

Art by Numbers

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

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!

Let's get started...

GXX2013

3

Administrivia

Software

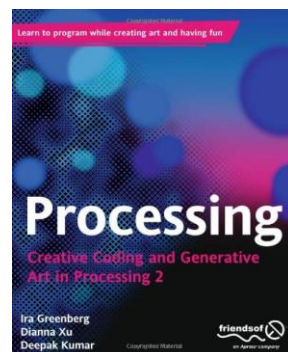
Processing 2.X

- Already installed in the CS Lab
- Also available for your own computer @ 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.



GXX2013

4

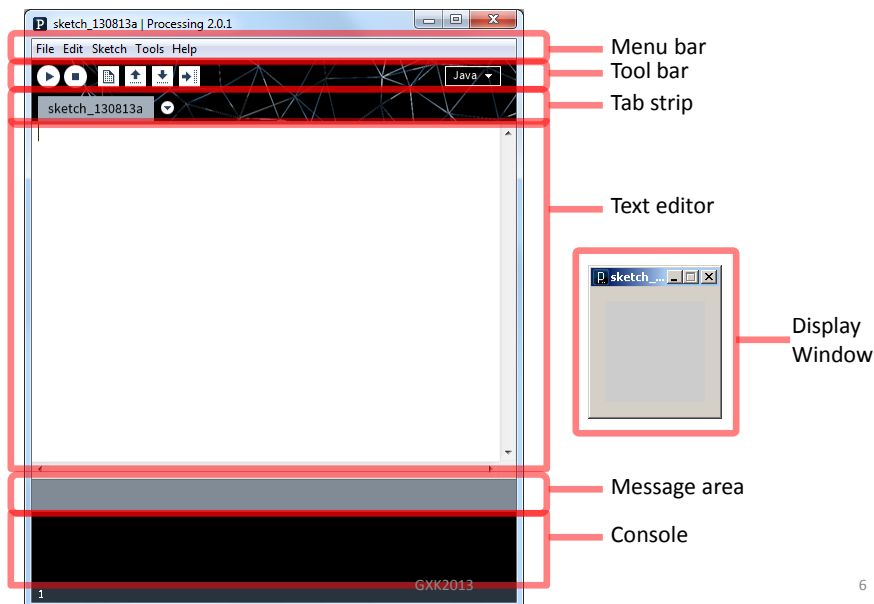
Did you do this?

- Go 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/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)

