Notes for Undergraduate Final Year project

There is a lot here on Processing (which I have played with a little, but the emphasis of the project is better to be on how you make an interface to navigate the `space' of different images - actually making more images is not in itself a Computer Science issue.

John Stell, October 2020

(most of the notes are from 2019)

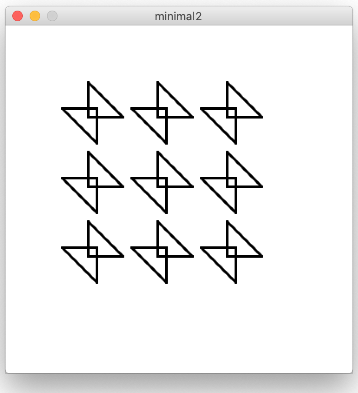
- Individual images can be created, but
- it is navigating the space of images that is challenging, and how to do this interactively
- there are many dimensions to this space:
  - position of elements (smaller shapes in image)
  - line thickness (stroke weight)
  - line density (black or shades of grey)
  - the small shapes themselves (even if number of sides is fixed they can vary, e.g. a square, a kite, a rectangle, a rhombus, etc)
  - randomness (and this might be several dimensions: randomness of placement of elements, randomness of (line thickness, density, colour, etc), randomness of vertices of an element with respect to a center, etc, etc, . . .
- Implementation: examples here are in Processing, but p5.js (`https://p5js.org/`) is closely related and might have advantages, especially in interaction
- For interactivity in processing see, for example:
  `https://processing.org/tutorials/interactivity/`
  `http://www.sojamo.de/libraries/controlP5/`

  And in p5.js: `https://p5js.org/examples/dom-slider.html`

  As part of the project, you can explore which way of adding interactivity is going to work

What does the existing code do?

What is needed to produce an image like this?



There are two main things here:

1. drawing the shape of two triangles
2. drawing nine of them in a grid

Drawing a single shape (including random variation)

Random points

To allow random variation in a single point the following class is defined

```
class RandomPoint {

  float xmin, xmax, ymin, ymax; // variation in position

     RandomPoint(){
                xmin = 0;
                xmax = 0;
                ymin = 0;
                ymax = 0;
              }

    ExactPoint location(){
      return new ExactPoint(random(xmin,xmax),random(ymin,ymax));
    }

    void setValues(float smallx, float bigx, float smally, float bigy){
                xmin = smallx;
                xmax = bigx;
                ymin = smally;
                ymax = bigy;
              }


    }
```

A random point is like a rectangular region within which an exact point can be located. The `location` method when called for a given Random Point will return different exact points each time it is called.

Exact Points

An exact point is just a two dimensional coordinate location.

```
class ExactPoint {

  float x, y; // x-coordinate, y-coordinate

  //Constructor
  ExactPoint(float xpos, float ypos){
      x = xpos;
      y = ypos;
  }

  //get the x coordinate
  float xVal(){
      return x;
  }

  //get the y coordinate
  float yVal(){
      return y;
  }

  //draw a line to another point
  void lineTo(ExactPoint otherPoint) {
      line(x,y,otherPoint.xVal(),otherPoint.yVal());
  }

}
```

The class consists of

- a constructor,
- methods to return the two coordinates, and
- a method to draw a line to another point

In earlier versions I had a function outside the `ExactPoint` class as follows

```
  void lineE(ExactPoint ep1, ExactPoint ep2) {
      line(ep1.xVal(),ep1.yVal(),ep2.xVal(),ep2.yVal());
  }
```

Random Shapes

```
class RandomShape {

  RandomPoint[] arrayOfPoints;

  boolean[][] theLines;
```

A Random Shape consists of an array of random points, together with a Boolean matrix specifying whether a line exists between each pair of points.

```
void drawShape(){
    int numberPoints = arrayOfPoints.length;
    ExactPoint[] exactPoints = new ExactPoint[numberPoints];

    //first pick an actual location for every one of the points

    for (int i = 0; i < numberPoints; i++){
      exactPoints[i] = arrayOfPoints[i].location();
    }

    //then draw all the lines in the shape between the chosen points

    for (int i = 0; i < numberPoints; i++){
      for (int j = 0; j < numberPoints; j++){
        if (theLines[i][j]) {
          exactPoints[i].lineTo(exactPoints[j]);
        }
      }
    }

  }
```

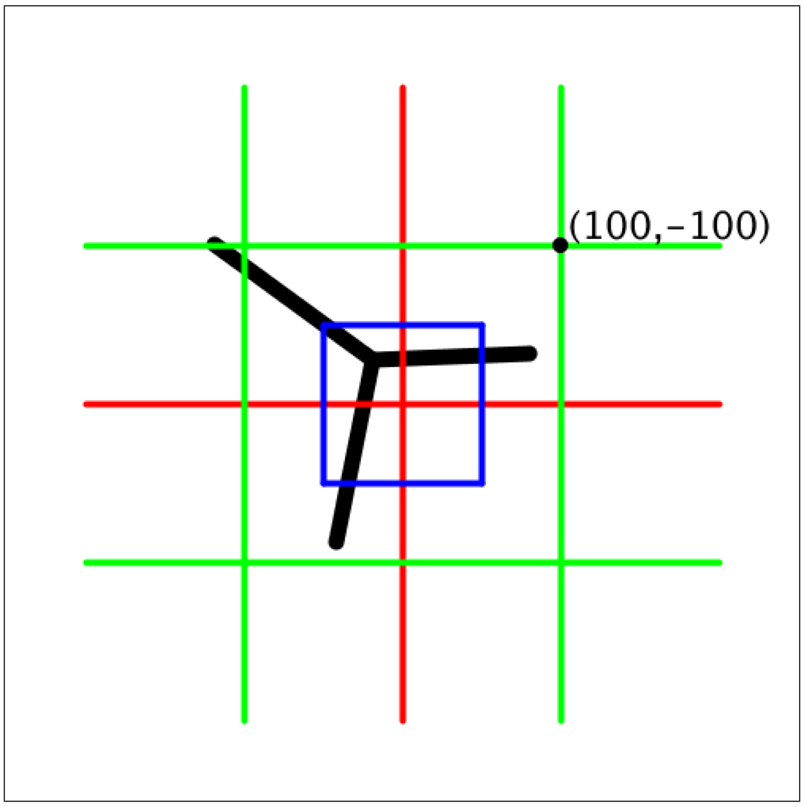
The `drawShape()` method for a random shape will

1. Pick an actual position for each point, using the location method of the point
2. Draw a line between every pair of chosen positions, for which the shape specifies a line

Only lines appear when a shape is drawn. Points that are not the endpoint of any line play no role.

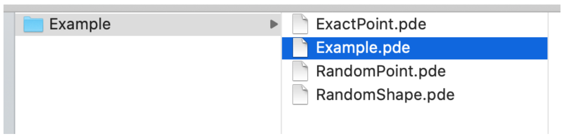
Example

To understand a basic example, I made this image



This shows a random shape with 4 points and 3 lines joining some of these points. I have added additional lines and text to show where the shape is in the coordinate system.

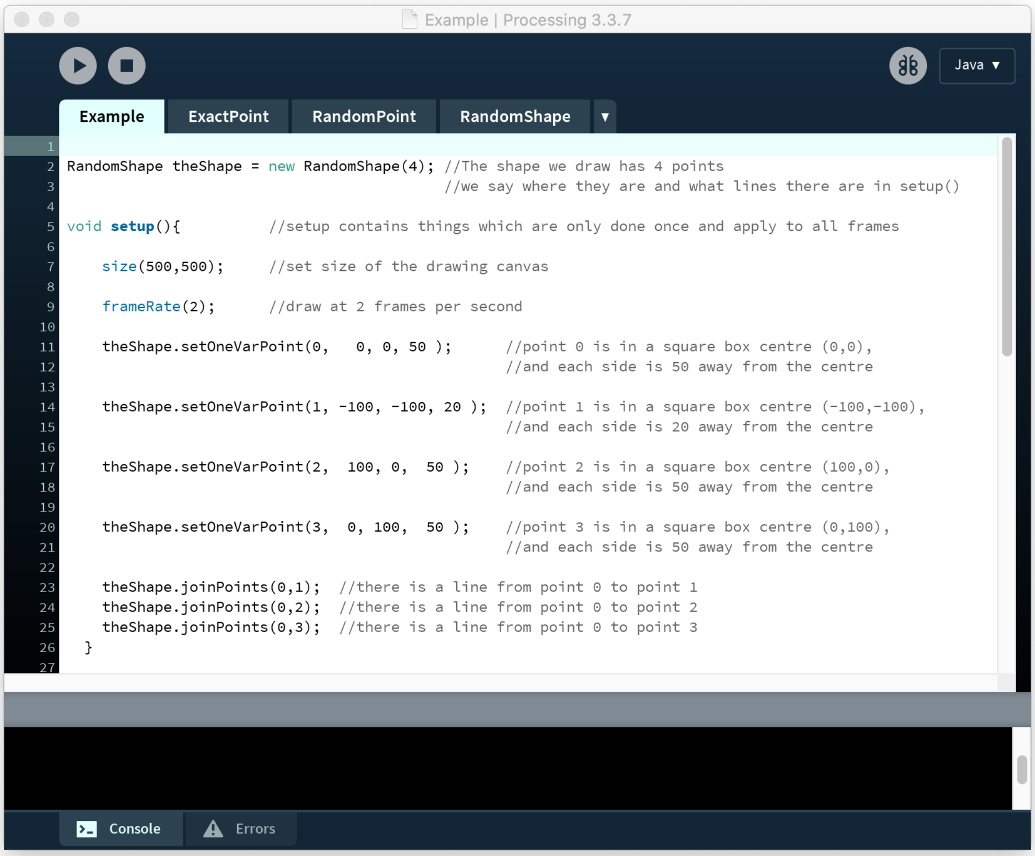
The code for this is in a folder called `Example`



The code consists of four files:

- The top level stuff: `Example.pde`
- Three files each for one of the classes:
  - `ExactPoint`
  - `RandomPoint`
  - `RandomShape`

If you open Example.pde in Processing you see this, with tabs for the three classes defined.



```

RandomShape theShape = new RandomShape(4); //The shape we draw has 4 points
                                           //we say where they are and what lines there are in setup()

void setup(){          //setup contains things which are only done once and apply to all frames

    size(500,500);     //set size of the drawing canvas

    frameRate(2);      //draw at 2 frames per second

    theShape.setOneVarPoint(0,   0, 0, 50 );      //point 0 is in a square box centre (0,0),
                                                  //and each side is 50 away from the centre

    theShape.setOneVarPoint(1, -100, -100, 20 );  //point 1 is in a square box centre (-100,-100),
                                                  //and each side is 20 away from the centre

    theShape.setOneVarPoint(2,  100, 0,   50 );   //point 2 is in a square box centre (100,0),
                                                  //and each side is 50 away from the centre

    theShape.setOneVarPoint(3,  0, 100,   50 );   //point 3 is in a square box centre (0,100),
                                                  //and each side is 50 away from the centre

    theShape.joinPoints(0,1);  //there is a line from point 0 to point 1
    theShape.joinPoints(0,2);  //there is a line from point 0 to point 2
    theShape.joinPoints(0,3);  //there is a line from point 0 to point 3
  }

```

I will list the code here to avoid any later problems with versions

Example.pde

```
RandomShape theShape = new RandomShape(4);
       //The shape we draw has 4 points
       //we say where they are and what lines there are in setup()

void setup(){
        //setup contains things which are only done once and apply to all frames

    size(500,500);     //set size of the drawing canvas

    frameRate(2);      //draw at 2 frames per second

    theShape.setOneVarPoint(0,   0, 0, 50 );
        //point 0 is in a square box centre (0,0),
        //and each side is 50 away from the centre

    theShape.setOneVarPoint(1, -100, -100, 20 );
        //point 1 is in a square box centre (-100,-100),
        //and each side is 20 away from the centre

    theShape.setOneVarPoint(2,  100, 0,  50 );
        //point 2 is in a square box centre (100,0),
        //and each side is 50 away from the centre

    theShape.setOneVarPoint(3,  0, 100,  50 );
        //point 3 is in a square box centre (0,100),
        //and each side is 50 away from the centre

    theShape.joinPoints(0,1);  //there is a line from point 0 to point 1
    theShape.joinPoints(0,2);  //there is a line from point 0 to point 2
    theShape.joinPoints(0,3);  //there is a line from point 0 to point 3
  }
```

The rest of `Example.pde` is on next page

`Example.pde` continued

```
void draw() {
   background(255);     //set image to white background;
                        //clears the previous frame instead of drawing on top

   translate(250,250);  //move origin to centre of image

   strokeWeight(10);     //set the line width to 10
   stroke(0);            //set darwing colour to black
   theShape.drawShape(); //draw the shape

   stroke(255,0,0);      //set draw colour to red
   strokeWeight(4);      //and set line thickness to 4
   line(-200,0,200,0);   //draw the horizontal axis
   line(0,-200,0,200);   //draw the vertical axis

   stroke(0,255,0);          //set draw colour to green
   line(100,-200,100,200);   //draw vertical green line, x = 100
   line(-100,-200,-100,200); //draw vertical greem line, x = -100
   line(-200,100,200,100);   //draw horizontal green line, y = 100
   line(-200,-100,200,-100); //draw horizontal green line, y = -100

   fill(0);                  //set fill colour for shapes to black
   noStroke();               //do not draw boundary of shapes
   ellipse(100,-100,10,10);  //draw a circular shape at (100, -100) radius 10

   textSize(24);                //set text size to 24
   text("(100,-100)",105,-105); //write string "(100,-100)" at position (105,-105)

   stroke(0,0,255);    //draw a blue box to show the places where point 0 can be drawn
   line(-50,50,50,50);
   line(50,-50,50,50);
   line(-50,-50,-50,50);
   line(-50,-50,50,-50);

   //noLoop();  //can use to stop re-drawing screen
}
```

ExactPoint.pde

```
class ExactPoint {

  float x, y; // x-coordinate, y-coordinate

  //Constructor
  ExactPoint(float xpos, float ypos){
      x = xpos;
      y = ypos;
  }

  //get the x coordinate
  float xVal(){
      return x;
  }

  //get the y coordinate
  float yVal(){
      return y;
  }

  //draw a line to another point
  void lineTo(ExactPoint otherPoint) {
      line(x,y,otherPoint.xVal(),otherPoint.yVal());
  }


}
```

RandomPoint.pde

```
class RandomPoint {

  float xmin, xmax, ymin, ymax; // variation in position

     RandomPoint(){
                xmin = 0;
                xmax = 0;
                ymin = 0;
                ymax = 0;
              }

    ExactPoint location(){
      return new ExactPoint(random(xmin,xmax),random(ymin,ymax));
    }

    void setValues(float smallx, float bigx, float smally, float bigy){
                xmin = smallx;
                xmax = bigx;
                ymin = smally;
                ymax = bigy;
              }


    }
```

RandomShape.pde

```
class RandomShape {

  RandomPoint[] arrayOfPoints;
  boolean[][] theLines;


  // Constructor
  // needs number of points; sets up no lines to start with;
  // points are not given positions here

  RandomShape(int n){
    arrayOfPoints = new RandomPoint[n];
    theLines = new boolean[n][n];
    for (int i = 0; i < n; i++){
      arrayOfPoints[i] = new RandomPoint();
      for (int j = 0; j < n; j++){
        theLines[i][j] = false;
      }
    }
  }

  // One way to set position of the n-th point
  // give centre of square box (x,y) and distance v of all sides from centre
  // Note: in general the constraint box need not be square.
  void setOneVarPoint(int n, float x, float y, float v){
    arrayOfPoints[n].setValues(x-v,x+v, y-v, y+v);
  }

  // Specifies that points m and n are joined by a line.
   void joinPoints(int m, int n){
    theLines[m][n] = true;
  }
```

The rest of `RandomShape.pde` is on next page

RandomShape.pde continued

```
  void drawShape(){
    int numberPoints = arrayOfPoints.length;
    ExactPoint[] exactPoints = new ExactPoint[numberPoints];

    //first pick an actual location for every one of the points

    for (int i = 0; i < numberPoints; i++){
      exactPoints[i] = arrayOfPoints[i].location();
    }

    //then draw all the lines in the shape between the chosen points

    for (int i = 0; i < numberPoints; i++){
      for (int j = 0; j < numberPoints; j++){
        if (theLines[i][j]) {
          exactPoints[i].lineTo(exactPoints[j]);
        }
      }
    }

  }


}
```

The Rest of this Document (added 28 June 2020)

Beware some of the things from here on refer to different versions of the code

The main thing that is different from the Example above is that there is a lot of mention of a function `mkCorners` which is used to set up a shape to be drawn.

```
void mkCorners(float var, int centreDisplace, int meetDisplace, int armSize){

    int centreX = 0;
    int centreY = 0;
    int upperRightX = armSize;
    int upperRightY = armSize;
    int lowerLeftX = -armSize;
    int lowerLeftY = -armSize;

    float variation = var;
    int xCentreDisplace = centreDisplace;
    int yCentreDisplace = centreDisplace;

    randCorners.setOneVarPoint(0, centreX, centreY,0);
    randCorners.setOneVarPoint(1, centreX - xCentreDisplace , centreY + yCentreDisplace, variation);
    randCorners.setOneVarPoint(2, centreX + xCentreDisplace , centreY - yCentreDisplace, variation);


    randCorners.setOneVarPoint(3, upperRightX, upperRightY, 0);
    randCorners.setOneVarPoint(4, centreX + xCentreDisplace, upperRightY, variation);
    randCorners.setOneVarPoint(5, upperRightX, centreY + yCentreDisplace, variation);


    randCorners.setOneVarPoint(6, lowerLeftX, lowerLeftY, 0);
    randCorners.setOneVarPoint(7, lowerLeftX, centreY - yCentreDisplace, variation);
    randCorners.setOneVarPoint(8, centreX - xCentreDisplace, lowerLeftY, variation);

    randCorners.joinPoints(1,5);
    randCorners.joinPoints(1,8);
    randCorners.joinPoints(2,4);
    randCorners.joinPoints(2,7);

    randCorners.joinPoints(5,8);
    randCorners.joinPoints(4,7);


}
```

Rather unhelpfully, the shape to be drawn is here called `randCorners`. In the Example, it is called `theShape`.

