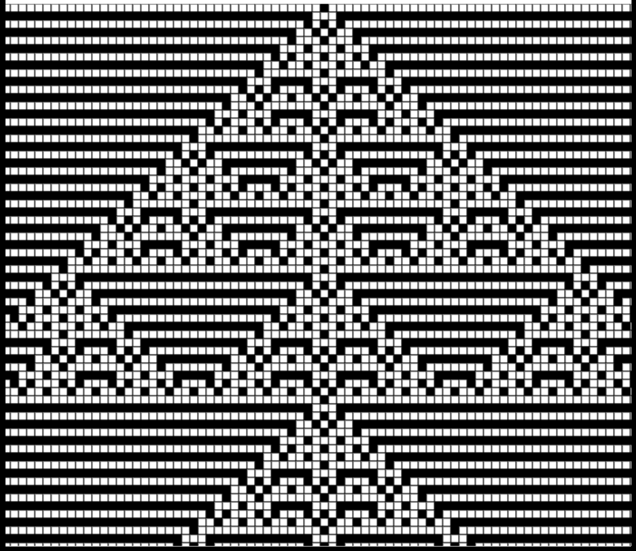
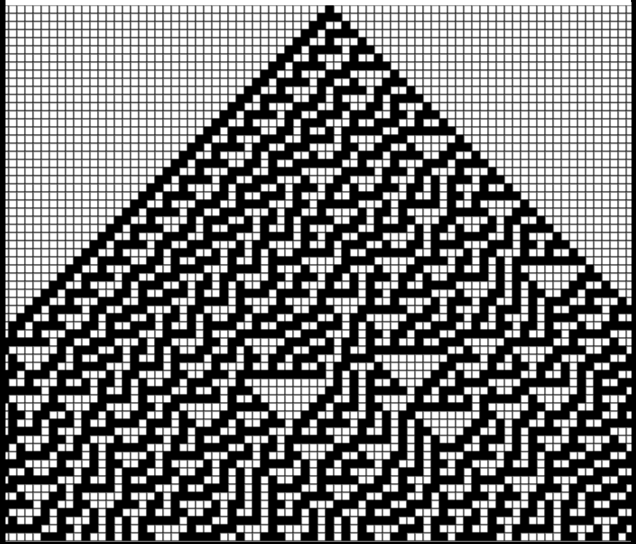
```
//(...)

function myRule(left, middle, right) {
    let binaryString = '' + left + middle + right;
    switch (binaryString) {
        case '111': return 0;
        case '110': return 1;
        case '101': return 1;
        case '100': return 0;
        case '011': return 1;
        case '010': return 0;
        case '001': return 0;
        case '000': return 1;
    }
}

function generateNextGen() {
    for (let i = 0; i < cols; i++) {
        let left = currentGen[(i - 1 + cols) % cols];
        let middle = currentGen[i];
        let right = currentGen[(i + 1) % cols];
        nextGen[i] = myRule(left, middle, right);
    }
    currentGen = nextGen.slice();
}

function drawGeneration(gen, row) {
    for (let i = 0; i < gen.length; i++) {
        let x = i * cellSize;
        let y = row * cellSize;
        if (gen[i] === 1) {
            fill(0);
        } else {
            fill(255);
        }
        stroke(0);
        rect(x, y, cellSize, cellSize);
    }
}

//(...)
```

