Rendering zonotope cross-sections

Aresh Pourkavoos

December 21, 2021

This project is the culmination of a train of thought that started around a year ago with the question of whether I could build an icosahedron in Minecraft. I had learned that the coordinates of its vertices were based on the golden ratio $\varphi = \frac{1+\sqrt{5}}{2}$, the positive solution to the equation $\varphi^2 = \varphi + 1$. More specifically, an icosahedron can be constructed with three perpendicular golden rectangles.

[Image]

If the icosahedron has a side length of 2, then the rectangles are $2 \times 2\varphi$, and the coordinates of their vertices are

$$(0, \pm 1, \pm\varphi), (\pm 1, \pm\varphi, 0), (\pm\varphi, 0, \pm 1),$$

or more succinctly, all permutations and sign changes of

$$(0, 1, \varphi).$$

Since φ is irrational, there is no way to scale these coordinates such that they fall exactly on integers. (For the readers who are considering rotations, the proof that they don't work either is left as an exercise.) However, it is possible to approximate it using the Fibonacci numbers, since they obey the same "power law" as φ, i.e. $\varphi^n + \varphi^{n+1} = \varphi^{n+2}$ and $F_n + F_{n+1} = F_{n+2}$. ($F_0 = 0$ and $F_1 = 1$.) For this reason, the ratio of consecutive Fibonacci numbers approaches φ, so we may approximate the coordinates of an icosahedron with

$$(0, F_n, F_{n+1})$$

for some n, e.g. (0, 3, 5) for $n = 4$. Placed into Minecraft, the vertices and the golden rectangles joining them look like this:

[Image]

But how do we fill in the edges? Some of them are just the short edges of the rectangles, so a straight line of blocks will suffice. However, these only account for 6 of the 30 edges, where the other 24 join vertices between different rectangles. For the example above, all of these are permutations and sign changes of

$$(5, 3, 2).$$

In the general Fibonacci case, they would be (F_{n+1}, F_n, F_{n-1}), and in the exact case, they would be $(\varphi, 1, \varphi - 1)$. So how do we fill such a line with blocks in a continuous and "voxel-perfect" way?

Before defining a voxel-perfect line in 3D, it is beneficial to define a pixel-perfect line drawn in 2D. Suppose without loss of generality that it is drawn between $(0, 0)$ and (x, y), where x and y are integers and $0 \leq x \leq y$. To make it pixel-perfect, exactly $y + 1$ pixels must be filled in (including both endpoints), one in each y-coordinate. For a given y-coordinate, the center of the pixel is decided by connecting the centers of the endpoints, intersecting this line with the appropriate row, and filling the pixel whose center lies closest to this intersection. If there is a tie in the rounding, *no* pixel-perfect line exists, since it could not be symmetric without adding an extra pixel in the center. This situation occurs when x is odd and y is even.

[Image]

For voxel-perfection, the situation is similar.

What is a zonotope?

Construction method

C program

The omnitruncated 120-cell

The golden ring

The omnitruncated 24-cell

The $\sqrt{2}$ ring