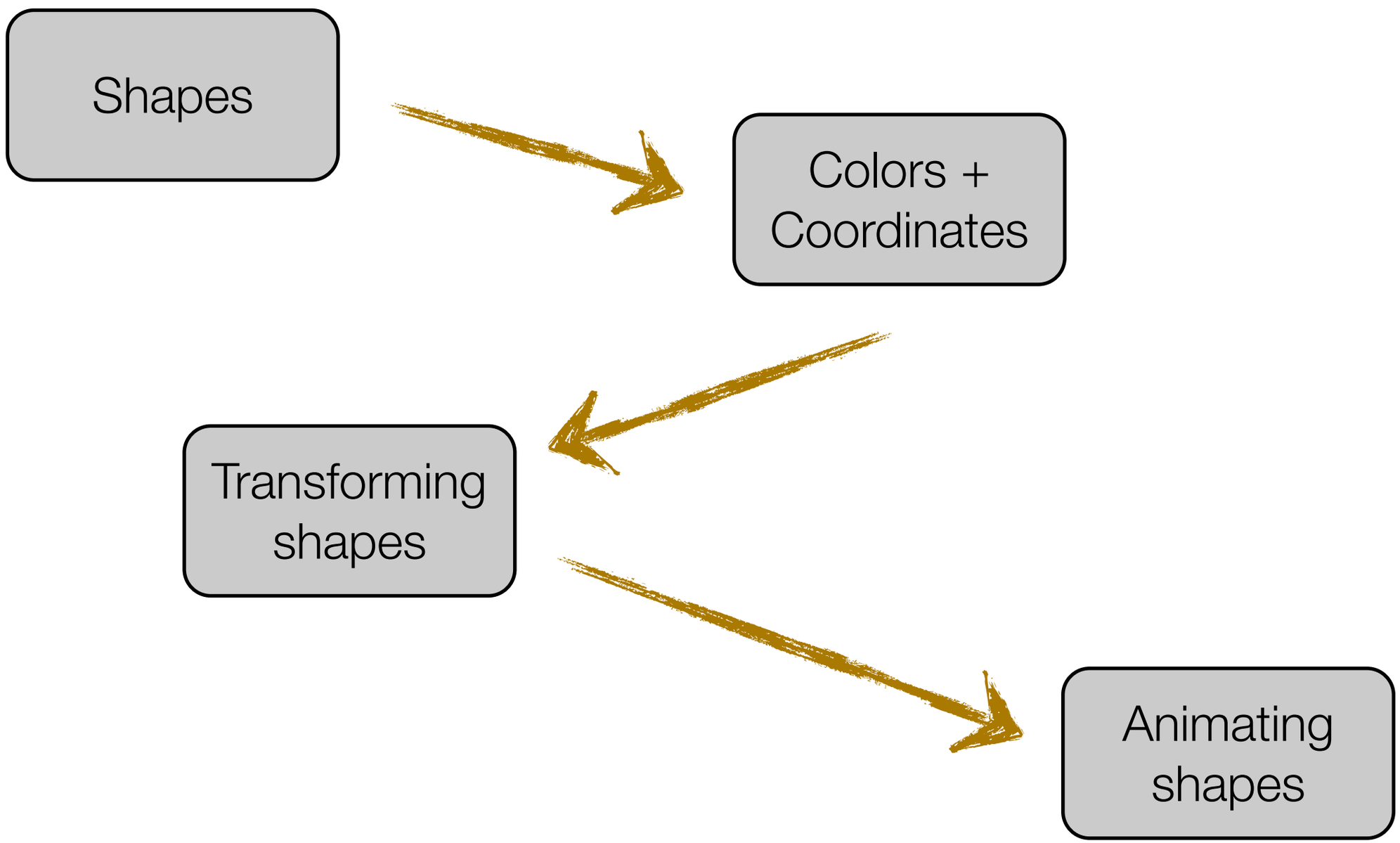
# Creative Programming – workshop 02

Mathias Funk (ID DI), Fall 2012

# What we do today



Processing

Search

Cover \ Exhibition \ Reference \ Learning \ Download \ Shop \ About

»Feed »Forum »Wiki »Code

↳ Language (A-Z) \ Libraries \ Tools \ Environment

Language (API). The Processing Language has been designed to facilitate the creation of sophisticated visual and conceptual structures.

Standard Processing | JavaScript (Processing.js)

**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
true
try
void

**Environment**

cursor()
displayHeight
displayWidth
focused
frameCount
frameRate()
frameRate
height
noCursor()
size()
width

**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()
noSmooth()
rectMode()
smooth()
strokeCap()
strokeJoin()
strokeWeight()

*Vertex*

beginShape()
bezierVertex()
curveVertex()
endShape()
quadraticVertex()
vertex()

*Loading & Displaying*

shape()
shapeMode()

**Color**

*Setting*

background()
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()

*Textures*

texture()
textureMode()
textureWrap()

