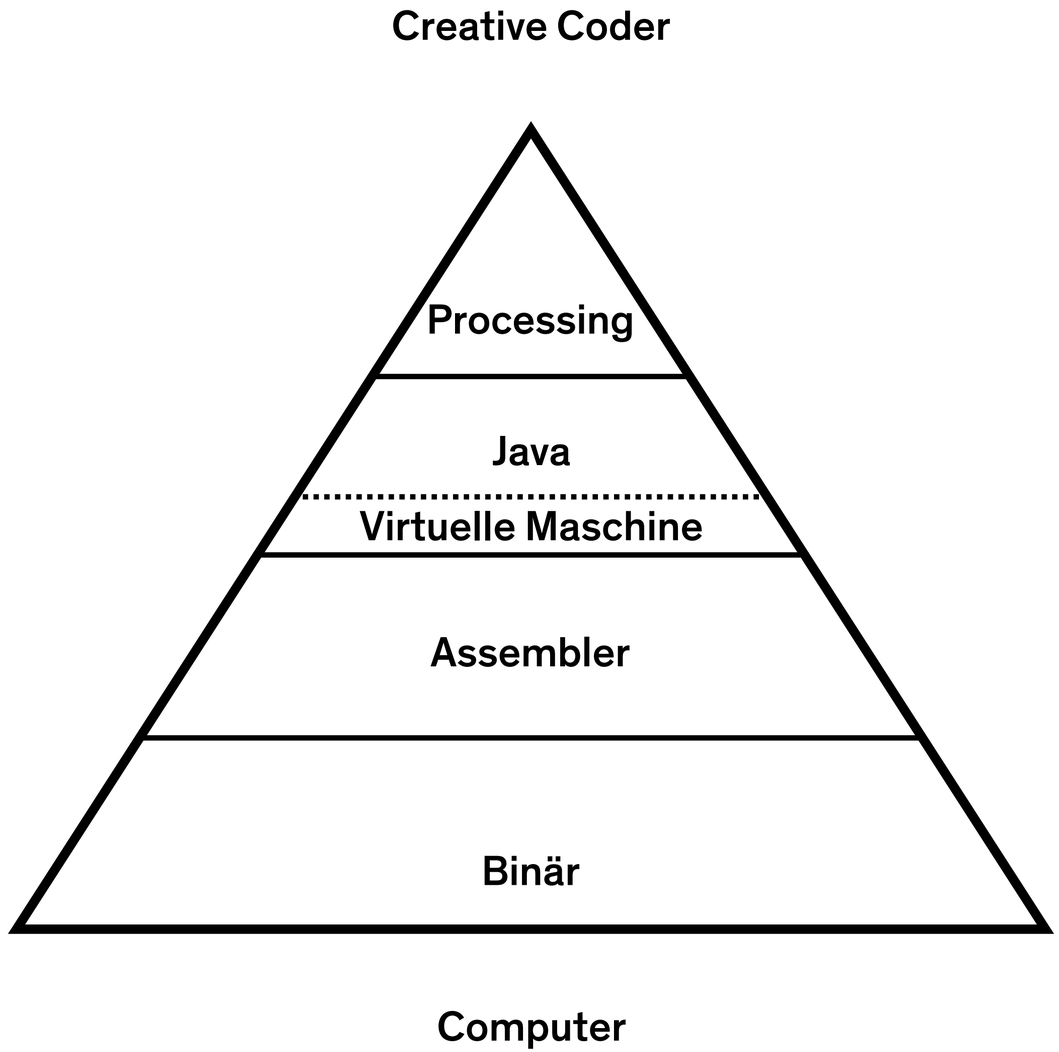
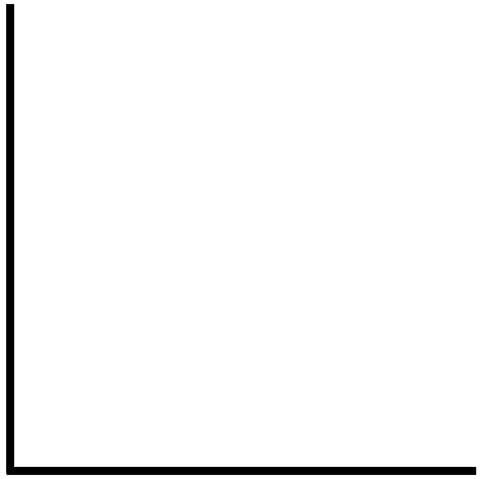
Creative Coding