GXXK2013

5

Processing 2.0 IDE



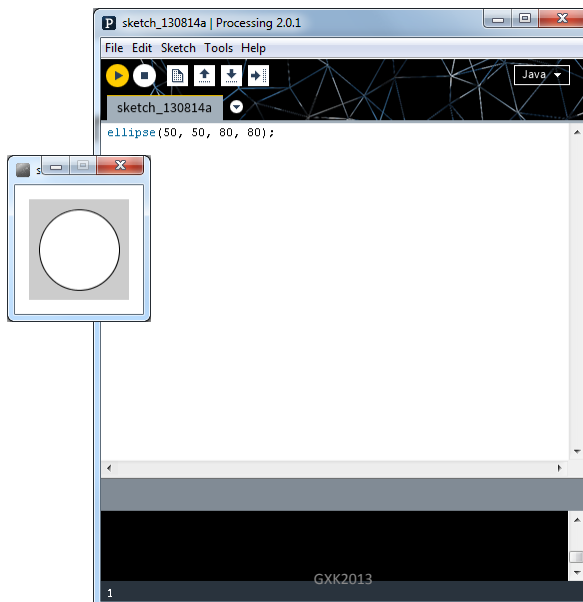
6

First Processing Program



7

First Processing Program



8

Drawing Basics

- Canvas
- Drawing Tools
- Colors



GXX2013

9

Drawing Basics

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



GXX2013

10

Canvas – Computer Screen

- Pixels

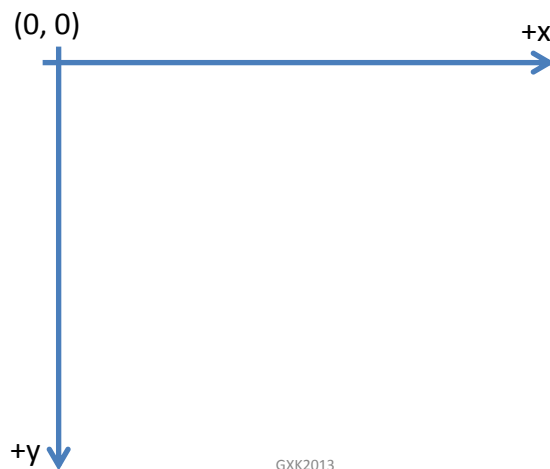


GXX2013

11

Canvas - Computer Screen

- Coordinate System



GXX2013

12

Canvas - Computer Screen

Processing Commands

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

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

GXX2013

13

Canvas - Computer Screen

Processing Commands

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

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

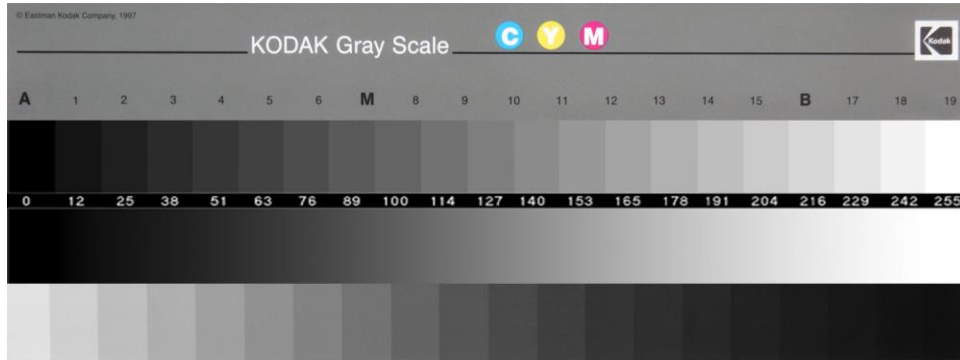
- **Canvas Color:** Canvas is gray in color

```
background(125);
```

GXX2013

14

256 Shades of Gray!



- 0 = black
- 255 = white

GXX2013

15

Drawing Basics





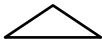




- **Canvas** – **computer screen**
`size(width, height);`
- **Drawing Tools** – **shape commands**
- **Colors** – **grayscale or RGB**
`background(125);`



GXX2013

16


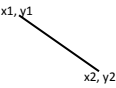
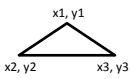
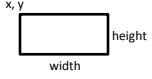
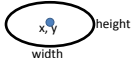
Drawing Tools - Basic Shapes

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

GXX2013

17

Drawing Tools - Basic Shapes

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

GXX2013

18

Drawing & Shape Attributes

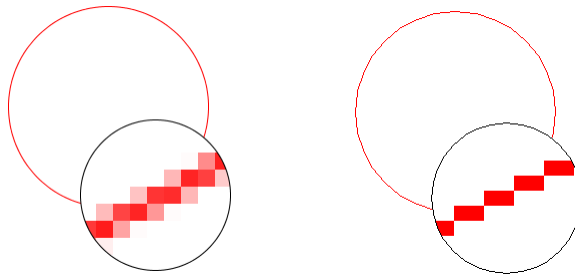
- **Anti-aliasing**
 - smooth();
 - noSmooth();
- **Stroke**
 - noStroke();
 - strokeWeight(<pixel width>);
 - stroke(<stroke color>);
- **Fill**
 - noFill();
 - fill(<fill color>);

GJK2013

19

Antialiasing




- smooth();
vs noSmooth();



GJK2013

20

Stroke Attributes

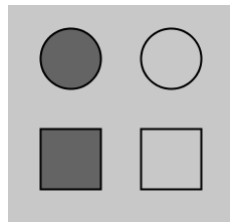
- `stroke();`
vs `noStroke();`

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

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


GJK2013

21

Fill Attributes

- `fill(100);`
vs `noFill();`



GJK2013

22



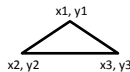
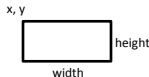