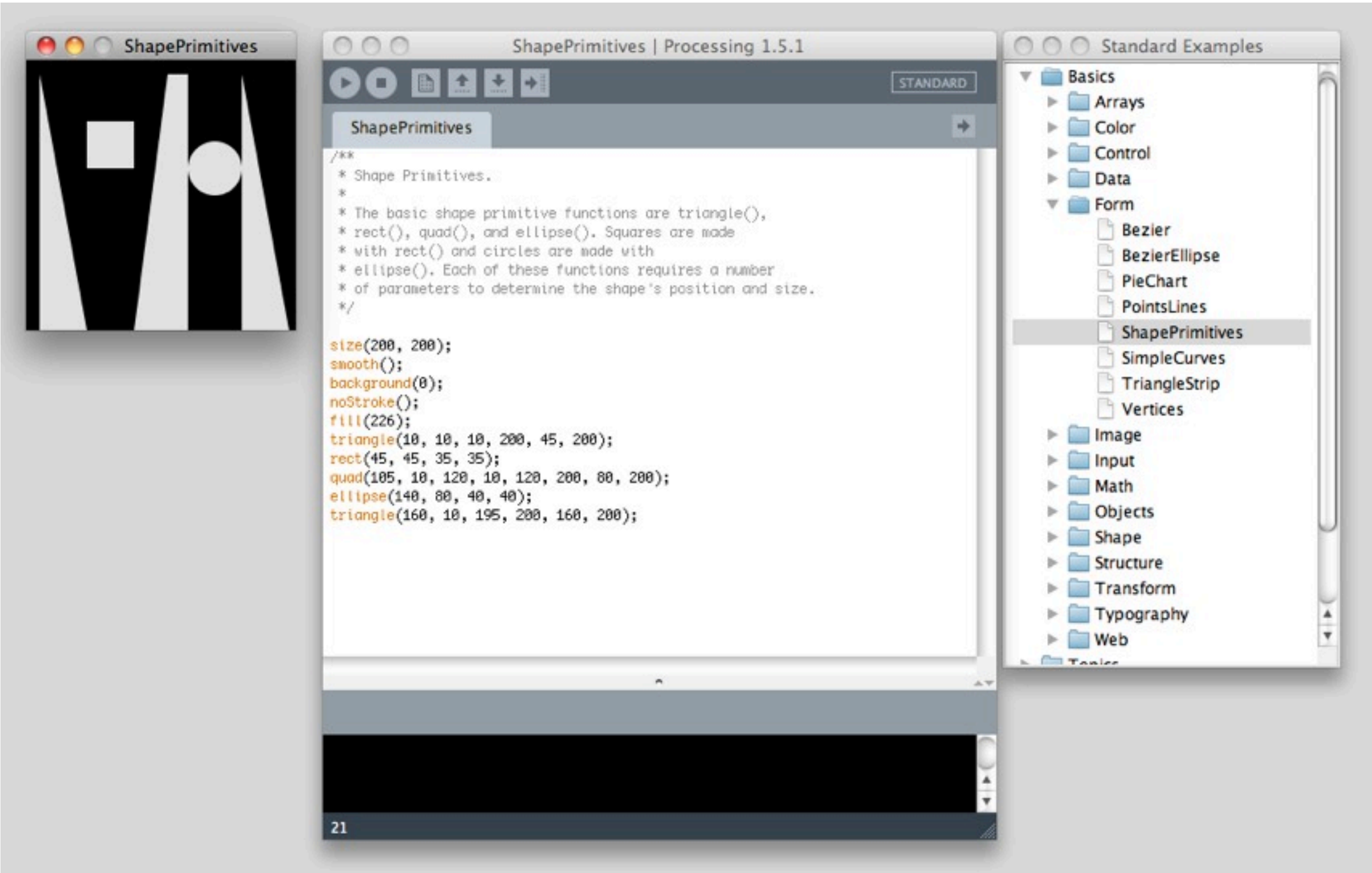
*Pixels*

blend()
copy()
filter()
get()
loadPixels()
pixels[]
set()
updatePixels()

