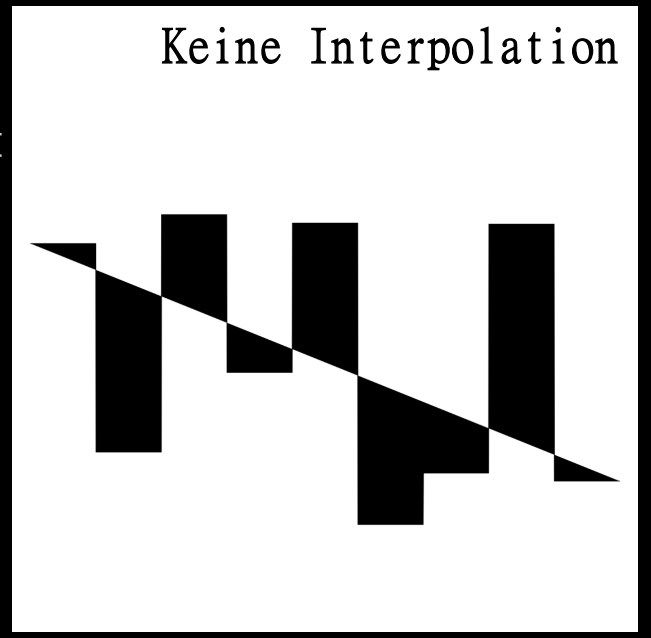
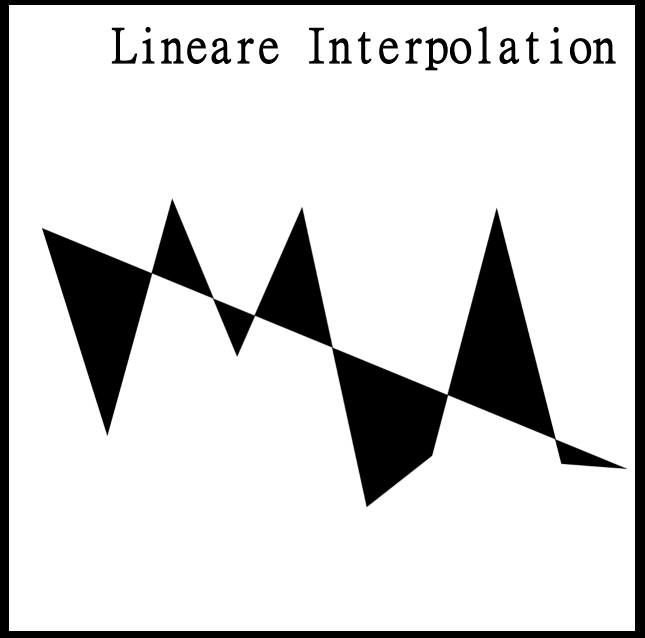
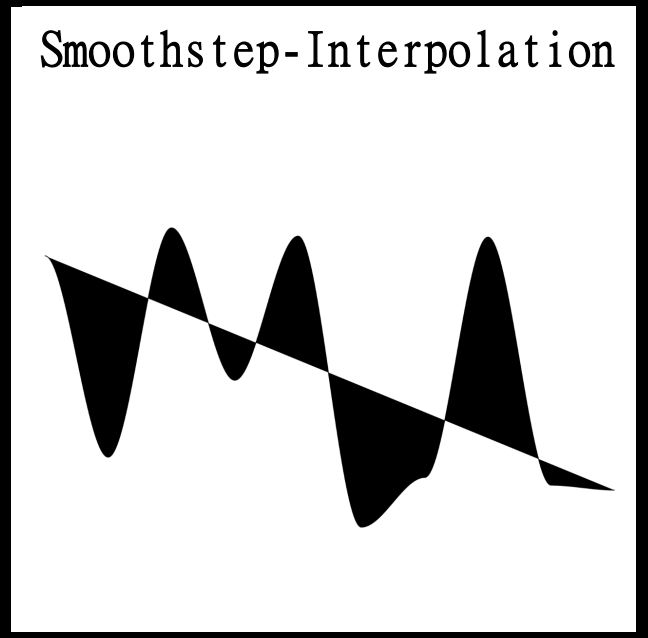
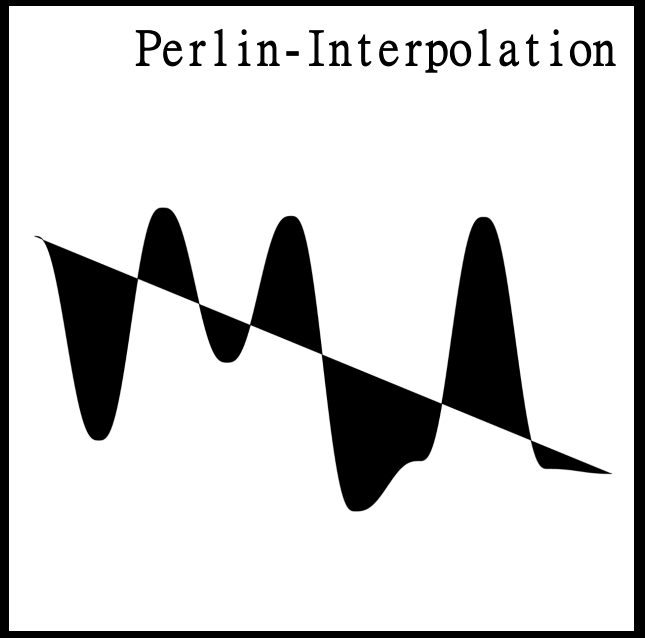
```
//(...)

for (let i = 0; i < numPoints; i++) {
    values.push(random(height / 2));
}
drawInterpolatedPoints(values, 800, modes[mode]);

function drawInterpolatedPoints(values, yOffset, interpolationFunc) {
    beginShape();
    for (let i = 0; i < values.length - 1; i++) {
        for (let t = 0; t < resolution; t++) {
            let x = (width / numPoints) * (i + t / resolution)
                        + width / numPoints / 2;
            let y = yOffset - interpolationFunc(values[i],
                        values[i + 1], t / resolution);
            vertex(x, y);
        }
    }
    if (interpolationFunc == noInterpolation) {
        vertex((width / numPoints) * (values.length - 1) + width /
numPoints / 2, yOffset - values[values.length - 1]);
    }
    endShape();
}

function linearInterpolation(a, b, t) {
    return a * (1 - t) + b * t;
}
function cosineInterpolation(a, b, t) {
    let ft = t * PI;
    let f = (1 - cos(ft)) * 0.5;
    return a * (1 - f) + b * f;
}
function perlinInterpolation(a, b, t) {
    return a * (1 - (6 * Math.pow(t, 5) - 15 * Math.pow(t, 4) +
            10 * Math.pow(t, 3))) + b * (6 * Math.pow(t, 5) -
            15 * Math.pow(t, 4) + 10 * Math.pow(t, 3));
}
function smoothstepInterpolation(a, b, t) {
    t = t * t * (3 - 2 * t);
    return a * (1 - t) + b * t;
}

//(...)
```

