Processing for Visualization

Processing

Welcome to Processing 3! Dan explains the new features and changes; the links Dan mentions are on the Vimeo page.

» Download Processing
» Browse Tutorials
» Visit the Reference

Processing is a flexible software sketchbook and a language for learning how to code within the context of the visual arts. Since 2001, Processing has promoted software literacy within the visual arts and visual literacy within technology. There are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning and prototyping.

» Free to download and open source

» Interactive programs with 2D, 3D or PDF output

» OpenGL integration for accelerated 2D and 3D

» For GNU/Linux, Mac OS X, and Windows

» Over 100 libraries extend the core software

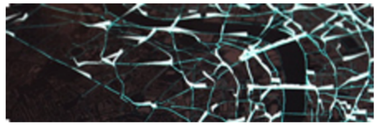
» Well documented, with many books available

» Exhibition

Behavioural Complexity
by AADRL

Terrapattern
by Golan Levin, David Newbury, Kyle McDonald, Irene Alvarado, Aman Tiwari, Manzil Zaheer and The Frank-Ratchye STUDIO for Creative Inquiry

Fluid Leaves
by Reinoud van Laar



cf.city flows
by Till Nagel and Christopher Pietsch

Processing

Reference. Processing was designed to be a flexible software sketchbook.

Structure

() (parentheses)
, (comma)
. (dot)
/* */ (multiline comment)
/** */ (doc comment)
// (comment)
; (semicolon)
= (assign)
[] (array access)
{} (curly braces)
catch
class
draw()
exit()
extends
false
final
implements
import
loop()
new
noLoop()
null
popStyle()
private
public
pushStyle()
redraw()
return
setup()
static
super
this

Shape

createShape()
loadShape()
PShape

2D Primitives
arc()
ellipse()
line()
point()
quad()
rect()
triangle()

Curves
bezier()
bezierDetail()
bezierPoint()
bezierTangent()
curve()
curveDetail()
curvePoint()
curveTangent()
curveTightness()

3D Primitives
box()
sphere()
sphereDetail()

Attributes
ellipseMode()
rectMode()
strokeCap()

Color

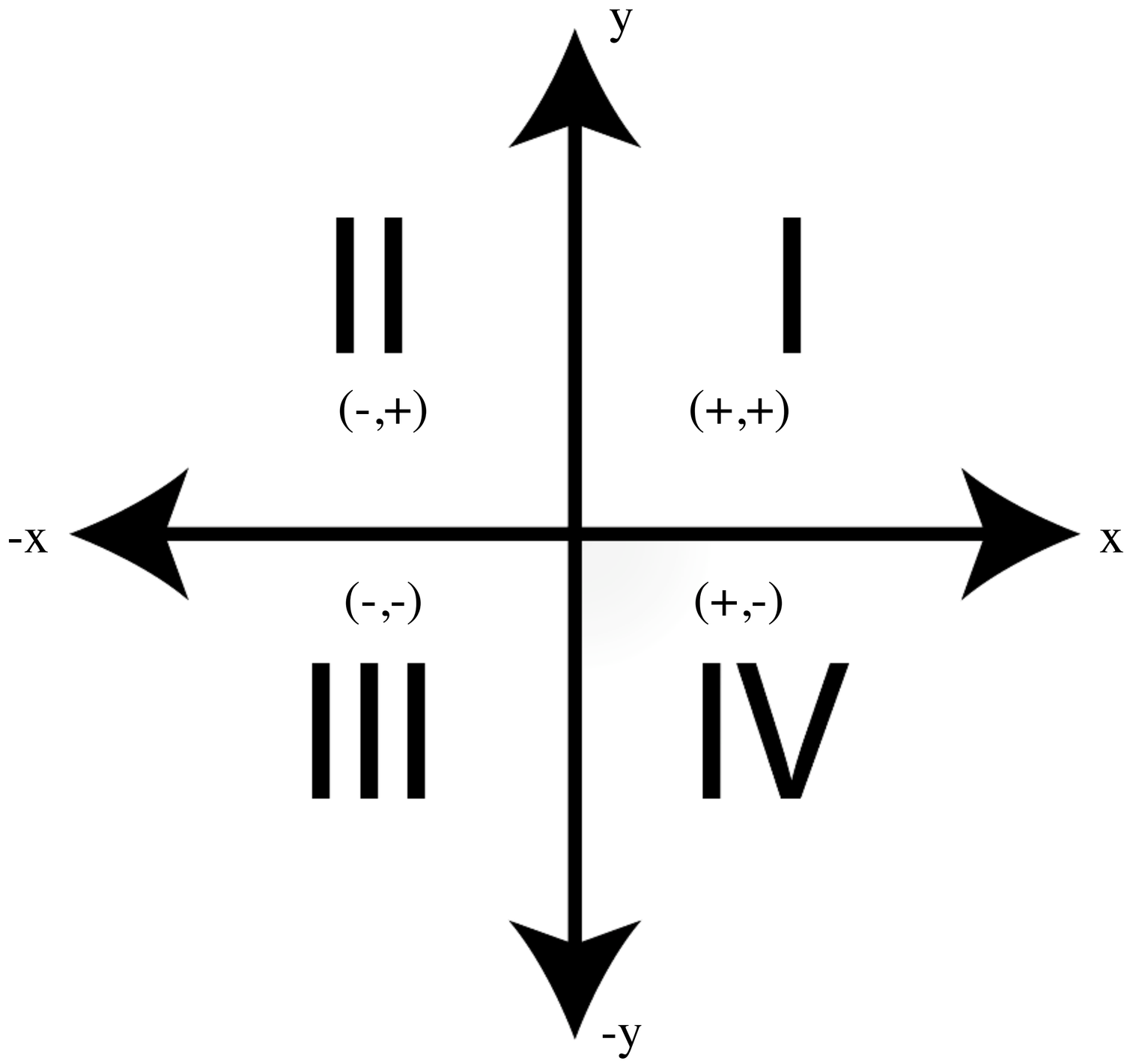
Setting
background()
clear()
colorMode()
fill()
noFill()
noStroke()
stroke()

