CSE 167:
Introduction to Computer Graphics
Lecture #5: Rasterization

Jürgen P. Schulze, Ph.D.
University of California, San Diego
Fall Quarter 2013

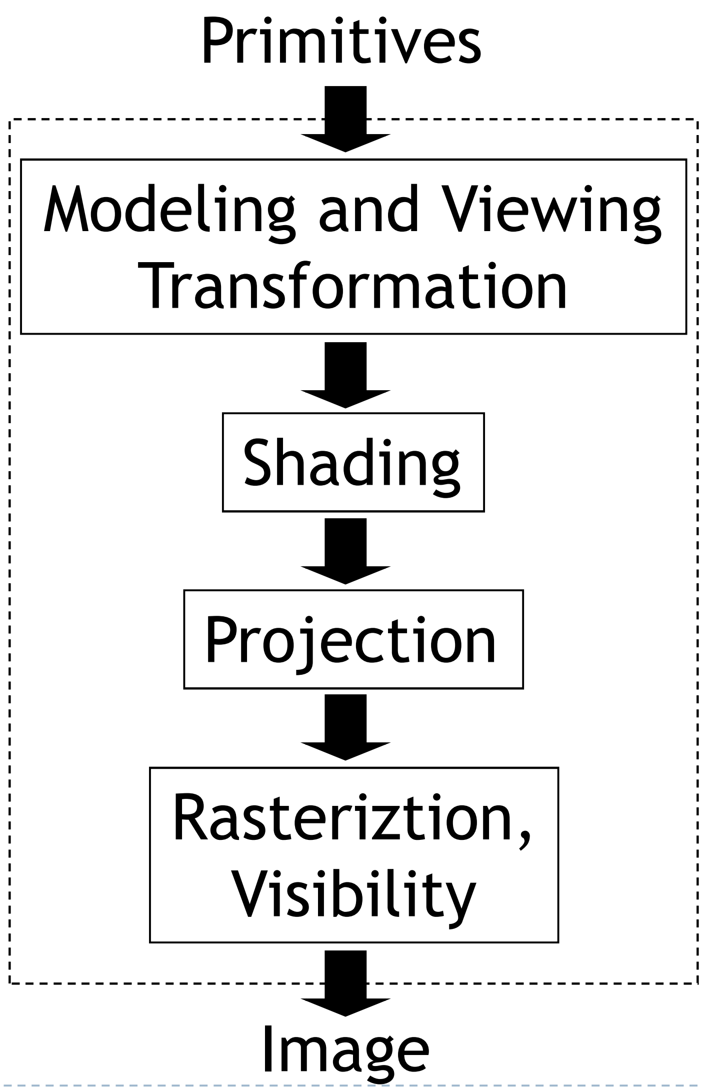
Announcements

- Homework project #2 due tomorrow, October 11
  - To be presented starting 1:30pm in labs 260/270
  - Late submissions for project #1 accepted

Lecture Overview

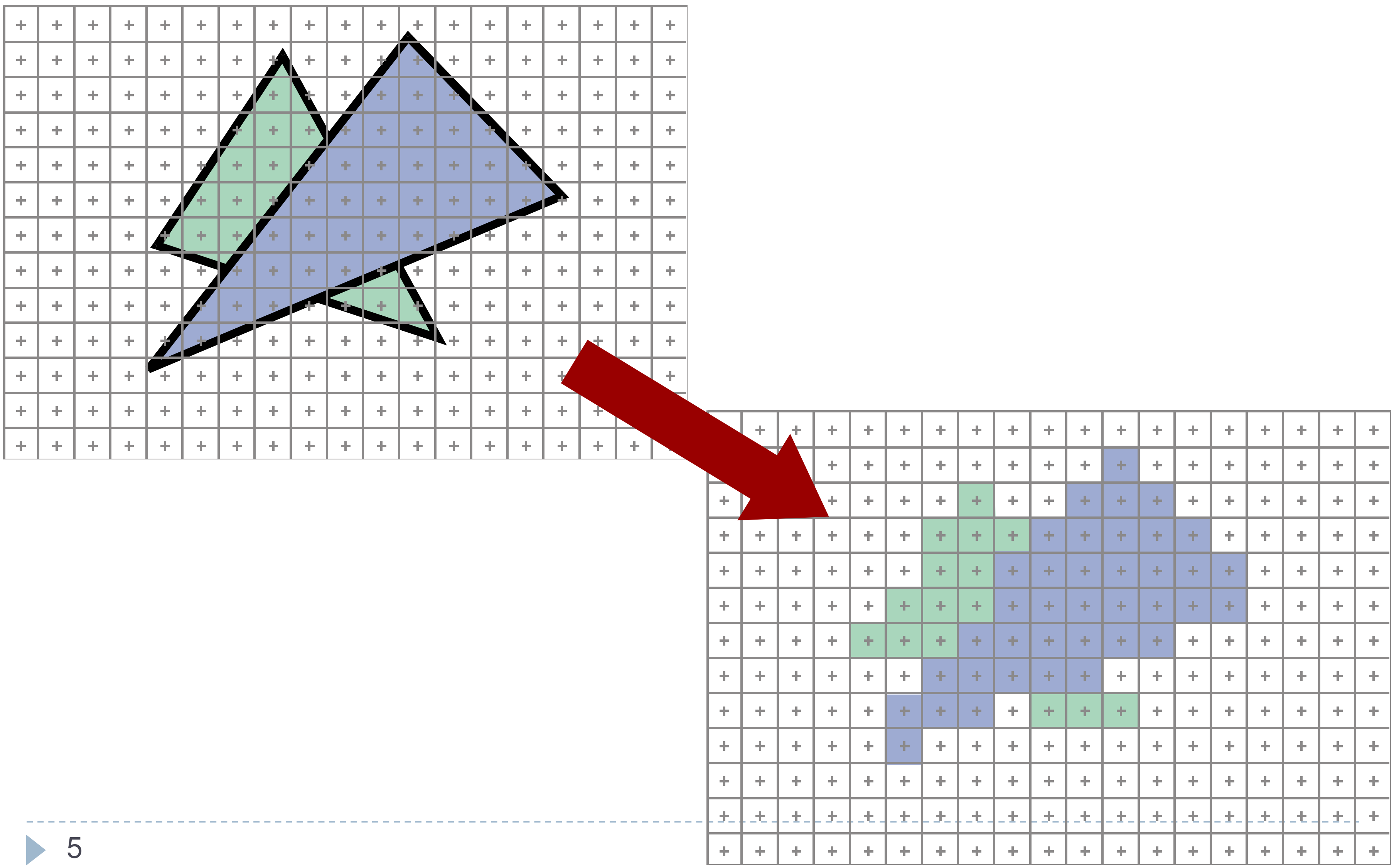
- Rasterization
- Visibility
- Barycentric Coordinates
- Color: Physical Background
- Color Perception

Rendering Pipeline



- Scan conversion and rasterization are synonyms
- One of the main operations performed by GPU
- Draw triangles, lines, points (squares)
- Focus on triangles in this lecture

Rasterization



Rasterization

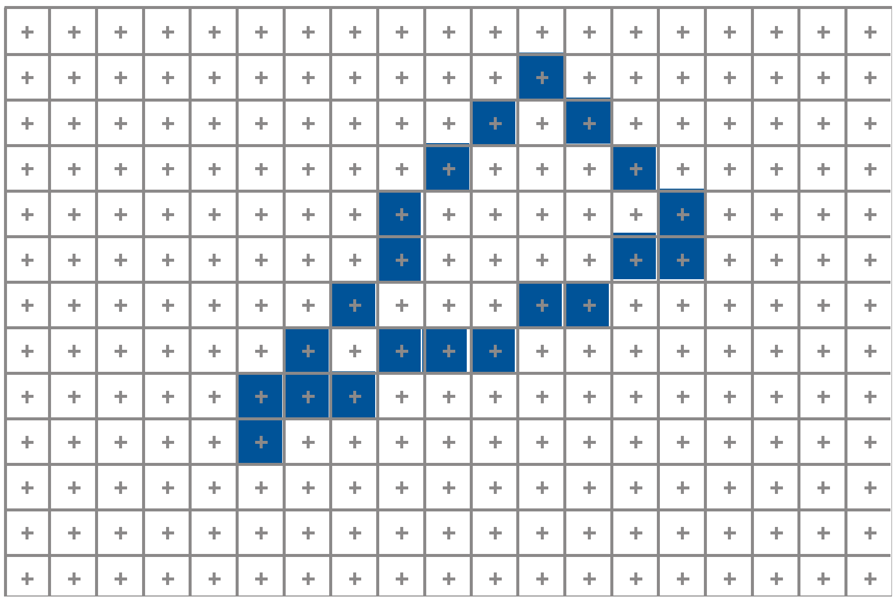
- How many pixels can a modern graphics processor draw per second?

Rasterization

- How many pixels can a modern graphics processor draw per second?
- NVidia GeForce GTX 780
  - 160 billion pixels per second
  - Multiple of what the fastest CPU could do

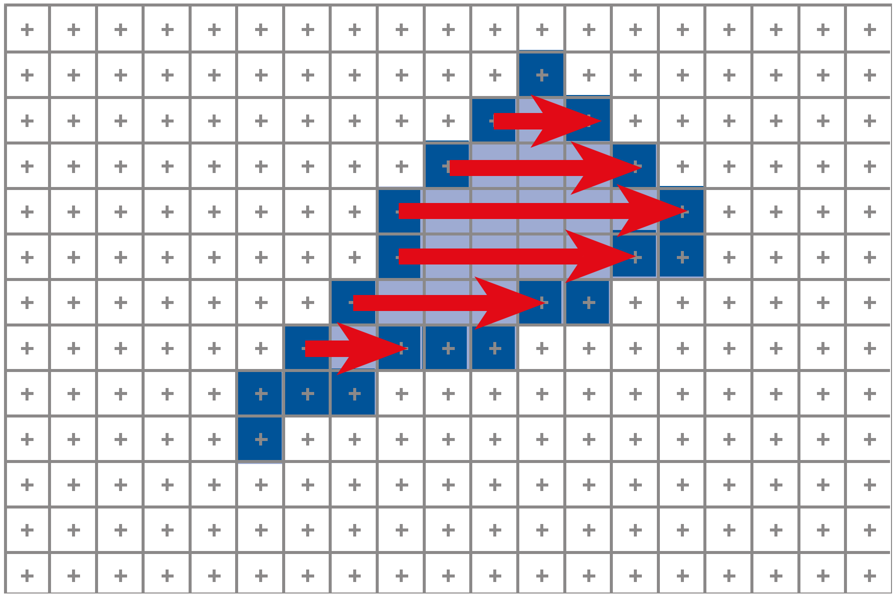
Rasterization

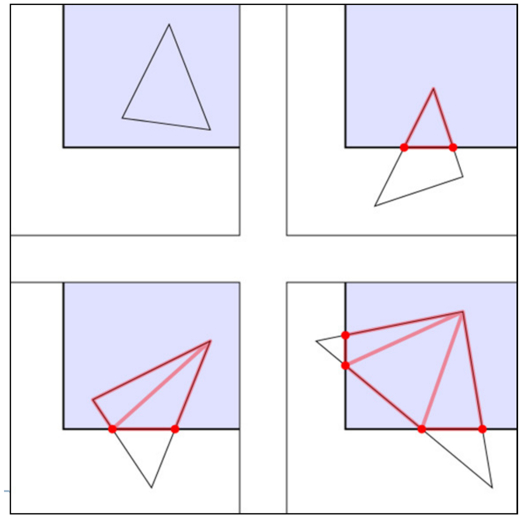
- Many different algorithms
- Old style
  - Rasterize edges first



Rasterization

- Many different algorithms
- Example:
  - Rasterize edges first
  - Fill the spans (scan lines)
- Disadvantage:
  - Requires clipping





Source: *http://www.arcsynthesis.org*

Rasterization

- GPU rasteriazation today based on "Homogeneous Rasterization"

  http://www.ece.unm.edu/course/ece595/docs/olano.pdf

  Olano, Marc and Trey Greer, "Triangle Scan Conversion Using 2D Homogeneous Coordinates", Proceedings of the 1997 SIGGRAPH/Eurographics Workshop on Graphics Hardware (Los Angeles, CA, August 2-4, 1997), ACM SIGGRAPH, New York, 1995.

Rasterization

- Given vertices in pixel coordinates

$$\mathbf{p}' = \mathbf{D}\mathbf{P}\mathbf{C}^{-1}\mathbf{M}\mathbf{p}$$

World space

Camera space

Clip space

Image space

$$\mathbf{p}' = \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix}$$

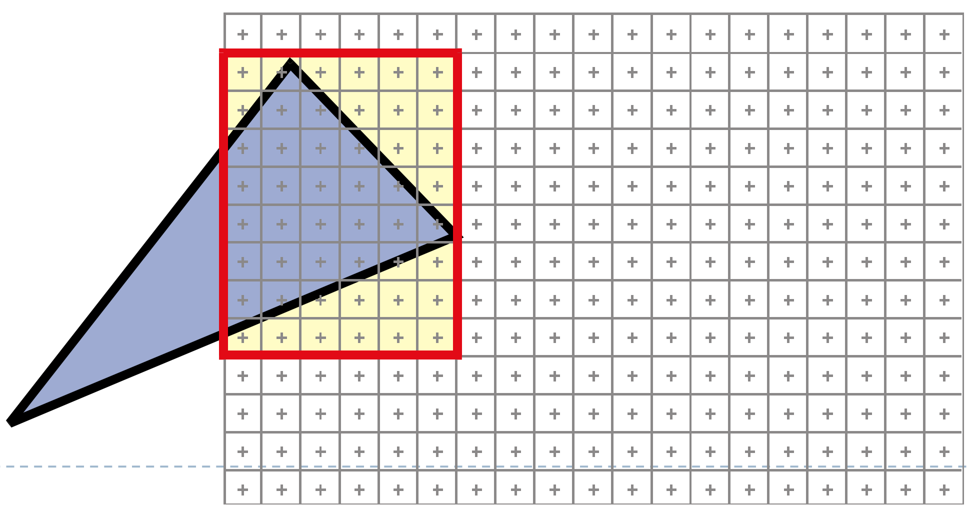
Pixel coordinates $\begin{matrix} x'/w' \\ y'/w' \end{matrix}$

Rasterization

- Simple algorithm

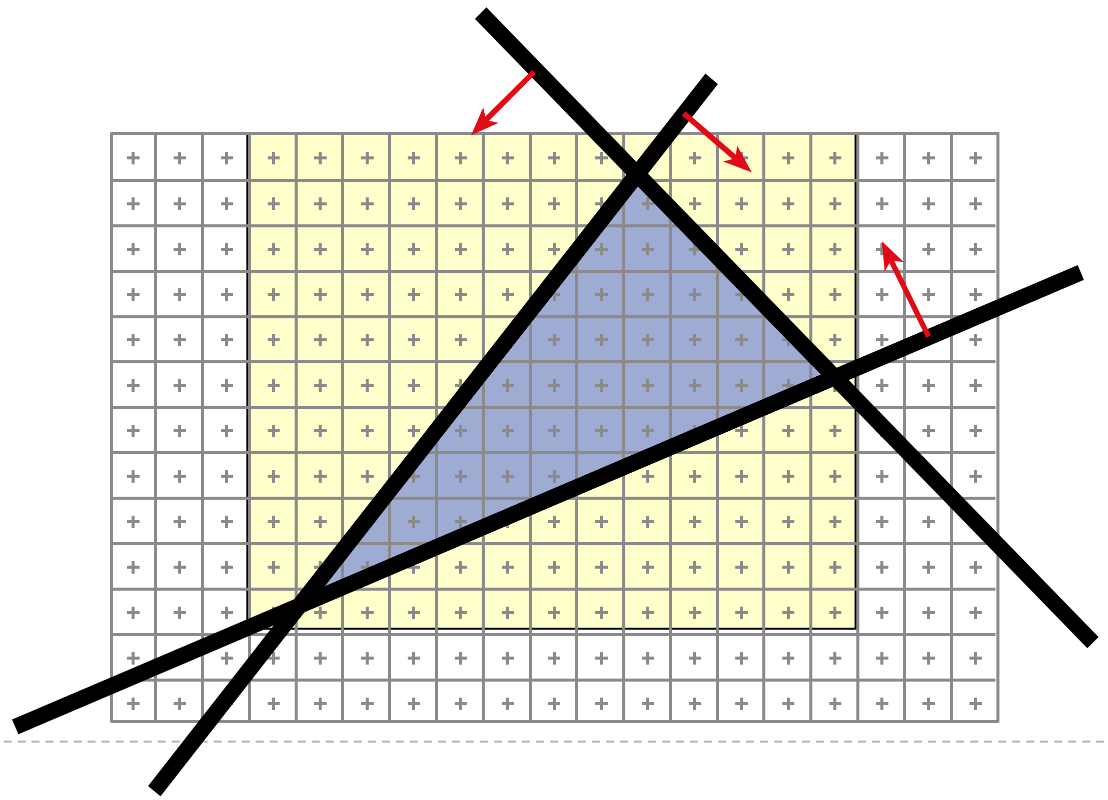
```
compute bbox
clip bbox to screen limits
for all pixels [x,y] in bbox
   compute barycentric coordinates alpha, beta, gamma
   if 0<alpha,beta,gamma<1 //pixel in triangle
          image[x,y]=triangleColor
```

- Bounding box clipping trivial



Rasterization

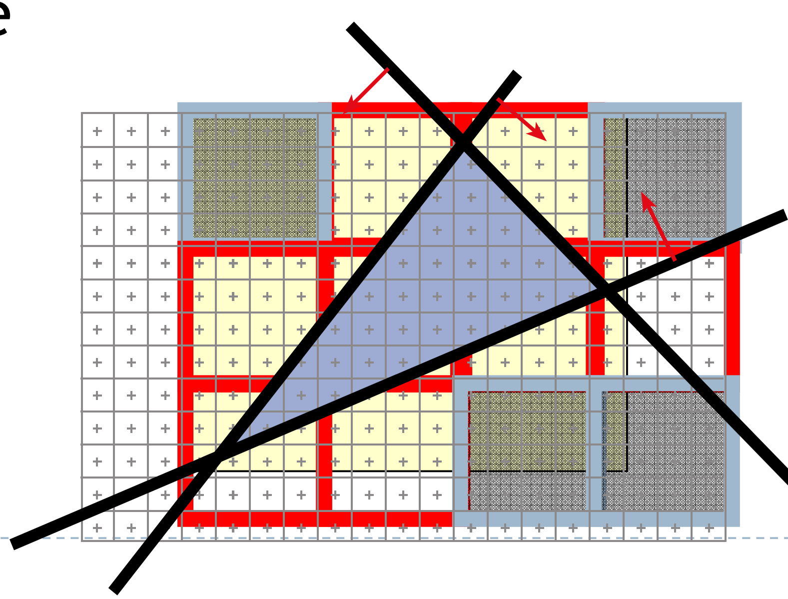
- So far, we compute barycentric coordinates of many useless pixels
- How can this be improved?



Rasterization

Hierarchy

- If block of pixels is outside triangle, no need to test individual pixels
- Can have several levels, usually two-level
- Find right granularity and size of blocks for optimal performance



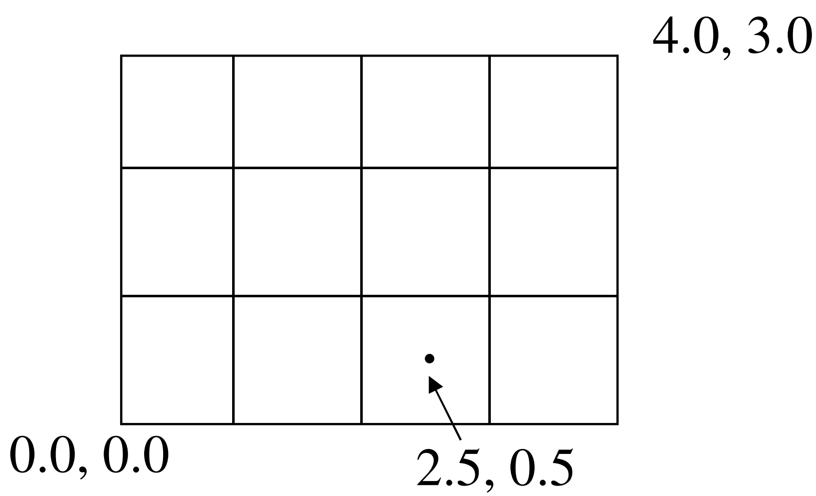
2D Triangle-Rectangle Intersection

- If one of the following tests returns true, the triangle intersects the rectangle:
  - Test if any of the triangle's vertices are inside the rectangle (e.g., by comparing the x/y coordinates to the min/max x/y coordinates of the rectangle)
  - Test if one of the quad's vertices is inside the triangle (e.g., using barycentric coordinates)
  - Intersect all edges of the triangle with all edges of the rectangle

Rasterization

Where is the center of a pixel?

