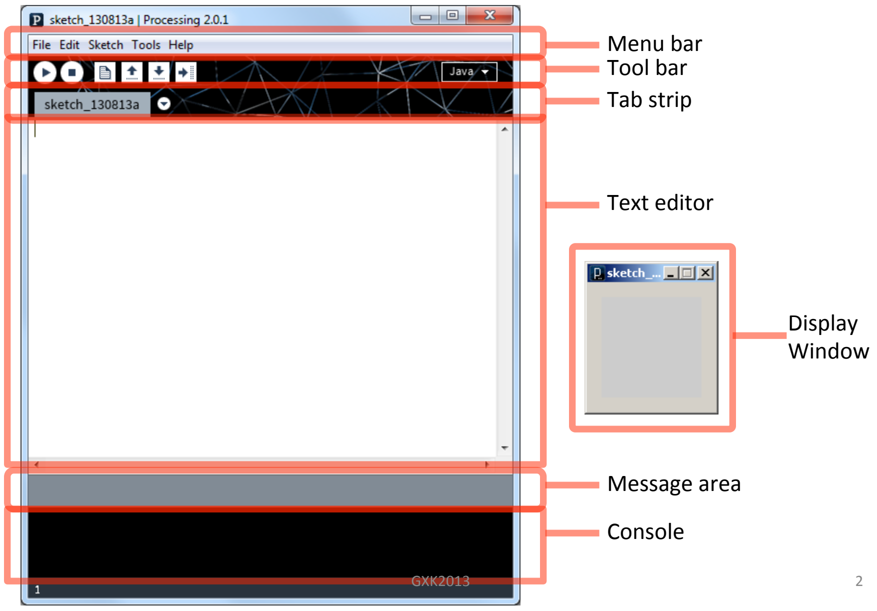
Art by Numbers

Creative Coding & Generative Art in Processing 2
Ira Greenberg, Dianna Xu, Deepak Kumar

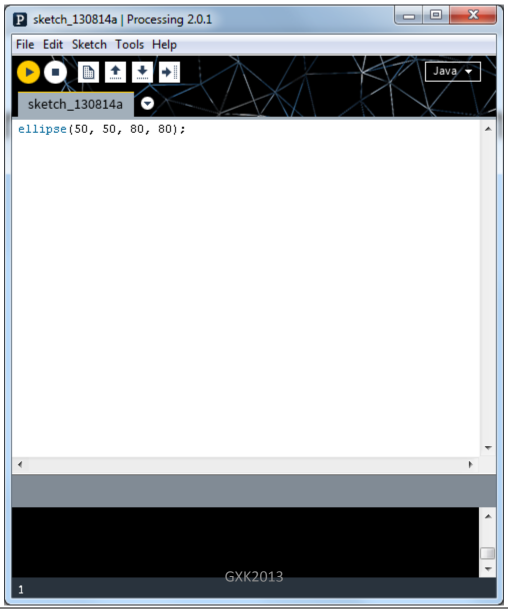
Slides modified by Michael Goldwasser

Processing 2.0 IDE



GXK2013 2

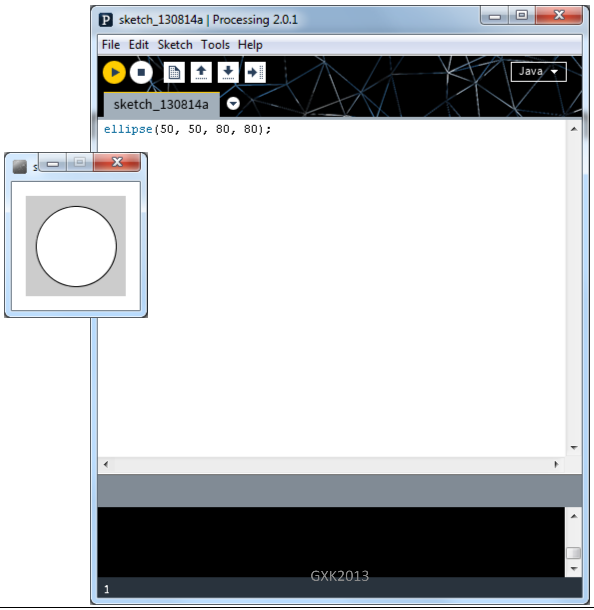
First Processing Program


sketch_130814a | Processing 2.0.1
File Edit Sketch Tools Help
Java
sketch_130814a
1

```
ellipse(50, 50, 80, 80);
```

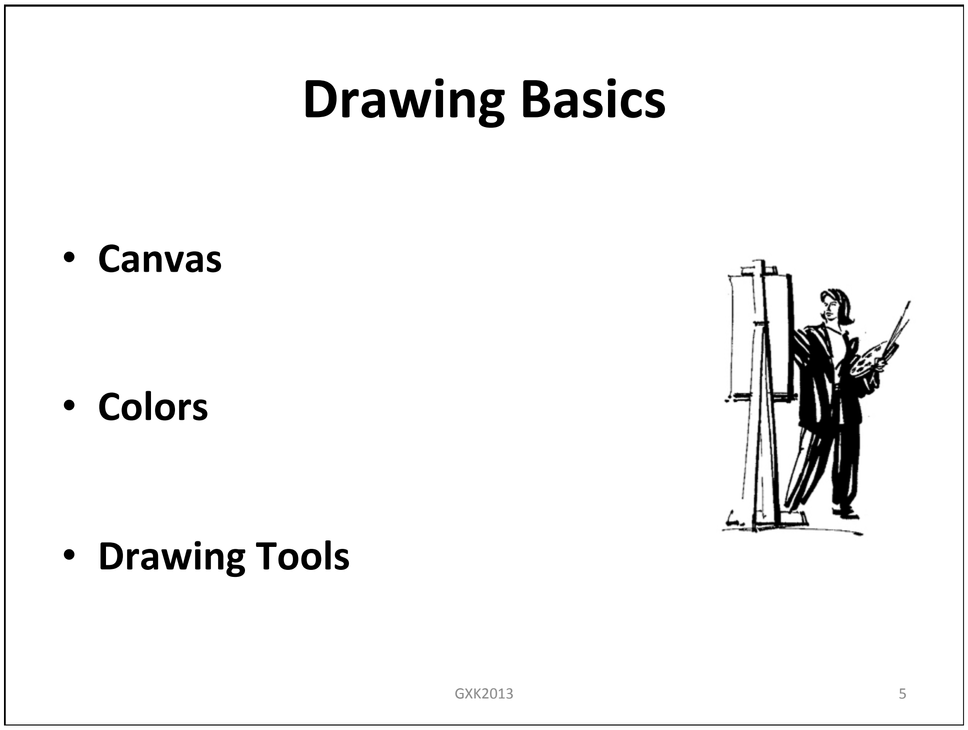
GXK2013 3

First Processing Program


sketch_130814a | Processing 2.0.1
File Edit Sketch Tools Help
Java
sketch_130814a
1

