# P.3

# Type

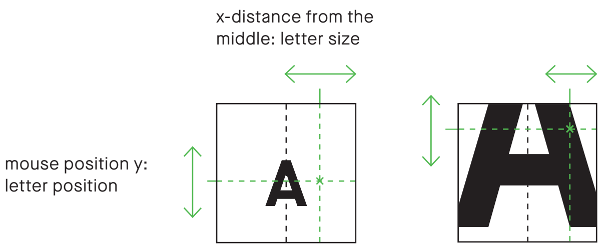
In the Shape chapter, we demonstrated how shapes can be generated using the principles of repetition (grid), iteration (agents), and interaction (drawing). This chapter is devoted to a special kind of form that is also extremely important in design: typography. Using various methods—from the visual analysis of a text to the outlines of a character—typography will be viewed in the following examples in the context of generative design.

## P.3.0 Hello, type

Letters become spaces. In generating a vector-based font, you can directly influence numerous parameters and design with letters in time and space. Traces of the emergence of the character and the interactive manipulation of its size and position can be made visible.

**→ P_3_0_01**

The size of the letter is controlled with the horizontal movement of the mouse, and its vertical movement moves the letters up and down. The letter leaves a trail when the mouse button is held down.


x-distance from the middle: letter size
mouse position y: letter position
A
A

→ P_3_0_01 The letter leaves the tracks of its changes, then becomes unrecognizable and generates new forms.

1
```
var font = "sans-serif";
var letter = "A";
```

```
function setup() {
  createCanvas(windowWidth, windowHeight);
  background(255);
  fill(0);

  textFont(font);
  textAlign(CENTER, CENTER);
}
```
2

3
```
function mouseMoved() {
  clear();
  textSize((mouseX - width / 2) * 5 + 1);
  text(letter, width / 2, mouseY);
}
```

4
```
function mouseDragged() {
  textSize((mouseX - width / 2) * 5 + 1);
  text(letter, width / 2, mouseY);
}
```

| | |
|---|---|
| Mouse: | Position x: Size |
| | Position y: Position |
| | Drag: Draw |
| Keys: | A–Z: Letter selection |
| | CTRL: Save image |

1 The name of the font to use is saved in the variable `font`.

2 The function `textFont()` makes it the current font. The horizontal and vertical alignment can be specified with `textAlign()`.

3 When the mouse is moved, the letter size changes according to the value of the horizontal mouse position. The `letter` is positioned horizontally in the middle of the display window `width / 2`, vertically in the position `mouseY`, and displayed using the `text()` command.

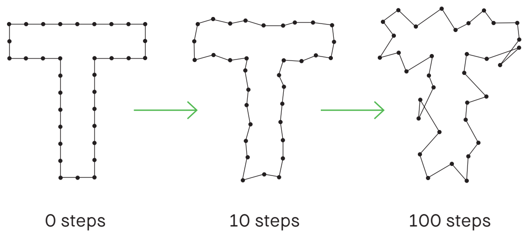
4 This also occurs when the mouse is moved with the mouse button held down, but now the background does not get a new color and the letter leaves a trail.

