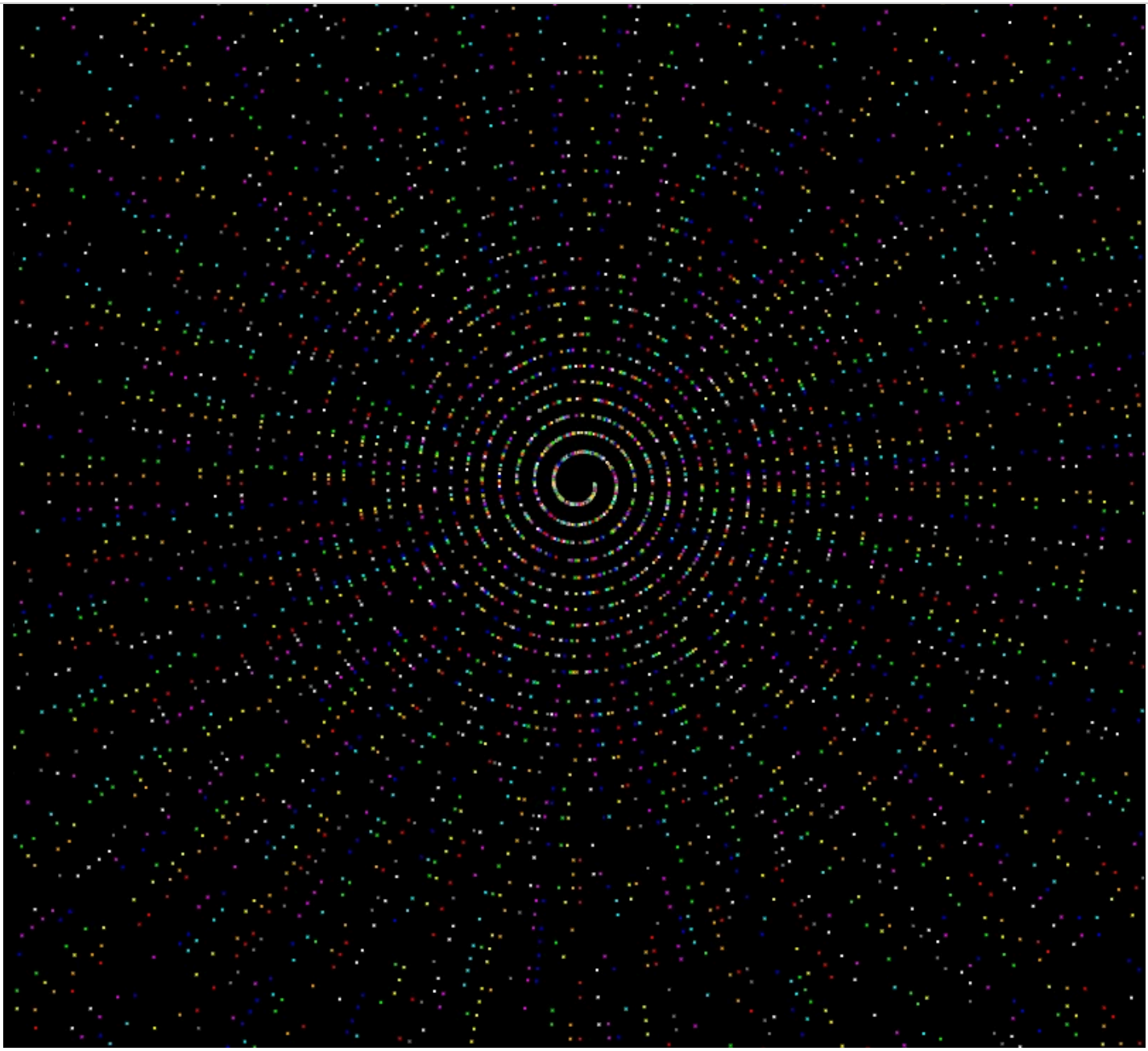
r/ThingsYouDidntKnow • 2 mo. ago
TheStocksGuy

Dirichlet's Theorem (Evolved by +3 Increasement of Interconnected Primes)

This code visualizes prime numbers using Dirichlet's theorem on a black canvas. It plots primes radially, changing colors every 3rd prime, and connecting them incrementally by 3. The result is a vibrant, dynamic display of prime distributions. 300,000 prime numbers here.

