

# Art by Numbers

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

# Our Goal

- Use computing to realize works of art
- Explore new metaphors from computing: images, animation, interactivity, visualizations
- Learn the basics of computing
- Have fun doing all of the above!

GXK2013

2

# Let's get started...

GXK2013

3

## Administrivia

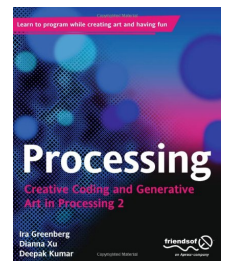
### Software

#### Processing 2.X

- Already installed in the CS Lab
- Also available for your own computer @ [www.processing.org](http://www.processing.org)
- Processing == Java

### Book

**Creative Coding & Generative Art in Processing 2**  
by Ira Greenberg, Dianna Xu, Deepak Kumar, friendsofEd/APress, 2013. Available at the Campus Bookstore or amazon.com or other vendors.



GXK2013

4

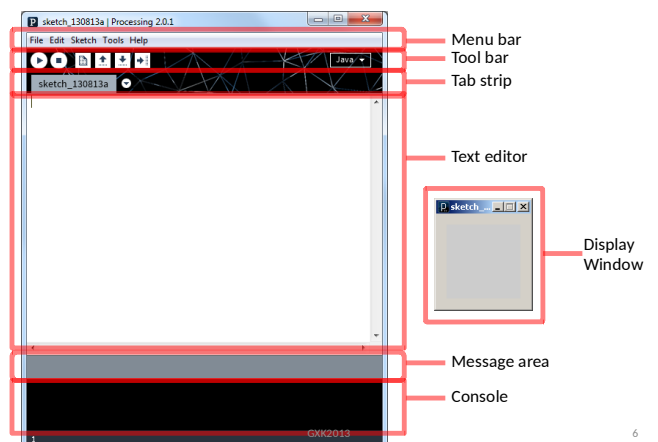
# Did you do this?

- Go to the CS Computer Lab (Room 231 PSB)
- Log in
- Start the Processing application (Make sure it is Version 2.x)
- In a web browser, go to the Tutorials section of [processing.org](http://www.processing.org)  
<http://www.processing.org/tutorials/gettingstarted/>
- Read the Getting Started tutorial (by Casey Reas & Ben Fry) and try out the two examples of simple Processing programs presented there
- If you'd like, install Processing 2.x on your own computer
- Read Chapter 1 (Read pages 1-12, skim 12-32)

GXK2013

5

# Processing 2.0 IDE



GXK2013

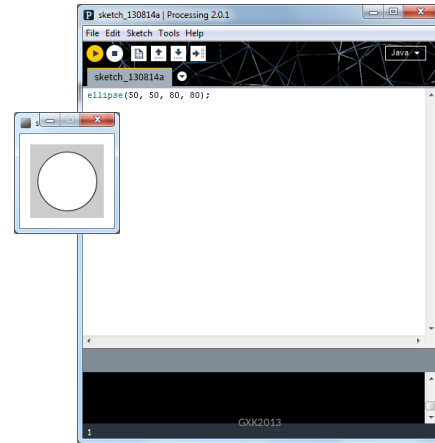
6

## First Processing Program



7

## First Processing Program



8

## Drawing Basics

- Canvas
- Drawing Tools
- Colors



GJK2013

9

## Drawing Basics

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



GJK2013

10

## Canvas - Computer Screen

- Pixels

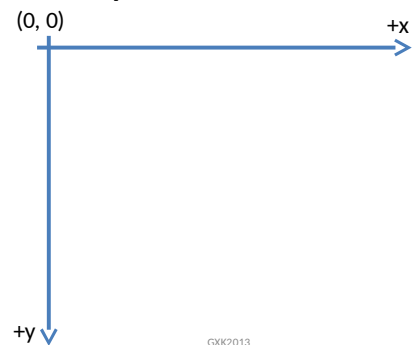


GJK2013

11

## Canvas - Computer Screen

- Coordinate System



GJK2013

12

## Canvas - Computer Screen

### Processing Commands

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

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

GJK2013

13

## Canvas - Computer Screen

### Processing Commands

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

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

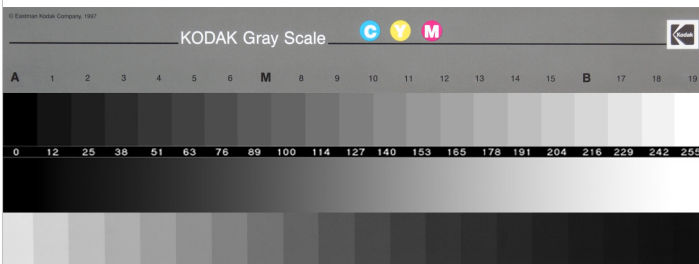
- **Canvas Color:** Canvas is gray in color

```
background(125);
```

