23. Colors, Images, and Animation

CPSC 120: Introduction to Programming
Pratishtha Soni~ CSU Fullerton

Agenda

0. Remind
    a. Notes Check 3 (random numbers) due Dec 3 (next Sun)
    b. 120L Portfolio due Dec 6 (next Wed)
    c. 120A Exam, Dec 11 and 13 (finals week)
    d. Student Opinion Questionnaires
1. Technical Q&A
2. Socially Responsible Computing Focus Group
3. Colors
4. Images
5. Animation

1. Technical Q&A

Technical Q&A

Let’s hear your noted questions about...

- This week’s Lab
- Linux
- Any other technical issues

Reminder: write these questions in your notebook during lab

2. Socially Responsible Computing Focus Group

Socially Responsible Computing Focus Group

- If you participate in the focus group, you will receive a $25 gift card
- https://www.surveymonkey.com/r/SRC_student_focusgroup

2. Colors

Electromagnetic Radiation

- Electromagnetic radiation: waves of energy that travel through space
  - Radio waves (wifi, Bluetooth)
  - Microwaves (microwave oven)
  - Infrared light (remote controls)
  - Ultraviolet (UV) light (sunburn)
  - Visible light (vision)
  - X-rays (medical imaging)
  - Gamma rays (radioactive decay)
- See *PHYS 226 - Fundamental Physics: Electricity and Magnetism*

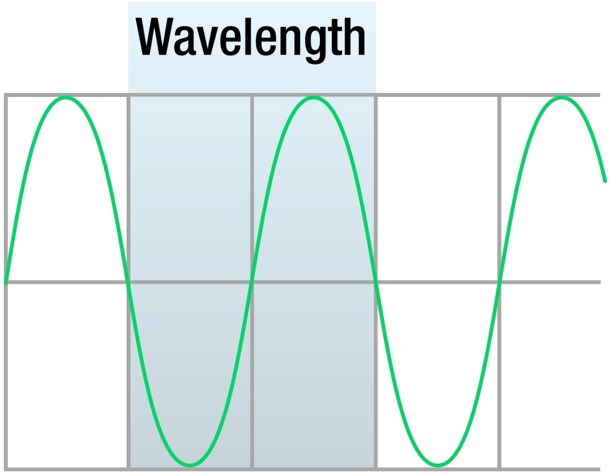


Image credit: hubblesite.org

Electromagnetic Waves

- EM radiation forms a **wave**
- Oscillates over time
- Analogy to water wave
- **Shape** of wave dictates what kind of phenomenon it is
    - Visible light, microwaves etc.

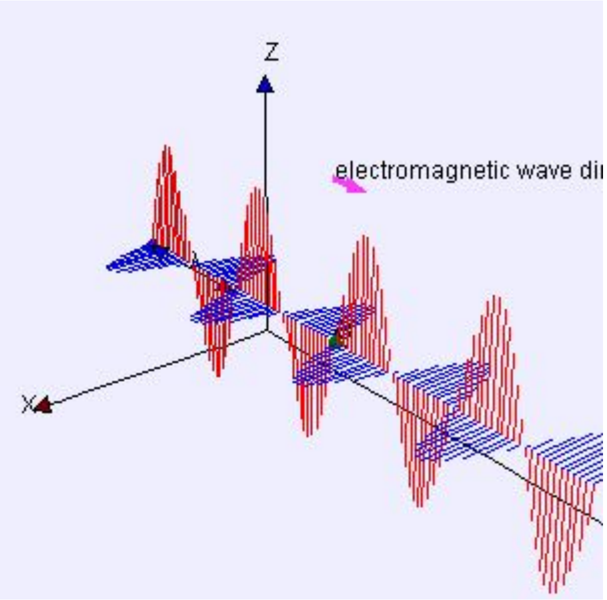


Image credit: Wikipedia

Amplitude

- **Amplitude:** height of wave
- Determines **energy** (strength) of wave
- Higher amplitude = more energy
- Explore amplitude: EMANIM
- *Audio amplifier:* increase amplitude, leave other aspects unchanged

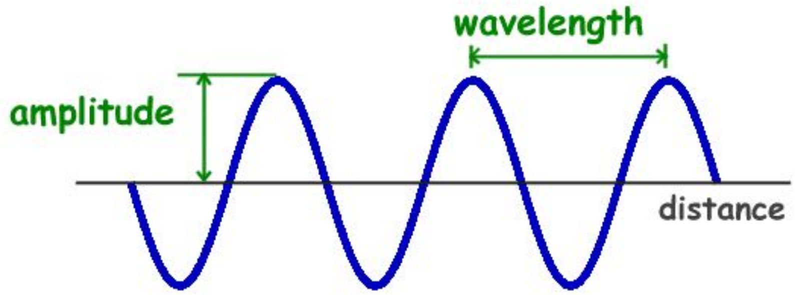


Image credit: ducksters.com

Wavelength

- **Wavelength:** length of wave, before it repeats
- Unit of length
  - Inch, meter
- Explore wavelength: EMANIM
- Determines type of radiation (light, microwave, etc.)