Feel free to experiment with this HTML Canvas on your own. Simply save the file as .html, then drag and drop it into your browser or open it directly. It serves as a fluent example for exploring and interacting with the code.



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Dirichlet Theorem Visualization</title>
    <style>
        body {
            background-color: black;
            color: white;
        }
        canvas {
            display: block;
```

```
    </style>
</head>
<body>
    <canvas id="primeCanvas" width="1200" height="1200"></canvas>
    <script>
        const canvas = document.getElementById('primeCanvas');
        const ctx = canvas.getContext('2d');
        const width = canvas.width;
        const height = canvas.height;
        const colors = ['#FF0000', '#FFA500', '#FFFF00', '#00FF00', '#00FFFF', '#0000FF', '#FF00FF
        let zoomLevel = 1;
        let offsetX = 0;
        let offsetY = 0;

        function isPrime(num) {
            if (num <= 1) return false;
            if (num <= 3) return true;
            if (num % 2 === 0 || num % 3 === 0) return false;
            for (let i = 5; i * i <= num; i += 6) {
                if (num % i === 0 || num % (i + 2) === 0) return false;
            }
            return true;
        }

        function drawPrimeChart() {
            const centerX = width / 2;
            const centerY = height / 2;
            let angleIncrement = 2 * Math.PI / 360;
            let radius = 10;

            ctx.clearRect(0, 0, canvas.width, canvas.height);
            ctx.save();
            ctx.translate(offsetX, offsetY);
            ctx.scale(zoomLevel, zoomLevel);

            for (let num = 1; num < 300000; num++) {
                if (isPrime(6 * num + 1)) {
                    let angle = num * angleIncrement;
                    let x = centerX + radius * Math.cos(angle);
                    let y = centerY + radius * Math.sin(angle);
                    ctx.fillStyle = colors[(num % 9)];
                    ctx.fillText(6 * num + 1, x, y);
                    radius += 0.1;
                }
            }
            ctx.restore();
```

```
            const mouseX = event.offsetX;
            const mouseY = event.offsetY;

            if (event.deltaY < 0) {
                zoomLevel *= 1.1;
                offsetX = mouseX - (mouseX - offsetX) * 1.1;
                offsetY = mouseY - (mouseY - offsetY) * 1.1;
            } else {
                zoomLevel /= 1.1;
                offsetX = mouseX - (mouseX - offsetX) / 1.1;
                offsetY = mouseY - (mouseY - offsetY) / 1.1;
            }
            drawPrimeChart();
        });

