P.4

Copyright © 2018. Princeton Architectural Press. All rights reserved.

Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.

Image

In the last chapter, we saw how text can be dissolved and how the resulting elements—words, letters, and even dots on contours—can be used for experimentation. Similarly, images can be manipulated: details can be copied, collages can be produced, and pixels—the digital image's smallest units of information—can become the basis of a new visual world.

P.4 Image 188

P.4.0 Hello, image 190
P.4.1 Image cutouts 192
P.4.1.1 Image cutouts in a grid 192
P.4.1.2 Feedback of image cutouts 196
P.4.2 Image collection 198
P.4.2.1 Collage from image collection 198
P.4.2.2 Time-based image collection 202
P.4.3 Pixel values 204
P.4.3.1 Graphics from pixel values 204
P.4.3.2 Type from pixel values 210
P.4.3.3 Real-time pixel values 214
P.4.3.4 Emojis from pixel values 220

Copyright © 2018. Princeton Architectural Press. All rights reserved.

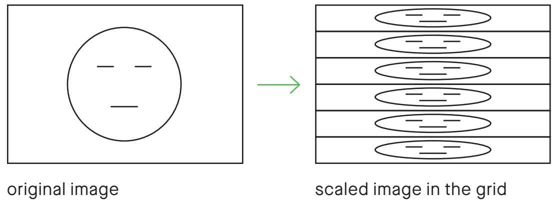
Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.

P.4.0 Hello, image

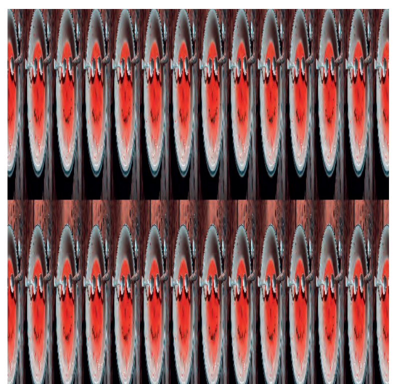
A digital image is a mosaic of small color tiles. Dynamic access to these tiny elements allows for the generation of new compositions. It is possible to create your own collection of image tools with the following programs.

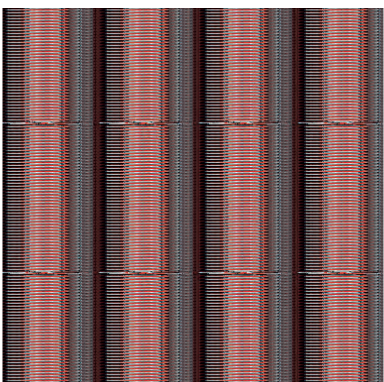
→ P_4_0_01

An image is loaded and displayed in a grid defined by the mouse. Each tile in the grid is filled with a scaled copy of the source image.



original image





→ P_4_0_01 Abstract images are created through the repeated copying and extreme scaling of the source image.

Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.

Copyright © 2018. Princeton Architectural Press. All rights reserved.

1

```
var img;

function preload() {
  img = loadImage('data/image.jpg');
}
```

```
function draw() {
  var tileCountX = mouseX / 3 + 1;
  var tileCountY = mouseY / 3 + 1;
  var stepX = width / tileCountX;
  var stepY = height / tileCountY;
  for (var gridY = 0; gridY < height; gridY += stepY) {
    for (var gridX = 0; gridX < width; gridX += stepX) {
      image(img, gridX, gridY, stepX, stepY);
    }
  }
}
```

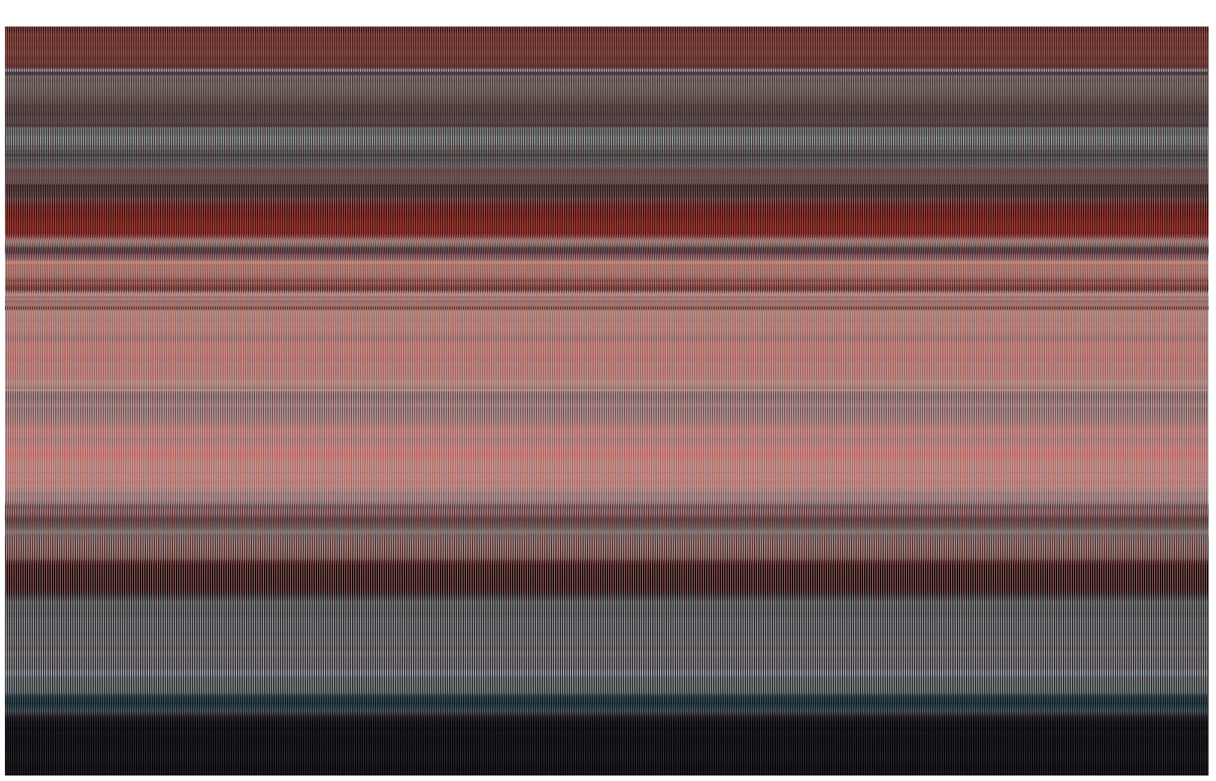
(Markers: 2 at the tileCountX line; 3 at the image() line.)

Mouse: Position x: Number of horizontal tiles
Position y: Number of vertical tiles
Keys: S: Save image

1 The image is loaded in the `preload()` function. This ensures that the loading process is completed before calling the `setup()` and `draw()` functions.

2 The mouse position determines `tileCountX` and `tileCountY` and, thereby, their width `stepX` and height `stepY`.

3 The image is drawn using the function `image()`. The upper-left corner of the image is located in the grid (`gridX, gridY`); width and height are determined by tile width `stepX` and tile height `stepY`.



Copyright © 2018. Princeton Architectural Press. All rights reserved.

Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.

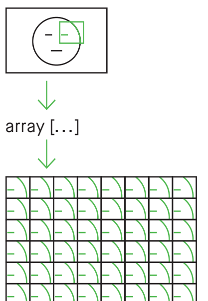
P.4.1.1 Image cutouts in a grid

The principle illustrated below is almost the same as the one in the previous example, and yet a whole new world of images emerges. An image's details and fine structures become pattern generators when only a portion of it is selected and configured into tiles. The results are even more interesting if these sections are randomly selected.

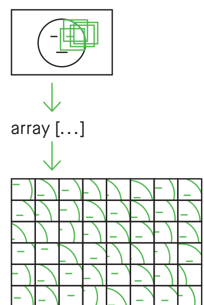
→ P_4_1_1_01

Using the mouse, a section of the image is selected in the display window. After releasing the mouse button, several copies of this section are stored in an array and organized in a grid. The program offers two variations. In variation one, all copies are made from the exact same section. In variation two, the section is shifted slightly at random each time.

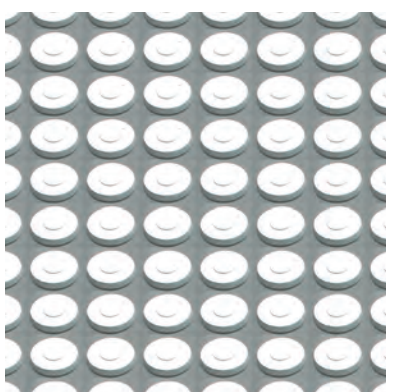
variation 1: no random shift



variation 2: random shift



original image



→ P_4_1_1_01 By repeatedly copying and moving image sections, abstract images are created.

Copyright © 2018. Princeton Architectural Press. All rights reserved.

Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.

```
1  function cropTiles() {
     tileWidth  = width / tileCountY;
     tileHeight = height / tileCountX;
2    imgTiles = [];

     for (var gridY = 0; gridY < tileCountY; gridY++) {
       for (var gridX = 0; gridX < tileCountX; gridX++) {
3        if (randomMode) {
           cropX = int(random(mouseX - tileWidth / 2,
                              mouseX + tileWidth / 2));
           cropY = int(random(mouseY - tileHeight / 2,
                              mouseY + tileHeight / 2));
         }
4        cropX = constrain(cropX, 0, width - tileWidth);
         cropY = constrain(cropY, 0, height - tileHeight);
5        imgTiles.push(img.get(cropX, cropY,
                               tileWidth, tileHeight));
       }
     }
   }
```

Mouse: Position x/y: Detail positioning
Left click: Copy detail
Keys: 1–3: Change detail size
R: Random on/off
S: Save image

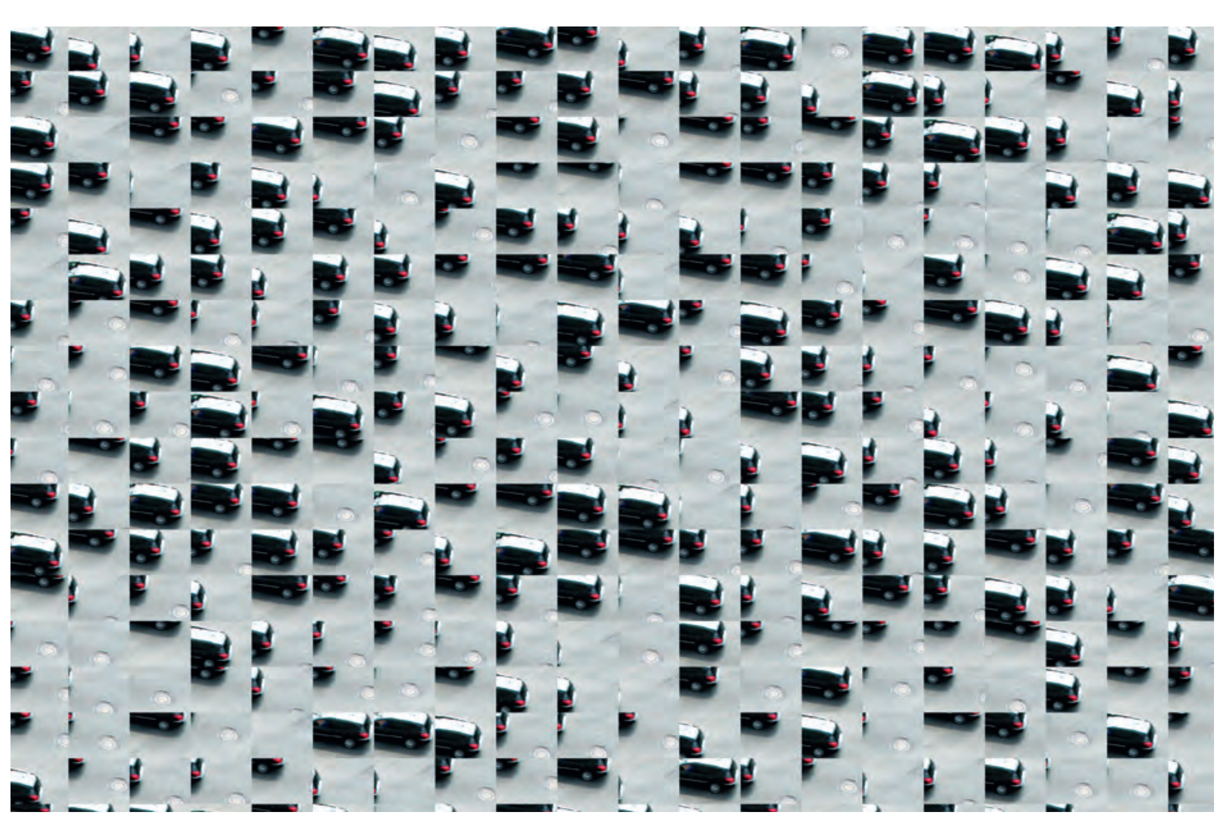
1 The core of the program is the `cropTiles()` function. Here the image is fragmented and the copies of the sections are stored in an array.

