

## Processing

What is Processing?

From <http://www.processing.org>: Processing is an open source programming language and environment for people who want to program images, animation, and interactions. It is used by students, artists, designers, researchers, and hobbyists for learning, prototyping, and production. It is created to teach fundamentals of computer programming within a visual context and to serve as a software sketchbook and professional production tool. Processing is developed by artists and designers as an alternative to proprietary software tools in the same domain.

Please read pp. 1–99 in “Processing – A Programming Handbook for Visual Designers and Artists” by Casey Reas and Ben Fry for a basic introduction to the software and its functions.

Mark a word in the Processing code and ctrl click (Macintosh) on it to display information about it in the Processing reference.

Go to:

File > Examples > Topics > Drawing > Continuous Lines

```
/**
 * Continuous Lines.
 *
 * Click and drag the mouse to draw a line.
 */

void setup() {
  size(200, 200);
  background(102);
}

void draw() {
  stroke(255);
  if(mousePressed) {
    line(mouseX, mouseY, pmouseX, pmouseY);
  }
}
```

Where can I find my files?

Each sketch resides in its own folder where the main program file is located with the ending “.pde”. You can browse to this folder by choosing Sketch > Show Sketch Folder from the Processing menu.

Example:

Sketch name: “Sketch\_01”, the directory for the sketch will be called “Sketch\_01”, the main file will be called “Sketch\_01.pde”.

Go to:  
File > Examples > Topics > Drawing > Patterns

```
/**
 * Patterns.
 *
 * Move the cursor over the image to draw with a software tool
 * which responds to the speed of the mouse.
 */

void setup()
{
  size(200, 200);
  background(102);
  smooth();
}

void draw()
{
  // Call the variableEllipse() method and send it the
  // parameters for the current mouse position
  // and the previous mouse position
  variableEllipse(mouseX, mouseY, pmouseX, pmouseY);
}

// The simple method variableEllipse() was created specifically
// for this program. It calculates the speed of the mouse
// and draws a small ellipse if the mouse is moving slowly
// and draws a large ellipse if the mouse is moving quickly

void variableEllipse(int x, int y, int px, int py)
{
  float speed = abs(x-px) + abs(y-py);
  stroke(speed);
  ellipse(x, y, speed, speed);
}
```

## Processing and the Arduino Board

Please read pp. 633–657 in “Processing – A Programming Handbook for Visual Designers and Artists” by Casey Reas and Ben Fry for an excellent introduction on how to connect the Arduino board with Processing and for code and circuit examples.

The following example illustrates how you can control 256 lines in a Processing window with a potentiometer connected to the Arduino board. If the potentiometer is at its minimum limit, the Processing sketch will draw just one line on the left hand side of the screen. The more the potentiometer’s knob is turned the more lines will be drawn on the screen until the whole screen is filled with 256 lines (potentiometer at its maximum limit). This example uses a `for` loop to iterate through a set of numbers ranging from 0 to the current number that the Arduino board sends from the potentiometer. This number determines the number of lines drawn on the screen and their distribution. We also use this number to gradually change the color of the lines drawn to the screen.

```

import processing.serial.*;

Serial port;
int val;

void setup() {
  size(1000, 500);
  //noStroke();
  frameRate(10);
  println(Serial.list());
  port = new Serial(this, Serial.list()[0], 9600);
  // choose the port that the Arduino is
  // connected to, on a Macintosh choose
  // the tty.usbserial port
}

void draw() {
  if (0 < port.available()) {
    val = port.read();
  }
  background(204);

  for (int i = 0; i < val+1; i++) {
    // this is the "for" loop, see pp.61
    // in the book "Processing" by Reas and Fry

stroke(255-i, 0, i); // choose the color of the line based on the
                    // potentiometer's value

line (i*5, 100, (i*5)+10, 400); // draw the lines based on the
                                // potentiometer's value
  }
  println(val);                // prints out values from Arduino board in
                              // the console, just to see what Processing
                              // actually receives from the Arduino board
}

```

The code for the Arduino board is the following:

```

int val;
int inputPin = 2;           // this is the pin to which we
                           // connect the potentiometer

void setup() {
  Serial. begin(9600);
}

void loop() {
  val = analogRead(inputPin)/4; // the potentiometer's range is from
                                // 0-1023 but we only need numbers from
                                // 0-255.

  Serial.print(val, BYTE);
  delay (100);
}

```

The circuit connected to the Arduino board is the simple potentiometer input circuit from previous examples and looks like this:

