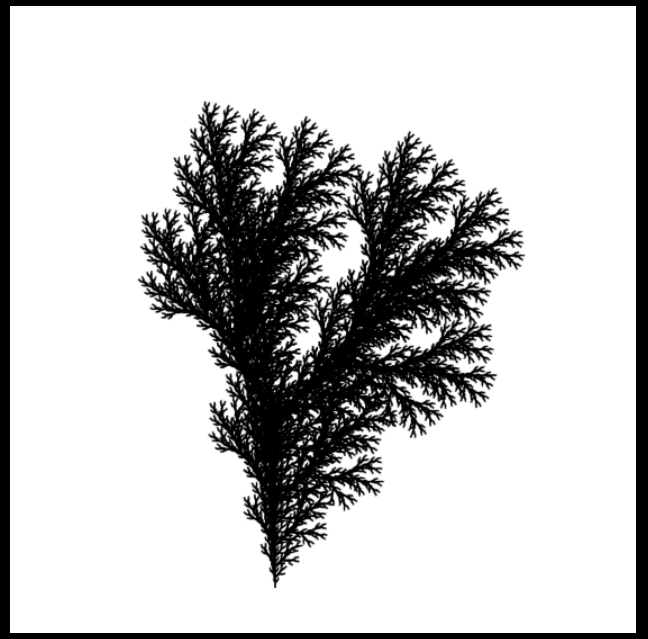
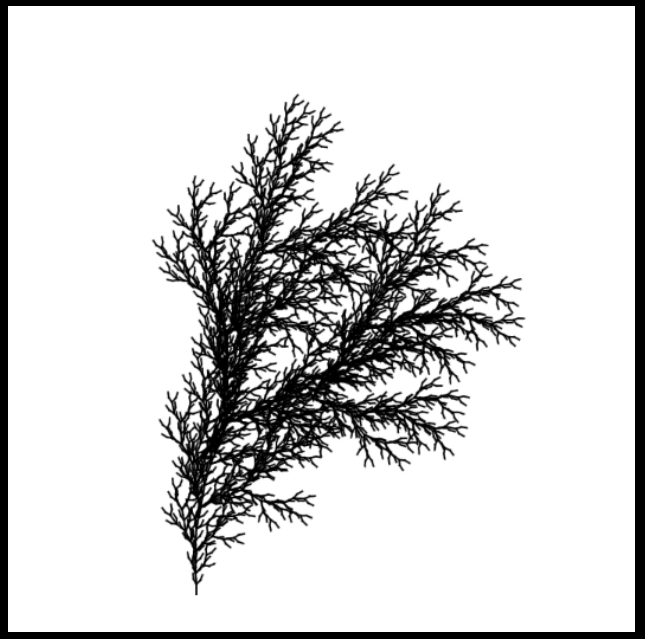
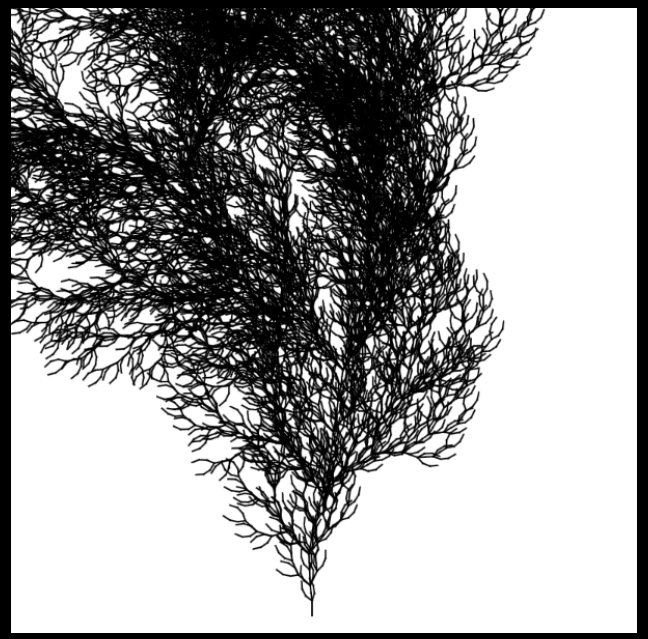
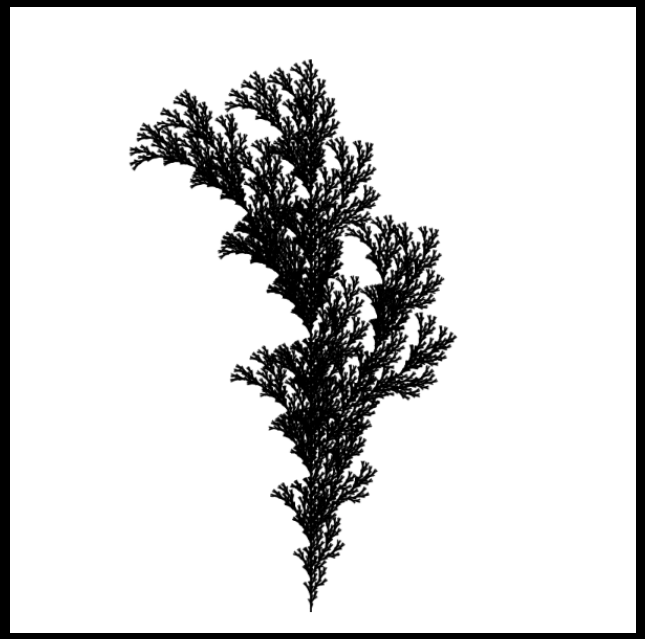
```
//(...)

function generate() {
    let nextSentence = "";
    for (let i = 0; i < sentence.length; i++) {
        let current = sentence.charAt(i);
        let found = false;
        for (let j = 0; j < rules.length; j++) {
            if (current == rules[j].predecessor) {
                found = true;
                nextSentence += rules[j].successor;           }
        }
        if (!found) {
            nextSentence += current;
        }
    }
    sentence = nextSentence;
    len *= 0.55;
    turtle();
}

function turtle() {
    //(...)
    for (let i = 0; i < sentence.length; i++) {
        angle = radians(random(20, 30));
        let current = sentence.charAt(i);
        if (current == "F") {
            line(0, 0, 0, -len);
            translate(0, -len);
        } else if (current == "G") {
            translate(0, -len);
        } else if (current == "+") {
            rotate(angle);
        } else if (current == "-") {
            rotate(-angle);
        } else if (current == "[") {
            push();
        } else if (current == "]") {
            pop();
        }
    }
}

//(...)
```