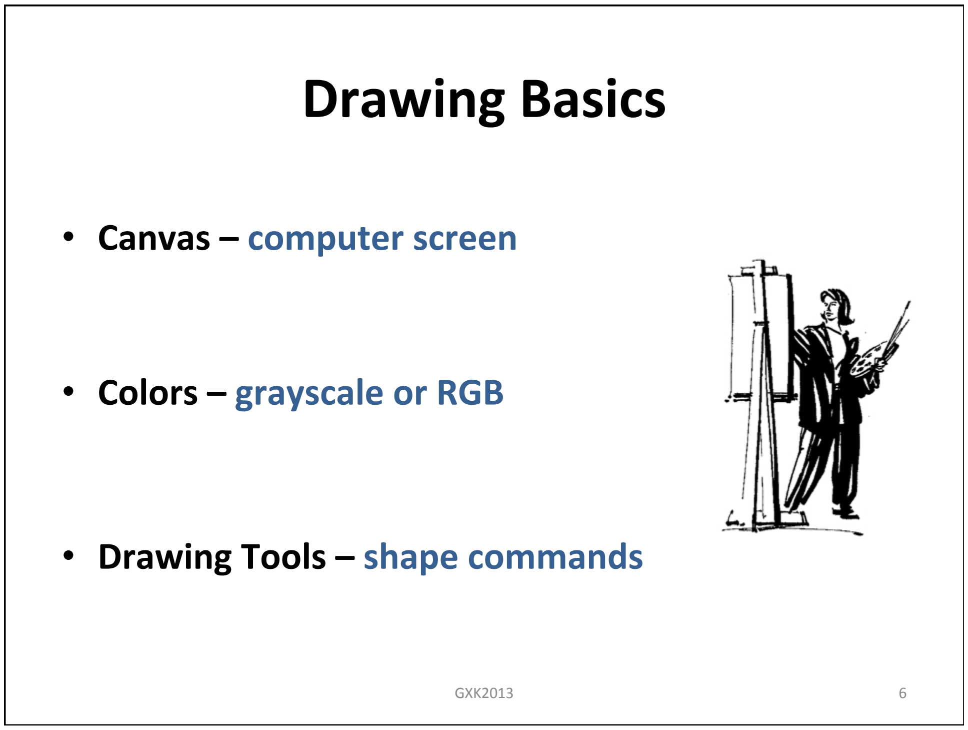
```
ellipse(50, 50, 80, 80);
```

GXK2013 4



Drawing Basics

- Canvas
- Colors
- Drawing Tools

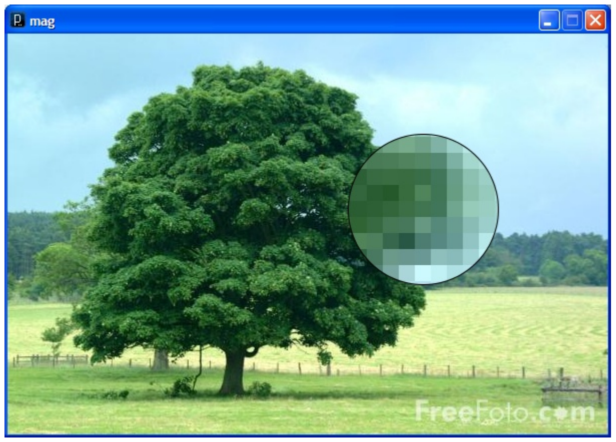


Drawing Basics

- Canvas – computer screen
- Colors – grayscale or RGB
- Drawing Tools – shape commands

Canvas – Computer Screen

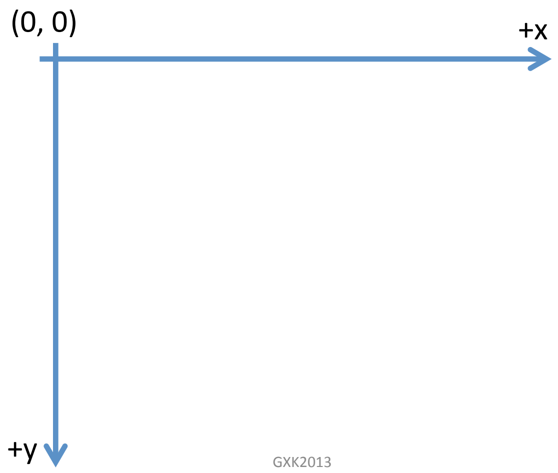
- **Pixels**



GXK2013 7

Canvas - Computer Screen

- **Coordinate System**



GXK2013 8

Canvas - Computer Screen

Processing Commands

- **Canvas:** Create a 400x400 pixel drawing area

```
size(400, 400);
```

Canvas - Computer Screen

Processing Commands

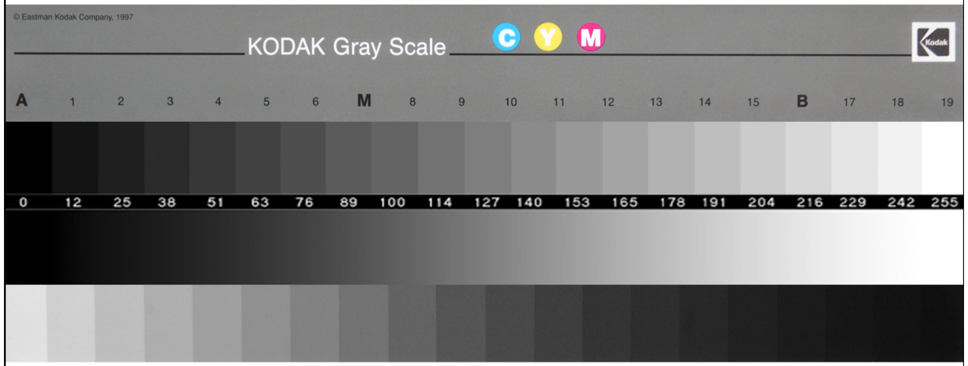
- **Canvas:** Create a 400x400 pixel drawing area

```
size(400, 400);
```