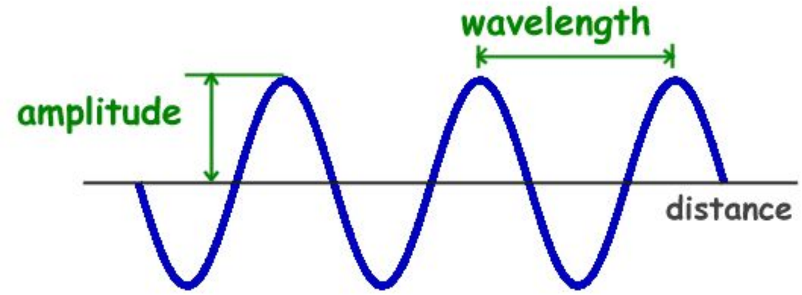


Image credit: ducksters.com

The Electromagnetic (EM) Spectrum

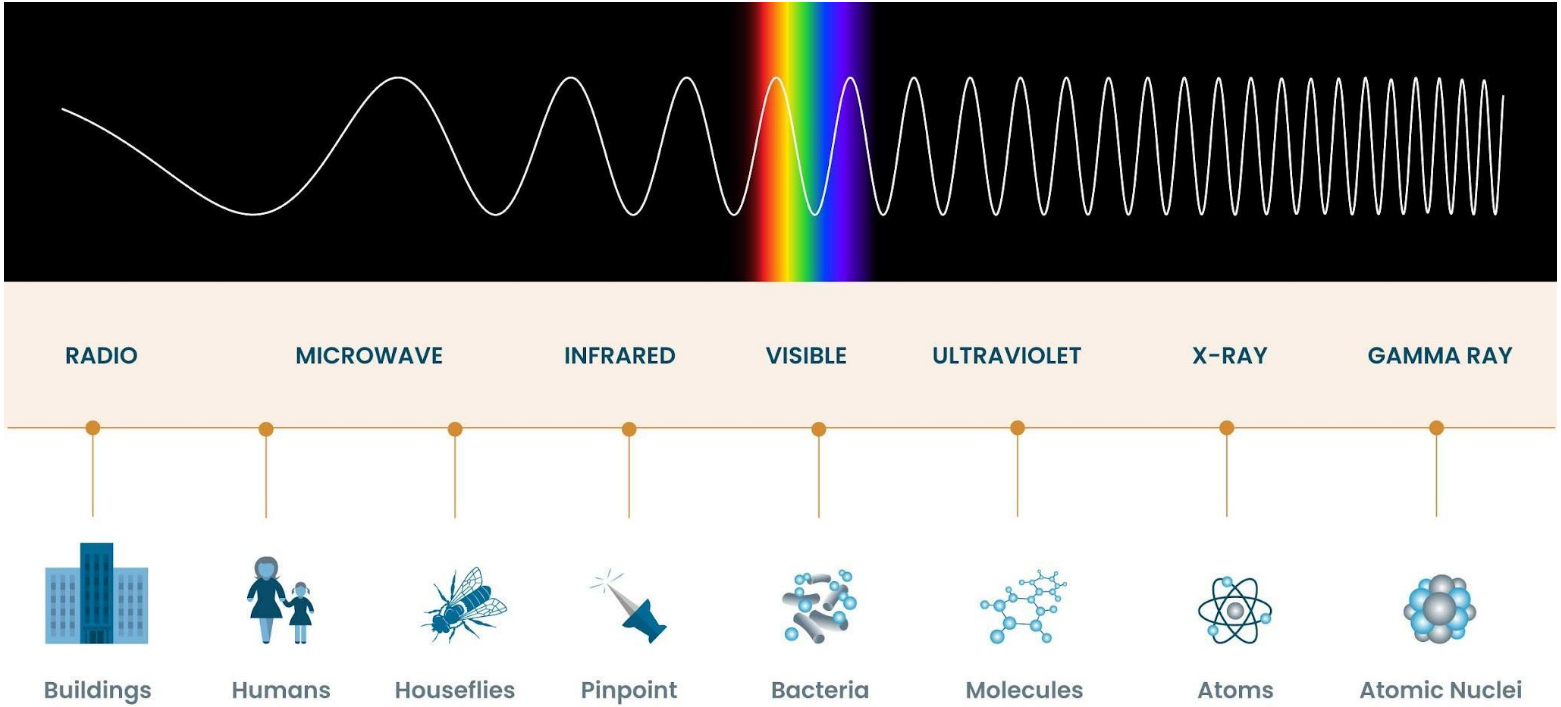
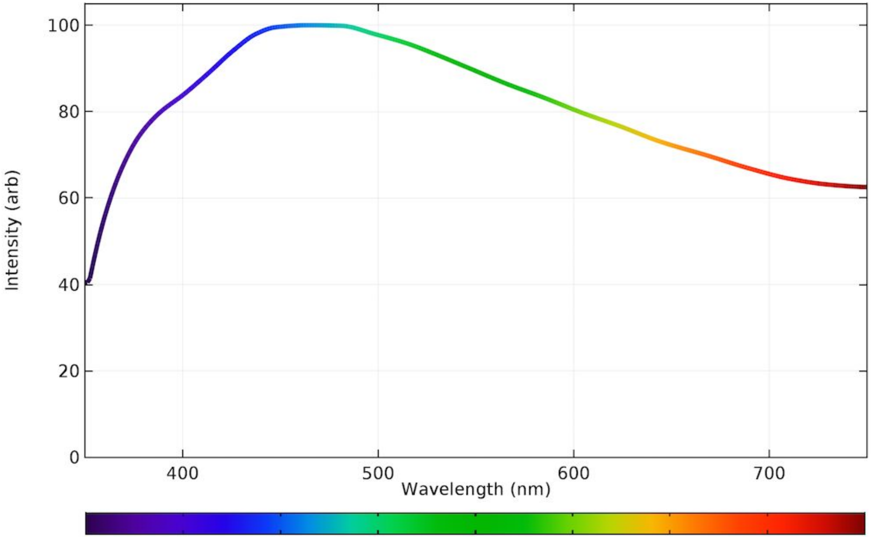