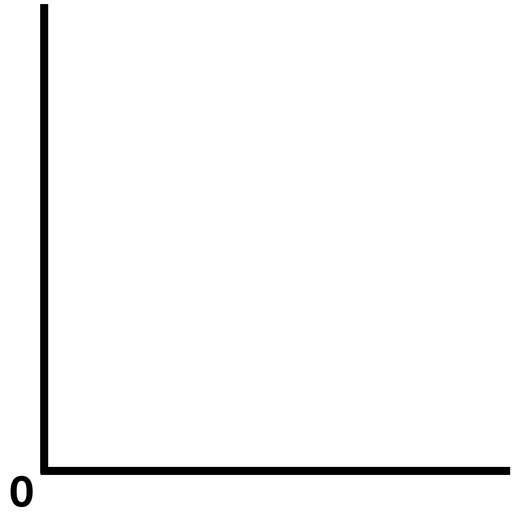
WiSe 2019/20 – Grundlagen



Processing IDE

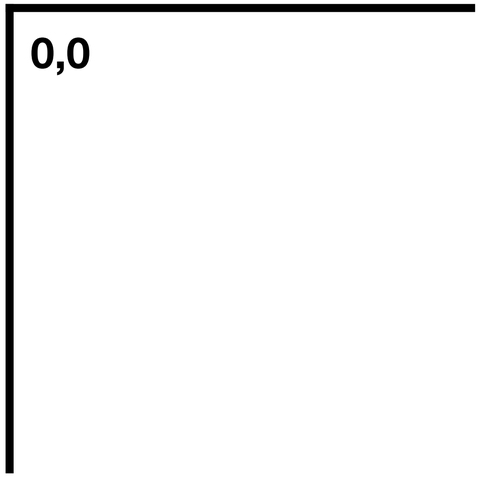


Orientierung

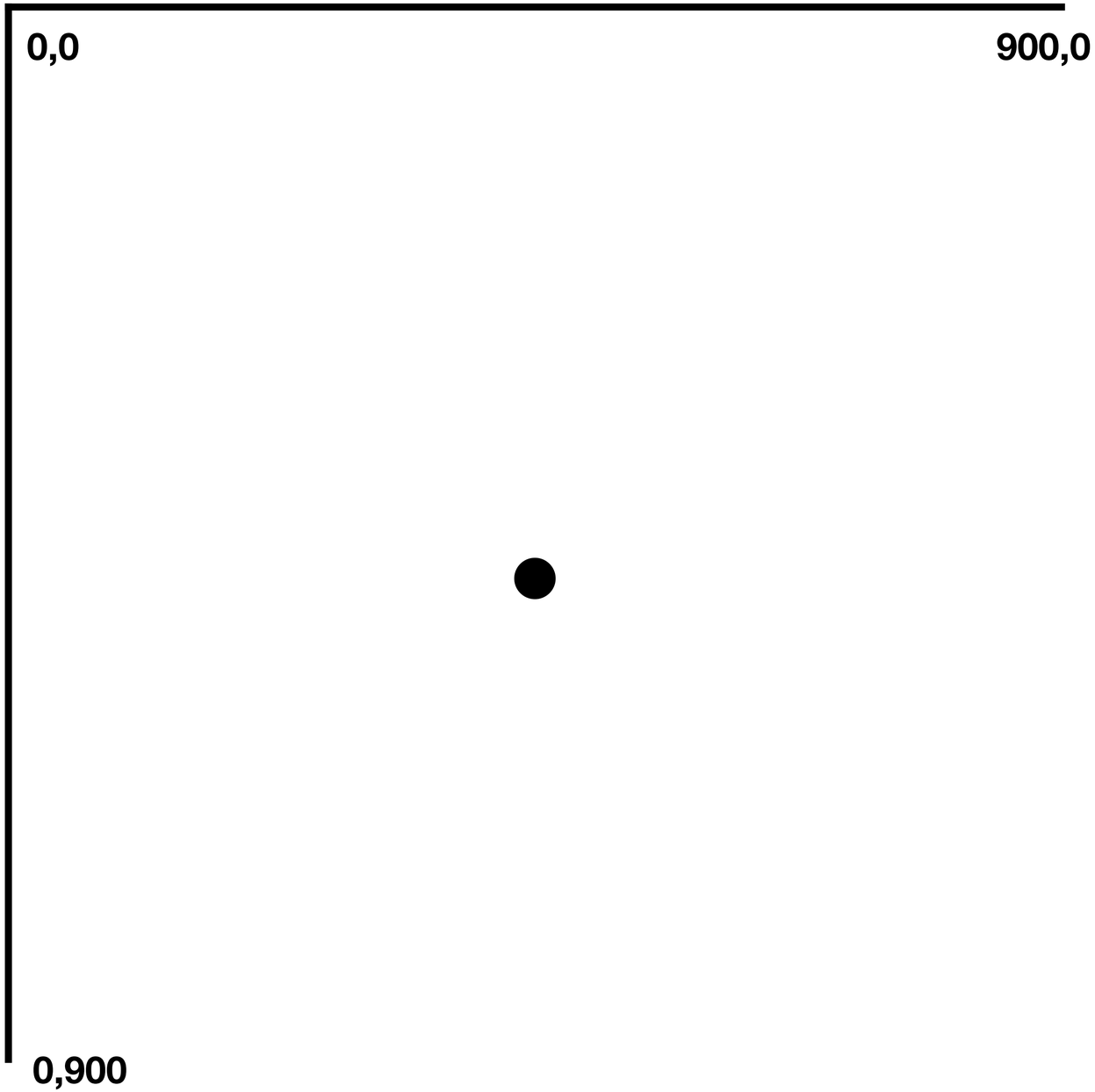


Orientierung

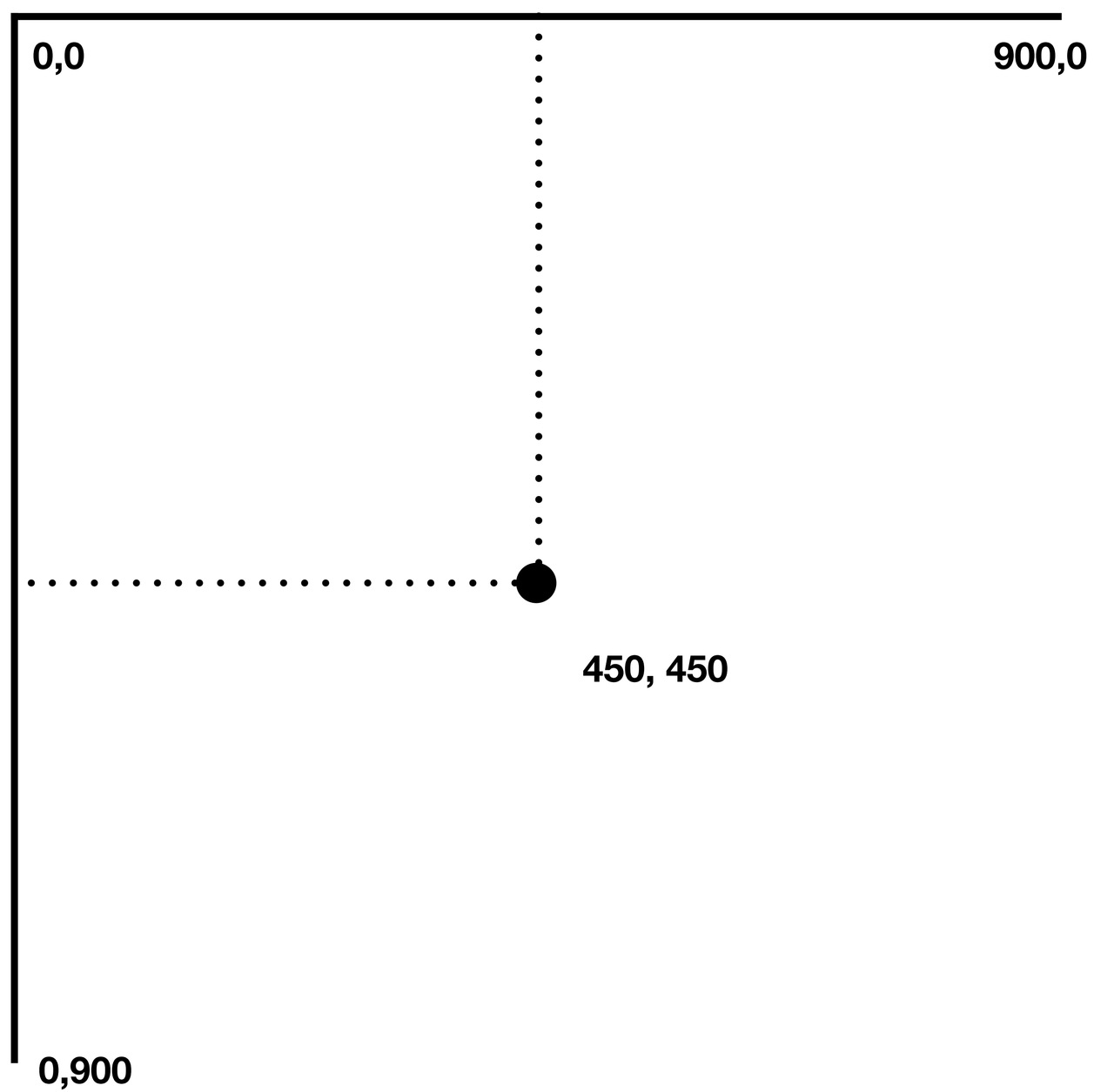
Orientierung



Orientierung



Orientierung



Orientierung

```
size()
```

Umgebung

```
void size(int height, int width);
void size(int height, int width, int renderer);
```

Umgebung

```
void size(int height, int width);
>> size(900, 900);
void size(int height, int width, int renderer);
>> size(900, 900, P3D); // P2D, P3D, FX2D, SVG, PDF
```

Umgebung

Datentyp Farbe

Farbe

```
color c1 = color(255, 0, 0);
color c2 = color(#FF0000);
color c3 = #FF0000;
```

Farbe

```
color c1 = color(255, 0, 0, 125);
color c2 = color(#FF0000, 125);
color c3 = #FF0000;
```

Farbe

```
color c1 = color(255, 0, 0, 125);
color c2 = color(#FF0000, 125);
color c3 = #FF0000;
color c4 = color(255);
```

Farbe

```
colorMode()
background()
fill()
stroke()
noFill()
noStroke()
```

Farbe

```
void colorMode(int mode);
void colorMode(int mode, float max);
background()
fill()
stroke()
noFill()
noStroke()
```

Farbe

```
void colorMode(int mode);
void colorMode(int mode, float max);
void colorMode(mode, float max1, float max2, float max3);
void colorMode(mode, float max1, float max2, float max3, float maxA);
background()
fill()
stroke()
noFill()
noStroke()
```

Farbe

```
void colorMode(int mode);
void colorMode(int mode, float max);
void colorMode(mode, float max1, float max2, float max3);
>>colorMode(HSB, 360, 100, 100);
void colorMode(mode, float max1, float max2, float max3, float maxA);
background()
fill()
stroke()
noFill()
noStroke()
```

Farbe

```
void background(color rgb);
void background(color rgb, float alpha);
void background(float gray);
void background(float gray, float alpha);
void background(float a, float b, float c);
void background(float a, float b, float c, float alpha);
void background(PImage p);

>> background(0);
>> background(255, 0, 0);
```

Farbe

```
void fill(color rgb);
void fill(color rgb, float alpha);
void fill(float gray);
void fill(float gray, float alpha);
void fill(float a, float b, float c);
void fill(float a, float b, float c, float alpha);

>> fill(0);
>> fill(255, 0, 0);
```

Farbe

```
void stroke(color rgb);
void stroke(color rgb, float alpha);
void stroke(float gray);
void stroke(float gray, float alpha);
void stroke(float a, float b, float c);
void stroke(float a, float b, float c, float alpha);

>> stroke(0);
>> stroke(255, 0, 0);
```

Farbe

```
void colorMode(int mode);
void colorMode(int mode, float max);
void colorMode(mode, float max1, float max2, float max3);
>>colorMode(HSB, 360, 100, 100);
void colorMode(mode, float max1, float max2, float max3, float maxA);
void background(color c);
void fill(color c);
void stroke(color c);
noFill()
noStroke()
```

Farbe

Grafische Primitive

```
point();
```


Grafische Primitive

```
point(Parameterliste);
```


Grafische Primitive

```
point(float x, float y);
point(float x, float y, float z);
```

Grafische Primitive

```
void point(float x, float y);
void point(float x, float y, float z);
```

Grafische Primitive

```
point(450, 450);
```


Grafische Primitive

```
point();
line();
triangle();

square();
rect();
quad();

circle();
ellipse();

arc();
```