2 The image framing array `imgTiles` is cleared.

3 When version two comes into play (`randomMode` is true), all the values for `cropX` and `cropY` are randomly selected from a value range around the mouse position.

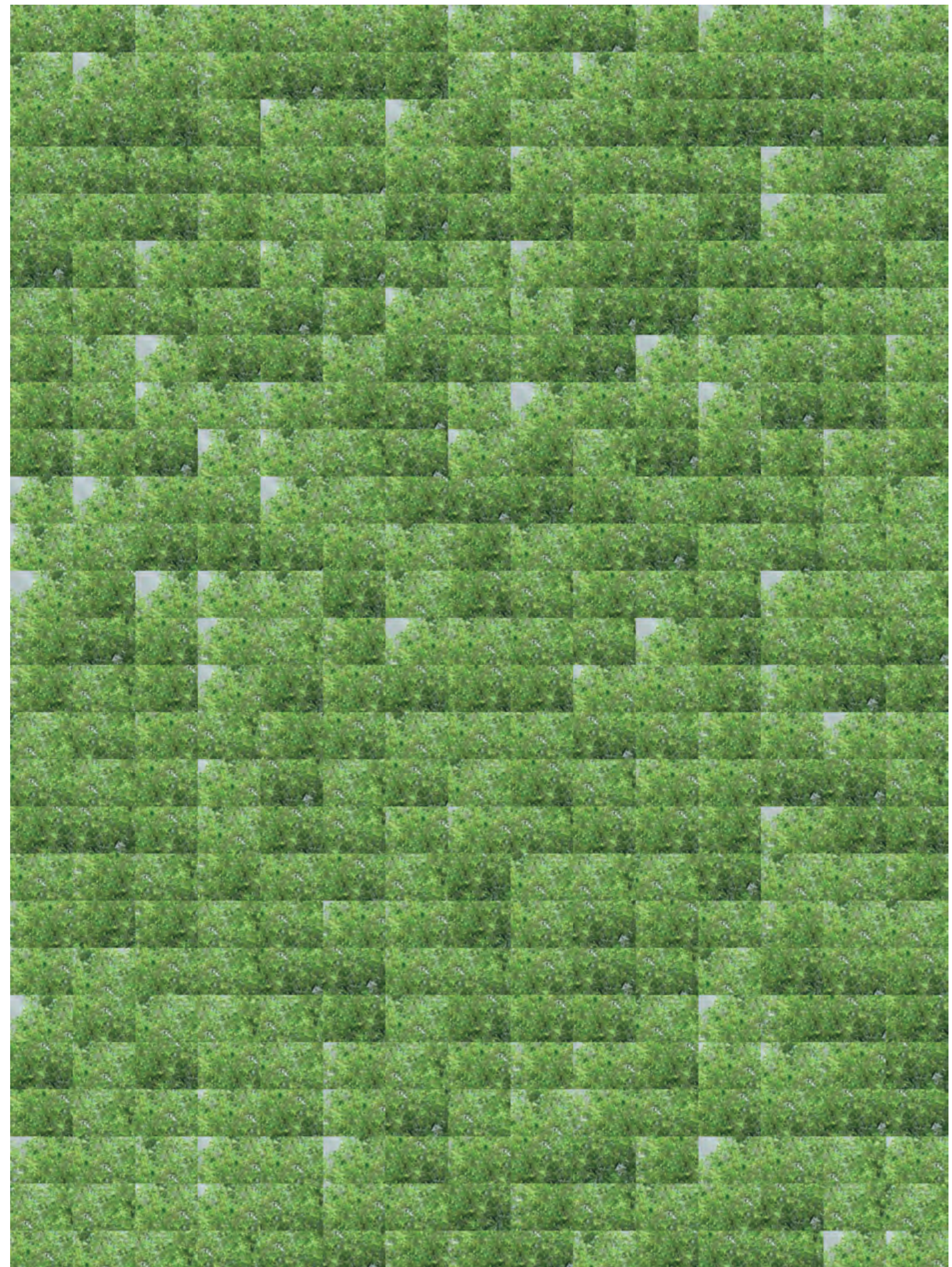
4 The `constrain()` function ensures that the cutout section does not extend beyond the image boundaries.

5 Finally, the section is copied from the image `img` using `get()` and stored in the array.



Copyright © 2018. Princeton Architectural Press. All rights reserved.

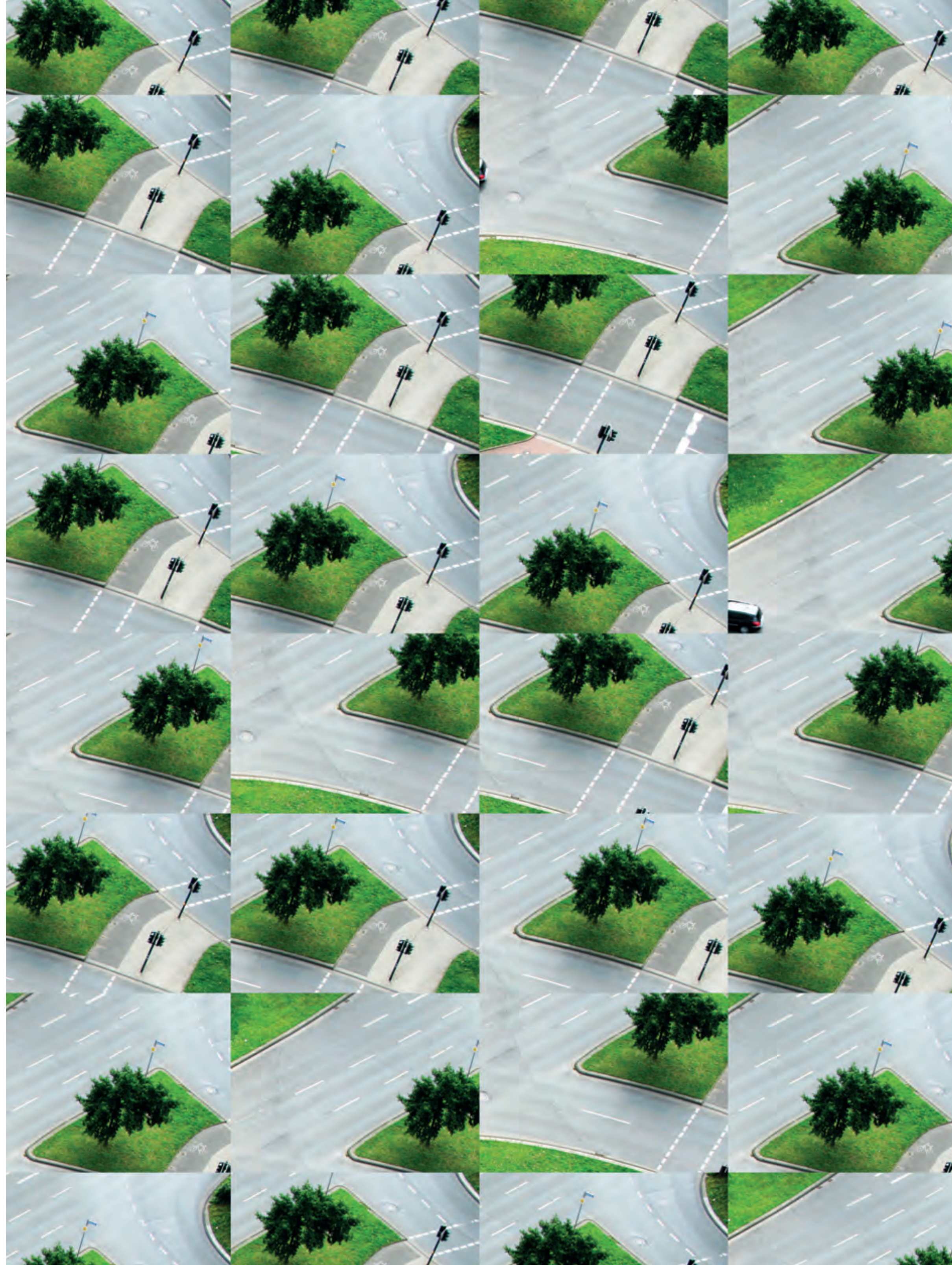
Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.



→ P_4_1_1_01 The multiplication of small image sections creates rhythmic structures that are only recognizable as image sections at a second glance.

Copyright © 2018. Princeton Architectural Press. All rights reserved.

Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.



→ P_4_1_1_01 Using keys 1 to 3, selections can be made among different portions of the image sections. The motifs are still recognizable in these large, detail-rich excerpts but now have an unsettling perspective.

Copyright © 2018. Princeton Architectural Press. All rights reserved.

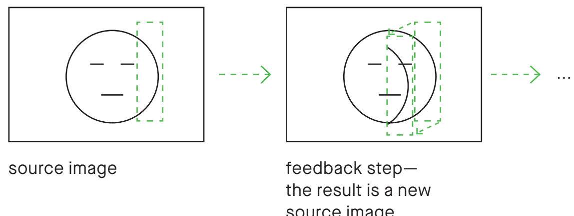
Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.

P.4.1.2 Feedback of image cutouts

A familiar example of feedback: a video camera is directed at a television screen that displays the image taken by the camera. After a short time, the television screen depicts an ever-recurring and distorted image. When this phenomenon is simulated, an image's level of complexity is increased. This repeated overlaying leads to a fragmentary composition.

→ P_4_1_2_01

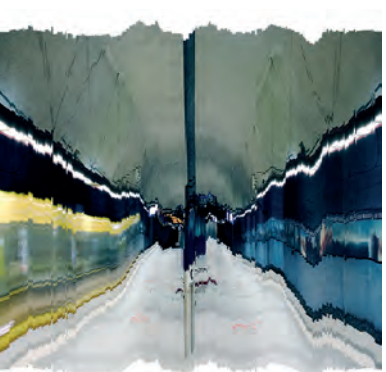
First, the image is loaded and shown in the display. A section of the image is copied to a new randomly selected position with each iteration step. The resulting image now serves as the basis for the next step—the principle of each and every feedback.

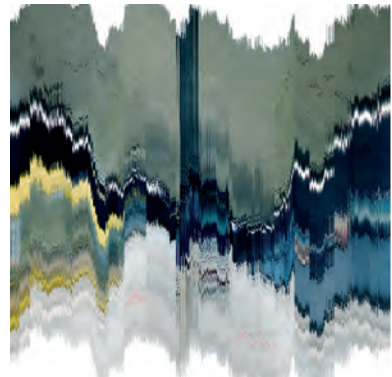


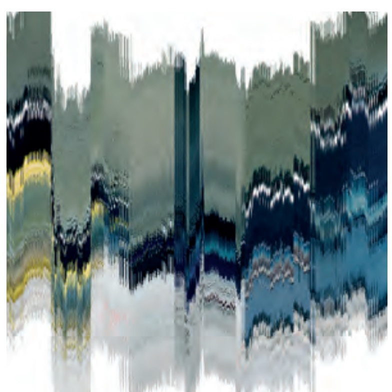
source image

feedback step—
the result is a new
source image

→ Fotografie: Stefan Eigner original image: subway tunnel







→ P_4_1_2_01 Right after the program starts, the motif is easily recognizable. It then dissolves more and more through the overlapping of copied image strips.

Copyright © 2018. Princeton Architectural Press. All rights reserved.

Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.