GJK2013

14

## 256 Shades of Gray!



- 0 = black
- 255 = white

GJK2013

15

## Drawing Basics

- **Canvas - computer screen**  
`size(width, height);`

- **Drawing Tools - shape commands**

- **Colors - grayscale or RGB**  
`background(125);`



GJK2013

16

## Drawing Tools - Basic Shapes

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

GJK2013

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);`

GJK2013

18

## Drawing & Shape Attributes

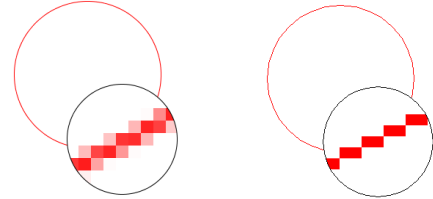
- **Anti-aliasing**
  - smooth();
  - noSmooth();
- **Stroke**
  - noStroke();
  - strokeWeight(<pixel width>);
  - stroke(<stroke color>);
- **Fill**
  - noFill();
  - fill(<fill color>);

GJK2013

19

## Antialiasing

- smooth();  
vs noSmooth();



GJK2013

20

## Stroke Attributes

- stroke();  
vs noStroke();
- strokeWeight(1);  
vs strokeWeight(5);
- stroke(125);  
vs stroke(0);

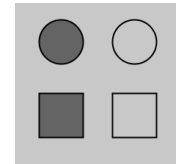


GJK2013

21

## Fill Attributes

- fill(100);  
vs noFill();



GJK2013

22

## Drawing & Shape Attributes

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

GJK2013

23

## 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);

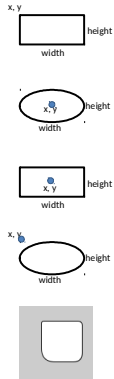


GJK2013

24

## Modes

- `rect(x, y, width, height);`
- `ellipse(x, y, width, height);`
- `rectMode(CENTER);`
- `ellipseMode(CORNER);`
- Also CORNERS (see Reference)
- Also rounded rectangles (see Reference)



GJK2013

25

## 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);

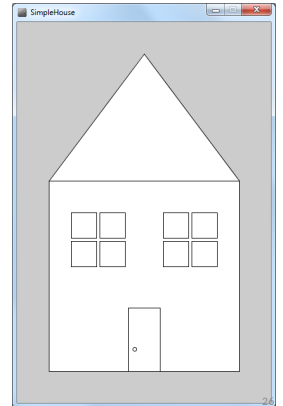
// roof
triangle(30, 250, 330, 230, 200, 310);