- **Canvas Color:** Canvas is gray in color

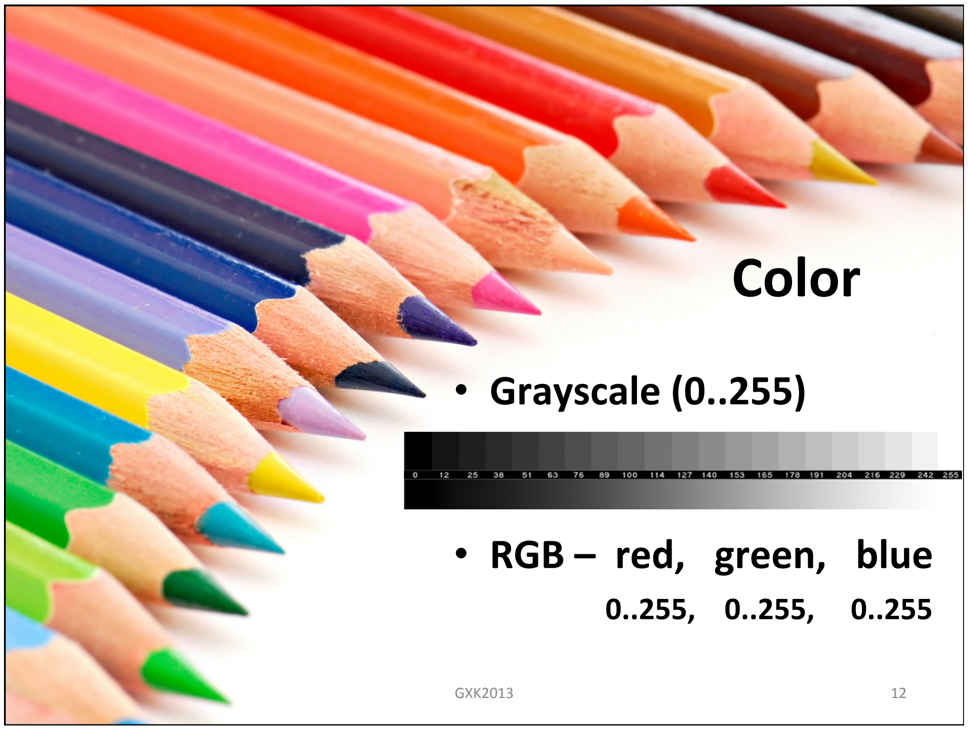
```
background(125);
```

256 Shades of Gray!



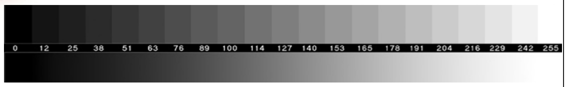
- 0 = black
- 255 = white

GXK2013 11



Color

- Grayscale (0..255)



- RGB – red, green, blue

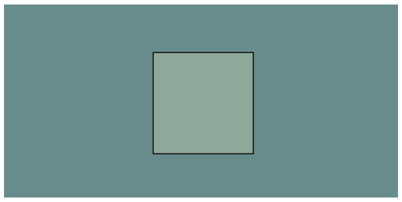
  0..255, 0..255, 0..255

GXK2013 12

Color

- Example:

```
size(400, 200);
smooth();
background(103, 140, 139);
fill(143, 168, 155);
rect(150, 50, 100, 100);
```



- Any command that takes a grayscale value, can also take RGB color values:

background(<grayscale value>);
background(R, G, B);
stroke (<grayscale value>);
stroke(R, G, B);
fill(<grayscale value>);
fill(R, G, B);

Color Transparency

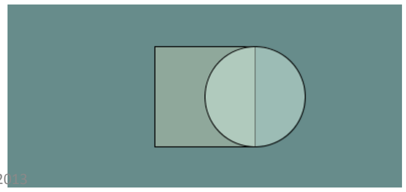
- Alpha values (0..255) specify transparency/opacity

ALPHA = 0	means completely transparent
ALPHA = 255	means completely opaque

background(<grayscale value>, **ALPHA**);
background(R, G, B, **ALPHA**);
stroke (<grayscale value>, **ALPHA**);
stroke(R, G, B, **ALPHA**);
fill(<grayscale value>, **ALPHA**);
fill(R, G, B, **ALPHA**);

- Example:

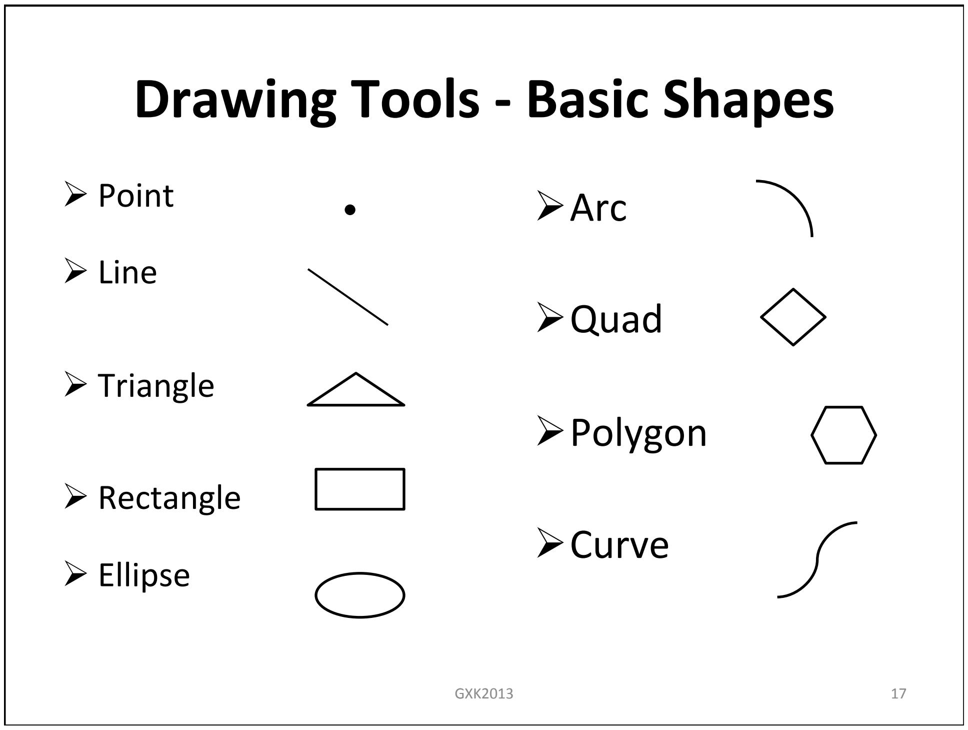
```
background(103, 140, 139);
fill(143, 168, 155);
rect(150, 50, 100, 100);
// Fill with alpha value
fill(208, 237, 222, 127);
ellipse(250, 100, 100, 100);
```