Image credit: hubblesite.org

A Light Source Contains Many Wavelengths

Sunlight



Fluorescent Bulb

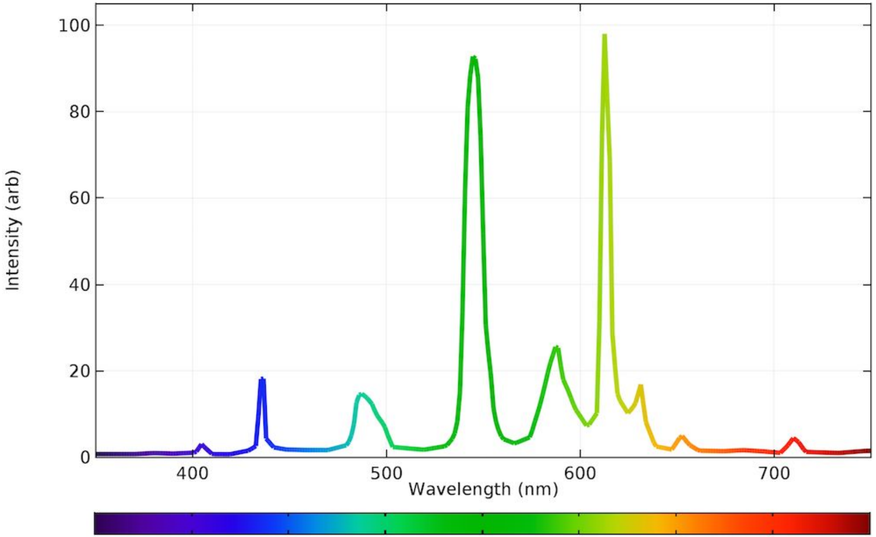


Image credit: comsol.com

The Human Eye

- Anatomy focuses light on the **retina**
- **Photoreceptors:** neurons that convert light waves to electrical signals
- Two kinds of photoreceptors
  - **Rods**
  - **Cones**
- Nerves carry electrical signals to brain

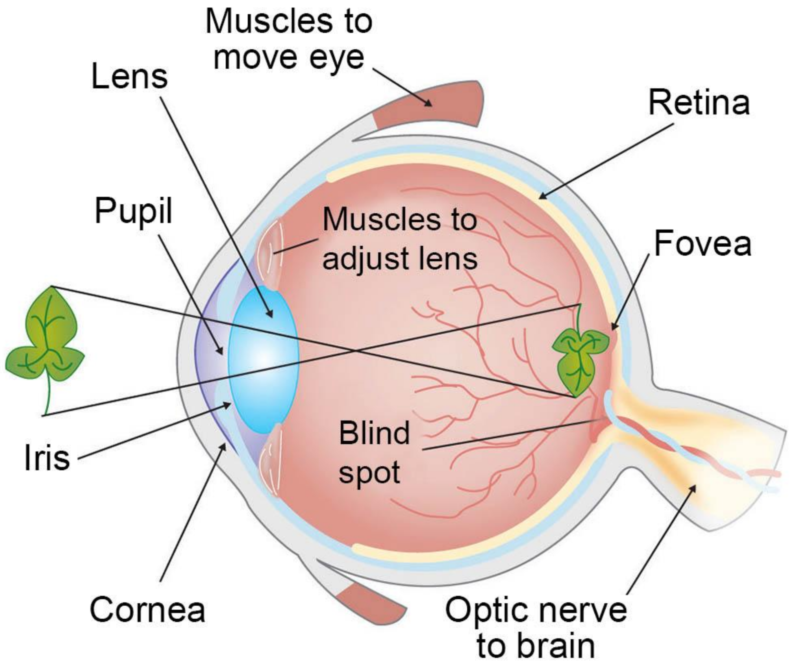
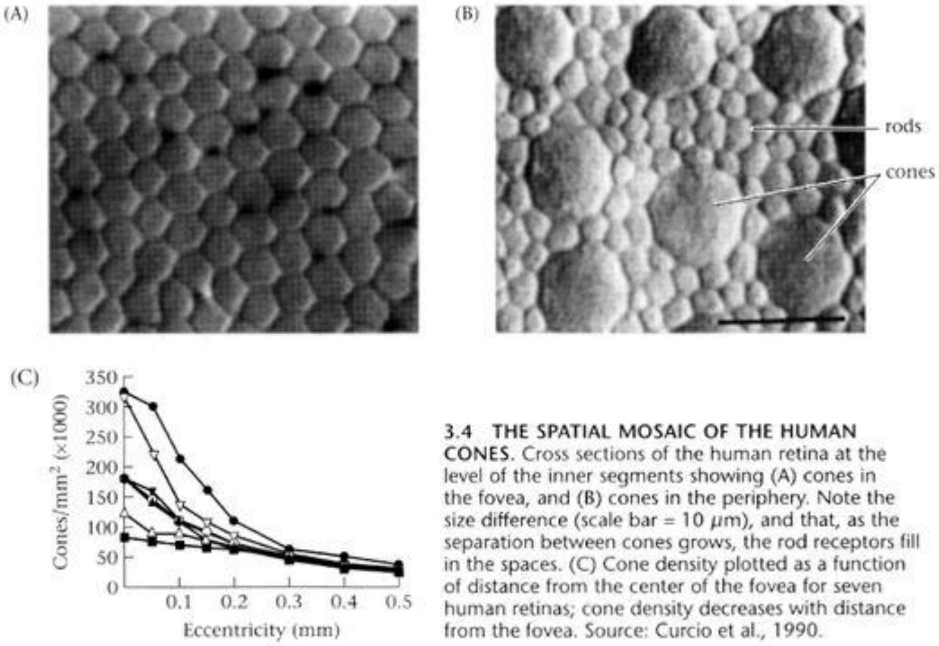


Image credit: askabiologist.asu.edu

Rods and Cones

- **Cones**
  - Three kinds
  - Create perception of **color**
  - Require bright light (high amplitude)
- **Rods**
  - One kind
  - Work in low light (low amplitude)
  - No perception of color
  - Why we see black-and-white at night



3.4 THE SPATIAL MOSAIC OF THE HUMAN CONES. Cross sections of the human retina at the level of the inner segments showing (A) cones in the fovea, and (B) cones in the periphery. Note the size difference (scale bar = 10 μm), and that, as the separation between cones grows, the rod receptors fill in the spaces. (C) Cone density plotted as a function of distance from the center of the fovea for seven human retinas; cone density decreases with distance from the fovea. Source: Curcio et al., 1990.