// 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(185, 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);
```

GJK2013



26

## 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(185, 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.

GJK2013

27

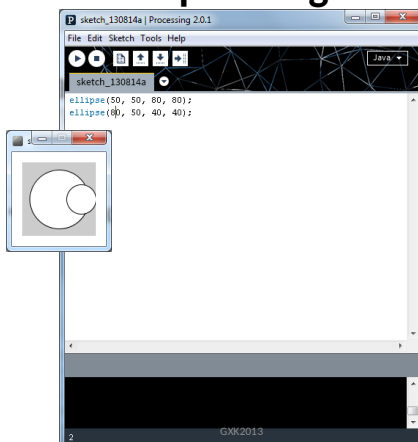
## Sequencing...



GJK2013

28

## Sequencing...



GJK2013

29

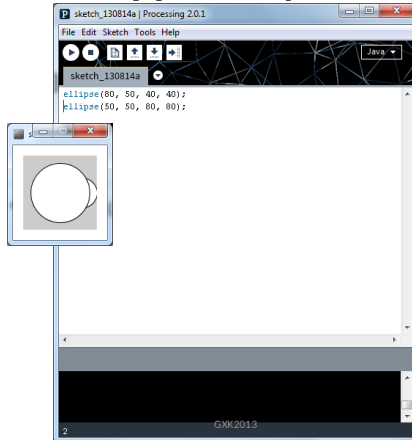
## What happens if you switch?



GJK2013

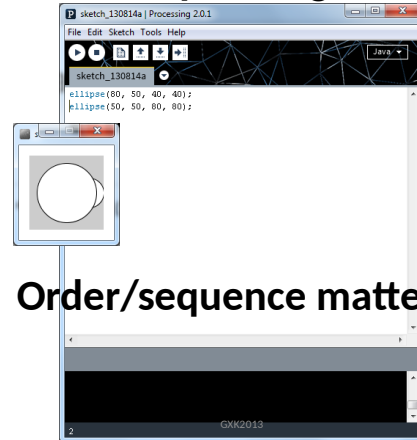
30

## What happens if you switch?



31

## Sequencing...

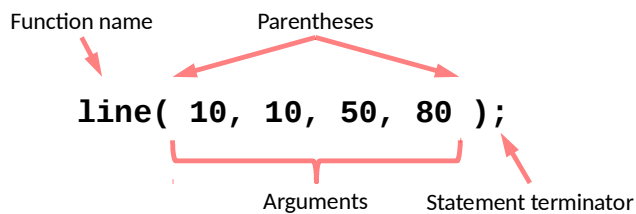


Order/sequence matters!

32

## Programming Principle#2

- Syntax is important!



GJK2013

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



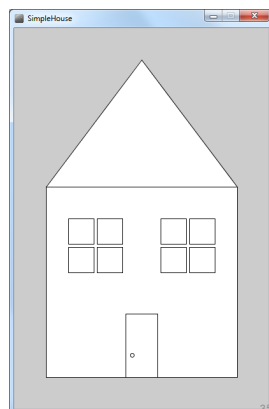
GJK2013

34

## CS Principle: Algorithms

### Draw a simple house

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



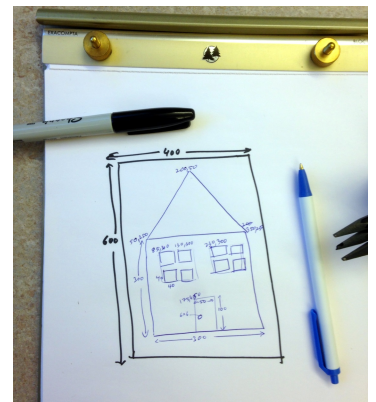
GJK2013

35

## 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



GJK2013

36

## 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(150, 250, 300, 300);

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

// door
rect(175, 450, 300, 500);
// door knob
ellipse(185, 515, 6, 6);

// left windows
rect(80, 350, 100, 400);
rect(110, 300, 140, 400);
rect(180, 40, 210, 40);
rect(130, 345, 160, 40);

// right windows
rect(220, 300, 250, 400);
rect(250, 350, 280, 400);
rect(220, 345, 250, 40);
rect(275, 345, 305, 40);
```

## CS Principle

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

## 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*);
- Drawing Tools - **shape commands**
- Colors - **grayscale or RGB**  
 background(*125*);



## 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*);

# Color

- Grayscale (0..255)

- RGB - red, green, blue  
0..255, 0..255, 0..255

GJK2013 43

# Color

- Example:

```
size(400, 200);
smooth();
background(100, 140, 130);
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);
```

GJK2013 44

# 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(100, 140, 130);
fill(143, 168, 155);
rect(150, 50, 100, 100);
// Fill with alpha value
fill(200, 237, 222, 127);
ellipse(250, 100, 100, 100);
```

GJK2013 45

# Why 0 .. 255?

GJK2013 46

Structure	Shape	Color	Image
<ul style="list-style-type: none"> <li>() parentheses</li> <li>[,] comma</li> <li>loadShape()</li> <li>PSshape</li> <li>/* */ (multiline comment)</li> <li>/**/ (doc comment)</li> <li>// (comment)</li> <li>/(parentheses)</li> <li>= (assign)</li> <li>[] (array access)</li> <li>{} (curly braces)</li> <li>catch</li> <li>class</li> <li>draw()</li> <li>exit()</li> <li>extends</li> <li>false</li> <li>final</li> <li>implements</li> <li>import</li> <li>loop()</li> <li>new</li> <li>noLoop()</li> <li>null</li> </ul>	<ul style="list-style-type: none"> <li>createShape()</li> <li>loadShape()</li> <li>PSshape</li> <li>2D Primitives</li> <li>arc()</li> <li>ellipse()</li> <li>line()</li> <li>point()</li> <li>quad()</li> <li>rect()</li> <li>triangle()</li> <li>Curves</li> <li>bezier()</li> <li>bezierDetail()</li> <li>bezierPoint()</li> <li>bezierTangent()</li> <li>curve()</li> <li>curveDetail()</li> <li>curvePoint()</li> <li>curveTangent()</li> <li>curveTightness()</li> </ul>	<ul style="list-style-type: none"> <li>Setting</li> <li>background()</li> <li>clear()</li> <li>colorMode()</li> <li>fill()</li> <li>noFill()</li> <li>noStroke()</li> <li>stroke()</li> <li>Creating &amp; Reading</li> <li>alpha()</li> <li>bhae()</li> <li>brightness()</li> <li>color()</li> <li>gamma()</li> <li>hae()</li> <li>lerpColor()</li> <li>red()</li> <li>saturation()</li> </ul>	<ul style="list-style-type: none"> <li>Image</li> </ul>

GJK2013 47

GJK2013 48