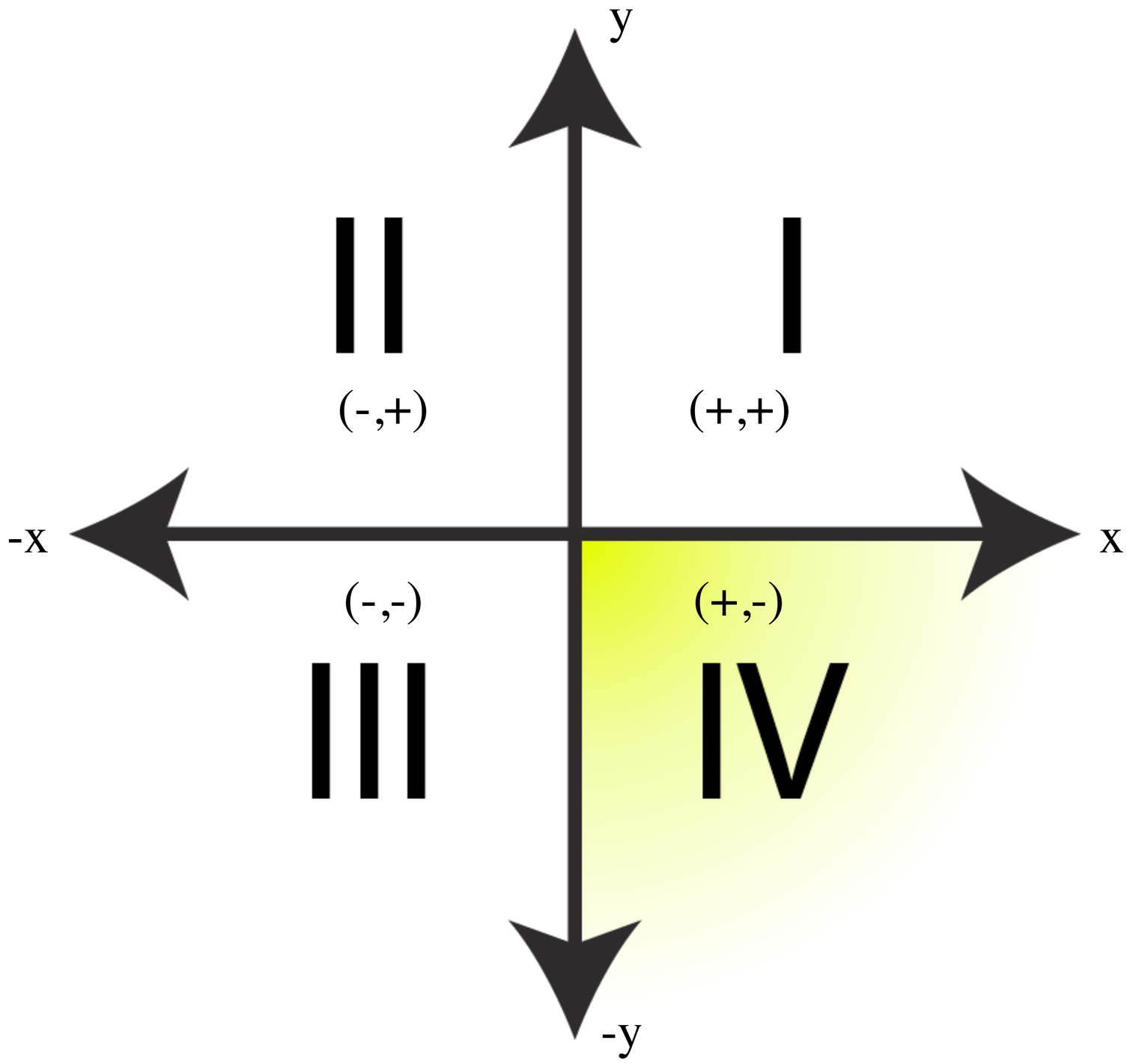
Creating & Reading
alpha()
blue()
brightness()
color()
green()
hue()
lerpColor()
red()
saturation()