**Rendering**

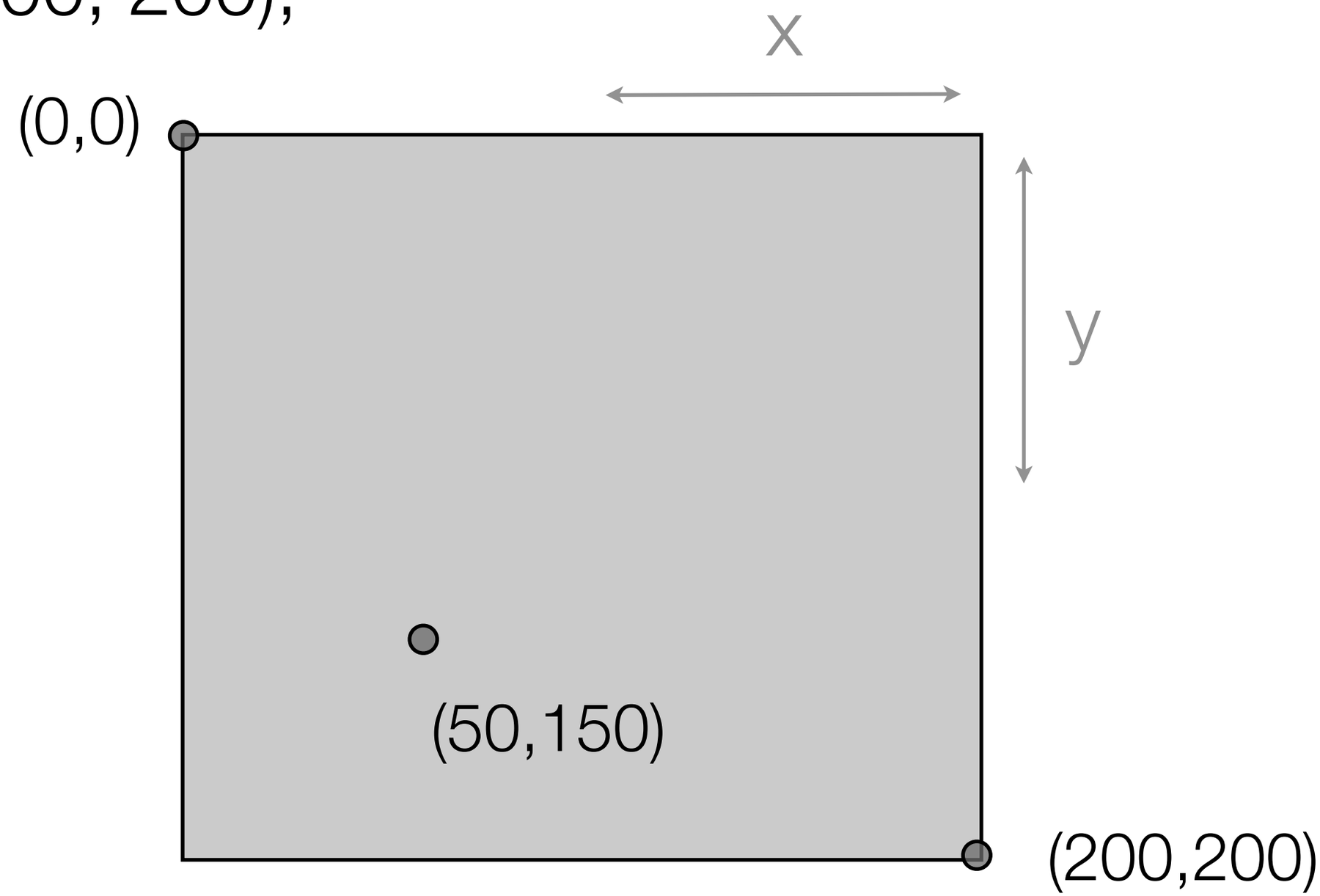
# Shapes

# First running sketch



# Coordinate system

size(200, 200);



# Shapes

- Triangle

```
triangle(x1, y1, x2, y2, x3, y3);
```

- Quad

```
quad(x1, y1, x2, y2, x3, y3, x4, y4);
```

- Rectangle

```
rect(x, y, width, height);
```

- Ellipse

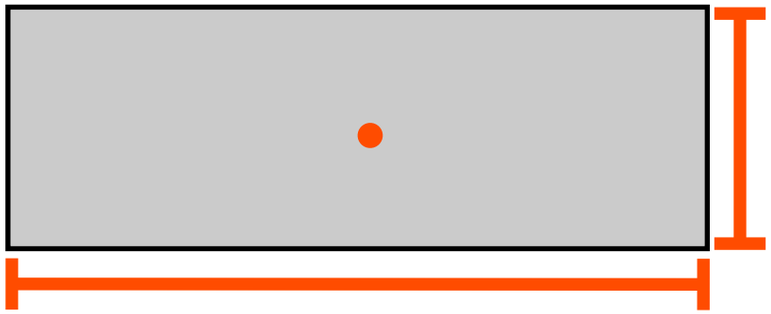
```
ellipse(x, y, width, height);
```

# Rectangle Drawing Mode

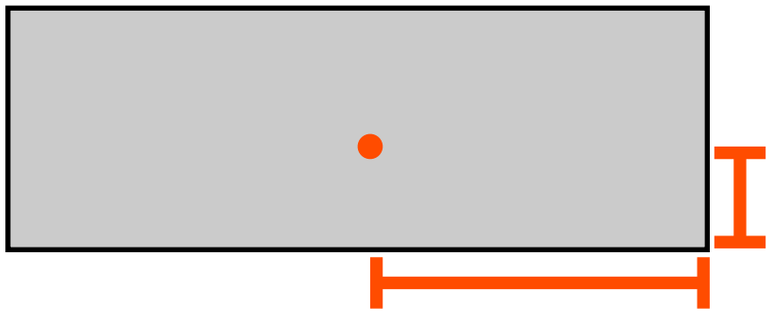
- rectMode(CORNER);

- rectMode(CORNERS);

- rectMode(CENTER);