`mkCorners` was used to make it easy to vary some aspects of the shape by the parameters but to have a given number of points joined up in a specific way.

Much of what follows is documenting experiments with varying these parameters and seeing what happens

I was also concerned with getting output in pdf of images of a number of frames in which the parameters are altered by which frame is being drawn. Arranging all the frames in a single pdf was not straightforward.

From here on is from November 2019, for Berwick Sound and Space event

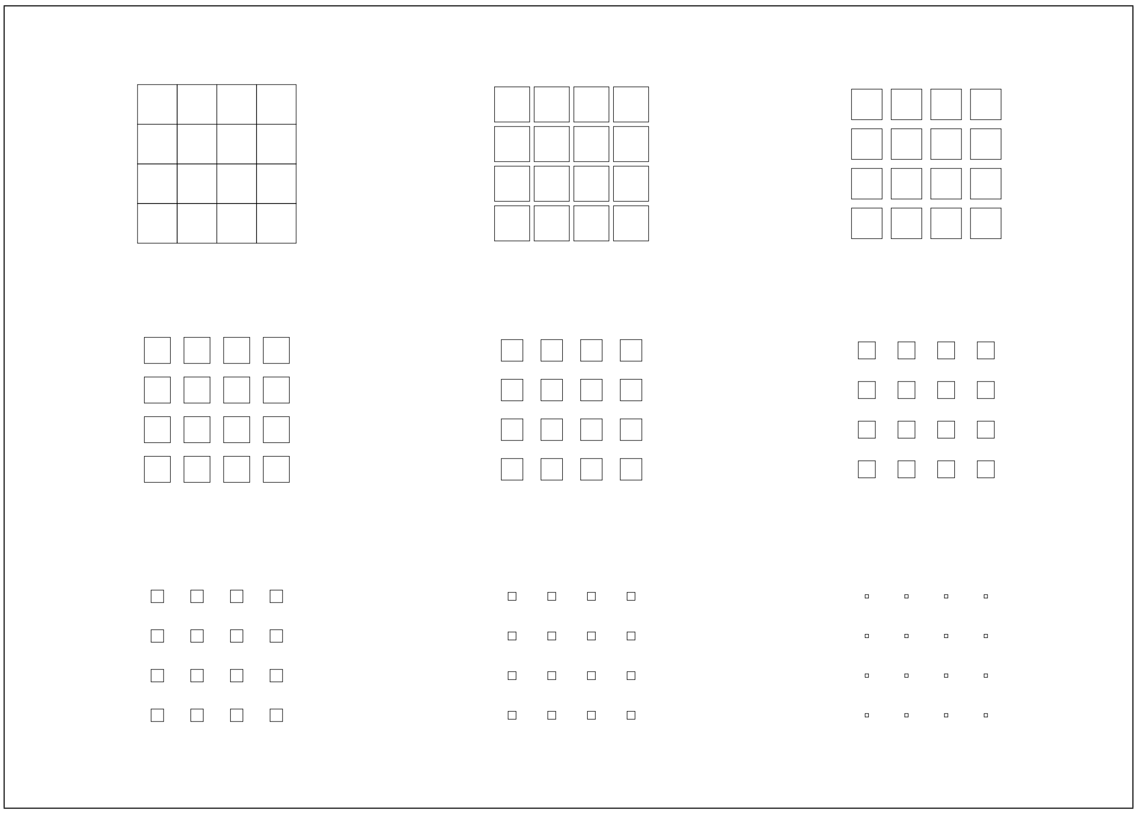
Creating pdf from processing

```
void setup(){

    //various set up stuff
    //including size(400,400)

  pdf = (PGraphicsPDF) createGraphics(width, height, PDF, "PdfTest.pdf");
  beginRecord(pdf);
  }

void draw() {

   //drawing stuff

  if (frameCount != 10) {
  // next page
  pdf.nextPage();
  } else {
  // finish
  endRecord();
  exit();
  }
}
```

J2

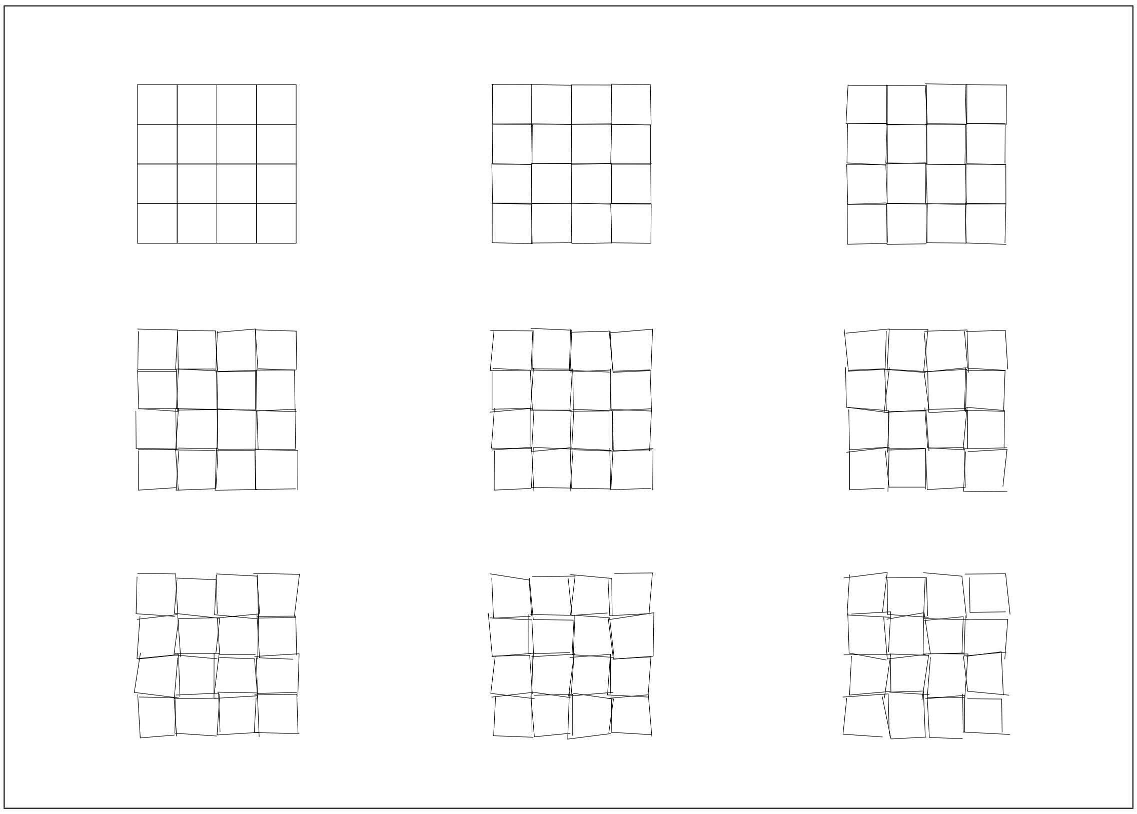
I generated file J2.pdf thus: in `draw()`

```
background(255);
translate(100,100);
mkCorners(0,39-(4*frameCount),0,39-(4*frameCount));
drawGridOfShapes(shapeToDraw,4,70);
```



J1

Here is J1



How was that produced?

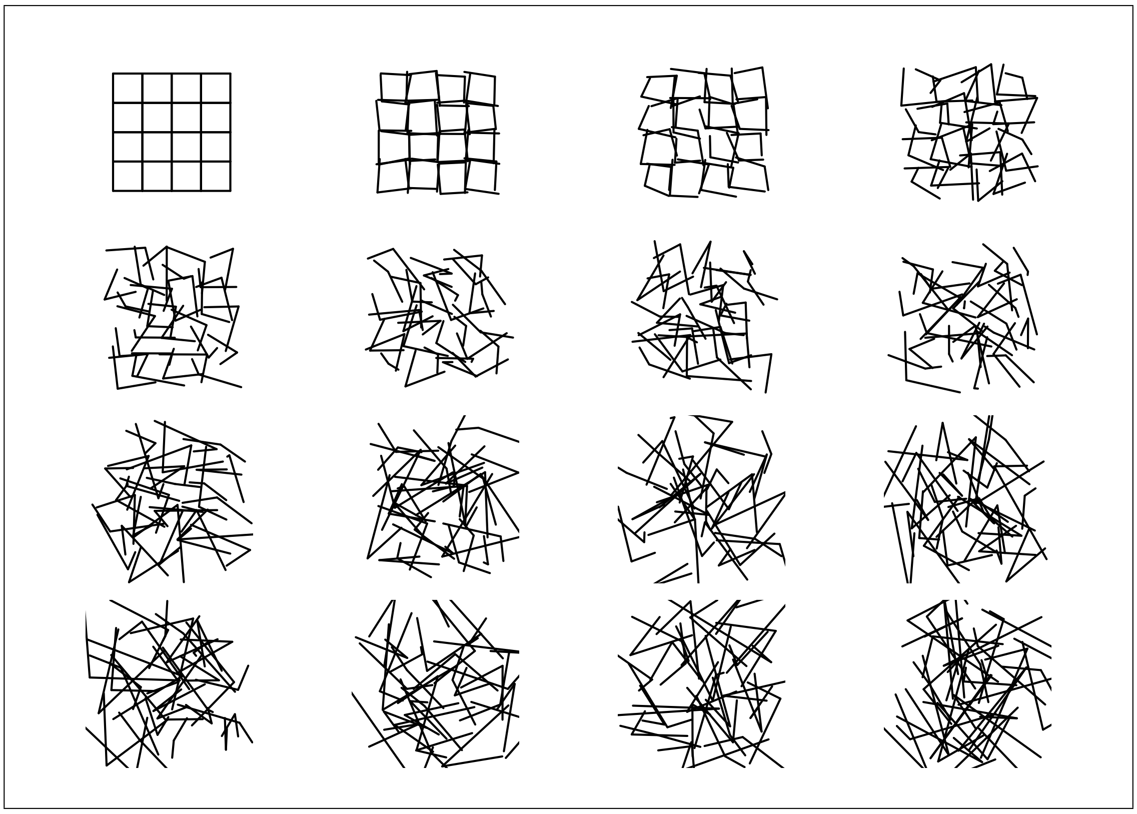
```
background(255);
translate(100,100);
mkCorners(frameCount-1,35,0,35);
drawGridOfShapes(shapeToDraw,4,70);
```

Stroke Weight in pdf: J3

```
pdf = (PGraphicsPDF) createGraphics(width, height, PDF, "PdfTest.pdf");

beginRecord(pdf);
pdf.strokeWeight(5);
```

You need that to get pdf to alter stroke weight. strokeWeight itself just alters the display.

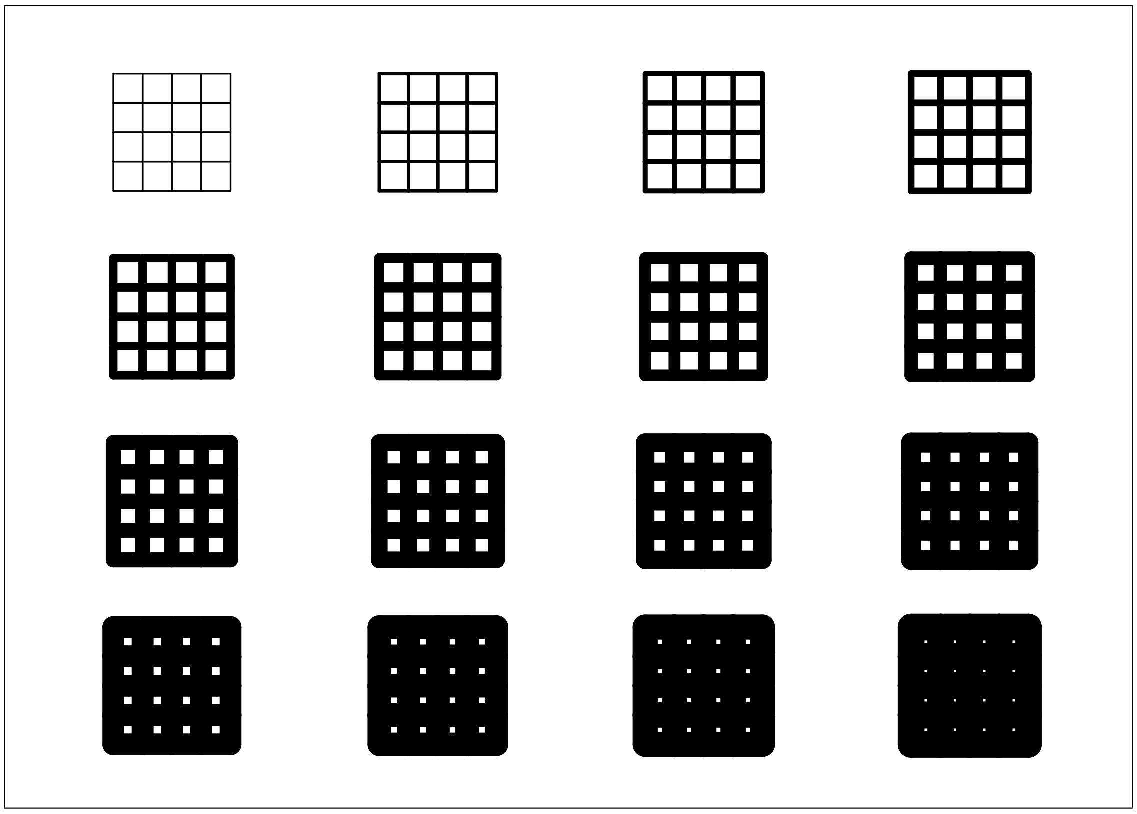
I used that and went to 16 iterations with bigger steps:



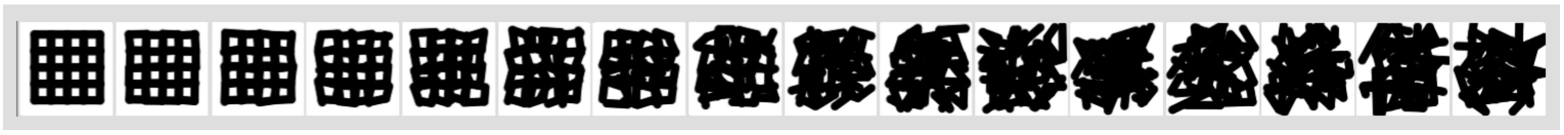
Varying pdf stroke weight with frameCount: J4