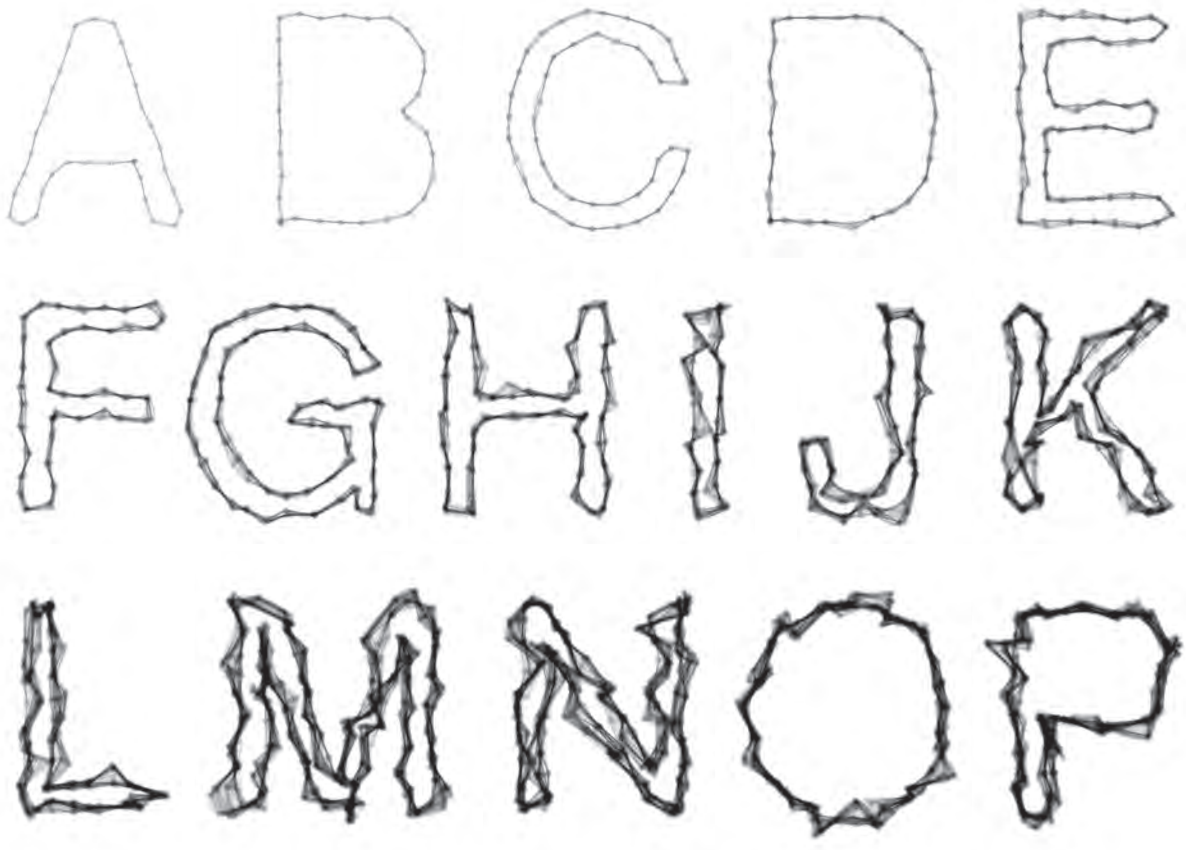
## P.3.2.3 Font outline from agents

How long is a letter recognizable as such? In this example, the outlines of a letter serve as the source shape. Each individual nodal point moves like a dumb agent. Over time, the letter becomes illegible and is transformed into something new.

**→ P_3_2_3_01**

Points are again generated from a font outline. Each point becomes an independent dumb agent but remains connected to its neighbor.





→ P_3_2_3_01 The more time that passes without a key being pressed, the more a character becomes deformed.

```
function draw() {
  ...
  translate(letterX, letterY);          [1]

  danceFactor = 1;
  if (mouseIsPressed && mouseButton == LEFT)          [2]
           danceFactor = map(mouseX, 0, width, 0, 3);

  if (pnts.length > 0) {
    for (var i = 0; i < pnts.length; i++) {
      pnts[i].x += random(-stepSize, stepSize)          [3]
                 * danceFactor;
      pnts[i].y += random(-stepSize, stepSize)
                 * danceFactor;
    }

    strokeWeight(0.1);
    stroke(0);
    beginShape();
    for (var i = 0; i < pnts.length; i++) {
      vertex(pnts[i].x, pnts[i].y);          [4]
      ellipse(pnts[i].x, pnts[i].y, 7, 7);
    }
    vertex(pnts[0].x, pnts[0].y);          [5]
    endShape();
  }

  pop();
}
```

1 The origin of the coordinate system is moved to the current writing position before a letter is written.

2 By keeping the mouse button pressed down, the variable `danceFactor` is set to a value, which increases proportionally to the value of the mouse's x-coordinate.