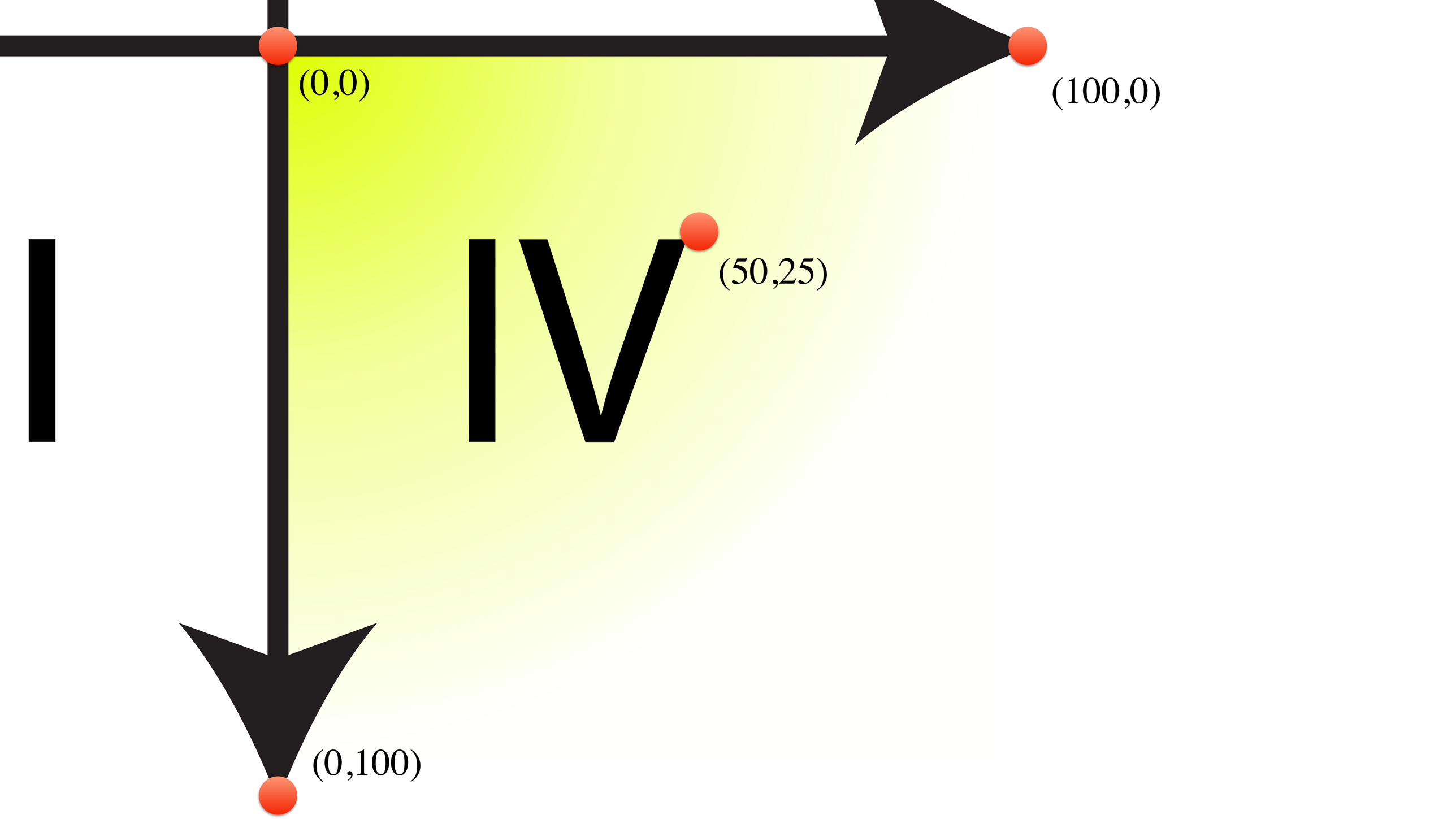
Image

createImage()
PImage

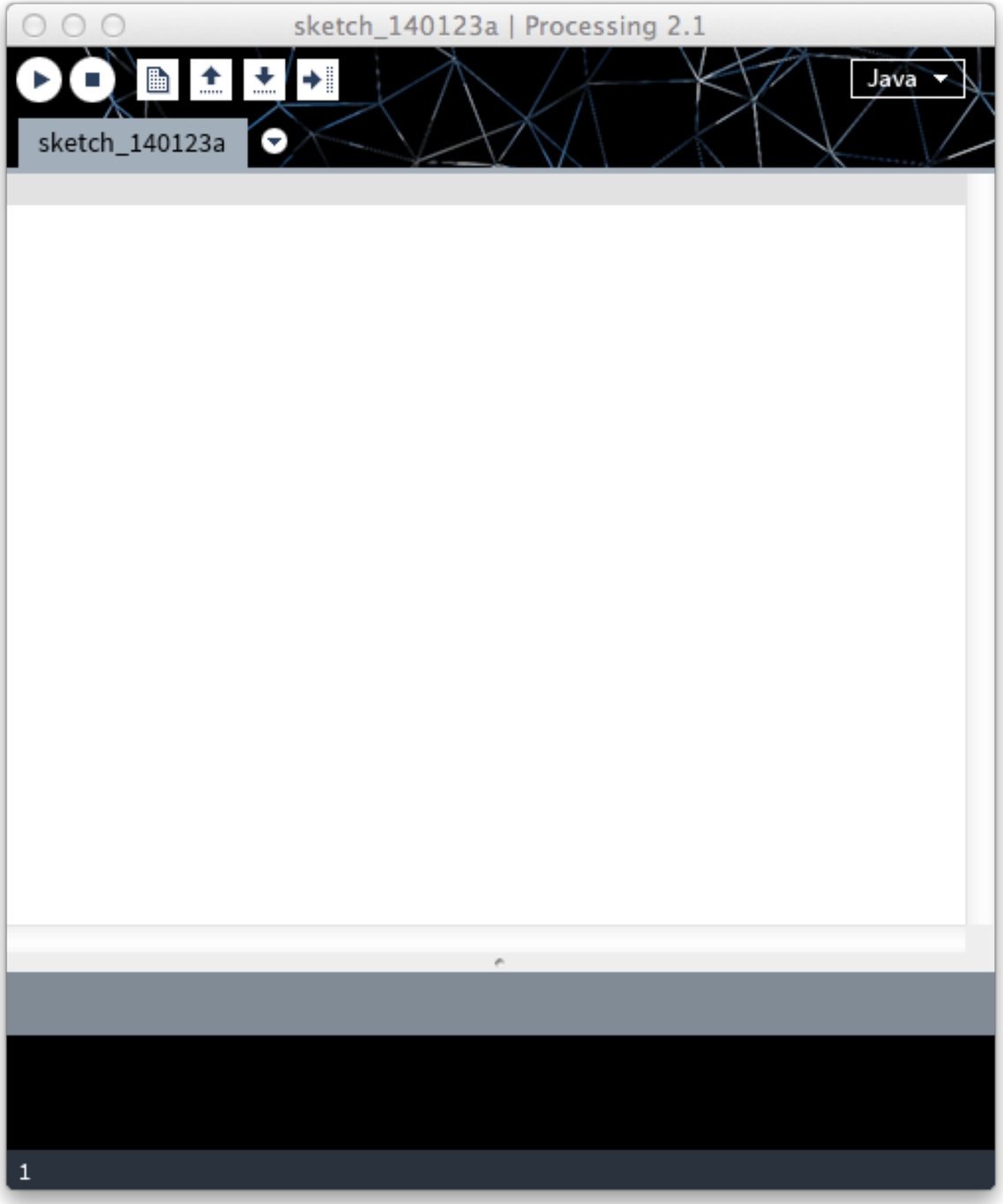
Loading & Displaying
image()
imageMode()
loadImage()
noTint()
requestImage()
tint()