1

```
function setup() {
  createCanvas(1024, 780);
  image(img, 0, 100);
}
```

2

```
function draw() {
  var x1 = floor(random(width));
  var y1 = 0;

  var x2 = round(x1 + random(-7, 7));
  var y2 = round(random(-5, 5));

  var w = floor(random(10, 40));
  var h = height;
```

3

```
  set(x2, y2, get(x1, y1, w, h));
}
```

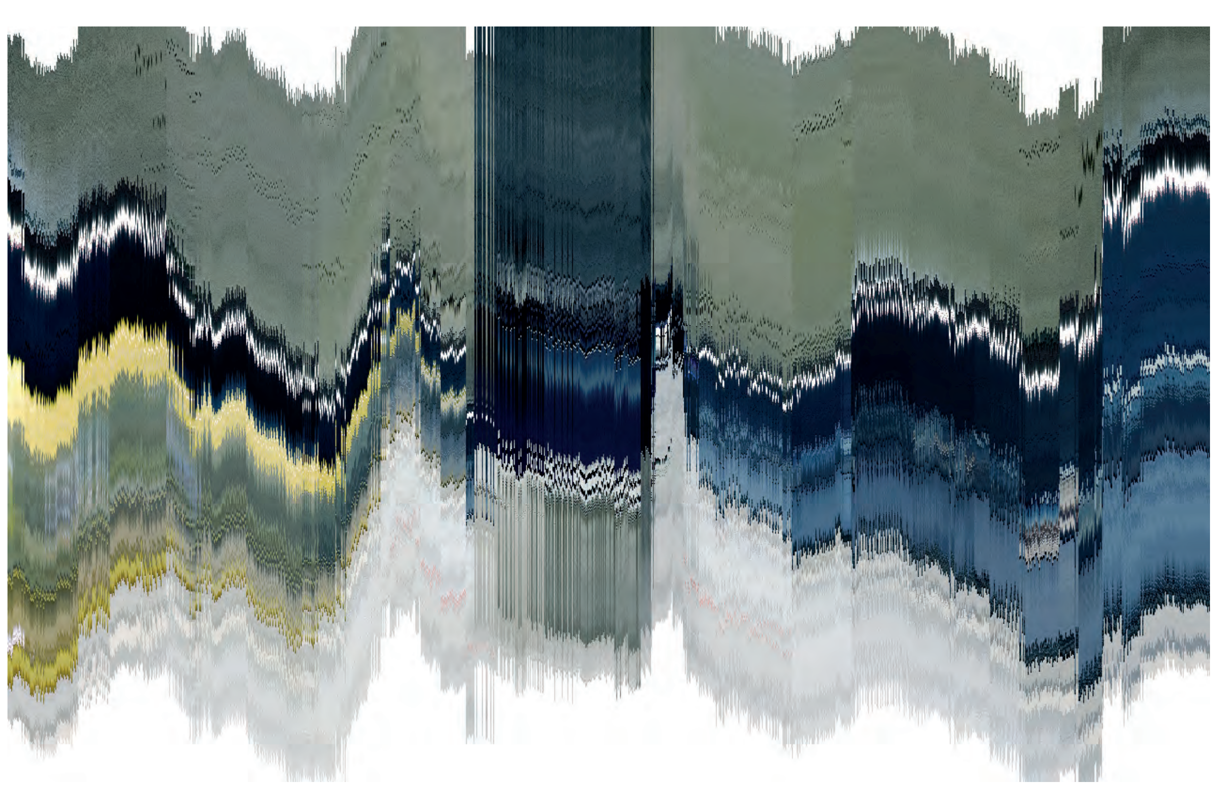
Keys: DEL: Delete display
S: Save image

1 When the program is started, the loaded image is offset one hundred pixels downward using the `image()` command and positioned on the drawing canvas.

2 The x-position of the detail to be copied (`x1`), the target position (`x2`, `y2`), and its width (`w`) are all determined randomly.

3 Using the function `get()`, some of the canvas content is copied and then pasted into the new position (`x2, y2`) using `set()`.

Copyright © 2018. Princeton Architectural Press. All rights reserved.



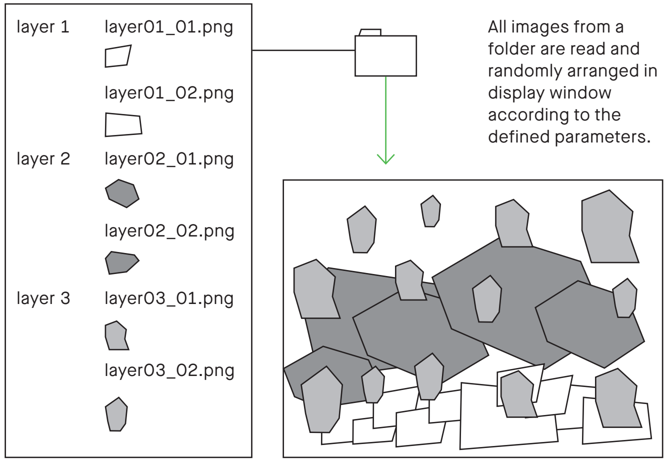
Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.

P.4.2.1 Collage from image collection

Your archive of photographs now becomes artistic material. This program assembles a collage from a folder of images. The cropping, cutting, and sorting of the source images are especially important, since only picture fragments are recombined in the collage.

→ P_4_2_1_01

All the pictures in a folder are read dynamically and assigned to one of several layers. This allows the semantic groups to be treated differently. The individual layers also have room for experimentation with rotation, position, and size when constructing the collage. Note the layer order; the first level is drawn first and is thus in the background.



All images from a folder are read and randomly arranged in display window according to the defined parameters.

Images are assigned to layers according to the filenames (e.g., “layer01_01.png”).

Here, only a few large elements are created from the images on layer 2, while many small ones are created from those on layer 3.

→ P_4_2_1_01 Illustration: Andrea von Danwitz The image consists of three levels: scraps of paper are on layer 1, cutouts of the sky on layer 2, and plants and street elements on layer 3.

A new composition is immediately created when the images on a level are switched or the parameters are changed.

Copyright © 2018. Princeton Architectural Press. All rights reserved.

Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.

[1]
```
var layer1Images = [];
var layer2Images = [];
var layer3Images = [];

var layer1Items = [];
var layer2Items = [];
var layer3Items = [];
```

```
function setup() {
  ...
  layer1Items = generateCollageItems(            // [2]
        layer1Images, 100, width / 2, height / 2,
        width, height, 0.1, 0.5, 0, 0);
  layer2Items = generateCollageItems(
        layer2Images, 150, width / 2, height / 2,
        width, height, 0.1, 0.3, -HALF_PI, HALF_PI);
  layer3Items = generateCollageItems(
        layer3Images, 110, width / 2, height / 2,
        width, height, 0.1, 0.4, 0, 0);

  drawCollageItems(layer1Items);                 // [3]
  drawCollageItems(layer2Items);
  drawCollageItems(layer3Items);
}
```

[4]
```
function CollageItem(image) {
  this.image = image;
  this.x = 0;
  this.y = 0;
  this.rotation = 0;
  this.scaling = 1;
}
```

Keys: 1–3: New random arrangement for one of the three levels
S: Save image

[1] Several arrays are needed to load the images and arrange the layout of the collage pieces. In `layer1Images`, e.g., the loaded images are saved for the first layer in order to be used later to create the collage items.

[2] The function `generateCollageItems()` fills the array layers (`layer1Items,...`) with collage items. The parameters determine which loaded images are to be used and how many items are to be created, and they specify value ranges for positions, scattering, scaling, and rotation. In this example, images in the array `layer1Items` are used. All instances are placed in the position (`width/2`, `height/2`) with a scattering of `width` and `height`. The scaling varies from `0.1` to `0.5` and no rotation is used.

[3] Each time the `drawCollageItems()` function is run, one of the layers is drawn. The order of the layers' invocation determines the construction of the final composition. The images from `layer1Items` are located in the background, those from `layer3Items` in the foreground.

[4] All features of a collage item are summarized in the class `CollageItem`.

Copyright © 2018. Princeton Architectural Press. All rights reserved.

Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.

→ P_4_2_1_02 → Illustration: Andrea von Danwitz A second version of the program allows the image details to be arranged radially around a particular center. The angle at which the images gather and their distance from the center can be specified for each layer.

Copyright © 2018. Princeton Architectural Press. All rights reserved.

Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.

Copyright © 2018. Princeton Architectural Press. All rights reserved.

Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.

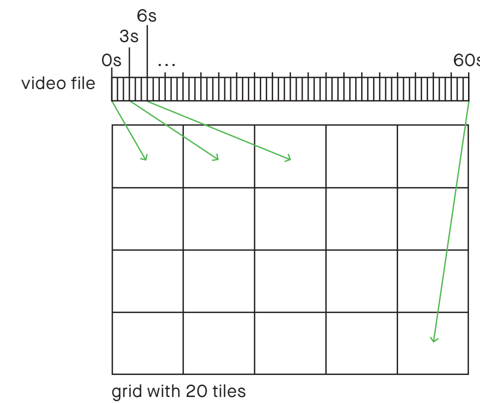
P.4.2.1 Collage from image collection

P.4.2.2 Time-based image collection

In this example, the inner structures of moving images are visualized. After extracting individual images from a video file, this program arranges the images in defined and regular time intervals in a grid. This grid depicts a compacted version of the entire video file and represents the rhythm of its cuts and frames.

→ P_4_2_2_01

To fill the grid, individual still images are extracted at regular intervals from the entire length of a video. Accordingly, a sixty-second video and a grid with twenty tiles results in three-second intervals.




```
1  function draw() {
     if(movie.elt.readyState == 4) {
       var posX = tileWidth * gridX;
       var posY = tileHeight * gridY;

       image(movie, posX, posY, tileWidth, tileHeight);

       currentImage++;
2      var nextTime = map(currentImage, 0, imageCount,
                          0, movie.duration());
3      movie.time(nextTime);

4      gridX++;
       if (gridX >= tileCountX) {
         gridX = 0;
         gridY++;
       }

5      if (currentImage >= imageCount) noLoop();
     }
   }
```

Keys: S: Save image

1 Every time the draw() function is run, an image is selected from the video and depicted in the grid. The first image at time 0 can be placed immediately.

2 The next time in the video (nextTime) is calculated. The variable currentImage (a number between 0 and imageCount) is converted to a second value between 0 and the entire playing time of the video.

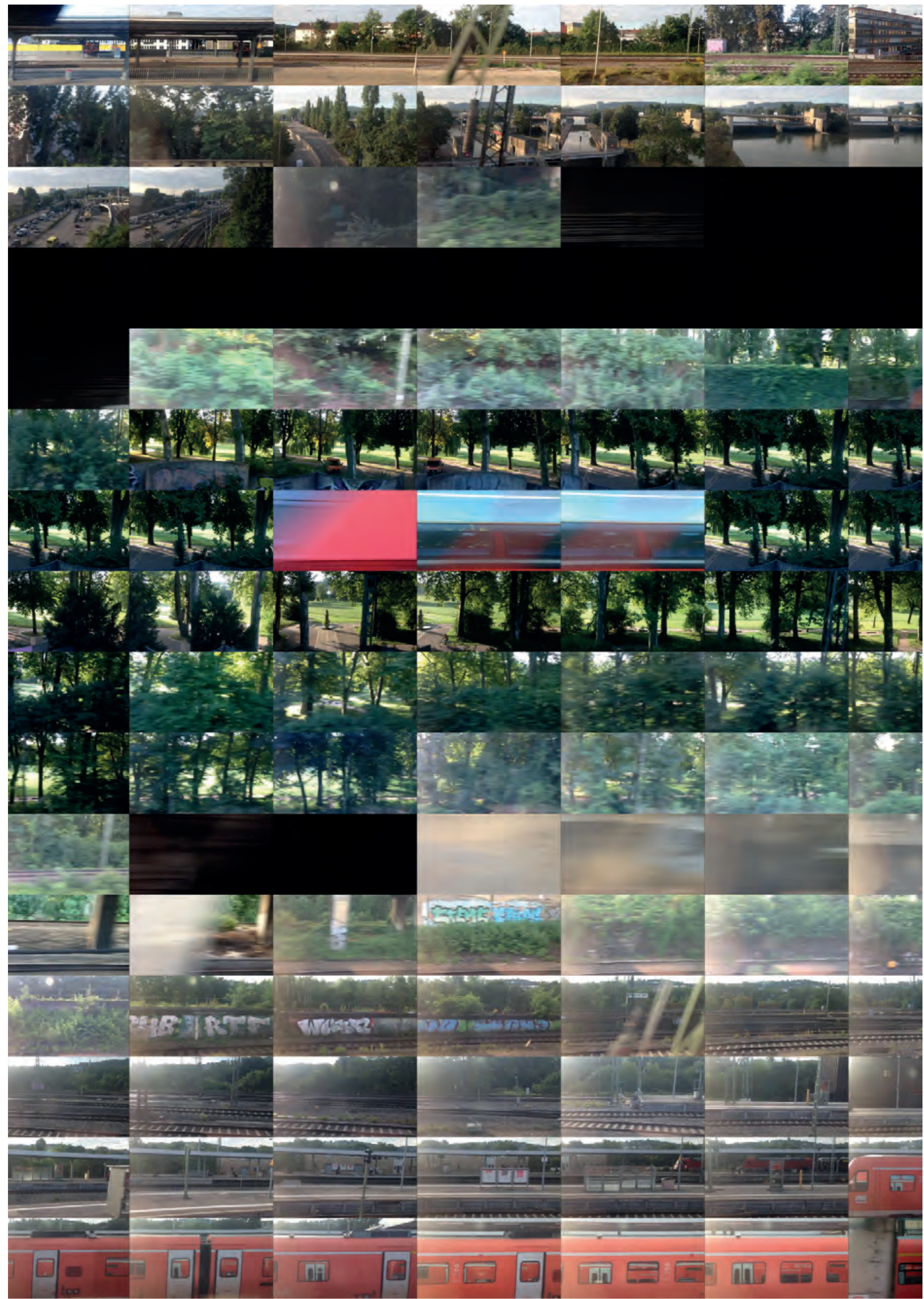
3 Using the time() function, the program jumps to the newly calculated time.

