

ProcessingSimpleExample

```
// drawLine -- our own function (we'll define these at the start of a file).
// "void" says drawLine returns no information (no variables)
// we "pass it" or give it four integer variables -- x0, y0, x1, y1
// x0, y0, x1, y1 exist temporarily as int variables inside the function
// to use as we wish
// Notice -- this is the simple (non-Bresenham) version to get started.
// Because of that, we'll use floats and round -- this is slow, though.
// Using only integer math would help us as in Bresenham's.

void drawLine(int x0,int y0,int x1,int y1)
{
  int x, y;           // declare any variables that will be used
  float m, b;        // and also declare by what type they are

  m = (y1-y0)/(x1-x0); // compute slope
  b = y0 - (m*x0);    // compute y-intercept

  for(x=x0; x<=x1; x++) // start a loop within { and } that proceeds
  {                       // at a start of x0 to and end of x1 step by 1
    y=round(m*x + b);    // good ol' line equation
    point(x,y);         // this is Processing's command for filling
                        // a pixel at a certain location.
                        // we'd directly write to a memory address
                        // in C, asm, etc. as discussed in class
                        // (y*row*bytesPerPixel)+(x*bytesPerPixel)
  }
}

// There are two main functions in basic Processing programs
// void setup() sets things up and returns "void" (or "nothing")
// void draw() is where all are drawing commands are and returns nothing
// ... later, we may use other update() functions or more complex
// setups for animation or other interactive programs

void setup() {
  size(400, 400);      // size of window
  background(0, 0, 0); // set background color
}

void draw() {
  stroke(255);         // set color to draw
  drawLine(150,150,200,200); // call our own function
}
}
```