Image credit: cis.rit.edu

Three Kinds of Cones

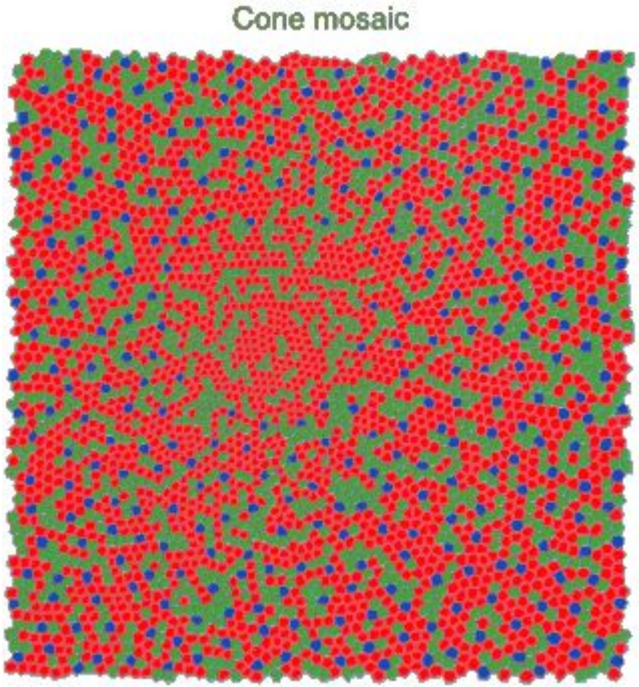


Image credit: cis.rit.edu

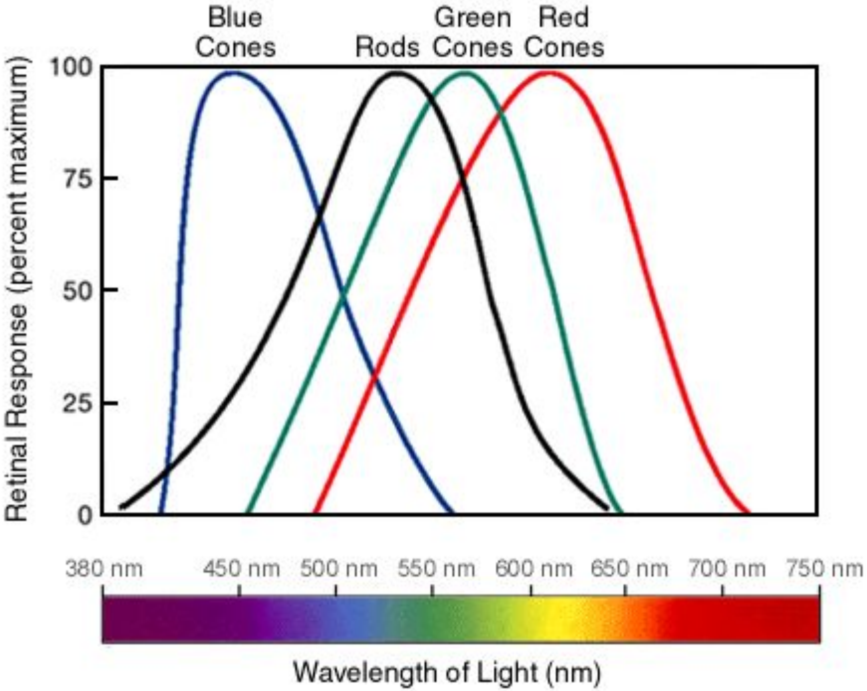


Image credit: askabiologist.asu.edu

Modeling a Color as Three Numbers

- ER radiation in the visible spectrum is made up of many waves of differing wavelengths
- Human eye **summarizes** this as just three things
- How much the...
  - blue cones are activated,
  - green cones are activated, and
  - red cones are activated

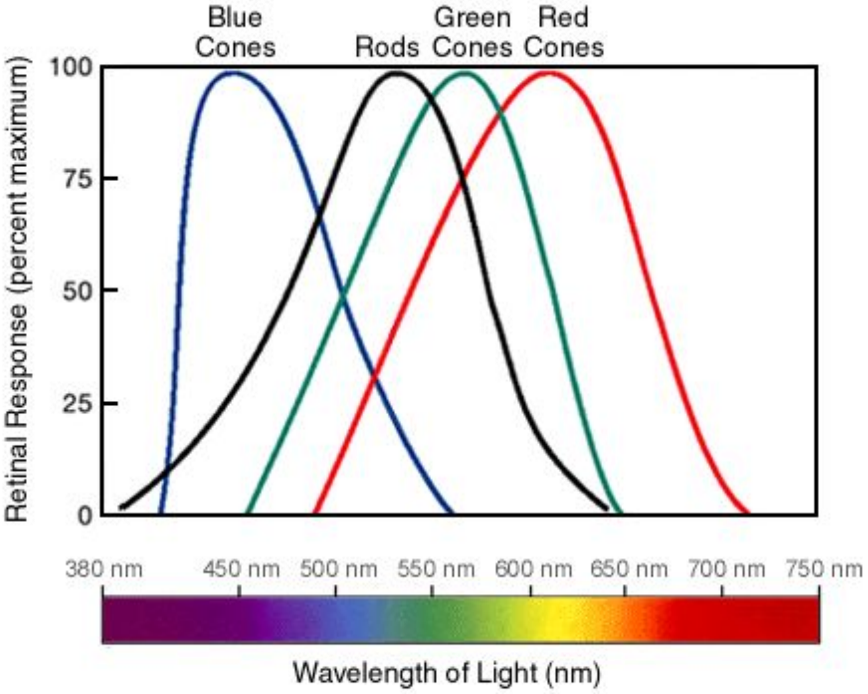


Image credit: askabiologist.asu.edu

RGB Color Model

- Color model: approach to representing a color as three numbers
- RGB color model: represent a specific color as...
  - amplitude of **red** (R)
  - amplitude of **green** (G)
  - amplitude of **blue** (B)
- **Component:** one of the R, G, B parts
- Can represent any visible color
- Explore: RapidTables RGB Color Codes Chart

