r/ThingsYouDidntKnow • 2 mo. ago
TheStocksGuy

(Short) Thesis on Prime Number Visualization

Thesis on Prime Number Visualization

With this visualization, I aim to represent prime numbers in a compelling geometric pattern that not only reveals the beauty of mathematics but also highlights certain challenges and imperfections in the visual presentation.

The core concept involves mapping prime numbers in a spiral formation using polar coordinates. This method ensures each prime number has a unique position defined by an ever-increasing radius and a steadily progressing angle. The result is a mesmerizing spiral that visually captures the distribution of primes.

However, while this visualization method feels more appropriate for highlighting the inherent beauty of prime numbers, it encounters several issues:

1. **Zooming Artifacts**: When zooming out, the visual clarity diminishes, making it difficult to distinguish individual primes and their connections. This is due to the limitations in how the graphical elements are scaled, which can result in a cluttered and less insightful view of the overall pattern.
2. **Line Defects**: The lines connecting the primes, especially those extending from the center, can appear visually jarring. These defects occur because of the line thickness and scaling inconsistencies, which can cause certain lines to dominate the visual field, detracting from the overall aesthetic and structural coherence.

Despite these challenges, the visualization serves as a testament to the elegance of prime numbers and their distribution. It underscores both the potential and the limitations of using graphical representations to convey complex mathematical concepts. Future improvements might focus on refining the scaling algorithms and enhancing the visual distinction between individual elements to mitigate these issues.

Try it yourself.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Poincaré Conjecture Visualization</title>
    <style>
        body {
            margin: 0;
            overflow: hidden;
            background-color: black;
        }
        canvas {
            display: block;
        }
    </style>
```

```
<script>
    const canvas = document.getElementById('canvas');
    const ctx = canvas.getContext('2d');
    canvas.width = window.innerWidth;
    canvas.height = window.innerHeight;

    const colors = ['#FF0000', '#FFA500', '#FFFF00', '#00FF00', '#00FFFF', '#0000FF', '#FF00FF
    const primes = [];
    let zoomLevel = 1;
    let offsetX = 0;
    let offsetY = 0;
    let numObjects = 30000;

    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() {
        let num = 2;
        while (primes.length < numObjects) {
            if (isPrime(num)) {
                primes.push(num);
            }
            num++;
        }
    }

    function drawVisualization() {
        const centerX = canvas.width / 2;
        const centerY = canvas.height / 2;
        let radius = 10;
        const incrementAngle = 3.20 * (2 * Math.PI / primes.length);

        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) => {
```

```
            ctx.fillStyle = colors[index % colors.length];
            ctx.fillRect(x, y, 10, 10);
            radius += 10;

            if (index > 0) {
                let prevAngle = (index - 1) * incrementAngle;
                let prevX = radius * Math.cos(prevAngle);
                let prevY = radius * Math.sin(prevAngle);

                // Interconnecting lines
                ctx.strokeStyle = colors[index % colors.length];
                ctx.beginPath();
                ctx.moveTo(prevX, prevY);
                ctx.lineTo(x, y);
                ctx.lineWidth = (index === 3 || index === 320) ? 0.01 * numObjects / 100 : 1;
                ctx.stroke();

                // Solid line from the center to the prime
                ctx.beginPath();
                ctx.moveTo(0, 0);
                ctx.lineTo(x, y);
                ctx.lineWidth = (index === 3 || index === 320) ? 0.01 * numObjects / 100 : 1;
                ctx.stroke();
            }
        });

        ctx.restore();
    }

    canvas.addEventListener('wheel', function(event) {
        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;
        }
        drawVisualization();
    });
```

```
            drawVisualization();
        });

        generatePrimes();
        drawVisualization();
    </script>
</body>
</html>
```

This is 30,000 primes by 3

by-step breakdown of the math involved:

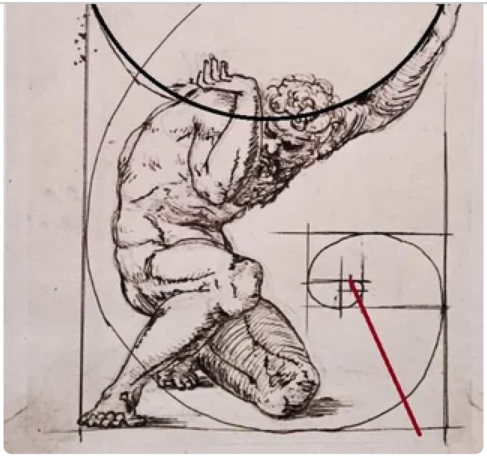
1. **Prime Numbers**: First, we generate a list of prime numbers. A prime number is a natural number greater than 1 that cannot be formed by multiplying two smaller natural numbers.
2. **Polar Coordinates**: Each prime number is plotted using polar coordinates. The polar coordinate system uses an angle and a radius to define the position of a point. In this system:
   - The radius `r` increases for each subsequent prime number.
   - The angle `theta` is incremented by a constant value.
3. **Angle Increment**: The angle increment `Delta theta` is calculated based on a fixed value multiplied by the total number of primes. For example:
   - `Delta theta = k * (2 * pi / total number of primes)` where `k` is a constant that affects the tightness of the spiral.
4. **Position Calculation**: Each prime number is positioned by converting polar coordinates to Cartesian coordinates using the formulas:
   - `x = r * cos(theta)`
   - `y = r * sin(theta)` Here, `r` is the radius (which increases with each prime), and `theta` is the angle for the current prime.
5. **Drawing Connections**: Lines are drawn to connect each prime to the previous prime, and optionally, from the center to the current prime. The width of these lines can be adjusted based on the position of the primes to emphasize certain connections.

By following these steps, you can manually plot prime numbers in a spiral pattern and visualize their distribution. Each prime is represented as a point on the spiral, and the connecting lines highlight the relationships between successive primes and the center.

1 ···

Approved 2 months ago

TheStocksGuy OP • 25d ago • Edited 24d ago

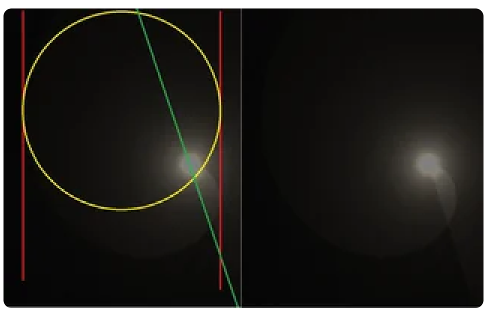


If you zoom out, you'll see the visual HTML/JavaScript Canvas I placed here for anyone to explore, learn from, or improve. I experiment with all of them to see their capabilities. I suspect that Jacob Bronowski created a similar image with scribbled lines that needed correction. I noticed it might align with my perspective, so I checked it out. It did, and you can view the image I found at... https://x.com/OfficialStickPM/status/1870706229938372908

Must be "The golden ratio" along with "Sisyphus and the golden ratio" combined with "The Myth of Sisyphus, by Albert Camus". Had to look this up the hard way lol, I was interested in finding the art.

1 ... Approved 25 days ago

TheStocksGuy OP • 24d ago



You can see this by zooming out on the start on the visual HTML/Javascript using scroll to zoom in & out and it's pretty neat that it matches Jacob Bronowski art or visual art done by someone. Would love more information about it.

1 ... Approved 24 days ago