揺れ比較の全コード

それぞれの index.html, script.js, style.css をメモ帳にコピーして、拡張子を txt から、各々の拡張子に変更して保存し、プログラムごとの一つのフォルダに納めたら、index.html をダブルクリックすることで、オフラインでも動かせます。

揺れるピン

index.html

```
<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>3 連ピン・揺れシミュレーター</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <h1>ピン・スイング・シミュレーター</h1>

        <div class="control-panel">
            <div class="rod-ctrl">
                <span class="label" style="color: #ff4d4d;">PIN A</span>
                <div class="digital-input">
                    <button onclick="changeValue(0, -0.1)">-</button>
                    <input type="number" id="period-0" value="1.0" step="0.1"
min="0.1" max="5.0">
                    <button onclick="changeValue(0, 0.1)">+</button>
                </div>
            </div>
            <div class="rod-ctrl">
                <span class="label" style="color: #00d4ff;">PIN B</span>
                <div class="digital-input">
                    <button onclick="changeValue(1, -0.1)">-</button>
                    <input type="number" id="period-1" value="2.0" step="0.1"
min="0.1" max="5.0">
                    <button onclick="changeValue(1, 0.1)">+</button>
                </div>
            </div>
            <div class="rod-ctrl">
                <span class="label" style="color: #4dff4d;">PIN C</span>
```

```
                <div class="digital-input">
                    <button onclick="changeValue(2, -0.1)">-</button>
                    <input type="number" id="period-2" value="3.0" step="0.1"
min="0.1" max="5.0">
                    <button onclick="changeValue(2, 0.1)">+</button>
                </div>
            </div>
        </div>

        <div class="main-buttons">
            <button id="startBtn">ALL START / STOP</button>
            <button id="resetBtn">RESET</button>
        </div>

        <canvas id="canvas"></canvas>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

script.js

```
const canvas = document.getElementById('canvas');
const ctx = canvas.getContext('2d');
const startBtn = document.getElementById('startBtn');
const resetBtn = document.getElementById('resetBtn');

let isRunning = false;
let startTime = 0;
let animationId;

const rods = [
    { id: 'period-0', factor: 0, color: '#ff4d4d' },
    { id: 'period-1', factor: 0, color: '#00d4ff' },
    { id: 'period-2', factor: 0, color: '#4dff4d' }
];

function resizeCanvas() {
    canvas.width = canvas.clientWidth;
    canvas.height = canvas.clientHeight;
    draw(0);
}
```

```
window.addEventListener('resize', resizeCanvas);
setTimeout(resizeCanvas, 10);

function changeValue(index, delta) {
    const input = document.getElementById(`period-${index}`);
    let val = parseFloat(input.value) + delta;
    input.value = Math.max(0.1, Math.min(5.0, val)).toFixed(1);
}

startBtn.addEventListener('click', () => {
    isRunning = !isRunning;
    if (isRunning) {
        startBtn.textContent = 'STOP';
        startBtn.classList.add('active');
        startTime = performance.now();
        animate();
    } else {
        startBtn.textContent = 'START';
        startBtn.classList.remove('active');
    }
});

resetBtn.addEventListener('click', () => {
    isRunning = false;
    rods.forEach(r => r.factor = 0);
    startBtn.textContent = 'START';
    startBtn.classList.remove('active');
    cancelAnimationFrame(animationId);
    draw(0);
});

function draw(elapsedTime) {
    ctx.clearRect(0, 0, canvas.width, canvas.height);

    const bottomY = canvas.height - 60;
    const pinLength = canvas.height * 0.6;

    //【ここを修正】振幅を半分(15 度)に変更
    const maxAngle = Math.PI / 12;
```

```
    rods.forEach((rod, index) => {
        const period = parseFloat(document.getElementById(rod.id).value) || 1.0;

        if (isRunning) {
            rod.factor += (1 - rod.factor) * 0.05;
        } else {
            rod.factor += (0 - rod.factor) * 0.05;
        }

        const centerX = (canvas.width / 4) * (index + 1);
        const angle = maxAngle * rod.factor * Math.sin((2 * Math.PI / period) *
(elapsedTime / 1000));

        const topX = centerX + pinLength * Math.sin(angle);
        const topY = bottomY - pinLength * Math.cos(angle);

        // 1. 針(棒)
        ctx.strokeStyle = '#eee';
        ctx.lineWidth = 4;
        ctx.beginPath();
        ctx.moveTo(centerX, bottomY);
        ctx.lineTo(topX, topY);
        ctx.stroke();

        // 2. 先端の丸(重り)
        ctx.fillStyle = rod.color;
        ctx.beginPath();
        ctx.arc(topX, topY, 15, 0, Math.PI * 2);
        ctx.fill();

        ctx.fillStyle = 'rgba(255,255,255,0.3)';
        ctx.beginPath();
        ctx.arc(topX - 4, topY - 4, 5, 0, Math.PI * 2);
        ctx.fill();