4 To define the next tile, gridX is increased by 1. If the end of the line has been reached, the program jumps to the first image of the next line by setting gridX to 0 and increasing gridY incrementally.

5 The end of the program is reached when all the tiles are filled with images.

Copyright © 2018. Princeton Architectural Press. All rights reserved.

Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.



→ P_4_2_2_01 Fifty-five images from a two-and-a-half-minute video clip, filmed on the way to the main train station in Stuttgart, Germany.

Copyright © 2018. Princeton Architectural Press. All rights reserved.

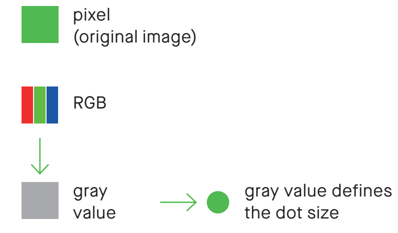
Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.

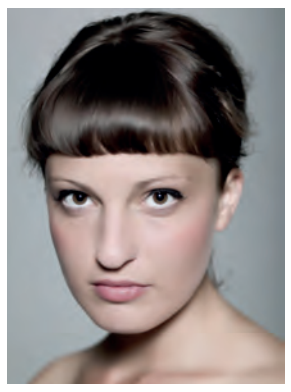
P.4.3.1 Graphics from pixel values

Pixels, the smallest elements of an image, can serve as the starting point for the composition of portraits. In this example, each pixel is reduced to its color value. These values modulate design parameters such as rotation, width, height, and area. The pixel is completely replaced by a new graphic representation, and the portrait becomes somewhat abstract.

→ P_4_3_1_01

The pixels of an image are analyzed sequentially and transformed into other graphic elements. The key to this is the conversion of the color values of pixels (RGB) into the corresponding gray values, because—in contrast to the pure RGB values—these can be practically applied to design aspects such as line width. It is advisable to reduce the resolution of the source image first.





→ P_4_3_1_01 → Photograph: Tom Ziora
The gray value of each pixel defines the size of its diameter; the pixels' original color values are kept.

Copyright © 2018. Princeton Architectural Press. All rights reserved.

Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.

```
for (var gridX=0; gridX<img.width; gridX++) {
  for (var gridY=0; gridY<img.height; gridY++) {
    var tileWidth = width / img.width;
    var tileHeight = height / img.height;
    var posX = tileWidth * gridX;
    var posY = tileHeight * gridY;

    img.loadPixels();
    var c = color(img.get(gridX, gridY));
    var grayscale = round(red(c) * 0.222 +
          green(c) * 0.707 + blue(c) * 0.071);

    switch (drawMode) {
      case 1:
        var w1 = map(grayscale, 0, 255, 15, 0.1);
        stroke(0);
        strokeWeight(w1 * mouseXFactor);
        line(posX, posY, posX + 5, posY + 5);
        break;
      case 2:
        ...
    }
  }
}
```

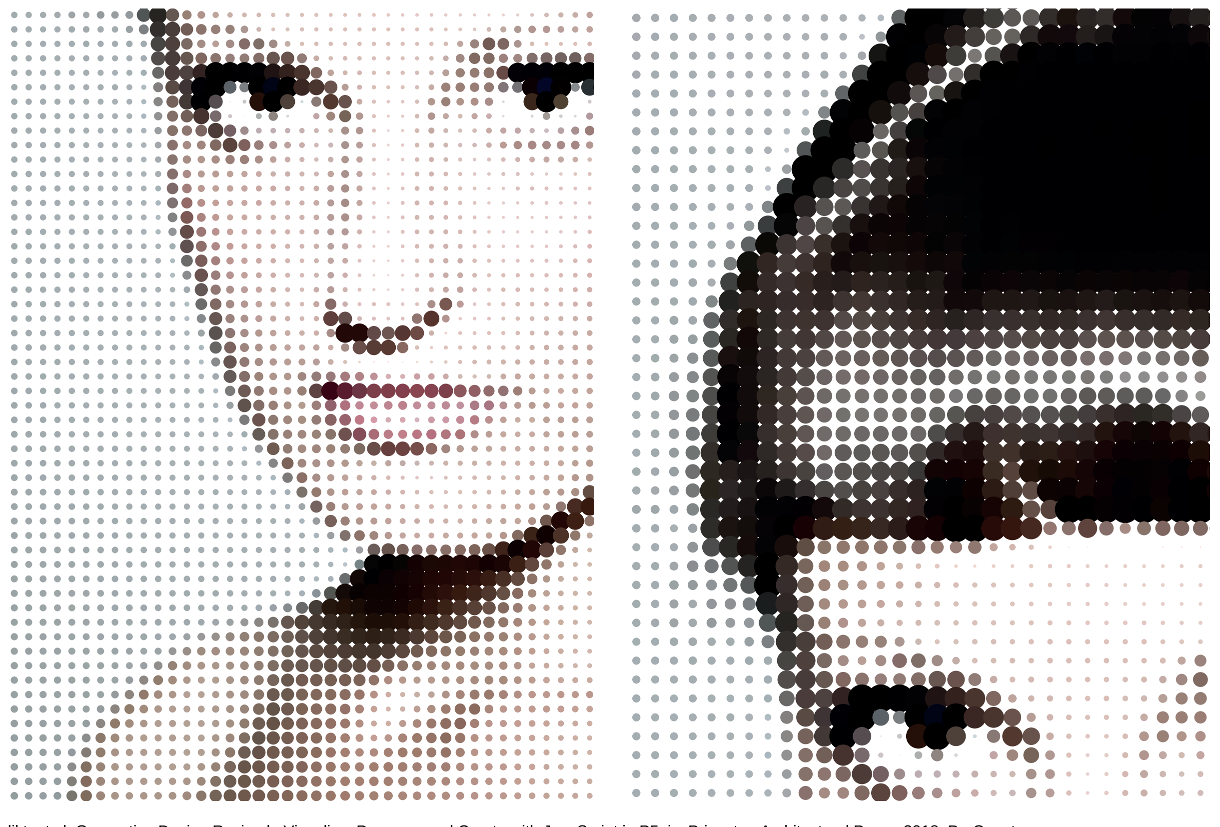
1. The width and height of the original image determines the resolution of the grid.

2. The color of the pixels at the current grid position (and thus the image) is defined.

3. In calculating the gray value, the values for red, green, and blue are weighted differently, whereby there are no absolutely correct weights since colors are both displayed and perceived differently. This gray value is used later to control individual parameters.

4. The program provides several drawing modes, `drawMode`, which can also be influenced by the horizontal mouse position. These were previously converted into a value between 0.05 and 1 and are available in the variable `mouseXFactor`.

Mouse: Position x/y: Different parameters (dependent on drawing mode)
Keys: S: Save image



Copyright © 2018. Princeton Architectural Press. All rights reserved.

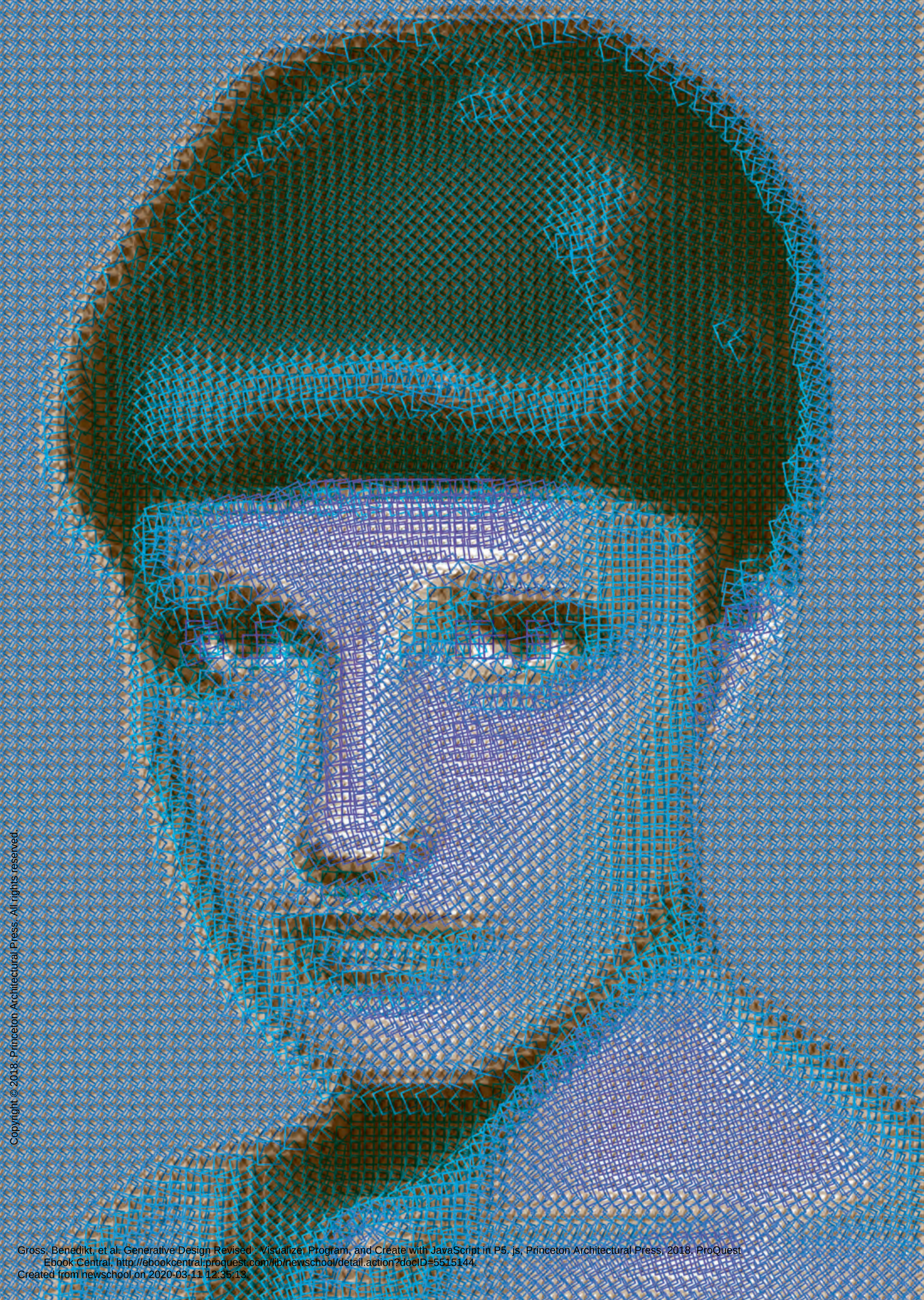
Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.

→ P_4_3_1_01 The gray value defines the size, stroke value, rotation, and position of the elements.

→ P_4_3_1_01 In this drawing mode (key 9), each pixel is represented by several color-distorting elements.

