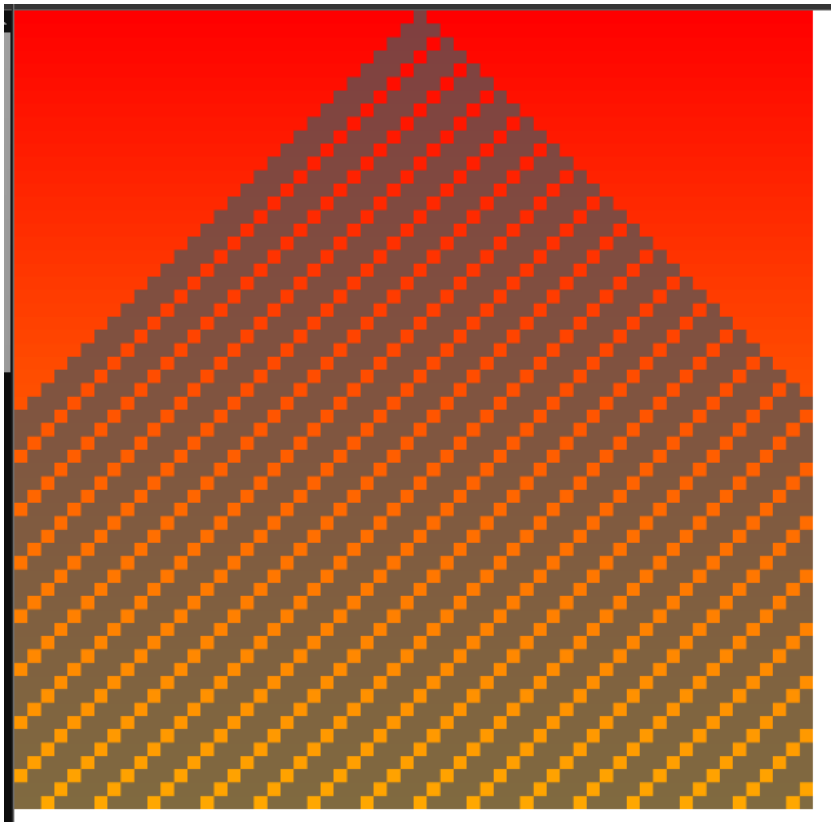
Algorithmic Art – Cellular Automata

1) 1D

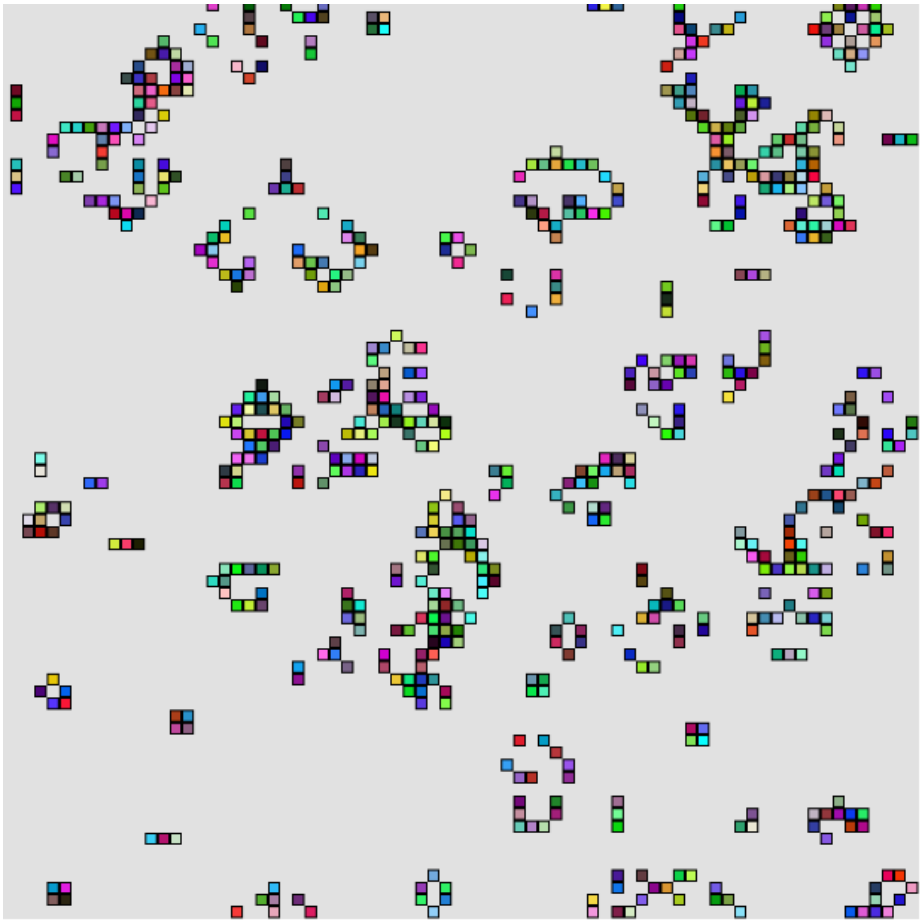
Rule Nr.