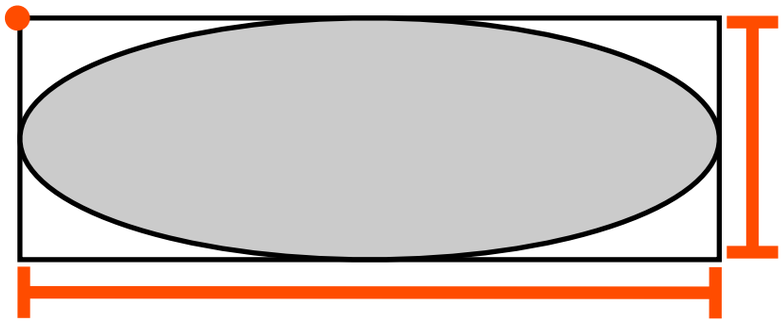


- rectMode(RADIUS);

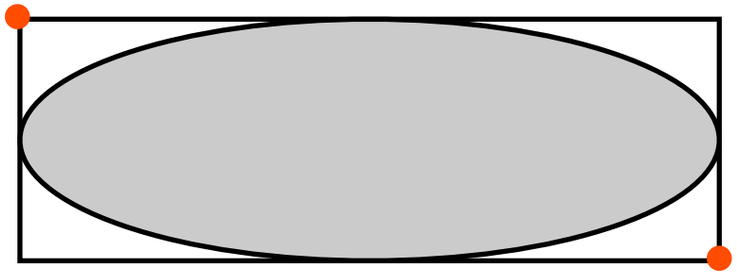


# Ellipse Drawing Mode

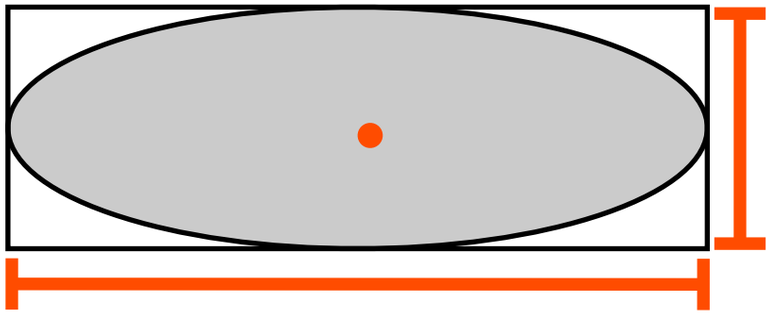
- ellipseMode(CORNER);



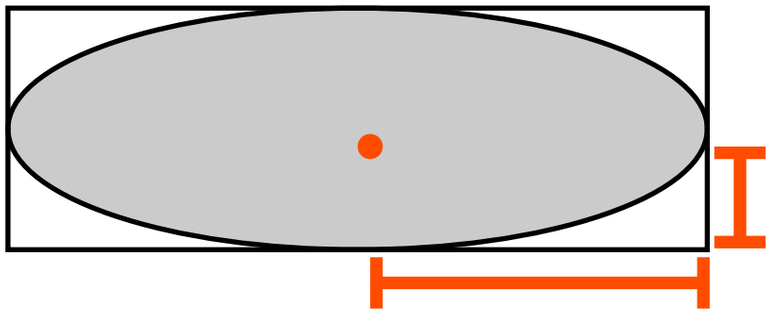
- ellipseMode(CORNERS);



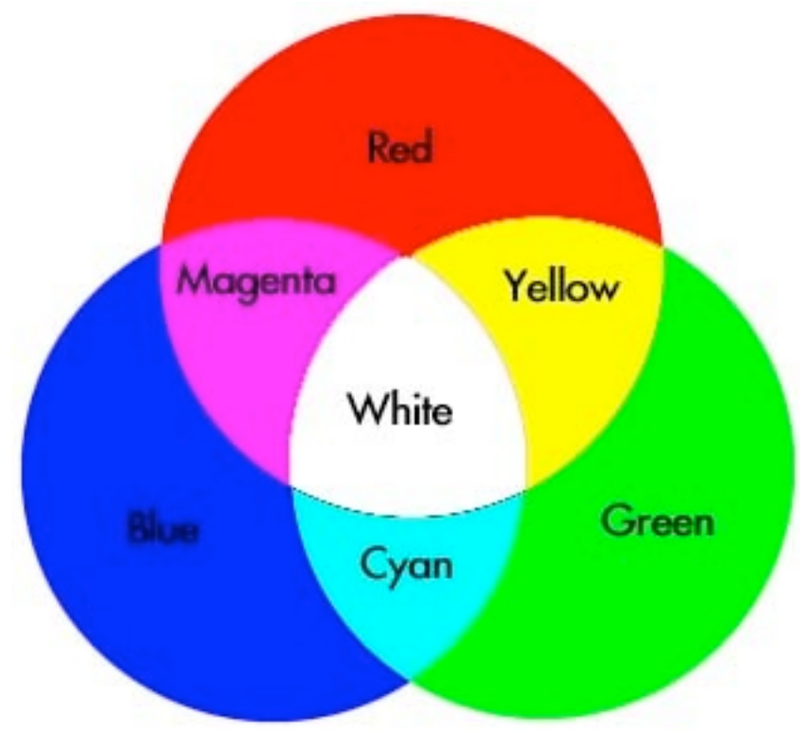
- ellipseMode(CENTER);



