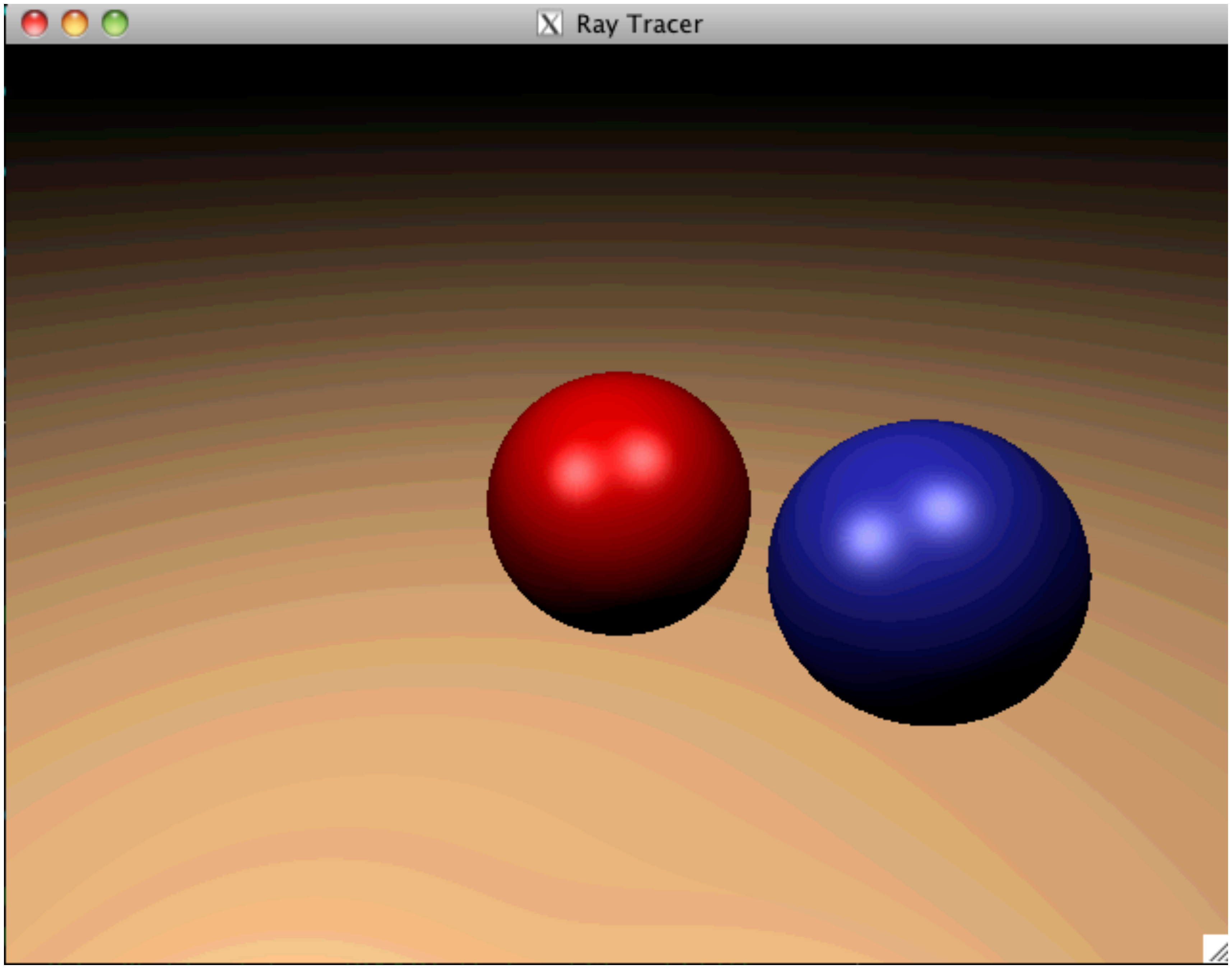
```
class Surface
{
    public:
        bool Intersection(RAY& ray)=0;
        Box Bounding_Box()=0;
}
```

