Spinning 3D Hexagon Illusion Source Code
Original Version; April 2021.
Pummelator

More information available at pummelator.com/jump/hexspin

Below is the HTML that makes the canvas element to be used

```
<canvas id="illusion" width="256" height="256"
    style = "border: 4px solid #0000FF;">
    Your browser does not support canvas.
</canvas>
```

Below is the JavaScript portion of the code

```
<!-- == Animated Image Canvas Elements == -->
  <script>
     var canvas = document.getElementById("illusion");
     var context = canvas.getContext("2d");
     var rotationStep = 0;

     context.fillStyle = "#000088";
     context.fillRect(0, 112, 32, 32);
     context.fillRect(224, 112, 32, 32);

     function startRevolvingIllusion()
     { setInterval(revolveIllusion, 30); }

     function revolveIllusion()
     {
        // Clear current lines
        context.clearRect(32, 0, 192, 255);
        // Line drawing settings shared by both sets
        context.strokeStyle = "#0000FF";
        context.lineJoin = "round";
        context.lineCap = "round";
        // Lines that start at the top
        context.beginPath();
        var topY = 48 + (rotationStep * 2) + (rotationStep / 2);
        context.lineWidth = 5 - parseInt((40 - Math.abs(topY - 128)) / 10);
        context.moveTo(32, 128);
        context.lineTo(72, topY);
        context.lineTo(184, topY);
        context.lineTo(224, 128);
        context.stroke();
        // Lines that start at the bottom
        context.beginPath();
        var bottomY = 208 - (rotationStep * 2) - (rotationStep / 2);
        context.lineWidth = 12 + parseInt((40 - Math.abs(topY - 128)) / 10);
        context.moveTo(32, 128);
        context.lineTo(72, bottomY);
        context.lineTo(184, bottomY);
        context.lineTo(224, 128);
        context.stroke();

        rotationStep = (rotationStep + 1) & 63;
        /* == Explanation of the formula for determining the Y positions:
           Each pair of lines starts at a Y coordinate of either 48
           or 208 and ends just shy of the opposite coordinate. There are
           64 "frames" in the animation and the distance between the two Y
           coordinates is 208 - 48 = 160. The number of pixels the lines need
```

```
47            to move in order to get from point A to point B at a constant rate
48            is 160 / 64 = 2.5. That's not a whole number, but alternating between
49            moving two pixels and three pixels each step will effectively create
50            the same rate of movement and will be virtually unnoticeable if
51            played at a fast enough rate.
52
53            == Explanation of the formula for determining line widths:
54            The idea with this was to make the bottom line get larger and the
55            top line get smaller as each gets closer to the center of the canvas,
56            then reverse the effects as they move away. This helps to reinforce
57            the illusion that the lines form a rotating 3D object. The first
thing
58            it finds is the absolute value of the difference between the current
59            Y coordinate and the middle of the canvas (128). It then subtracts
60            that value from the maximum absolute distance the Y coordinate can
61            be from the middle of the canvas (40) and divides by 10. This newly
62            calculated value is then used as a factor to adjust a preset line
63            width.
64
65            == Rotation step incrementation:
66            The line that handles it starts by adding one to the current value
67            of the rotation step, then it performs a bitwise AND on that, then
68            assigns the value back to the rotation step variable. When step+1
69            reaches 64, the bitwise will cause the value to become 0,
70            effectively resetting the position (hence, why I chose to have this
71            animation be comprised of 64 steps)
72         */
73      }
74
75        window.addEventListener("load", startRevolvingIllusion, false);
76    </script>
```