Put the weight change within `draw()`

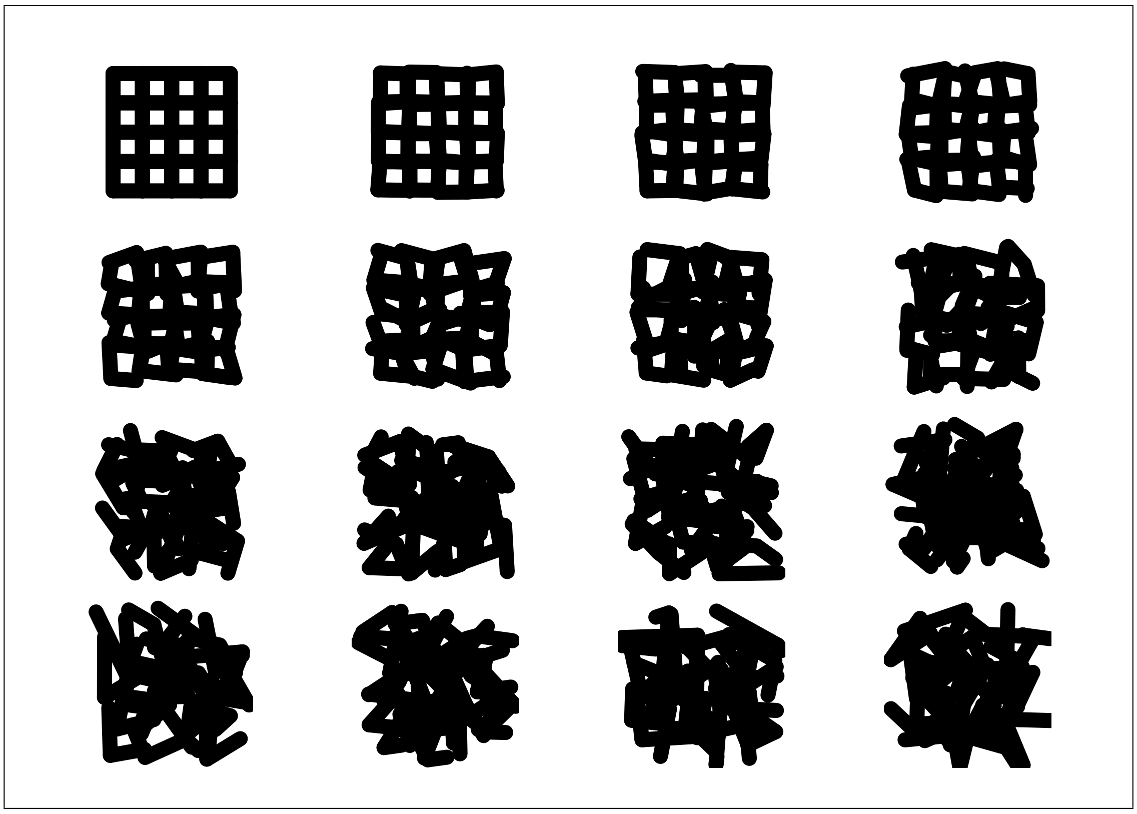
```
void draw() {
   background(255);
   translate(100,100);
   //scale(0.2);
   pdf.strokeWeight(36);
   mkCorners((framecount*5)-5,35,0,35);
   drawGridOfShapes(shapeToDraw,4,70);

  if (frameCount != 16) {
  // next page
  pdf.nextPage();
} else {
  // finish
  endRecord();
  exit();
}
}
```



Variation with given weight: J5

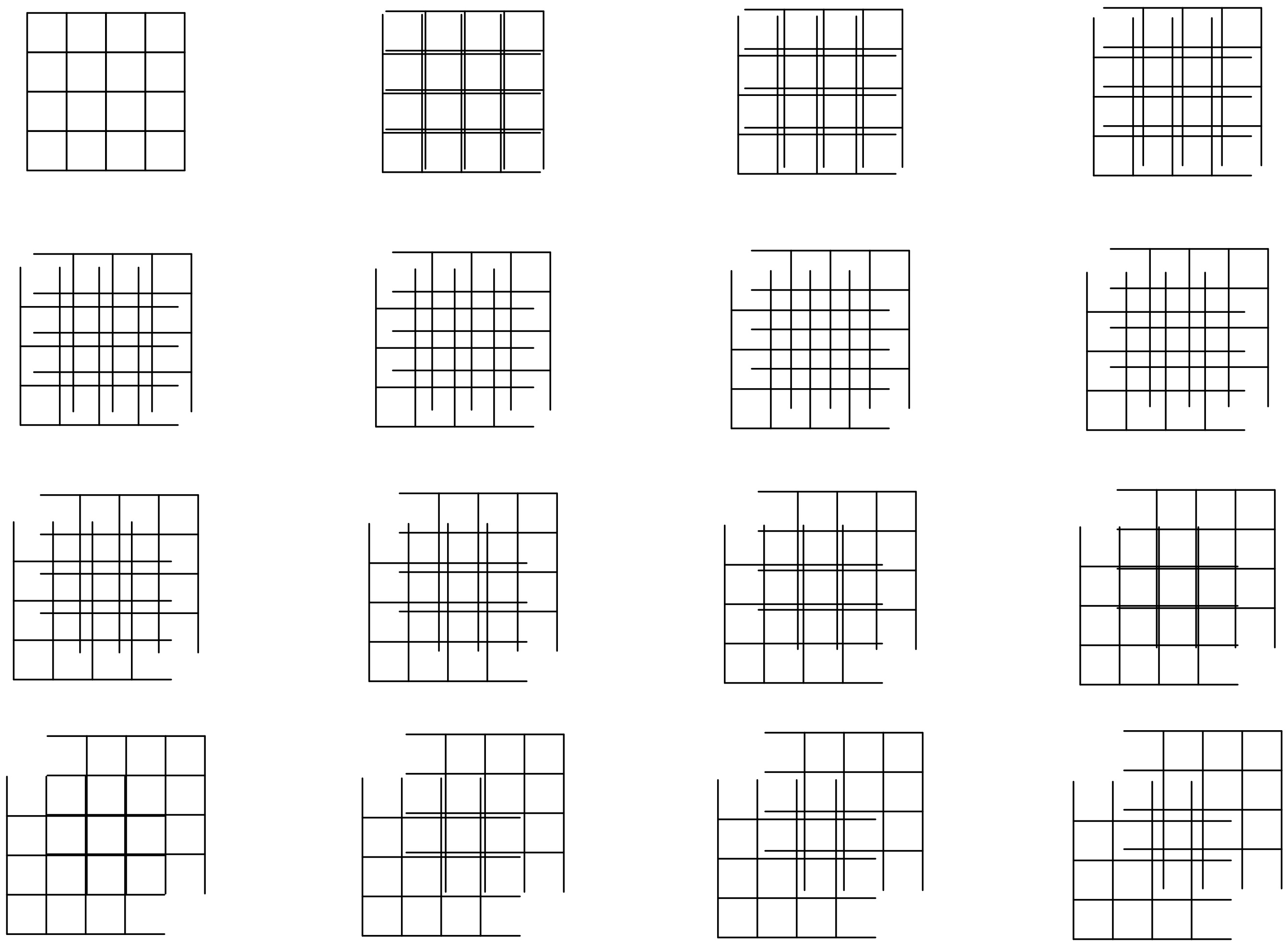


```
pdf.strokeWeight(36);
   mkCorners((4*frameCount-4),35,0,35);
   drawGridOfShapes(shapeToDraw,4,70);
```



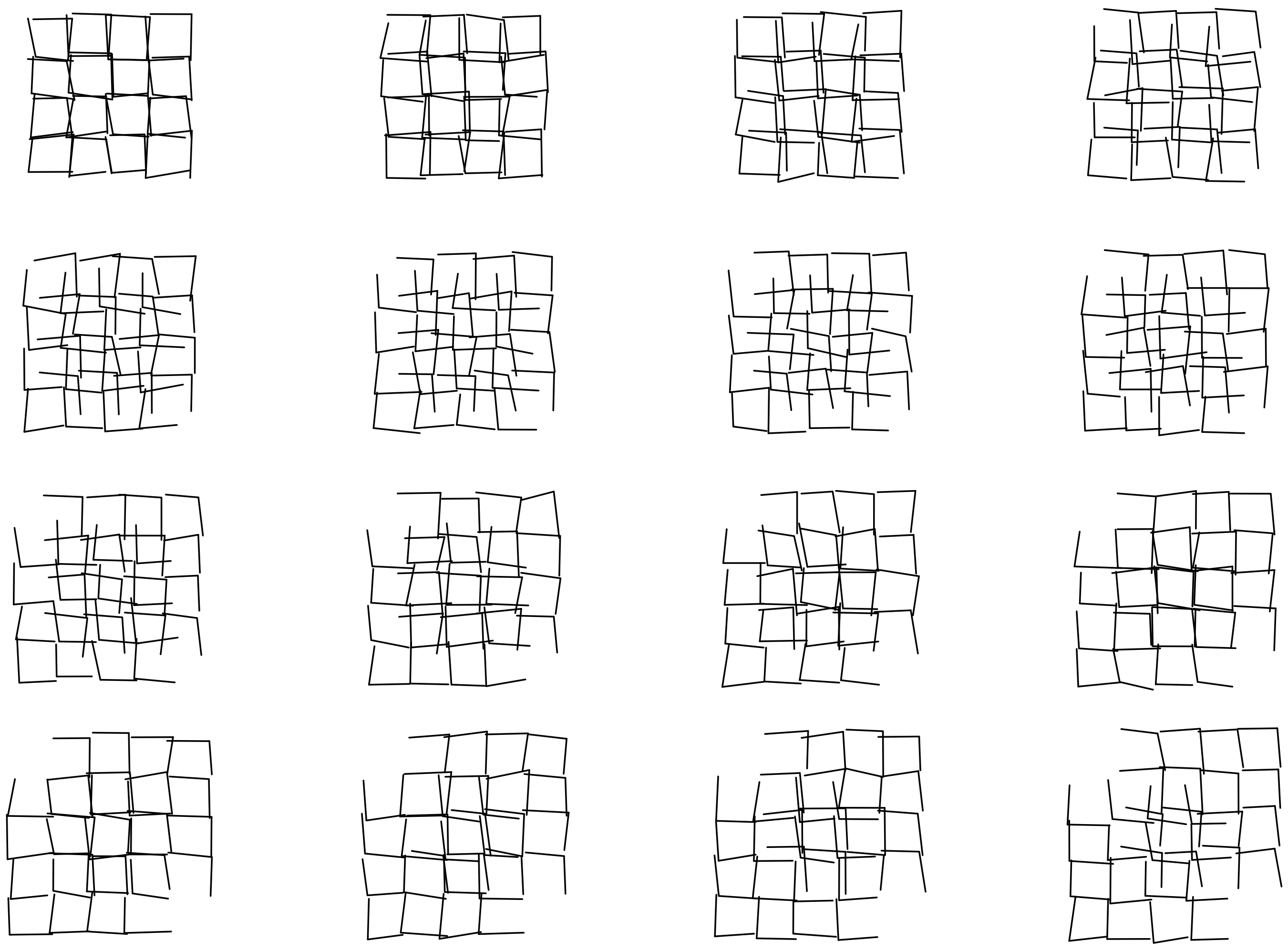
Notice how it appears very different as a sequence from a grid

Arm length, no variation: J6



```
pdf.strokeWeight(3);
   mkCorners(0,35+3*(frameCount-1),0,35-3*(frameCount-1));
   drawGridOfShapes(shapeToDraw,4,70);
```

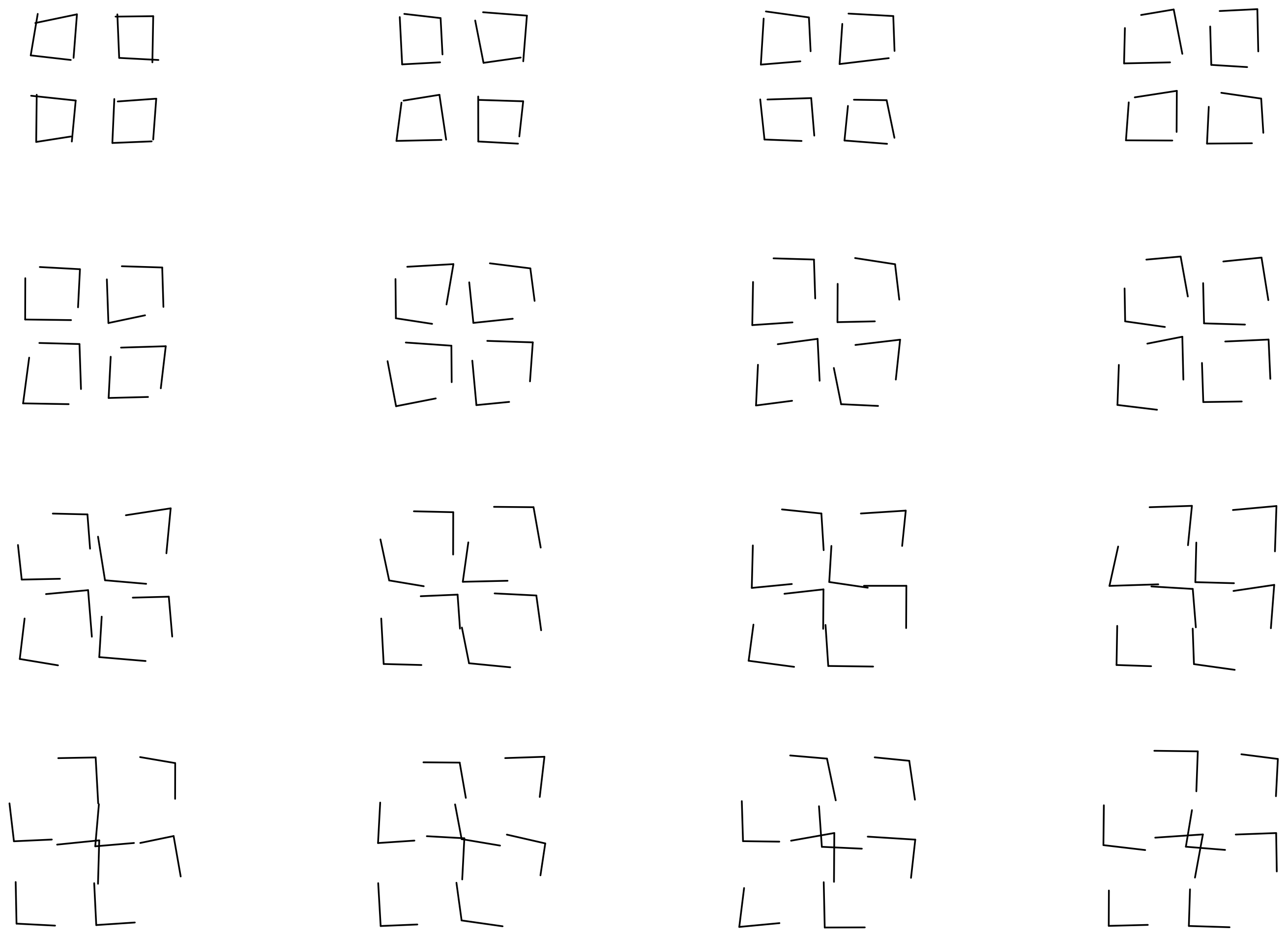
Arm length, with fixed variation: J7



```
pdf.strokeWeight(3);
   mkCorners(8,35+3*(frameCount-1),0,35-3*(frameCount-1));
   drawGridOfShapes(shapeToDraw,4,70);
```