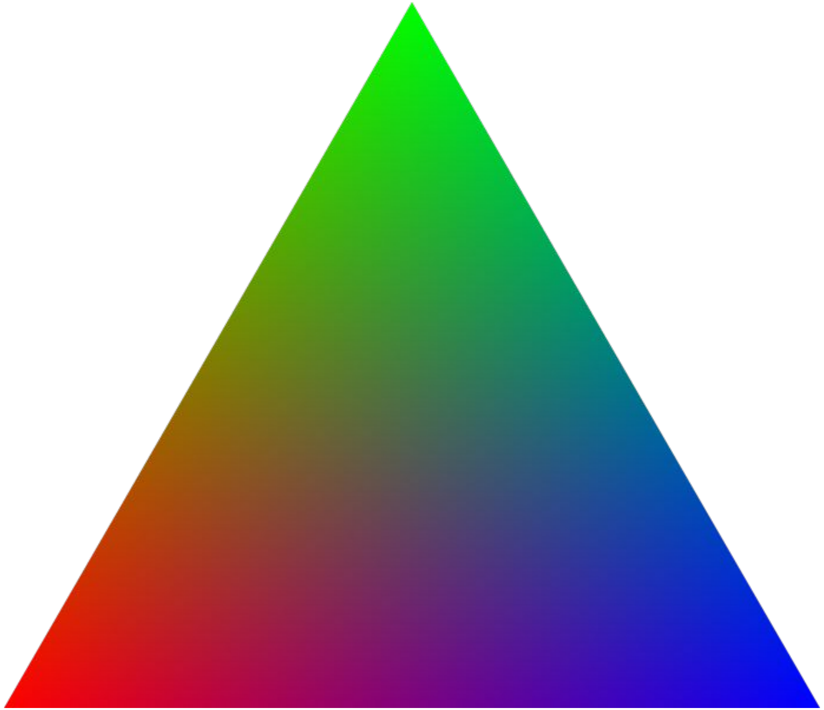


Image credit: Wikipedia

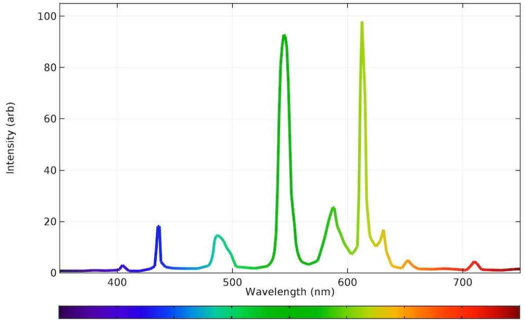
Representing Each Color Component

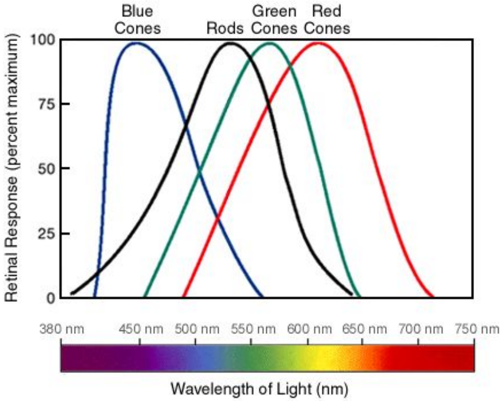
- Need: data type to model components in code
- Two common alternatives

Data Type	Minimum (dark)	Maximum (bright)	White	Black	Red
`int`	0	255	255, 255, 255	0, 0, 0	255, 0, 0
`double`	0.0	1.0	1.0, 1.0, 1.0	0.0, 0.0, 0.0	1.0, 0.0, 0.0

Light is Not Actually Three-Dimensional

- Reminder
- Light is made up of many (e.g. millions) of different waves
- Human eye is limited to measuring three kinds of waves
- RGB color model is a “hack” that suffices for human beings
- Astronomy, other animals, aliens, would use a different model
  - Ex. dogs have only two kinds of cones





GraphicsMagick

- API = Application Programming Interface
    - Headers, functions, classes
- GraphicsMagick: C library for colors, images, animations
- Magick++: C++ API in `<Magick++.h>`
- Used in lab
- Reuse: we use Magick++ instead of writing our own image code from scratch

GraphicsMagick Initialization

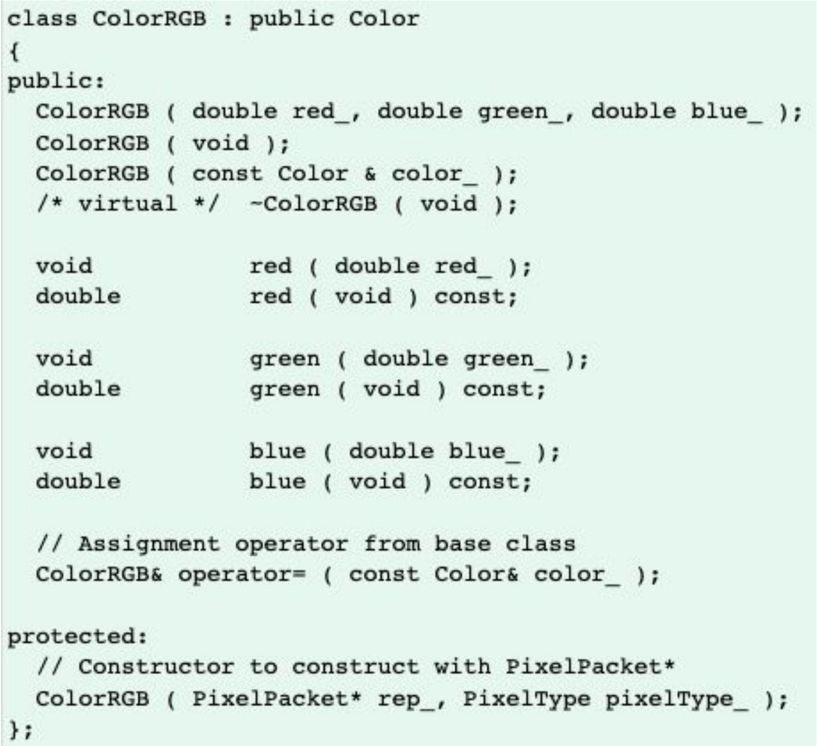
- **Initialize** (v): prepare for use
- Program must initialize GraphicsMagick before using any GraphicsMagick functions or classes
- **Boilerplate**: at start of `main`,

```
Magick::InitializeMagick(*argv);
```

```
#include <Magick++.h>

int main(int argc, char* argv[]) {
 Magick::InitializeMagick(*argv);

 ...
```