Why 0 .. 255?

Drawing Basics

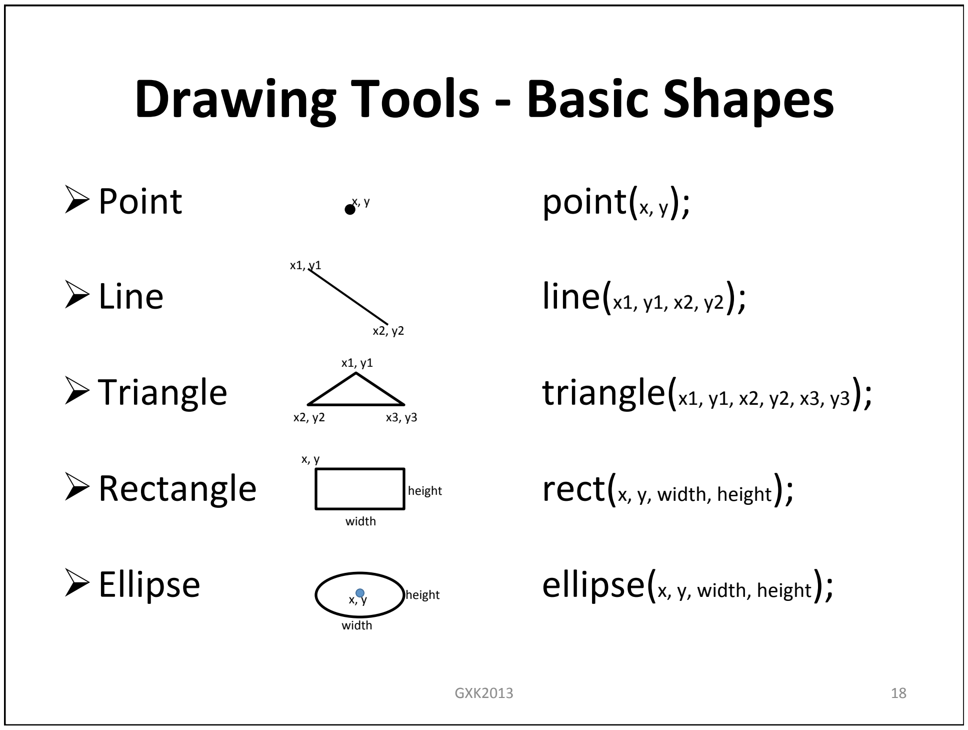
- **Canvas – computer screen**
  `size(width, height);`
- **Colors – grayscale or RGB**
  `background(125);`
- **Drawing Tools – shape commands**



Drawing Tools - Basic Shapes

- Point
- Line
- Triangle
- Rectangle
- Ellipse
- Arc
- Quad
- Polygon
- Curve

GXK2013 17

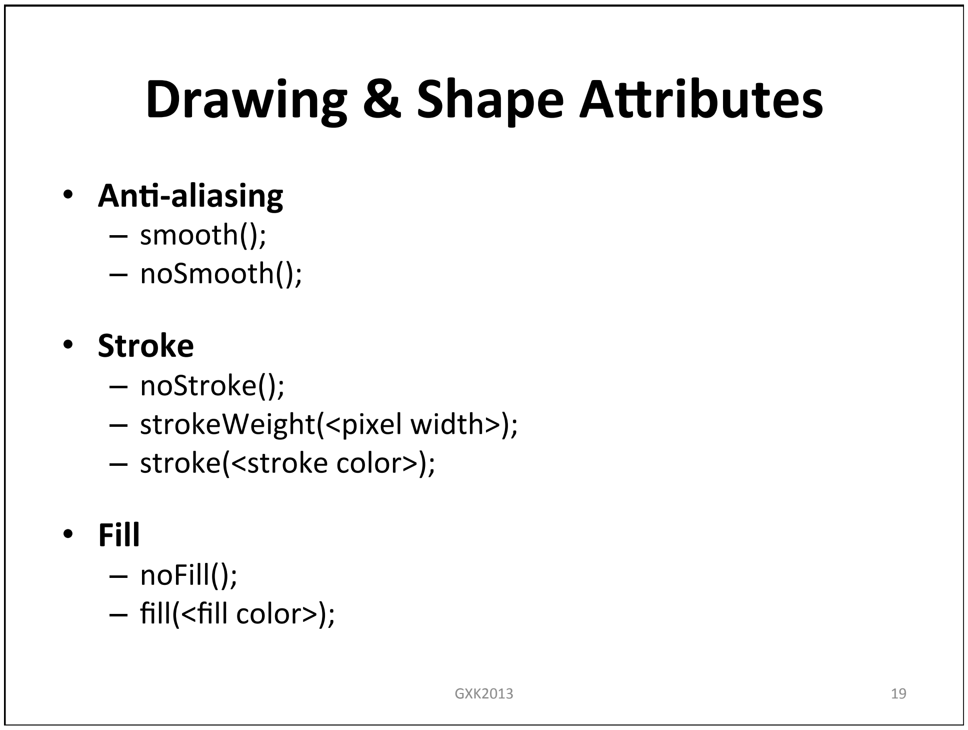


Drawing Tools - Basic Shapes

Point	point(x, y);
Line	line(x1, y1, x2, y2);
Triangle	triangle(x1, y1, x2, y2, x3, y3);
Rectangle	rect(x, y, width, height);
Ellipse	ellipse(x, y, width, height);