3 Random values are added to a point's position in every iteration. The value `danceFactor` increases the speed of the movement.

4 Lines connect the dots.

5 Finally, another line is drawn to the first point, closing the outline.

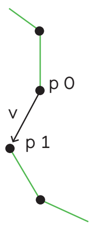
Mouse: Left click + position x: Deformation speed
Keys: Keyboard: Input text
SHIFT: Movement start/stop
DEL: Clear canvas
CTRL: Save image

## P.3.2.4 Parallel font outlines

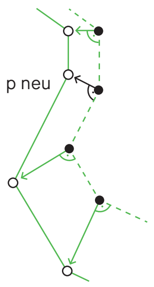
Using the moiré effect of overlapping grid structures, you can create optical illusions that affect font outlines and change the impression of font volumes. Eventually forms emerge that detach themselves from the font and lead a life of their own.

**→ P_3_2_4_01**

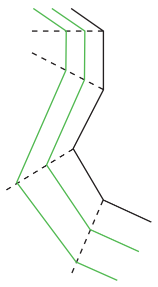
The starting point is the font contour of a letter. For each of the many short sections that make up the font's outline, the same calculation procedure is used: the section is rotated 90° and set to the correct length. The result is a path that runs parallel to the original path. The grid structure arises when this is repeated several times with ever-increasing distances.

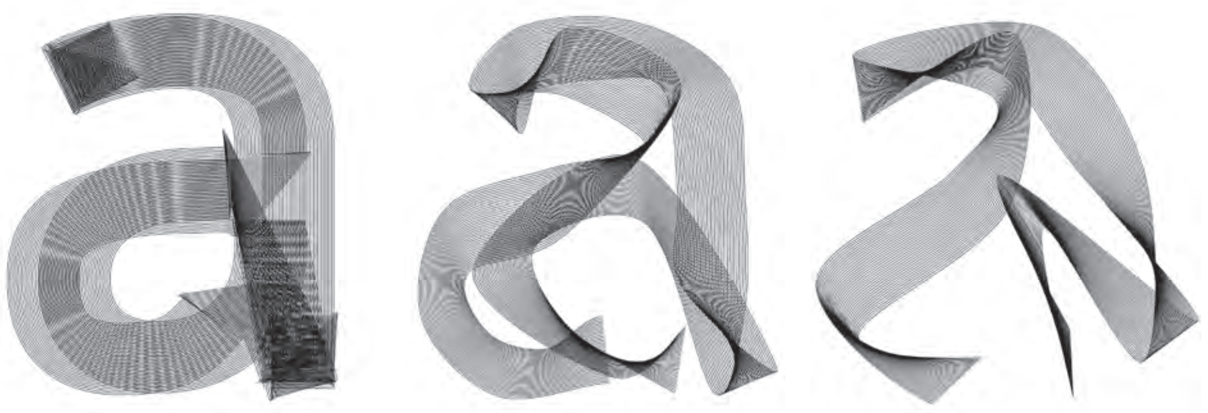


original path deconstructed into sections



section with altered length rotated by 90°


many parallel paths


→ P_3_2_4_01 The lowercase letter "a" shown in three variations. The font outline here, however, has been increasingly simplified.

```
1    function createLetters() {
       letters = [];
2      var chars = textTyped.split('');

       var x = 0;
       for (var i = 0; i < chars.length; i++) {
         if (i > 0) {
3          var charsBefore = textTyped.substring(0, i);
           x = font.textBounds(charsBefore, 0, 0, fontSize).w;
         }
4        var newLetter = new Letter(chars[i], x, 0);
         letters.push(newLetter);
       }
     }
```

```
     function Letter(char, x, y) {
       this.char = char;
       this.x = x;
       this.y = y;

5      Letter.prototype.draw = function() {
6        var path = font.textToPoints(
                          this.char, this.x, this.y, fontSize,
                          {sampleFactor: pathSampleFactor});
         stroke(shapeColor);

7        for (var d = 0; d < ribbonWidth; d += density) {
           beginShape();

           for (var i = 0; i < path.length; i++) {
             var pos = path[i];
8            var nextPos = path[i + 1];

9            if (nextPos) {
               var p0 = createVector(pos.x, pos.y);
               var p1 = createVector(nextPos.x, nextPos.y);
10             var v = p5.Vector.sub(p1, p0);
11             v.normalize();
               v.rotate(HALF_PI);
               v.mult(d);
12             var pneu = p5.Vector.add(p0, v);
               curveVertex(pneu.x, pneu.y);

             }
           }

           endShape(CLOSE);
         }
       }
     }
```