        drawPrimeChart();
    </script>
</body>
</html>
```

1 5 Share Approved 2 months ago

Post Insights

Only the post author and moderators can see this

224	100%	5	3
Total Views	Upvote Rate	Comments	Total Shares

Hourly views for first 48 hours

Some insights are no longer available because this post is older than 45 days

Add a comment

Sort by: Best Search Comments

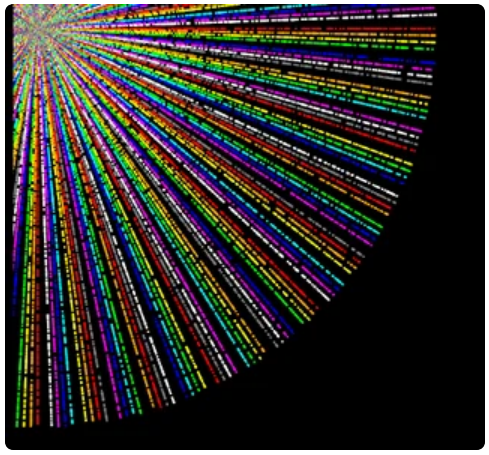
TheStocksGuy OP • 2mo ago • Stickied comment

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Dirichlet Theorem Visualization</title>
    <style>
        body {
            background-color: black;
            color: white;
        }
        canvas {
            display: block;
            margin: auto;
            background-color: black;
        }
    </style>
</head>
<body>
    <canvas id="primeCanvas" width="1200" height="1200"></canvas>
    <script>
        const canvas = document.getElementById('primeCanvas');
        const ctx = canvas.getContext('2d');
        const width = canvas.width;
        const height = canvas.height;
        const colors = ['#FF0000', '#FFA500', '#FFFF00', '#00FF00', '#00FFFF', '#0000FF', '#FF
        let primes = [];
        let zoomLevel = 1;
        let offsetX = 0;
        let offsetY = 0;
        let animationFrameId;
        let isAnimating = true;

        function isPrime(num) {
            if (num <= 1) return false;
            if (num <= 3) return true;
            if (num % 2 === 0 || num % 3 === 0) return false;
            for (let i = 5; i * i <= num; i += 6) {
                if (num % i === 0 || num % (i + 2) === 0) return false;
            }
            return true;
        }

        function generatePrimes() {
            for (let num = 1; num < 300000; num++) {
                if (isPrime(6 * num + 1)) {
                    primes.push(6 * num + 1);
```

```
function drawPrimeChart() {
    const centerX = width / 2;
    const centerY = height / 2;
    let angleIncrement = 2 * Math.PI / 360;
    let radius = 10;

    ctx.clearRect(0, 0, canvas.width, canvas.height);
    ctx.save();
    ctx.translate(offsetX, offsetY);
    ctx.scale(zoomLevel, zoomLevel);
    ctx.translate(centerX, centerY);

    primes.forEach((prime, index) => {
        let angle = index * angleIncrement;
        let x = radius * Math.cos(angle);
        let y = radius * Math.sin(angle);
        ctx.fillStyle = colors[(index % 9)];
        ctx.fillText(prime, x, y);
        if (index > 0) {
            let prevAngle = (index - 1) * angleIncrement;
            let prevX = radius * Math.cos(prevAngle);
            let prevY = radius * Math.sin(prevAngle);
            ctx.strokeStyle = colors[(index % 9)];
            ctx.beginPath();
            ctx.moveTo(prevX, prevY);
            ctx.lineTo(x, y);
            ctx.stroke();
        }
        radius += 0.1;
    });

    ctx.restore();
}

function animate() {
    primes.push(primes.shift()); // Rotate primes
    drawPrimeChart();
    if (isAnimating) {
        animationFrameId = requestAnimationFrame(animate);
    }
}

canvas.addEventListener('wheel', function(event) {
    const mouseX = event.offsetX;
    const mouseY = event.offsetY;
```

```
                offsetX = mouseX - (mouseX - offsetX) * 1.1;
                offsetY = mouseY - (mouseY - offsetY) * 1.1;
            } else {
                zoomLevel /= 1.1;
                offsetX = mouseX - (mouseX - offsetX) / 1.1;
                offsetY = mouseY - (mouseY - offsetY) / 1.1;
            }
            drawPrimeChart();
        });

        canvas.addEventListener('click', function() {
            if (isAnimating) {
                cancelAnimationFrame(animationFrameId);
            } else {
                animate();
            }
            isAnimating = !isAnimating;
        });

        generatePrimes();
        drawPrimeChart();
        animate();
    </script>
</body>
</html>
```

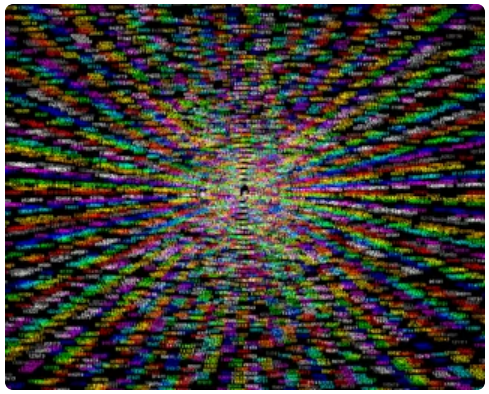
TheStocksGuy OP • 2mo ago



1

Approved 2 months ago

TheStocksGuy OP • 2mo ago



Updated the code so it can be zoomed in and out on.