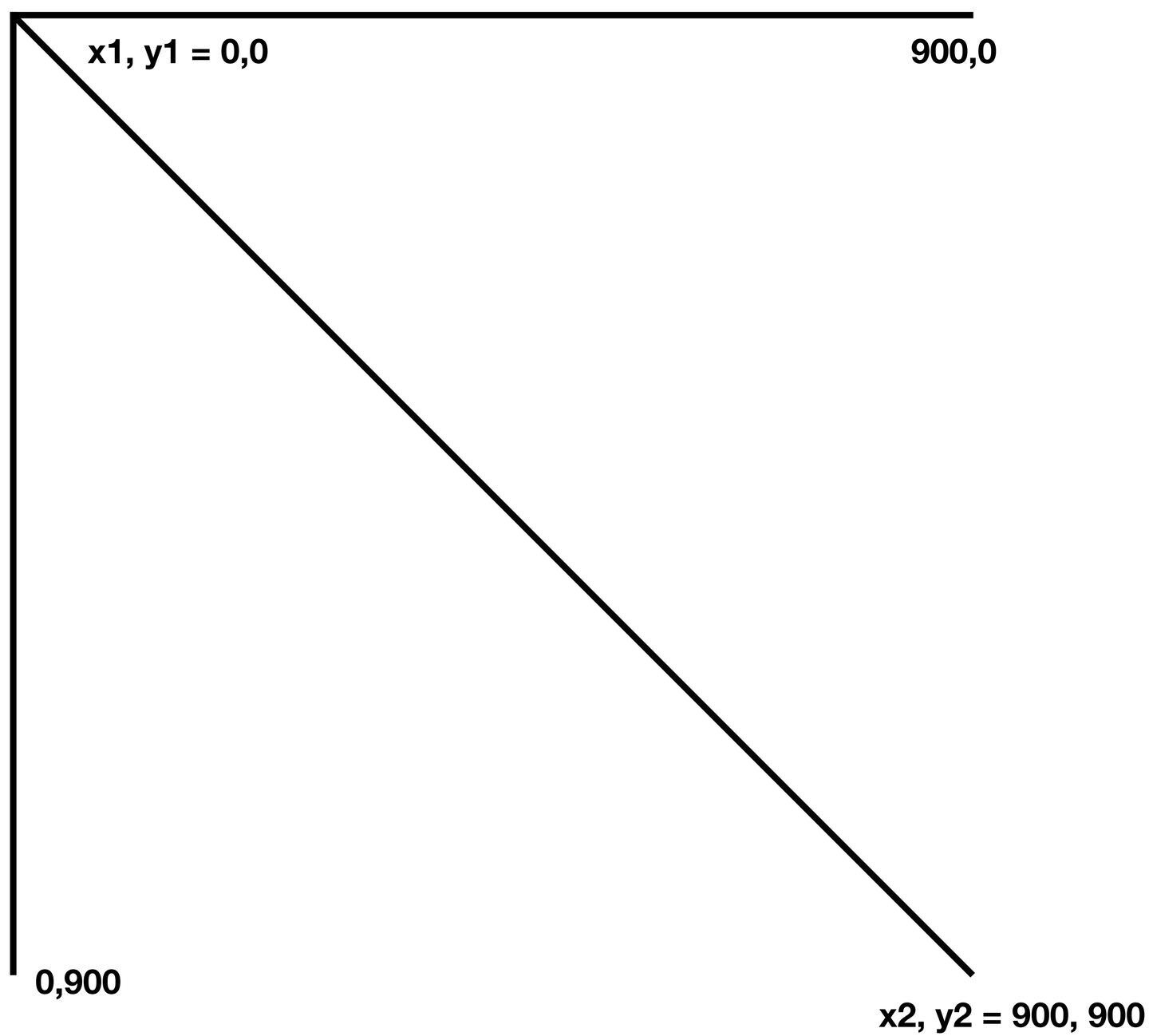
Grafische Primitive

```
void line(float x1, float y1, float x2, float y2);
void line(float x1, float y1, float z1, float x2, float y2, float z2);
```

Grafische Primitive

```
line(0, 0, 900, 900);
```


Grafische Primitive



Grafische Primitive

```
void triangle(float x1, float y1, float x2, float y2, float x3, float y3);
```


Grafische Primitive

```
square();
quad();
rect();
```


Grafische Primitive

```
void square(float x, float y, float extent);
void rect(float x, float y, float width, float height);
quad();
```

Grafische Primitive

```
void square(float x, float y, float extent);
void rect(float x, float y, float width, float height);
void rect(float x, float y, float width, float height, float r);
void rect(x, y, width, height, tl, tr, br, bl);
quad();
```

Grafische Primitive

```
void square(float x, float y, float extent);
void rect(float x, float y, float width, float height);
void rect(float x, float y, float width, float height, float r);
void rect(x, y, width, height, tl, tr, br, bl);
void quad(x1, y1, x2, y2, x3, y3, x4, y4);
```

Grafische Primitive

```
void cicle(float x, float y, float extent);
void ellipse(float x, float y, float width, float height);
```


Grafische Primitive

```
arc();
```


Grafische Primitive

```
void arc(x, y, width, height, float start, float stop);
```


Grafische Primitive

```
void arc(450, 450, 50, 50, float start, float stop);
```


Grafische Primitive

```
// float radians(float degrees);
void arc(450, 450, 50, 50, radians(0), radians(180));
```

Grafische Primitive

```
void arc(450, 450, 50, 50, 0, HALF_PI);
```


Grafische Primitive

```
ellipseMode()
rectMode()
strokeCap()
strokeJoin()
strokeWeight()
```

Attribute

```
void ellipseMode(int mode);
void rectMode(int mode);


strokeCap()
strokeJoin()
strokeWeight()
```

Attribute

```
void ellipseMode(int mode); // CENTER, RADIUS, CORNER oder CORNERS
void rectMode(int mode); // CORNER, CENTER, RADIUS oder CORNERS


strokeCap()
strokeJoin()
strokeWeight()
```

Attribute

```
void ellipseMode(int mode); // CENTER, RADIUS, CORNER oder CORNERS
void rectMode(int mode); // CORNER, CENTER, RADIUS oder CORNERS

void strokeCap(int cap); // SQUARE, PROJECT oder ROUND
strokeJoin()
strokeWeight()
```

Attribute

```
void ellipseMode(int mode); // CENTER, RADIUS, CORNER oder CORNERS
void rectMode(int mode); // CORNER, CENTER, RADIUS oder CORNERS

void strokeCap(int cap); // SQUARE, PROJECT oder ROUND
void strokeJoin(int join); // MITER, BEVEL oder ROUND
strokeWeight()
```

Attribute

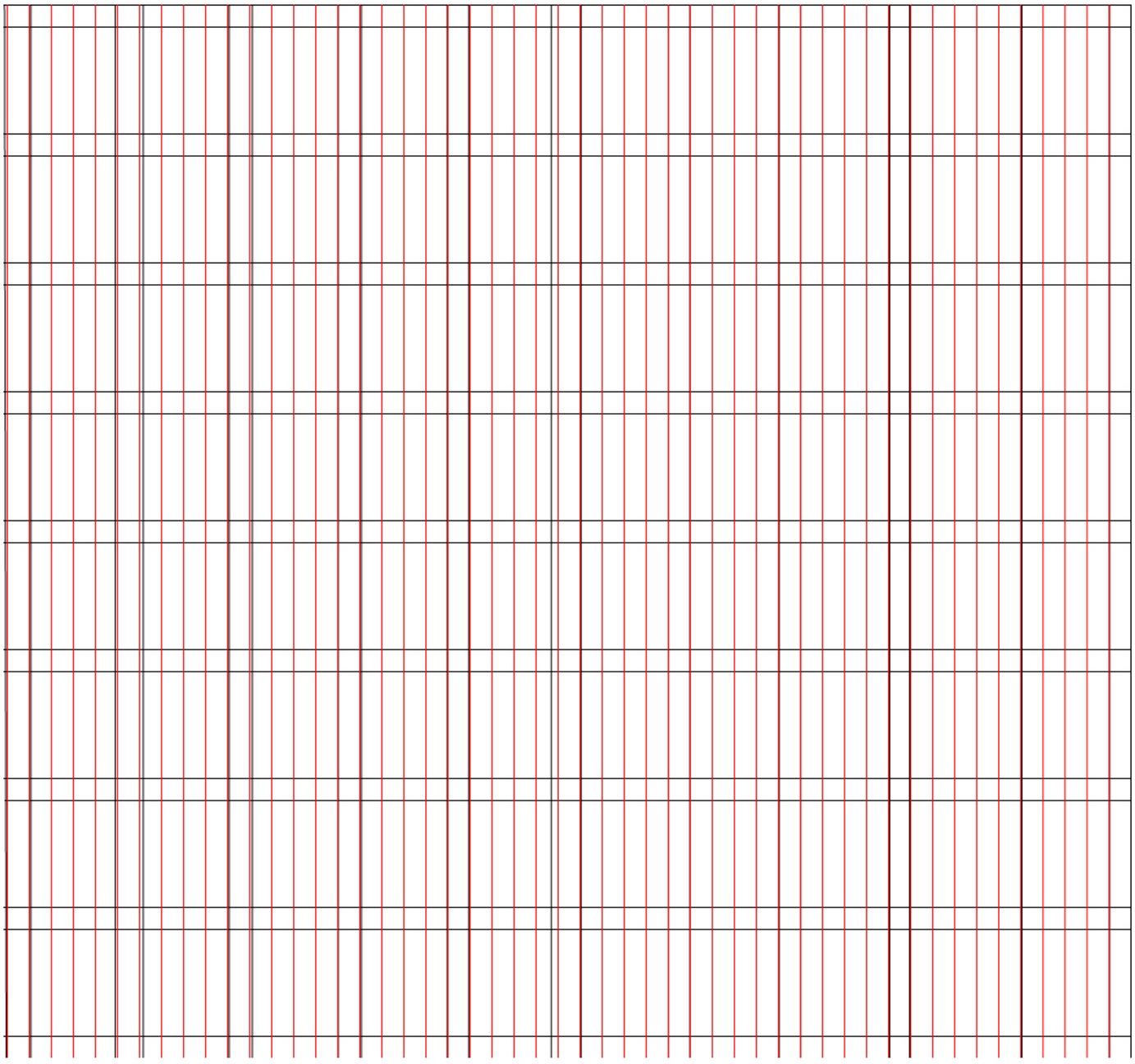
```
void ellipseMode(int mode); // CENTER, RADIUS, CORNER oder CORNERS
void rectMode(int mode); // CORNER, CENTER, RADIUS oder CORNERS

void strokeCap(int cap); // SQUARE, PROJECT oder ROUND
void strokeJoin(int join); // MITER, BEVEL oder ROUND
void strokeWeight(int weight); // in Pixeln
```