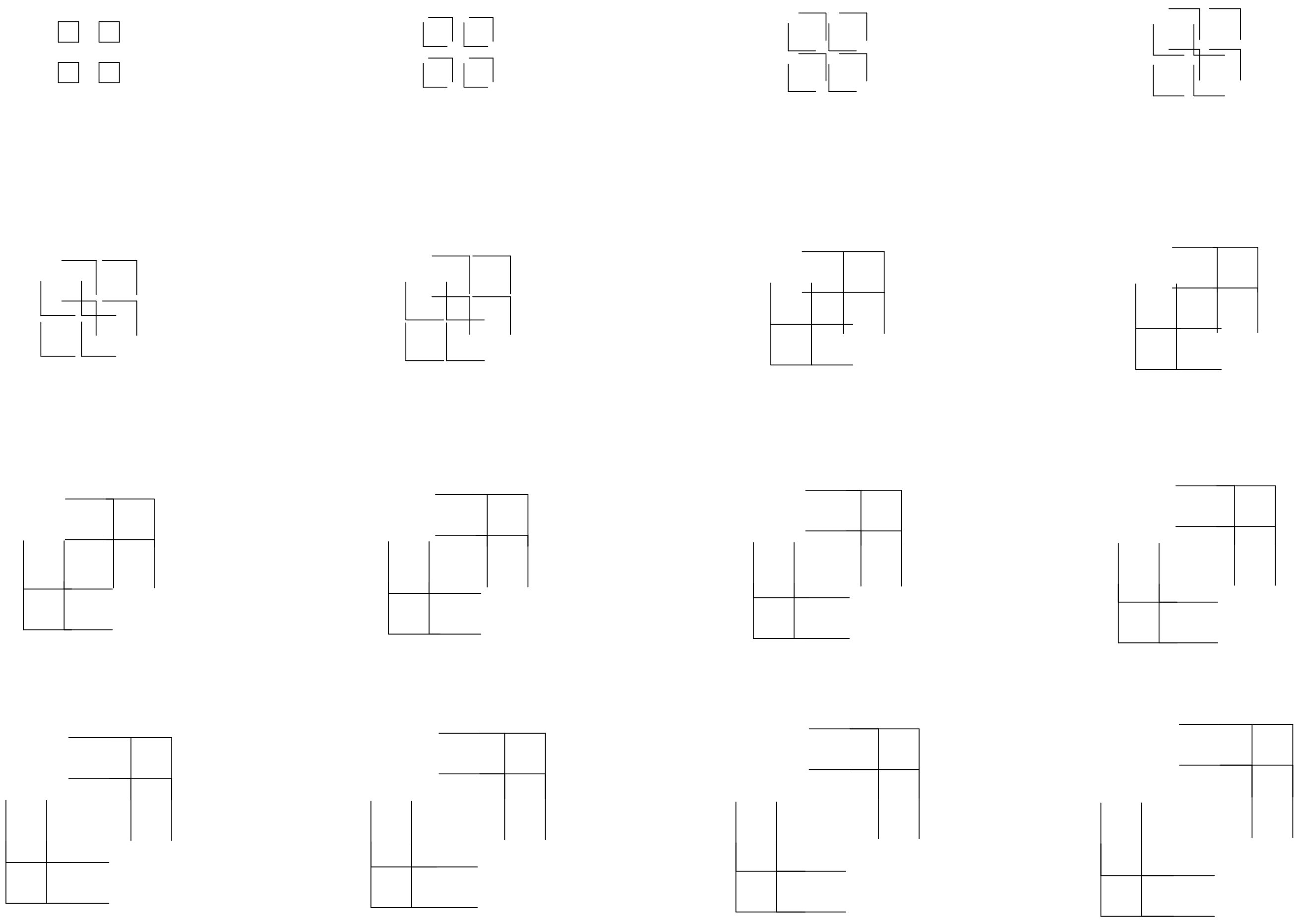
Arm length, with fixed variation: J8

Fewer items so can see strcure more easily



```
translate(120,120);
pdf.strokeWeight(3);
mkCorners(8,35+3*(frameCount-1),0,35-3*(frameCount-1));
drawGridOfShapes(shapeToDraw,2,140);
```

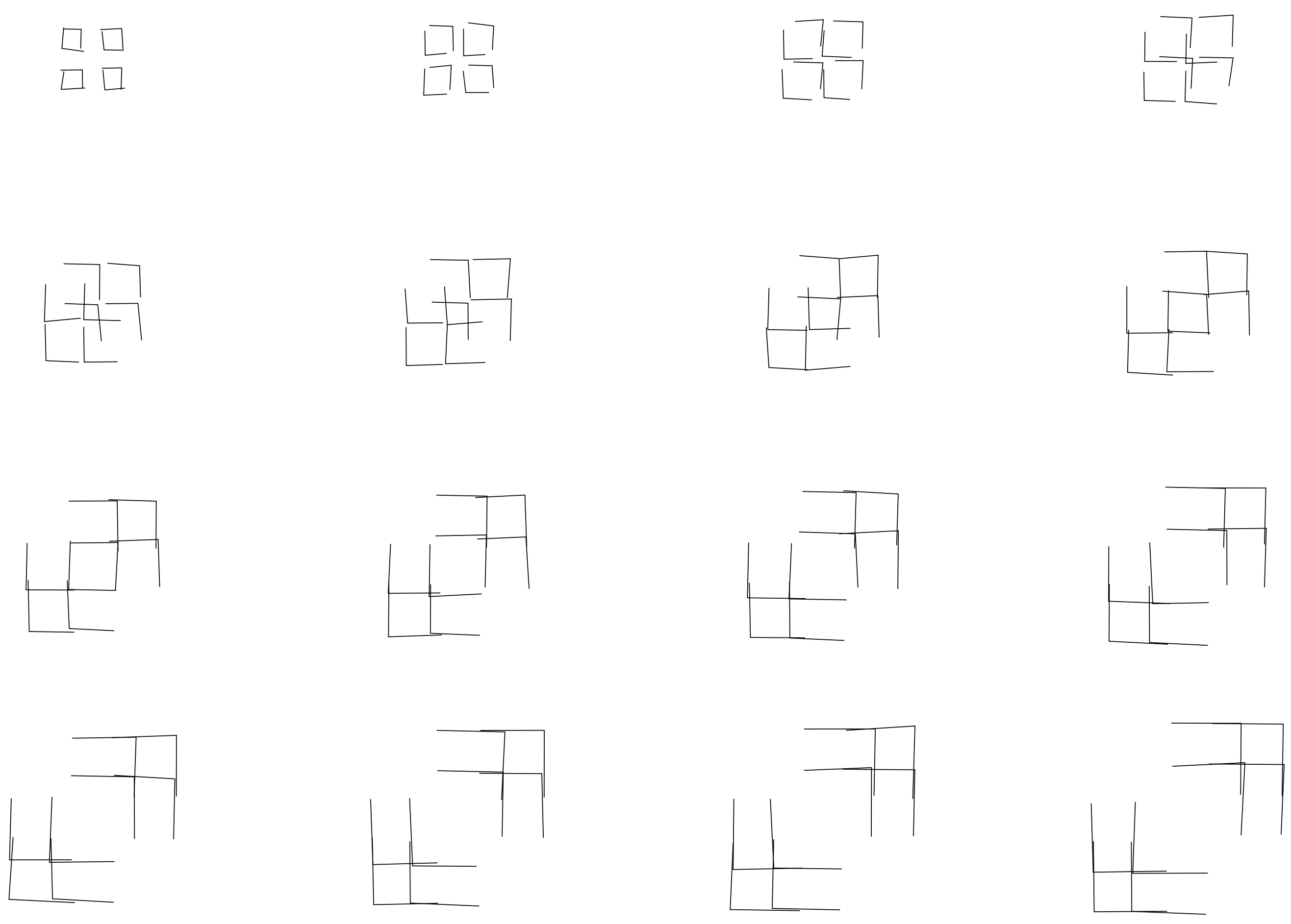
J9



In setup: `size(800,800)`, and in draw:

```
translate(320,320);
pdf.strokeWeight(3);
mkCorners(0,35+15*(frameCount-1),0,35-3*(frameCount-1));
drawGridOfShapes(shapeToDraw,2,140);
```

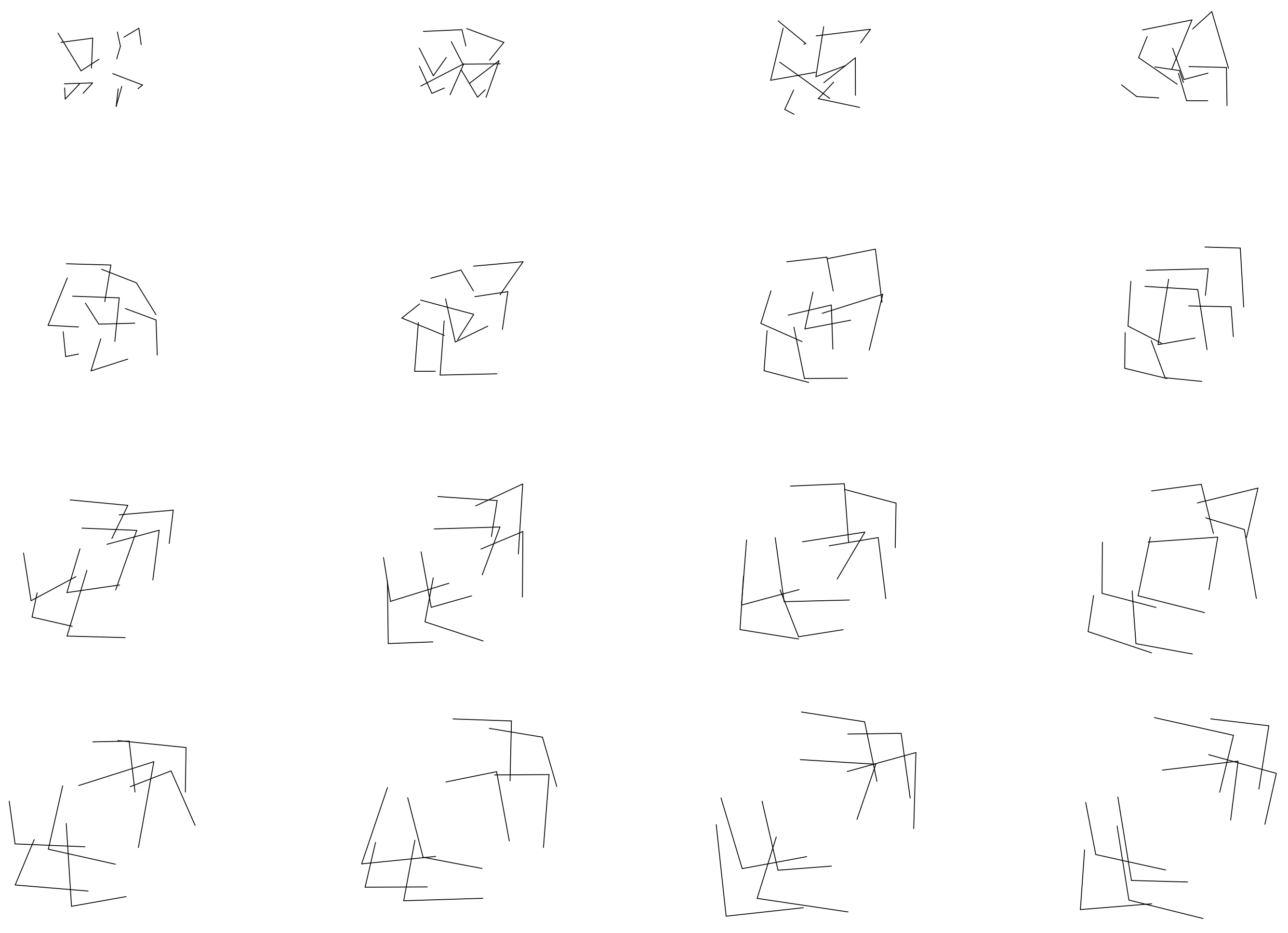
J10: as J9 but fixed variation



In setup: `size(800,800)`, and in draw:

```
translate(320,320);
pdf.strokeWeight(3);
mkCorners(8,35+15*(frameCount-1),0,35-3*(frameCount-1));
drawGridOfShapes(shapeToDraw,2,140);
```

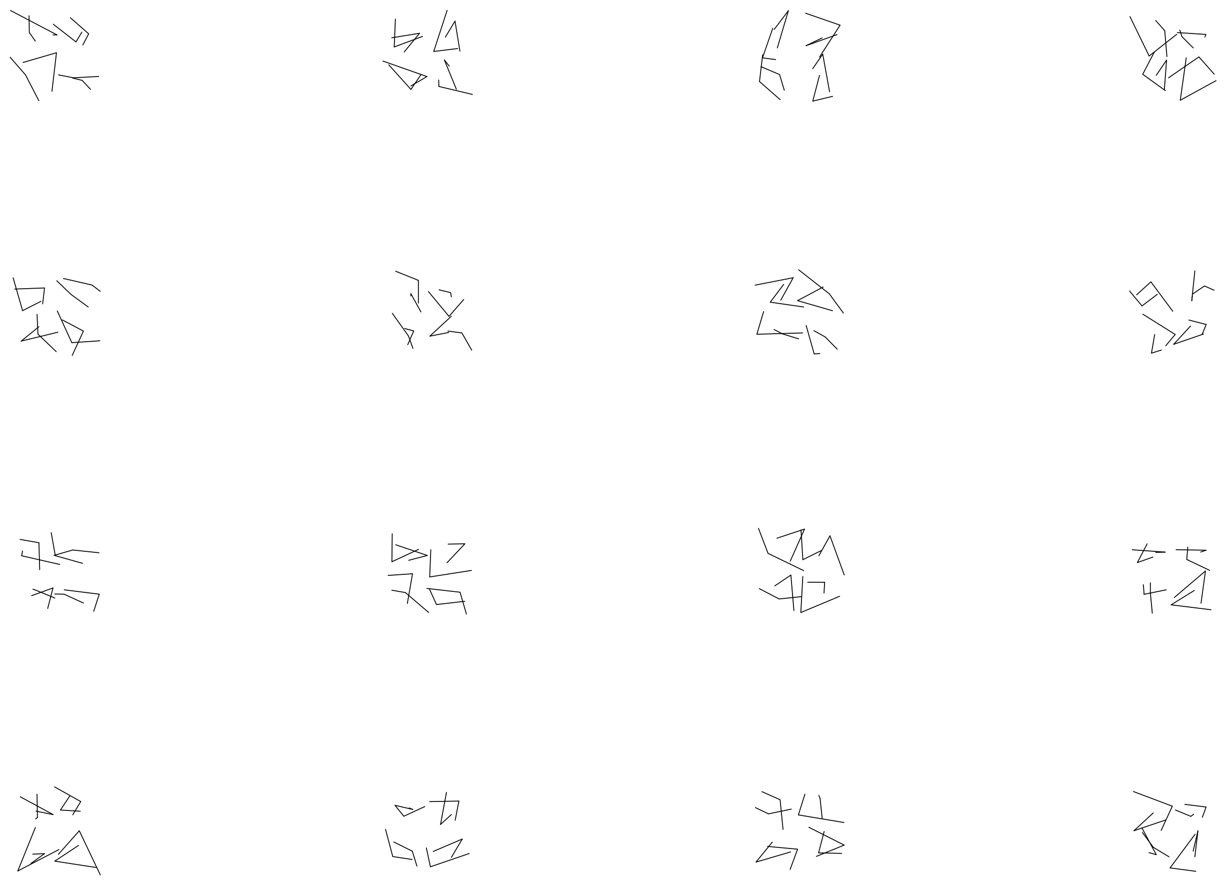
J11: as J9 but fixed variation bigger than J10



In setup: `size(800,800)`, and in draw:

```
translate(320,320);
pdf.strokeWeight(3);
mkCorners(50,35+15*(frameCount-1),0,35-3*(frameCount-1));
drawGridOfShapes(shapeToDraw,2,140);
```