My Rule

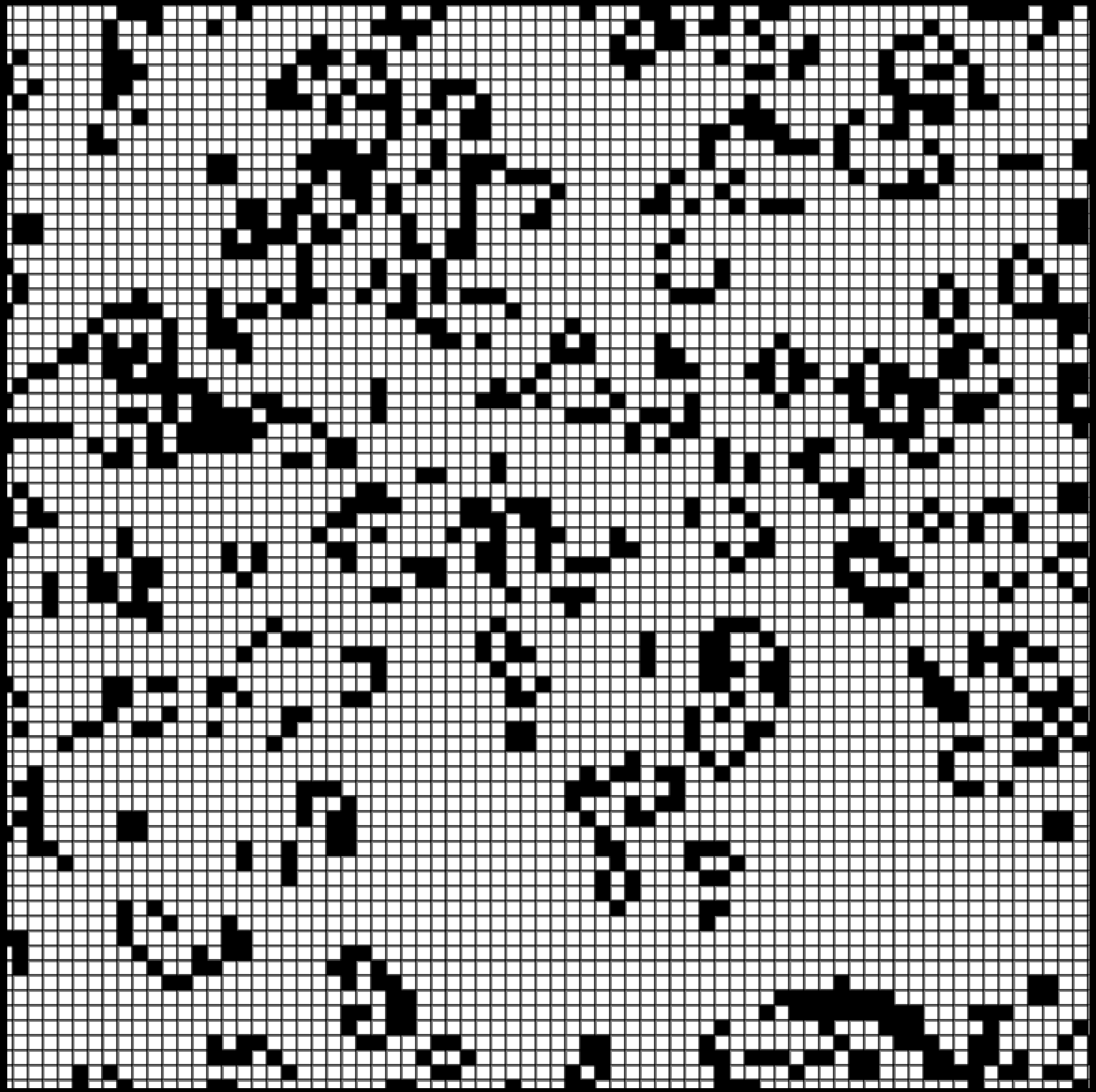


Rule 30



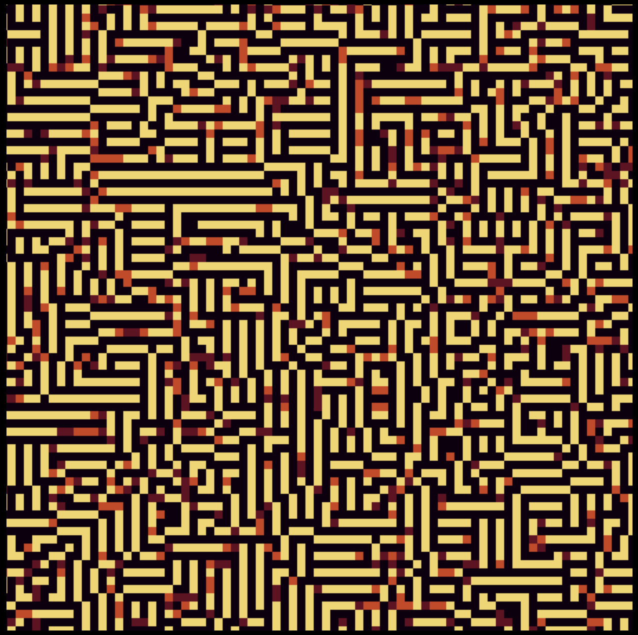
Zelluläre Automaten in 1D

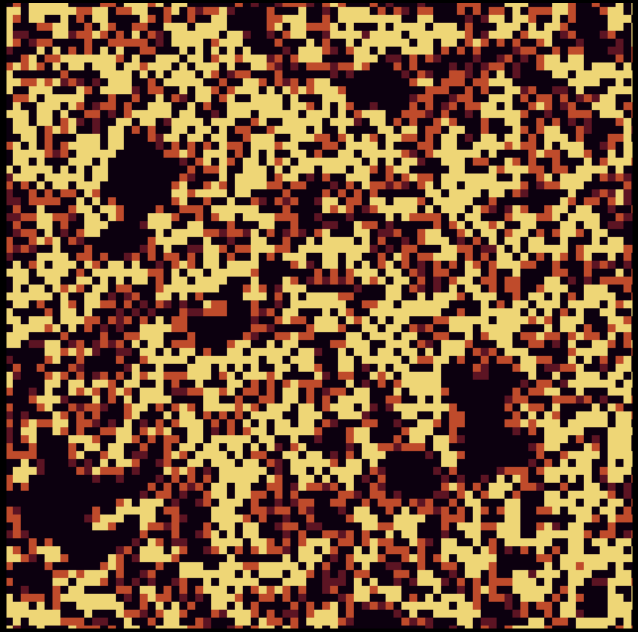
```
//(...)

function draw() {
    background(255);

    for (let i = 0; i < cols; i++) {
        for (let j = 0; j < rows; j++) {
            let x = i * cellSize;
            let y = j * cellSize;
            if (grid[i][j] === 1) {
                fill(0);
            } else {
                fill(255);
            }
            stroke(255);
            rect(x, y, cellSize, cellSize);
        }
    }
    generateNextGen();
}

function generateNextGen() {
    for (let i = 0; i < cols; i++) {
        for (let j = 0; j < rows; j++) {
            let state = grid[i][j];
            let neighbors = countNeighbors(grid, i, j);

            if (state === 0 && neighbors === 3) {
                nextGrid[i][j] = 1; // Birth
            } else if (state === 1 && (neighbors < 2 || neighbors > 3)) {
                nextGrid[i][j] = 0; // Death
            } else {
                nextGrid[i][j] = state; // Stays the same
            }
        }
    }
    let temp = grid;
    grid = nextGrid;
    nextGrid = temp;
}

//(...)
```



Zellulärer Automat in 2D «Game of Life»