        // 3. 根元の支点
        ctx.fillStyle = '#666';
        ctx.beginPath();
        ctx.arc(centerX, bottomY, 6, 0, Math.PI * 2);
        ctx.fill();
```

```
        // 4. 土台
        ctx.fillStyle = '#444';
        ctx.fillRect(centerX - 20, bottomY, 40, 10);
    });

    // 地面
    ctx.strokeStyle = '#333';
    ctx.lineWidth = 2;
    ctx.beginPath();
    ctx.moveTo(0, bottomY + 10);
    ctx.lineTo(canvas.width, bottomY + 10);
    ctx.stroke();

    if (!isRunning && rods.every(r => r.factor < 0.001)) {
        cancelAnimationFrame(animationId);
    }
}

function animate() {
    const elapsedTime = performance.now() - startTime;
    draw(elapsedTime);
    animationId = requestAnimationFrame(animate);
}
```

style.css

```
body {
    margin: 0;
    background-color: #1a1a1a;
    color: white;
    font-family: 'Segoe UI', sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    overflow: hidden;
}

.container {
    background: #2a2a2a;
    padding: 25px;
    border-radius: 20px;
```

```
    width: 90vw;
    max-width: 600px;
    text-align: center;
    box-shadow: 0 15px 35px rgba(0,0,0,0.7);
}

.control-panel {
    display: flex;
    justify-content: space-between;
    gap: 10px;
    margin-bottom: 25px;
}

.rod-ctrl {
    background: #333;
    padding: 12px;
    border-radius: 12px;
    flex: 1;
}

.label {
    display: block;
    margin-bottom: 10px;
    font-weight: bold;
    font-size: 0.9rem;
}

.digital-input {
    display: flex;
    align-items: center;
    justify-content: center;
    gap: 4px;
}

.digital-input input {
    width: 45px;
    background: #000;
    color: #0f0;
    border: 1px solid #555;
    padding: 6px 2px;
    text-align: center;
```

```
    font-family: monospace;
    font-size: 1rem;
    border-radius: 4px;
}

.digital-input button {
    width: 28px;
    height: 28px;
    background: #555;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 1.2rem;
}

.main-buttons {
    margin-bottom: 20px;
}

#startBtn {
    background: #27ae60;
    color: white;
    width: 180px;
    padding: 12px;
    border-radius: 30px;
    font-weight: bold;
    border: none;
    cursor: pointer;
}

#startBtn.active { background: #e74c3c; }

#resetBtn {
    background: #555;
    color: white;
    padding: 12px 20px;
    border-radius: 30px;
    border: none;
    cursor: pointer;
    margin-left: 10px;
```

```
}

canvas {
    background: #111;
    border-radius: 15px;
    width: 100%;
    height: 380px;
    border: 1px solid #444;
}
```

揺れる棒

index.html

```
<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>3 連・揺れシミュレーター</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <h1>マルチ・スイング・シミュレーター</h1>

        <div class="control-panel">
            <div class="rod-ctrl" id="ctrl-0">
                <span class="label">棒 A</span>
                <div class="digital-input">
                    <button onclick="changeValue(0, -0.1)">-</button>
                    <input type="number" id="period-0" value="1.0" step="0.1"
min="0.1" max="5.0">
                    <button onclick="changeValue(0, 0.1)">+</button>
                </div>
            </div>
            <div class="rod-ctrl" id="ctrl-1">
                <span class="label">棒 B</span>
                <div class="digital-input">
                    <button onclick="changeValue(1, -0.1)">-</button>
                    <input type="number" id="period-1" value="2.0" step="0.1"
min="0.1" max="5.0">
```

```
                    <button onclick="changeValue(1, 0.1)">+</button>
                </div>
            </div>
            <div class="rod-ctrl" id="ctrl-2">
                <span class="label">棒 C</span>
                <div class="digital-input">
                    <button onclick="changeValue(0.1, -0.1)">-</button>
                    <input type="number" id="period-2" value="3.0" step="0.1"
min="0.1" max="5.0">
                    <button onclick="changeValue(2, 0.1)">+</button>
                </div>
            </div>
        </div>

        <div class="main-buttons">
            <button id="startBtn">ALL START / STOP</button>
            <button id="resetBtn">RESET</button>
        </div>