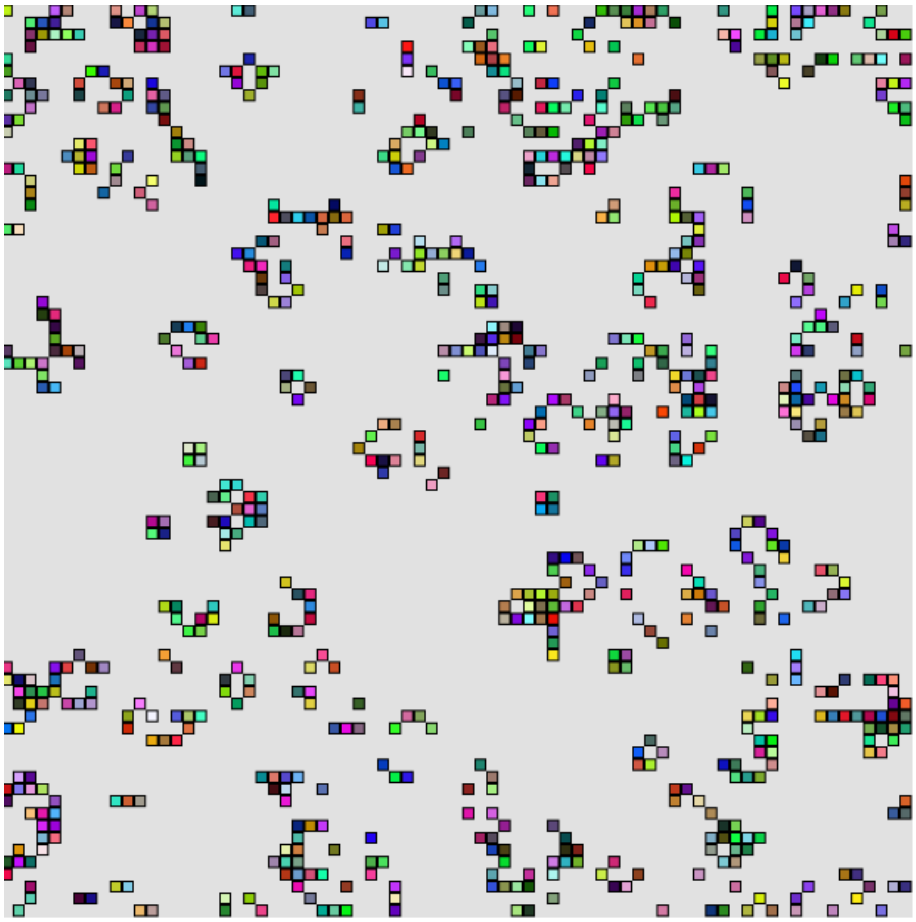
150



190

2) 2D





3) Code 2D

sketch.js Donut | sketch.js Cellular Automata2D | sketch.js Cellular Automata1D

Cellular Automata2D > sketch.js > draw

```
// Regel B3 besagt, dass eine tote Zelle mit genau drei lebenden Nachbarn wiederbelebt wird.
// Regel S23 besagt, dass eine lebende Zelle mit zwei oder drei lebenden Nachbarn am Leben bleibt.

// Erstellt ein zweidimensionales Array, das als Raster für das Spiel dient.
function make2DArray(cols, rows) {
  let arr = new Array(cols);
  for (let i = 0; i < arr.length; i++) {
    arr[i] = new Array(rows);
  }
  return arr;
}

let grid;
let cols;
let rows;
let resolution = 8;

// Hier wird das Raster erstellt
function setup() {
  createCanvas(600, 600);
  cols = width / resolution; // Divided by resolution (defined as 8) to know how many pixels fits into cols
  rows = height / resolution; // same as above but for rows

  // Hier wird für jede Zelle ein zuständiger Zustand festgelegt (2 Zustände-> lebendig (weiss) oder tot (schwarz))
  grid = make2DArray(cols, rows);
  for (let i = 0; i < cols; i++) {
    for (let j = 0; j < rows; j++) {
      grid[i][j] = floor(random(2));
    }
  }
}

function draw() {
  background(225);

  // Aktuelle Raster wird gezeichnet
  for (let i = 0; i < cols; i++) {
    for (let j = 0; j < rows; j++) {
      let x = i * resolution;
      let y = j * resolution;
      let r = random(255); // Zufällige Farben in den Zellen
      let g = random(255); // same
      let b = random(255); // same
      if (grid[i][j] == 1) {
        fill(r,g,b); // Füllt die Zellen mit der zufälligen Farbe
        stroke(0);
        rect(x, y, resolution - 1, resolution - 1);
      }
    }
  }

  let next = make2DArray(cols, rows); // Neues Raster (next) wird erstellt (gleiches wie grid)

  // Iteriert über das aktuelle Raster und berechnet den Zustand der Zellen im neuen Raster
  for (let i = 0; i < cols; i++) {
    for (let j = 0; j < rows; j++) {
      let state = grid[i][j];
      // 0 = tot, 1 = lebendig, es wird die Anzahl der lebenden Nachbarn gezählt
      let sum = 0;
      let neighbors = countNeighbors(grid, i, j);
      // Regeln B3 und S23
      if (state == 0 && neighbors == 3) {
        next[i][j] = 1;
      } else if (state == 1 && (neighbors < 2 || neighbors > 3)) {
        next[i][j] = 0;
      } else {
        next[i][j] = state;
      }

    }
  }

  grid = next; // Das aktuelle Raster (grid) wird mit dem neuen Raster (next) ersetzt.



}

// Zählt die Anzahl der lebenden Nachbarn einer Zelle
function countNeighbors(grid, x, y) {
  let sum = 0;
  for (let i = -1; i < 2; i++) {
    for (let j = -1; j < 2; j++) {
      let col = (x + i + cols) % cols;
      let row = (y + j + rows) % rows;
      sum += grid[col][row];
    }
  }
  sum -= grid[x][y]; //
  return sum;
}
```