Mouse: Position x: Simplification of font outline
Position y: Width of ribbon outline
Keys: Arrow ←/→: Change line density
Arrow ↓/↑: Change font size

1 When the program starts, or whenever the entered text changes, the `createLetters()` function is called.

2 There, the input text with `split()` generates an array of single letters, `chars`.

3 To determine the `x`-position of a letter, use `substring()` to remove the substring up to the current character and use the `textBounds()` function to calculate its width, `w`.

4 For each letter a new instance of the `letter` class is created and added to the array `letters`.

5 The letter class has a `draw()` function called by the main program in each frame. There, the font outline is moved farther and farther inward.

6 The `textToPoints()` function turns the `char` character into an array of points.

7 This loop draws the individual paths. In each loop, the variable `d` contains the distance of the path to be drawn from the original path.

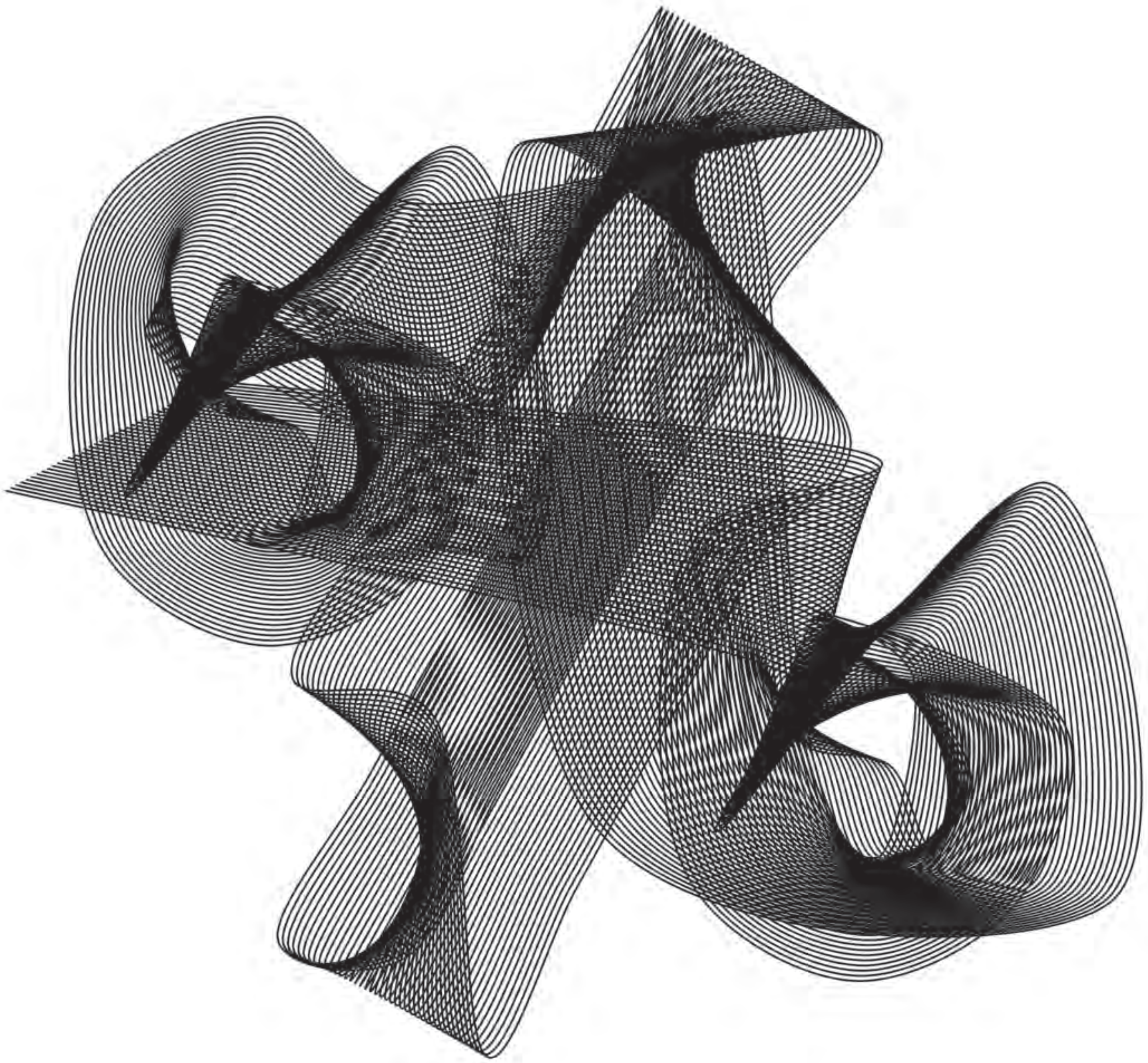
8 Two consecutive positions are taken from the array `path`.