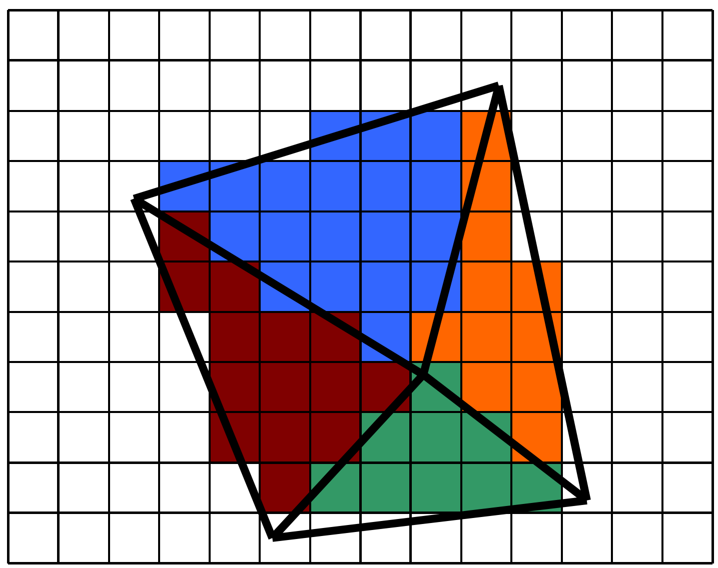
- Depends on conventions
- With our viewport transformation:
  - 800 x 600 pixels ⇔ viewport coordinates are in [0…800]x[0…600]
  - Center of lower left pixel is 0.5, 0.5
  - Center of upper right pixel is 799.5, 599.5



Rasterization

Shared Edges

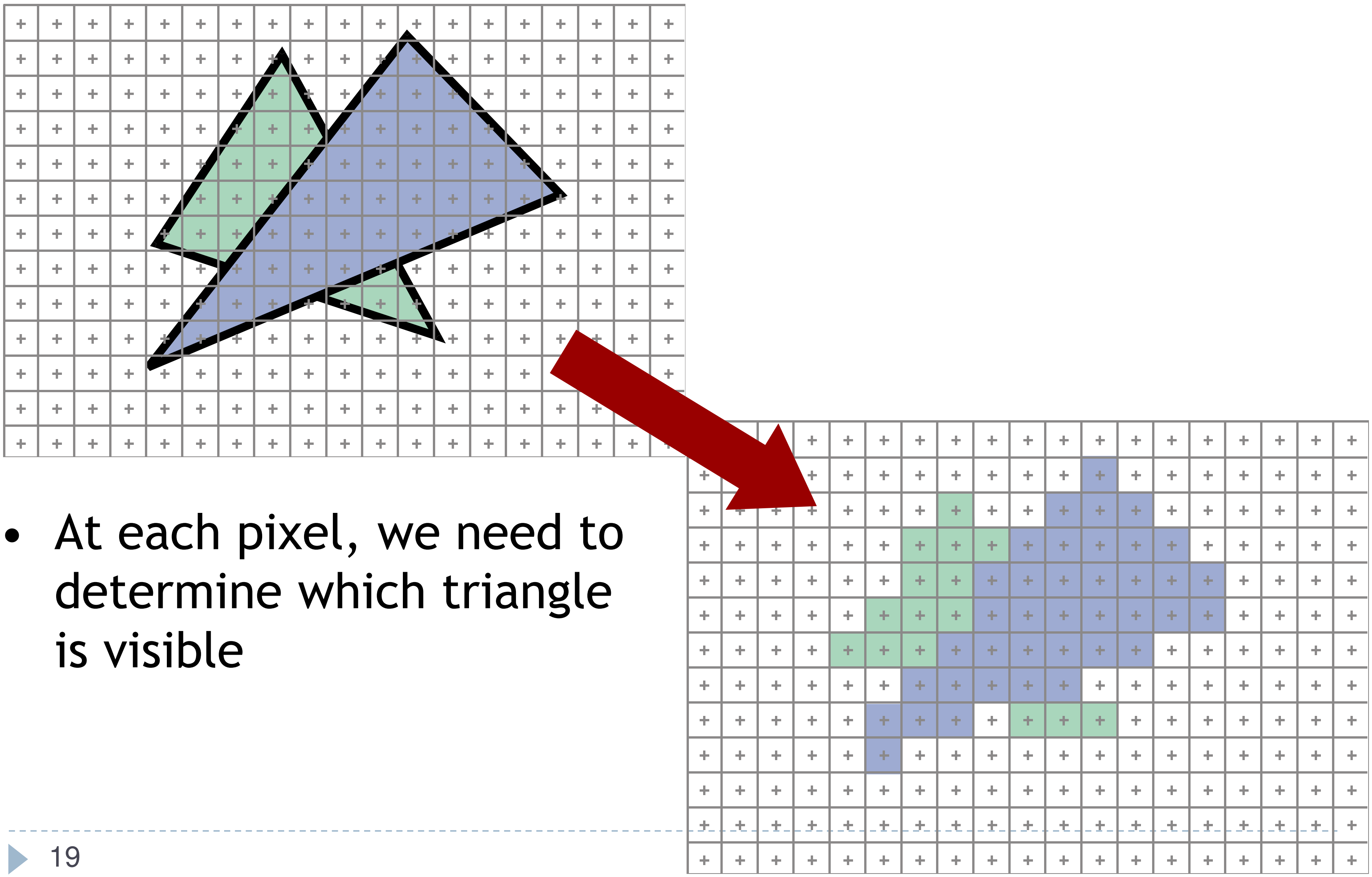
- Each pixel needs to be rasterized exactly once
- Resulting image is independent of drawing order
- Rule: If pixel center exactly touches an edge or vertex
  - Fill pixel only if triangle extends to the right or down



Lecture Overview

- Rasterization
- Visibility
- Barycentric Coordinates
- Color: Physical Background
- Color Perception

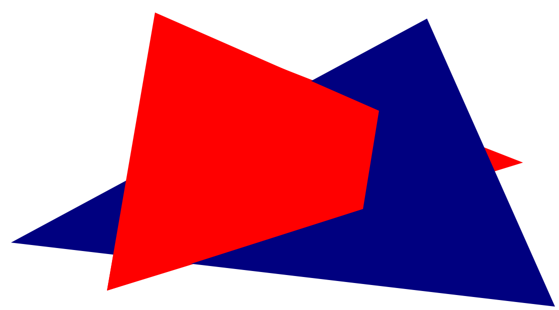
Visibility



- At each pixel, we need to determine which triangle is visible

Painter's Algorithm

- Paint from back to front
- Every new pixel always paints over previous pixel in frame buffer
- Need to sort geometry according to depth
- May need to split triangles if they intersect



- Outdated algorithm, created when memory was expensive

Z-Buffering

- Store z-value for each pixel
- Depth test
  - During rasterization, compare stored value to new value
  - Update pixel only if new value is smaller

```
setpixel(int x, int y, color c, float z)
if(z<zbuffer(x,y)) then
  zbuffer(x,y) = z
  color(x,y) = c
```

- z-buffer is dedicated memory reserved for GPU (graphics memory)
- Depth test is performed by GPU

Z-Buffering in OpenGL

- In your application:
  - Ask for a depth buffer when you create your window.
  - Place a call to glEnable (GL_DEPTH_TEST) in your program's initialization routine.
  - Ensure that your *zNear* and *zFar* clipping planes are set correctly (in glOrtho, glFrustum or gluPerspective) and in a way that provides adequate depth buffer precision.
  - Pass GL_DEPTH_BUFFER_BIT as a parameter to glClear.

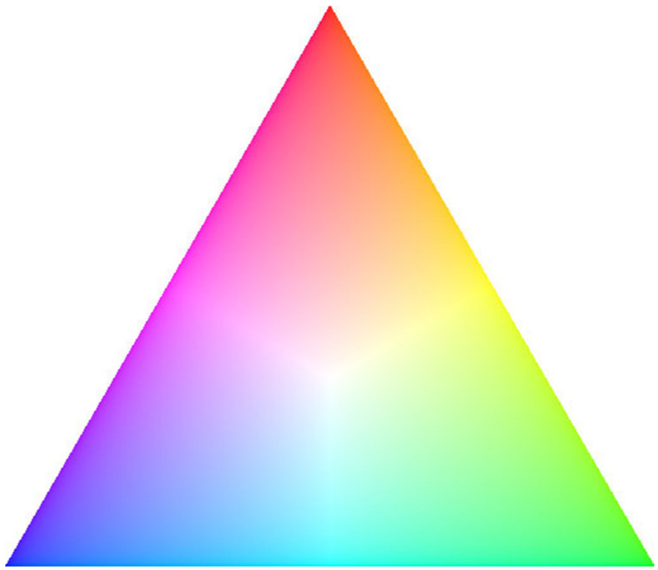
Z-Buffering

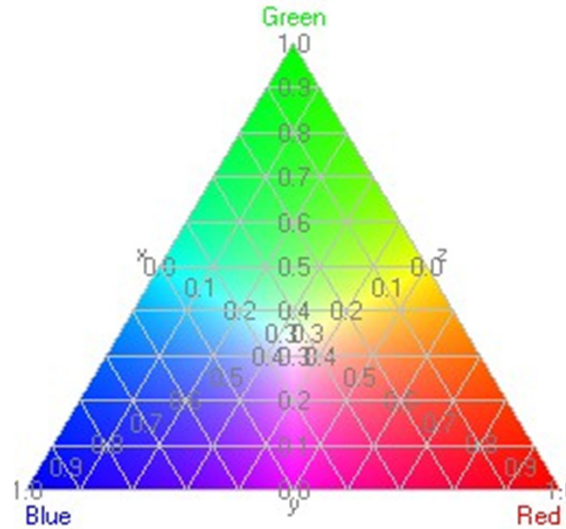
- Problem: translucent geometry
  - Storage of multiple depth and color values per pixel (not practical in real-time graphics)
  - Or back to front rendering of translucent geometry, after rendering opaque geometry
    - Does not always work correctly: programmer has to weight rendering correctness against computational effort