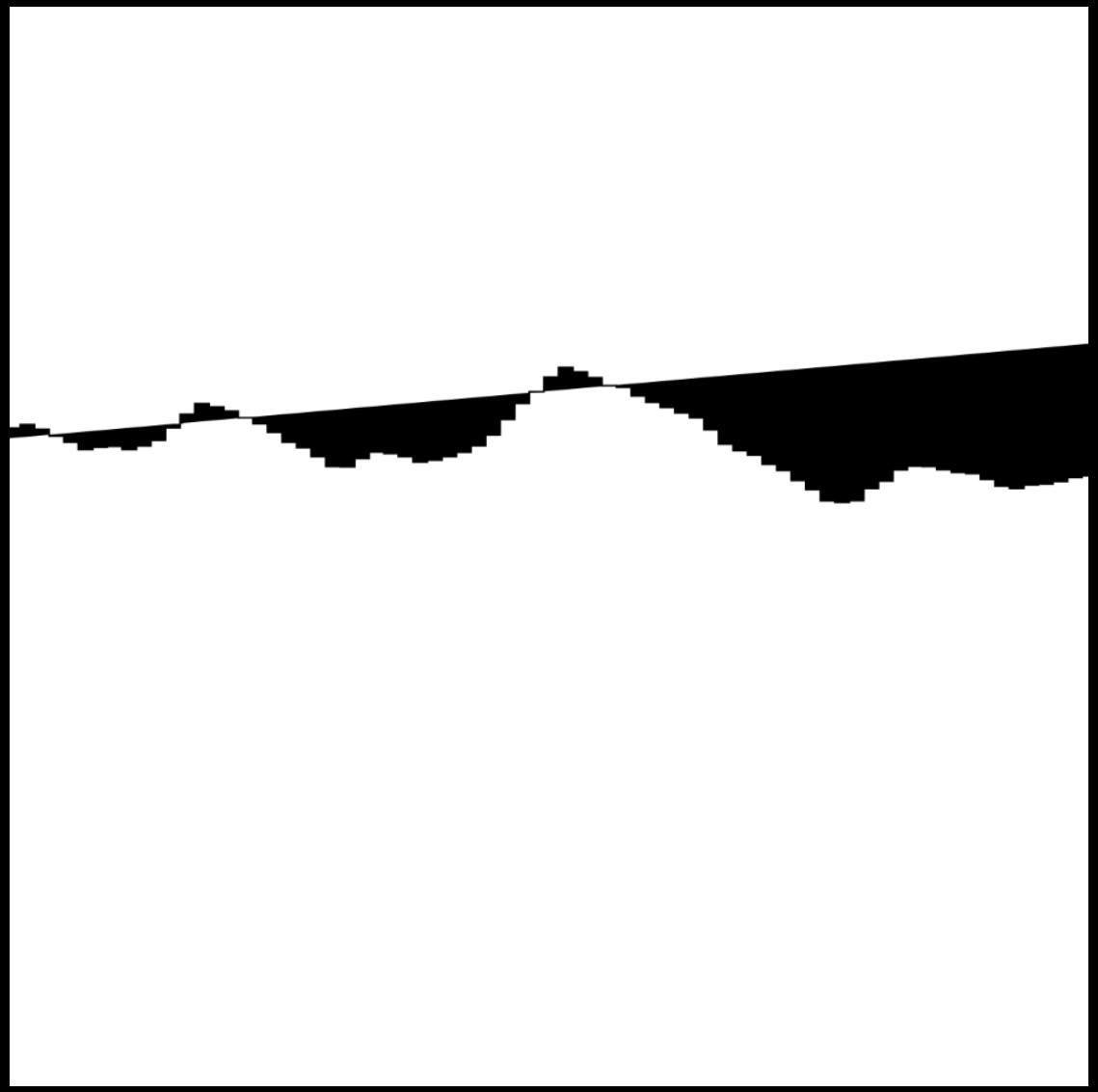








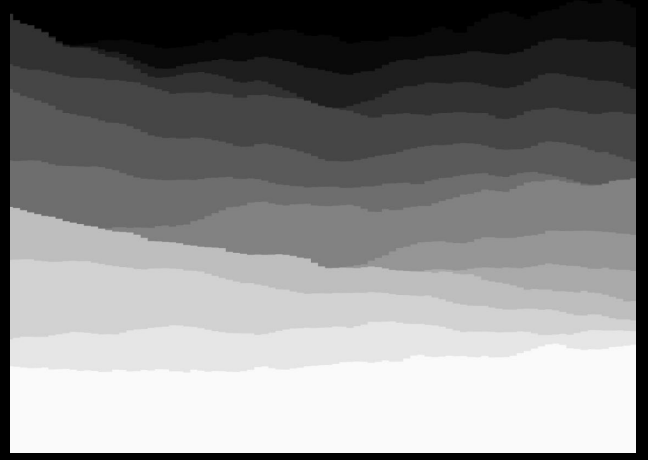
```
//(...)

let values = [];
let numPoints = 100;
let persistence = 0.5;

function setup() {
    let canvas = createCanvas(windowWidth, windowHeight);
    canvas.position(0, 0);
    stroke(0);
    fill(0);

    generateFractalNoiseValues();
}

function generateFractalNoiseValues() {
    values = [];
    for (let i = 0; i < numPoints; i++) {
        values.push(fractalNoise(i / numPoints) * height / 2);
    }
}

function fractalNoise(x) {
    let total = 0;
    let frequency = 1;
    let amplitude = 1;
    let maxAmplitude = 0;

    for (let i = 0; i < 8; i++) { // Number of octaves
        total += noise(x * frequency) * amplitude;
        frequency *= 2;
        maxAmplitude += amplitude;
        amplitude *= persistence;
    }

    return total / maxAmplitude;
}

//(...)
```