- ellipseMode(RADIUS);



# Colors



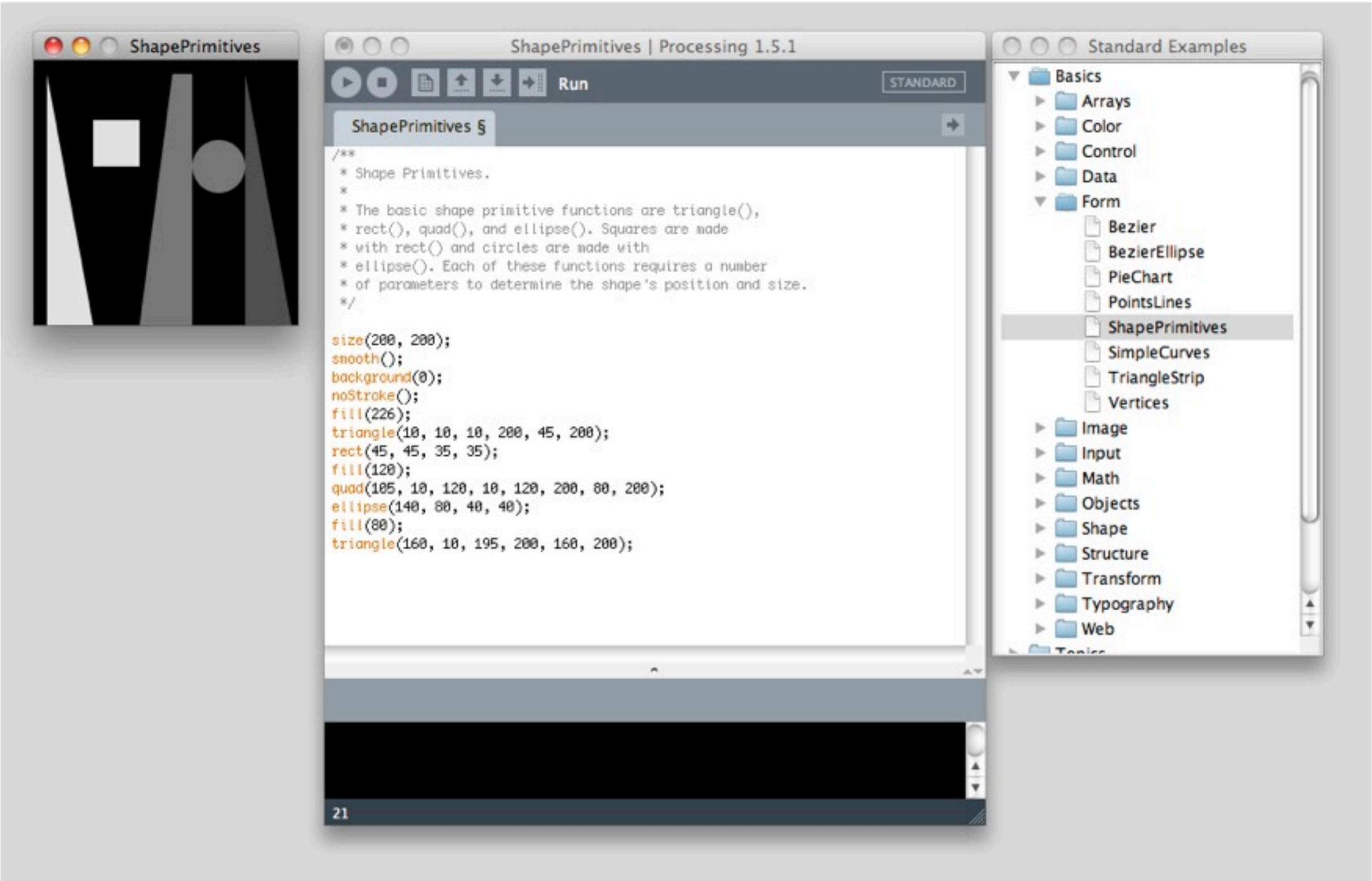
# How colors work in Processing

- Color rendering in Processing works in the additive color model: RGB
- fill (<RED>, <GREEN>, <BLUE>); // all values from 0 - 255 possible