GraphicsMagick ColorRGB

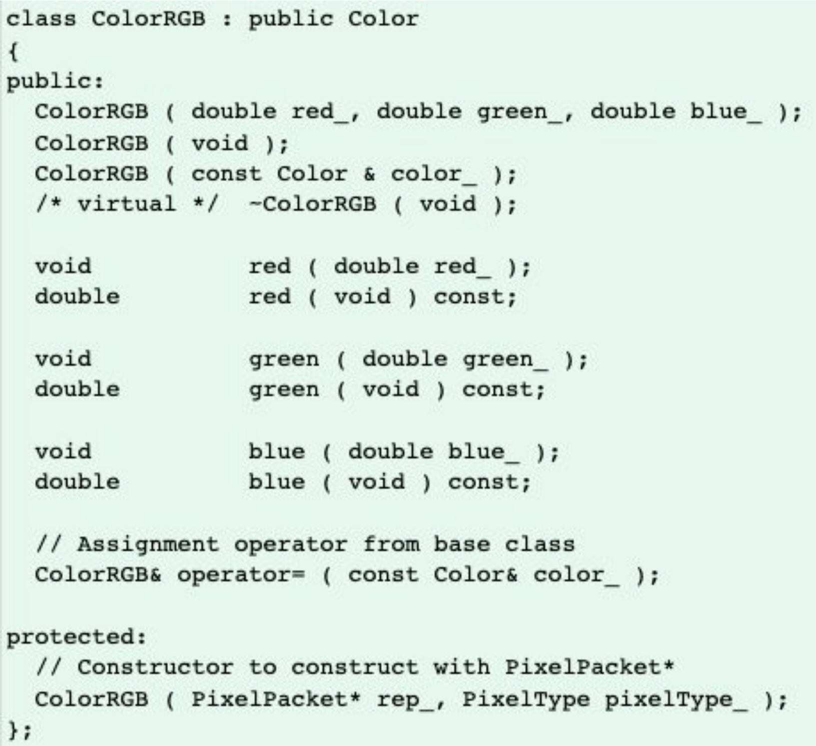
- Magick::Color API documentation
- ColorRGB class
- *"Representation of an RGB color in floating point. All color arguments have a valid range of 0.0 - 1.0."*
- Observe
  - constructor
  - accessors, mutators: red, green, blue



```
class ColorRGB : public Color
{
public:
  ColorRGB ( double red_, double green_, double blue_ );
  ColorRGB ( void );
  ColorRGB ( const Color & color_ );
  /* virtual */  ~ColorRGB ( void );

  void           red ( double red_ );
  double         red ( void ) const;

  void           green ( double green_ );
  double         green ( void ) const;

  void           blue ( double blue_ );
  double         blue ( void ) const;

  // Assignment operator from base class
  ColorRGB& operator= ( const Color& color_ );

protected:
  // Constructor to construct with PixelPacket*
  ColorRGB ( PixelPacket* rep_, PixelType pixelType_ );
};
```

Example: Construct ColorRGB objects

```
#include <Magick++.h>

int main(int argc, char* argv[]) {
 Magick::InitializeMagick(*argv);

 Magick::ColorRGB white(1.0, 1.0, 1.0);

 Magick::ColorRGB black(0.0, 0.0, 0.0);

 Magick::ColorRGB pure_red(1.0, 0.0, 0.0);

 Magick::ColorRGB pure_green(0.0, 1.0, 0.0);

 Magick::ColorRGB pure_blue(0.0, 0.0, 1.0);

 Magick::ColorRGB purple(0.8, 0.0, 0.8);


 return 0;
}
```



```
class ColorRGB : public Color
{
public:
  ColorRGB ( double red_, double green_, double blue_ );
  ColorRGB ( void );
  ColorRGB ( const Color & color_ );
  /* virtual */  ~ColorRGB ( void );

  void             red ( double red_ );
  double           red ( void ) const;

  void             green ( double green_ );
  double           green ( void ) const;

  void             blue ( double blue_ );
  double           blue ( void ) const;

  // Assignment operator from base class
  ColorRGB& operator= ( const Color& color_ );

protected:
  // Constructor to construct with PixelPacket*
  ColorRGB ( PixelPacket* rep_, PixelType pixelType_ );
};
```

3. Images

Image

- **Image** (n): appearance of a flat rectangle
- Photo, picture, painting, ...
- Corresponds to light entering our eye
- Different wavelengths of light create different colors

Image credit: CSUF Photos (flickr)

Human Visual Acuity

- Visual acuity: ability to tell two nearby shapes apart
- Limited
- Every person is different
- “Normal” acuity: *“...discriminate two contours separated by 1 arc minute – 1.75 mm at 6 meters”*
- Details smaller than this blend together

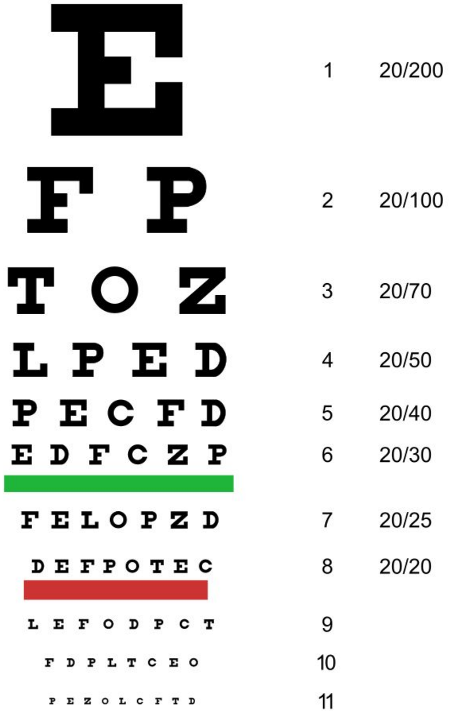


Image credit: Wikipedia

Raster Image

- Raster image: image represented by a rectangular grid of pixels
- **Pixel**
  - “picture element”
  - small rectangle of a single color