        <canvas id="canvas"></canvas>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

script.js

```
const canvas = document.getElementById('canvas');
const ctx = canvas.getContext('2d');
const startBtn = document.getElementById('startBtn');
const resetBtn = document.getElementById('resetBtn');

let isRunning = false;
let startTime = 0;
let animationId;

// 3 本の棒の状態を管理
const rods = [
    { id: 'period-0', factor: 0, color: '#ff4d4d' }, // 赤
    { id: 'period-1', factor: 0, color: '#00d4ff' }, // 青
    { id: 'period-2', factor: 0, color: '#4dff4d' }  // 緑
];
```

```
function resizeCanvas() {
    canvas.width = canvas.clientWidth;
    canvas.height = canvas.clientHeight;
    draw(0);
}

window.addEventListener('resize', resizeCanvas);
setTimeout(resizeCanvas, 10);

// デジタル入力の+/−ボタン用
function changeValue(index, delta) {
    const input = document.getElementById(`period-${index}`);
    let val = parseFloat(input.value) + delta;
    input.value = Math.max(0.1, Math.min(5.0, val)).toFixed(1);
}

startBtn.addEventListener('click', () => {
    isRunning = !isRunning;
    if (isRunning) {
        startBtn.textContent = 'STOP';
        startBtn.classList.add('active');
        startTime = performance.now();
        animate();
    } else {
        startBtn.textContent = 'START';
        startBtn.classList.remove('active');
    }
});

resetBtn.addEventListener('click', () => {
    isRunning = false;
    rods.forEach(r => r.factor = 0);
    startBtn.textContent = 'START';
    startBtn.classList.remove('active');
    cancelAnimationFrame(animationId);
    draw(0);
});

function draw(elapsedTime) {
    ctx.clearRect(0, 0, canvas.width, canvas.height);
```

```
    const bottomY = canvas.height - 40;
    const maxAmplitude = canvas.width * 0.07;

    rods.forEach((rod, index) => {
        // 各棒の周期を取得
        const period = parseFloat(document.getElementById(rod.id).value) || 1.0;

        // 振幅のフェードイン/アウト
        if (isRunning) {
            rod.factor += (1 - rod.factor) * 0.05;
        } else {
            rod.factor += (0 - rod.factor) * 0.05;
        }

        const currentAmplitude = maxAmplitude * rod.factor;
        const centerX = (canvas.width / 4) * (index + 1);
        const offsetX = currentAmplitude * Math.sin((2 * Math.PI / period) *
(elapsedTime / 1000));
        const baseX = centerX + offsetX;

        // 棒の描画
        const segments = 15;
        const totalHeight = canvas.height * 0.7;
        const segmentLength = totalHeight / segments;

        ctx.strokeStyle = rod.color;
        ctx.lineWidth = 6;
        ctx.lineCap = 'round';
        ctx.beginPath();
        ctx.moveTo(baseX, bottomY);

        for (let i = 1; i <= segments; i++) {
            const delay = i * 0.03 * (1 / period);
            const segX = centerX + currentAmplitude * Math.sin((2 * Math.PI /
period) * (elapsedTime / 1000 - delay));
            const targetY = bottomY - (i * segmentLength);
            ctx.lineTo(segX, targetY);
        }
        ctx.stroke();
```

```
        // 土台
        ctx.fillStyle = '#666';
        ctx.fillRect(baseX - 15, bottomY - 5, 30, 10);
    });

    // 地面
    ctx.strokeStyle = '#444';
    ctx.lineWidth = 2;
    ctx.beginPath();
    ctx.moveTo(0, bottomY + 5);
    ctx.lineTo(canvas.width, bottomY + 5);
    ctx.stroke();

    if (!isRunning && rods.every(r => r.factor < 0.001)) {
        cancelAnimationFrame(animationId);
    }
}

function animate() {
    const elapsedTime = performance.now() - startTime;
    draw(elapsedTime);
    animationId = requestAnimationFrame(animate);
}
```

style.css

```
body {
    margin: 0;
    background-color: #1a1a1a;
    color: white;
    font-family: sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
}

.container {
    background: #2a2a2a;
    padding: 20px;
    border-radius: 15px;
    width: 95vw;
```

```
    max-width: 600px;
    text-align: center;
}

.control-panel {
    display: flex;
    justify-content: space-around;
    gap: 10px;
    margin-bottom: 20px;
}

.rod-ctrl {
    background: #333;
    padding: 10px;
    border-radius: 8px;
    flex: 1;
}

.label {
    display: block;
    margin-bottom: 8px;
    font-weight: bold;
    color: #00d4ff;
}

.digital-input {
    display: flex;
    align-items: center;
    justify-content: center;
    gap: 5px;
}

.digital-input input {
    width: 50px;
    background: #000;
    color: #0f0;
    border: 1px solid #444;
    padding: 5px;
    text-align: center;
    font-family: 'Courier New', Courier, monospace;
    font-size: 1rem;
```

```
}

.digital-input button {
    width: 25px;
    height: 25px;
    padding: 0;
    line-height: 1;
}

.main-buttons {
    margin-bottom: 15px;
}

#startBtn { background: #27ae60; color: white; width: 200px; }
#startBtn.active { background: #e74c3c; }

canvas {
    background: #000;
    border-radius: 10px;
    width: 100%;
    height: 350px;
}
```