Functions

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

Review: Drawing Basics

- **Canvas**
  - `size(width, height)`

- **Drawing Tools**
  - `point(x, y)`
  - `line(x1, y1, x2, y2)`
  - `triangle(x1, y1, x2, y2, x3, y3)`
  - `rect(x, y, width, height)`
  - `ellipse(x, y, width, height)`
  - `arc(x, y, width, height, startAngle, endAngle)`
  - `curve(cpx1, cpy1, x1, y1, x2, y2, cpx2, cpy2)`

- **Function**
  - `beginShape()`, `endShape()` (CLOSE)
  - `vertex(x, y)`
  - `curveVertex(x, y)`

- **Colors**
  - `grayscale[0..255]`, `RGB[0..255], [0..255], [0..255]`, `alpha[0..255]`
  - `background(color)`

- **Drawing & Shape Attributes**
  - `smooth()`, `noSmooth()`
  - `stroke(color)`, `noStroke()`, `strokeWeight(pixelWidth)`
  - `fill(color)`, `noFill()`

- **Processing: Predefined Variables**
  - `width, height`
    - The width & height of the canvas used in the sketch
  - `PI, HALF_PI, TWO_PI`
    - For different values of $\pi$. Note that
    - $HALF PI = PI/2$
    - $TWO PI = 2*PI$
  - `displayWidth, displayHeight`
    - The width and height of the monitor being used. This is useful in running fullscreen sketches using:
    - `size(displayWidth, displayHeight)`
  - `mouseX, mouseY`
    - The current mouse location in sketch (...coming soon!)

Simple Program Structure

```
// Create and set canvas
size(width, height);
smooth();
background(color);

// Draw something
--
// Draw something else
--
// etc.
```

```
// Draw a simple house
// Create and set canvas
size(300, 300);
smooth();
background(187, 193, 127);

// Wall
fill(206, 224, 14);
rect(50, 150, 200, 100);

// Draw Door
fill(72, 26, 2);
rect(125, 200, 50, 50);

// Draw roof
fill(224, 14, 14);
triangle(50, 150, 150, 50, 250, 150);
```

Variables: Naming Values

- **Values**
  - 42, 3.14159, 2013, "Hi, my name is Joe!", true, false, etc.

  - **Integers**
    - `int meaningOfLife = 42;`
    - `int year = 2013;`

  - **Floating point numbers**
    - `float pi = 3.14159;`

  - **Strings**
    - `String greeting = "Hi, my name is Joe!";`

  - **Boolean**
    - `boolean keyPressed = true;`
Program Structure: Dynamic Mode

Most Processing programs we will write will have the following structure:

```java
<Declare variables>
void setup() {
  <initial canvas set up goes here>
} // setup()
void draw() {
  <drawing stuff goes here>
} // draw()
```

Processing: Dynamic Sketches

```java
// Draw a simple house
void setup() {
  size(300, 300); smooth();
  background(187, 193, 127);
} // setup()
void draw() {
  // wall
  fill(206, 224, 14);
  rect(50, 150, 200, 100);
  // Draw Door
  fill(72, 26, 2);
  rect(125, 200, 50, 50);
  // Draw roof
  fill(224, 14, 14);
  triangle(50, 150, 150, 50, 250, 150);
} // draw()
```

But...
What are these???
For now...
Necessary syntax
More later...

Something More Interesting...

```java
color color1 = color(227, 220, 0);
color color2 = color(37, 220, 0);
color color3 = color(0);
void setup() {
  // create and set canvas
  size(300, 300);
  smooth();
  background(color1);
} // setup()
void draw() {
  stroke(color3);
  fill(color2);
  ellipse(mouseX, mouseY, 40, 40);
} // draw()
```
Predefined variables:

- pmouseX
- pmouseY

```
color color1 = color(227, 220, 0);
color color2 = color(37, 220, 0);
color color3 = color(0);
```

```void setup() {
    // create and set canvas
    size(300, 300);
    smooth();
    background(color1);
} // setup()
```

```void draw() {
    stroke(color2);
    strokeWeight(5);
    line(pmouseX, pmouseY, mouseX, mouseY);
} // draw()
```

Events: More Interactivity

```
circles are drawn
ONLY when mouse is pressed.
```

```
void mousePressed() {
    stroke(color3);
    fill(color2);
    ellipse(mouseX, mouseY, 40, 40);
} // mousePressed()
```

Something More Interesting...

```
void setup() {
    // create and set canvas
    size(300, 300);
    smooth();
    background(color1);
} // setup()
```

```
void draw() {
    stroke(color3);
    fill(color2);
    ellipse(mouseX, mouseY, 40, 40);
} // draw()
```

Controlling Frame Rate

```
frameRate(N);  // Changes frame rate to N times/second

<Declare variables>
void setup() {
    frameRate(30);
} // setup()

void draw() {
    // drawing stuff goes here
} // draw()
```

Mathematical Functions

```
y = f(x)
y = twice(x) = 2x
y = f(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}
y = \sum_{i=1}^{n} i
```
Functions: Terminology

\[ y = \text{twice}(x) = 2x \]

Function application:

\[ y = \text{twice}(5) \]

\[ y = 10 \]

Processing: Defining Functions

\[ y = \text{twice}(x) = 2x \]

float twice(float x) {
    \[ \text{return } 2x; \]
} // twice()

Processing: Defining Functions

Syntax:

\[ \text{returnType functionName(parameters)} \{ \text{return expression}; \} \]

Example:

float twice(float x) {
    return 2*x;
} // twice()

Use:

\[ y = \text{twice}(5); \]

Defining Functions: void

Use void as returnType when no value is returned.