Lecture Overview

- Rasterization
- Visibility
- Barycentric Coordinates
- Color: Physical Background
- Color Perception

Color Interpolation



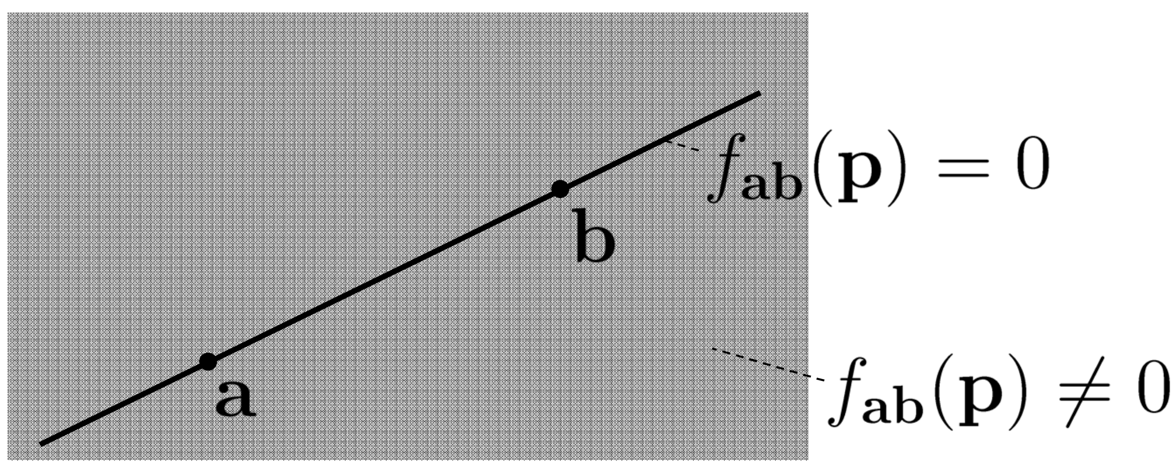


Source: efg's computer lab

- What if a triangle's vertex colors are different?
- Need to interpolate across triangle
  - How to calculate interpolation weights?

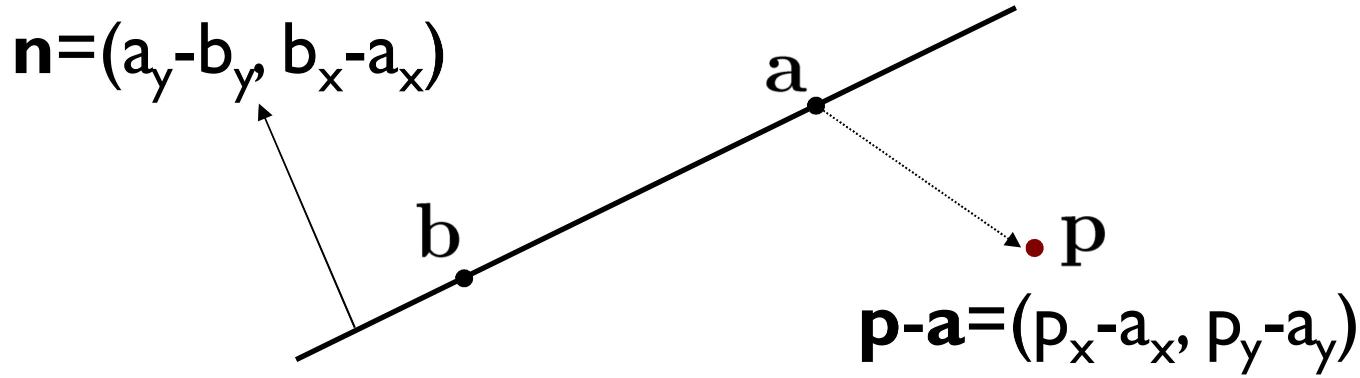
Implicit 2D Lines

- Given two 2D points **a, b**
- Define function $f_{\mathbf{ab}}(\mathbf{p})$ such that $f_{\mathbf{ab}}(\mathbf{p}) = 0$ if **p** lies on the line defined by **a, b**



Implicit 2D Lines

- Point **p** lies on the line, if **p-a** is perpendicular to the normal **n** of the line

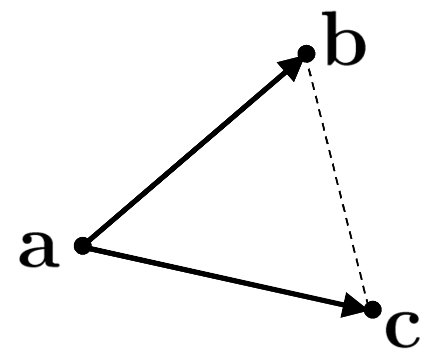


- Use dot product to determine on which side of the line **p** lies. If f(**p**)>0, **p** is on same side as normal, if f(**p**)<0 **p** is on opposite side. If dot product is 0, **p** lies on the line.

$$f_{\mathbf{ab}}(\mathbf{p}) = (a_y - b_y, b_x - a_x) \cdot (p_x - a_x, p_y - a_y)$$

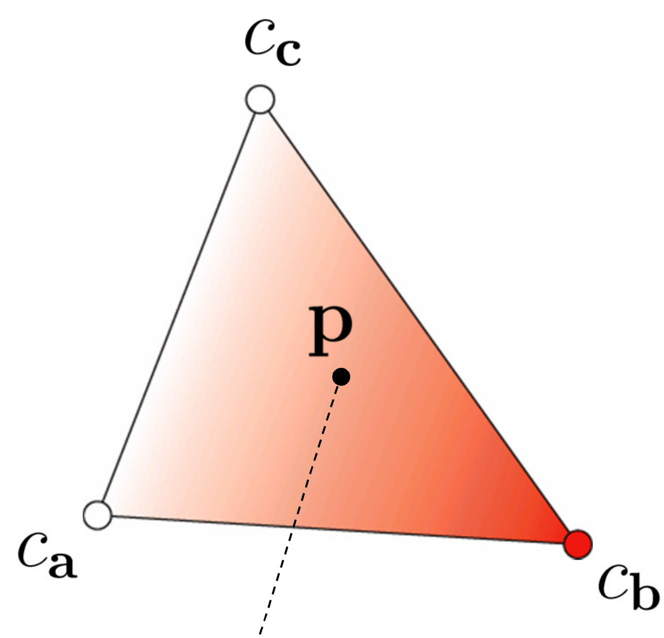
Barycentric Coordinates

- Coordinates for 2D plane defined by triangle vertices **a, b, c**
- Any point **p** in the plane defined by **a, b, c** is
  $\mathbf{p} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a})$
- Solved for a, b, c:
  $\mathbf{p} = (1 - \beta - \gamma)\,\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}$
- We define $\alpha = 1 - \beta - \gamma$
  ➔ $\mathbf{p} = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}$
- α, β, γ are called **barycentric** coordinates
- If we imagine masses equal to α, β, γ in the locations of the vertices of the triangle, the center of mass (the Barycenter) is then **p**. This is the origin of the term "barycentric" (introduced 1827 by Möbius)



Barycentric Interpolation

- Interpolate values across triangles, e.g., colors



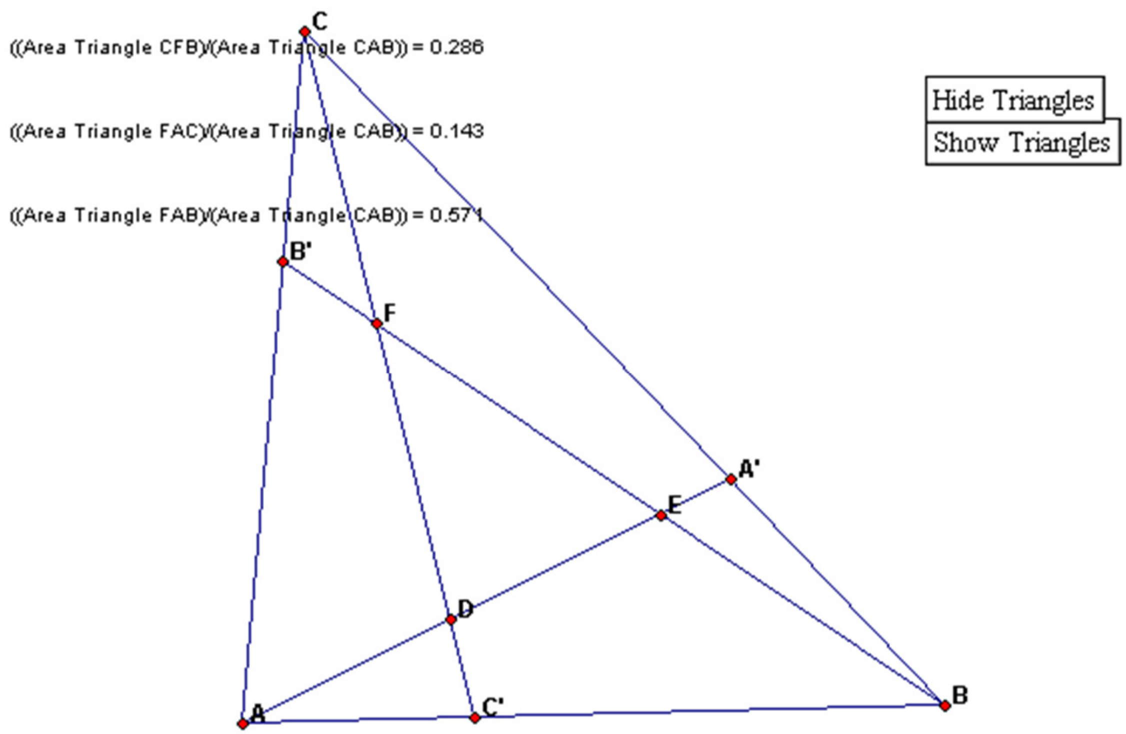
- Done by linear interpolation on triangle:

$$c(\mathbf{p}) = \alpha(\mathbf{p})c_{\mathbf{a}} + \beta(\mathbf{p})c_{\mathbf{b}} + \gamma(\mathbf{p})c_{\mathbf{c}}$$