Drawing & Shape Attributes

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

GJK2013

23

Drawing Tools - Basic Shapes

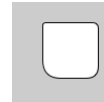
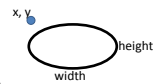
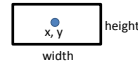
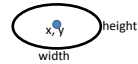
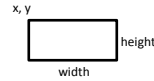
- Point  `point(x, y);`
- Line  `line(x1, y1, x2, y2);`
- Triangle  `triangle(x1, y1, x2, y2, x3, y3);`
- Rectangle  `rect(x, y, width, height);`
- Ellipse  `ellipse(x, y, width, height);`

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);

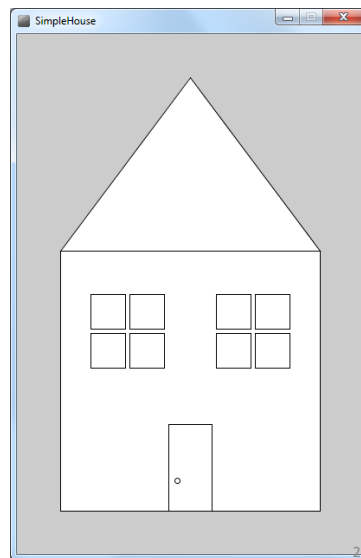
// roof
triangle(50, 250, 350, 250, 200, 50);