Syntax:

\[ \text{returnType functionName(parameters)} \{ \text{return expression}; \} \]

Example:

void square(float x, float y, float side) {
    rectMode(CORNER);
    rect(x, y, side, side);
} // square()

Use:

\[ \text{square}(50, 50, 100); // draws a 100x100 square at 50, 50 \]

Program Structure: Functions

Variables & Scope

Global Variables
Either pre-defined or defined at top
Are visible everywhere in the program
Processing: Math Functions

• Math functions return values:
  Example:
  ```java
  void square(float x, float y, float side) {
    rectMode(CORNER);
    rect(x, y, side, side);
  } // square()
  ```

Use:
  ```java
  square(50, 50, 100); // draws a 100x100 square at 50, 50
  ```

• Processing has several pre-defined Math functions for calculation, trigonometry, and random number generation

Processing: Math Functions

• Math functions return values:
  Example:
  ```java
  void square(float x, float y, float side) {
    rectMode(CORNER);
    rect(x, y, side, side);
  } // square()
  ```

Use:
  ```java
  square(50, 50, 100); // draws a 100x100 square at 50, 50
  ```

• Processing has several pre-defined Math functions for calculation, trigonometry, and random number generation

Variables & Scope

- Local Variables
  - Either parameters or defined inside blocks
  - Are visible ONLY in the block
  - After they are defined

House() function example

```java
void house(int x, int y, int width, int height) {
  // House
  int houseX = x;
  int houseY = y;
  int houseWidth = width;
  int houseHeight = height;
  int doorWidth = (width * 1/4);
  int doorHeight = (height / 2); // height of wall is 1/2 of house height
  int wallHeight = (width * 1/4);
  // Create and set up canvas
  void setup() {
    background(187, 193, 127);
    smooth();
    size(300, 300);
  }
  // Draw a simple house
  void draw() {
    // square()
    square(50, 50, 100); // draws a 100x100 square at 50, 50
  }
}
```
Processing: Pre-defined Math Functions

- **Calculation**
  - abs(), ceil(), constrain(), dist(), exp(), floor(), lerp()
  - log(), mag(), map(), max(), min(), norm(), pow()
  - round(), sq(), sort()

- **Trigonometry**
  - acos(), asin(), atan(), atan2(), cos(), degrees(),
  - radians(), sin(), tan()

- **Random**
  - noise(), noiseDetail(), noiseSeed(), random(),
  - randomGaussian(), randomSeed()

Math Functions: Examples

- **Calculation**
  - float x, y;
  - y = 42;
  - x = sqrt(y);

- **Trigonometry**
  - float rad = radians(180);
  - float deg = degrees(PI/4);

- **Random**
  - float x = random(10);
    // returns a random number [0.0..10.0)
  - float y = random(1, 6);
    // returns a random number [1.0, 6.0)

Example: Using random()

```java
void setup() { // Create and set canvas
  size(300, 300);
  smooth();
  background(255);
} // setup()

void draw() {
  stroke(0);
  rect(random(width),
       random(height),
       random(5, 20),
       random(5, 20));
} // draw();
```

2D Transformations: Translate

- rect(20, 20, 40, 40);
- rect(20+60, 20+80, 40, 40);
Preserving Context

• translate() will change the coordinate system for the entire duration of the draw() cycle. It resets at each cycle.

• Use pushMatrix() and popMatrix() to preserve context during a draw() cycle. i.e.

  pushMatrix();
  translate(<x>, <y>);
  // Do something in the new coordinate context
  popMatrix();

Example: House() again!

```cpp
void setup() {
  size(300, 300);
  smooth();
  background(187, 193, 127);
} // setup()

void draw() {
  // Draw a house at 50, 250 in 200x200 pixels
  house(50, 250, 200, 200);
} // draw()

void house(int houseX, int houseY, int houseWidth, int houseHeight) {
  int wallHeight = houseHeight / 2;  // height of wall is 1/2 of house height
  int roofHeight = houseHeight / 2;
  int doorHeight = houseHeight / 4;
  int doorWidth = houseWidth / 4;
  pushMatrix();
  translate(houseX, houseY);
  // wall
  fill(206, 224, 14);
  rect(0, -wallHeight, houseWidth, wallHeight);
  // Draw Door
  fill(72, 26, 2);
  rect(houseWidth / 2 - doorWidth / 2, -doorHeight, doorWidth, doorHeight);
  // Draw roof
  fill(224, 14, 14);
  triangle(0, -wallHeight, houseWidth / 2, -houseHeight, houseWidth, -wallHeight);
  popMatrix();
} // house()
```

2D Transformations: Rotate

```cpp
void setup() {
  size(200, 200);
  background(255);
  smooth();
  fill(192);
  noStroke();
  rect(40, 40, 40, 40);
  pushMatrix();
  rotate(radians(45));
  fill(0);
  rect(40, 40, 40, 40);
  popMatrix();
} // setup()
```

2D Transformations: Scaling

```cpp
void setup() {
  size(200, 200);
  background(255);
  stroke(128);
  rect(20, 20, 40, 40);
  stroke(0);
  pushMatrix();
  scale(2.0);
  rect(20, 20, 40, 40);
  popMatrix();
} // setup()
```