- Works well at common edges of neighboring triangles

Barycentric Coordinates

- Demo Applet:
  - http://www.math.washington.edu/~king/java/gsp/one-third-triangle-area.html



Lecture Overview

- Rasterization
- Visibility
- Barycentric Coordinates
- Color: Physical Background
- Color Perception

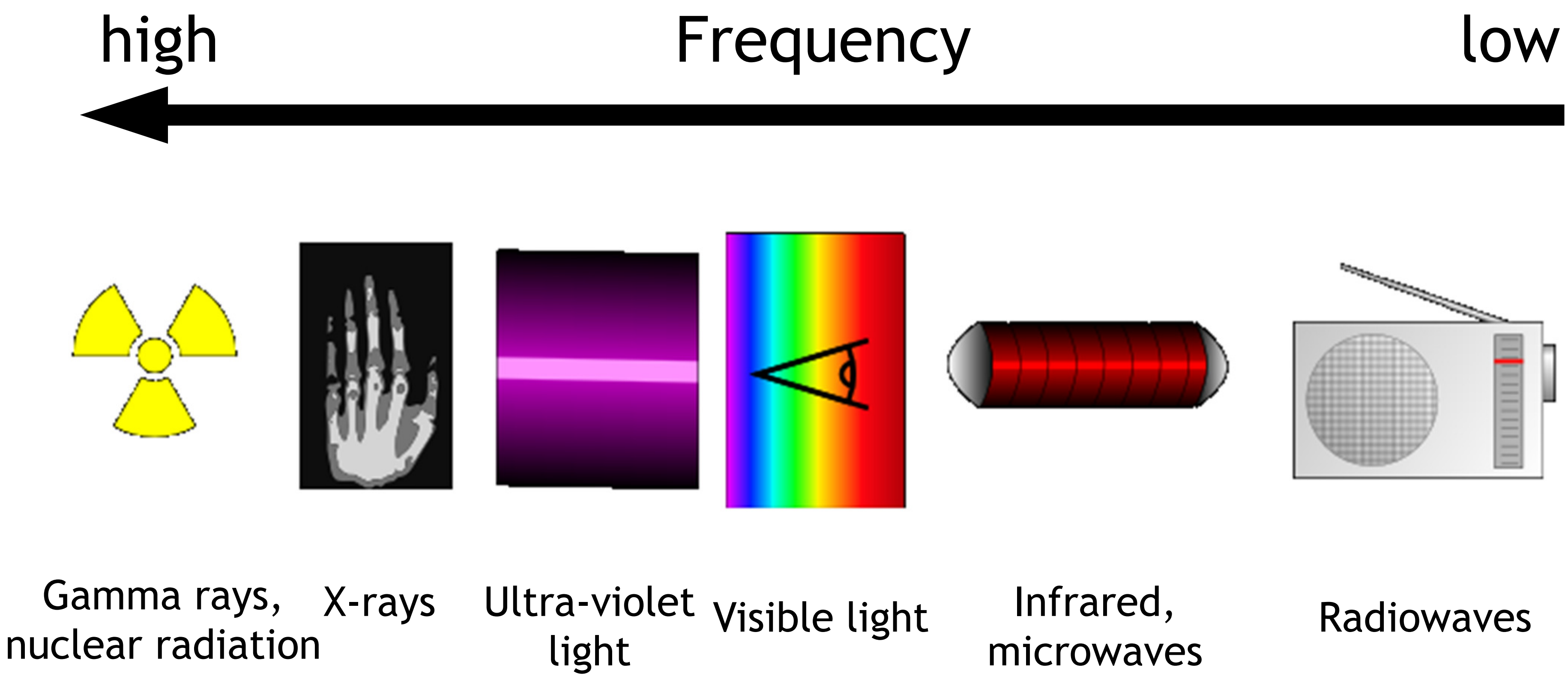
Light

Physical models

- Electromagnetic waves [Maxwell 1862]
- Photons (tiny particles) [Planck 1900]
- Wave-particle duality [Einstein, early 1900]
  "It depends on the experiment you are doing whether light behaves as particles or waves"
- Simplified models in computer graphics

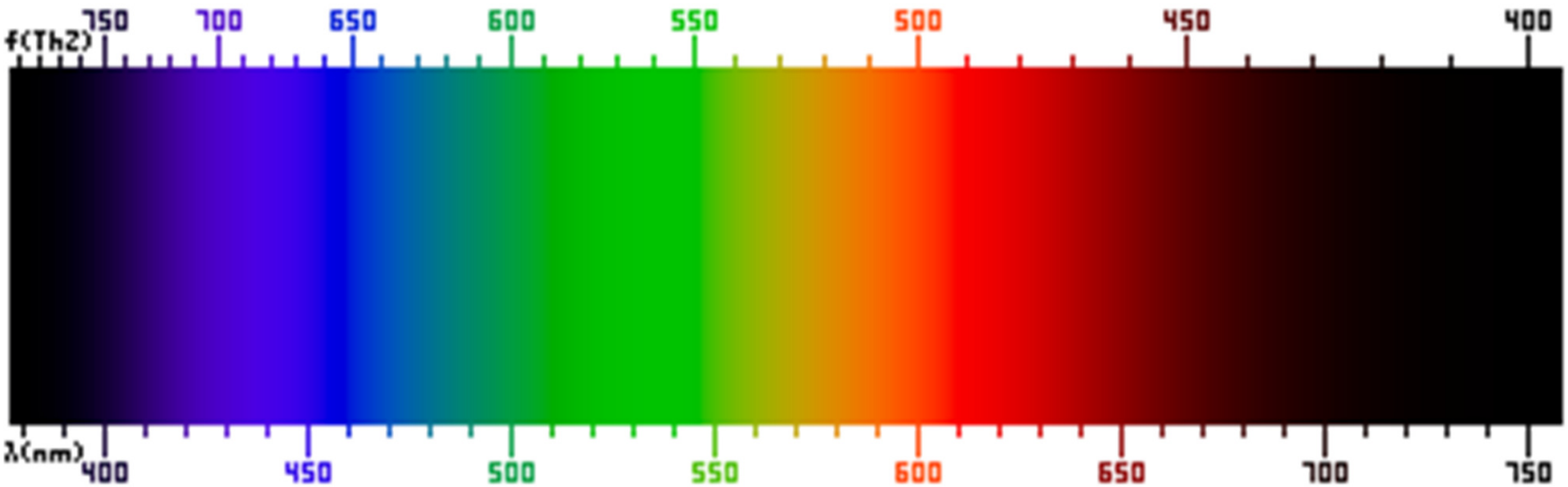
Electromagnetic Waves

- Large range of frequencies



Visible Light

Frequency



Wavelength: 1nm=10^-9 meters

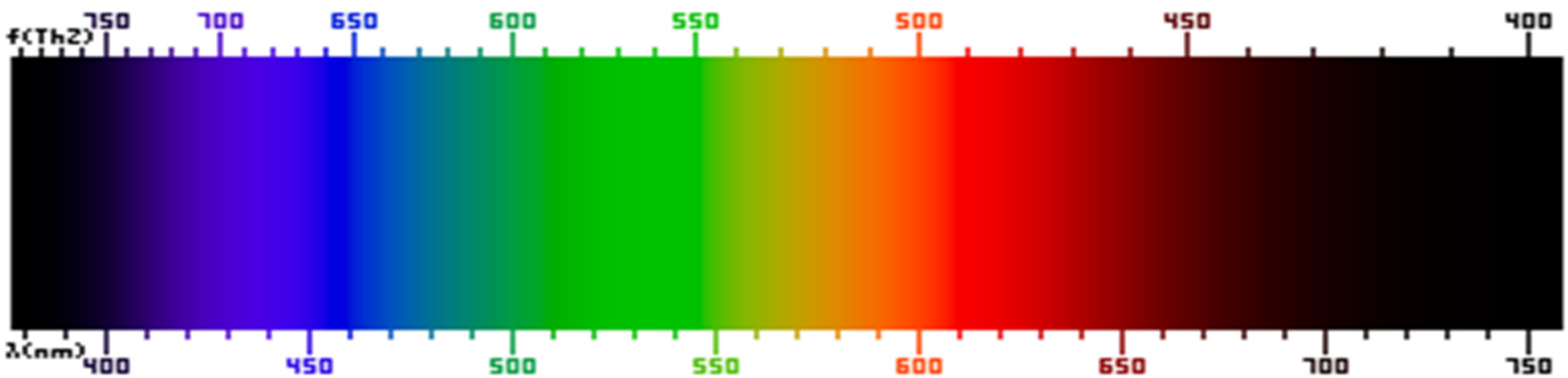
speed of light = wavelength * frequency

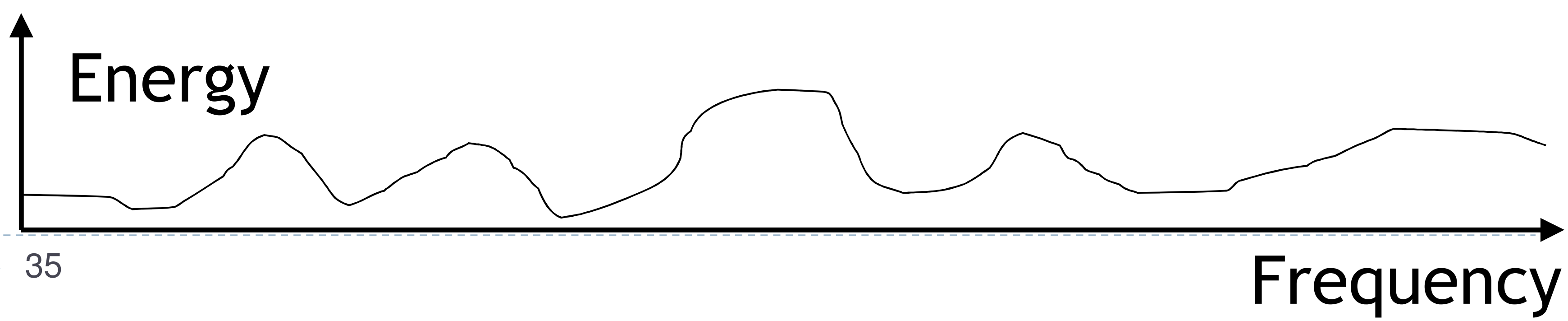
Example 91.1MHz: $$\frac{300*10^6 \frac{m}{s}}{91.1*10^6 \frac{1}{s}} = 3.29m$$