// 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);
```

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(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.

GJK2013

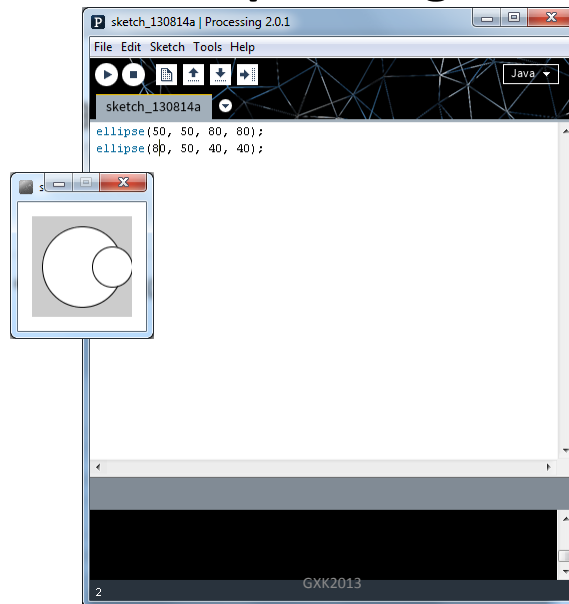
27

Sequencing...



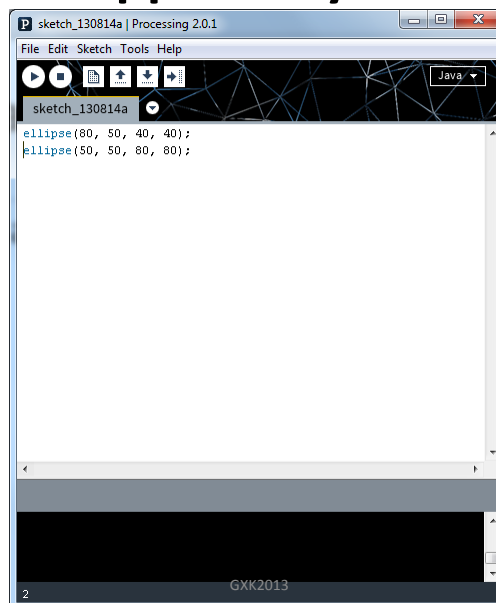
28

Sequencing...



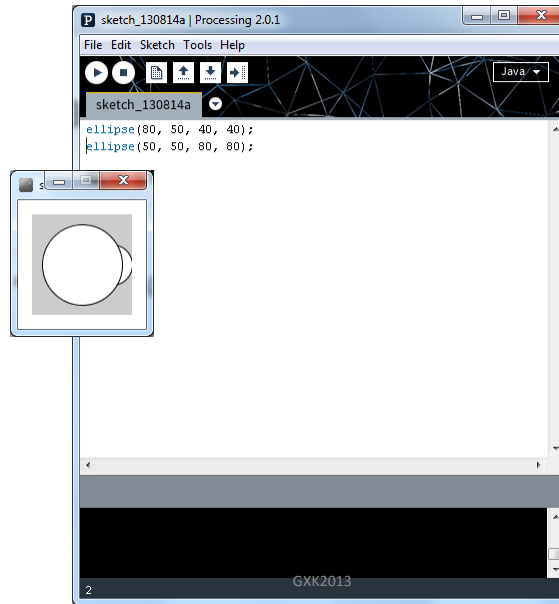
29

What happens if you switch?



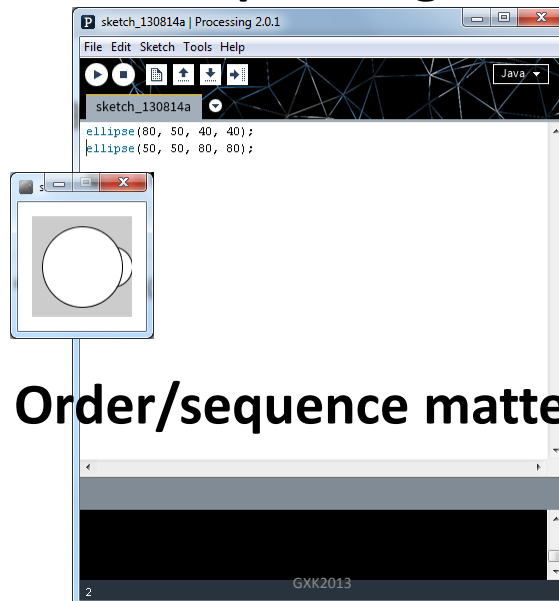
30

What happens if you switch?



31

Sequencing...

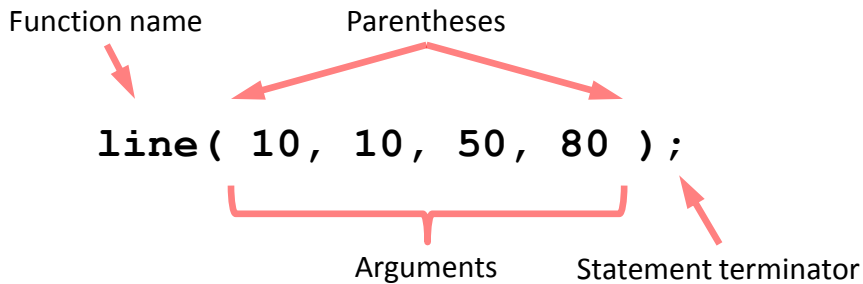


Order/sequence matters!

32

Programming Principle#2

- **Syntax is important!**



GXX2013

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



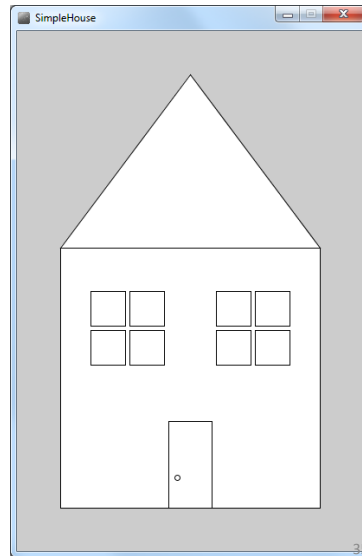
GXX2013

34

CS Principle: Algorithms

Draw a simple house

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



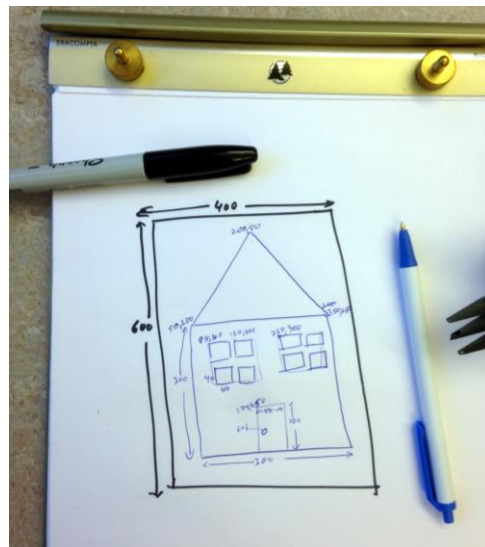
GXX2013

35

Algorithms to Pseudocode

Draw a simple house

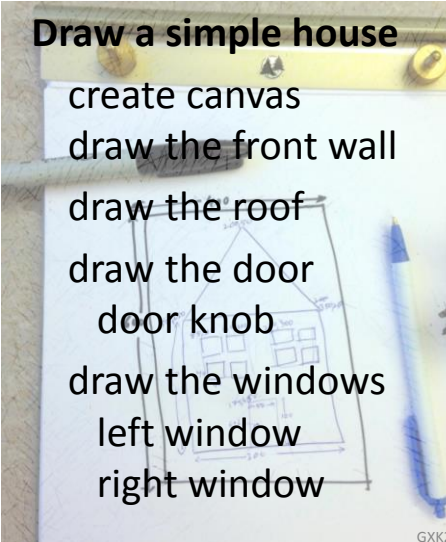
- create canvas
- draw the front wall
- draw the roof
- draw the door
- door knob
- draw the windows
- left window
- right window



GXX2013

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(50, 250, 300, 300);

// roof
triangle(50, 250, 350, 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**

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!

GXX2013

39

Drawing Basics





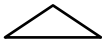




- **Canvas – computer screen**
`size(width, height);`
- **Drawing Tools – shape commands**
- **Colors – grayscale or RGB**
`background(125);`



GXX2013

40


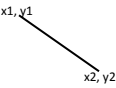
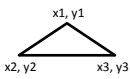
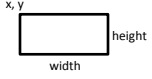
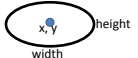
Drawing Tools - Basic Shapes

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

GXX2013

41

Drawing Tools - Basic Shapes

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

GXX2013

42



Color

- Grayscale (0..255)



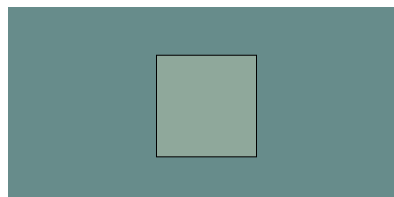
- RGB – red, green, blue
0..255, 0..255, 0..255

GXX2013 43

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);
```