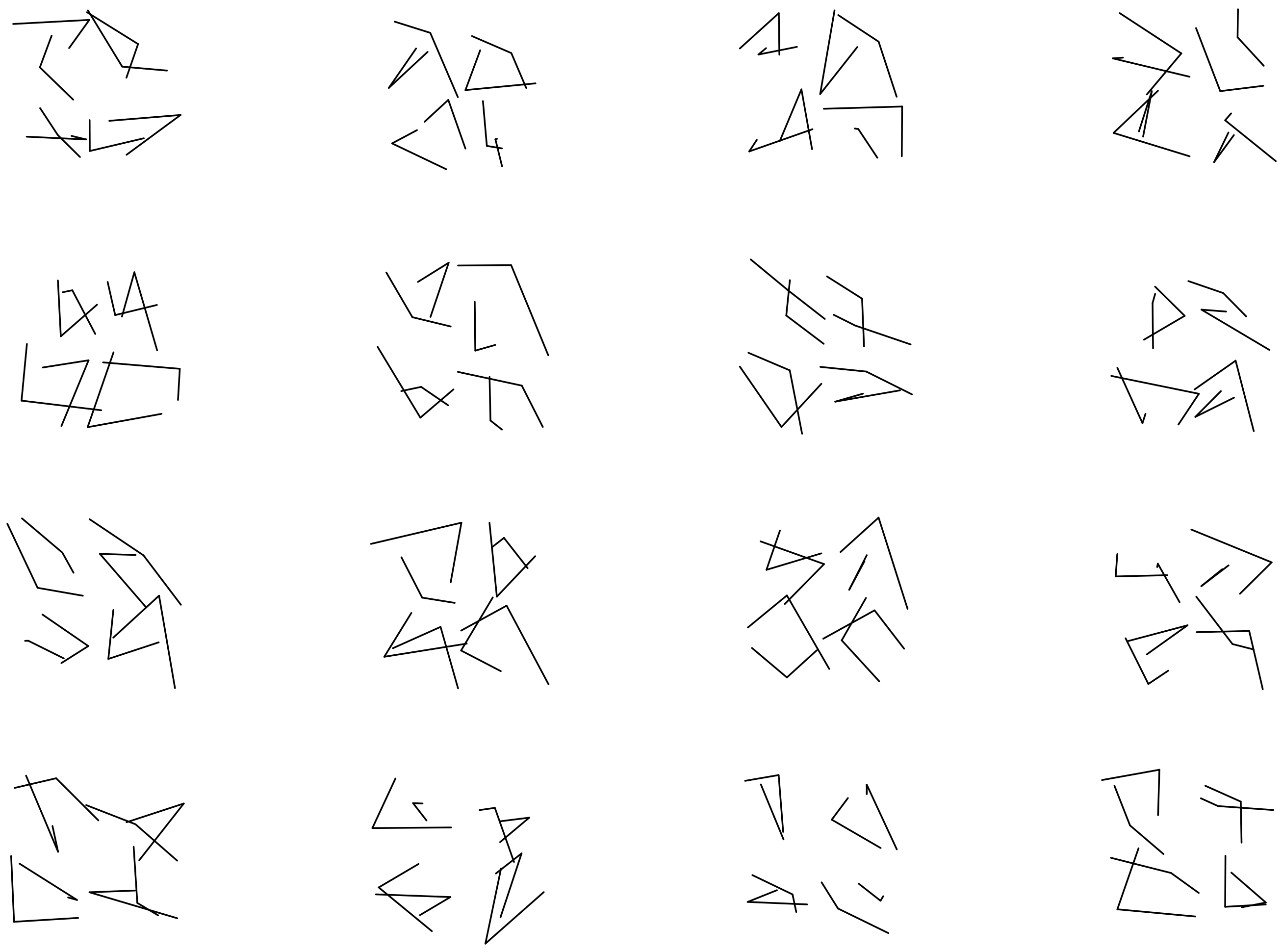
J12: different versions of 1st one in J11



In setup: `size(800,800)`, and in draw:

```
translate(320,320);
pdf.strokeWeight(3);
mkCorners(50,35,0,35);
drawGridOfShapes(shapeToDraw,2,140);
```

J13: J12 but bigger



In setup: `size(400,400)`, and in draw:

```
translate(120,120);
pdf.strokeWeight(3);
mkCorners(50,35,0,35);
drawGridOfShapes(shapeToDraw,2,140);
```

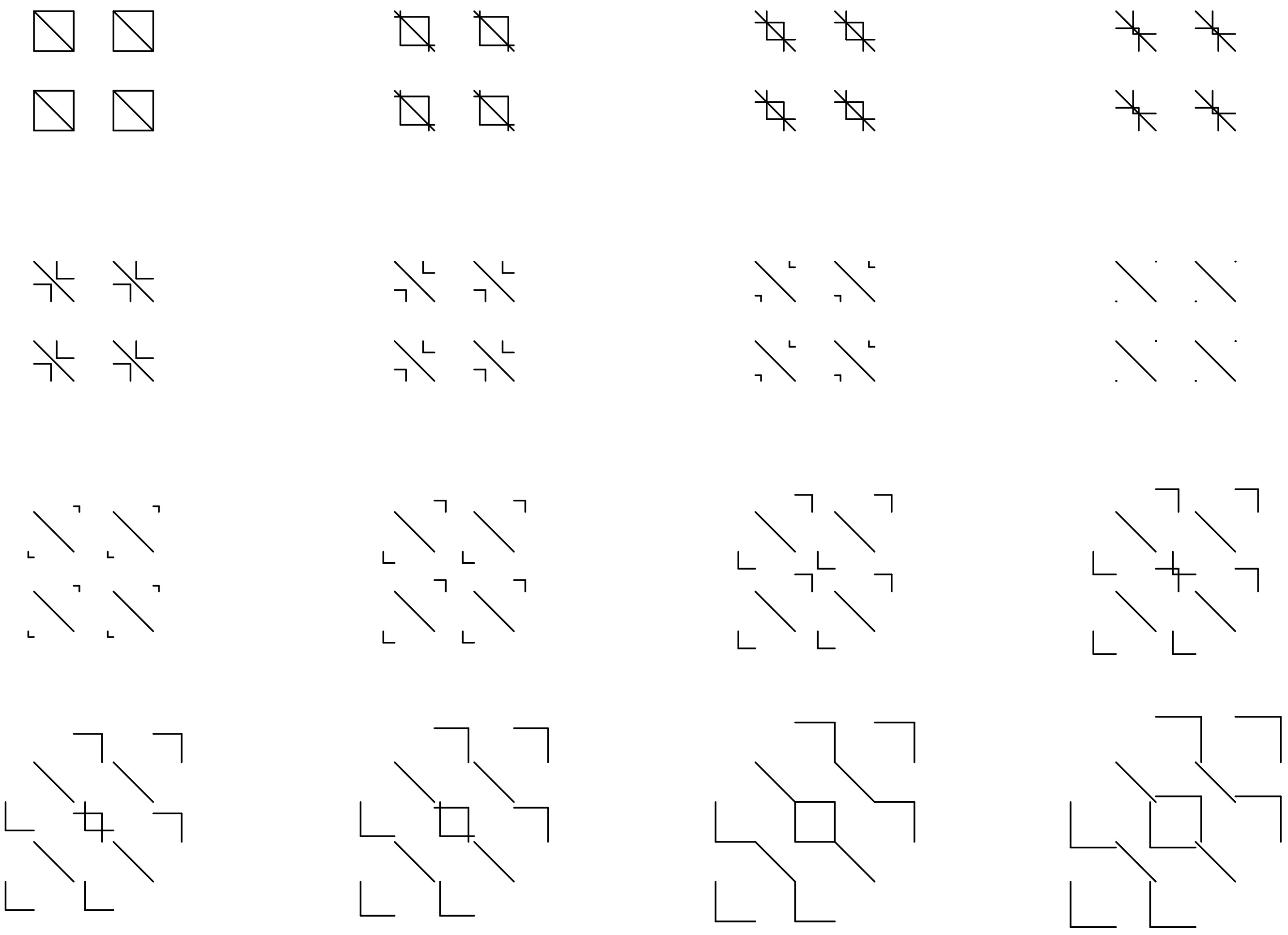
J14: diagonals drawn

Added this to `mkCorners`

```
randCorners.joinPoints(3,6);
```

and this was in `draw`

```
mkCorners(0,35-(10*(frameCount-1)),0,35);
   drawGridOfShapes(shapeToDraw,2,140);
```



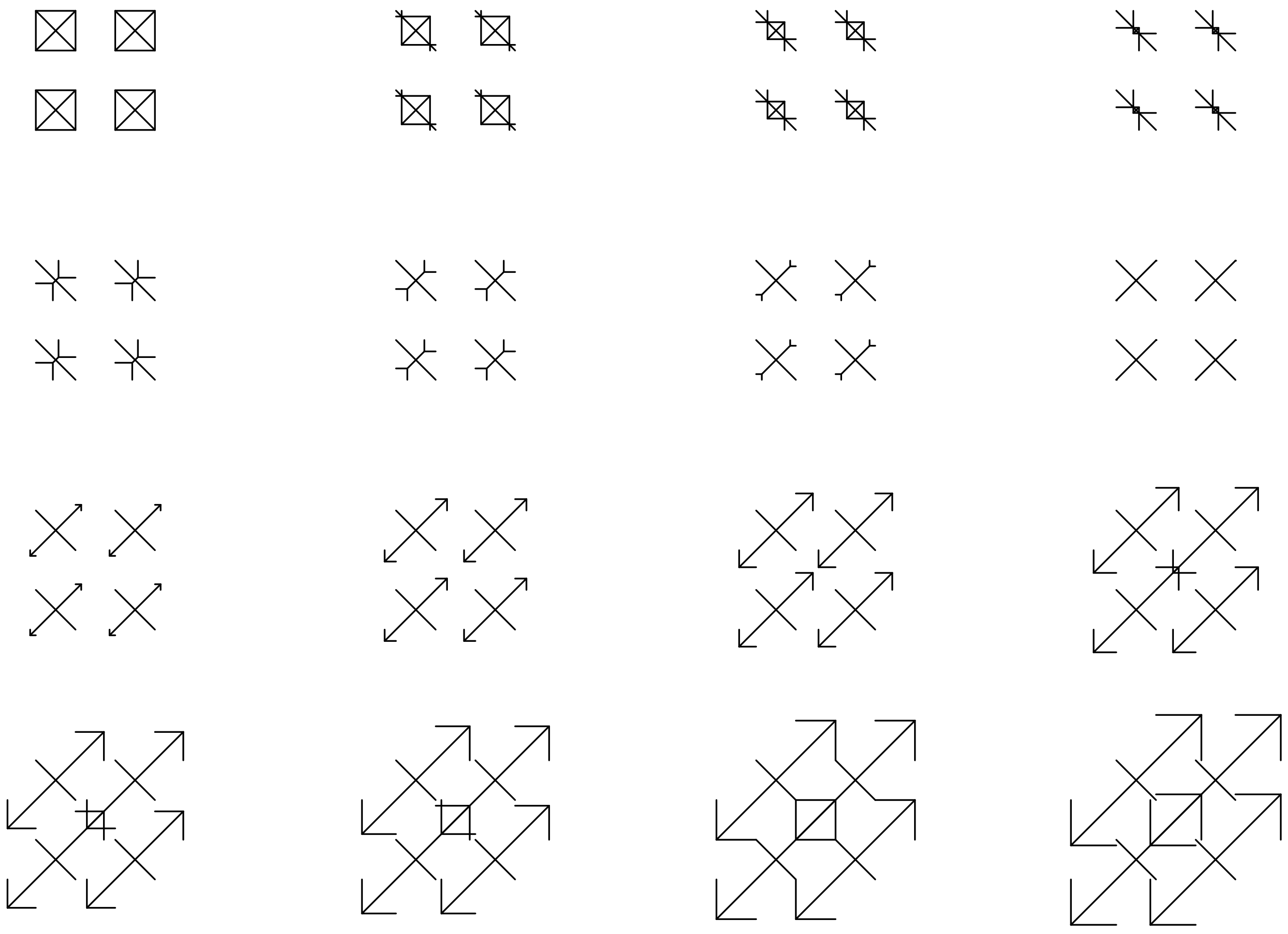
J15: both diagonals drawn

Added this to `mkCorners`

```
randCorners.joinPoints(3,6);
randCorners.joinPoints(1,2);
```

and this was in `draw`

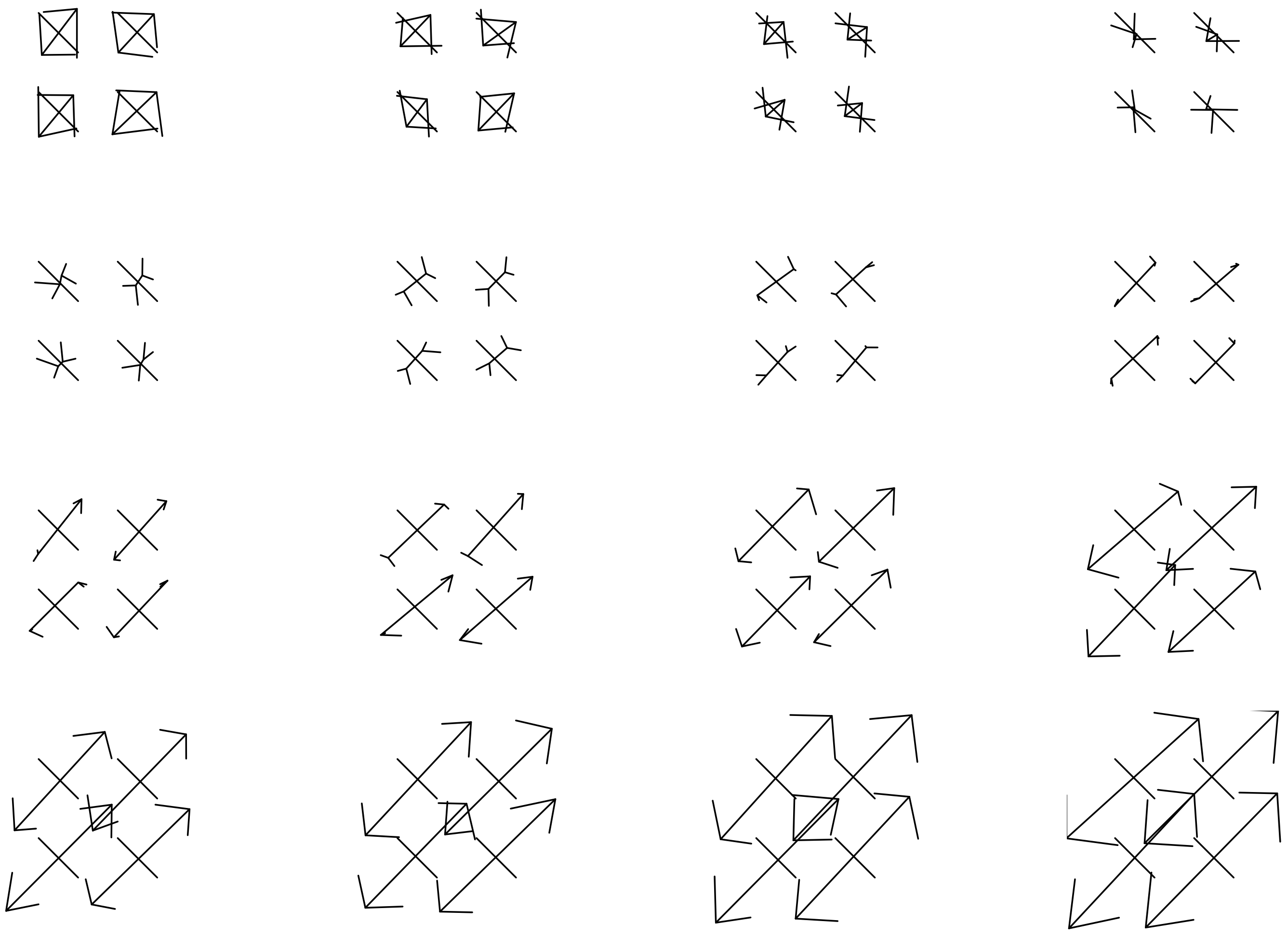
```
mkCorners(0,35-(10*(frameCount-1)),0,35);
   drawGridOfShapes(shapeToDraw,2,140);
```



