Mazes and Space-Filling Self-Similar Trees

Marie-Pascale Corcuff

Rennes, France; m-p.c@wanadoo.fr

Abstract

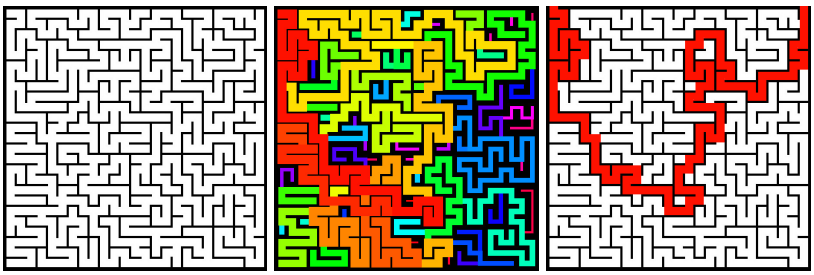
Having defined maze patterns as space-filling trees, a method for generating space-filling self-similar trees from maze pattern seeds, or tiles, is described, as well as their relationship to space-filling self-similar curves.

Mazes

Mazes and labyrinths are often mixed up, but specialists disagree: in a stricter sense "a labyrinth has one winding path with many turns and changes of direction" [1] while a maze has multiple path choices. The path of a labyrinth is unicursal, which means it is impossible to get lost, a maze comports dead ends and sometimes loops, the right path is not straightforward. We explored earlier the relationship between labyrinths and space-filling self-similar curves [2]. We now turn our attention to mazes, with a similar point of view.

Mazes derived from labyrinths, and have evolved today into patterns based upon a grid, where every cell is attained. There are two things to consider when designing a maze. The first, and most important, concerns the pattern itself. Does it comprise only dead ends or also loops? A *perfect* or simply connected maze contains no loop and is considered as a tree in graph theory [3]. We shall consider such patterns going forward. The second concerns the location of the starting and arrival points. Any cell of the grid may be considered as a starting point as well as an arrival point, they do not have to be on the border. However, as for most popular entertaining or testing mazes, we shall put the starting and arrival points on the border, making them entry and exit points; moreover, in order to restrain the possibilities, the entry and exit points will be on corners of a square grid.

One way of generating such a maze pattern consists in drawing a random path starting from one cell of a square grid till it is impossible to go on, and then do the same starting from any cell already attained, till every cell of the grid has been reached.



(a) (b) (c)

Figure 1: *Maze-generating algorithm result: maze pattern (a), chronological tree (b), solved maze (c).*

In Figure 1(a) we see one result of the algorithm in a 25×25 grid, in Figure 1(b) the order in which the different branches have been produced represented by color and width, and in Figure 1(c) one possible maze (with entry at top left corner and exit at top right corner) with its unique solution path .

Such patterns on which mazes are defined are trees, and potentially space-filling since every cell of the grid is attained. That motivates the idea of space-filling self-similar trees on a square grid.

Space-Filling Self-Similar Trees

One of the most known space-filling self-similar trees [4], let us call it X-tree, is defined by the following branching L-system, where ω is the axiom, or initial state:

ω : [X]+[X]+[X]+[X]

F → FF

X → F+[X]+[X]+[X]+[X]

The symbols of the alphabet are interpreted as such:

F : draw a line of unit length; translate to end of line

\+ : rotate 90 degrees counterclockwise

[: push position and angle

] : pop position and angle

With an initial position at the center, an initial angle of 45 degrees, and a reduction of the unit length by half at each step, we obtain the iterations shown in Figure 2.

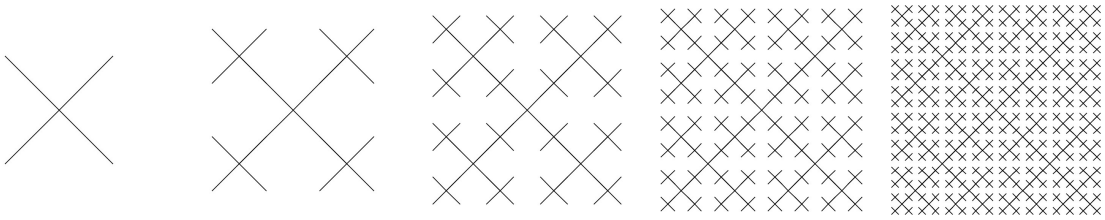


Figure 2: *X-tree, steps 1 to 5.*

We can interpret each vertex as a cell in a grid. In order to better see the underlying grid, let us draw this pattern rotated by 45 degrees (Figure 3).