Light Transport

Simplified model in computer graphics

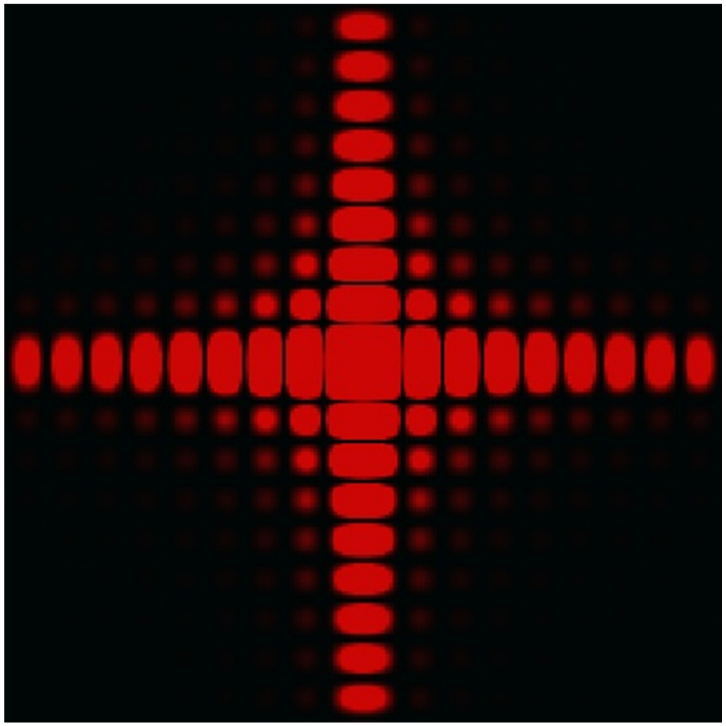
- Light is transported along straight rays
- Rays carry a spectrum of electromagnetic energy





Limitations

- OpenGL ignores wave nature of light
  - → no diffraction effects



Diffraction pattern of a small square aperture

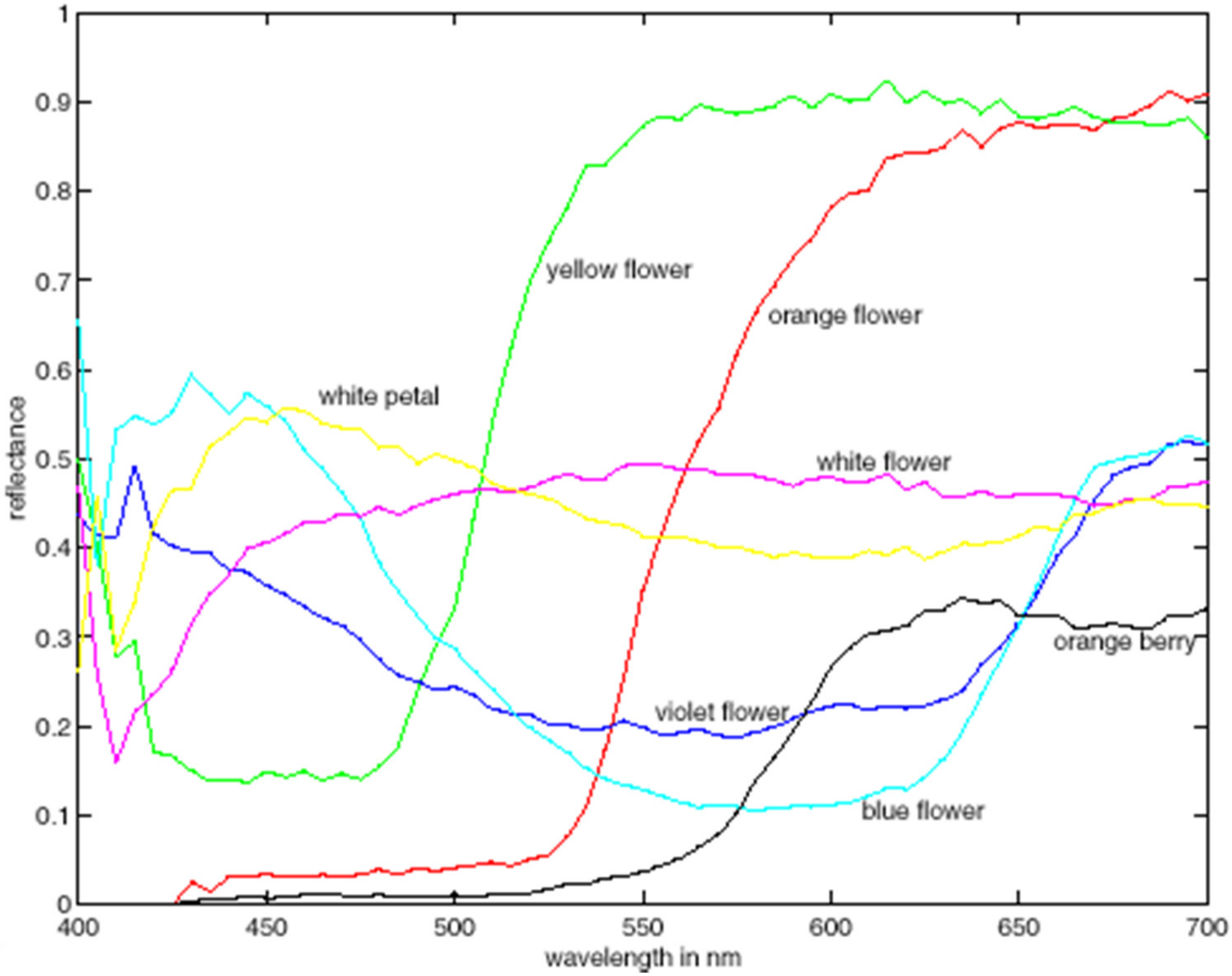
Surface of a CD shows diffraction grating

Lecture Overview

- Rasterization
- Visibility
- Barycentric Coordinates
- Color: Physical Background
- Color Perception

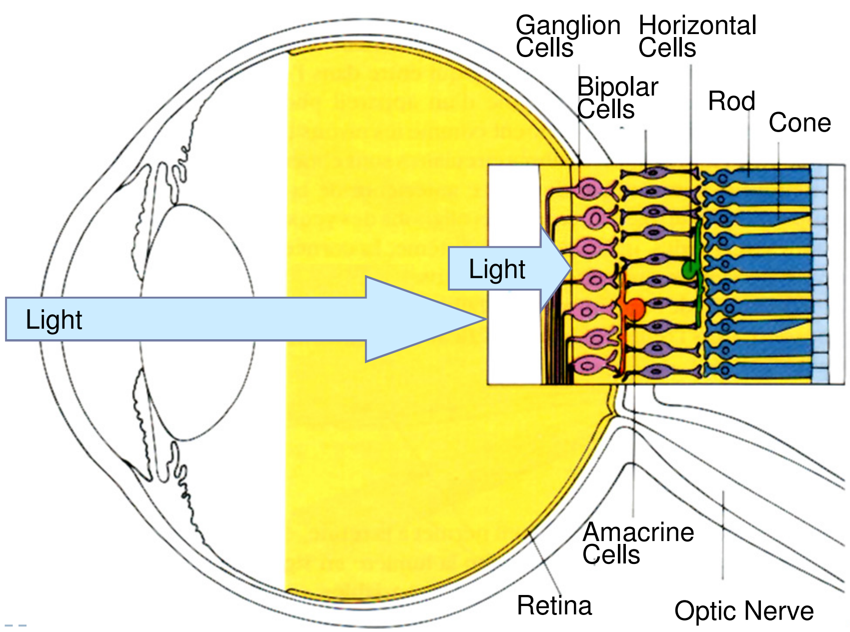
Light and Color

- Different spectra may be perceived as the same color

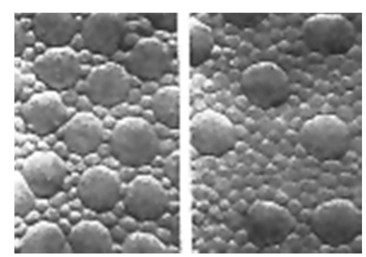


Color Perception

- Photoreceptor cells
- Light sensitive
- Two types, rods and cones



Rods Cones



Distribution of Cones and Rods

Photoreceptor Cells

Rods

- More than 1,000 times more sensitive than cones
- Low light vision
- Brightness perception only, no color
- Predominate in peripheral vision

Cones

- Responsible for high-resolution vision
- 3 types of cones for different wavelengths (LMS):
  - L: long, red
  - M: medium, green
  - S: short, blue

