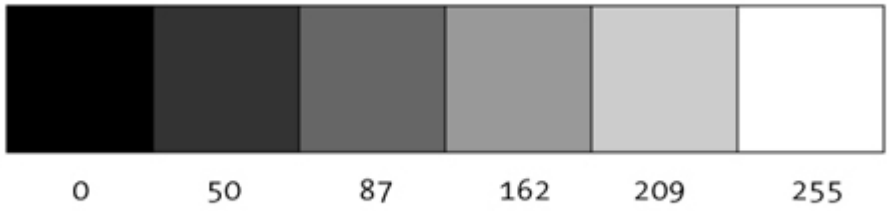
Grayscale Color

In the digital world, when we want to talk about a color, precision is required. Saying "Hey, can you make that circle bluish-green?" will not do. Color, rather, is defined as a range of numbers. Let's start with the simplest case: black & white or grayscale. 0 means black, 255 means white. In between, every other number - 50, 87, 162, 209, and so on - is a shade of gray ranging from black to white.



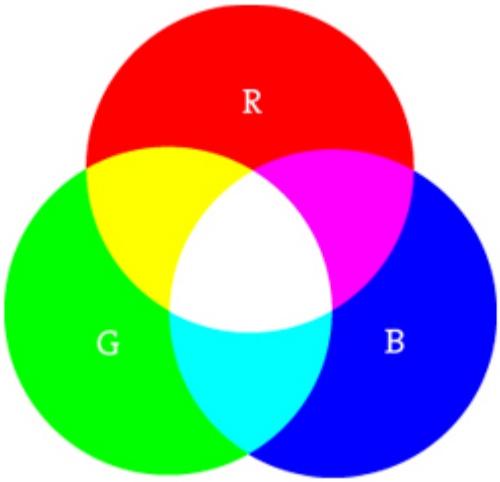
Does 0-255 seem arbitrary to you?
Color for a given shape needs to be stored in the computer's memory. This memory is just a long sequence of 0's and 1's (a whole bunch of on or off switches.) Each one of these switches is a bit, eight of them together is a byte. Imagine if we had eight bits (one byte) in sequence - how many ways can we configure these switches? The answer is (and doing a little research into binary numbers will prove this point) 256 possibilities, or a range of numbers between 0 and 255. We will use eight bit color for our grayscale range and 24 bit for full color (eight bits for each of the red, green, and blue color components).

By adding the **stroke (line color)** and **fill (interior color)** options when something is drawn, we can set the color of any given shape. There is also a **background (or clear color) option**, which sets a background color for a window before anything else is drawn.

RGB Color

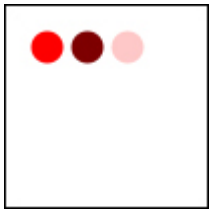
Remember finger painting? By mixing three "primary" colors, any color could be generated. Swirling all colors together resulted in a muddy brown. The more paint you added, the darker it got. Digital colors are also constructed by mixing three primary colors, but it works differently from paint. First, the primaries are different: red, green, and blue (i.e., "RGB" color). And with color on the screen, you are mixing light, not paint,

so the mixing rules are different as well.



- Red + Green = Yellow
- Red + Blue = Purple
- Green + Blue = Cyan (blue-green)
- Red + Green + Blue = White
- no colors = Black

This assumes that the colors are all as bright as possible, but of course, you have a range of color available, so some red plus some green plus some blue equals gray, and a bit of red plus a bit of blue equals dark purple. While this may take some getting used to, the more you program and experiment with RGB color, the more it will become instinctive, much like swirling colors with your fingers. And of course you can't say "Mix some red with a bit of blue," you have to provide an exact amount. As with grayscale, the individual color elements are expressed as ranges from 0 (none of that color) to 255 (as much as possible), and they are listed in the order R, G, and B. You will get the hang of RGB color mixing through experimentation, but next we will cover some code using some common colors.



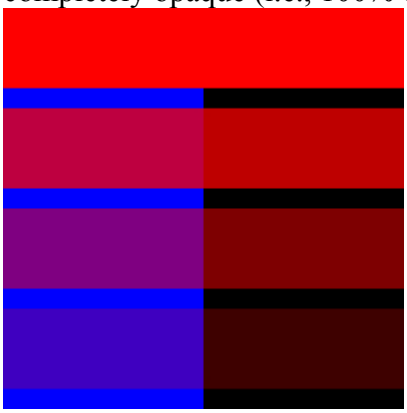
Example: RGB color

```
Background  255, 255, 255
No Stroke

// Bright red
Circle at 20,20 radius 16
Fill  255,0,0


// Dark red
Circle at 40,20 radius 16
Fill  127,0,0

// Pink (pale red)
Circle at 60,20 radius 16
Fill  255,200,200
```

In addition to the red, green, and blue components of each color, there is an additional optional fourth component, referred to as the color's "alpha." Alpha means transparency and is particularly useful when you want to draw elements that appear partially see-through on top of one another. The alpha values for an image are sometimes referred to collectively as the "alpha channel" of an image.

It is important to realize that pixels are not literally transparent, this is simply a convenient illusion that is accomplished by blending colors. Behind the scenes, the GPU takes the color numbers and adds a percentage of one to a percentage of another, creating the optical perception of blending. (If you are interested in programming "rose-colored" glasses, this is where you would begin.)

Alpha values also range from 0 to 255, with 0 being completely transparent (i.e., 0% opaque) and 255 completely opaque (i.e., 100% opaque).



Example: Alpha transparency

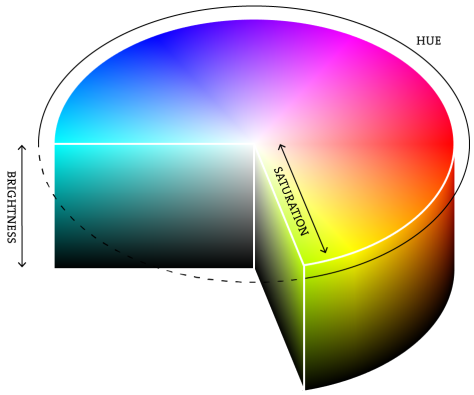
```
Window size 200,200
Background  0,0,0
No Stroke

// No fourth argument means 100% opacity.
Rectangle at 0,0 dimensions 100,200
Fill 0,0,255


// 255 means 100% opacity.
Rectangle at 0,0 dimensions 200,40
Fill 255,0,0,255


// 75% opacity.
Rectangle at 0,50 dimensions 200,40
Fill 255,0,0,191


// 55% opacity.
Rectangle at 0,100 dimensions 200,40
Fill 255,0,0,127


// 25% opacity.
Rectangle at 0,150 dimensions 200,40
Fill 255,0,0,63
```

Other Color Models

RGB color with ranges of 0 to 255 is not the only way you can handle color. In fact, that convention is undesirable because it ties the programming interface to the hardware's use of an 8-bit RGBA representation. If the hardware improves in the future, hundreds of books and thousands of programs will need to be updated.

A better convention is to use real numbers, which can be increased in precision by using smaller fractions. As before, there are still RGBA components but each is represented by a real number between 0.0 and 1.0. 0.0,0.0,0.0 is black and 1.0,1.0,1.0 is white.

Finally, while you will likely only need RGB color for all of your programming needs, you can also specify colors in the HSB (hue, saturation, and brightness) mode. Without getting into too much detail, HSB color works as follows:



- **Hue** - The color type, ranges from 0 to 255 by default.
- **Saturation** - The vibrancy of the color, 0 to 255 by default.
- **Brightness** – The brightness of the color, 0 to 255 by default.