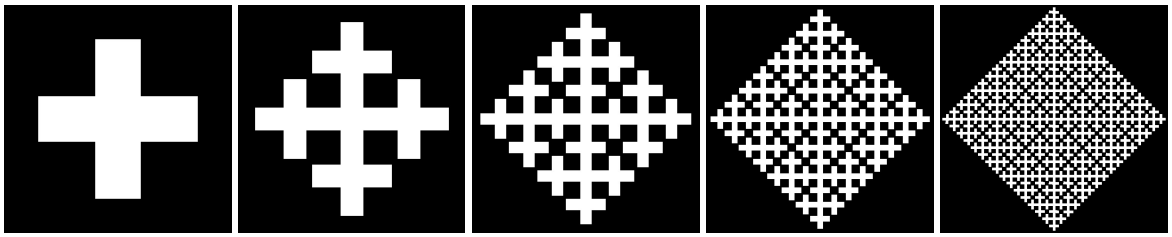


Figure 3: *X-tree on a grid, steps 1 to 5.*

We see that the grid is diamond shaped and not square, and that not every cell is attained, though it does not contradict the space-filling property of this pattern.

The second most known space-filling self-similar tree is commonly named H-tree, and is produced by the following L-system:

ω : [X]++[X]

X → F+[X]++[X]

Symbols are interpreted as before, except for:

[: push position and angle; divide unit length by √2

] : pop position and angle; multiply unit length by √2

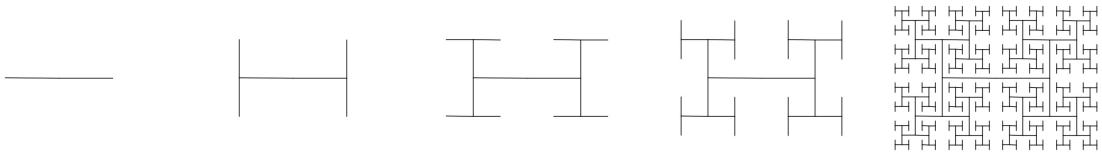


Figure 4: *H-tree: steps 1, 2, 3, 4, 8.*

Let us now try to detect an underlying grid in this case.

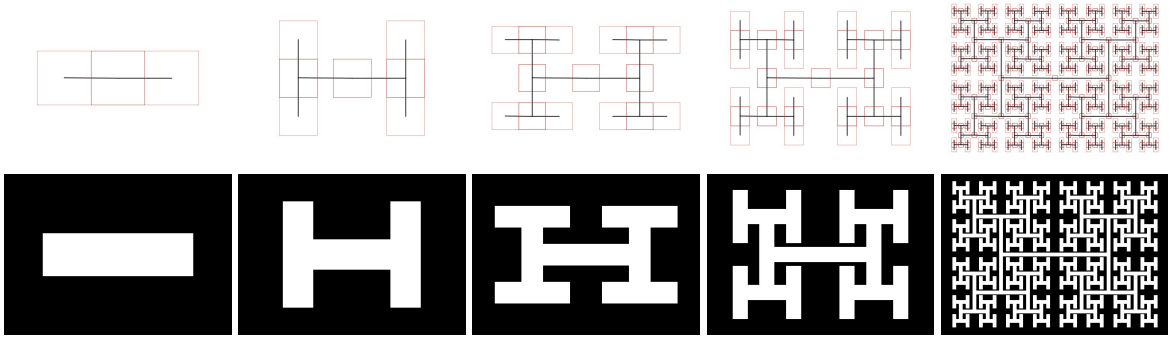


Figure 5: *Search for an underlying grid in H-tree: steps 1, 2, 3, 4, 8.*

We see in Figure 5 an alternately horizontal and vertical regular spacing, but not both at the same time, though it tends to a grid at the limit.

