The Imaginary Institute
www.imaginary-institute.com
2Dcourse@imaginary-institute.com

2D ANIMATION & INTERACTION COURSE
WEEK 1 QUICK REFERENCE

COLORS

To choose a color-naming mode:

`colorMode(RGB);`	Interpret colors as red, green, and blue
`colorMode(HSB);`	Interpret colors as hue, saturation, and brightness

To specify a color:

`#`	A shade of gray from 0 (black) to 255 (white)
`#,#,#`	Three values from 0 (off) to 255 (fully on) for RGB or HSB
`#,#,#,#`	RGB or HSB, with a fourth value for opacity

Functions related to color:

`background(color);`	clear entire window to this color
`fill(color);`	fill shapes with this color
`noFill();`	don't fill shapes with any color
`stroke(color);`	draw a stroke of this color around shapes
`noStroke();`	don't draw a stroke around shapes
`strokeWeight(#);`	drawn strokes are this many pixels thick

General notes about colors:

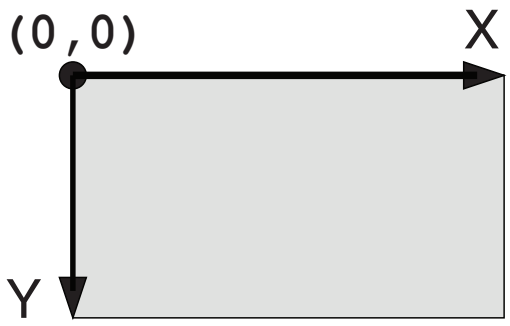
1. Color values are all from 0 (fully off) to 255 (fully on).
2. If you provide just one number for a color, it will be interpreted as a shade of gray.
3. If you provide three numbers, they are either RGB or HSB depending on the current mode.
4. If you provide four numbers, the fourth number is opacity: 0=transparent, 255=opaque.
5. Fill and stroke settings apply to all following shapes until you change one or more settings, and then the new settings apply from then on. Shapes you've already drawn don't change.
6. The default color mode is RGB. If you switch to HSB, once you're

finished specifying your colors, I recommend immediately switching back to RGB.

7. The defaults are a neutral gray background, white fill, and a 1-pixel thick black stroke

COORDINATES

The coordinate system starts in the upper left. This is the point (x,y)=(0,0), called the *origin*. Increasing values of X specify points to the right of the origin. Increasing values of Y specify points below the origin.

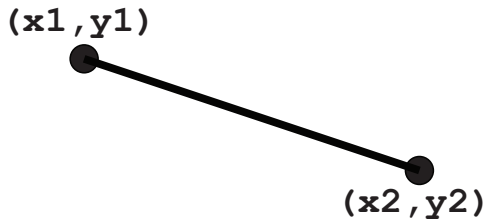


SHAPES

Lines

`line(x1, y1, x2, y2);`

Draw a line from `(x1, y1)` to `(x2, y2).`

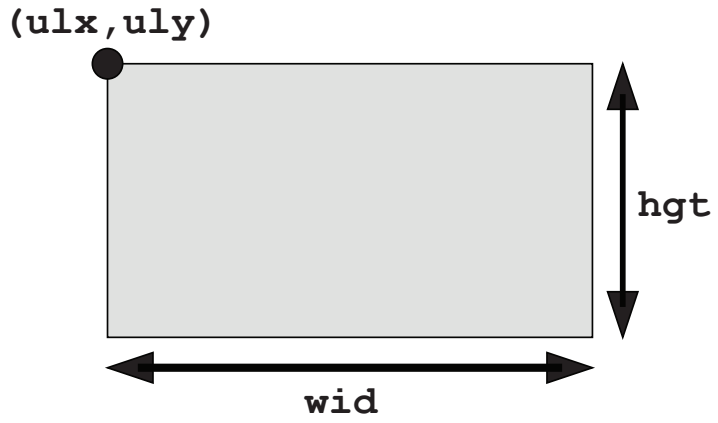


Rectangles

`Rect(ulx, uly, wid, hgt);`

Draw a rectangle with its upper-left corner at `(ulx, uly)` and extending right by `wid` and down by `hgt` If `wid` and `hgt` are equal, you'll get a

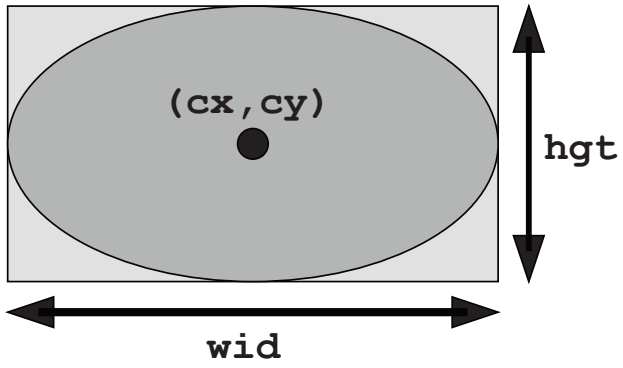
square.



Ellipses

`Ellipse(cx, cy, wid, hgt);`

Draw an ellipse. The center is at `(cx, cy)`. The total width and height of the box that just surrounds the ellipse are given by `wid` and `hgt`. This surrounding box is only a concept for thinking about the ellipse, and is not drawn. If `wid` and `hgt` are equal, you'll get a circle.



A SKELETON PROGRAM

Here's an outline (or skeleton) program that you can use to get started. Choose the size of your graphics window in the call to `size()`, and then call `smooth()` to avoid any jaggy edges. Your window will start out filled with a neutral gray, so you'll typically want to start by filling it with your background color. Then call the routines above to create your picture.

Remember that there's an opening curly brace at the start of the first line, and a closing curly brace at the end of the last line. Also remember to end every line between these curly braces with a semicolon.

```
void setup() {     // how programs start off
  Size(600, 400);  // graphics window size
                   // (keep between 300 and 1000)
  smooth();        // draw smooth edges
  background(100, 150, 200); // clear to a
                             // medium blue
  /* Create your composition here by choosing
  colors and drawing shapes.  To see your
  results, press the Run button (the right-
  pointing triangle in a circle at the top
  of the Processing window).
  To make changes, stop the program: either
  click the close box at the top of
  the graphics window, or press the Stop
  button (the square in a  circle) in the
  Processing window.  Make your changes to
  the program text and run it again.
  Lather, rinse, repeat!
  */
}               // the necessary closing brace
```