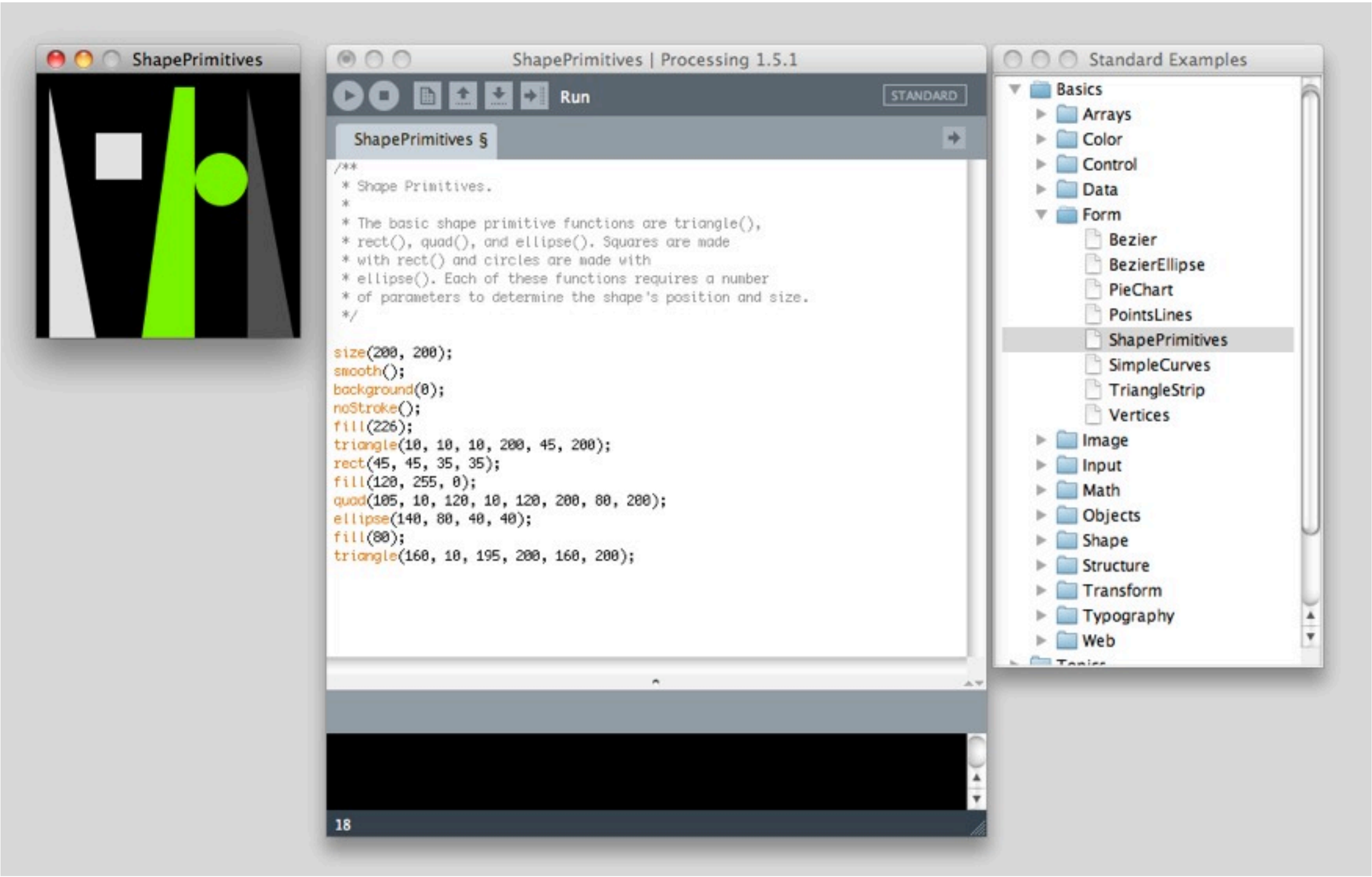
- fill(255, 0, 0); // pure red
- fill(0, 0, 130); // dark blue
- How to get yellow?

- When all values are same you will get grayscale colors (or white or black).
- “fill(120)” is a shortcut for “fill(120, 120, 120)”

# Colors...


ShapePrimitives
ShapePrimitives | Processing 1.5.1
Run
STANDARD
ShapePrimitives §
/**
* Shape Primitives.
*
* The basic shape primitive functions are triangle(),
* rect(), quad(), and ellipse(). Squares are made
* with rect() and circles are made with
* ellipse(). Each of these functions requires a number
* of parameters to determine the shape's position and size.
*/
size(200, 200);
smooth();
background(0);
noStroke();
fill(226);
triangle(10, 10, 10, 200, 45, 200);
rect(45, 45, 35, 35);
fill(120);
quad(105, 10, 120, 10, 120, 200, 80, 200);
ellipse(140, 80, 40, 40);
fill(80);
triangle(160, 10, 195, 200, 160, 200);
21
Standard Examples
Basics
Arrays
Color
Control
Data
Form
Bezier
BezierEllipse
PieChart
PointsLines
ShapePrimitives
SimpleCurves
TriangleStrip
Vertices
Image
Input
Math
Objects
Shape
Structure
Transform
Typography
Web

# Colors, really



# Outline aka Border aka Stroke

```
noStroke();
```

```
stroke(0,255,0);
```

# Smoothing aka Anti-Aliasing

`smooth();` `noSmooth();`

# Transforming shapes

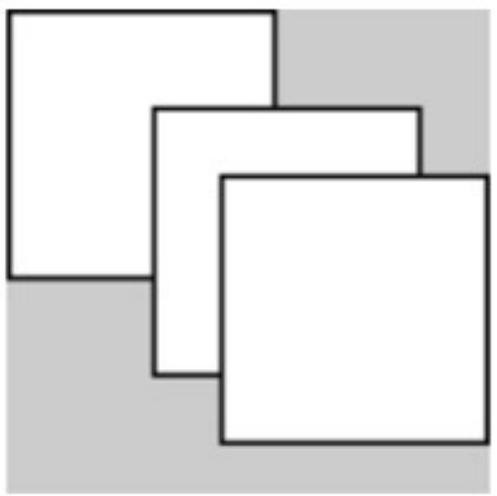
# Transformations?