In order to find how to generate space-filling self-similar trees on a square grid, we shall turn back to maze patterns.

Generating Space-Filling Self-Similar Trees from Maze Patterns

The idea is to use maze patterns as *seeds* for an L-system with a process similar to the one described for generating space-Filling, self-Avoiding, Simple and self-Similar - or FASS - curves [5]. As our aim is to generate space-Filling self-Similar trees, they will be called FS Trees for short.

We use a first arbitrary 3×3 maze pattern (seed 3×3_1) obtained by the previously described algorithm and interpret it as an L-system with the edge rewriting method.

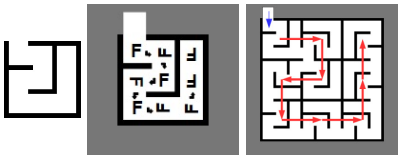


Figure 6: *Seed 3×3_1 and its schemas for an L-system.*

ω : F

F → [TF-YF+YF+YF-YF-YFYF-YFYF] (3×3_1)

T → TTT

Y → YYY

The initial direction is along the y-axis downwards, and the initial position is at the center. The interpretation of symbols is as following:

F : draw a line from (0,-*l*) to (0,0) (without translation)

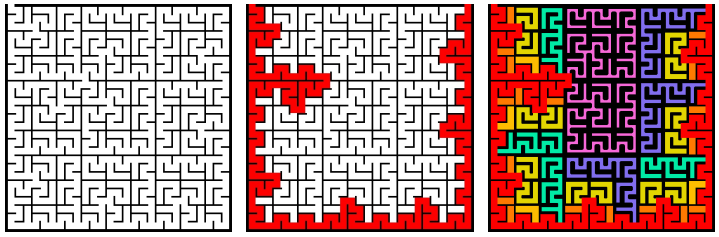
T : translate (-*l*, -*l*)

Y : translate (0, *l*)

+ : rotate 90 degrees counterclockwise

- : rotate 90 degrees clockwise

Lines are drawn with a width of 0,8 *l*. The interpretation of this L-system at step 3 (27×27 grid) is shown in Figure 7, as well as its treatment as a maze with entry and exit points at left top and right top corners respectively.



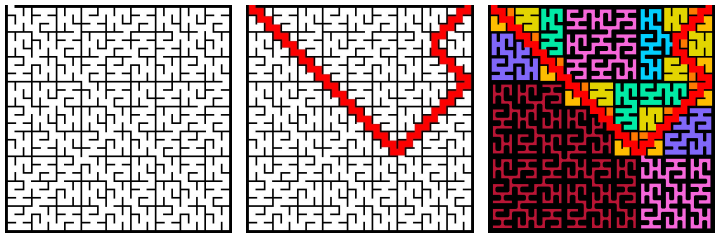
(a) (b) (c)

Figure 7: *Results from seed 3×3_1: FS Tree (a), maze and solution path (b), dead ends (c).*

A second 3×3 seed leads to a new rule for the edge replacement, and its interpretation (Figure 9).

Figure 8: *Seed 3×3_2.*

F → [TF-YF+YF[YF+YF+YF]-YF[+YF]-YF] (3×3_2)



(a) (b) (c)

Figure 9: *Results from seed 3×3_2: FS Tree (a), maze and solving path (b), dead ends (c).*

We can also interpret this process as a recursive tiling, with paths between some adjacent tiles, as seen in Figure 10; colors correspond to the orientations of the tiles.