y
II
I
(-,+)
(+,+)
-x
x
(-,-)
(+,-)
III
IV
-y


y
II
I
(-,+)
(+,+)
-x
x
(-,-)
(+,-)
III
IV
-y


(0,0)
(100,0)
I
IV
(50,25)
(0,100)

File > New



A Basic Processing “Sketch”

```
// your initialization
void setup() {
   size(500, 500);
   //…
}

// your drawing code, called repeatedly by Processing’s
internal rendering loop
void draw() {
   background(0,0,0);
   rect(100,100,300,300);
   //…
}
```

Size must be the first line in setup()

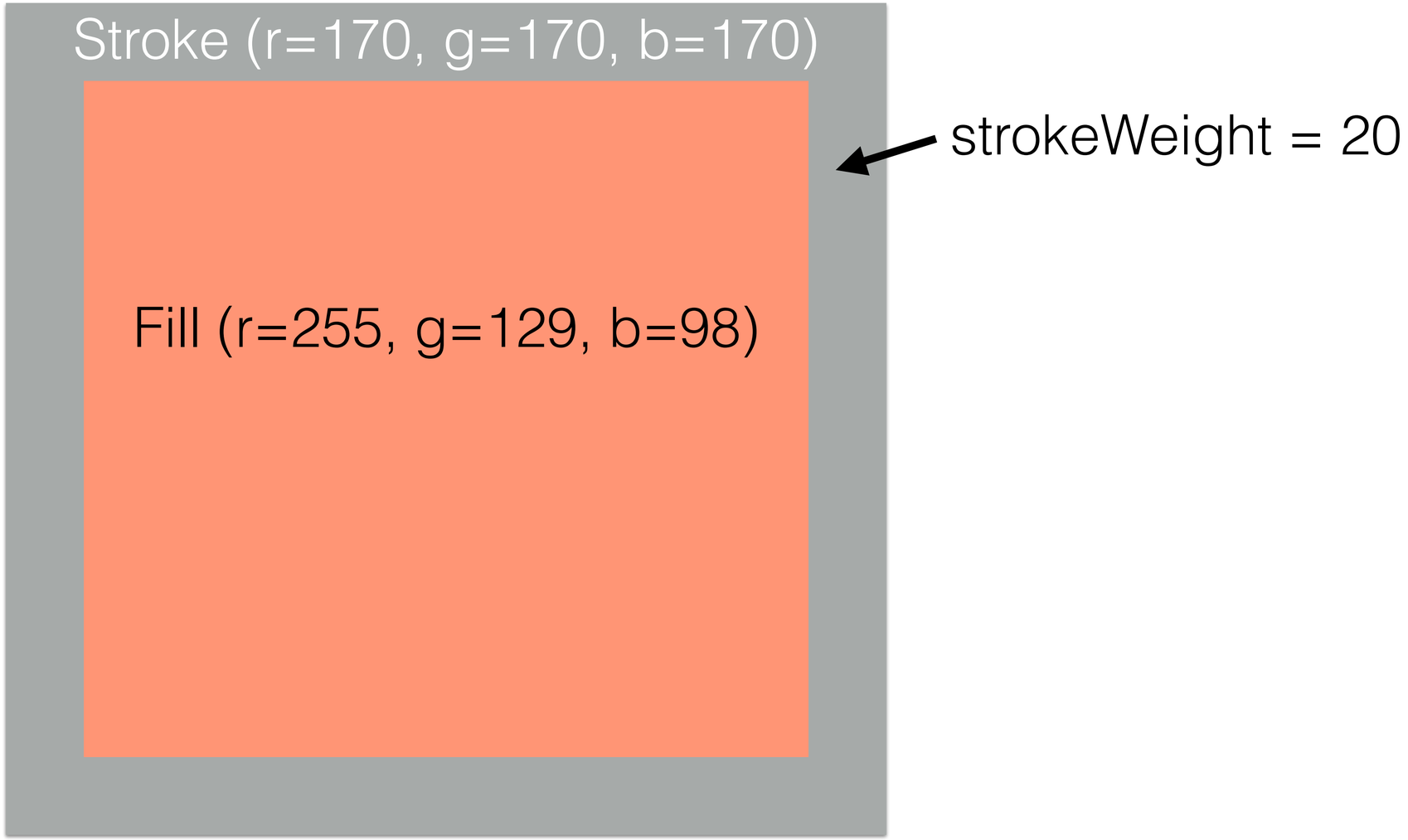
- `size(w,h); //sets window size`

Primitive drawing functions

- `point(x,y);`
- `rect(a, b, c, d);`
- `ellipse(..);`
- `circle(..);`
- `triangle(..);`
- `etc..`
- The Processing Reference is great! Just keep it open as you code.
  - https://processing.org/reference/

Setting the state of the “pen” and “brush”

- `stroke(color);   or noStroke();`
- `strokeWeight(size);`
- `fill(color);     or noFill();`
- Color is specified with red, green, blue, and alpha components. Look at the processing color tutorial:
  - https://processing.org/tutorials/color/
- These calls set the state of the pen that outlines the shape and brush that fills in the shape you draw. They affect all of the primitive calls that follow until you change the state again.