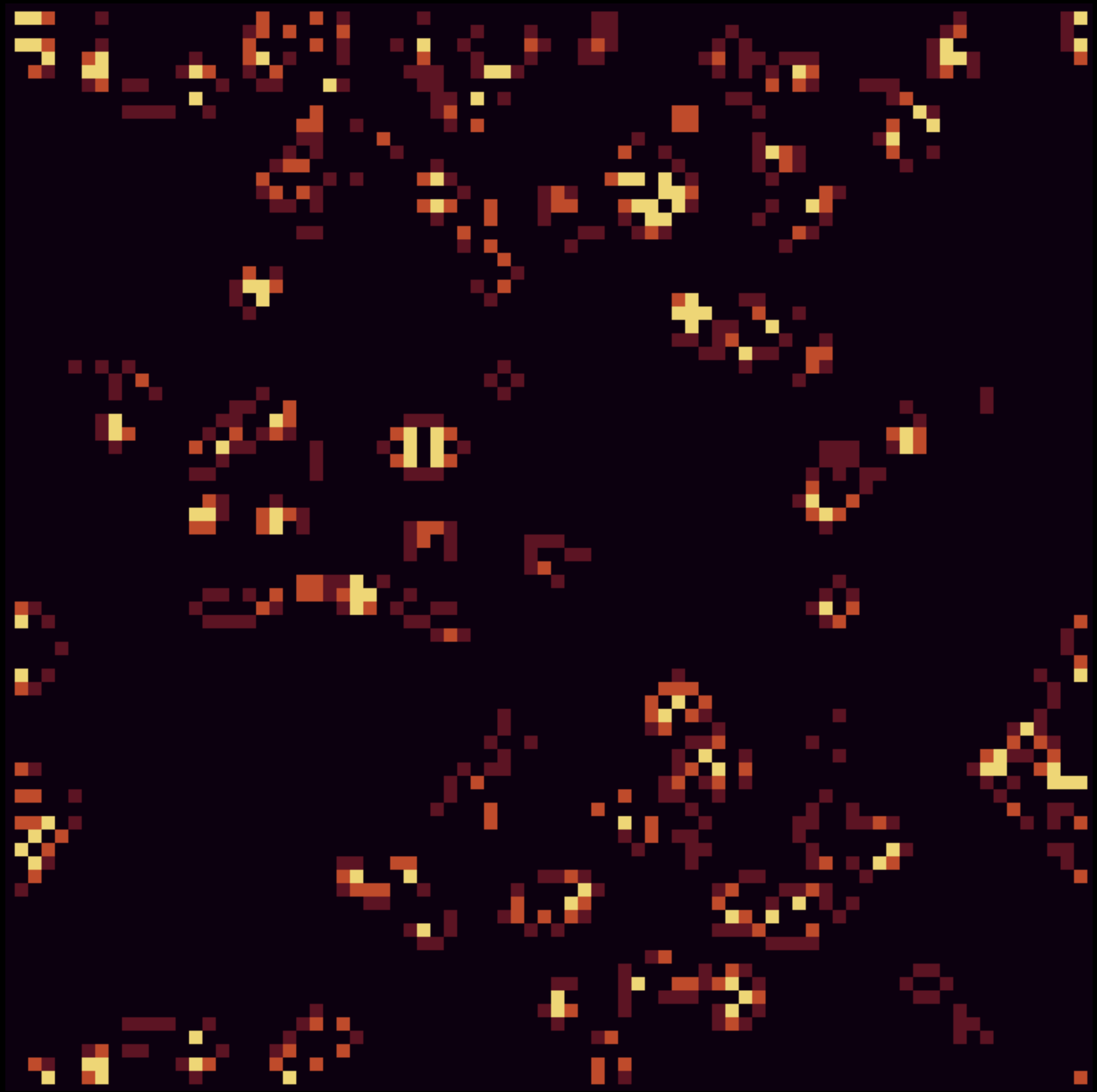
```
//(...)

for (let i = 0; i < cols; i++) {
        for (let j = 0; j < rows; j++) {
            let state = grid[i][j];
            let neighbors = countNeighbors(grid, i, j);

            if (neighbors > 10 && neighbors < 14) {
                if (state === 3) {
                    nextGrid[i][j] = state; // Stay the same
                } else {
                    nextGrid[i][j] = state + 1; // Upgrade
                }
            } else if (neighbors < 8 || neighbors > 16) {
                if (state === 0) {
                    nextGrid[i][j] = state; // Stay the same
                } else {
                    nextGrid[i][j] = state - 1; // Downgrade
                }
            } else {
                nextGrid[i][j] = state; // Stay the same
            }
        }
}

//(...)

function countNeighbors(x, y) {
    let sum = 0;
    for (let i = -1; i <= 1; i++) {
        for (let j = -1; j <= 1; j++) {
            let col = (x + i + cols) % cols;
            let row = (y + j + rows) % rows;
            sum += grid[col][row];
        }
    }
    sum -= grid[x][y];
    return sum;
}

//(...)
```