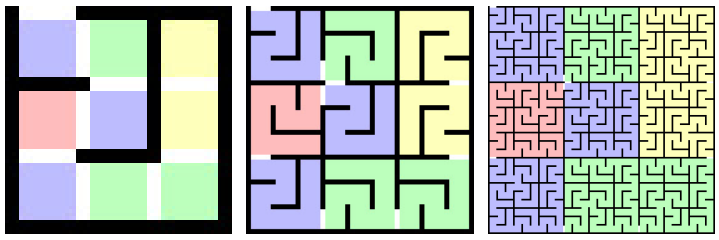


Figure 10: *Recursive tiling with seed 3×3_1.*

How many of these seeds, or tiles, are there on a 3×3 grid? According to the On-Line Encyclopedia of Integer Sequences, there are 192 perfect mazes on a grid of 3×3 cells [6], which produce as many different FS Trees. In comparison to the search for FASS curves this method is much easier because any

pattern is usable, the only constraint in writing the L-system is that each pattern must be linked to the previous one. And the same pattern in a different position will yield a different FS tree.

As there are too many 3×3 seeds to thoroughly explore, let us turn to 2×2 seeds. The only shape that fits is a U shape, but as we can use any position, this leads to four 2×2 seeds:

Figure 11: *2×2 seeds: 2×2_1, 2×2_2, 2×2_3, 2×2_4.*

Those seeds generate the following rewriting rules:

F → [TFYF-YF-YF] (2×2_1)

F → [TF-YF+YF+YF] (2×2_2)

F → [TF[YF]-YF+YF] (2×2_3)

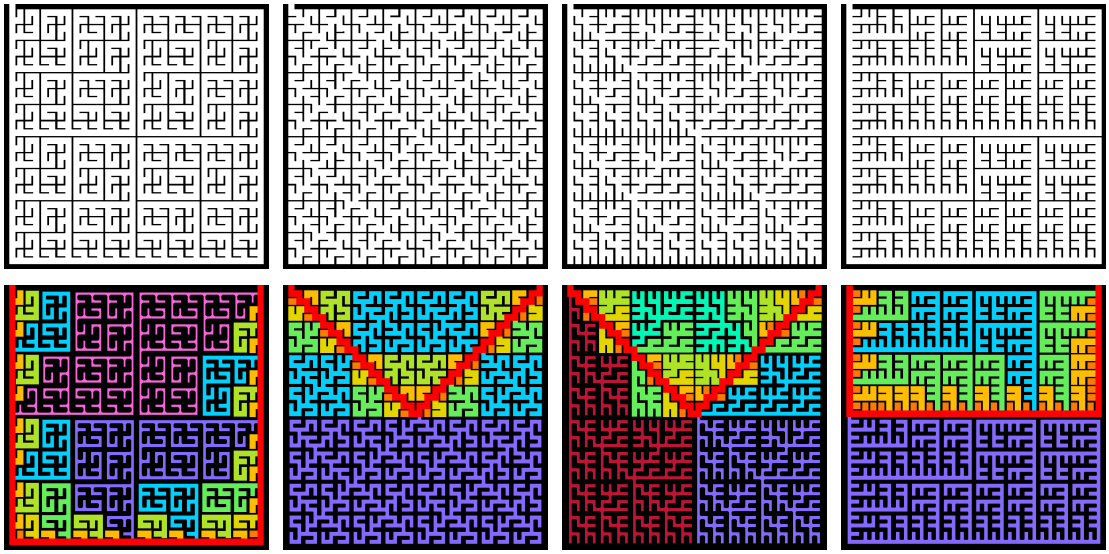
F → [TF[-YF]YF-YF] (2×2_4)

In each case we have:

T → TT

Y → YY

Those seeds produce FS Trees shown in Figure 10 at step 5 (32×32 grid), with their corresponding solved mazes.



(a) (b) (c) (d)

Figure 12: *FS Trees and solved mazes, with seeds: 2×2_1 (a), 2×2_2 (b), 2×2_3 (c), 2×2_4 (d).*

Let us now focus on the self-similarity of those patterns, with 2×2 seeds (Figures 13-16).