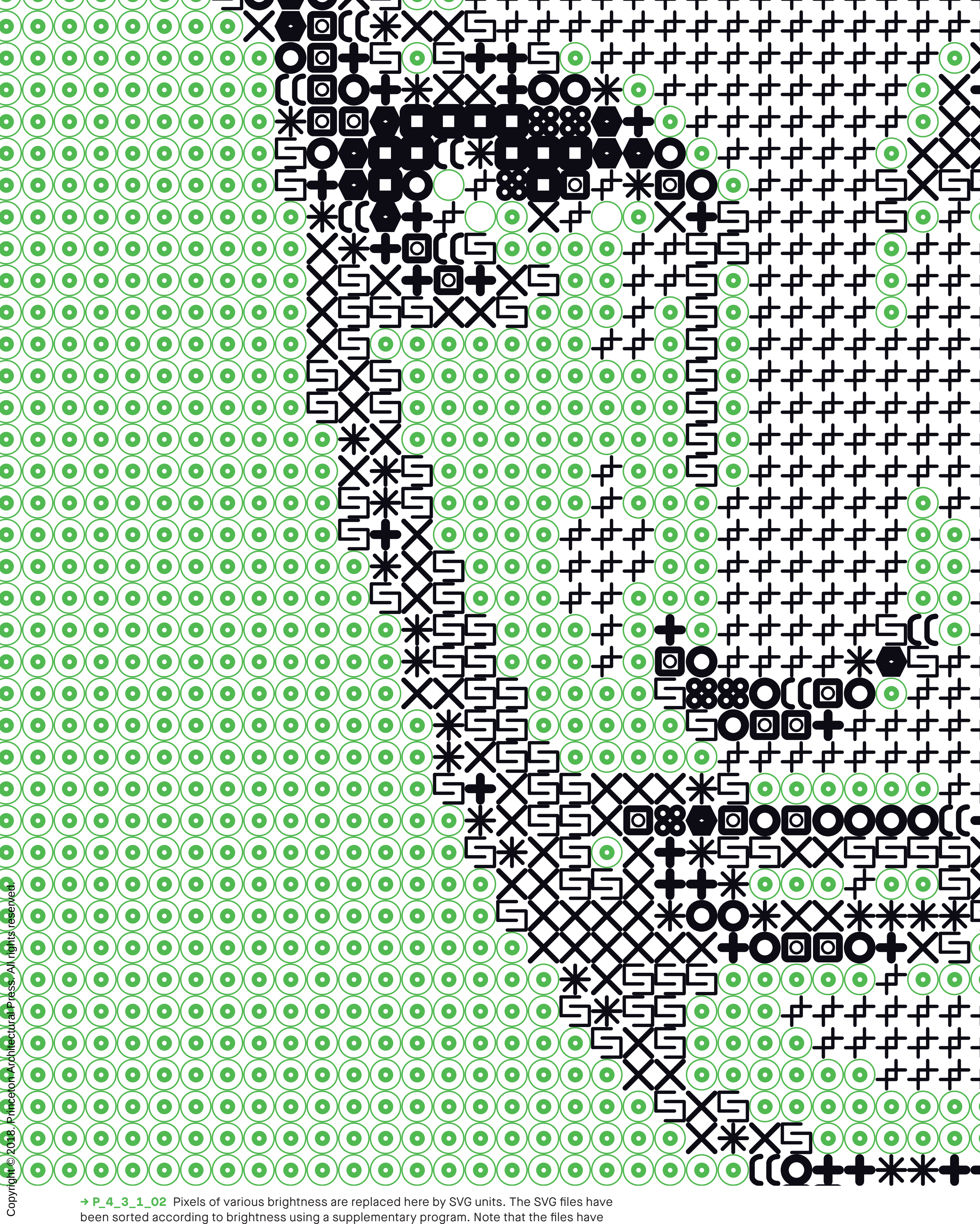
Copyright © 2018. Princeton Architectural Press. All rights reserved.

Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.



Copyright © 2018. Princeton Architectural Press. All rights reserved.

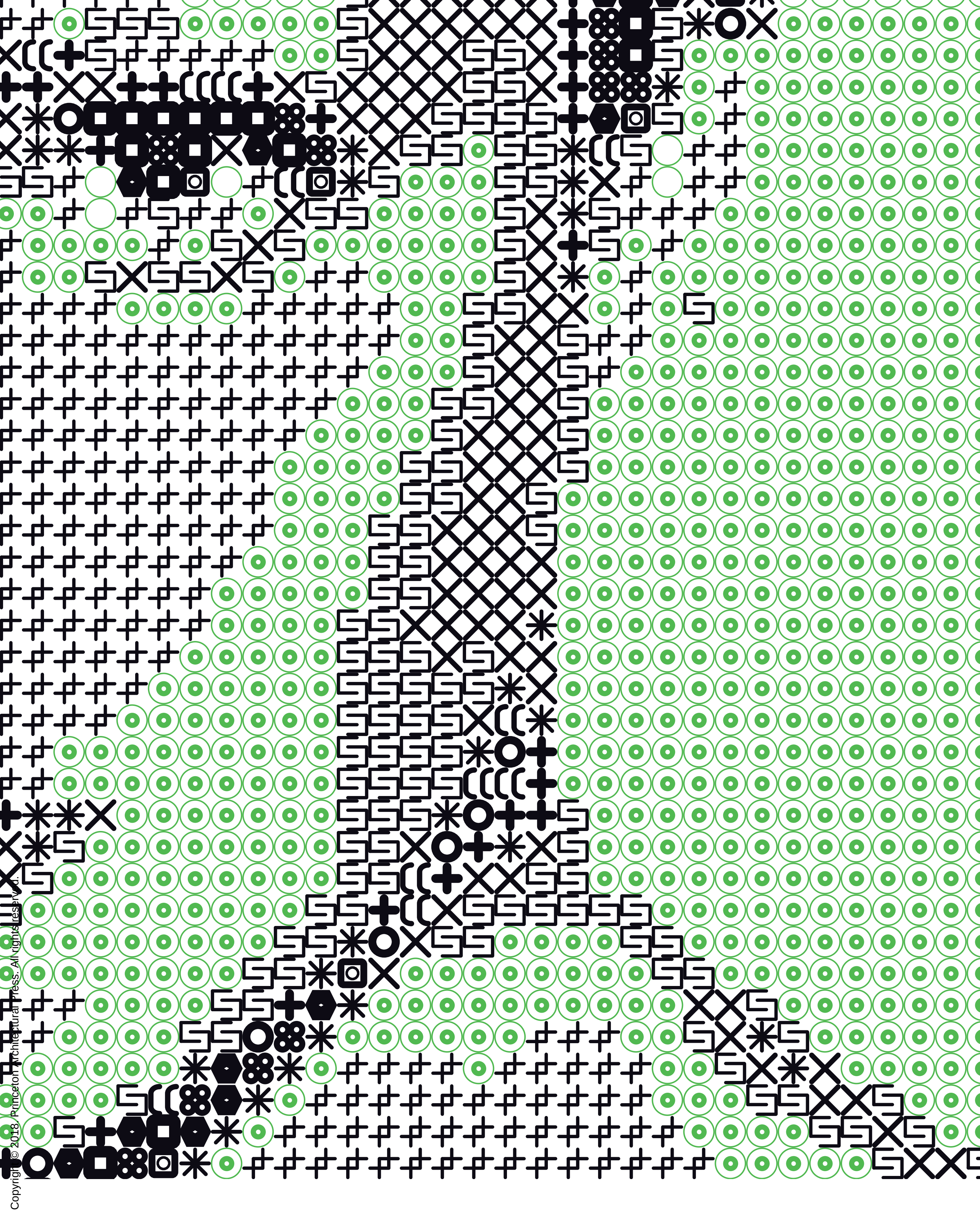
Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.



→ P_4_3_1_02 Pixels of various brightness are replaced here by SVG units. The SVG files have been sorted according to brightness using a supplementary program. Note that the files have been renamed (the brightness value forms the beginning of the file name).

Copyright © 2018. Princeton Architectural Press. All rights reserved.

Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.



Copyright © 2018. Princeton Architectural Press. All rights reserved.

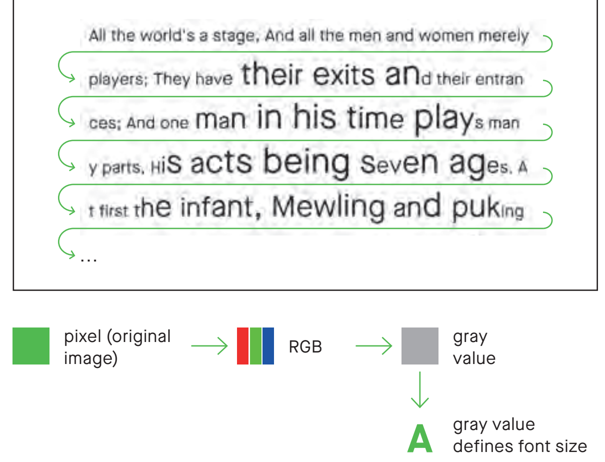
Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.

P.4.3.2 Type from pixel values

The following text image is ambiguous. It can be read for its meaning, or viewed at a distance and perceived as a picture. The pixels from the image control the configuration of the letters. The size of each letter depends on the gray values of the pixels in the original image and thereby creates an additional message.

→ P_4_3_2_01

A character string is processed letter by letter → P.3.1.1/P.3.1.2 and constructed row by row in the normal writing direction. Before a character is drawn, its position in display coordinates is matched to the corresponding position in the original image in pixel coordinates. Only a subset of the original pixels is used—merely those for which a corresponding character position exists. The color of the selected pixel can now be converted into its gray value and the gray value used to modulate the font size, for example.



→ P_4_3_2_01 The color of a pixel can define the size or color of the letters, or both.

Copyright © 2018. Princeton Architectural Press. All rights reserved.

Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.

```
function draw() {
  ...
  var x = 0;
  var y = 10;
  var counter = 0;

  while (y < height) {                                        // 1
    img.loadPixels();

    var imgX = round(map(x, 0, width, 0, img.width))          // 2
    var imgY = round(map(y, 0, height, 0, img.height))
    var c = color(img.get(imgX, imgY));
    var grayscale = round(red(c) * 0.222 +
                          green(c) * 0.707 +
                          blue(c) * 0.071);

    push();
    translate(x, y);

    if (fontSizeStatic) {                                     // 3
      textSize(fontSizeMax);
      if (blackAndWhite) fill(grayscale);
      else fill(c);
    } else {
      var fontSize = map(grayscale, 0, 255,
                          fontSizeMax, fontSizeMin);
      fontSize = max(fontSize, 1);                            // 4
      textSize(fontSize);
      if (blackAndWhite) fill(0);
      else fill(c);
    }

    var letter = inputText.charAt(counter);
    text(letter, 0, 0);
    var letterWidth = textWidth(letter) + kerning;

    x += letterWidth;                                         // 5

    pop();

    if (x + letterWidth >= width) {                           // 6
      x = 0;
      y += spacing;
    }

    counter++;
    if (counter >= inputText.length) {
      counter = 0;
    }
  }
  noLoop();
}
```

Keys: 1: Switch character size mode
2: Switch character color mode
Arrow ↓/↑: Maximum character size –/+
Arrow ←/→: Minimum character size –/+
S: Save image

1. The writing process continues as long as the y-coordinate of the current writing position `y` is still less than the height of the display.

2. Using the `map()` function, the display coordinates are converted into image coordinates; e.g., the x-coordinate `x` is proportionally converted from a value between `0` and the display `width` to a corresponding value between `0` and the width of the image `img.width`.

3. Depending on the selected mode `fontSizeStatic` (key 1 or 2), the font size is set to a fixed value `fontSizeMax` or is varied by the gray value.

4. The value `fontSize` cannot be zero or negative, as this would cause problems. Therefore, the function `max()` ensures this value is at least 1.

5. The value of the variable `x` is increased by the character width.

6. If `x` is greater than or equal to the width of the drawing canvas, the line is wrapped. The y value then increases by the line spacing and `x` restarts from 0 on the far left of the drawing canvas.