x, y
x1, y1
x2, y2
x1, y1
x2, y2
x3, y3
x, y
height
width
x, y
height
width

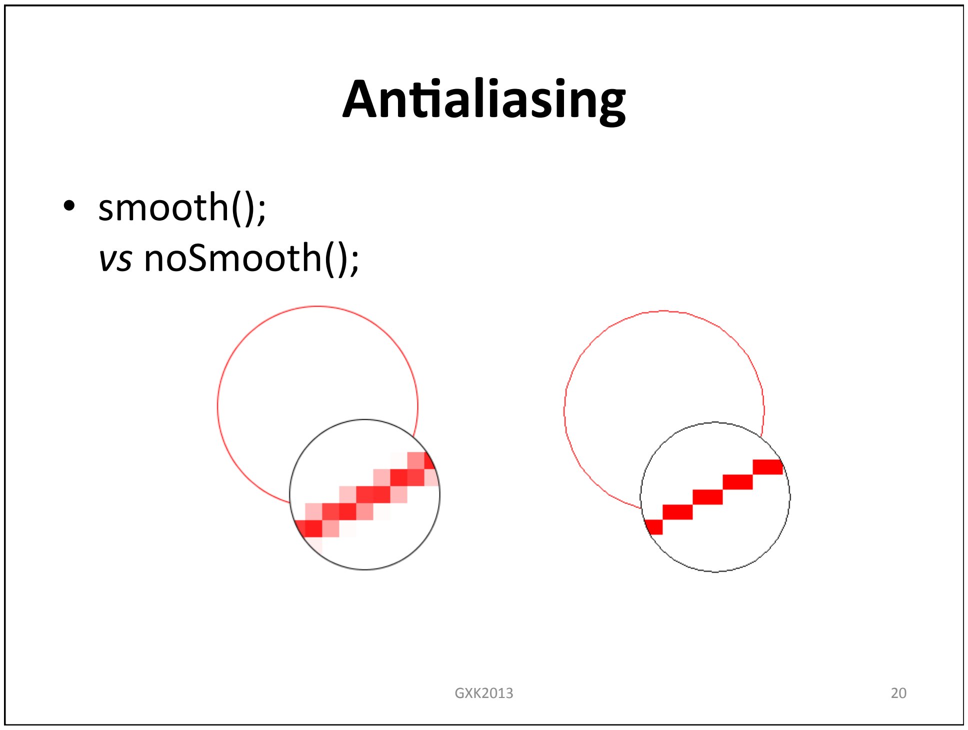
GXK2013 18



Drawing & Shape Attributes

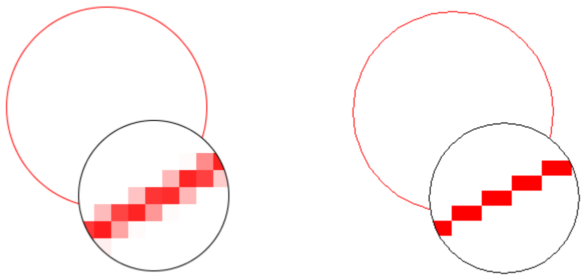
- **Anti-aliasing**
  - smooth();
  - noSmooth();
- **Stroke**
  - noStroke();
  - strokeWeight(<pixel width>);
  - stroke(<stroke color>);
- **Fill**
  - noFill();
  - fill(<fill color>);

GXK2013 19



Antialiasing

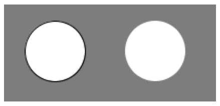
- smooth();
  vs noSmooth();



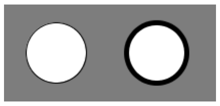
GXK2013 20

Stroke Attributes

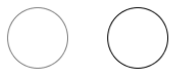
- stroke();
  vs noStroke();



- strokeWeight(1);
  vs strokeWeight(5);



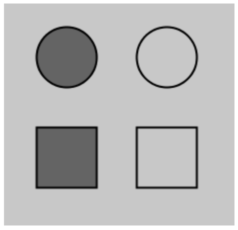
- stroke(125);
  vs stroke(0);



GXK2013 21

Fill Attributes

- fill(100);
  vs noFill();



GXK2013 22

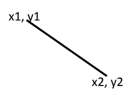
Drawing & Shape Attributes

- **Anti-aliasing**
  - smooth();
  - noSmooth();
- **Stroke**
  - noStroke();
  - strokeWeight(<pixel width>);
  - stroke(<stroke color>);
- **Fill**
  - noFill();
  - fill(<fill color>);

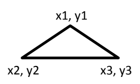
Drawing Tools - Basic Shapes

➢ Point

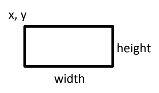
point(x, y);

➢ Line


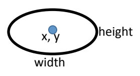
line(x1, y1, x2, y2);

➢ Triangle


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

➢ Rectangle


rect(x, y, width, height);

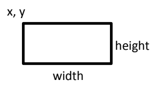
➢ Ellipse


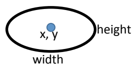
ellipse(x, y, width, height);

Modes

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

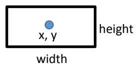
  ellipse(x, y, width, height);

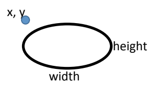




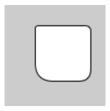
- rectMode(CENTER);

  ellipseMode(CORNER);

