

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Gurdjieff Multiplication Animation</title>
  <style>
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

    body {
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
      background: linear-gradient(135deg, #0f1419 0%, #1a2332 100%);
      color: #e0e0e0;
      padding: 15px;
      min-height: 100vh;
    }

    .container {
      max-width: 1200px;
      margin: 0 auto;
    }

    header {
      text-align: center;
      margin-bottom: 25px;
    }

    h1 {
      font-size: clamp(1.8em, 5vw, 2.8em);
      margin-bottom: 8px;
      background: linear-gradient(135deg, #00d4ff, #0099ff);
      -webkit-background-clip: text;
      -webkit-text-fill-color: transparent;
      background-clip: text;
      letter-spacing: 2px;
    }

    .subtitle {
      font-size: clamp(0.85em, 3vw, 1.1em);
      color: #00d4ff;
      font-weight: 300;
      letter-spacing: 1px;
      margin-bottom: 10px;
    }
  </style>

```

```
}

.description {
  color: #a0a0a0;
  max-width: 700px;
  margin: 0 auto;
  line-height: 1.6;
  font-size: clamp(0.8em, 2.5vw, 0.95em);
}

.info-section {
  background: rgba(0, 212, 255, 0.08);
  border: 1px solid rgba(0, 212, 255, 0.3);
  border-radius: 10px;
  padding: clamp(15px, 3vw, 25px);
  margin-bottom: 20px;
}

.cycle-display {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(140px, 1fr));
  gap: 15px;
  margin-bottom: 15px;
}

.cycle-item {
  display: flex;
  flex-direction: column;
  gap: 6px;
}

.cycle-label {
  font-size: clamp(0.7em, 2vw, 0.85em);
  color: #00d4ff;
  text-transform: uppercase;
  letter-spacing: 1px;
  font-weight: 600;
}

.cycle-value {
  font-size: clamp(1.3em, 4vw, 1.8em);
  font-weight: bold;
  color: #fff;
}

.executant-display {
  background: rgba(233, 30, 99, 0.15);
```

```
border: 2px solid rgba(233, 30, 99, 0.3);
border-radius: 8px;
padding: 12px;
text-align: center;
}

.executant-display .cycle-label {
  color: #E91E63;
}

.executant-display .cycle-value {
  font-size: clamp(1.5em, 4vw, 2.2em);
}

.progress-dots {
  display: flex;
  gap: 8px;
  justify-content: center;
  margin: 15px 0;
  flex-wrap: wrap;
}

.dot {
  width: 12px;
  height: 12px;
  border-radius: 50%;
  background: rgba(0, 212, 255, 0.3);
  transition: all 0.3s ease;
  cursor: pointer;
}

.dot.active {
  background: #00d4ff;
  box-shadow: 0 0 12px rgba(0, 212, 255, 0.8);
  transform: scale(1.2);
}

.controls {
  display: flex;
  gap: 10px;
  justify-content: center;
  flex-wrap: wrap;
  margin-top: 15px;
}

button {
  padding: clamp(8px, 2vw, 12px) clamp(16px, 3vw, 28px);
```

```
background: linear-gradient(135deg, #00d4ff, #0099ff);
color: #000;
border: none;
border-radius: 6px;
font-weight: 600;
cursor: pointer;
font-size: clamp(0.75em, 2vw, 1em);
transition: all 0.3s ease;
box-shadow: 0 4px 15px rgba(0, 212, 255, 0.3);
white-space: nowrap;
}

button:hover {
    transform: translateY(-2px);
    box-shadow: 0 6px 20px rgba(0, 212, 255, 0.5);
}

button:active {
    transform: translateY(0);
}

button.clear-btn {
    background: linear-gradient(135deg, #E91E63, #c2185b);
    color: #fff;
    box-shadow: 0 4px 15px rgba(233, 30, 99, 0.3);
}

button.clear-btn:hover {
    box-shadow: 0 6px 20px rgba(233, 30, 99, 0.5);
}

.speed-control {
    display: flex;
    align-items: center;
    gap: 8px;
    background: rgba(0, 212, 255, 0.1);
    padding: clamp(8px, 1.5vw, 10px) clamp(10px, 2vw, 15px);
    border-radius: 6px;
    border: 1px solid rgba(0, 212, 255, 0.2);
}