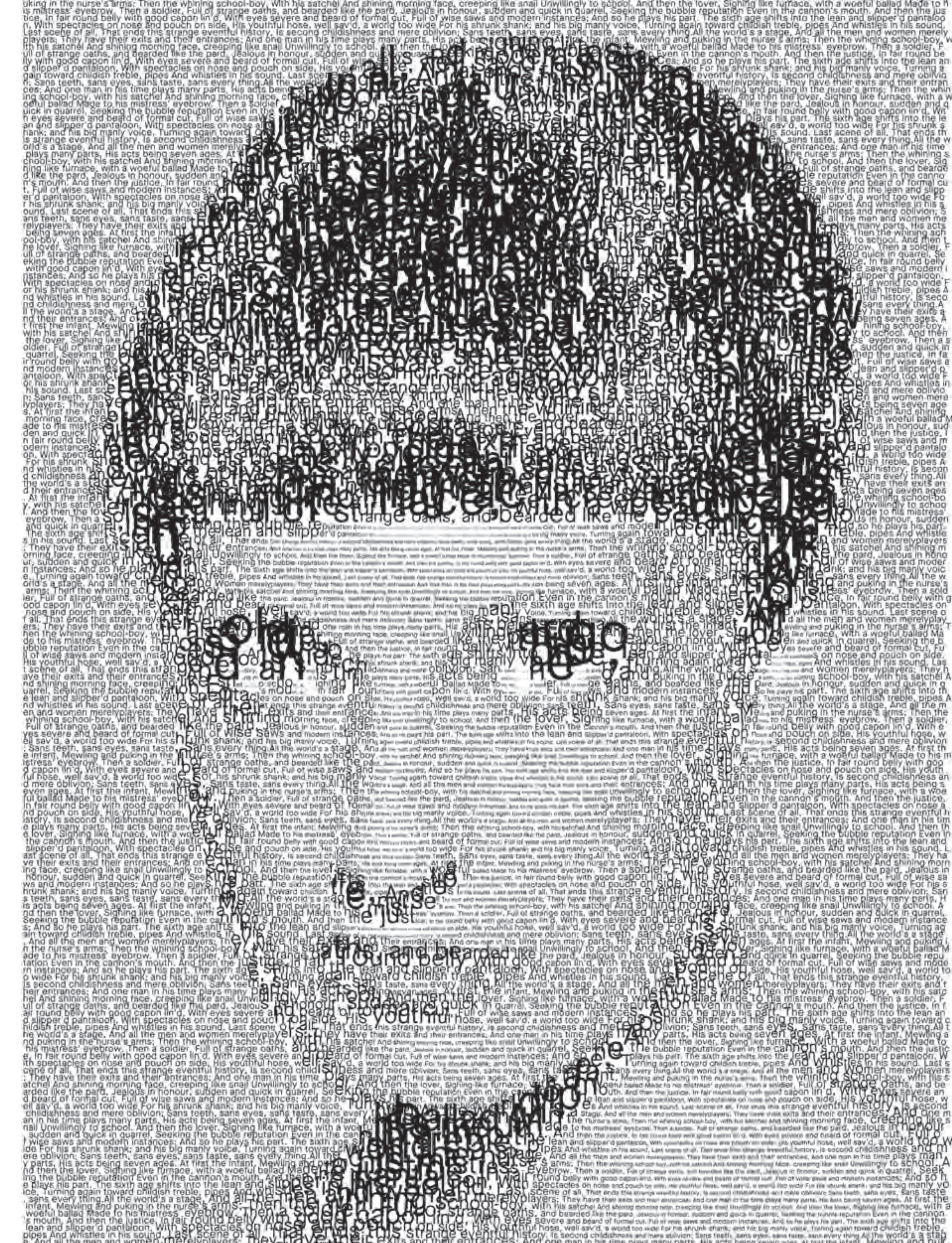
Copyright © 2018. Princeton Architectural Press. All rights reserved.

Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.

→ **P_4_3_2_01** The size and color of the characters are defined by an underlying image.

Copyright © 2018. Princeton Architectural Press. All rights reserved.

Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.



→ P_4_3_2_01 Here the gray value of the pixels determines font size.

Copyright © 2018. Princeton Architectural Press. All rights reserved.

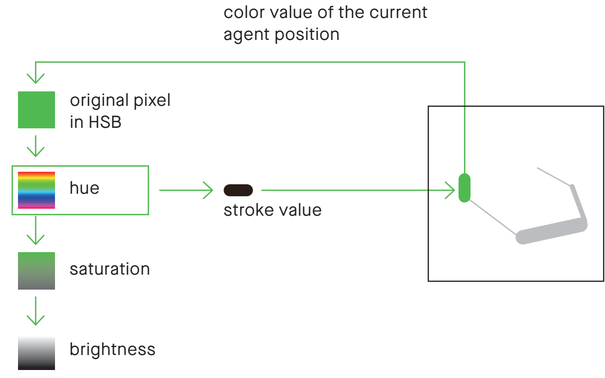
Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.

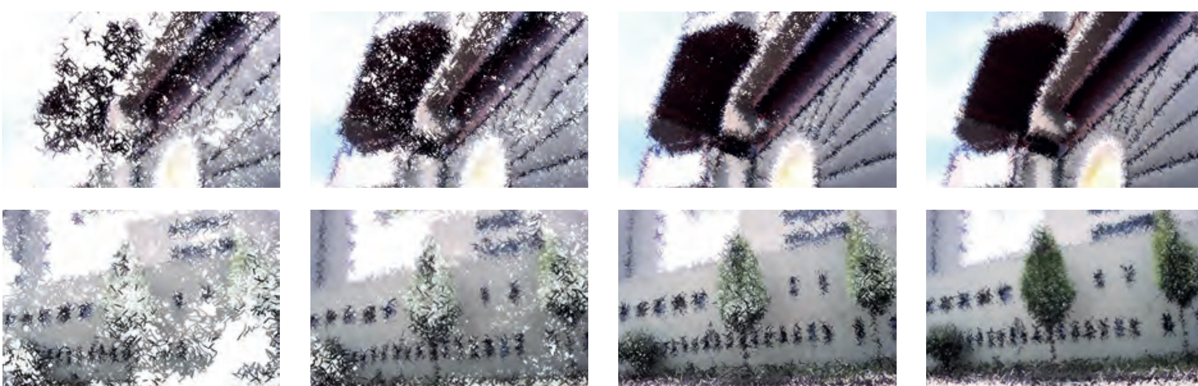
P.4.3.3 Real-time pixel values

The color values of pixels can again be translated into graphic elements but with two important differences: first, the pixels are constantly changing because the images come from a video camera, and second, pixels are translated sequentially by dumb agents that are constantly in motion rather than all at once. The motion captured by the camera and the migration of the agents thus can paint a picture right before our eyes.

→ P_4_3_3_01

A dumb agent moves over the drawing canvas. The color value of the current real-time video image is analyzed at each position and serves as a parameter for each color and stroke value. The mouse position defines the stroke length and the speed of the agent.
→P.2.2.1 Dumb agents





→ P_4_3_3_01 The agent's path gradually creates an image.

Copyright © 2018. Princeton Architectural Press. All rights reserved.

Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.

```
function setup() {
  ...
  video = createCapture(VIDEO, function(){
    streamReady = true
  });
  video.size(width, height);
  video.hide();
  ...
}
```

```
function draw() {
  if(streamReady) {
    for (var j = 0; j <= mouseX / 50; j++) {
      video.loadPixels();

      var pixelIndex = (((video.width - 1 - x)
                   + y * video.width) * 4);
      var c = color(video.pixels[pixelIndex],
                    video.pixels[pixelIndex + 1],
                    video.pixels[pixelIndex + 2]);

      var cHSV = chroma(red(c), green(c), blue(c));
      strokeWeight(cHSV.get('hsv.h') / 50);
      stroke(c);

      diffusion = map(mouseY, 0, height, 5, 100);

      beginShape();
      curveVertex(x, y);
      curveVertex(x, y);

      for (var i = 0; i < pointCount; i++) {
        var rx = int( random(-diffusion, diffusion));
        curvePointX = constrain(x + rx, 0, width - 1);
        var ry = int( random(-diffusion, diffusion));
        curvePointY = constrain(y + ry, 0, height - 1);
        curveVertex(curvePointX, curvePointY);
      }

      curveVertex(curvePointX, curvePointY);
      endShape();

      x = curvePointX;
      y = curvePointY;
    }
  }
}
```

Mouse: Position x: Drawing speed
Position y: Direction

Keys: Arrow ↓/↑: Number of curve points –/+
Q: Stop drawing
W: Continue drawing
Arrow ←/→: Minimum font size –/+
S: Save image

1 The function `createCapture()` creates a video element to access images on the webcam.

2 The live video images from the connected video camera are reduced to the size of the drawing canvas.

3 The `hide()` command prevents the video image from being automatically displayed on the drawing canvas.

4 If the video signal is available, the pixels of the current video image are loaded.

5 As in a static pixel image, the pixels in a video image are also numbered row by row. Therefore, the pixel index has to be calculated from the current writing position (`x, y`). When using webcams directed at the user, it is useful to mirror the video image horizontally using the calculation `video.width-1-x`.

6 The stroke value is set so it is defined by the hue of the pixel. The chroma.js library helps to convert RGB to HSV values.

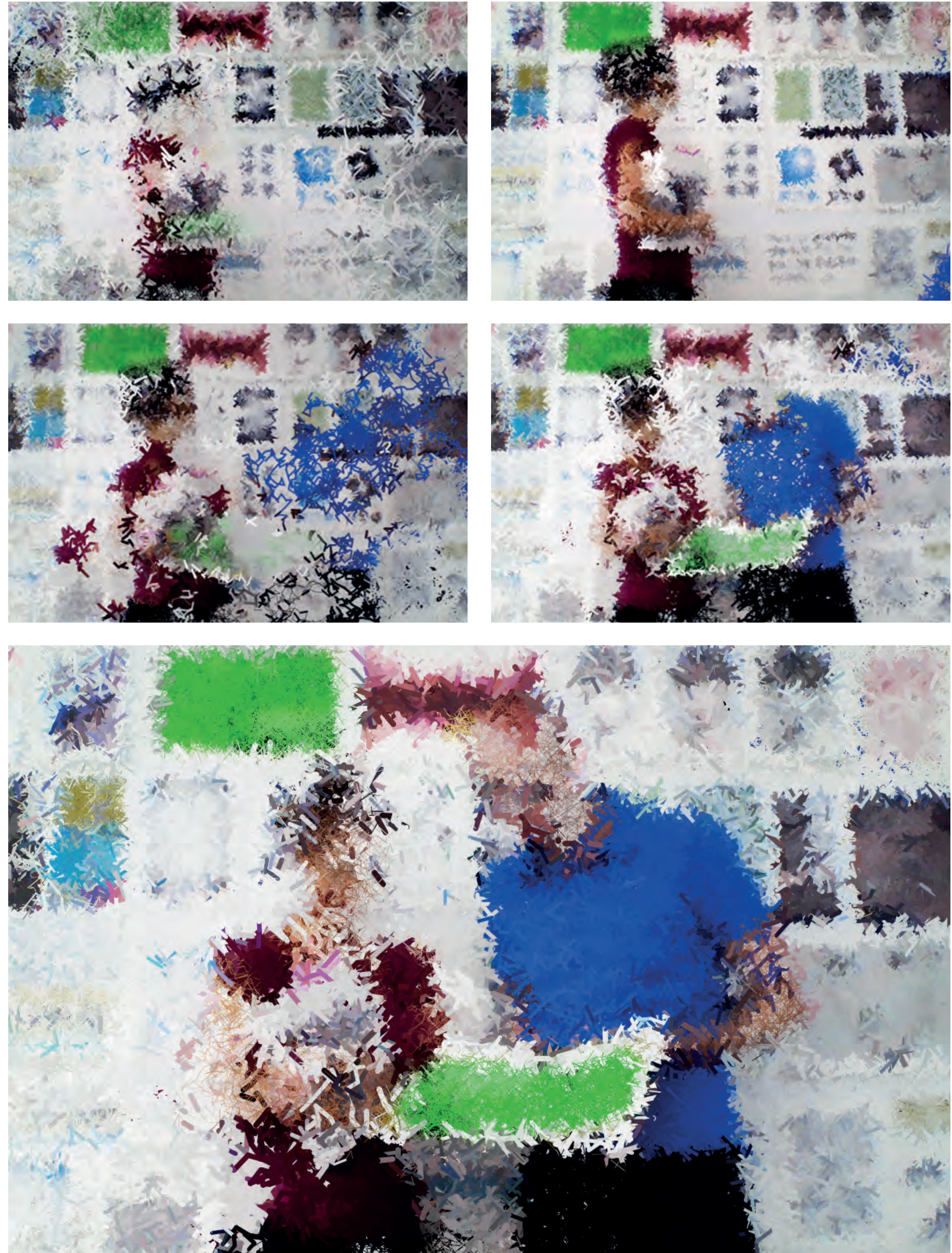
7 The line element can now be drawn. The first curve point is placed on the current drawing position. This is done twice because the first and last points are not drawn when drawing lines with `curveVertex()`.

8 The variable `pointCount` now specifies how many curve points are to be drawn. The default value is 1, so only one line is drawn. The curve points are placed in random positions around the drawing position. The value `diffusion` specifies how large this area is.

9 The last curve point is specified as the new drawing position.

Copyright © 2018. Princeton Architectural Press. All rights reserved.