9 If `nextPos` is not empty (i.e., the end of the path has not yet been reached), the two positions are converted to values of type `p5.Vector` with `createVector()`.

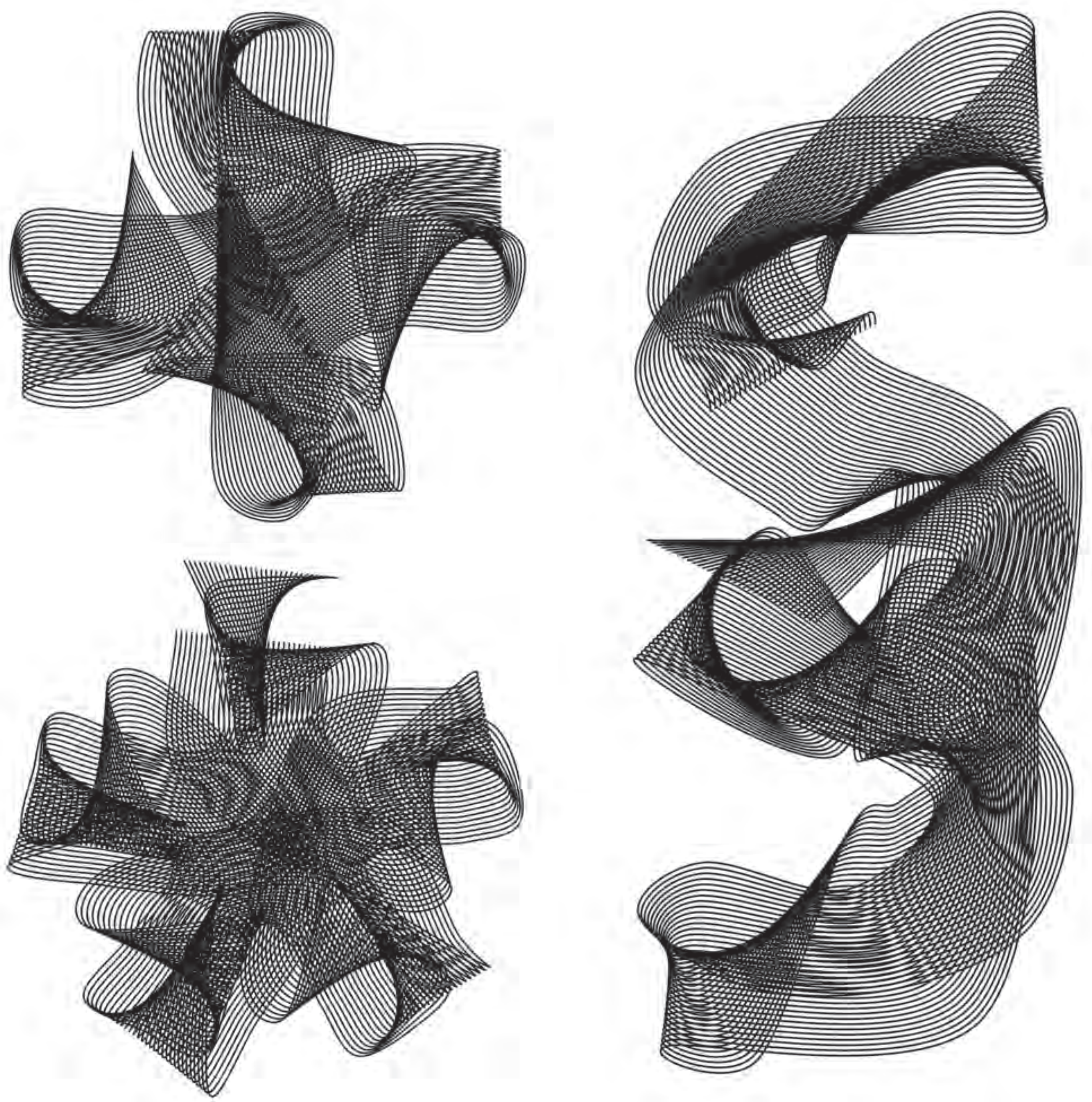
10 `sub()` calculates the difference between the two points and stores them in `v`.