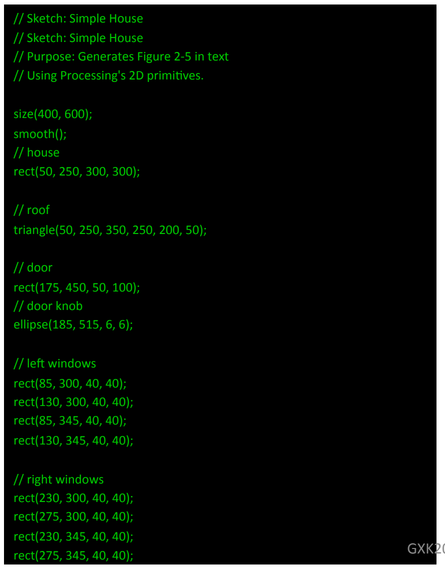




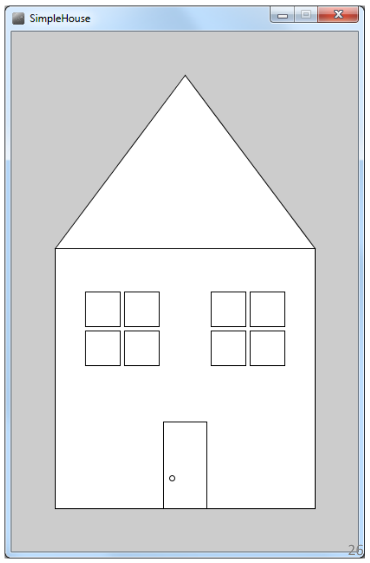
- Also CORNERS (see Reference)
- Also rounded rectangles (see Reference)



Structure of a basic program



```
// Sketch: Simple House
// Sketch: Simple House
// Purpose: Generates Figure 2-5 in text
// Using Processing's 2D primitives.

size(400, 600);
smooth();
// house
rect(50, 250, 300, 300);

// roof
triangle(50, 250, 350, 250, 200, 50);

// door
rect(175, 450, 50, 100);
// door knob
ellipse(185, 515, 6, 6);

// left windows
rect(85, 300, 40, 40);
rect(130, 300, 40, 40);
rect(85, 345, 40, 40);
rect(130, 345, 40, 40);

// right windows
rect(230, 300, 40, 40);
rect(275, 300, 40, 40);
rect(230, 345, 40, 40);
rect(275, 345, 40, 40);
```



Programming Principle#1

- **Sequencing**

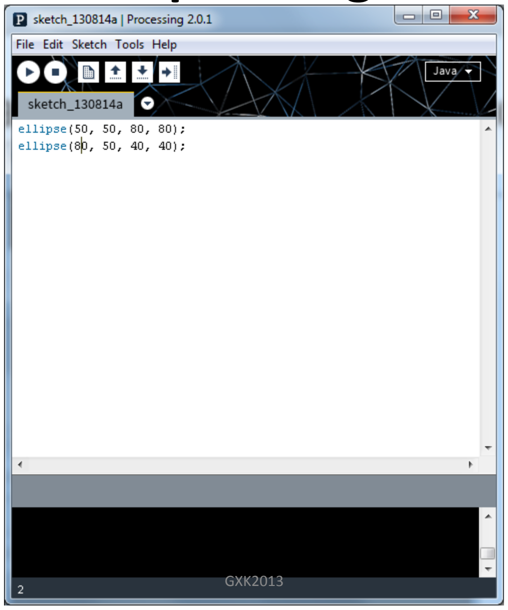
  do this
  and this
  and this
  and this
  …

```
// left windows
rect(85, 300, 40, 40);
rect(130, 300, 40, 40);
rect(85, 345, 40, 40);
rect(130, 345, 40, 40);

// right windows
rect(230, 300, 40, 40);
rect(275, 300, 40, 40);
rect(230, 345, 40, 40);
rect(275, 345, 40, 40);
```

All commands are carried out in the order they are written.

GXK2013 27

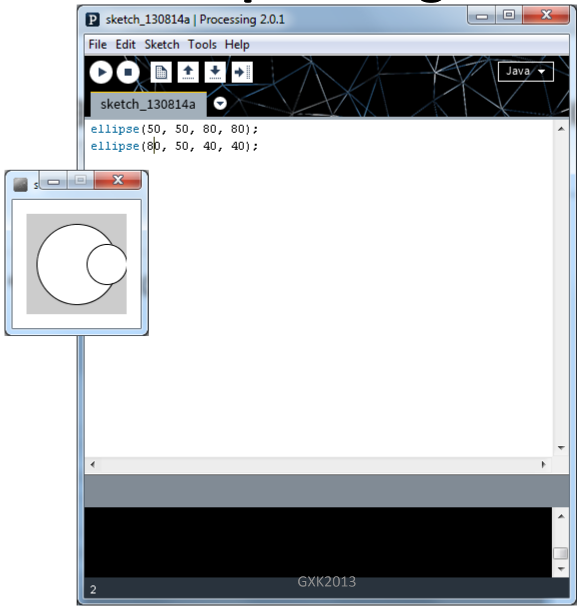
Sequencing…


sketch_130814a | Processing 2.0.1
File Edit Sketch Tools Help
Java
sketch_130814a
2

```
ellipse(50, 50, 80, 80);
ellipse(80, 50, 40, 40);
```

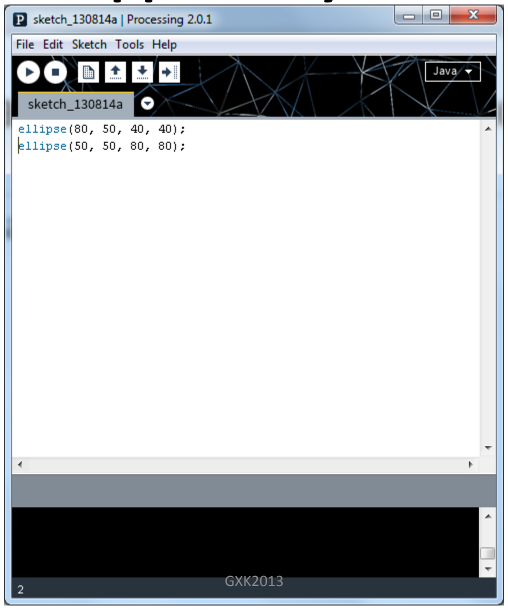
GXK2013 28

Sequencing…


sketch_130814a | Processing 2.0.1
File Edit Sketch Tools Help
Java
sketch_130814a