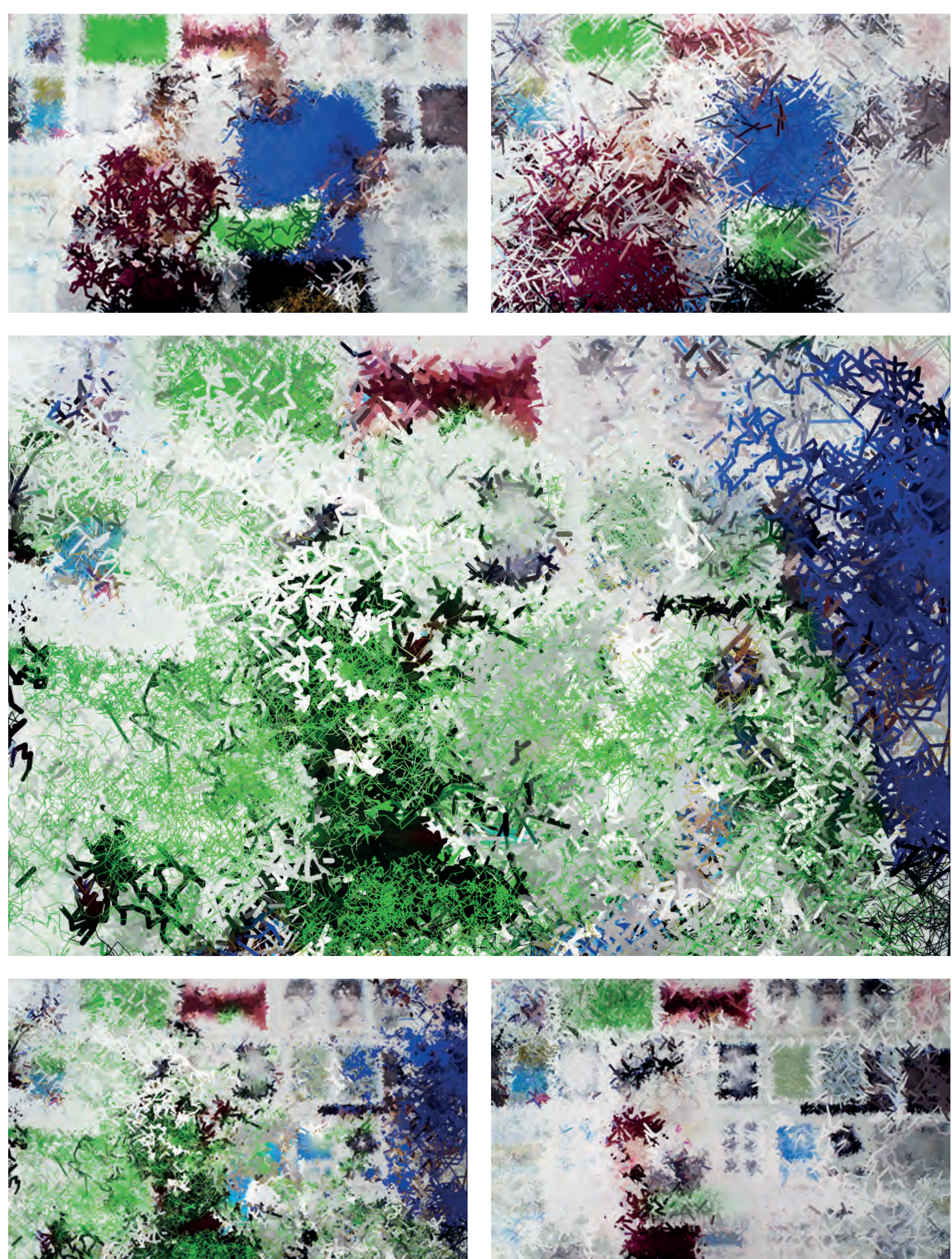
Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.



→ P_4_3_3_01 The people leave tracks in the image with their movements. The length of the lines varies throughout the drawing process, whereby the image is sometimes more detailed and sometimes more abstract.

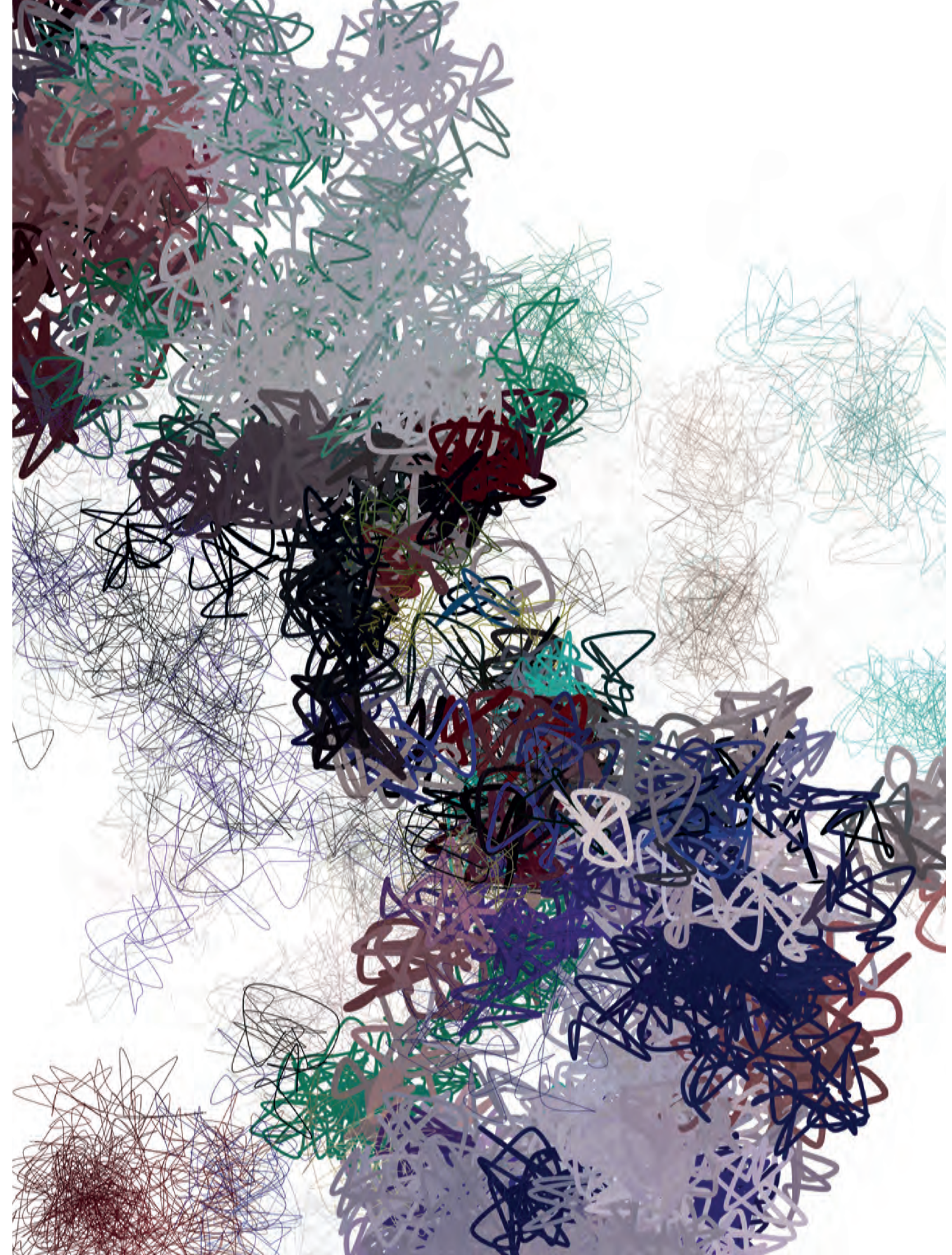
Copyright © 2018. Princeton Architectural Press. All rights reserved.

Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.



Copyright © 2018. Princeton Architectural Press. All rights reserved.

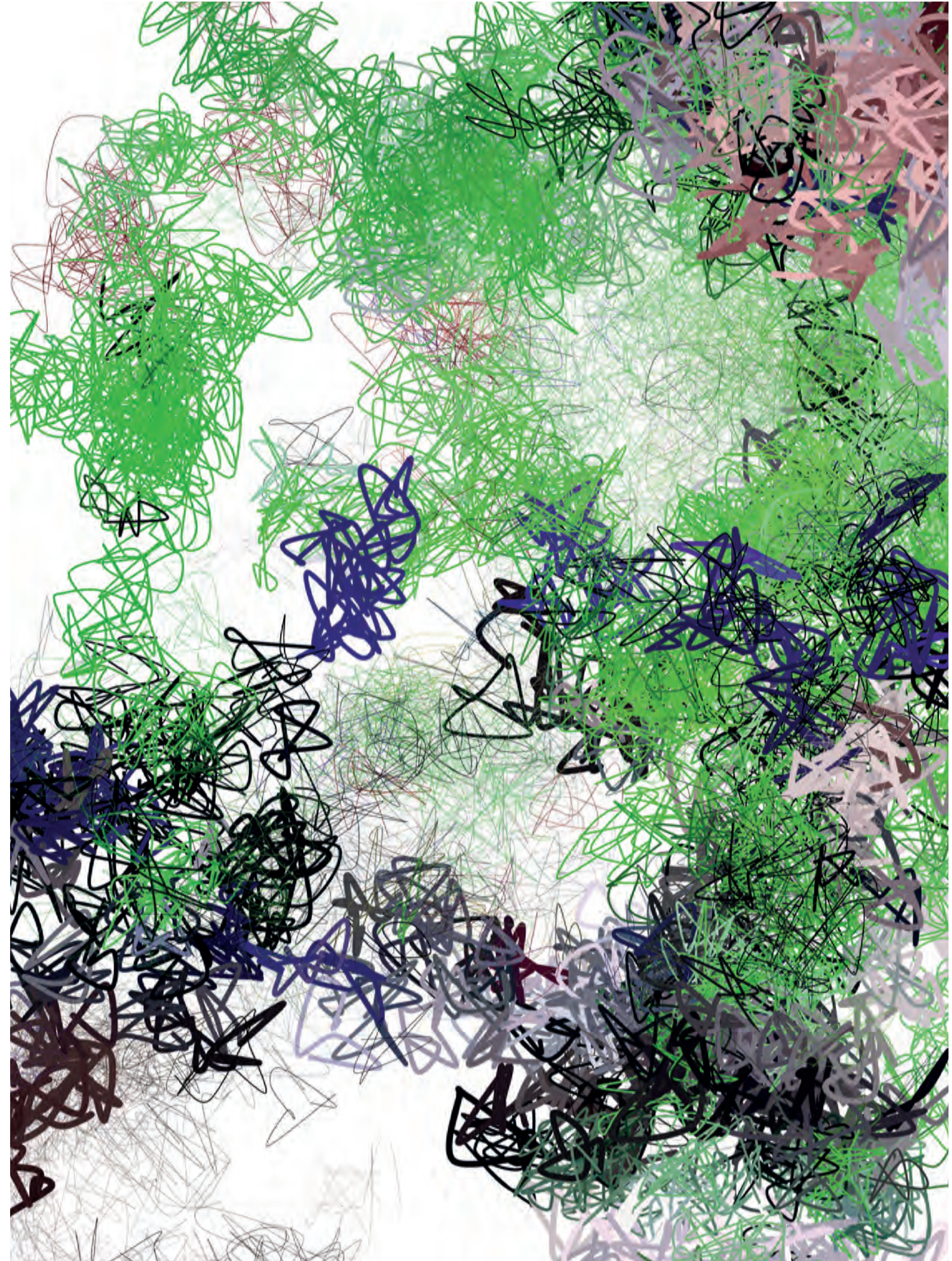
Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.



→ P_4_3_3_02 Three agents move around the display in this version of the program. The first agent's stroke value is defined by the pixel's hue; the second's by the pixel's saturation; and the third's by the pixel's brightness.

Copyright © 2018. Princeton Architectural Press. All rights reserved.

Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.



→ **P_4_3_3_02** When a subject moves in front of the camera, a collection of seemingly random scribbles come together to represent the subject's form.

Copyright © 2018. Princeton Architectural Press. All rights reserved.

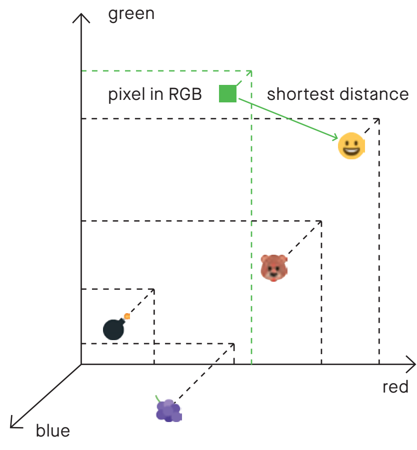
Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.

P.4.3.4 Emojis from pixel values

An emotional transformation from a raster element to a symbol: here the things that visualize feelings in an SMS become a small part of a greater whole. Any collection of thumbnails in this program can be the source material, and the pixel values determine which can join.