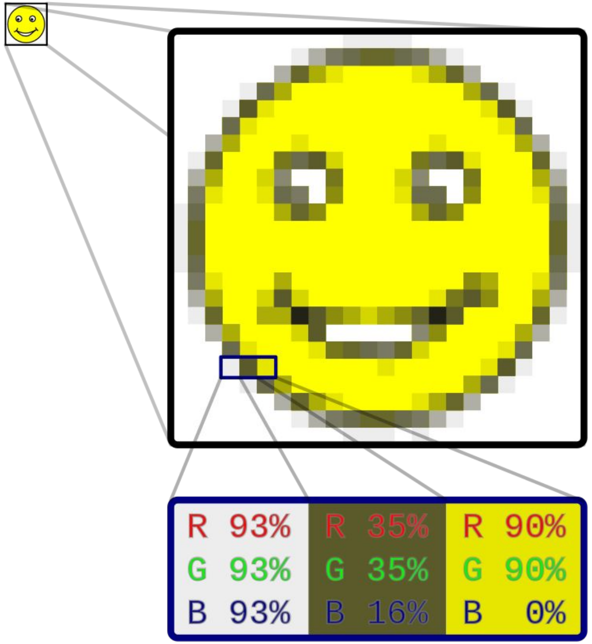


Image credit: Wikipedia

Resolution

- **Resolution**: dimensions of grid
  - width
  - height
- Low resolution: individual pixels are visible
  - blocky
- High resolution: pixels are smaller than human visual acuity
  - appears realistic
- Another “hack” based on human limitations

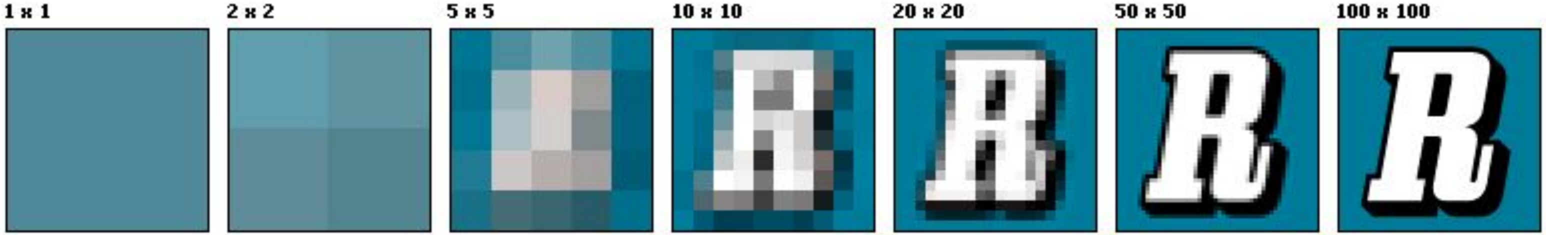
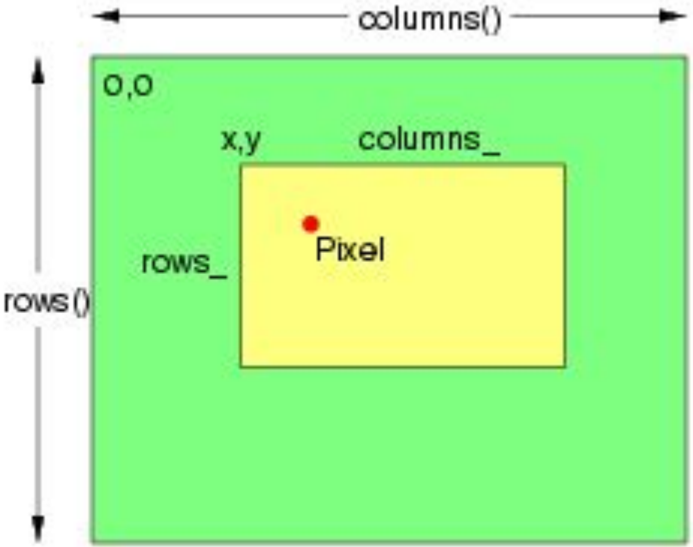


Image credit: Wikipedia

Image Coordinates

- Same convention as 2D vector
- Width = # columns
- Height = # rows
- (0, 0) is **top-left corner**
- x-coordinates increase toward right
- y-coordinates increase toward bottom



Procedural Image Generation

Procedural generation: program creates an image from scratch

Algorithm:

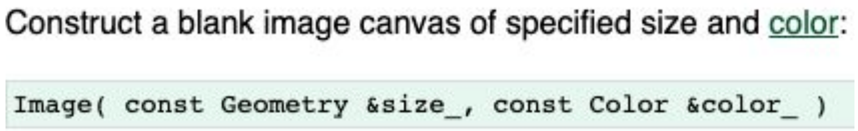
1. Create blank image (ex. all black pixels)
2. Loop for each pixel that needs to change:
    a. Compute RGB color for that pixel
    b. Change the image pixel to that color

GraphicsMagick Image

- Magick::Image API documentation
- Image object represents a raster image
- Can
  - Create
  - Read/write image file (JPEG, PNG, GIF)
  - Get pixel color
  - Set pixel color
  - Image processing operations (crop, flip, scale, etc.)

Constructing an Image

- Construct an Image API documentation

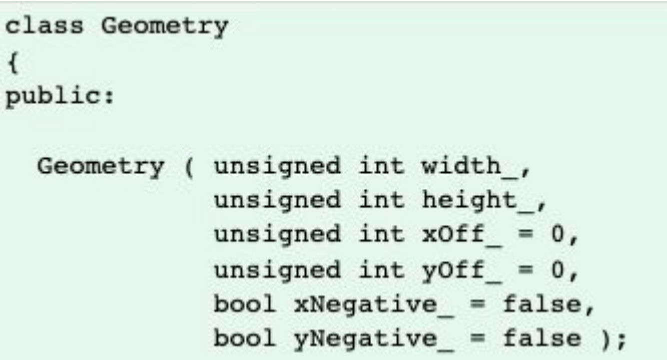


Construct a blank image canvas of specified size and color:

```
Image( const Geometry &size_, const Color &color_ )
```