.speed-control label {
    font-size: clamp(0.7em, 2vw, 0.9em);
    color: #00d4ff;
    font-weight: 600;
    white-space: nowrap;
}
```

```
input[type="range"] {  
    width: clamp(100px, 20vw, 150px);  
    cursor: pointer;  
}  
  
.speed-value {  
    font-weight: bold;  
    color: #fff;  
    min-width: 35px;  
    text-align: right;  
    font-size: clamp(0.75em, 2vw, 0.9em);  
}  
  
.canvas-wrapper {  
    position: relative;  
    width: 100%;  
    background: rgba(15, 52, 96, 0.3);  
    border: 2px solid rgba(0, 212, 255, 0.2);  
    border-radius: 12px;  
    padding: clamp(15px, 3vw, 30px);  
    margin-bottom: 20px;  
    overflow: visible;  
}  
  
canvas {  
    display: block;  
    width: 100%;  
    height: auto;  
}  
  
#gridContainer {  
    width: 100%;  
}  
  
.grid-display {  
    display: grid;  
    grid-template-columns: repeat(6, 1fr);  
    gap: clamp(10px, 2vw, 20px);  
    margin-bottom: 20px;  
}  
  
.position-circle {  
    aspect-ratio: 1;  
    display: flex;  
    align-items: center;  
    justify-content: center;
```

```
border-radius: 50%;  
font-size: clamp(1.2em, 3vw, 1.8em);  
font-weight: bold;  
border: 3px solid rgba(255, 255, 255, 0.2);  
cursor: pointer;  
transition: all 0.3s ease;  
box-shadow: 0 0 20px rgba(0, 0, 0, 0.5);  
}  
  
.position-circle:hover {  
    transform: scale(1.1);  
    border-color: rgba(255, 255, 255, 0.6);  
    box-shadow: 0 0 30px rgba(255, 255, 255, 0.3);  
}  
  
.position-circle.selected {  
    border-width: 5px;  
    box-shadow: 0 0 40px rgba(255, 255, 255, 0.8);  
    transform: scale(1.15);  
}  
  
.num-1 { background: linear-gradient(135deg, #FFD700, rgba(255, 215, 0, 0.7)); color: #000; }  
.num-2 { background: linear-gradient(135deg, #2ECC71, rgba(46, 204, 113, 0.7)); color: #000; }  
.num-4 { background: linear-gradient(135deg, #E74C3C, rgba(231, 76, 60, 0.7)); color: #fff; }  
.num-5 { background: linear-gradient(135deg, #3498DB, rgba(52, 152, 219, 0.7)); color: #fff; }  
.num-7 { background: linear-gradient(135deg, #E91E63, rgba(233, 30, 99, 0.7)); color: #fff; }  
.num-8 { background: linear-gradient(135deg, #4B3882, rgba(75, 56, 130, 0.7)); color: #fff; }  
  
.row-label {  
    font-size: clamp(0.7em, 2vw, 0.85em);  
    color: #00d4ff;  
    font-weight: 600;  
    margin-bottom: 10px;  
    letter-spacing: 1px;  
}  
  
.row-container {  
    margin-bottom: 25px;  
}  
  
.row-container.inactive {
```

```
    opacity: 0.35;
    transform: scale(0.98);
}

.row-container.active {
    opacity: 1;
    transform: scale(1);
}

.legend {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(120px, 1fr));
    gap: 12px;
    background: rgba(0, 212, 255, 0.08);
    border: 1px solid rgba(0, 212, 255, 0.2);
    border-radius: 10px;
    padding: clamp(12px, 2vw, 20px);
}

.legend-item {
    display: flex;
    align-items: center;
    gap: 8px;
    font-size: clamp(0.75em, 2vw, 0.9em);
    cursor: pointer;
    transition: all 0.3s ease;
    padding: 8px;
    border-radius: 6px;
}