J16: both diagonals drawn with variation

Juat as J15 except this was in `draw`

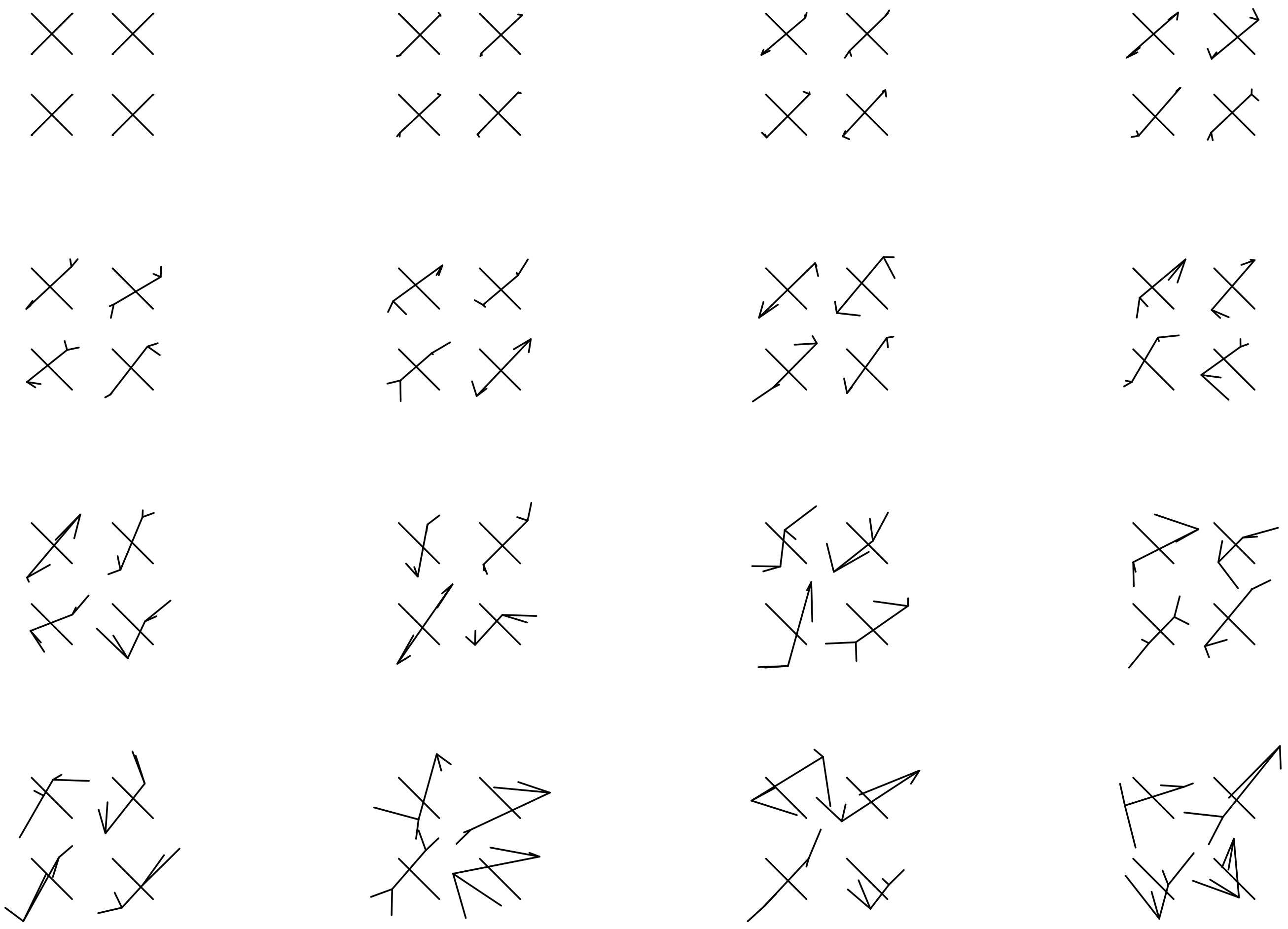
```
mkCorners(10,35-(10*(frameCount-1)),0,35);
   drawGridOfShapes(shapeToDraw,2,140);
```



J17: starting from frameCount = 8 in J15

In `draw`

```
mkCorners(4*(frameCount - 1),35-(70),0,35);
```

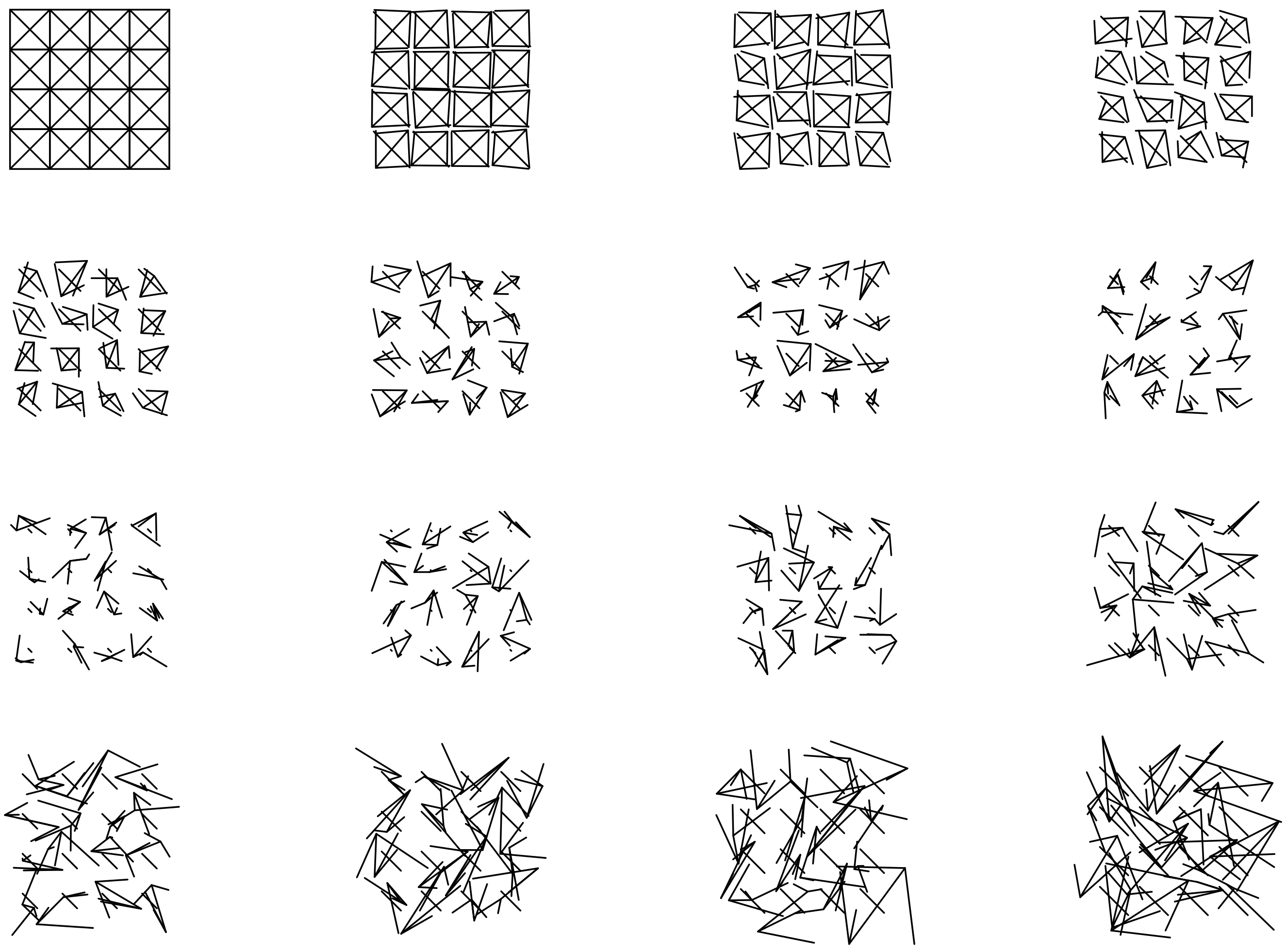


J18: back to J2 but now with diagonals

```
void draw() {
   background(255);
   translate(100,100);
   pdf.strokeWeight(3);
   mkCorners(4*(frameCount - 1),39-(4*frameCount),0,39-(4*frameCount));
   drawGridOfShapes(shapeToDraw,4,70);
```

With this in `mkCorners`

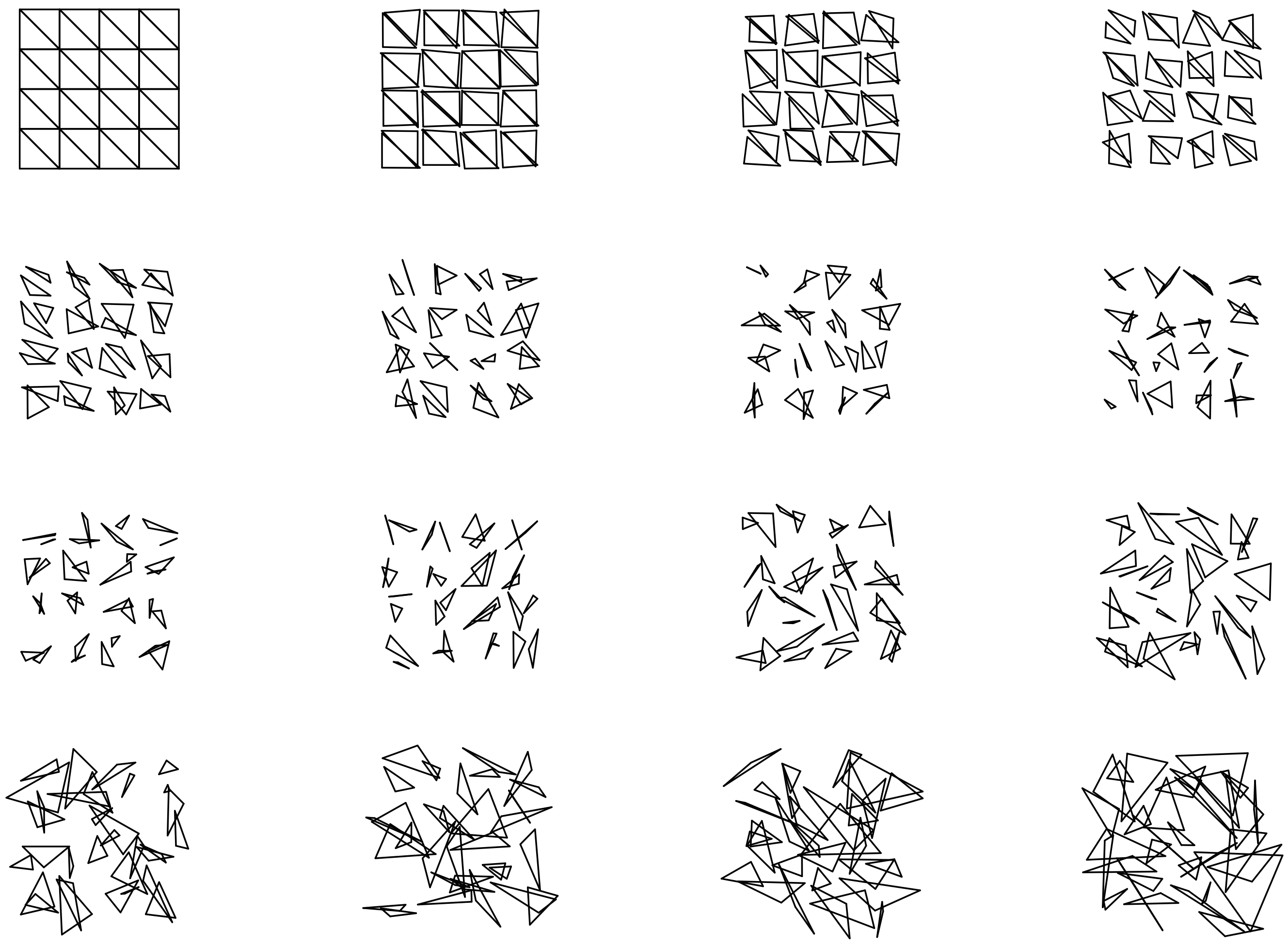
```
    randCorners.joinPoints(3,6);
    randCorners.joinPoints(1,2);
```



J19: as J18 but triangles not diagonals

`draw()` as in J18, but join different points in `mkCorners`

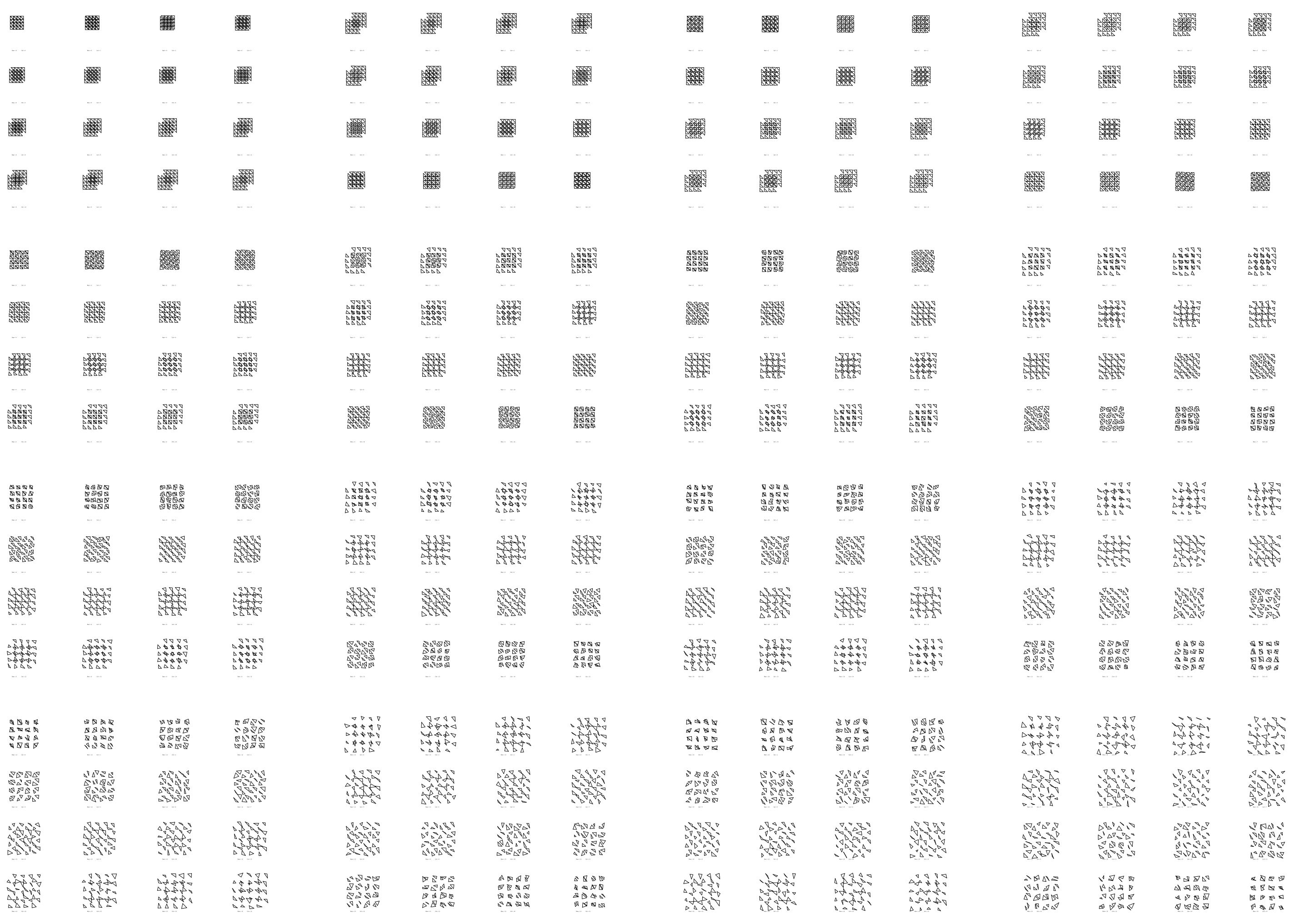
```
    randCorners.joinPoints(5,8);
    randCorners.joinPoints(4,7);
```



J20: combining J19 and J7, increasing separation, boustrophedon

Combines J19 and J20 with increasing separation, and laid out from left to right in first row then right to left in next row and so on. Not really boustrophedon as within the right to left lines the images are not reveresed. This gives a linear sequence that jumps in small steps when the whole is seen as an animation.

I included text in this example to figure out the rows and columns.

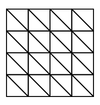


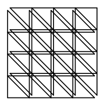
Code on next page. The pdf is 256 pages.