11 The vector `v` is moved to length 1 with `normalize()`, rotated 90° with `rotate()`, and then multiplied by `d`.

12 The position on the offset path is determined by adding `v` to `p0`.



→ P_3_2_4_01 Four characters: percent, plus, star, and paragraph. The outline was greatly simplified. This results in ornate figures.

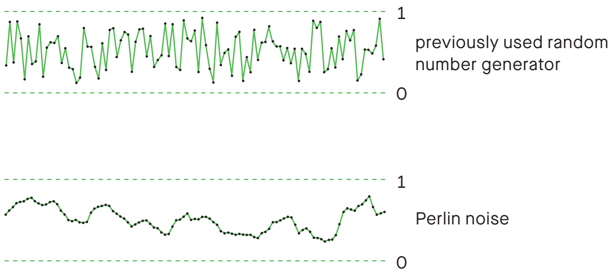


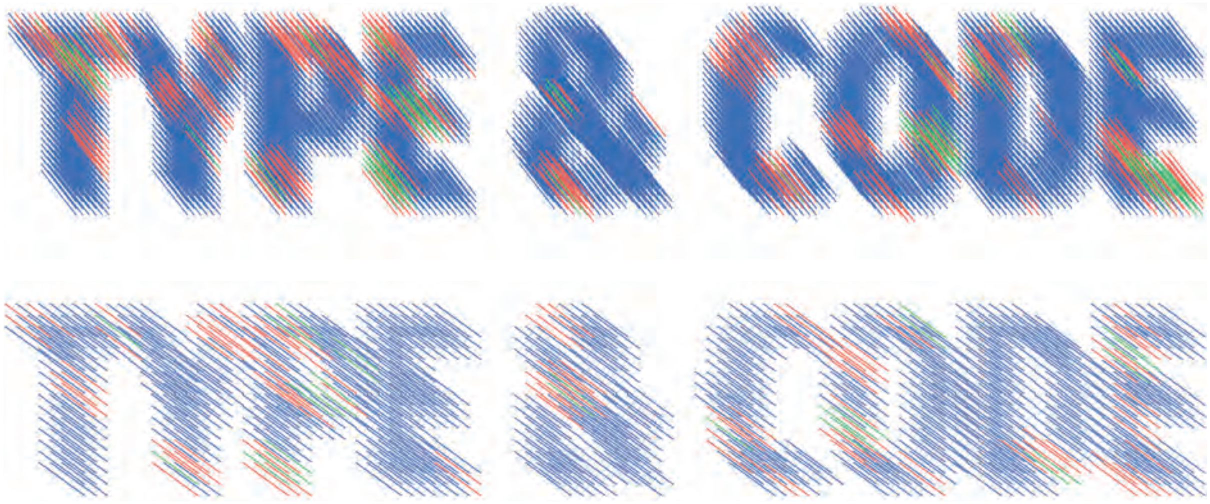
## P.3.2.5 Kinetic font

Here the font outline may do as it wishes. Ignoring legibility, it transforms into patterns and leaves no formal gimmick untried. In constant motion, these metamorphoses remain alive and make us wonder: When does writing become a form of its own?

**→ P_3_2_5_01**

Normally, it is good if every newly generated random number is truly random. When creating animations, however, this usually leads to the image flickering. The use of Perlin noise prevents this. This method of calculating random numbers generates values where the difference from one to the next is never very large.





→ P_3_2_5_01 Rotating lines—sometimes densely arranged, sometimes with a greater distance between them.

```
1  function setupText() {
     textImg = createGraphics(width, height);
     textImg.pixelDensity(1);
     textImg.background(255);
     textImg.textFont(font);
     textImg.textSize(fontSize)
2    textImg.text(textTyped, 100, fontSize + 50);