Photoreceptor Cells

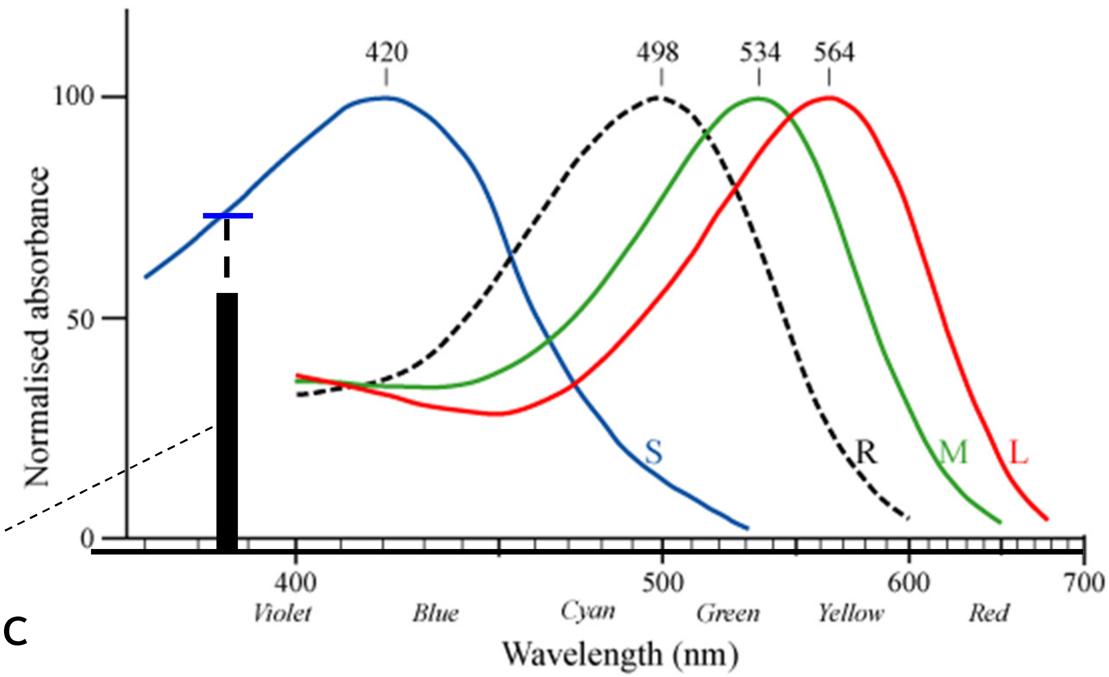
The Austrian naturalist Karl von Frisch has demonstrated that honeybees, although blind to red light, distinguish at least four different color regions, namely:

- yellow (including orange and yellow green)
- blue green
- blue (including purple and violet)
- ultraviolet

(Source: Encyclopedia Britannica)

Photoreceptor Cells

- Response curves $s(\lambda), m(\lambda), l(\lambda)$ to monochromatic spectral stimuli

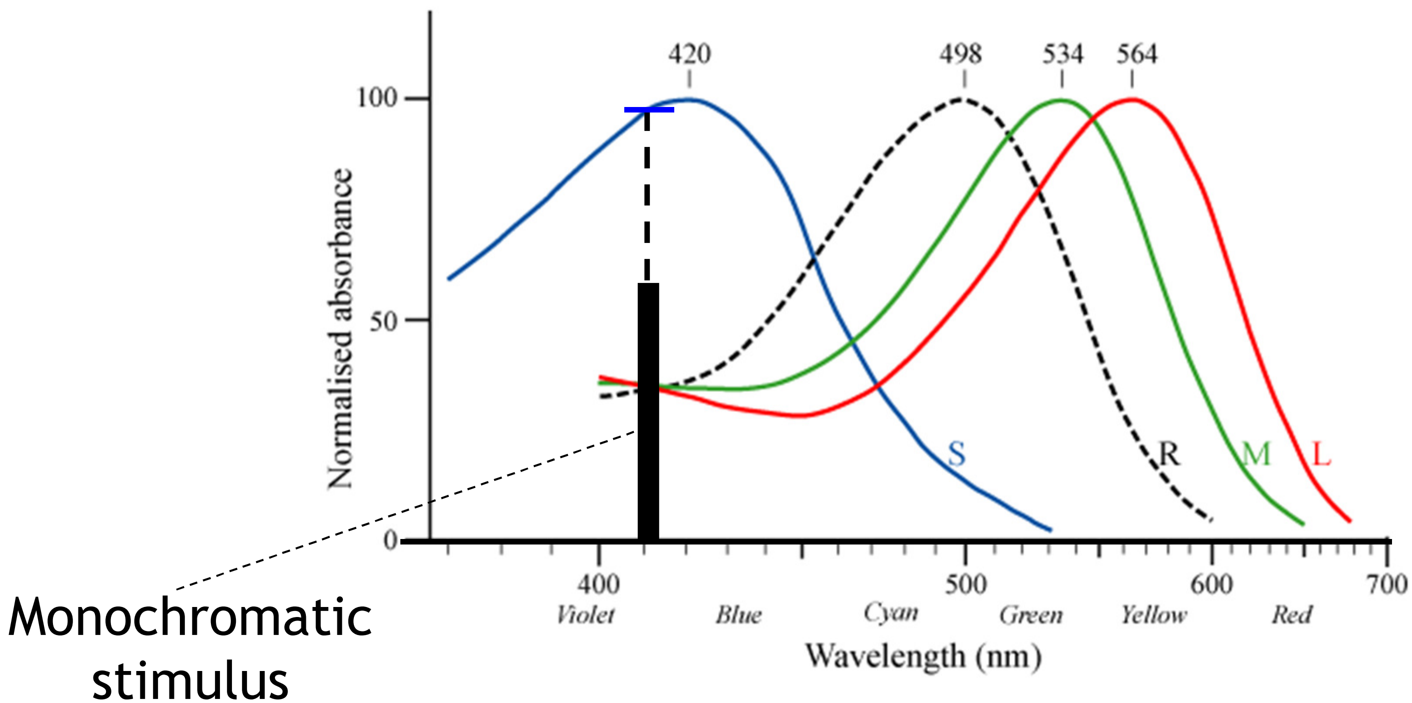


Monochromatic stimulus

- Experimentally determined in the 1980s

Photoreceptor Cells

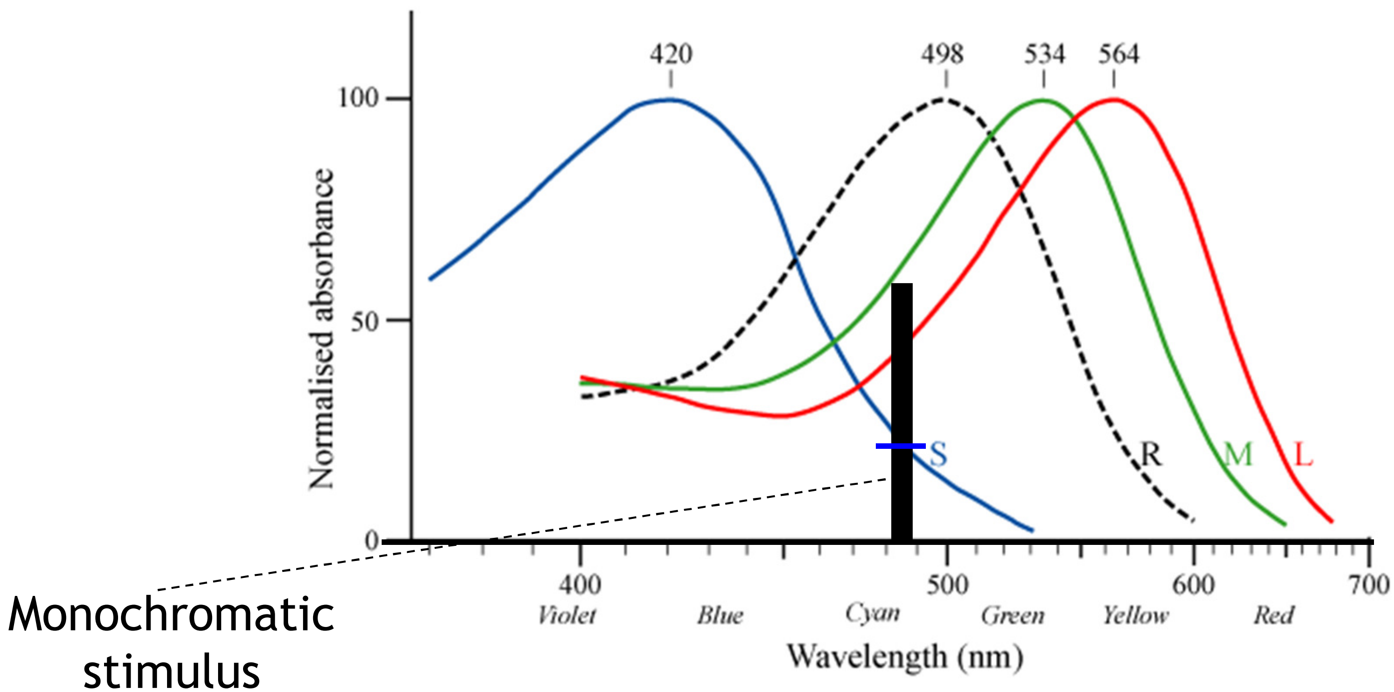
- Response curves $s(\lambda), m(\lambda), l(\lambda)$ to monochromatic spectral stimuli