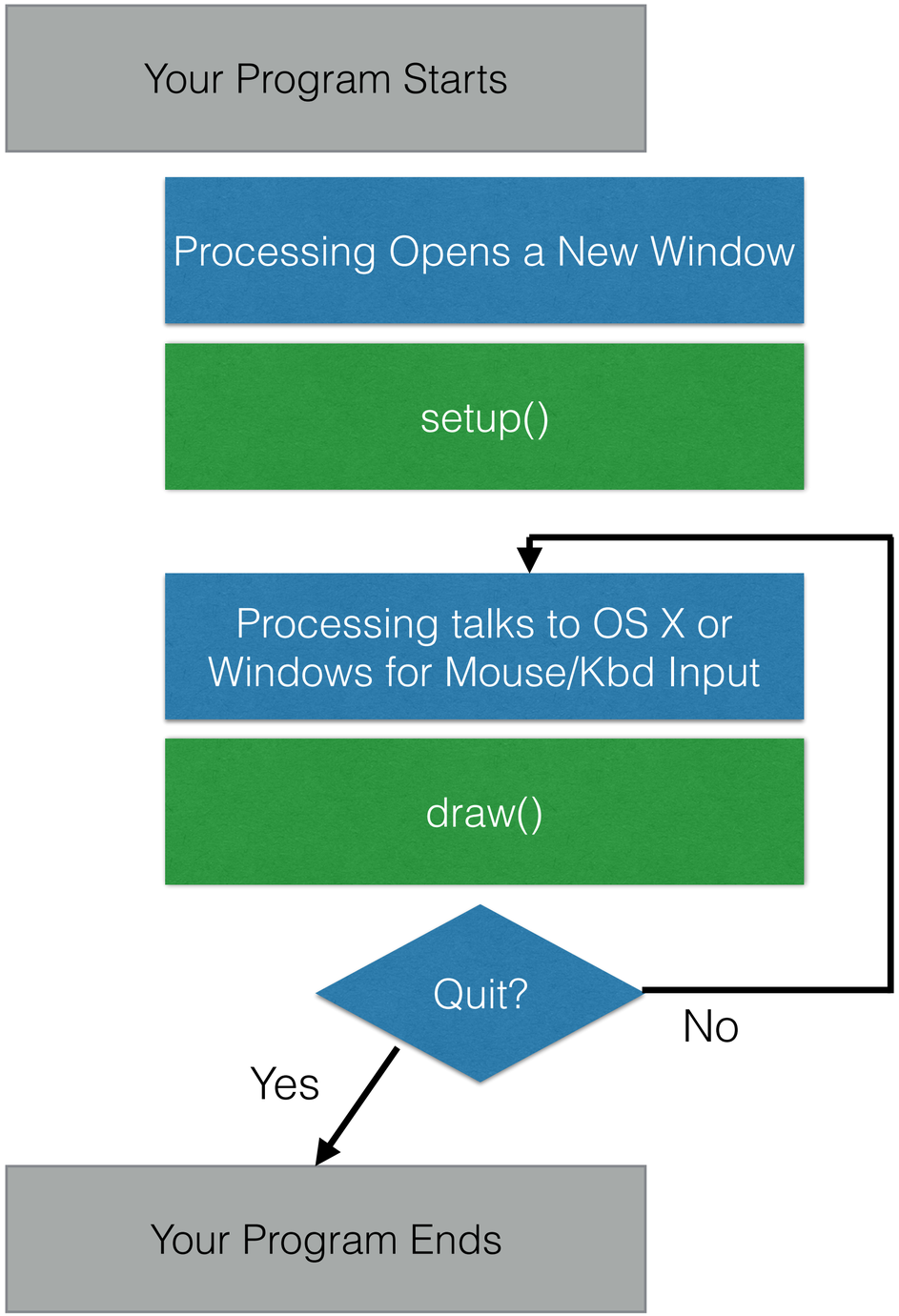

Stroke (r=170, g=170, b=170)
Fill (r=255, g=129, b=98)
strokeWeight = 20

Why repeated calls to draw()?

```
// your initialization
void setup() {
    size(500, 500);
    //…
}

// your drawing code, called repeatedly
void draw() {
  background(0,0,0);
  rect(100,100,300,300);
    //…
}
```

- Animation
- Interactivity - respond to user input



The Parts of the Program You Write

The Parts Processing Does For You

This loop repeats at about 60 times per second!