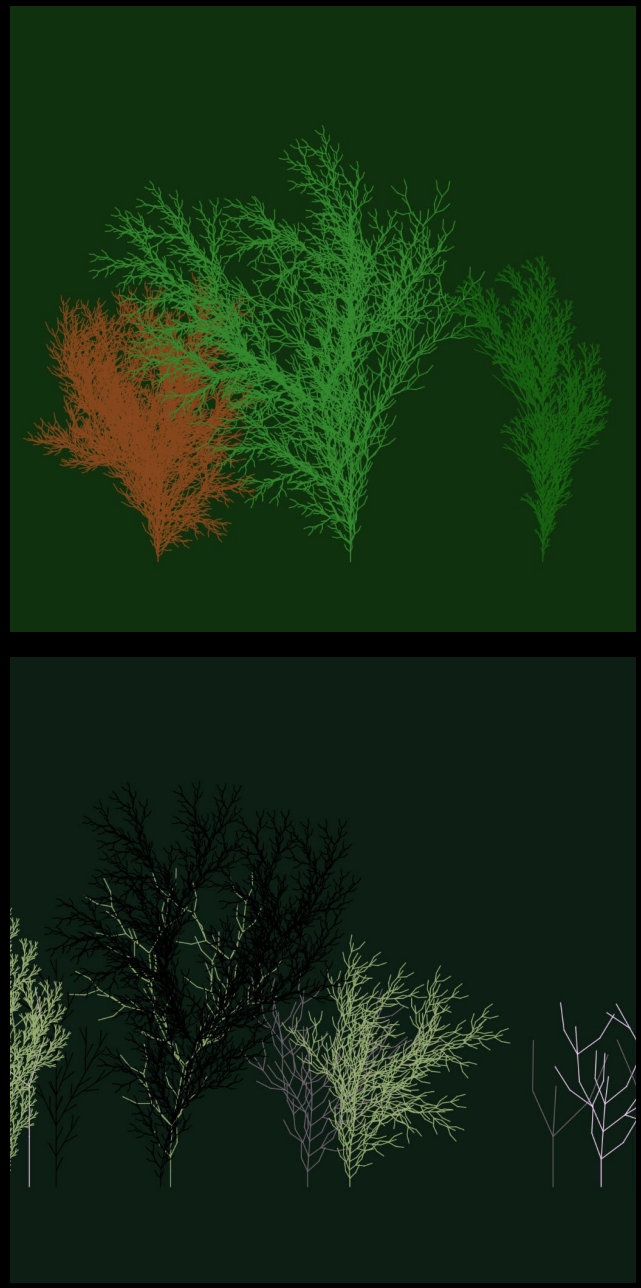






Baum mit L-System

```
//(...)

function setup() {
    let canvas = createCanvas(windowWidth * 1.1, windowHeight * 1.1);
    canvas.position(-windowWidth * 0.05, 0);
    angle = radians(12);
    background(10, 30, 20);
    nTrees = Math.floor(width / 20);
    let colors = [
        '#98B06F',
        '#776472',
        'black'
    ];
    for (let i = 0; i < nTrees; i++) {
        sentence = generateSentence(Math.floor(random(0, ruleSets.length)),
            Math.floor(random(3, 6)));
        len = 100 * Math.pow(0.55, 5);
        drawTree(sentence, random(-width / 2, width / 2),
            colors[Math.floor(random(0, colors.length))]);
    }
}

//(...)
```



Mehrere Bäume mit unterschiedlichen Farben & Rulesets

```
//(...)

let ruleSets = [
    [{ predecessor: "F", successor: "F+[+F-F]-[-F+F]F" }],
    [{ predecessor: "F", successor: "FF-[-F+F]+[+F-F]" }],
    [{ predecessor: "F", successor: "F[+F]F[-F][F]" }],
    [{ predecessor: "F", successor: "F[+F]F[-F]F" }],
    [{ predecessor: "F", successor: "F[+F[-F[+F]]]" }]
];

function setup() {
    let canvas = createCanvas(windowWidth * 1.1, windowHeight * 1.1);
    canvas.position(-windowWidth * 0.05, 0);
    angle = radians(12);
    background('#051f20');
    let colors = [
        '#0b2b26',
        '#163832',
        '#235347',
        '#8eb69b',
        '#daf1de'
    ];
    nTrees = 4;
    let initialStrokeWeight = 2.5;

    for (let layer = 0; layer < colors.length; layer++) {
        let strokeWeightValue = initialStrokeWeight - layer * 0.5;
        if (strokeWeightValue < 0.5) strokeWeightValue = 0.5;
        for (let i = 0; i < nTrees; i++) {
            sentence = generateSentence(Math.floor(random(0,
                        ruleSets.length)), 5);
            len = 100 * Math.pow(0.6, 5);
            drawTree(sentence, random(-width / 2, width / 2),
                        colors[layer], strokeWeightValue);
        }
    }
}

//(...)
```

Layers von Bäumen