- Experimentally determined in the 1980s

Photoreceptor Cells

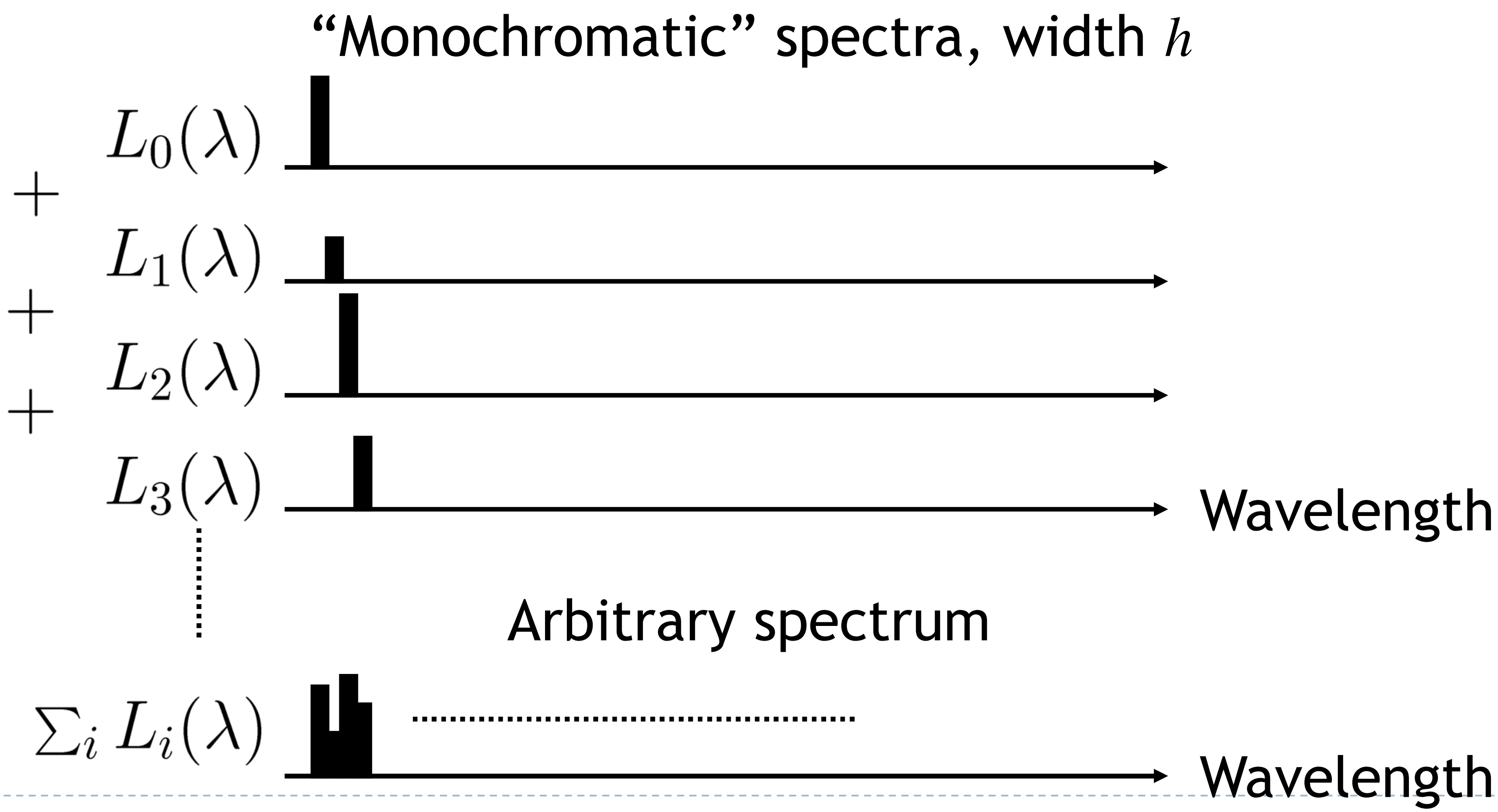
- Response curves $s(\lambda), m(\lambda), l(\lambda)$ to monochromatic spectral stimuli



- Experimentally determined in the 1980s

Response to Arbitrary Spectrum

- Arbitrary spectrum as sum of "mono-chromatic" spectra



Response to Arbitrary Spectrum

Assume linearity (superposition principle)

- Response to sum of spectra is equal to sum of responses to each spectrum
- S-cone $\text{response}_s = \sum_i s(\lambda) h L_i(\lambda)$

  Input: light intensity $L(\lambda)$ impulse width *h*
  Response to monochromatic impulse $s(\lambda)$
- In the limit $h \rightarrow 0$

$$\text{response}_s = \int s(\lambda) L(\lambda) d\lambda$$

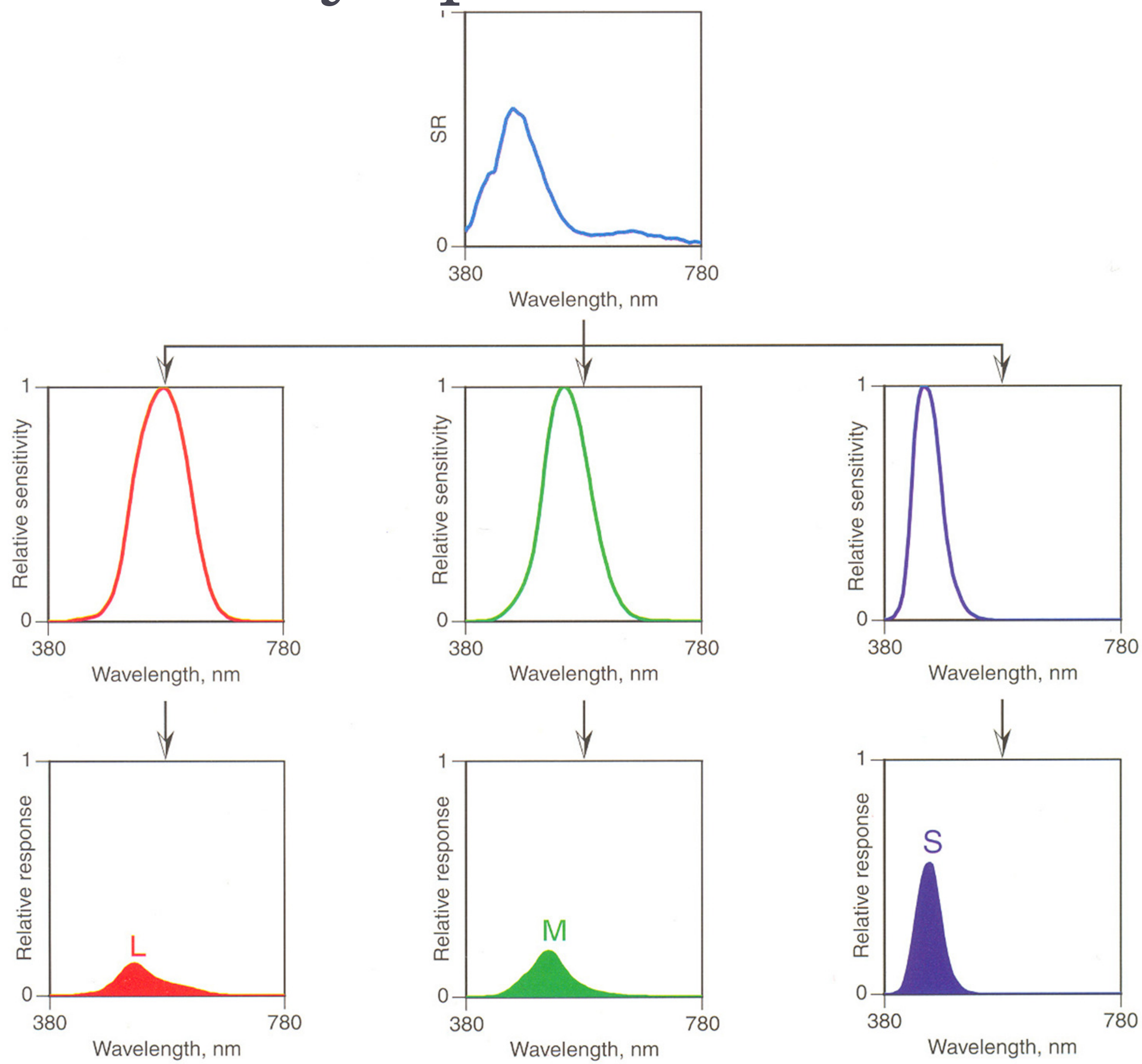
Response to Arbitrary Spectrum

Stimulus

Response curves

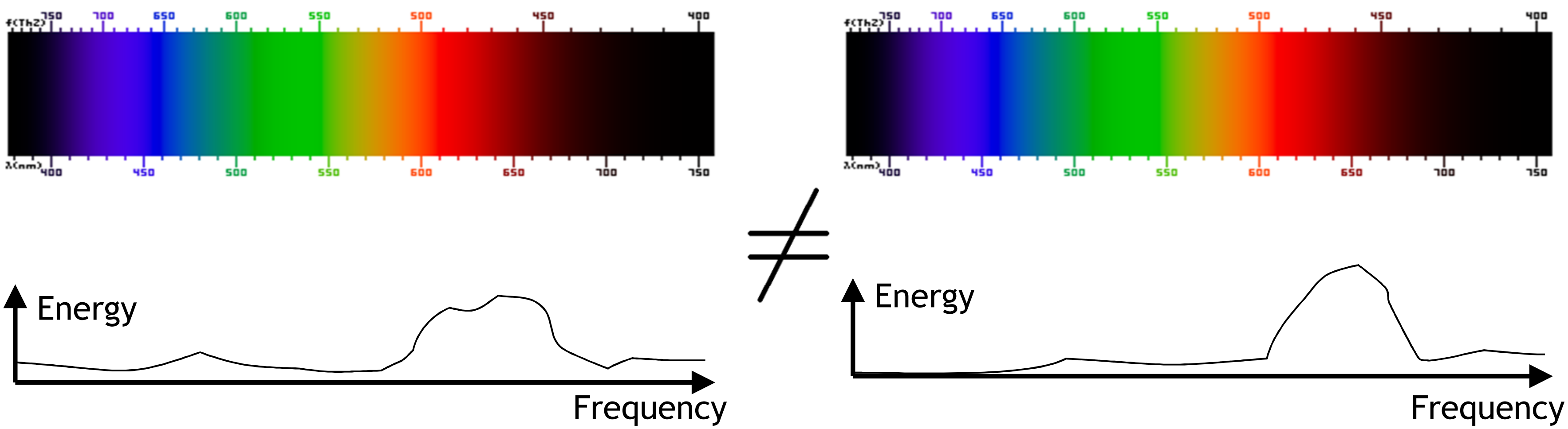
Multiply

Integrate



Metamers

- Different spectra, same response
- Cannot distinguish spectra
  - Arbitrary spectrum is *infinitely dimensional* (has infinite number of degrees of freedom)
  - Response has three dimensions
  - Information is lost



Perceived color: red = Perceived color: red

Color Blindness

- One type of cone missing, damaged
- Different types of color blindness, depending on type of cone
- Can distinguish even fewer colors
- But we are all a little color blind…