3    textImg.loadPixels();
   }
```

```
function draw() {
  background(255);

  nOff++;

  for (var x = 0; x < textImg.width; x+=pointDensity) {
    for (var y = 0; y < textImg.height; y+=pointDensity)
{
4       var index = (x + y * textImg.width) * 4;
5       var r = textImg.pixels[index];

        if (r < 128) {

          if(drawMode == 1){
            strokeWeight(1);

            var noiseFac = map(mouseX, 0,width, 0,1);
            var lengthFac = map(mouseY, 0,height, 0.01,1);

6           var num = noise((x+nOff) * noiseFac,
                            y * noiseFac);
7           if (num < 0.6) {
              stroke(colors[0]);
            } else if (num < 0.7) {
              stroke(colors[1]);
            } else {
              stroke(colors[2]);
            }

            push();
            translate(x, y);
            rotate(radians(frameCount));
8           line(0, 0, fontSize * lengthFactor, 0);
            pop();
          }
          ...
        }
      }
  }
}
```

Mouse: Position x/y: Different parameters (depending on drawing mode)

Keys: Keyboard: Text input
Arrow ←/→: Change drawing mode
Arrow ↓/↑: Change point density
DEL: Clear canvas
CTRL: Save image

1 Each time the text is changed, the `setupText()` function is called. This creates a so-called off-screen graphic using `createGraphics()`. This is an image that is not visible but exists only in memory.