.legend-item:hover {
    background: rgba(0, 212, 255, 0.1);
}

.legend-dot {
    width: 18px;
    height: 18px;
    border-radius: 50%;
    border: 2px solid rgba(255, 255, 255, 0.2);
    flex-shrink: 0;
}

.note {
    background: rgba(233, 30, 99, 0.1);
    border-left: 4px solid #E91E63;
    padding: clamp(12px, 2vw, 15px);
    border-radius: 4px;
}
```

```
        font-size: clamp(0.75em, 2vw, 0.85em);
        color: #ddd;
        line-height: 1.6;
        margin-top: 15px;
    }

    .note strong {
        color: #E91E63;
    }

    @media (max-width: 480px) {
        .grid-display {
            grid-template-columns: repeat(3, 1fr);
        }

        .cycle-display {
            grid-template-columns: 1fr;
        }

        .controls {
            gap: 8px;
        }

        button {
            flex: 1 1 auto;
            min-width: 70px;
        }
    }

    @media (max-width: 768px) {
        .cycle-display {
            grid-template-columns: 1fr 1fr;
        }
    }
}

</style>
</head>
<body>
    <div class="container">
        <header>
            <div class="subtitle">CUARTO CAMINO · GURDJIEFF</div>
            <h1>MULTIPLICACIÓN</h1>
            <p class="description">
                Visualización interactiva del ejercicio sagrado de rotaciones
                permutativas.
                Haz clic en cualquier círculo para seguir los movimientos de ese
                ejecutante.
            </p>
        </header>
    </div>

```

```
</header>

<div class="info-section">
  <div class="cycle-display">
    <div class="cycle-item">
      <div class="cycle-label">Ciclo Actual</div>
      <div class="cycle-value"><span id="cycleNum">1</span> / 6</div>
    </div>
    <div class="cycle-item">
      <div class="cycle-label">Estado</div>
      <div class="cycle-value" id="status">Pausado</div>
    </div>
    <div class="cycle-item executant-display">
      <div class="cycle-label">Ejecutante</div>
      <div class="cycle-value" id="selectedExec">—</div>
    </div>
  </div>

  <div class="progress-dots">
    <div class="dot active" onclick="jumpToCycle(0)"></div>
    <div class="dot" onclick="jumpToCycle(1)"></div>
    <div class="dot" onclick="jumpToCycle(2)"></div>
    <div class="dot" onclick="jumpToCycle(3)"></div>
    <div class="dot" onclick="jumpToCycle(4)"></div>
    <div class="dot" onclick="jumpToCycle(5)"></div>
  </div>

  <div class="controls">
    <button onclick="togglePlay()">▶</button>
    <button onclick="pauseAnim()">⏸</button>
    <button onclick="resetAnim()">⟲</button>
    <button class="clear-btn" onclick="clearSelection()">✖</button>
    <div class="speed-control">
      <label for="speedSlider">Vel:</label>
      <input type="range" id="speedSlider" min="0.5" max="3" step="0.1" value="1.5" onchange="updateSpeed()">
        <span class="speed-value" id="speedValue">1.5x</span>
      </div>
    </div>
  </div>

  <div class="note">
    <strong>Interactivo:</strong> Haz clic en un número para rastrear su movimiento. Usa flechas del teclado para navegar o ESPACIO para Play/Pause.
  </div>
</div>
```

```
<div class="canvas-wrapper">
  <canvas id="flowCanvas"></canvas>
  <div id="gridContainer"></div>
</div>