→ P_4_3_4_01

Color values can be interpreted as points in 3D space. For this program, it is necessary to find the shortest distance from a color value to a set of other color values, here the average colors of the individual emojis. There are various mathematical methods for this search. Particularly fast and comparatively easy to use is the search in a so-called k-dimensional tree. The functionality for this is provided by the library kdTree.js.




```
1  <script src="../../libraries/kd-tree/kdTree.js"
           type="text/javascript"></script>
2  <script src="data/emoji-average-colors.js"
           type="text/javascript"></script>
```

```
   var emojis = {
3    "1f4a3": {"averageColor": {"r":57, "g":57, "b":52}},
4    "1f43b": {"averageColor": {"r":187, "g":111,"b":88}},
5    "1f600": {"averageColor": {"r":227, "g":181, "b":70}},
     ...
   }
```

```
   function preload() {
     img = loadImage("data/pic.png");
6    icons = {};
     for (var name in emojis) {
       icons[name] = loadImage(emojisPath + "36x36/" +
                               name + ".png");
     }
   }
```

1 Two additional scripts are required for this program. These are loaded in index.html.

2 Information about the average colors of emoji files and their respective file names is contained in emoji-average-colors.js. The calculation of the average colors takes place in an additional program: → P_4_3_4_emoji_color_analyser.

3 file 1f4a3.png:

4 file 1f43b.png:

5 file 1f600.png:

6 In the variable icons, all images of the emojis are loaded and can be called later using their names (name).

Copyright © 2018. Princeton Architectural Press. All rights reserved.

Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.

```
function setup(){
  ...
7  var colors = [];
  for (var name in emojis) {
    var col = emojis[name].averageColor;
    col.name = name;
    colors.push(col);
  }

8  var distance = function(a, b){
    return pow(a.r - b.r, 2) +
           pow(a.g - b.g, 2) +
           pow(a.b - b.b, 2);
  }

9  tree = new kdTree(colors, distance, ["r", "g", "b"]);
}
```

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

  for (var gridX = 0; gridX < img.width; gridX++) {
    for(var gridY = 0; gridY < img.height; gridY++) {
      var posX = tileWidth * gridX;
      var posY = tileHeight * gridY;

10      var c = color(img.get(gridX, gridY));

11      var nearest = tree.nearest(
                    {r:red(c), g:green(c), b:blue(c)},
                    1);

12      var name = nearest[0][0].name;
      image(icons[name], posX, posY,
                         tileWidth, tileHeight);
    }
  }
  noLoop();
}
```

Keys: S: Save image

7 The search for the closest color begins here. Two things must be created: first, an array of points and their corresponding `r`, `g`, and `b` `colors`. Each entry in the `colors` array is also assigned the name `name`, in order to display the appropriate image files later.

8 Second, a function is defined that specifies how the distance between two points should be calculated. Typically, this is the length of the diagonal from color `a` to color `b` in the RGB color space.

9 With the help of the `kdTree` library, a so-called k-dimensional tree is created. For this, the newly created point list `colors` and the distance function `distance` are passed as parameters. In addition, a list of dimensions must be passed.

10 The image, `img`, to be displayed is scanned and the pixel color is stored in `c`.

11 The `nearest()` function in the `kdTree` library looks for the closest points to the one just passed. The last parameter indicates how many results should be returned; only one is needed here.

12 The search result is an array with the closest points. Each entry in it is itself an array with two elements: the point itself and its distance to the point passed in `nearest()`. In both arrays the first entry is required: `[0] [0]`. The image of the emoji can now be placed on the drawing canvas via the name.

Copyright © 2018. Princeton Architectural Press. All rights reserved.

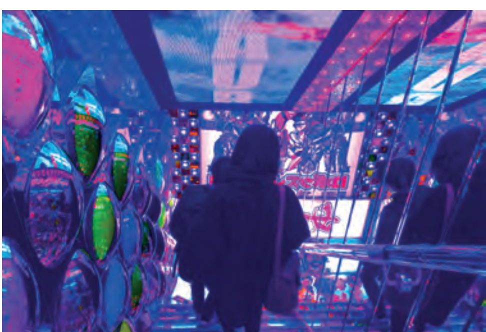
Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.

→ P_4_3_4_01 Every pixel becomes an emoji. Any image can be used, as long as it is large enough to include many color values.

Copyright © 2018. Princeton Architectural Press. All rights reserved.

original photo

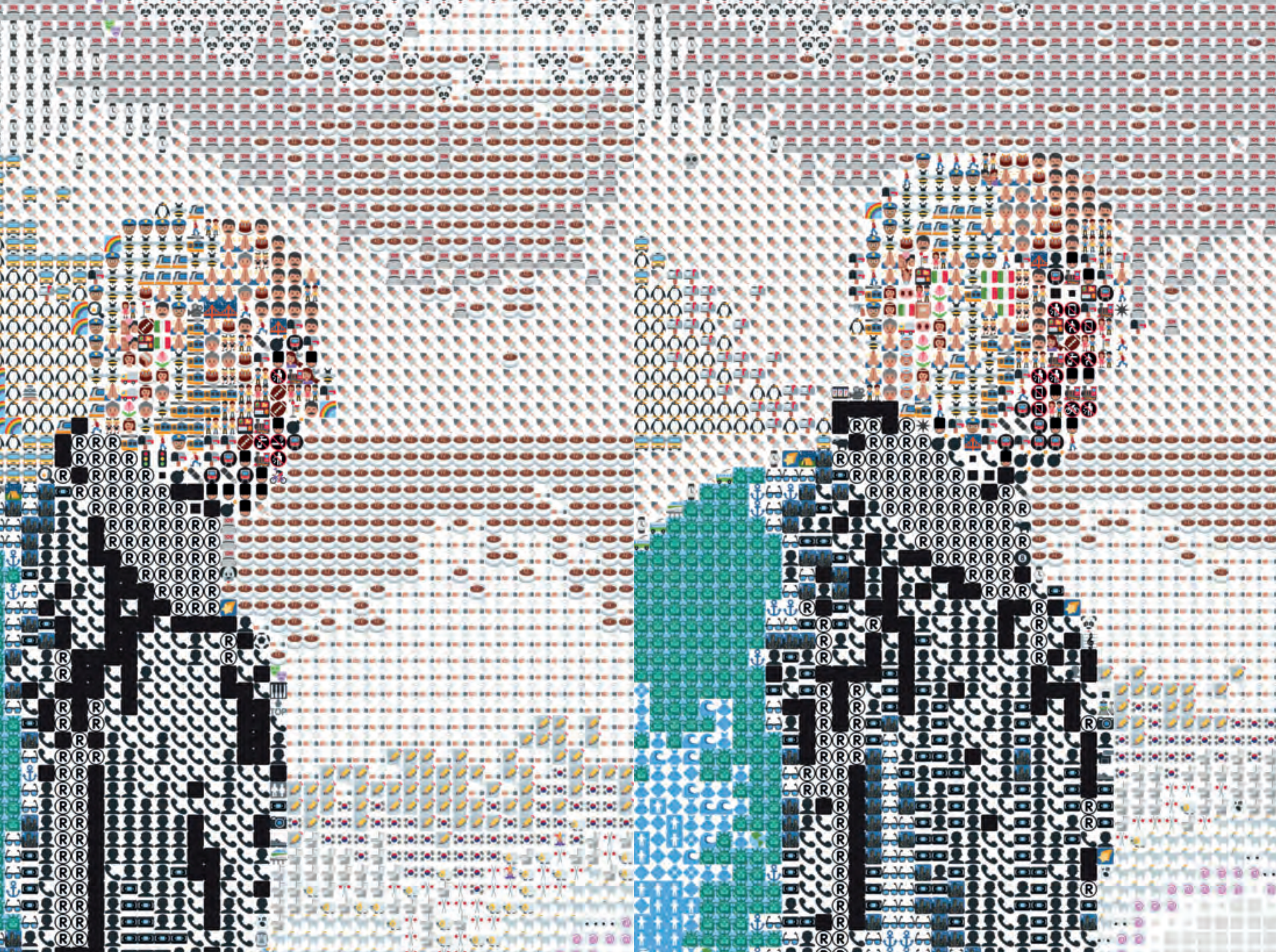
Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.



original photo

Copyright © 2018. Princeton Architectural Press. All rights reserved.

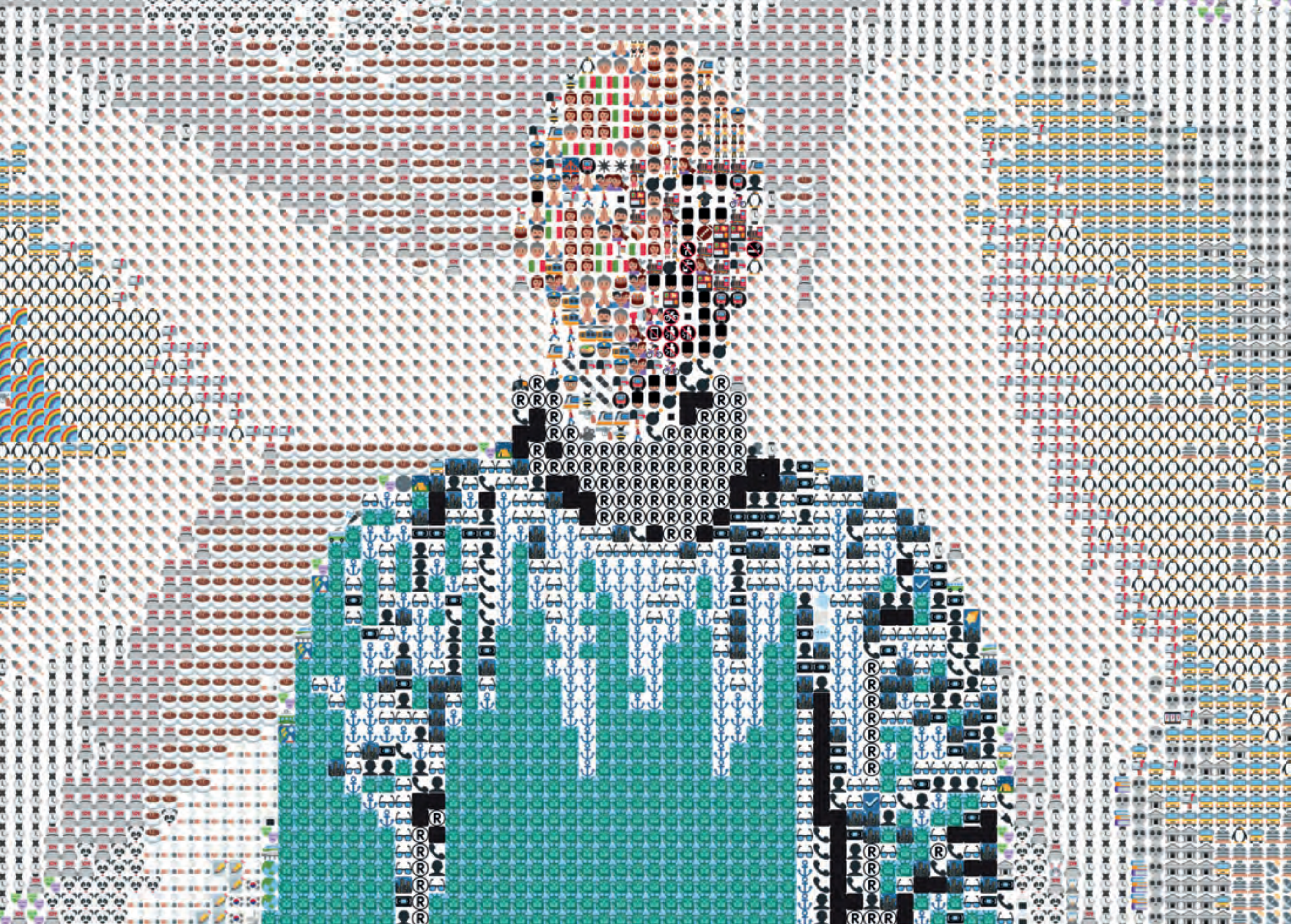
Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.



→ **P_4_3_4_02** An image from a webcam could also be the basis of a rasterization in emojis.

Copyright © 2018. Princeton Architectural Press. All rights reserved.

Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.



Copyright © 2018. Princeton Architectural Press. All rights reserved.

Gross, Benedikt, et al. Generative Design Revised : Visualize, Program, and Create with JavaScript in P5. js, Princeton Architectural Press, 2018. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/newschool/detail.action?docID=5515144.
Created from newschool on 2020-03-11 12:35:13.