2 The entered text, `textTyped`, is written in this image in the previously set font and size.

3 Call `loadPixels()` to be able to read the individual pixel values later.

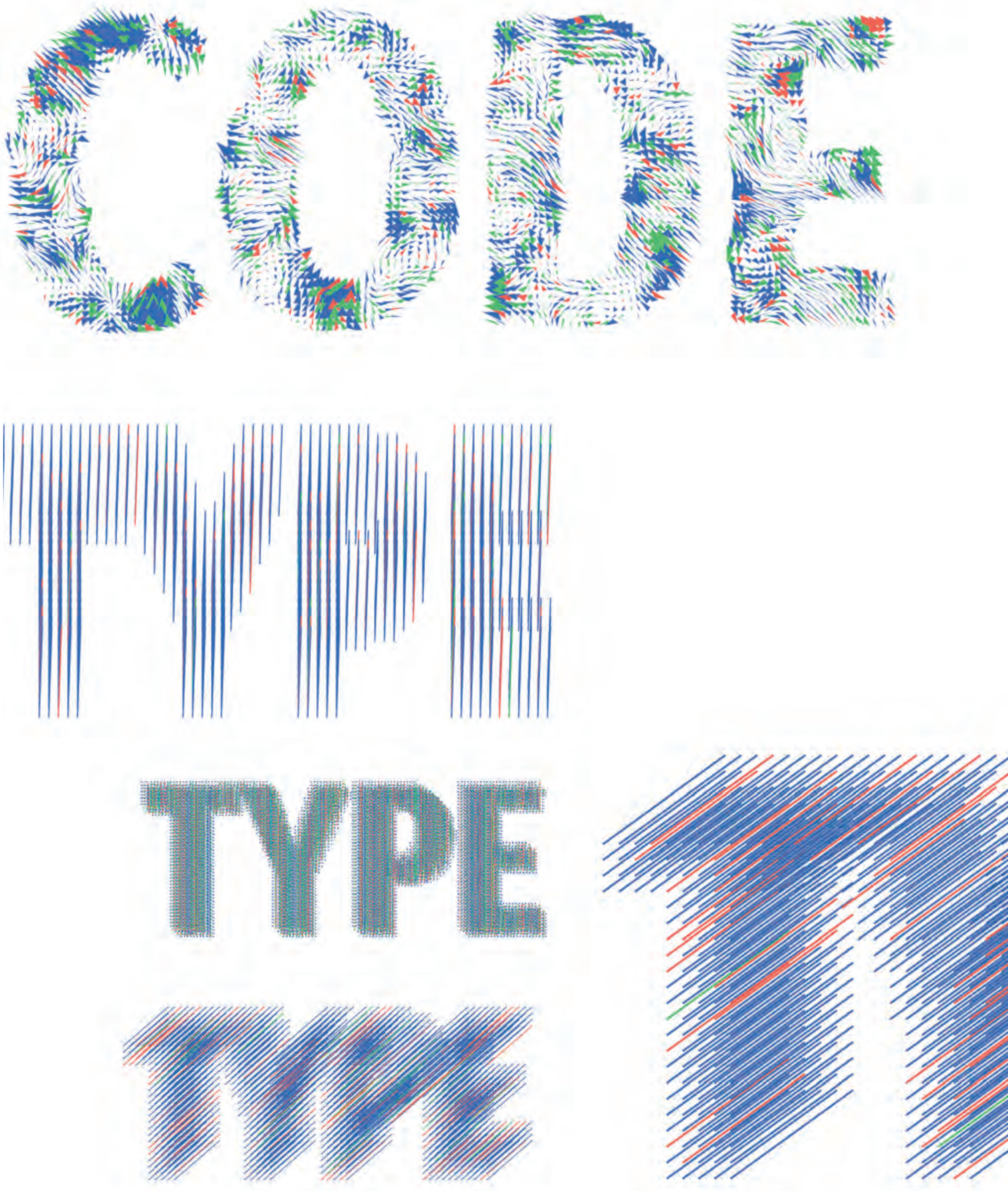
4 The color values of the image are stored as a long string of values. Therefore, to get the color value of a pixel, an index must be calculated from `x` and `y`. The factor `4` is necessary because one pixel consists of four separately stored values (one each for red, green, blue, and transparency).

5 The image with the text consists only of black, white, and a few gray pixels. Therefore, it is sufficient to check only if the red value `r` is below a certain limit, in which case it is a dark pixel.

6 A random value is required to color the lines. To avoid flickering, `noise()` is preferable to the `random()` function. This produces random numbers, similar to a mountainous landscape. The function `noise()` is called with two parameters here, the first dependent on `x`, the second on `y`. The variable `nOff` is incremented continuously, thus ensuring an animation of the random numbers.

7 Depending on `num`, one of the three predefined colors will be selected.

8 A horizontal line is drawn in a previously shifted and rotated coordinate system.



**→ P_3_2_5_01 to → P_3_2_5_03** Different parameter settings and results from all three versions of the program. The letter shapes are generated in different ways: from the pixels (version 01), entirely programmed (version 02), or from the font contours (version 03).