Attribute

Das Raster

Das Raster

Das Raster



Dimension

1D

****

1D

2D

****

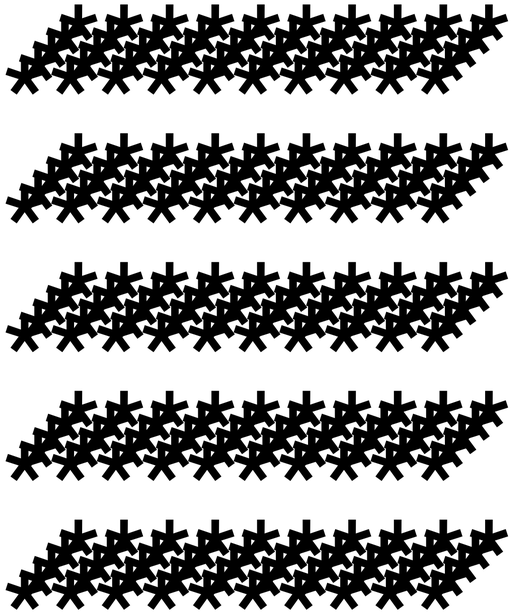
****

****

****

****

3D



3D

Iteration

```
for (Initialisierung; Test; Fortsetzung) {
    Anweisung
}
```

for-Schleife

```
for (int i = 0; Test; Fortsetzung) {
    Anweisung
}
```

for-Schleife

```
for (int i = 0; i < 10; Fortsetzung) {
    Anweisung
}
```

for-Schleife

```
for (int i = 0; i < 10; i++) {
    Anweisung
}
```

for-Schleife

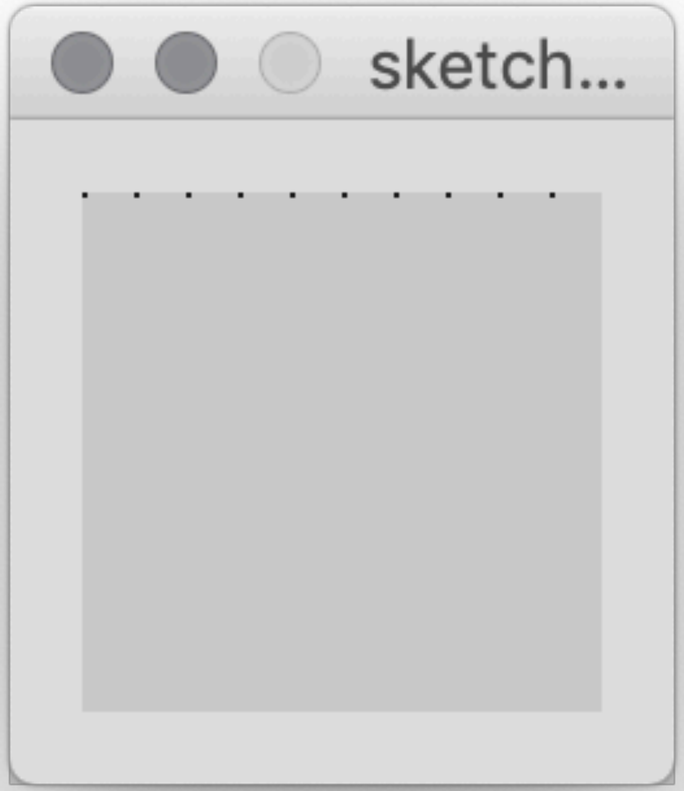
```
for (int i = 0; i < 10; i++) {
    println("iteration= "+ i);
}

// konsole:
iteration= 0
iteration= 1
iteration= 2
iteration= 3
iteration= 4
iteration= 5
usw. usf
*/
```

for-Schleife

```
for (int x = 0; x < 10; x++) {
    point(x * 10, 0);
}
```

Iteration 1D

```
for (int x = 0; x < 10; x++) {
    point(x * 10, 0);
}
```

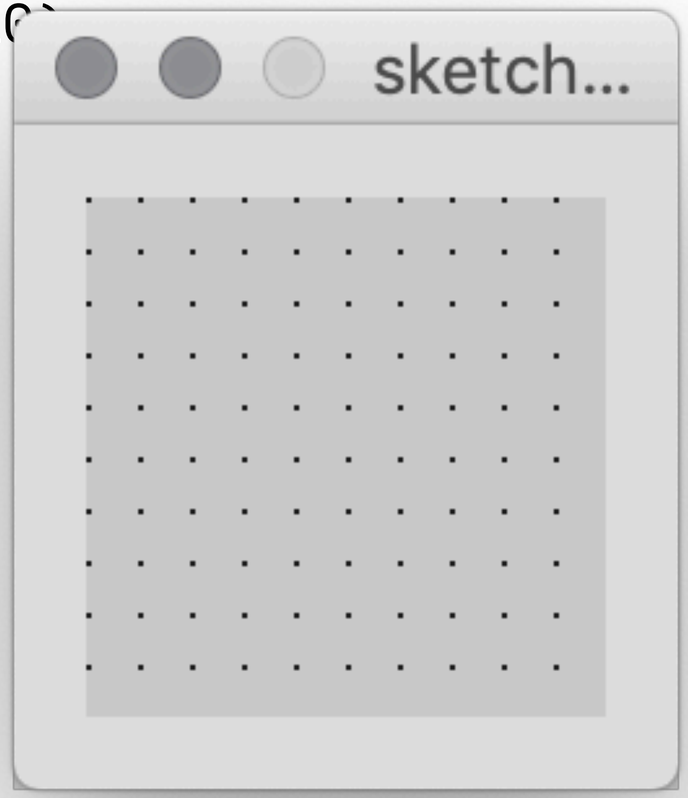


Iteration 1D

```
for (int y = 0; y < 10; y++) {
    for (int x = 0; x < 10; x++) {
        point(x * 10, y * 10);
    }
}
```

Iteration 2D

```
for (int y = 0; y < 10; y++) {
    for (int x = 0; x < 10; x++) {
        point(x * 10, y * 10);
    }
}
```

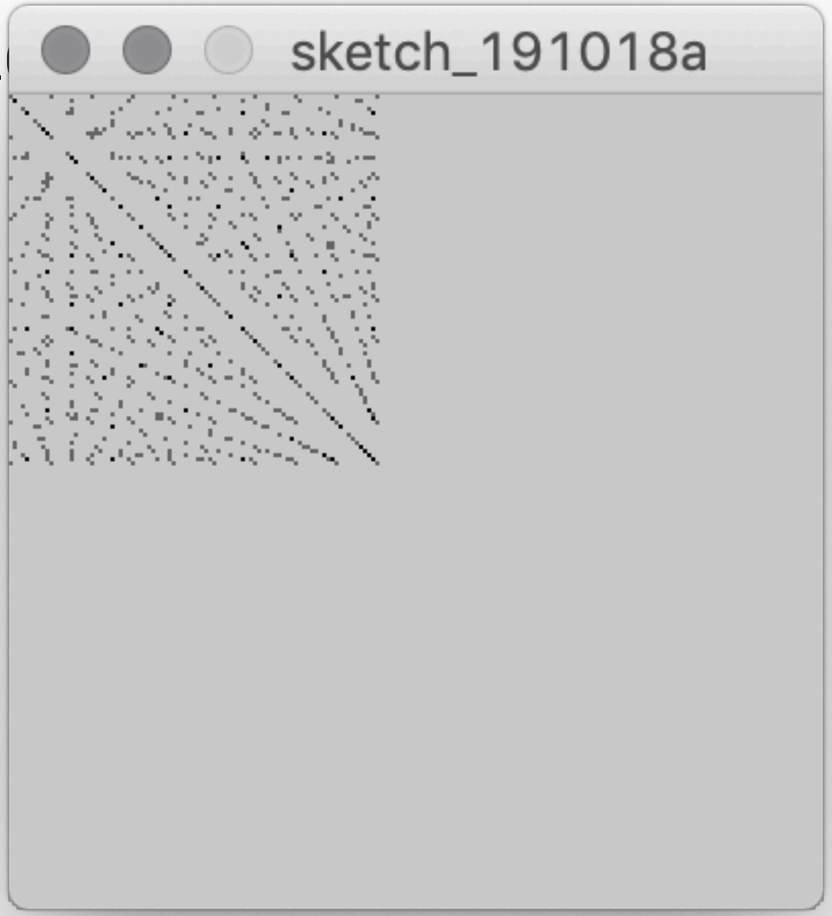


Iteration 2D

```
size(200, 200, P3D);
for (int z = 0; z < 10; z++) {
    for (int y = 0; y < 10; y++) {
        for (int x = 0; x < 10; x++) {
            point(x * 10, y * 10, z * 10);
        }
    }
}
```