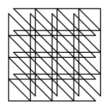
J20: combining J19 and J7, increasing separation, boustrophedon

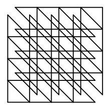
```
void setup(){
    size(1000,1000);
    background(255);
    strokeWeight(3);
    stroke(0);
    frameRate(10);
  pdf = (PGraphicsPDF) createGraphics(width, height, PDF, "PdfTest.pdf");

  beginRecord(pdf);
  pdf.textSize(24);
  pdf.strokeWeight(3);
}

void draw() {
   int col = (frameCount-1) % 16;
   int row = (frameCount-1) / 16;
   int parity = row % 2;

   background(255);
   translate(230,230);

   fill(0,0,0);
   pdf.text("Row = "+str(row), 0,700);
   pdf.text("Col = "+str(col), 200,700);
   if (parity == 0) {
     mkCorners((2*row),35+4*col,0,35-4*col);
   }
   else
   {mkCorners((2*row),35+4*(16-col),0,35-4*(16-col));
   }
   drawGridOfShapes(shapeToDraw,4,70+(frameCount/2));

  if (frameCount != 16*16) {pdf.nextPage();}
  else {endRecord();
        exit();}
}
```

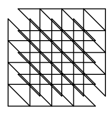
J20: combining J19 and J7, increasing separation, boustrophedon

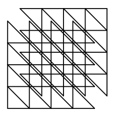

Row = 0 Col = 0

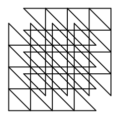

Row = 0 Col = 1

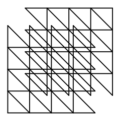

Row = 0 Col = 2

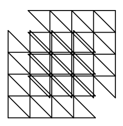

Row = 0 Col = 3

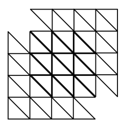

Row = 0 Col = 4

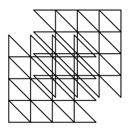

Row = 0 Col = 5

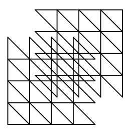

Row = 0 Col = 6

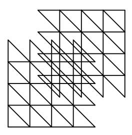

Row = 0 Col = 7

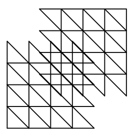

Row = 0 Col = 8

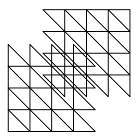

Row = 0 Col = 9

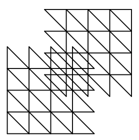

Row = 0 Col = 10

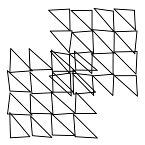

Row = 0 Col = 11

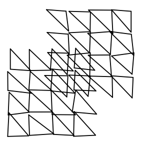

Row = 0 Col = 12

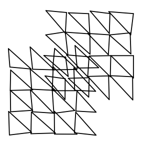

Row = 0 Col = 13

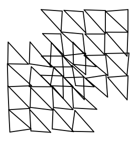

Row = 0 Col = 14

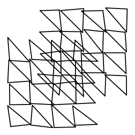

Row = 0 Col = 15

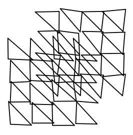

Row = 1 Col = 0

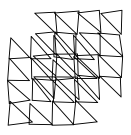

Row = 1 Col = 1

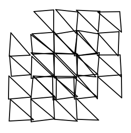

Row = 1 Col = 2

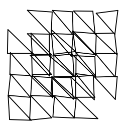

Row = 1 Col = 3

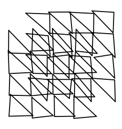

Row = 1 Col = 4

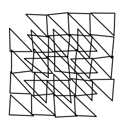

Row = 1 Col = 5

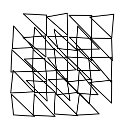

Row = 1 Col = 6

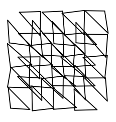

Row = 1 Col = 7

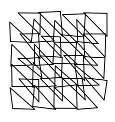

Row = 1 Col = 8

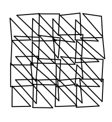

Row = 1 Col = 9

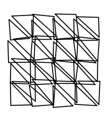

Row = 1 Col = 10


Row = 1 Col = 11


Row = 1 Col = 12

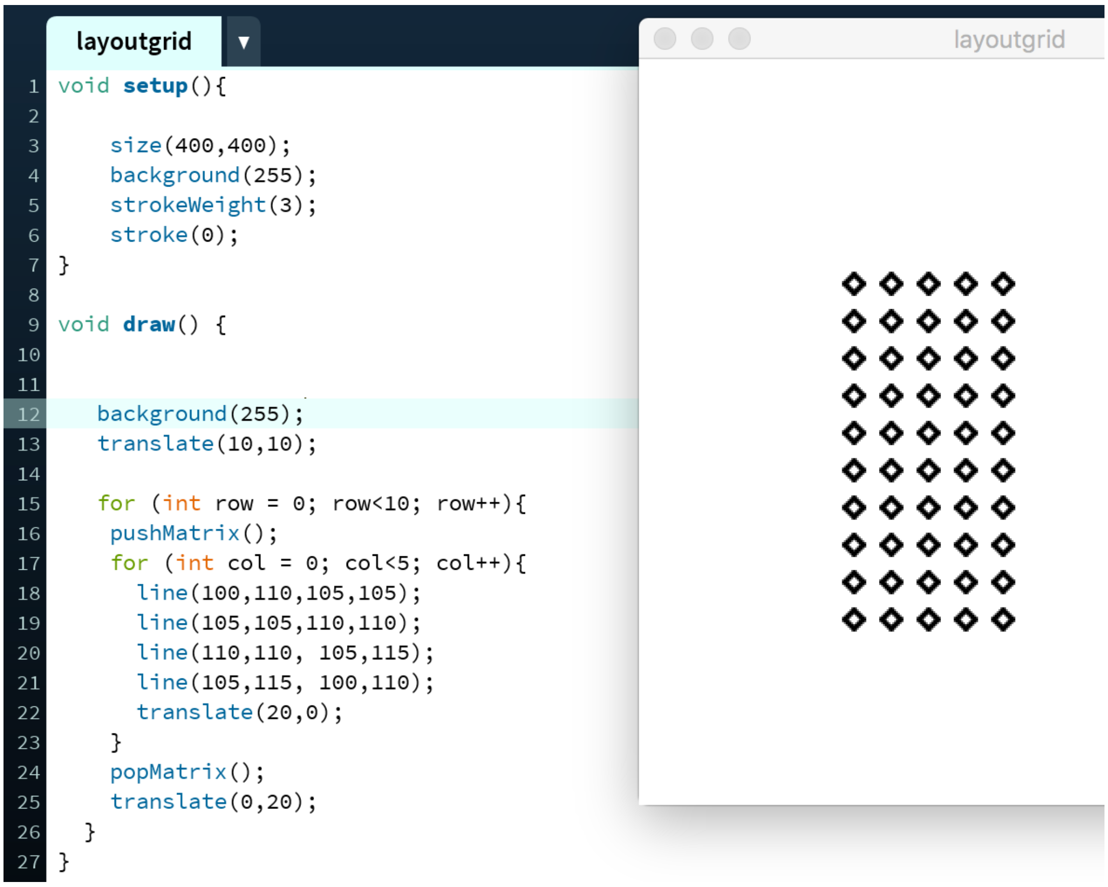

Row = 1 Col = 13


Row = 1 Col = 14


Row = 1 Col = 15

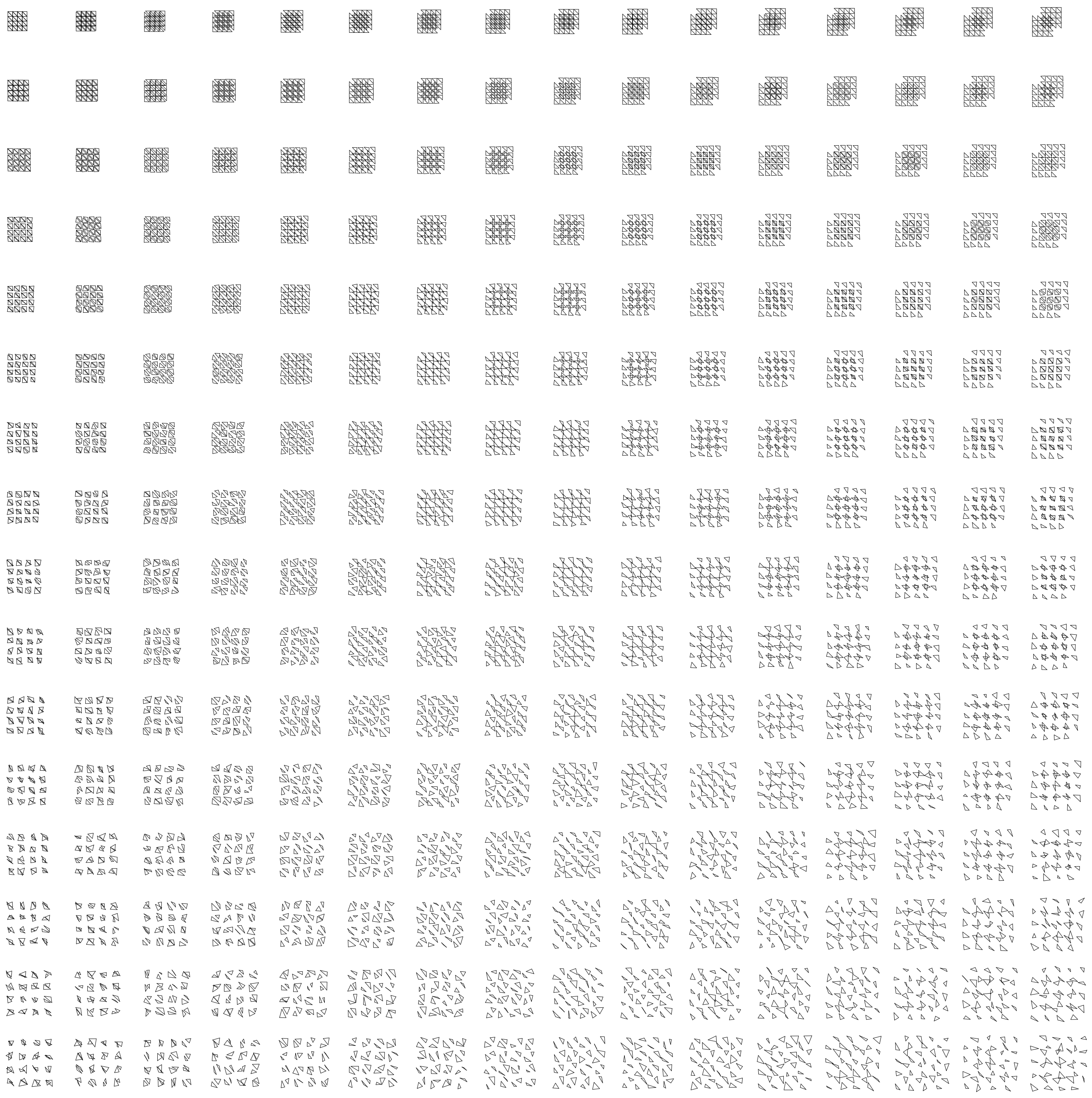
Making a grid on one page

I made this example to prepare for drawing all of the 256 examples in J20 on a single sheet.



```
void setup(){

    size(400,400);
    background(255);
    strokeWeight(3);
    stroke(0);
}

void draw() {


   background(255);
   translate(10,10);

   for (int row = 0; row<10; row++){
    pushMatrix();
    for (int col = 0; col<5; col++){
      line(100,110,105,105);
      line(105,105,110,110);
      line(110,110, 105,115);
      line(105,115, 100,110);
      translate(20,0);
    }
    popMatrix();
    translate(0,20);
  }
}
```

J21: Making a grid on one page

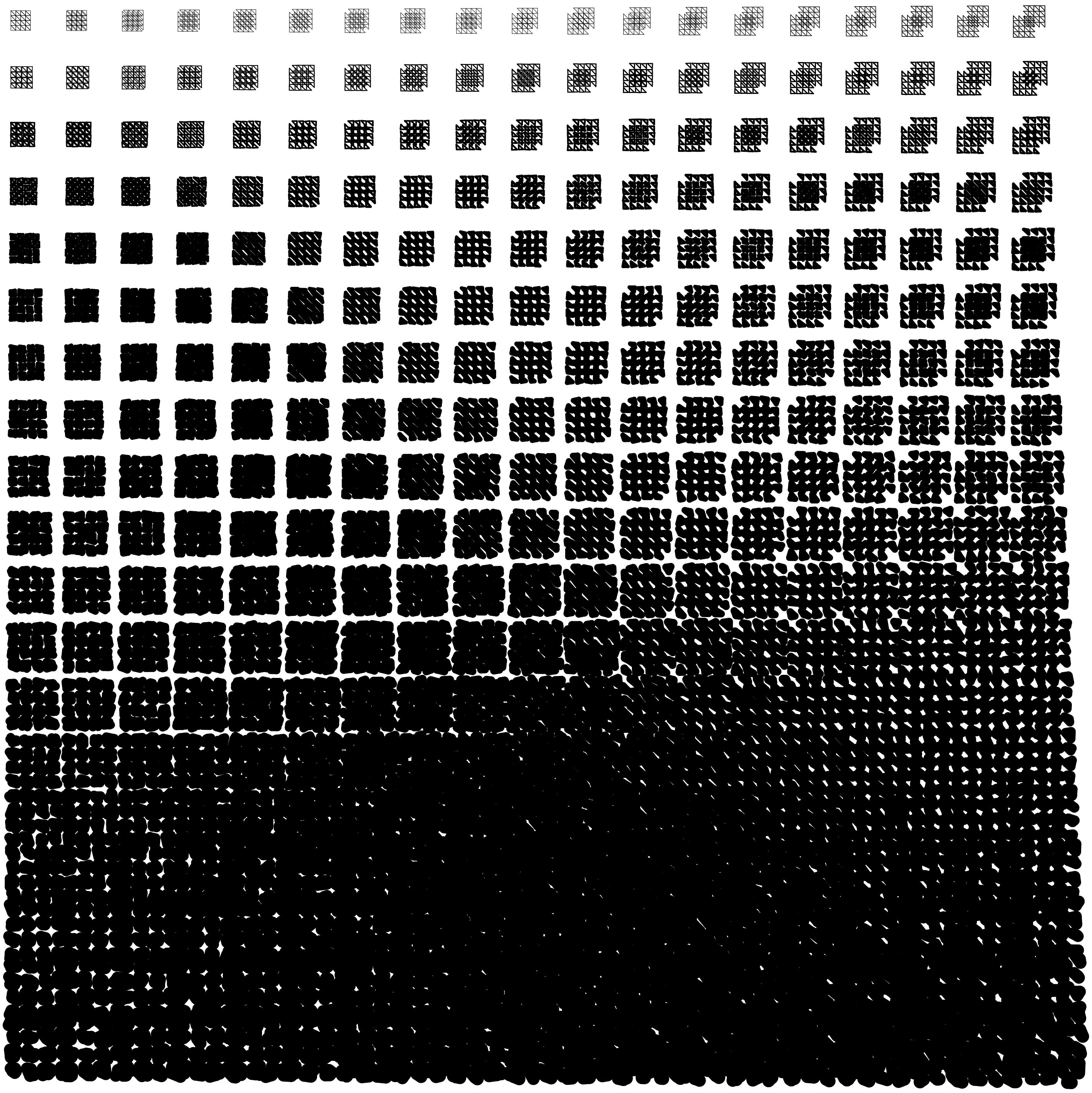


Code for J21

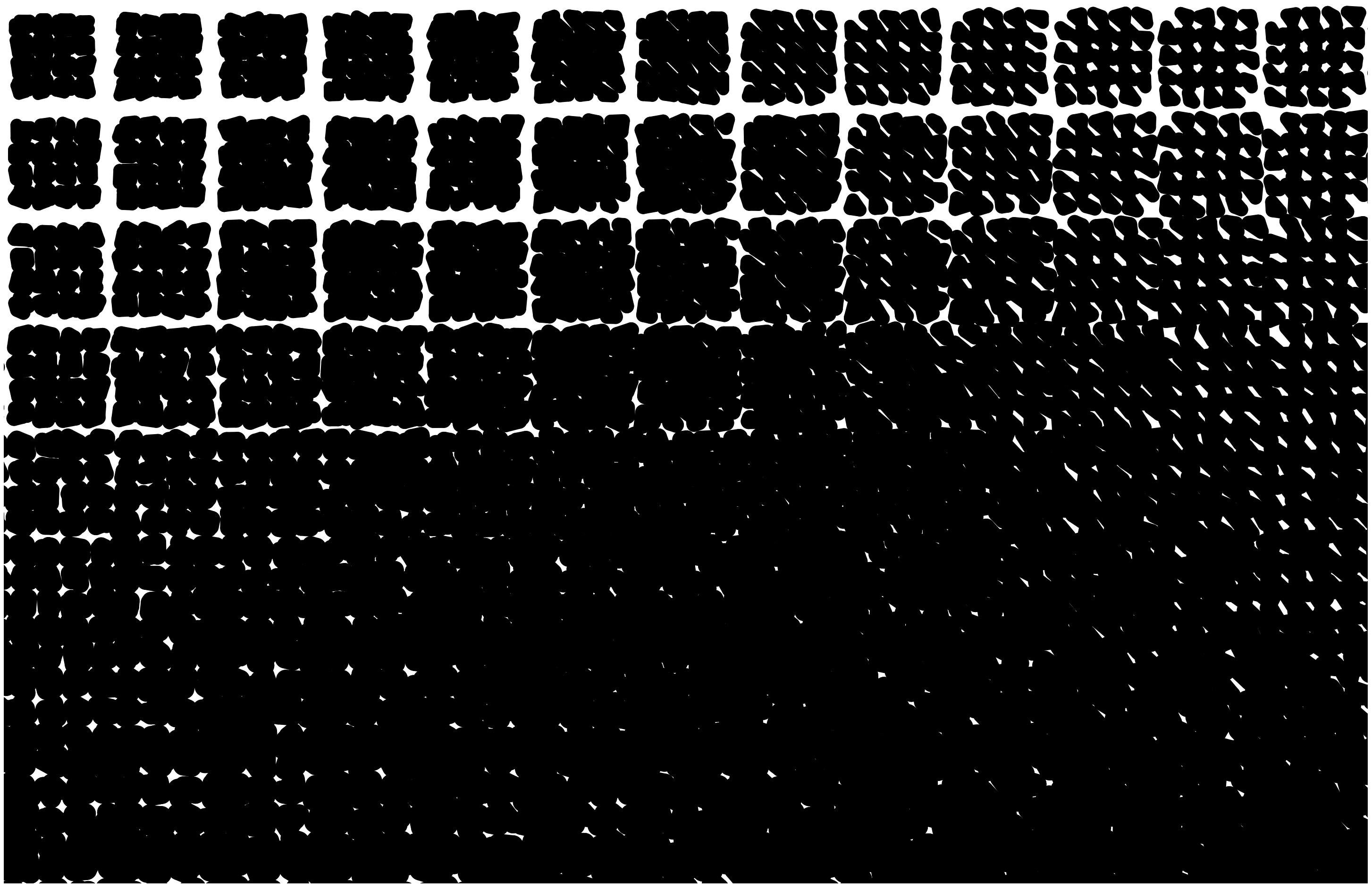
I had trouble without the innermost push and pop. I don't understand why it needs this

```
void setup(){

    size(16000,16000);
    background(255);
    strokeWeight(3);
    stroke(0);
  pdf = (PGraphicsPDF) createGraphics(width, height, PDF, "PdfTest.pdf");
  beginRecord(pdf);
  pdf.textSize(24);
  pdf.strokeWeight(3);

  }

void draw() {
   background(255);
   translate (200,200);
   for (int row = 0; row<16; row++){
    pushMatrix();
    for (int col = 0; col<16; col++){
      mkCorners((2*row),35+4*col,0,35-4*col);
      pushMatrix();
          drawGridOfShapes(shapeToDraw,4,70+((16*row)+col)/2);
      popMatrix();
      translate(1000,0);
    }
    popMatrix();
    translate(0,1000);
  }

  endRecord();
  exit();
}
```