GXX2013

44

Color Transparency

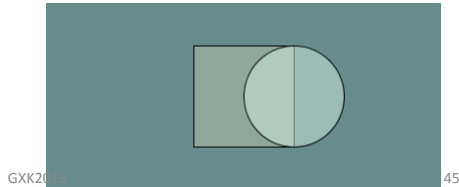
- Alpha values (0..255) specify transparency/opacity

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(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?

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

Structure	Shape	Color	Setting
<code>()</code> (parentheses)	<code>createShape()</code>		<code>background()</code>
<code>,</code> (comma)	<code>loadShape()</code>		<code>clear()</code>
<code>.</code> (dot)	<code>PShape</code>		<code>colorMode()</code>
<code>/**</code> (multiline comment)			<code>fill()</code>
<code>/**</code> (doc comment)	2D Primitives		<code>noFill()</code>
<code>//</code> (comment)	<code>arc()</code>		<code>noStroke()</code>
<code>;</code> (semicolon)	<code>ellipse()</code>		<code>stroke()</code>
<code>=</code> (assign)	<code>line()</code>		
<code>[]</code> (array access)	<code>point()</code>		
<code>{}</code> (curly braces)	<code>quad()</code>		Creating & Reading
<code>catch</code>	<code>rect()</code>		<code>alpha()</code>
<code>class</code>	<code>triangle()</code>		<code>blue()</code>
<code>draw()</code>			<code>brightness()</code>
<code>exit()</code>	Curves		<code>color()</code>
<code>extends</code>	<code>bezier()</code>		<code>green()</code>
<code>false</code>	<code>bezierDetail()</code>		<code>hue()</code>
<code>final</code>	<code>bezierPoint()</code>		<code>lerpColor()</code>
<code>implements</code>	<code>bezierTangent()</code>		<code>red()</code>
<code>import</code>	<code>curve()</code>		<code>saturation()</code>
<code>loop()</code>	<code>curveDetail()</code>		
<code>new</code>	<code>curvePoint()</code>		
<code>noLoop()</code>	<code>curveTangent()</code>		
<code>null</code>	<code>curveTightness()</code>		
			Image
			<code>createImage()</code>

47