1

Approved 2 months ago

TheStocksGuy OP • 2mo ago

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Dirichlet Theorem Visualization</title>
    <style>
        body {
            background-color: black;
            color: white;
        }
        canvas {
            display: block;
            margin: auto;
            background-color: black;
        }
    </style>
</head>
<body>
    <canvas id="primeCanvas" width="1200" height="1200"></canvas>
    <script>
        const canvas = document.getElementById('primeCanvas');
        const ctx = canvas.getContext('2d');
        const width = canvas.width;
        const height = canvas.height;
        const colors = ['#FF0000', '#FFA500', '#FFFF00', '#00FF00', '#00FFFF', '#0000FF', '#FF
        let primes = [];
        let zoomLevel = 1;
        let offsetX = 0;
        let offsetY = 0;
        let animationFrameId;
        let isAnimating = true;

        function isPrime(num) {
            if (num <= 1) return false;
            if (num <= 3) return true;
            if (num % 2 === 0 || num % 3 === 0) return false;
            for (let i = 5; i * i <= num; i += 6) {
                if (num % i === 0 || num % (i + 2) === 0) return false;
            }
            return true;
        }

        function generatePrimes() {
            for (let num = 1; num < 300000; num++) {
                if (isPrime(6 * num + 1)) {
```

```
}

function drawPrimeChart() {
    const centerX = width / 2;
    const centerY = height / 2;
    let angleIncrement = 2 * Math.PI / 360;
    let radius = 10;

    ctx.clearRect(0, 0, canvas.width, canvas.height);
    ctx.save();
    ctx.translate(offsetX, offsetY);
    ctx.scale(zoomLevel, zoomLevel);
    ctx.translate(centerX, centerY);

    primes.forEach((prime, index) => {
        let angle = index * angleIncrement;
        let x = radius * Math.cos(angle);
        let y = radius * Math.sin(angle);
        ctx.fillStyle = colors[(index % 9)];
        ctx.fillText(prime, x, y);
        if (index > 0) {
            let prevAngle = (index - 1) * angleIncrement;
            let prevX = radius * Math.cos(prevAngle);
            let prevY = radius * Math.sin(prevAngle);
            ctx.strokeStyle = colors[(index % 9)];
            ctx.beginPath();
            ctx.moveTo(prevX, prevY);
            ctx.lineTo(x, y);
            ctx.stroke();
        }
        radius += 0.1;
    });

    ctx.restore();
}

function animate() {
    primes.push(primes.shift()); // Rotate primes
    drawPrimeChart();
    if (isAnimating) {
        animationFrameId = requestAnimationFrame(animate);
    }
}

canvas.addEventListener('wheel', function(event) {
    const mouseX = event.offsetX;
```

```
            zoomLevel *= 1.1;
            offsetX = mouseX - (mouseX - offsetX) * 1.1;
            offsetY = mouseY - (mouseY - offsetY) * 1.1;
        } else {
            zoomLevel /= 1.1;
            offsetX = mouseX - (mouseX - offsetX) / 1.1;
            offsetY = mouseY - (mouseY - offsetY) / 1.1;
        }
        drawPrimeChart();
    });

    canvas.addEventListener('click', function() {
        if (isAnimating) {
            cancelAnimationFrame(animationFrameId);
        } else {
            animate();
        }
        isAnimating = !isAnimating;
    });

    generatePrimes();
    drawPrimeChart();
    animate();
</script>
</body>
</html>
```

To give you a video I have made an example of how this process works on youtube. [Dirichlet Theorem Visualization with primes]

1 ···

Approved 2 months ago