<div class="legend">
  <div class="legend-item" onclick="selectExecutant(1)">
    <div class="legend-dot num-1"></div>
    <span>Ejecutante 1</span>
  </div>
  <div class="legend-item" onclick="selectExecutant(2)">
    <div class="legend-dot num-2"></div>
    <span>Ejecutante 2</span>
  </div>
  <div class="legend-item" onclick="selectExecutant(4)">
    <div class="legend-dot num-4"></div>
    <span>Ejecutante 4</span>
  </div>
  <div class="legend-item" onclick="selectExecutant(5)">
    <div class="legend-dot num-5"></div>
    <span>Ejecutante 5</span>
  </div>
  <div class="legend-item" onclick="selectExecutant(7)">
    <div class="legend-dot num-7"></div>
    <span>Ejecutante 7</span>
  </div>
  <div class="legend-item" onclick="selectExecutant(8)">
    <div class="legend-dot num-8"></div>
    <span>Ejecutante 8</span>
  </div>
</div>
</div>

<script>
  const data = [
    [1, 4, 2, 8, 5, 7],
    [2, 8, 5, 7, 1, 4],
    [4, 2, 8, 5, 7, 1],
    [5, 7, 1, 4, 2, 8],
    [7, 1, 4, 2, 8, 5],
    [8, 5, 7, 1, 4, 2]
  ];
  const colors = {
    1: '#FFD700',
    2: '#2ECC71',
  }
</script>
```

```
        4: '#E74C3C',
        5: '#3498DB',
        7: '#E91E63',
        8: '#4B3882'
    };

let currentCycle = 0;
let isPlaying = false;
let speed = 1.5;
let animTimeout = null;
let selectedExecutant = null;

function initGrid() {
    const container = document.getElementById('gridContainer');
    container.innerHTML = '';

    data.forEach((row, rowIdx) => {
        const rowDiv = document.createElement('div');
        rowDiv.className = `row-container ${rowIdx === 0 ? 'active' : 'inactive'}`;
        rowDiv.id = `row-${rowIdx}`;

        const label = document.createElement('div');
        label.className = 'row-label';
        label.textContent = `FILA ${rowIdx + 1}`;
        rowDiv.appendChild(label);

        const grid = document.createElement('div');
        grid.className = 'grid-display';

        row.forEach((num) => {
            const circle = document.createElement('div');
            circle.className = `position-circle num-${num}`;
            circle.textContent = num;
            circle.onclick = () => selectExecutant(num);
            grid.appendChild(circle);
        });

        rowDiv.appendChild(grid);
        container.appendChild(rowDiv);
    });
}

drawFlowLines();
}

function drawFlowLines() {
    const canvas = document.getElementById('flowCanvas');
```

```
const container = document.getElementById('gridContainer');
const wrapper = canvas.parentElement;

canvas.width = wrapper.offsetWidth;
canvas.height = container.offsetHeight;
canvas.style.position = 'absolute';
canvas.style.top = '0';
canvas.style.left = '0';
canvas.style.pointerEvents = 'none';

const ctx = canvas.getContext('2d');
ctx.clearRect(0, 0, canvas.width, canvas.height);

const circles = document.querySelectorAll('.position-circle');
const positions = [];

circles.forEach((circle) => {
    const rect = circle.getBoundingClientRect();
    const wrapperRect = wrapper.getBoundingClientRect();
    positions.push({
        x: rect.left - wrapperRect.left + rect.width / 2,
        y: rect.top - wrapperRect.top + rect.height / 2
    });
});

if (selectedExecutant !== null) {
    const num = selectedExecutant;
    ctx.strokeStyle = colors[num];
    ctx.lineWidth = 3.5;
    ctx.setLineDash([8, 4]);
    ctx.lineCap = 'round';
    ctx.lineJoin = 'round';
    ctx.globalAlpha = 0.9;

    let started = false;
    for (let cycleIdx = 0; cycleIdx <= currentCycle; cycleIdx++) {
        const row = data[cycleIdx];
        const posInRow = row.indexOf(num);
        const globalIdx = cycleIdx * 6 + posInRow;

        if (positions[globalIdx]) {
            if (!started) {
                ctx.beginPath();
                ctx.moveTo(positions[globalIdx].x, positions[globalIdx].y);
                started = true;
            } else {
                ctx.lineTo(positions[globalIdx].x, positions[globalIdx].y);
            }
        }
    }
}
```