Figure 13: *2×2_1 FS Tree, steps 1 to 5.*

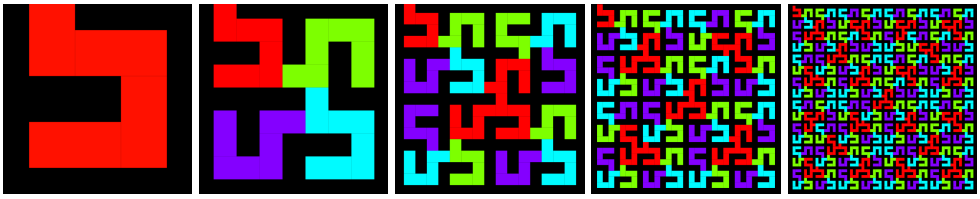


Figure 14: *2×2_2 FS Tree, steps 1 to 5.*

Figure 15: *2×2_3FS Tree, steps 1 to 5.*

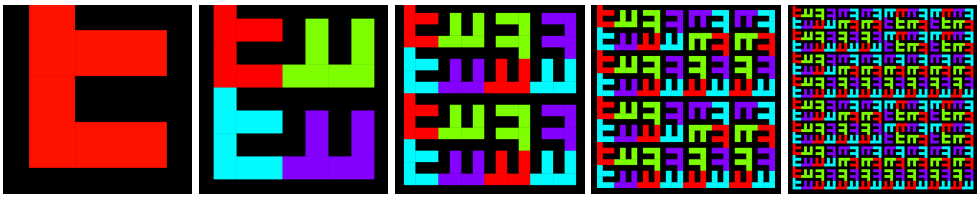


Figure 16: *2×2_4 FS Tree, steps 1 to 5.*

Lines are drawn with a width of 0,5 *l*. The unit length *l* is equal to the initial length *l_init* divided by 2^n for each step *n*. Color changes periodically every 4 F in order to emphasize the basic pattern, or seed.

We can apply the same treatment to FS Trees with 3×3 seeds, the two we have already described, plus two of a particular interest, an S shape and a spiral (Figures 17-20):