GraphicsMagick Geometry

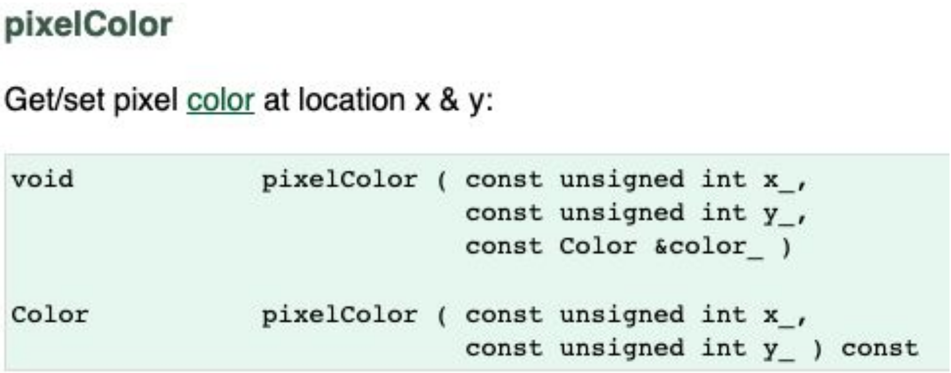
- Magick::Geometry API documentation
- Geometry object represents the **resolution** of an image
    - width
    - height



```
class Geometry
{
public:

  Geometry ( unsigned int width_,
             unsigned int height_,
             unsigned int xOff_ = 0,
             unsigned int yOff_ = 0,
             bool xNegative_ = false,
             bool yNegative_ = false );
```

Setting the Color of a Pixel

- GraphicsMagick::Image::pixelColor API documentation
- pixelColor is an accessor/mutator function



pixelColor

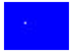
Get/set pixel color at location x & y:

```
void              pixelColor ( const unsigned int x_,
                               const unsigned int y_,
                               const Color &color_ )

Color             pixelColor ( const unsigned int x_,
                               const unsigned int y_ ) const
```

Example: Generating an Image

```
#include <Magick++.h>

int main(int argc, char* argv[]) {
 Magick::InitializeMagick(*argv);

 Magick::Geometry resolution{32, 24};
 Magick::ColorRGB blue{0.0, 0.0, 1.0};
 Magick::Image image{resolution, blue};

 Magick::ColorRGB white{1.0, 1.0, 1.0};
 image.pixelColor(10, 10, white);

 image.write("white_dot.png");

 return 0;
}
```

white_dot.png



4. Animation

Animation

- **Animation** (n): moving image
    - Movie, video
- Made up of many individual images

Image credit: Tim And Eric Awesome Show, Great Job!

Persistence of Vision and Frames

- Persistence of vision: optical illusion where a person continues to see an image even after it disappears
- **Frame:** one individual image in an animation
- **Frame rate:** frequency of changing frames
- Images persist for approximately 1/24 second
- Frame rate > 24 frames/second appears smooth
- Often use 30 or 60 frames/second to be safe
- Another “hack” based on human limitations

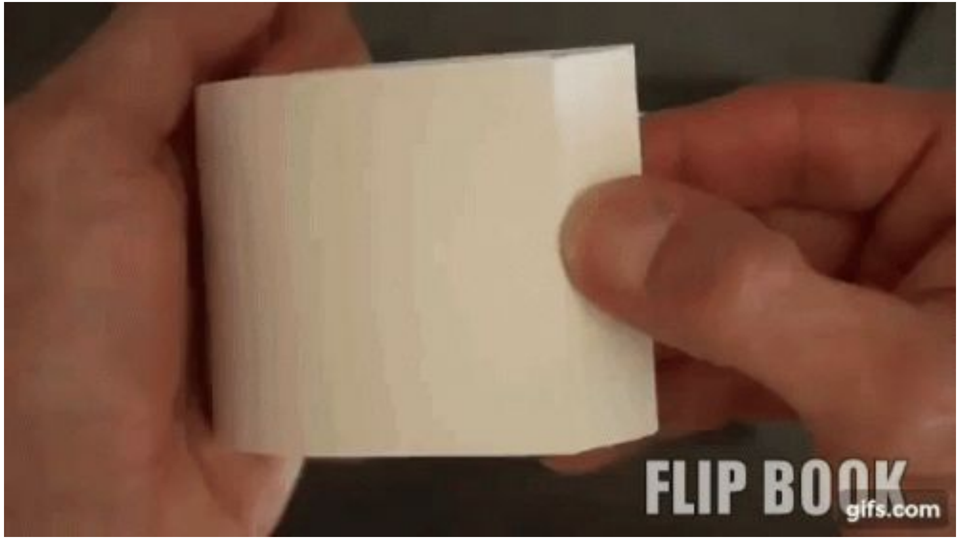


Image credit: gifs.com

GIF File

- **Graphics Interchange Format (GIF)**: legacy file format for images
- Controversial pronunciation
  - “g” as in “geometry” versus “gift”
- Limited to 256 colors
- Supports animation
  - Otherwise obsolete

GraphicsMagick Animation

- Magick::writeImages API documentation
- Function prototype:

```
void Magick::writeImages(
 InputIterator first,
 InputIterator last,
 const std::string& filename)
```

- So
  - Create a std::vector of Magick::Image objects
  - Build a Vector algorithm fills vector
  - `writeImages` to write GIF file

Example: Generating an Animation

```
#include <Magick++.h>

#include <vector>

int main(int argc, char* argv[]) {
 Magick::InitializeMagick(*argv);

 Magick::Geometry resolution{320, 240};
 Magick::ColorRGB blue{0.0, 0.0, 1.0};
 Magick::ColorRGB white{1.0, 1.0, 1.0};

 std::vector<Magick::Image> frames;
 for (int i = 0; i < 320; ++i) {
   Magick::Image frame{resolution, blue};
   for (int y = 0; y < 240; ++y) {
     frame.pixelColor(i, y, white);
   }
   frames.push_back(frame);
 }

 Magick::writeImages(frames.begin(), frames.end(), "wipe.gif");

 return 0;
}
```