Carl Kingsford, 02-201, Fall 2015

Lecture 23: Processing

Processing is a system that lets you sketch drawings, data visualizations, and animations quickly and fairly easily. With it, people have created cool art installations, neat ways of viewing data, and other applications.

Processing programs are written in the "Processing" program, which has a text editor built in. You write Java programs in this window and can run your programs directly from Processing. You can also export your program to an executable that can run on Windows, Mac OS, or Linux.

Processing's version of Java is nearly the same as normal Java, but with a few simplifications.

Processing is well documented, with a lot of tutorials online: https://processing.org/tutorials/. So I'll only highlight some important notes below.

Basic structure of a Processing program

Unlike plain Java, you don't have to create a class that contains your drawing function. Processing also doesn't use the standard Java `main` function. Instead, you create two functions:

```
void setup() {
    // this is run once at the start of your program
    // use this function to set up various parameters
    // (size, background color, etc.)
}

void draw() {
    // this function is called constantly, at every "frame" of the
    // animation. you can think of it as if Processing has a loop in it:
    //  for { draw(); }
}
```

Drawing commands

Processing has a lot of built-in drawing functions. The most important of which are:

- `point(x,y)` -- draw a point
- `line(x1,y1,x2,y2)` -- draw a line
- `rect(x1, y1, width, height)` -- draw a rectangle
- `ellipse(x, y, width, height)` -- draw an ellipse (or circle)
- `text(s, x, y)` -- draw text `s`

Coordinate system: (0,0) is at the top left corner of the screen. The y-coordinate is up-down and the x is left-right. The point (100, 100) would be to the right and below (0,0).

Processing also has functions that change the current line color, line width, and fill color, etc.:

- `stroke(color)` -- change the line color
- `noStroke()` -- don't draw lines
- `strokeWidth(x)` -- the line width
- `background(color)` -- set the background color
- `fill(color)` -- set the fill color
- `textSize(x)` -- set the text size

If you call one of these functions, it will change how all subsequent drawing commands behave.

Built in Global Variables

Processing has some built-in variables you can use anywhere in your program:

- `width` -- the width of the canvas
- `height` -- the height of the canvas
- `mouseX` -- the X position of the mouse
- `mouseY` -- the Y position of the mouse
- `mousePressed` -- true if the mouse is pressed
- `mouseButton` -- which button was pressed
- `keyPressed` -- true if a key is pressed
- `key` -- which key is pressed

Other functions that are automatically called

In addition to `setup()` and `draw()` you can write other functions that are called when certain things happen:

- `mousePressed()` -- called when a mouse button is pressed
- `mouseReleased()` -- called when a mouse button is released

- `mouseMoved()` -- called when the mouse is moved
- `mouseDragged()` -- called when the mouse is moved with a button down
- `keyPressed()` -- called when a key is pressed
- `keyReleased()` --- called when a key is released

Other useful functions

- `size(width, height)` --- set the size of the canvas; must be the first thing called in `setup()` if you call it
- `random(x)` -- return a random number between 0 and x.
- `saveFrame(filename)` -- write the current picture to the given file
- `loadStrings(filename)` -- return a `String[]` where the ith element is the ith line in the file.

Example

Here's a quickly written example that shows how you can animate a moving ball that you can control with the WADS keys.

```

int iteration = 0;

int deltax, deltay;

Ball[] currentBalls = new Ball[100];
int numBalls = 0;

class Ball {
   int x, y;

   /* constructor */
   Ball(int xx, int yy) {
       x = xx;
       y = yy;
   }
}

void setup() {
  size(500,500);
  fill(255,0,0,127);

  // set up the ball
  Ball b = new Ball(0,0);
  currentBalls[0] = b;
  numBalls = 1;

  // current movement:
  deltax = 1;
  deltay = 1;
}

void draw() {
  iteration++;
  background(0,0,iteration % 255);

  for (int i =0; i< numBalls; i++) {
    Ball ball = currentBalls[i];

    ellipse(abs(ball.x % width),abs(ball.y % height),30,30);

    ball.x += deltax;
    ball.y += deltay;
  }
}


void keyPressed() {
  System.out.println(key);
   switch(key) {
     case 'w': deltay = -1; break;
     case 'a': deltax = -1; break;
     case 'd': deltax = +1; break;
     case 's': deltay = +1; break;
   }
}

void mousePressed() {

  Ball b = new Ball(mouseX, mouseY);
  currentBalls[numBalls] = b;
  numBalls++;
}
```

Summary

Processing is a system built on Java that lets you draw easily.

The syntax is the same as Java, except that there are some additional built-in functions, and you don't have to define a main class or `main` function.