```

        }
    }
}

if (started) ctx.stroke();
} else {
    [1, 2, 4, 5, 7, 8].forEach(num => {
        ctx.strokeStyle = colors[num];
        ctx.lineWidth = 2.5;
        ctx.setLineDash([8, 4]);
        ctx.lineCap = 'round';
        ctx.lineJoin = 'round';
        ctx.globalAlpha = 0.7;

        let started = false;
        for (let cycleIdx = 0; cycleIdx <= currentCycle; cycleIdx++) {
            const row = data[cycleIdx];
            const posInRow = row.indexOf(num);
            const globalIdx = cycleIdx * 6 + posInRow;

            if (positions[globalIdx]) {
                if (!started) {
                    ctx.beginPath();
                    ctx.moveTo(positions[globalIdx].x, positions[globalIdx].y);
                    started = true;
                } else {
                    ctx.lineTo(positions[globalIdx].x, positions[globalIdx].y);
                }
            }
            if (started) ctx.stroke();
        });
    });
}

ctx.globalAlpha = 1;
ctx.setLineDash([]);
}

function selectExecutant(num) {
    selectedExecutant = selectedExecutant === num ? null : num;
    pauseAnim();
    currentCycle = 0;
    updateDisplay();
    drawFlowLines();
}

function clearSelection() {
    selectedExecutant = null;
}

```

```
updateDisplay();
drawFlowLines();
}

function togglePlay() {
    isPlaying = !isPlaying;
    if (isPlaying) animate();
    updateDisplay();
}

function pauseAnim() {
    isPlaying = false;
    clearTimeout(animTimeout);
    updateDisplay();
}

function resetAnim() {
    currentCycle = 0;
    isPlaying = false;
    clearTimeout(animTimeout);
    updateDisplay();
    drawFlowLines();
}

function jumpToCycle(idx) {
    currentCycle = idx;
    pauseAnim();
    updateDisplay();
    drawFlowLines();
}

function updateSpeed() {
    speed = parseFloat(document.getElementById('speedSlider').value);
    document.getElementById('speedValue').textContent = speed.toFixed(1)
+ 'x';
}

function animate() {
    if (!isPlaying) return;
    const delay = (2000 / speed);
    animTimeout = setTimeout(() => {
        currentCycle = (currentCycle + 1) % 6;
        updateDisplay();
        drawFlowLines();
        animate();
    }, delay);
}
```

```

function updateDisplay() {
    document.getElementById('cycleNum').textContent = currentCycle + 1;
    document.getElementById('status').textContent = isPlaying ?
'Animando' : 'Pausado';
    document.getElementById('status').style.color = isPlaying ? '#00d4ff' :
'#E91E63';
    document.getElementById('selectedExec').textContent =
selectedExecutant !== null ? selectedExecutant : '—';
    document.getElementById('selectedExec').style.color =
selectedExecutant !== null ? colors[selectedExecutant] : '#ffff';

    document.querySelectorAll('.dot').forEach((dot, idx) => {
        dot.classList.toggle('active', idx === currentCycle);
    });

    document.querySelectorAll('.row-container').forEach((row, idx) => {
        row.classList.toggle('active', idx === currentCycle);
        row.classList.toggle('inactive', idx !== currentCycle);
    });

    document.querySelectorAll('.position-circle').forEach(circle => {
        const circleNum = parseInt(circle.textContent);
        if (selectedExecutant !== null && circleNum === selectedExecutant) {
            circle.classList.add('selected');
        } else {
            circle.classList.remove('selected');
        }
    });
}

initGrid();
updateDisplay();

document.addEventListener('keydown', (e) => {
    if (e.code === 'Space') {
        e.preventDefault();
        togglePlay();
    }
    if (e.code === 'ArrowRight') {
        jumpToCycle((currentCycle + 1) % 6);
    }
    if (e.code === 'ArrowLeft') {
        jumpToCycle((currentCycle - 1 + 6) % 6);
    }
});

```

```
window.addEventListener('resize', () => {
    drawFlowLines();
});
</script>
</body>
</html>
```