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
Drawing Basics

- Canvas
- Colors
- Drawing Tools

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

• Pixels

Canvas - Computer Screen

• Coordinate System

(0, 0)
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

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:

background(103, 140, 139);
fill(143, 168, 155);
// 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

```cpp
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 Diagram]
  
  point(x, y);

- **Line**
  
  ![Line Diagram]
  
  line(x1, y1, x2, y2);

- **Triangle**
  
  ![Triangle Diagram]
  
  triangle(x1, y1, x2, y2, x3, y3);

- **Rectangle**
  
  ![Rectangle Diagram]
  
  rect(x, y, width, height);

- **Ellipse**
  
  ![Ellipse Diagram]
  
  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)
Programming Principle#1

- Sequencing

  do this
  and this
  and this
  ...

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

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

Sequencing...

Order/sequence matters!
Programming Principle#2

- Syntax is important!

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

Function name Parentheses
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
door knob
draw the windows
left window
right window

```javascript
// 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
door(175, 450, 50, 100);

// door knob
e11(185, 515, 6, 6);

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

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

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

- **Drawing Tools** – **shape commands**
Drawing Tools - Basic Shapes

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

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: \( \text{ellipse}(x, y, \text{width}, \text{height}); \)
**Reference.** The Processing Language was designed to facilitate the creation of sophisticated visual structures.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Shape</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>[] (array)</td>
<td>rect()</td>
<td>Fill</td>
</tr>
<tr>
<td>[] (array)</td>
<td>rect()</td>
<td>Stroke</td>
</tr>
<tr>
<td></td>
<td>ellipse()</td>
<td>Stroke</td>
</tr>
<tr>
<td></td>
<td>line()</td>
<td>Stroke</td>
</tr>
<tr>
<td></td>
<td>circle()</td>
<td>Stroke</td>
</tr>
<tr>
<td></td>
<td>triangle</td>
<td>Stroke</td>
</tr>
<tr>
<td></td>
<td>text()</td>
<td>Stroke</td>
</tr>
</tbody>
</table>

- **Curves**
  - bezier(x1, y1, x2, y2)
  - bezier(x1, y1, x2, y2, x3, y3, x4, y4)
  - curve(x1, y1, x2, y2)
  - curve(x1, y1, x2, y2, x3, y3, x4, y4)
  - curveTo(x1, y1, x2, y2)

- **Images**
  - createImage()
  - image(x, y, w, h)
  - loadImage()