translate

scale

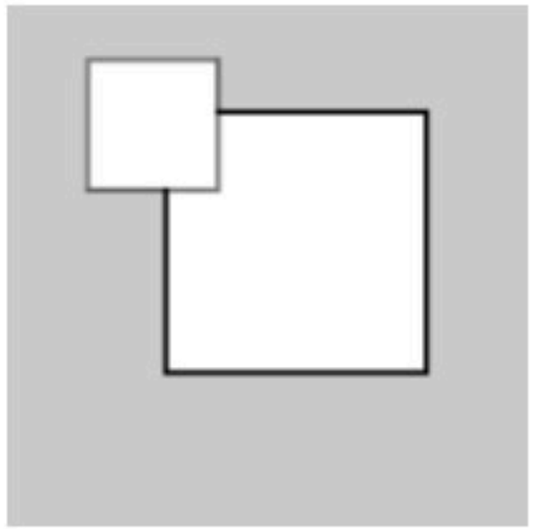
rotate

# Translate

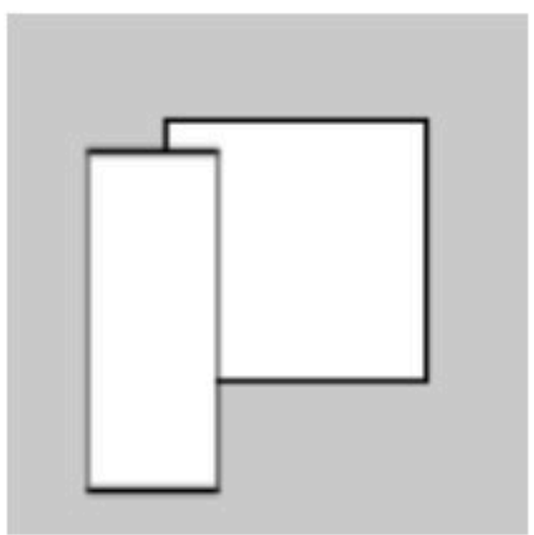


```
rect(0, 0, 55, 55);  // Draw rect at original 0,0
translate(30, 20);
rect(0, 0, 55, 55);  // Draw rect at new 0,0
translate(14, 14);
rect(0, 0, 55, 55);  // Draw rect at new 0,0
```

# Scale

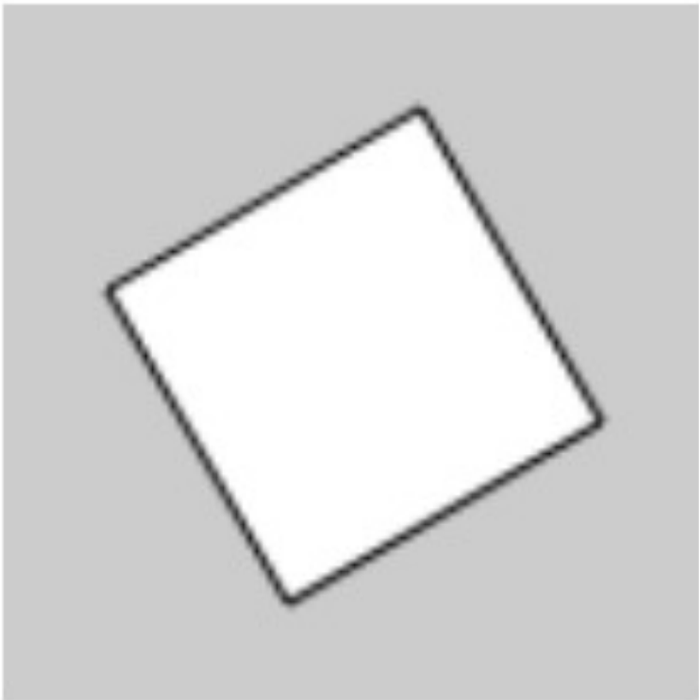


```
rect(30, 20, 50, 50);
scale(0.5);
rect(30, 20, 50, 50);
```



```
rect(30, 20, 50, 50);
scale(0.5, 1.3);
rect(30, 20, 50, 50);
```

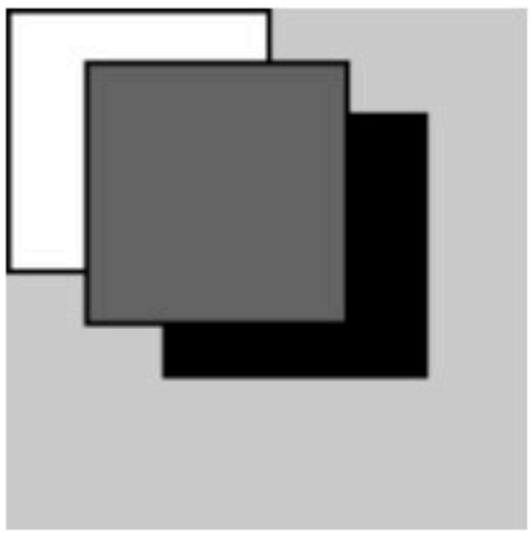
# Rotate



```
translate(width/2, height/2);
rotate(PI/3.0);
rect(-26, -26, 52, 52);
```