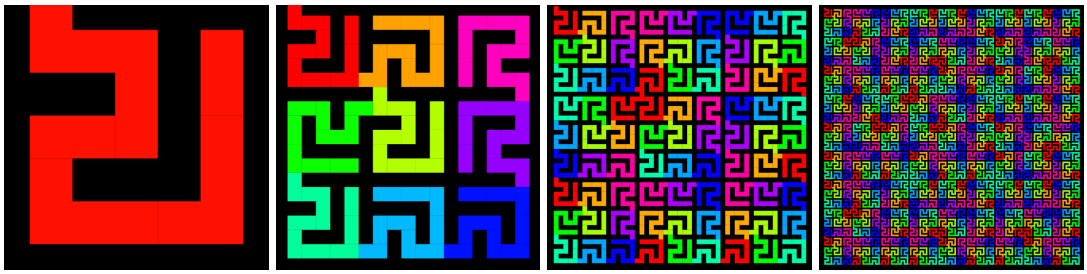


Figure 17: *3×3_1 FS Tree, steps 1 to 4.*

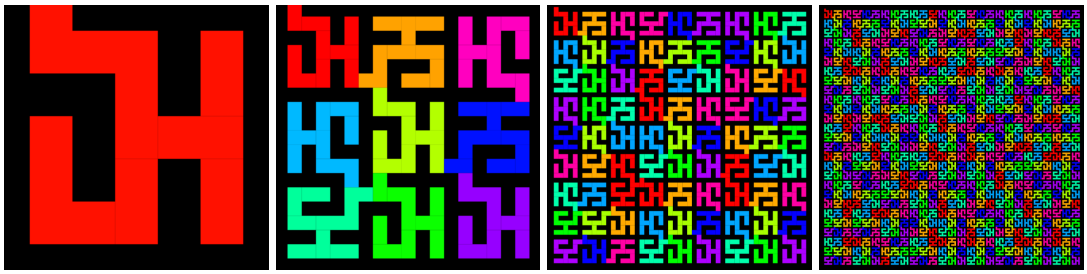


Figure 18: *3×3_2 FS Tree, steps 1 to 4.*

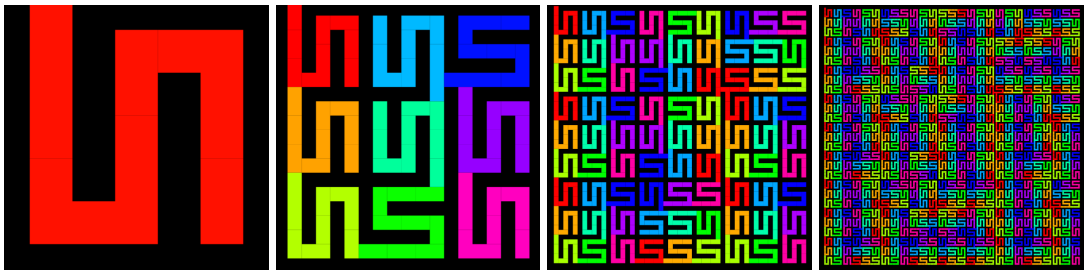


Figure 19: *3×3_3 FS Tree, steps 1 to 4.*

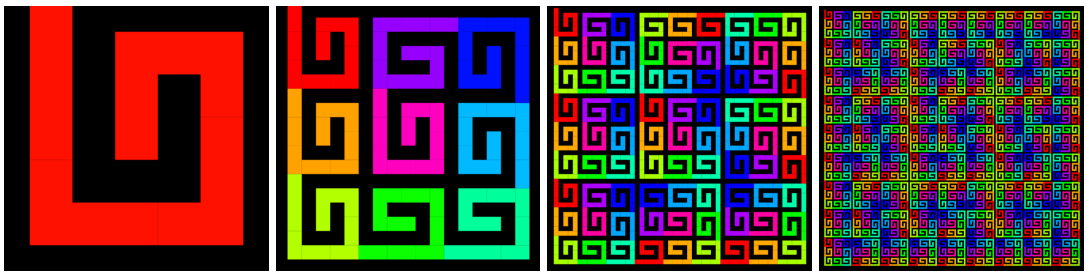


Figure 20: *3×3_4 FS Tree, steps 1 to 4.*

The color change has been applied every 9 F, and $l = l\_init / 3^n$.

Let us now look at these patterns a little differently. The U shape in Figure 9 looks a lot like the first step in the construction of the Hilbert curve, for which the rewriting rules are commonly of the node replacement kind, as following, schematized and interpreted in Figure 21.

ω : R

L → +RF-LFL-FR+

R → -LF+RFR+FL-

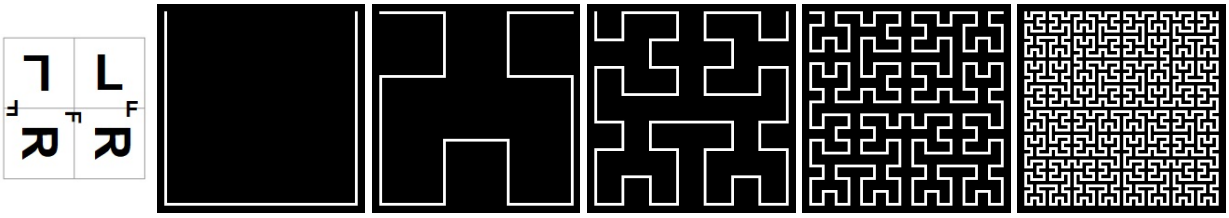


Figure 21: *Hilbert curve: schema of node rewriting rule; steps 1 to 5.*

This gives the idea of a new way to write an L-system for the 2×2_1 FS Tree, with the following node rewriting rule, schematized and interpreted in Figure 22.

ω : U

U → [U]YF[U]Y+YF[U]Y+YF[U]