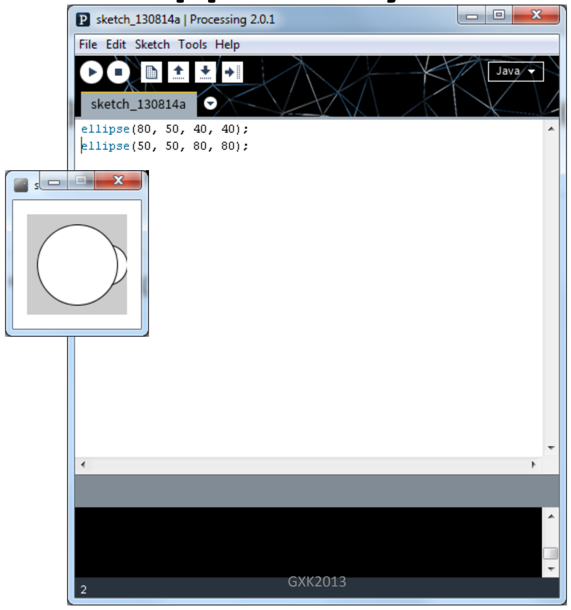
```
ellipse(50, 50, 80, 80);
ellipse(80, 50, 40, 40);
```

What happens if you switch?


sketch_130814a | Processing 2.0.1
File Edit Sketch Tools Help
Java
sketch_130814a

```
ellipse(80, 50, 40, 40);
ellipse(50, 50, 80, 80);
```

What happens if you switch?

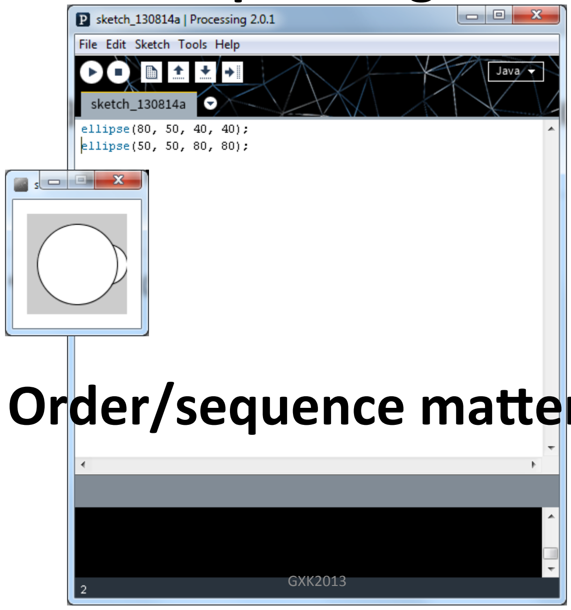

sketch_130814a | Processing 2.0.1
File Edit Sketch Tools Help
Java
sketch_130814a
2

```
ellipse(80, 50, 40, 40);
ellipse(50, 50, 80, 80);
```

GXK2013

31

Sequencing…


sketch_130814a | Processing 2.0.1
File Edit Sketch Tools Help
Java
sketch_130814a
2

```
ellipse(80, 50, 40, 40);
ellipse(50, 50, 80, 80);
```

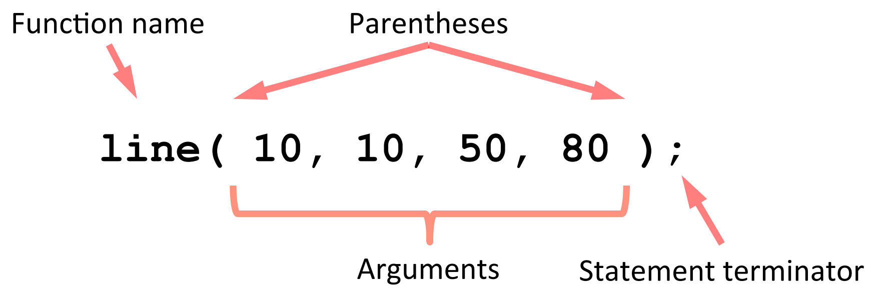
Order/sequence matters!

GXK2013

32

Programming Principle#2

- **Syntax is important!**



```
line( 10, 10, 50, 80 );
```

GXK2013 33

CS Principle: Algorithms

An **algorithm** is an effective method for solving a problem expressed as a finite sequence of instructions. For example,

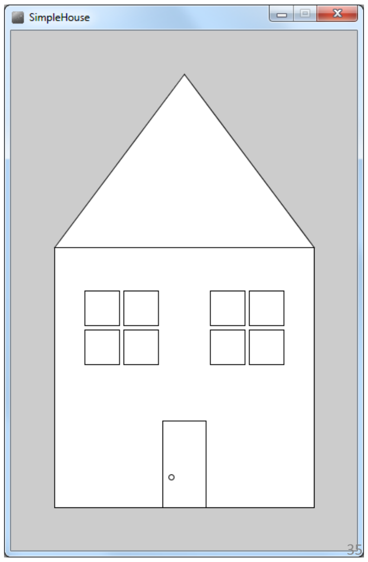
Put on shoes
- left sock
- right sock
- left shoe
- right shoe

GXK2013 34

CS Principle: Algorithms

Draw a simple house

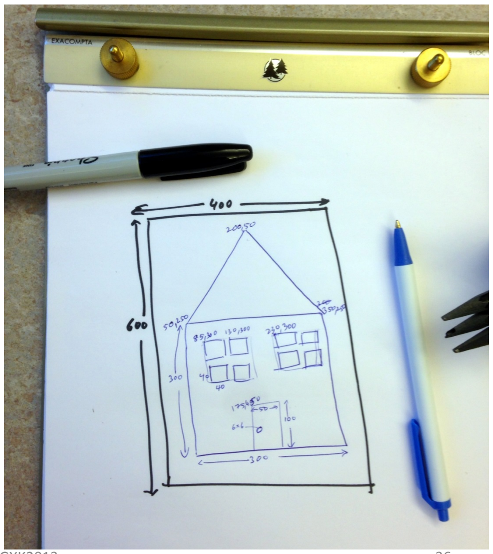
- draw the front wall
- draw the roof
- draw the door
- draw the windows