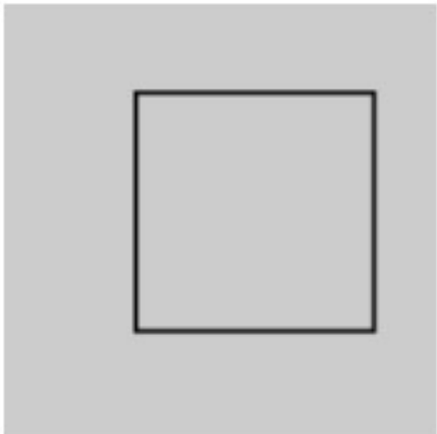
Hint: rotate(radians(30));

# Transformation UNDO

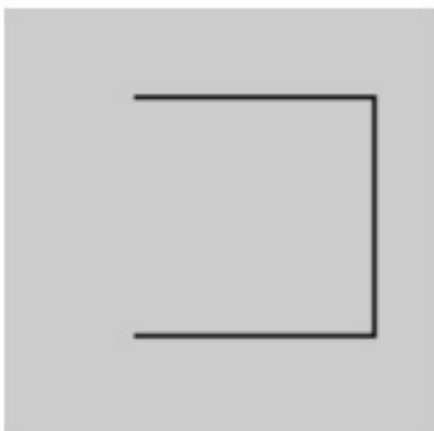


```
fill(255);
rect(0, 0, 50, 50);  // White rectangle

pushMatrix();
translate(30, 20);
fill(0);
rect(0, 0, 50, 50);  // Black rectangle
popMatrix();

fill(100);
rect(15, 10, 50, 50);  // Gray rectangle
```

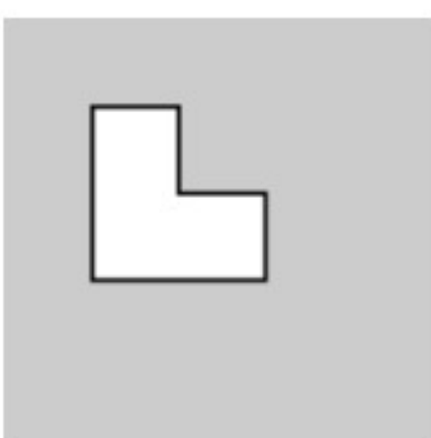
# Polygon shapes



```
noFill();
beginShape();
vertex(30, 20);
vertex(85, 20);
vertex(85, 75);
vertex(30, 75);
endShape(CLOSE);
```



```
noFill();
beginShape();
vertex(30, 20);
vertex(85, 20);
vertex(85, 75);
vertex(30, 75);
endShape();
```



```
beginShape();
vertex(20, 20);
vertex(40, 20);
vertex(40, 40);
vertex(60, 40);
vertex(60, 60);
vertex(20, 60);
endShape(CLOSE);
```

# Dynamics – animating shapes

```
void setup( ) {
  size(200, 200);
}
```

just once
on start up

```
void draw( ) {
  // erase background
  background(0);
  // draw some stuff
  // ...
}
```

every frame,
this happens

by the way: every
frame starts without
any transformations

```
// declare variable and set start value
int x = 0;
```

```
void setup( ) {
    size(400, 400);
}
```

```
void draw( ) {
    // erase background
    background(0);
    // add 1 to variable
    x = x + 1;
    // draw a rectangle of 20 by 20 pixels
    rect(x, x, 20, 20);
}
```

just once
on start up

every frame
this happens

# Exercises

- Rotate a rectangle around the center of the window
- Rotate two rectangles around each other
- Bounce a rectangle off the screen’s borders
- Bounce a rectangle ... and change the color on every bounce
- Grow a rectangle and fade it out

# Books

## Must-have

*Getting Started with Processing*, by By Casey Reas, Ben Fry
e-Book and hard copy available from O'Reilly

## Recommended-to-have

*Learning Processing: A Beginner's Guide to Programming Images, Animation, and Interaction*
Daniel Shiffman.
Published August 2008, Morgan Kaufmann. 450 pages. Paperback.
Available from LUCID, or from Amazon

*Programming Interactivity: A Designer's Guide to Processing, Arduino, and openFrameworks* (Paperback) by Joshua Noble (Author). **Very good one, covers many topics in Competency II**.
Available from LUCID. Also see http://programminginteractivity.com

*Processing: Creative Coding and Computational Art (Foundation)*
Ira Greenberg (Foreword by Keith Peters).
Published 28 May 2007, Friends of Ed. 840 pages. Hardcover.
Available from LUCID

*Making Things Talk: Practical Methods for Connecting Physical Objects*
Tom Igoe.
Published 28 September 2007, O'Reilly. 428 pages. Paperback.
Available from LUCID

# More help...

http://wiki.id.tue.nl/creapro/