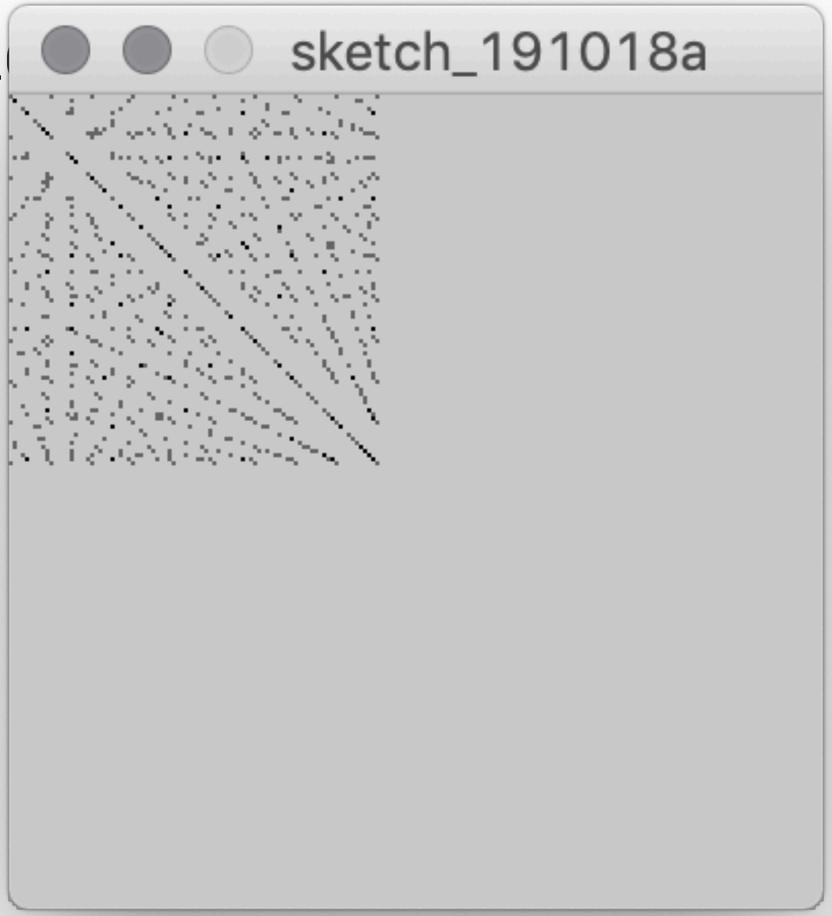
Iteration 3D

```
size(200, 200, P3D);
for (int z = 0; z < 10; z++) {
    for (int y = 0; y < 1
        for (int x = 0; x
            point(x * 10,
        }
    }
}
```



Iteration 3D

```
size(200, 200, P3D);
for (int z = 0; z < 10; z++) {
    for (int y = 0; y < 1
        for (int x = 0; x
            point(x * 10,
        }
    }
}
```



Iteration 3D

Entwerft ein regelmäßiges Raster, bestehend aus min. 3 × 3 Rasterelementen.
Jedes Rasterelement besteht aus min. 3 geometrischen Formen.
Zeichnet jedes Element einzeln. (Keine for-Schleife)

10 Minuten Übung

Fingerübung

Zeichenfläche / Canvas

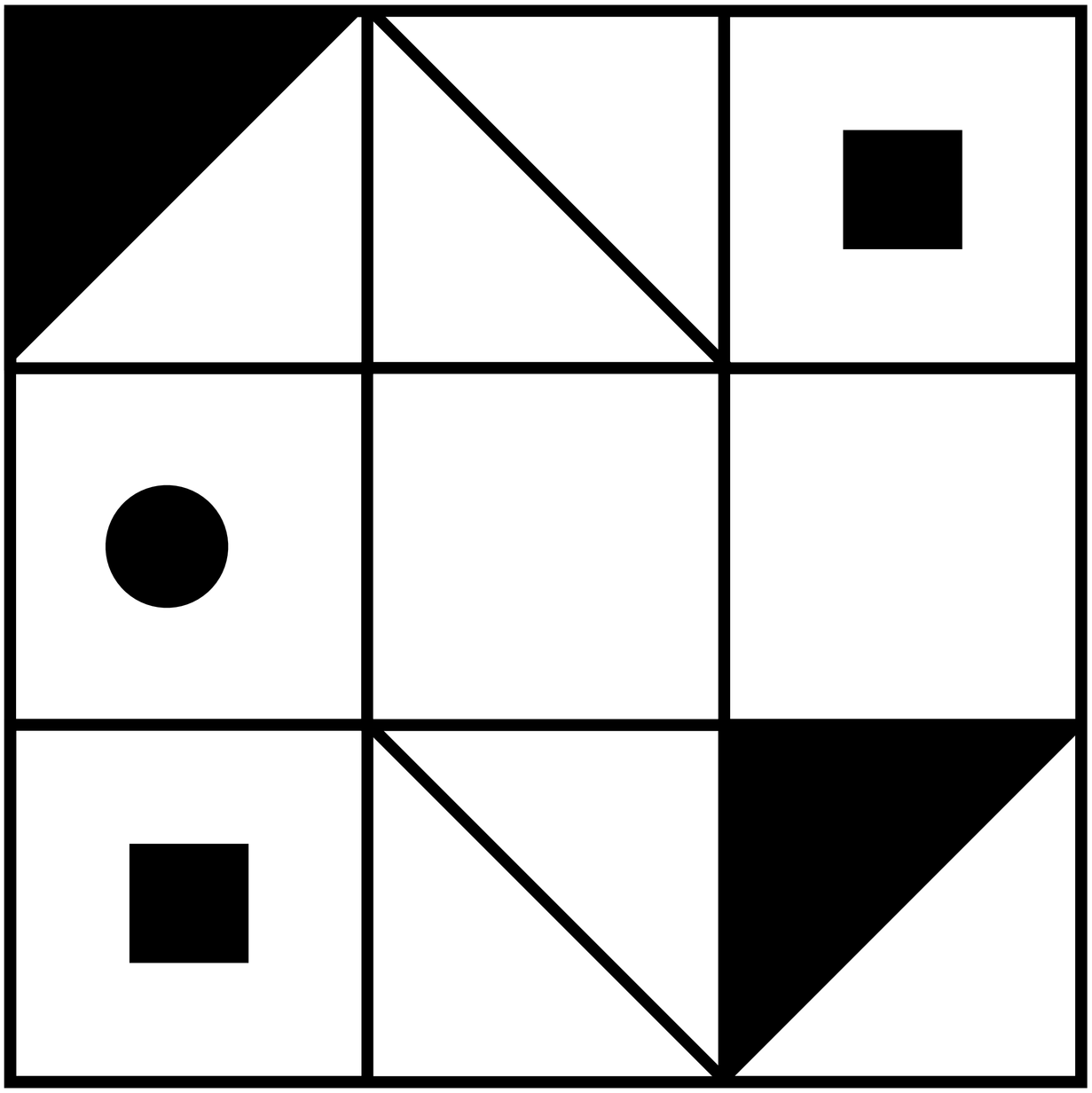
size()

Folgende Befehle können benutzt werden:

arc()
ellipse()
line()
point()
quad()
rect()
triangle()

Folgende Attribute min. einmal erproben:

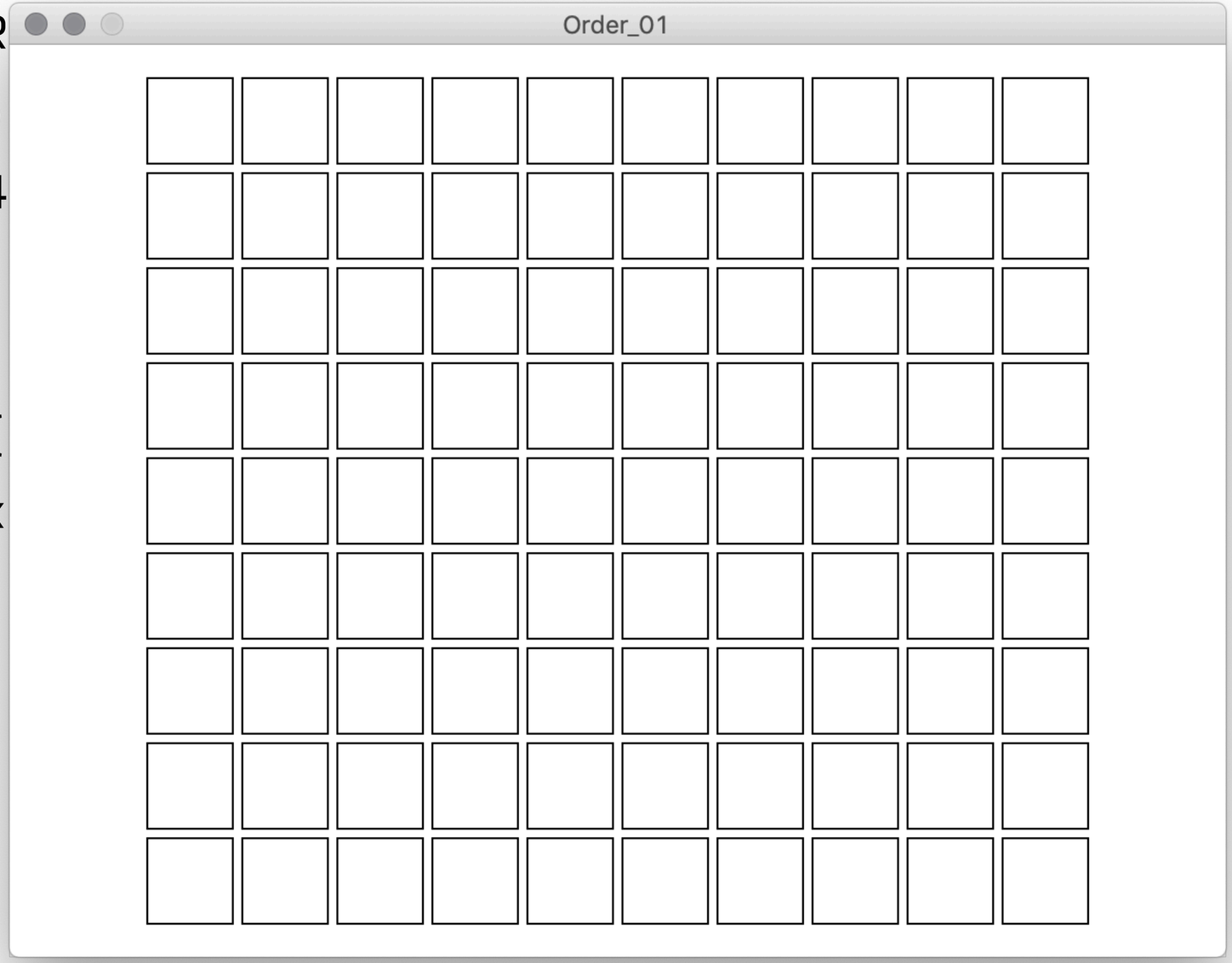
ellipseMode()
rectMode()
strokeCap()
strokeJoin()
strokeWeight()



Creative Coding

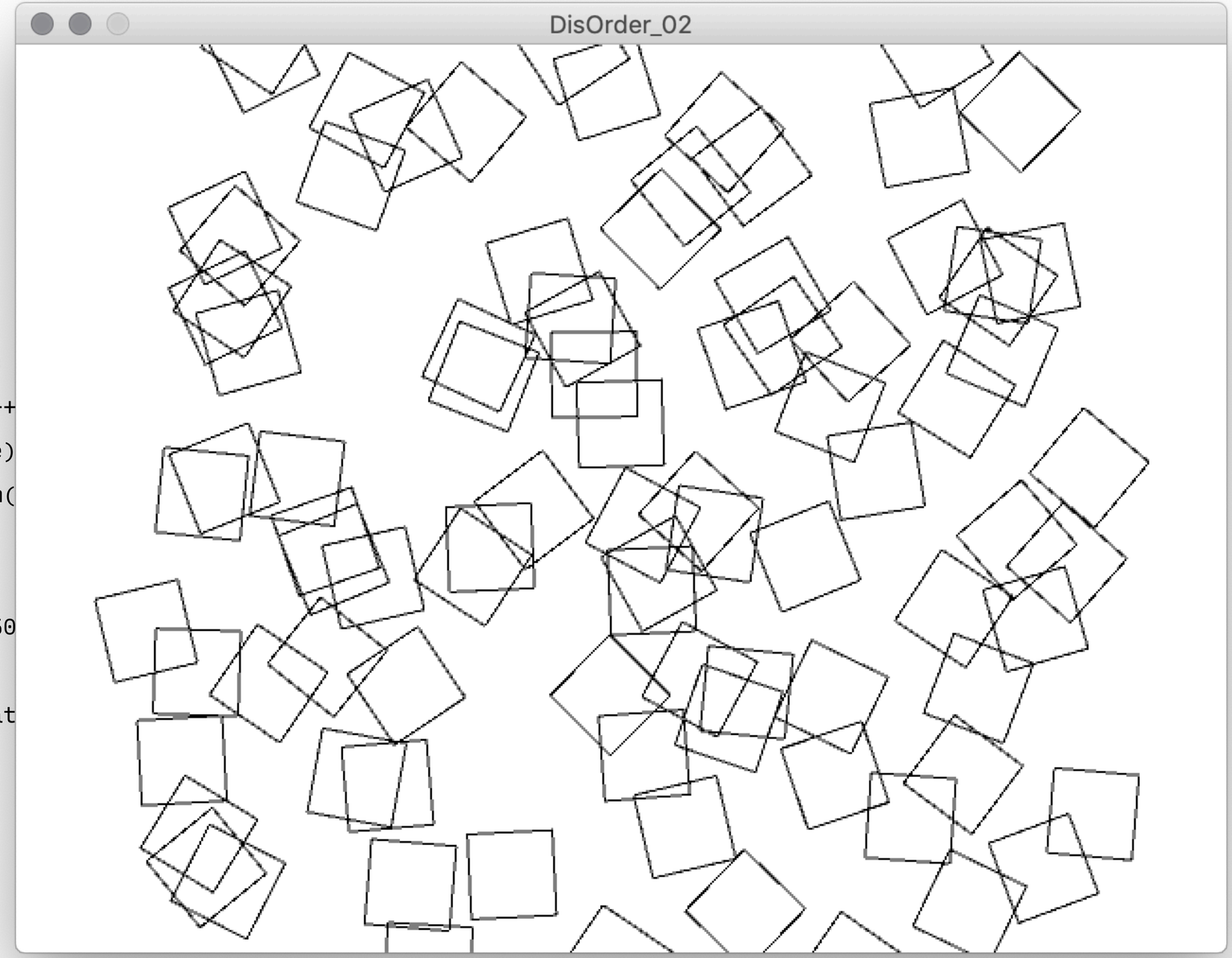
```
size(640, 480);
rectMode(CENTER);
background(255);
translate(95, 40);
for (int y = 0; y < 9; y++) {
  for (int x = 0; x < 10; x++) {
    pushMatrix();
    translate(x * 50, y * 50);
    rect(0, 0, 45, 45);
   popMatrix();
  }
}
```

```
size(640, 480);
rectMode(CENTER
background(255)
translate(95, 4
for (int y = 0;
  for (int x =
    pushMatrix(
    translate(x
    rect(0, 0,
   popMatrix();
  }
}
```