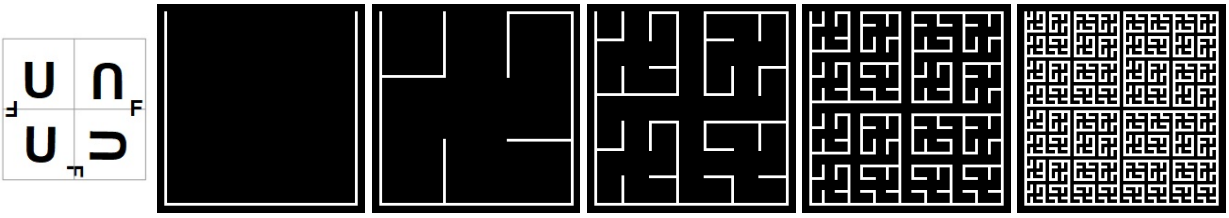


Figure 22: *2×2_1 FS Tree: schema of node rewriting rule; steps 1 to 5.*

In each case, the initial position is at the top left corner; the initial direction is the x-axis for the Hilbert curve, the y-axis for the 2×2_1 FS Tree. F is interpreted as a line and a translation to the end of the line. The unit length must be divided by $d_n = 1+2\ d_{n-1}$ at each step n, with $d_0 = 0$. The width of the lines is constant. Y represents a jump, or a translation without drawing, which has to be adequately tuned for each step according to a rule similar to the one for the unit length.

Finally, we can develop this construction in 3D, with for instance the following rewriting rule:

A → [A]YF[A]Y-YF[A]Y-YF[A]Y#YF[A]Y#YF[A]Y-YF[A]Y-YF[A] (2×2×2_1)

- : rotate 90 degrees clockwise around z-axis

: rotate 90 degrees clockwise around x-axis

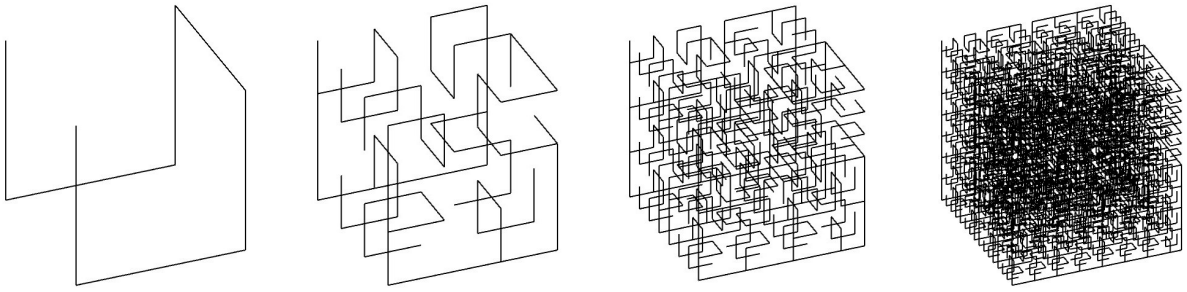


Figure 23: *3D 2×2×2_1 FS Tree, steps 1 to 4.*

Summary and Conclusions

Instigated by a curiosity for mazes, those successors of the labyrinths much more adequate in inducing a feeling of disorientation, this research lead to a method for producing space-filling self-similar trees or FS Trees from seeds which are perfect mazes on a grid. While there are only 4 such seeds on a 2×2 grid their number for larger grids increases rapidly, which accounts for a huge number of different FS Trees. This procedure may be automatized, the rewriting rule being directly created by the maze-generating algorithm.

References

[1] R. Burrows. “Labyrinths: Mysteries and Methods.” *Bridges Conference Proceedings*, On Line, Aug. 1-5, 2020, pp. 345-352. https://archive.bridgesmathart.org/2020/bridges2020-345.pdf

[2] M.-P. Corcuff. “Labyrinths and Space-Filling Curves, Spirals and Tessellations: Topological and Geometrical Implications of Cartesian to Polar Transformations.” *Bridges Conference Proceedings*, Richmond, Virginia, USA, Aug. 1-5, 2024, pp. 107-114. https://archive.bridgesmathart.org/2024/bridges2024-107.pdf

[3] Wikipedia. https://en.wikipedia.org/wiki/Maze-solving_algorithm

[4] J. Kuffner and S. M. LaValle. “Space-Filling Trees.”, Tech. Report, CMU-RI-TR-09-47, Robotics Institute, Carnegie Mellon University, December, 2009. https://lavalle.pl/papers/KufLav09.pdf

[5] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, 1990.

[6] The On-Line Encyclopedia of Integer Sequences. https://oeis.org/A007341