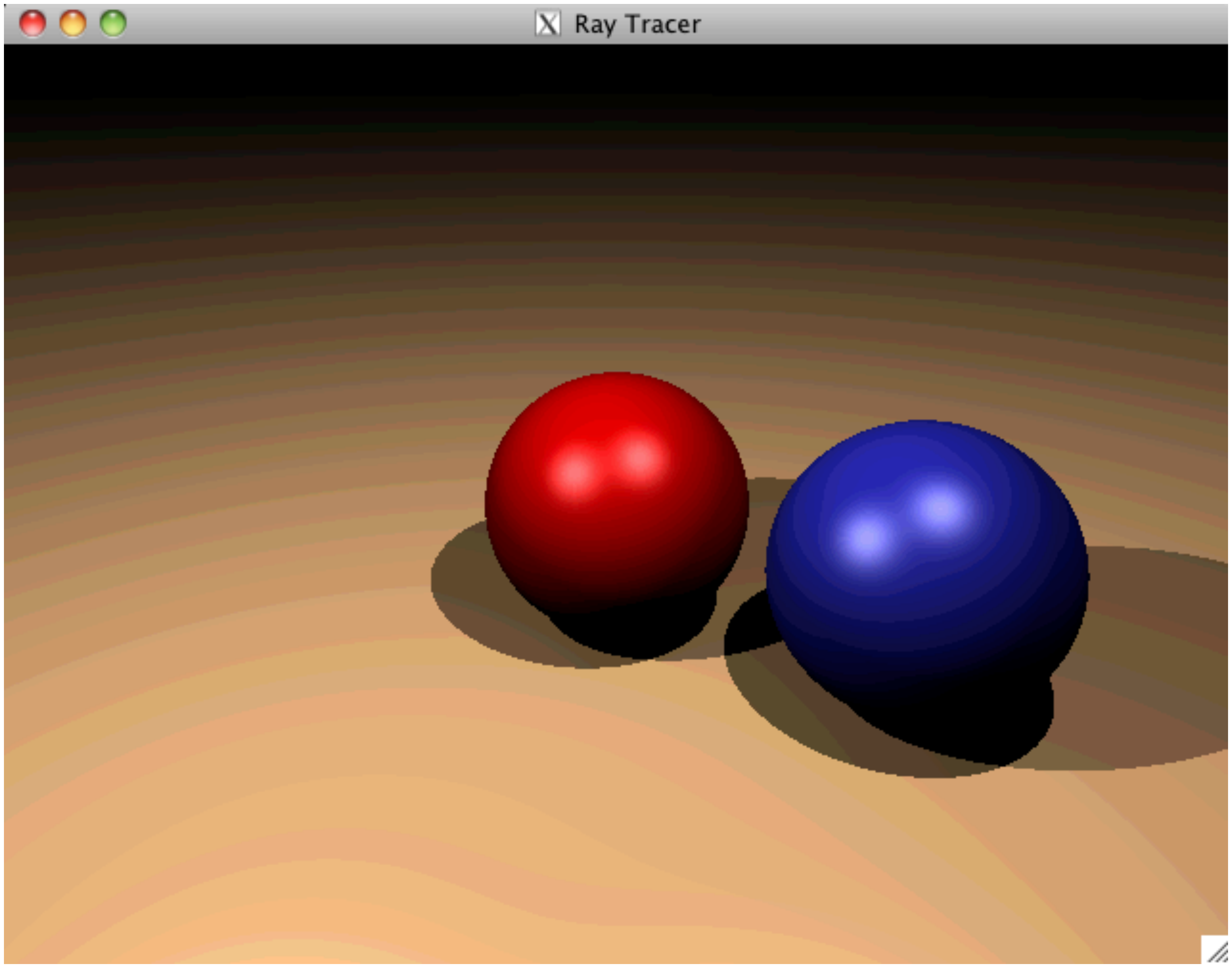


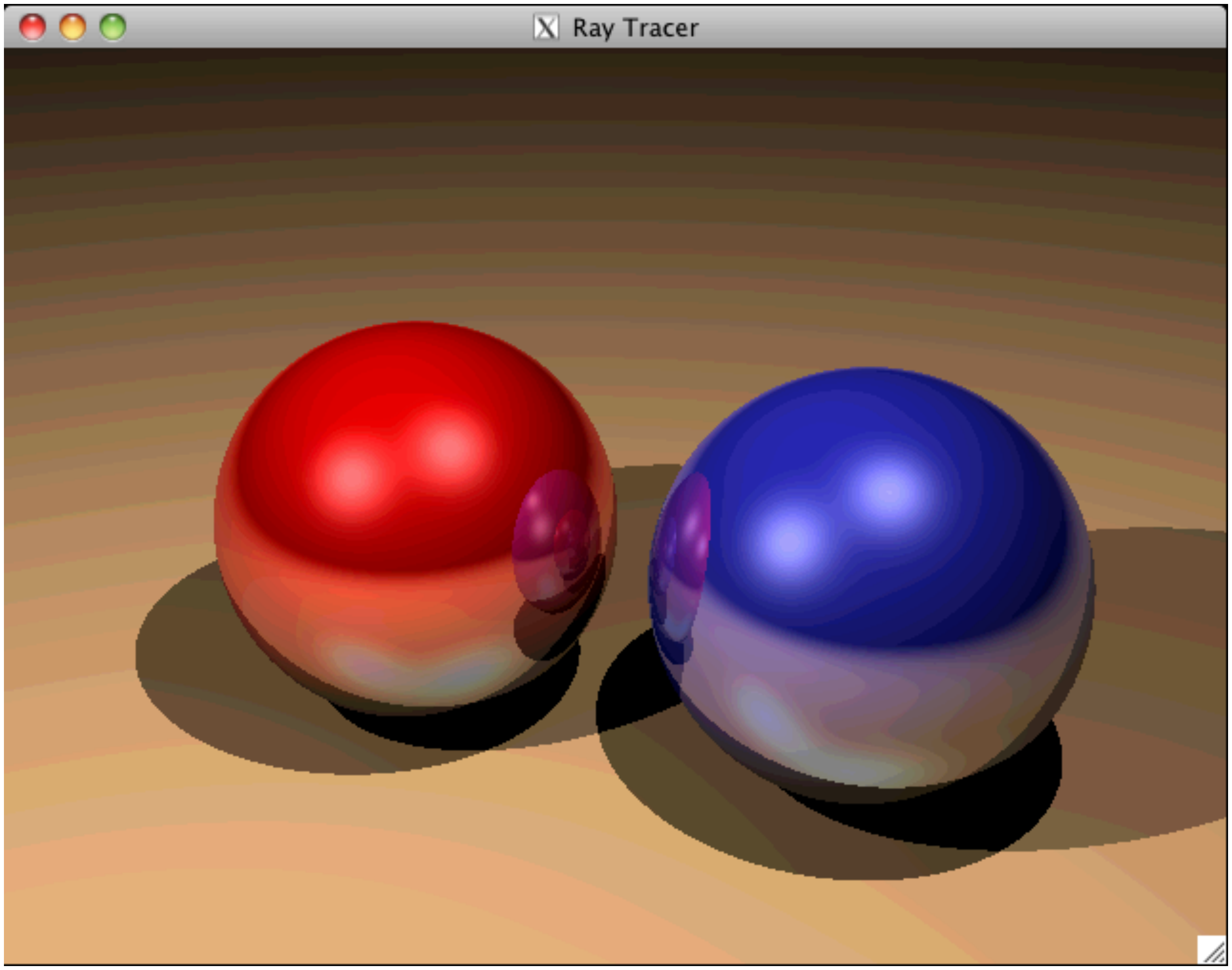
Other objects: Ray, Light, Material, Camera, Film, World

Simple Ray Tracer




Ray Tracer


Ray Tracer


Ray Tracer