J22: Big grid with stroke weight increase



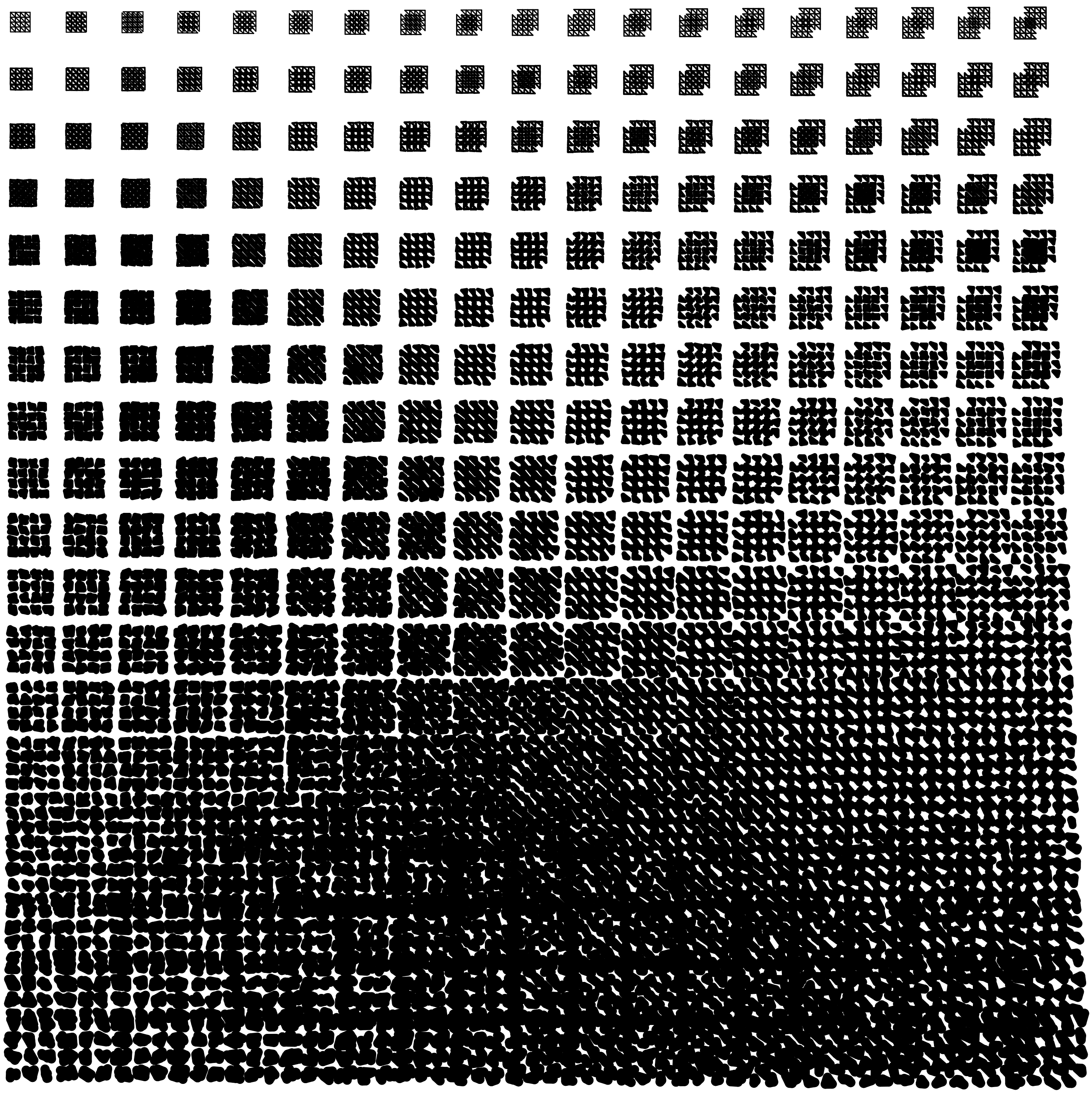
J22: Big grid with stroke weight increase: zoom in



J22: Big grid with stroke weight increase: code

```
import processing.pdf.*;          // Import PDF code
PGraphicsPDF pdf;

RandomShape randCorners = new RandomShape(9);
RandomShape shapeToDraw = randCorners;


void setup(){
    size(16000,16000);
    background(255);
    stroke(0);

  pdf = (PGraphicsPDF) createGraphics(width, height, PDF, "PdfTest.pdf");
  beginRecord(pdf);
  }

void draw() {
   int rowMax = 19;
   int colMax = rowMax;
   background(255);
   translate (200,200);
   for (int row = 0; row<rowMax; row++){
    pushMatrix();
    for (int col = 0; col<colMax; col++){
      mkCorners((2*row),35+4*col,0,35-4*col);
      pushMatrix();
          pdf.strokeWeight(1+ ((rowMax*row)+col)/2);
          drawGridOfShapes(shapeToDraw,4,70+((rowMax*row)+col)/2);
      popMatrix();
      translate(800,0);
    }
    popMatrix();
    translate(0,800);
  }

  endRecord();
  exit();
}
```

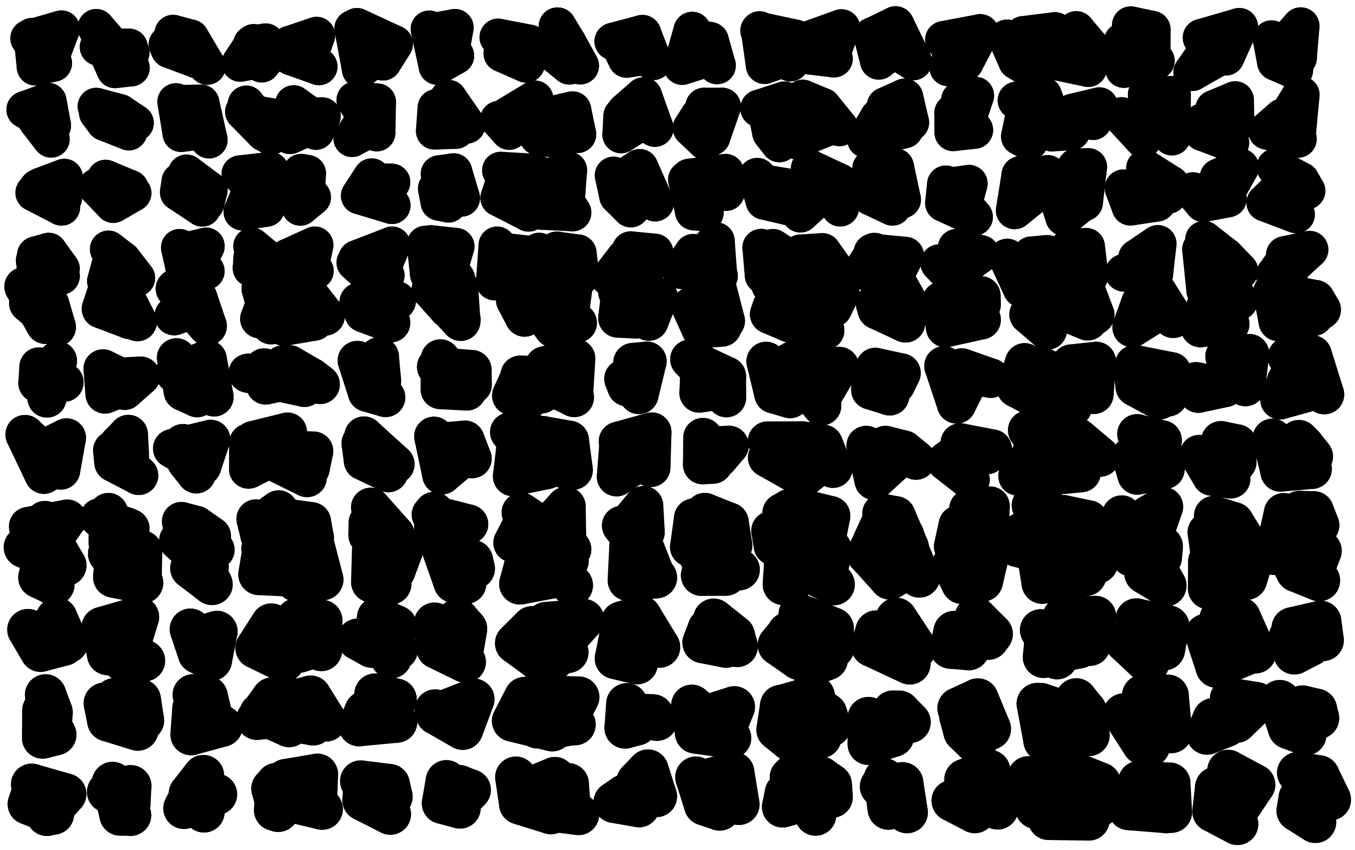
J23: Big grid with stroke weight increase; different again



J23: Big grid with stroke weight increase; different again: code

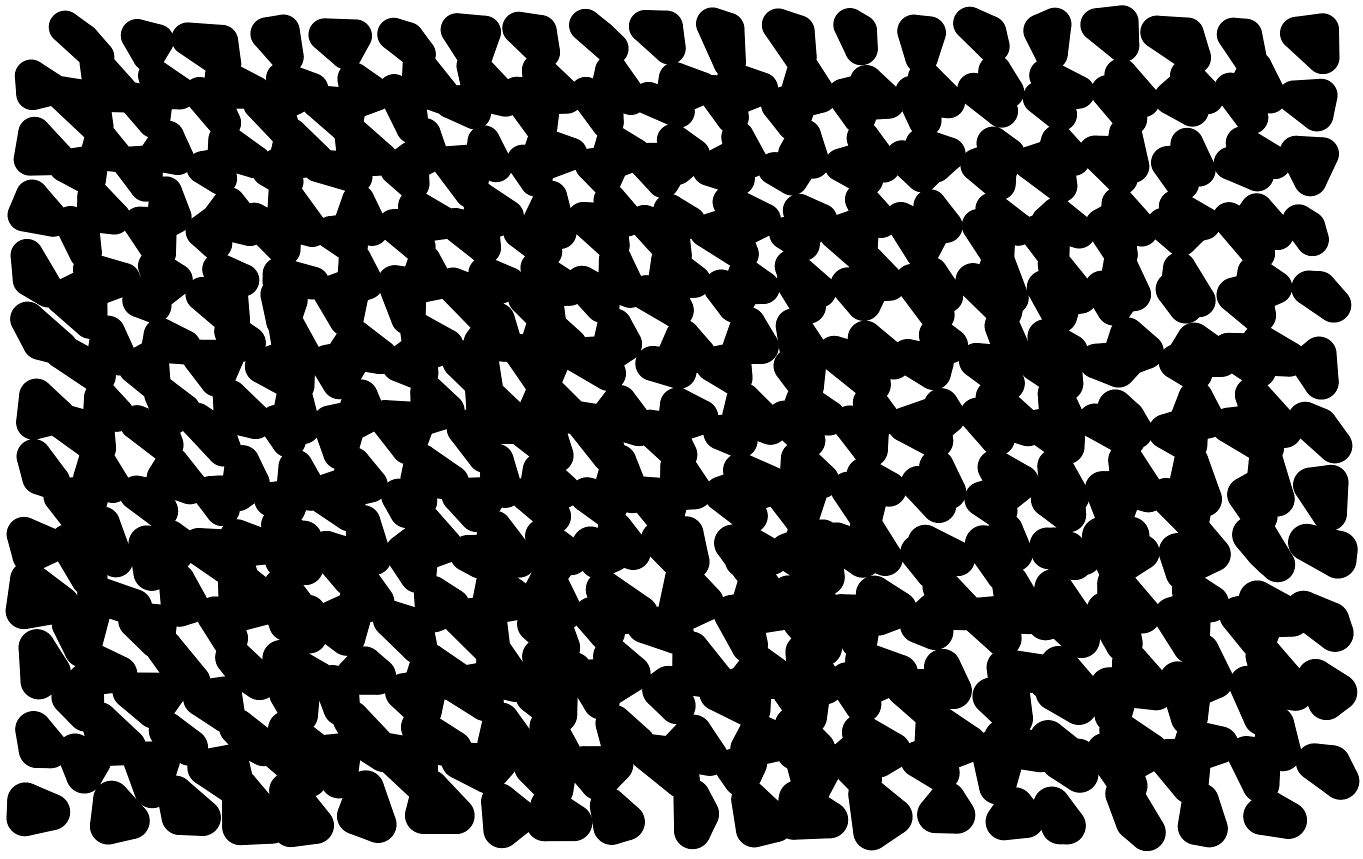
```
void draw() {
    int rowMax = 19;
    int colMax = rowMax;
   background(255);
   translate (200,200);
   for (int row = 0; row<rowMax; row++){
    pushMatrix();
    for (int col = 0; col<colMax; col++){
      mkCorners((2*row),35+4*col,0,35-4*col);
      pushMatrix();
          pdf.strokeWeight(15+ ((rowMax*row)+col)/3);
          drawGridOfShapes(shapeToDraw,4,70+((rowMax*row)+col)/2);
      popMatrix();
      translate(800,0);
    }
    popMatrix();
    translate(0,800);
  }

  endRecord();
  exit();
}
```

J24: zoom in on 23 (bottom left)



```
for (int row = 16; row<rowMax; row++){
  pushMatrix();
  for (int col = 0; col<5; col++){
    mkCorners((2*row),35+4*col,0,35-4*col);
    pushMatrix();
        pdf.strokeWeight(15+ ((rowMax*row)+col)/3);
        drawGridOfShapes(shapeToDraw,4,70+((rowMax*row)+col)/2);
    popMatrix();
    translate(800,0);
```

J25: zoom in on 23 (right side above bottom)



```
for (int row = 13; row<16; row++){
  pushMatrix();
  for (int col = 14; col<colMax; col++){
    mkCorners((2*row),35+4*col,0,35-4*col);
    pushMatrix();
        pdf.strokeWeight(15+ ((rowMax*row)+col)/3);
        drawGridOfShapes(shapeToDraw,4,70+((rowMax*row)+col)/2);
    popMatrix();
    translate(800,0);
```