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 review the syllabus, then get started...

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.
Homework

- Go the CS Computer Lab (Room 231 Park)
  - Ask me for the code now.
- 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/](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 Ch. 1 (pgs 1-12, skim 12-32) and Ch. 2, pgs. 33-48
Drawing Basics

• Canvas

• Drawing Tools

• Colors

Canvas – Computer Screen

• Pixels

Canvass - Computer Screen

• Coordinate System

Canvas – computer screen

• Drawing Tools – shape commands

• Colors – grayscale or RGB
Canvas - Computer Screen

Processing Commands

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

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

Canvas - Computer Screen

Processing Commands

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

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

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

```java
background(125);
```

256 Shades of Gray!

- 0 = black
- 255 = white

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10000000</td>
<td>128</td>
<td>80</td>
</tr>
<tr>
<td>11000000</td>
<td>192</td>
<td>C0</td>
</tr>
<tr>
<td>11100000</td>
<td>224</td>
<td>E0</td>
</tr>
<tr>
<td>11110000</td>
<td>240</td>
<td>F0</td>
</tr>
<tr>
<td>00001000</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>00001100</td>
<td>12</td>
<td>C</td>
</tr>
<tr>
<td>00001110</td>
<td>14</td>
<td>E</td>
</tr>
<tr>
<td>00001111</td>
<td>15</td>
<td>F</td>
</tr>
</tbody>
</table>

128 \times 1 = 128
64 \times 1 = 64
32 \times 1 = 32
16 \times 1 = 16
8 \times 1 = 8
4 \times 0 = 0
2 \times 0 = 0
1 \times 1 = +1
\frac{249}{249}
\frac{249}{249}
\frac{249}{249}
Drawing Basics

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

- **Drawing Tools** – shape commands

- **Colors** – grayscale or RGB
t  background(125);

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);

- **Arc**

- **Quad**

- **Polygon**

- **Curve**

Drawing & Shape Attributes

- **Anti-aliasing**
  - smooth();
  - noSmooth();

- **Stroke**
  - noStroke();
  - strokeWeight(<pixel width>);
  - stroke(<stroke color>);

- **Fill**
  - noFill();
  - fill(<fill color>);
Antialiasing

- smooth(); vs noSmooth();

Fill Attributes

- fill(100); vs noFill();

Stroke Attributes

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

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**
  
  \[ \text{point}(x, y); \]

- **Line**
  
  \[ \text{line}(x_1, y_1, x_2, y_2); \]

- **Triangle**
  
  \[ \text{triangle}(x_1, y_1, x_2, y_2, x_3, y_3); \]

- **Rectangle**
  
  \[ \text{rect}(x, y, \text{width}, \text{height}); \]

- **Ellipse**
  
  \[ \text{ellipse}(x, y, \text{width}, \text{height}); \]

**Modes**

- **rect** \((x, y, \text{width}, \text{height})\)

- **ellipse** \((x, y, \text{width}, \text{height})\)

- **rectMode** \(\text{CENTER}\)

- **ellipseMode** \(\text{CORNER}\)

- **Also CORNERS (see Reference)**

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

**Structure of a basic program**

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

size(400, 600);
smooth();

// house
\text{rect}(50, 250, 300, 300);

// roof
\text{triangle}(50, 250, 350, 250, 200, 50);

// door
\text{rect}(175, 450, 50, 100);

// door knob
\text{ellipse}(185, 515, 6, 6);

// left windows
\text{rect}(85, 300, 40, 40);
\text{rect}(130, 300, 40, 40);
\text{rect}(85, 345, 40, 40);
\text{rect}(130, 345, 40, 40);

// right windows
\text{rect}(230, 300, 40, 40);
\text{rect}(275, 300, 40, 40);
\text{rect}(230, 345, 40, 40);
\text{rect}(275, 345, 40, 40);
```

**Programming Principle#1**

- **Sequencing**

  do this
  and this
  and this
  and this ...

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

What happens if you switch?

Sequencing...

What happens if you switch?
Order/sequence matters!

Programming Principle#2

• Syntax is important!

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

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

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

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**
  - `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
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:

```
// Fill with alpha value
fill(208, 237, 222, 127);
ellipse(250, 100, 100, 100);
```

Why 0 .. 255?

Referenced on: http://processing.org/reference/