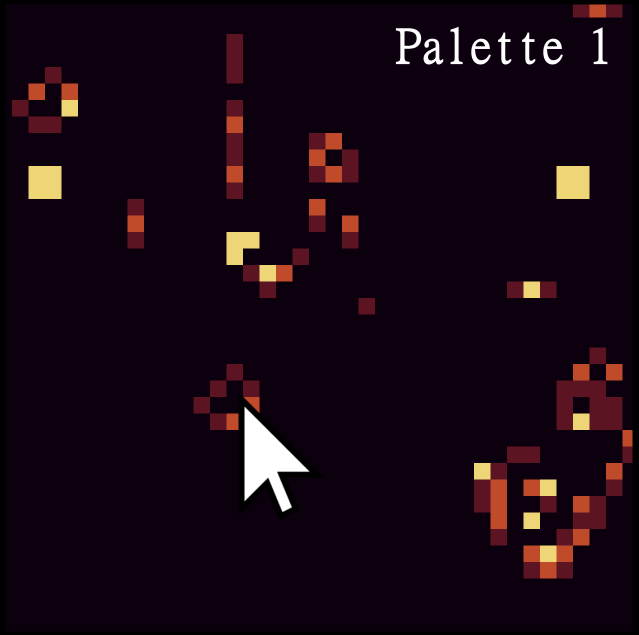
Erste Versuche mit mehr Zuständen

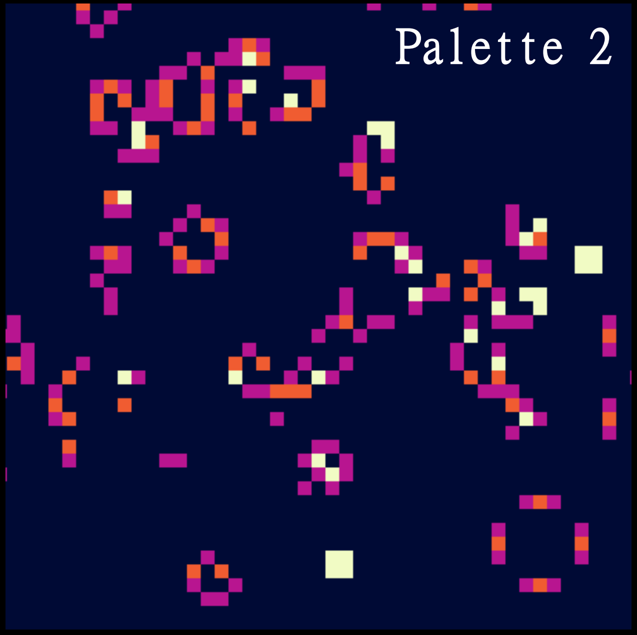
```
//(...)

for (let i = 0; i < cols; i++) {
        for (let j = 0; j < rows; j++) {
            let state = grid[i][j];
            let neighbors = countNeighbors(i, j);
            if (neighbors === 3) {
                if (state === 3) {
                    nextGrid[i][j] = state; // Stay the same
                } else {
                    nextGrid[i][j] = state + 1; // Upgrade
                }
            } else if (neighbors < 2 || neighbors > 3) {
                nextGrid[i][j] = 0; // Death
            } else {
                nextGrid[i][j] = state; // Stay the same
            }
        }
}

function countNeighbors(x, y) {
    let sum = 0;
    for (let i = -1; i <= 1; i++) {
        for (let j = -1; j <= 1; j++) {
            let col = (x + i + cols) % cols;
            let row = (y + j + rows) % rows;
            if (grid[col][row] !== 0) {
                sum++;
            }
        }
    }
    if (grid[x][y] !== 0) {
        sum--;
    }
    return sum;
}

//(...)
```