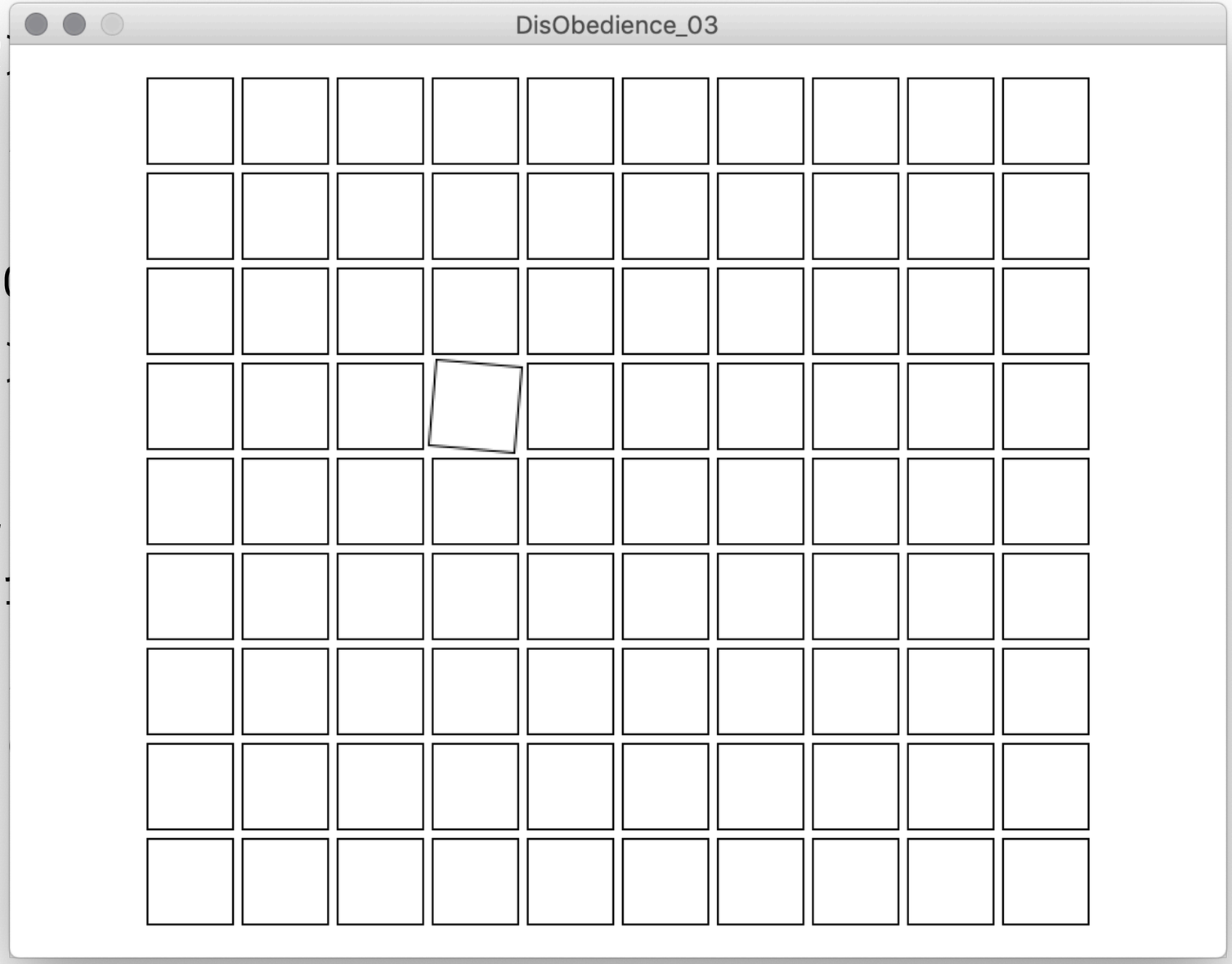
```
boolean mToggle = false;


void setup() {
  size(640, 480, P3D);
 rectMode(CENTER); smooth();
}
void draw() {
  background(255);
  translate(95, 40);
  for (int y = 0; y < 9; y++) {
   for (int x = 0; x < 10; x++) {
    pushMatrix(); if (mToggle) {
    translate(x * 50 + random(-50, 50), y * 50 + random(-50, 50));
    rotate(random(0, PI)); }
    else {
      translate(x * 50, y * 50);
    }
    rect(0, 0, 45, 45); popMatrix();
    }
  }
  noLoop(); }
  void keyPressed() {
    loop();
    mToggle = !mToggle;
  }
```

```
boolean mToggle = false;


void setup() {
  size(640, 480, P3D);
 rectMode(CENTER); smooth();
}
void draw() {
  background(255);
  translate(95, 40);
  for (int y = 0; y < 9; y++)
   for (int x = 0; x < 10; x++
    pushMatrix(); if (mToggle)
    translate(x * 50 + random(
    rotate(random(0, PI)); }
    else {
      translate(x * 50, y * 50
    }
    rect(0, 0, 45, 45); popMat
    }
  }
  noLoop(); }
  void keyPressed() {
    loop();
    mToggle = !mToggle;
  }
```



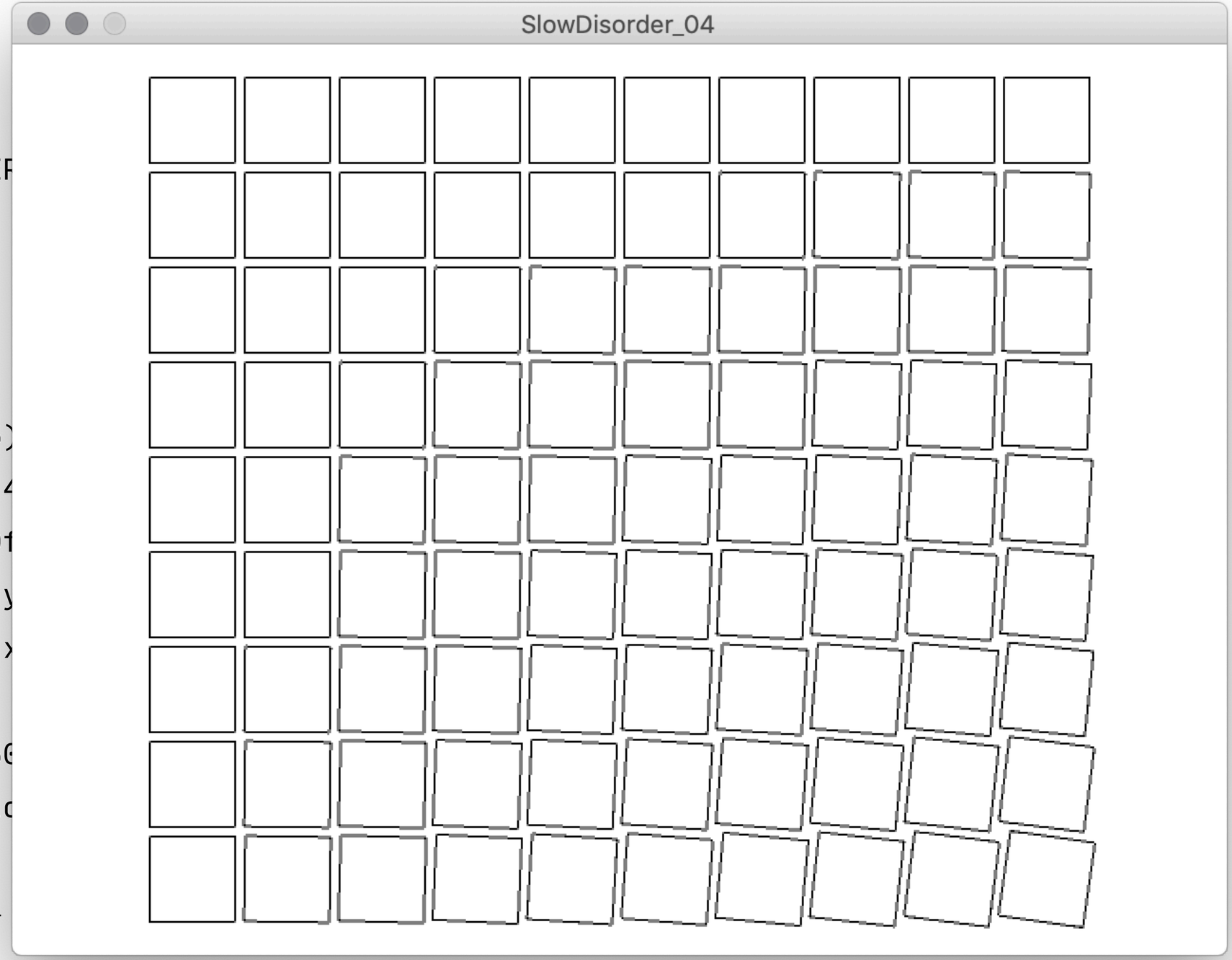
```
size(640, 480);
rectMode(CENTER);
background(255); translate(95, 40);
for (int y = 0; y < 9; y++) {
  for (int x = 0; x < 10; x++) {
    pushMatrix();
    translate(x * 50, y * 50);
    if (y==3 && x==3) {
      rotate(0.1); }
      rect(0, 0, 45, 45);
      popMatrix();
    }
  }
```

```
size(640, 480);
rectMode(CENTER)
background(255)
for (int y = 0;
  for (int x = 0
    pushMatrix()
    translate(x
    if (y==3 &&
      rotate(0.1
      rect(0, 0
      popMatrix
    }
  }
```