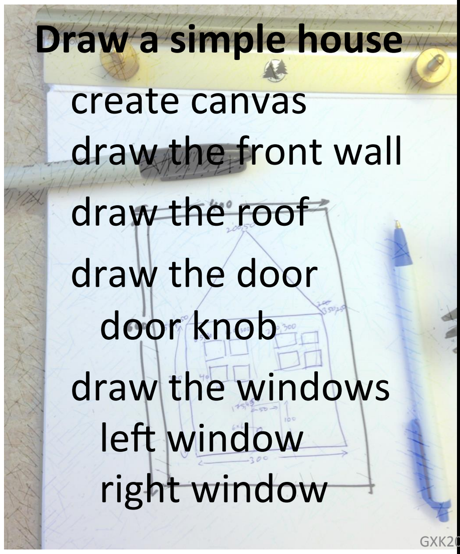
Algorithms to Pseudocode

Draw a simple house

```
create canvas
draw the front wall
draw the roof
draw the door
    door knob
draw the windows
    left window
    right window
```



Pseudocode to Code



Draw a simple house

```
create canvas
draw the front wall
draw the roof
draw the door
  door knob
draw the windows
  left window
  right window
```

```
// Sketch: Simple House
// Sketch: Simple House
// Purpose: Generates Figure 2-5 in text
// Using Processing's 2D primitives.

size(400, 600);

// house
rect(50, 250, 300, 300);

// roof
triangle(50, 250, 350, 250, 200, 50);

// door
rect(175, 450, 50, 100);
// door knob
ellipse(185, 515, 6, 6);

// left windows
rect(85, 300, 40, 40);
rect(130, 300, 40, 40);
rect(85, 345, 40, 40);
rect(130, 345, 40, 40);

// right windows
rect(230, 300, 40, 40);
rect(275, 300, 40, 40);
rect(230, 345, 40, 40);
rect(275, 345, 40, 40);
```

GXK2013 37

CS Principle

To solve any problem on a computer
- First **analyze** the problem
- Then design an **algorithm**
- Write **pseudocode**
- **Code** it
- **Test** and **debug**

GXK2013 38

CS Principle

To solve any problem on a computer

First **analyze** the problem

Then design an **algorithm**

Write **pseudocode**

Code it

Test and **debug**

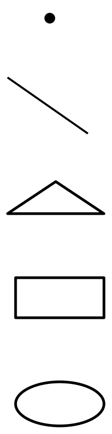
Much work happens on paper!

Drawing Basics

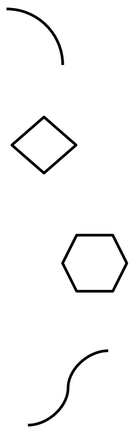
- **Canvas – computer screen**
  `size(`*`width, height`*`);`
- **Colors – grayscale or RGB**
  `background(125);`
- **Drawing Tools – shape commands**

Drawing Tools - Basic Shapes

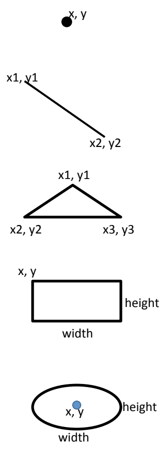
- Point
- Line
- Triangle
- Rectangle
- Ellipse



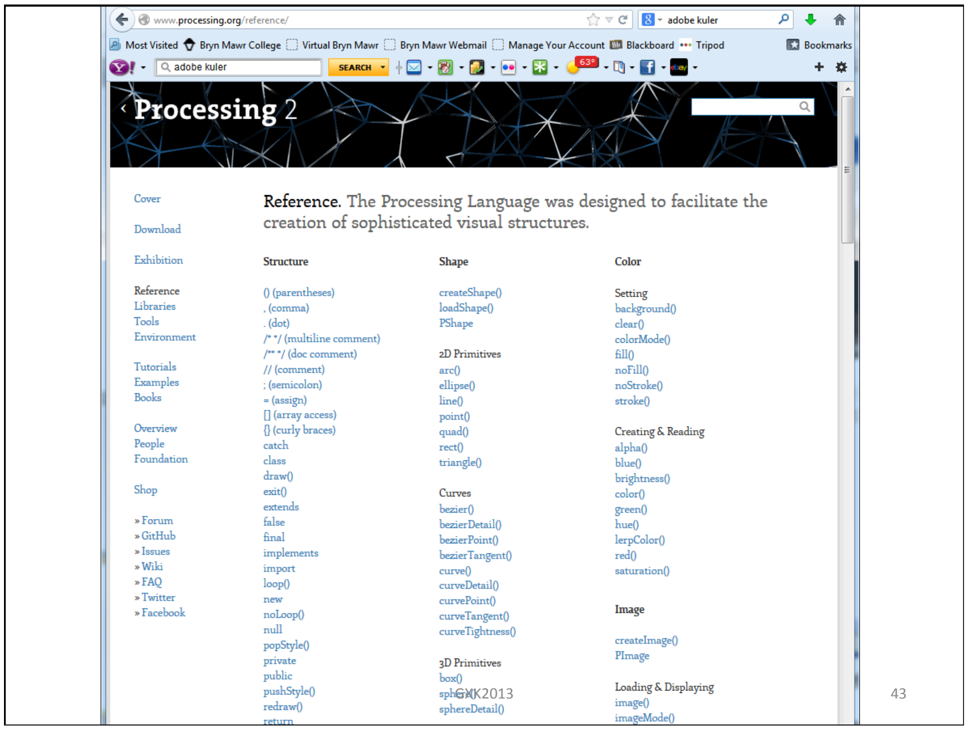
- Arc
- Quad
- Polygon
- Curve



Drawing Tools - Basic Shapes



- Point point(x, y);
- Line line(x1, y1, x2, y2);
- Triangle triangle(x1, y1, x2, y2, x3, y3);
- Rectangle rect(x, y, width, height);
- Ellipse ellipse(x, y, width, height);



www.processing.org/reference/

Most Visited · Bryn Mawr College · Virtual Bryn Mawr · Bryn Mawr Webmail · Manage Your Account · Blackboard · Tripod · Bookmarks

adobe kuler

Processing 2

Cover

Download

Exhibition

Reference
Libraries
Tools
Environment

Tutorials
Examples
Books

Overview
People
Foundation

Shop

» Forum
» GitHub
» Issues
» Wiki
» FAQ
» Twitter
» Facebook

Reference. The Processing Language was designed to facilitate the creation of sophisticated visual structures.

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

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()

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()

GXK2013

43