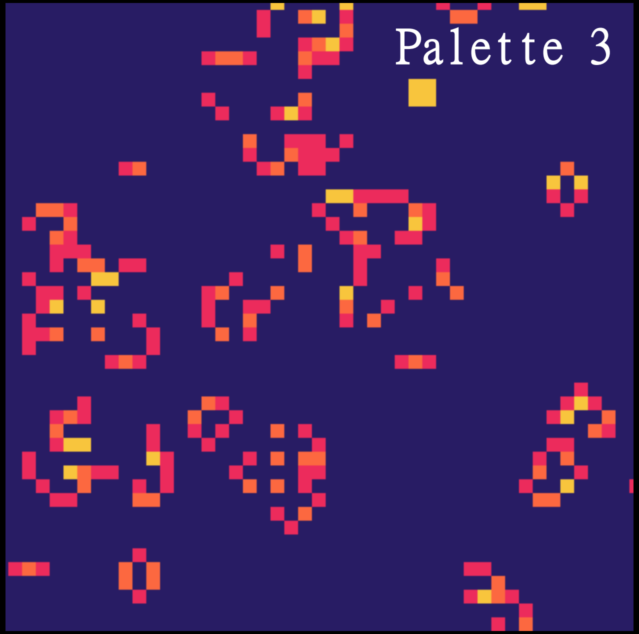


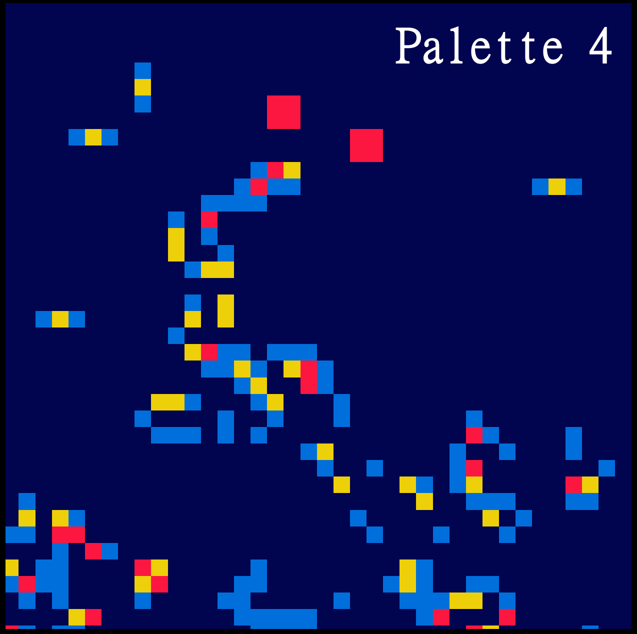
«Game of Life» mit mehr Zuständen

```
//(...)

//Palette 1
let color0 = [12, 0, 15]; // Dark purple
let color1 = [92, 20, 35]; // Dark red
let color2 = [191, 75, 43]; // Orange
let color3 = [238, 214, 118]; // Yellow
/*
//Palette 2
let color0 = [40, 28, 100]; // Dark purple
let color1 = [236, 43, 92]; // Pink
let color2 = [252, 104, 64]; // Orange
let color3 = [248, 197, 61]; // Yellow

//Palette 3
let color0 = [0, 10, 53]; // Dark purple
let color1 = [184, 21, 144]; // Pink
let color2 = [240, 92, 49]; // Orange
let color3 = [241, 251, 196]; // Yellow

//Palette 4
let color0 = [1, 4, 79]; // Dark blue
let color1 = [0, 111, 219]; // Blue
let color2 = [237, 208, 9]; // Yellow
let color3 = [252, 23, 65]; // Red
*/

function checkInput() {
    if (mouseIsPressed) {
        let x = Math.floor(mouseX / cellSize);
        let y = Math.floor(mouseY / cellSize);
        if (x >= 0 && x < cols && y >= 0 && y < rows) {
            for (let i = -1; i <= 1; i++) {
                for (let j = -1; j <= 1; j++) {
                    let col = (x + i + cols) % cols;
                    let row = (y + j + rows) % rows;
                    grid[col][row] = Math.floor(random(4));
                }
            }
        }
    }
}

//(...)
```









Experimente mit Farben & Interaktivität