Working with Text

1. Create a font data file using the Processing Editor

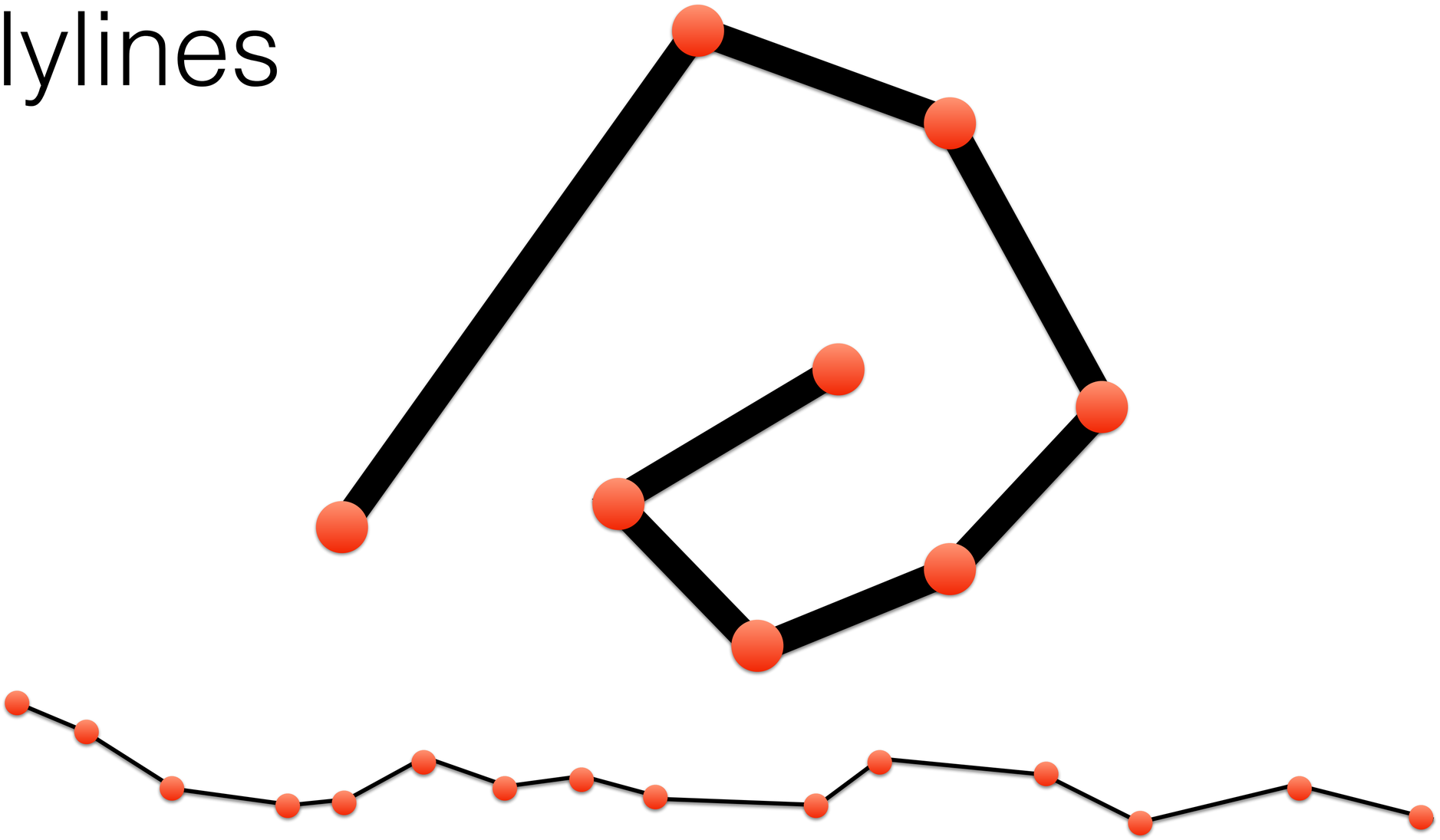
2. Load the font data file in setup():

```
textFont(loadFont("AmericanTypewriter-20.vlw"));
```

3. Render text to the screen in draw():

```
text(string, x,y);
```

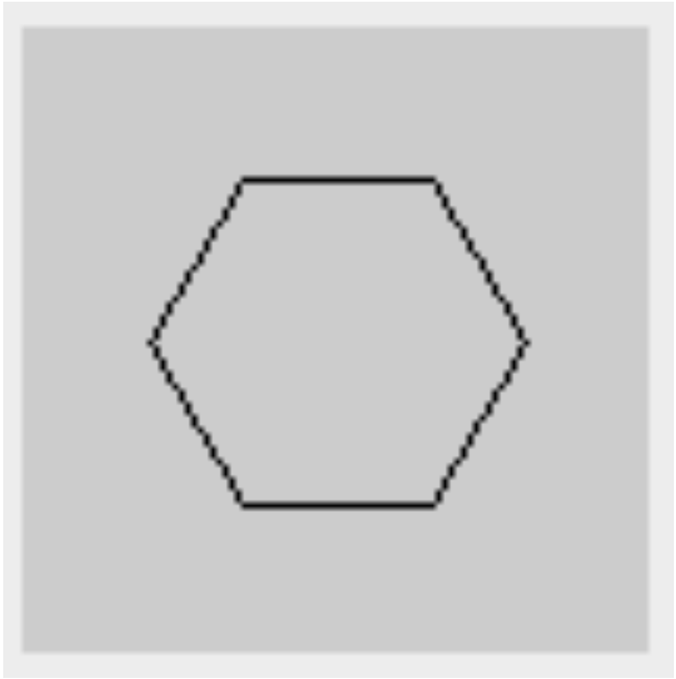
Polylines



Polylines

```
noFill();

beginShape();
for (int x=0; x<=6; x++) {
  vertex(50 + cos(x*PI/3)*30,
         50 + sin(x*PI/3)*30);
}
endShape();
```



Triangles