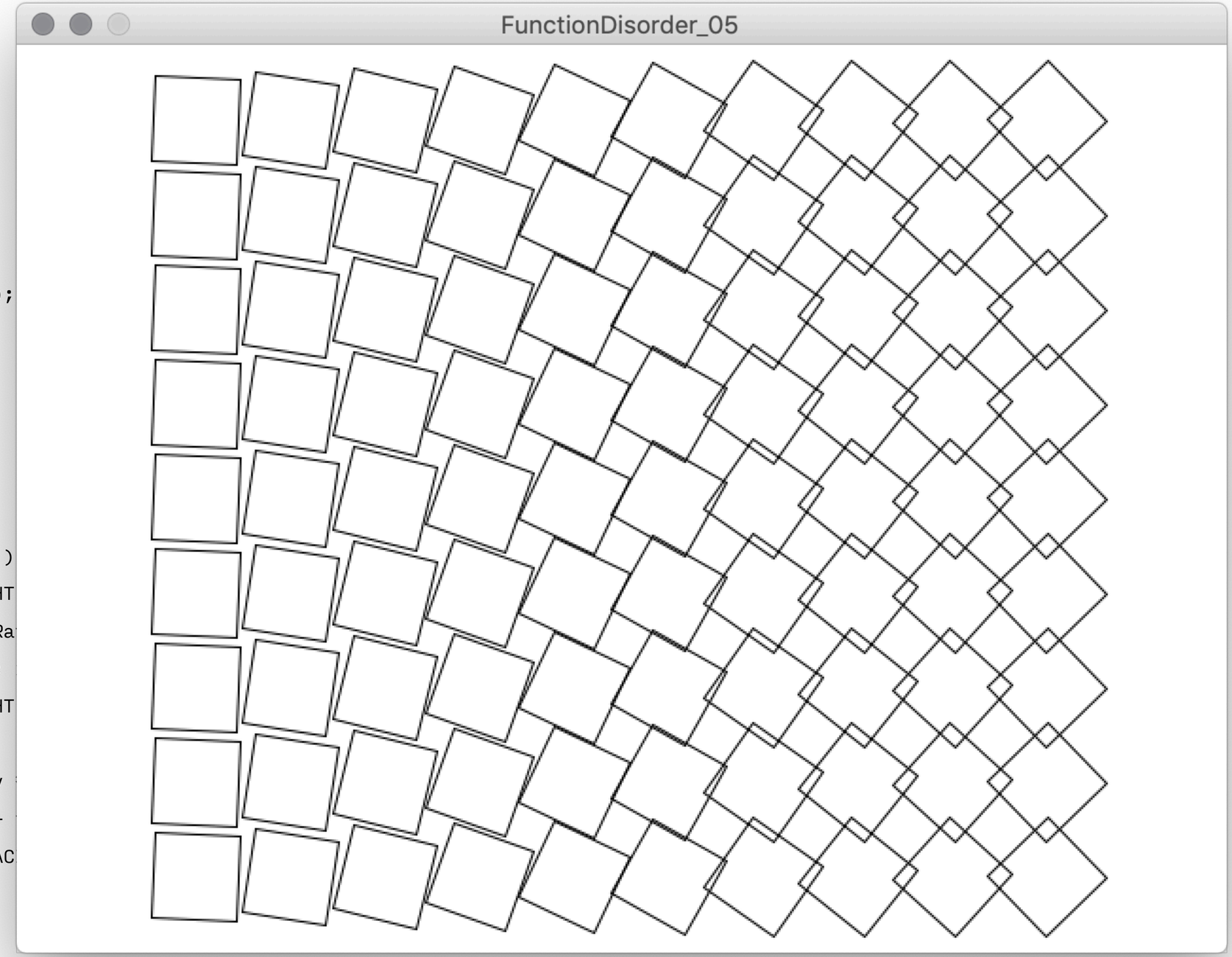
```
float mRotation;
void setup() {
  size(640, 480, P3D);
  rectMode(CENTER);
  noFill();
smooth();
}
void draw() {
  background(255);
  translate(95, 40);
mRotation += 1.0f / frameRate;
for (int y = 0; y < 9; y++) {
for (int x = 0; x < 10; x++) {
  pushMatrix();
  translate(x *50, y * 50);
  rotate(mRotation * 0.0001f * x * y);
  rect(0, 0, 45, 45);
  popMatrix(); }
} }
```

```
float mRotation;
void setup() {
  size(640, 480,
  rectMode(CENTER
  noFill();
smooth();
}
void draw() {
  background(255)
  translate(95, 4
mRotation += 1.0f
for (int y = 0; y
for (int x = 0; x
  pushMatrix();
  translate(x *50
  rotate(mRotatio
  rect(0, 0, 45,
  popMatrix(); }
} }
```



```
float mRotation;
final int WIDTH = 10;
final int HEIGHT = 9;
final float SCALE = 50.0f;
final float SPACING = 5.0f;

void setup() {
  size(640, 480);
  rectMode(CENTER); noFill();
  smooth();
}

void draw() {
  background(255);
  translate(
    ( width - SCALE * WIDTH ) / 2 + SCALE / 2,
    ( height - SCALE * HEIGHT) / 2 + SCALE / 2);
  mRotation += 1.0f / frameRate;
  for (int x = 0; x < WIDTH; x++) {
   for (int y = 0; y < HEIGHT; y++) {
      pushMatrix();
      translate(x * SCALE, y * SCALE);
      rotate(sin(mRotation + float(x) / WIDTH - float(Y) / HEIGHT));
      rect(0, 0, SCALE - SPACING, SCALE - SPACING);
      popMatrix();
    }
  }
}
```

```
float mRotation;
final int WIDTH = 10;
final int HEIGHT = 9;
final float SCALE = 50.0f;
final float SPACING = 5.0f;

void setup() {
  size(640, 480);
  rectMode(CENTER); noFill();
  smooth();
}

void draw() {
  background(255);
  translate(
    ( width - SCALE * WIDTH )
    ( height - SCALE * HEIGHT
  mRotation += 1.0f / frameRa
  for (int x = 0; x < WIDTH;
   for (int y = 0; y < HEIGHT
      pushMatrix();
      translate(x * SCALE, y
      rotate(sin(mRotation +
      rect(0, 0, SCALE - SPAC
      popMatrix();
    }
  }
}
```

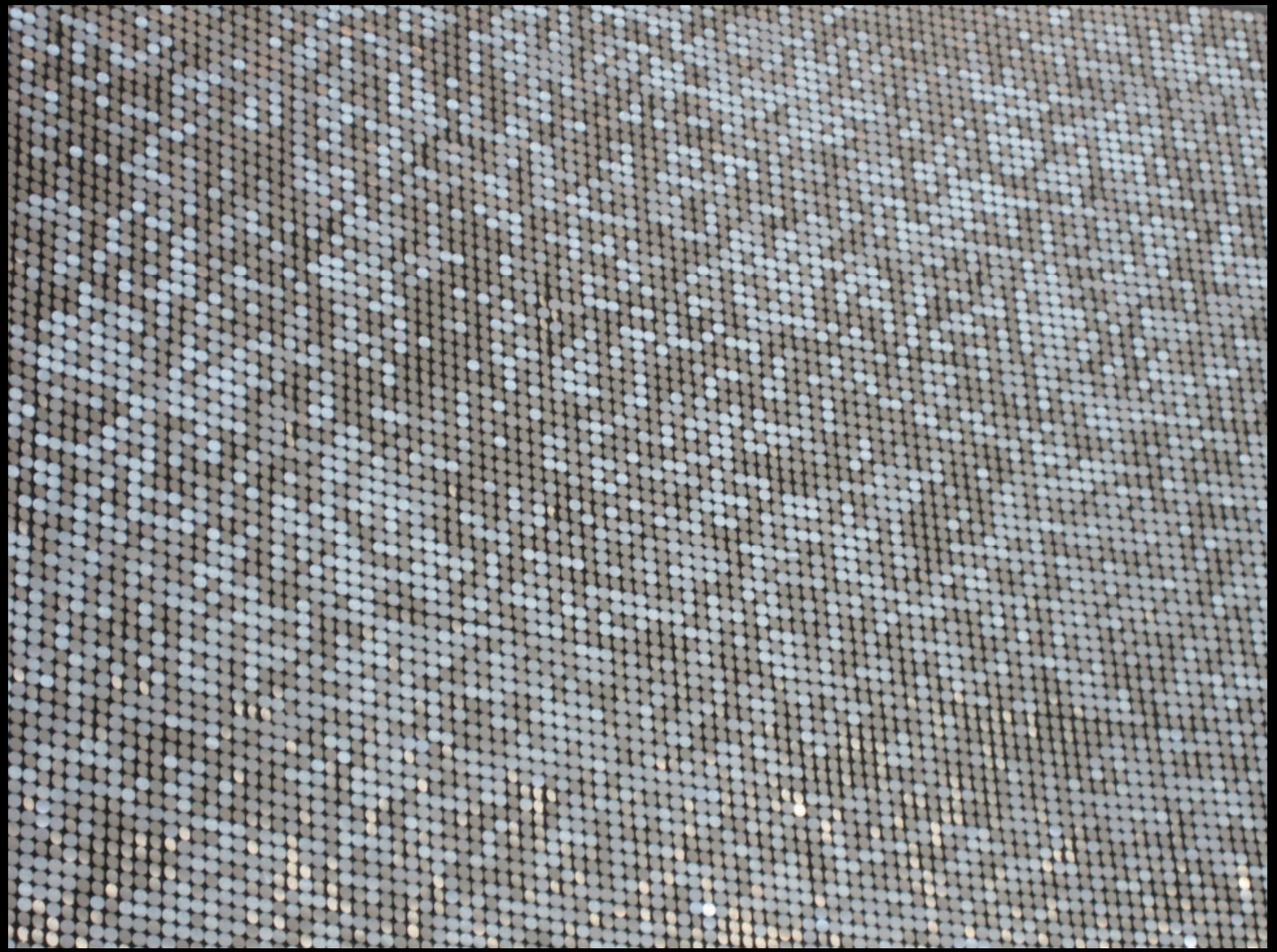


Creative Coding

Creative Coding

Creative Coding

Creative Coding



Creative Coding

Creative Coding

Entwerft 1, 2 oder 3D iterative Raster (3 Stück)
Ordnet diesem einen Begriff oder Konzept zu (oder umgekehrt)

(Für die, die es ein wenig schwerer haben wollen)
– ArrayListen
– Klassen
– PGraphics
– Github

Aufgabe 2 – Bis 08.11.2019

Dokumentiert die Ausgabe + Code.

Ausdrucke auf hochwertigem A4 Papier (präsentierbar)

https://processing.org/reference/

Aufgabe 2 – Bis 08.11.2019