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...

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 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)

Processing 2.0 IDE
First Processing Program

First Processing Program

Drawing Basics

- Canvas
- Drawing Tools
- Colors

Canvas – Computer Screen

- Pixels

Canvas - Computer Screen

- Coordinate System
  - $(0, 0)$
  - $+x$
  - $+y$
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

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);
  ```
Drawing & Shape Attributes

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

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

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

Antialiasing

- smooth(); vs noSmooth();

Stroke Attributes

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

Fill Attributes

- fill(100); vs noFill();

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
// 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
  ... 
  ```

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

**Sequencing...**

```
// 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);
```

**Sequencing...**

```
// 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);
```

**What happens if you switch?**
What happens if you switch?

Sequencing...

Order/sequence matters!

Programming Principle#2

• Syntax is important!

Function name Parentheses
line( 10, 10, 50, 80 );
Arguments Statement terminator

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
left window
to 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**
- **Line**
- **Triangle**
- **Rectangle**
- **Ellipse**
- **Arc**
- **Quad**
- **Polygon**
- **Curve**

**Drawing Tools - Basic Shapes**

- **Point**
  point(x);
- **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

Grayscale examples:
- background(<grayscale value>);
- stroke(<grayscale value>);
- fill(<grayscale value>);

RGB examples:
- background(R, G, B);
- stroke(R, G, B);
- 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);
  stroke(<grayscale value>, ALPHA);
  fill(<grayscale value>, ALPHA);

Example:
- background(103,140,139);
- fill(143,168,155);
- rect(150,50,100,100);
- fill(208,237,222,127);
- ellipse(250,100,100,100);

Why 0..255?

Why 0..255?