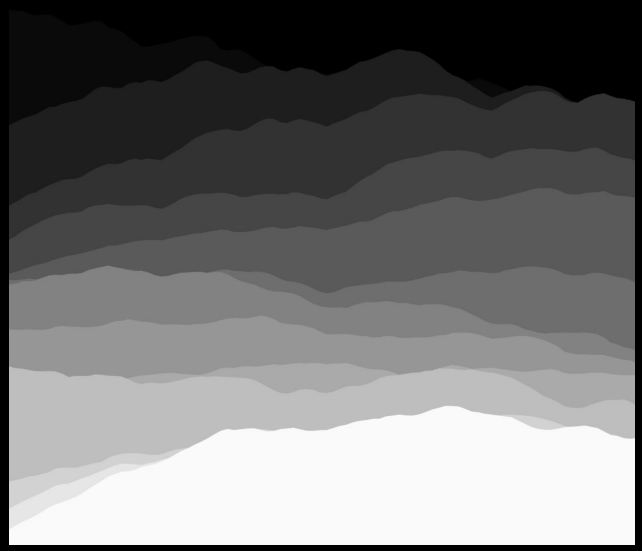
Fractal Noise mit Persistenz 1/2

```
//(...)

let tints = [
    [0.6, 1.2, 1.6], // Blue
    [1.2, 1, 0.5], // Yellow
    [1, 1, 1], // Gray
    [1.5, 0.5, 1], // Pink
    [1, 1, 1.5], // Purple
    [1, 1.5, 1], // Green
    [1.5, 1, 1], // Red
    [1.5, 1.5, 1], // Orange
    [1, 1.5, 1.5], // Teal
];
let tint = tints[0];

let colors = [
    [10, 10, 10],
    [20, 20, 20],
    //(...)
];

function setup() {
    let canvas = createCanvas(windowWidth * 1.1, windowHeight);
    canvas.position(-windowWidth * 0.05, 0);
    noStroke();

    background(0)

    generateFractalNoiseValues();
}

function draw() {
    for (let i = 0; i < colors.length; i++) {
        fill(colors[i]);
        // Draw interpolated points
        drawInterpolatedPoints(values[i],
            (i / colors.length + 1) * 500, modes[mode]);
    }
}

//(...)
```

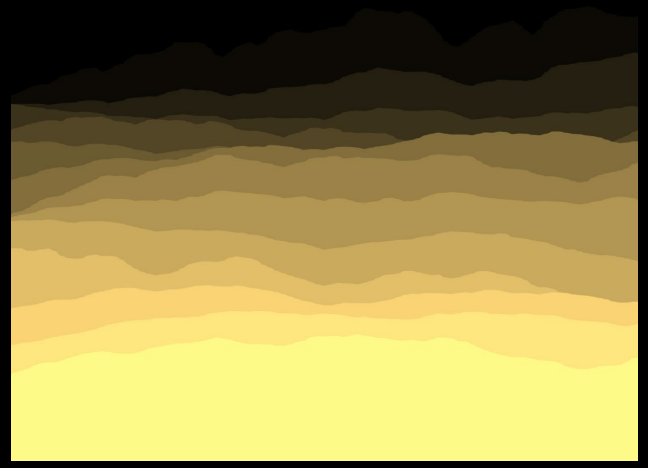
Keine Interpolation

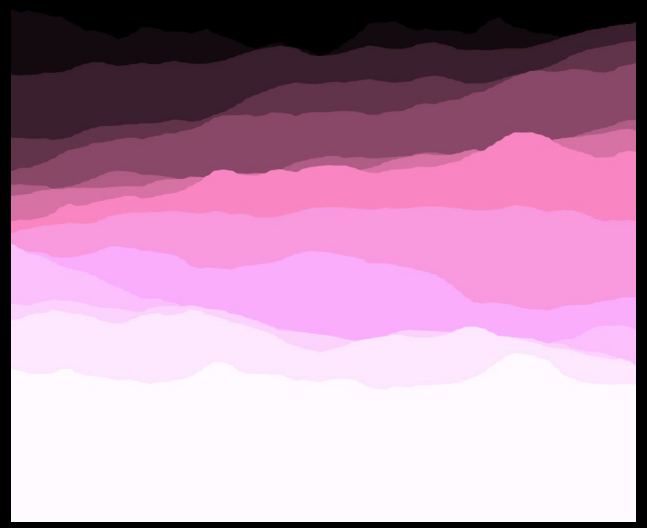


Lineare Interpolation



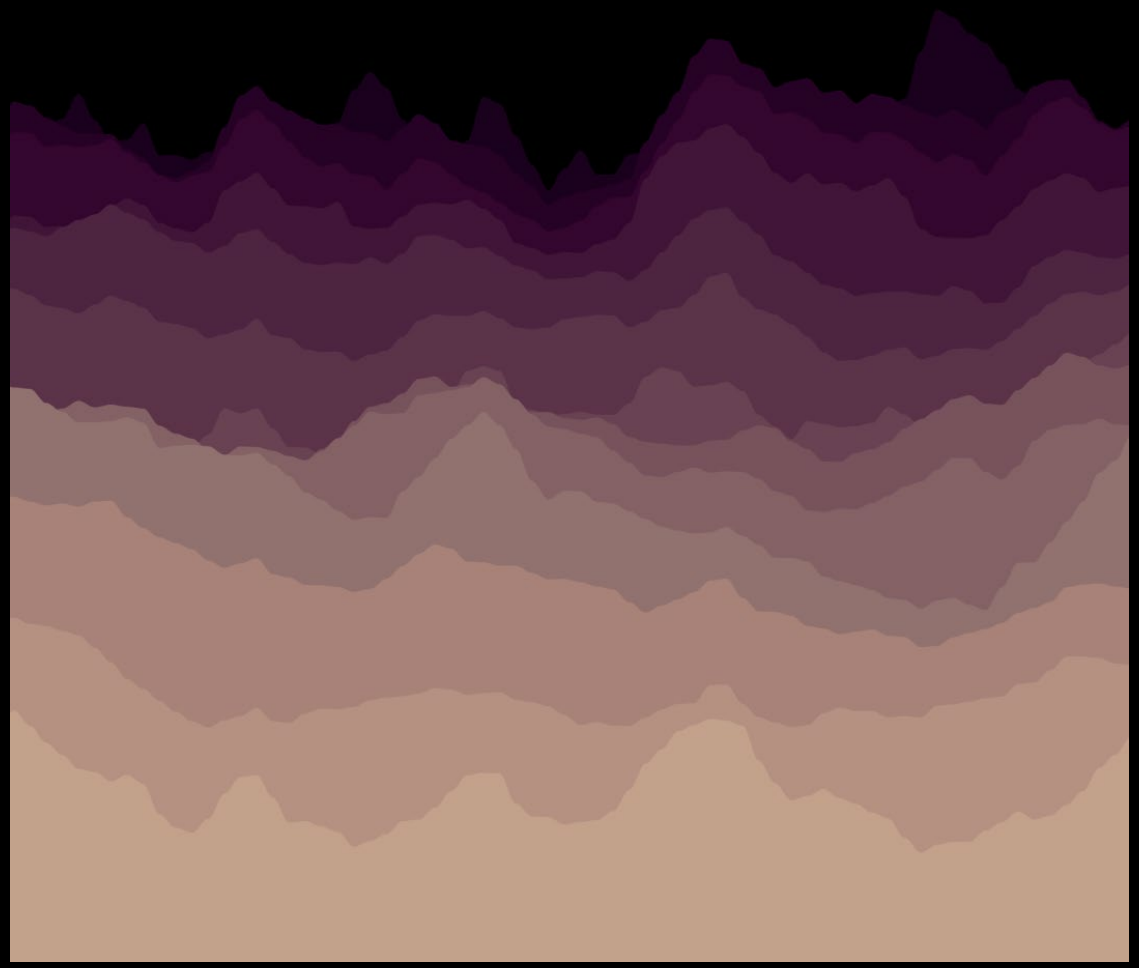
Varianten mit Smoothstep-Interpolation





Layers durch 2D-Noise

```
//(...)

function draw() {
    for (let i = 0; i < colors.length; i++) {
        colors[i] = [colors[i][0] * 0.97,
            colors[i][1] * 0.95, colors[i][2] * 0.96];
    }

    for (let i = 0; i < colors.length; i++) {
        fill(colors[i]);
        // Draw interpolated points
        drawInterpolatedPoints(values[i],
            (i / colors.length + 1) * 500, modes[mode]);
    }
}

function mouseClicked() {
    tint = tints[Math.floor(random(tints.length))];

    colors = [
        [30, 0, 50],
        [45, 0, 65],
        [60, 5, 80],
        [75, 30, 95],
        [90, 55, 110],
        [105, 80, 125],
        [120, 105, 140],
        [135, 130, 155],
        [150, 155, 170],
        [165, 180, 185],
        [180, 205, 200],
        [195, 230, 215],
        [210, 255, 230]
    ];

    for (let i = 0; i < colors.length; i++) {
        colors[i] = [colors[i][0] * tint[0],
            colors[i][1] * tint[1], colors[i][2] * tint[2]];
    }
}

//(...)
```



Farbverläufe & Fade-Out