Chapter 1

Declarative texture synthesis

Draft chapter of Paul Harrison's PhD thesis, 7 April 2005.

Previous chapters have examined texture operations based on synthesizing texture *given a sample*. This chapter looks at how an interesting class of textures may be *designed* using rules specifying which elements in a pattern may be adjacent to one another.

The approach of this chapter is to procedural texture synthesis as declarative programming languages are to procedural programming languages, and might be called "declarative texture synthesis". In a procedural language, one specifies a step-by-step procedure by which some output may be produced. In a declarative language, one specifies a set of rules which constrain the output, and it is possible for there to be multiple outputs. Similarly procedural synthesis texture requires a step-by-step procedure for producing the texture to be specified, whereas "declarative" texture systhesis requires a set of rules that constrain the texture that may be produced.

The specific type of rules considered here are specifications of a set of tiles which are to be assembled into a pattern. Some rules will lead to many possible results, some to just one possible result, and some to none at all. If there are many possible results, one is chosen at random.

How simple can local rules of interactions be made, while still producing interesting large-scale structure? It has been mentioned previously that the best-fit synthesis method bears some resemblance to the assembling of a jigsaw puzzle. If best-fit synthesis is like a jigsaw puzzle, it is one with many different pieces, and it is not therefore surprising that the results also can be complex. However, it is shown in this chapter that even simple jigsaw pieces can result in surprisingly complex forms.

1.1 Specification of the problem domain

The objective is to find a connected structure composed only of pieces from a specified set. Pieces are based on squares or hexagons with edges of various types. An edge of a certain type may only abut edges of some other certain type. Some edges of a piece may be "empty", and these edges do not need to abut another piece. All other edges must abut an appropriate opposite edge. Pieces may be rotated, but not flipped.

There are two possible types of edge:

- Like-compatible edges. These edges must abut another edge of the same type, and will be denoted by numbers (`1`, `2`, `3`, ...). "`-`" shall denote an empty edge, which can also be seen as "like compatible".
- Opposite-compatible edges. These edges must abut some different type of edge, and will be denoted by letters, with each lower case letter compatible only with its corresponding upper-case (`a`, `A`, `b`, `B`, `c`, `C`, ...).

A piece may be fully specified by a string listing its edges in clockwise order. For example, `--AaAa`.

Were pieces of this type to be used in a physical jigsaw-like puzzle, edges might be shaped as in Figure 1.1. Alternatively, an appropriate arrangement of magnetic or electrical charges on each edge could be used to ensure each edge type connects only to its intended other. If subjected to random impulses of an appropriate magnitude such pieces would, eventually, self-assemble. This would be interesting to attempt at a molecular level (see also Wolfram [2002, pp. 1193]).

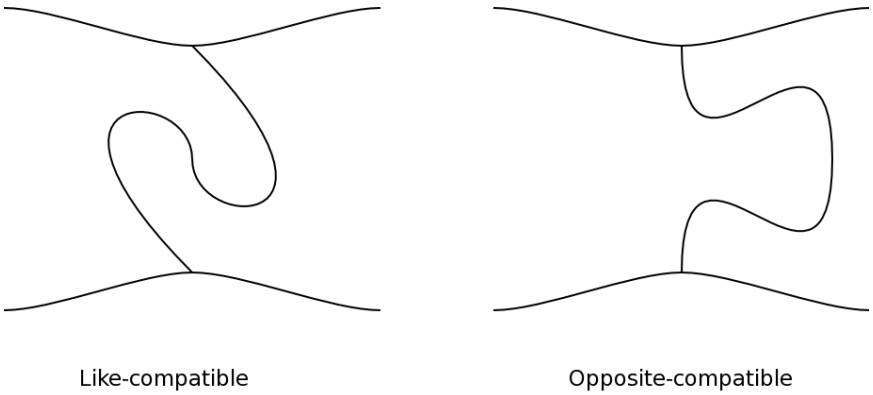

Figure 1.1: The two types of edge, as jigsaw-puzzle type joins.

1.1.1 Relation to Markov Random Fields

Arrangement of tiles depends only on local interactions (the edges of tiles must match), and may be used to define a Markov Random Field. The likelihood of a particular tile being placed at a particular location could be set to either zero, if the tile does not match surrounding edges, or one divided by the number of tiles that could potentially be placed at that location. Similarly the likelihood of an assemblage (of some given size) will be zero if there are edges that do not match, and non-zero otherwise.

1.2 Literature review

The topic of this chapter has one foot in the highly theoretical topic of tiling patterns, and the other in the highly practical field of procedural texture synthesis. Some surprising results from the field of tiling patterns will first be reviewed. A brief overview of procedural texture synthesis will then be given, with some commentary on its limitations.

1.2.1 Tiling patterns

Before the invention of the modern techniques of procedural texture synthesis and tiling of images, one of the most common forms of texturing was tiling patterns produced from small sets of tiles. Such patterns were almost universally repeating patterns, though sometimes of great intricacy.

It is now known that there exist tiles sets that may be used to cover the plane, but for which there are no arrangements of tiles that form a repeating pattern. This has particularly been studied in the context of "Wang tiles". Wang tiles are much like the tile sets considered in this chapter, but with the additional restrictions that the tiles are square, and may not be rotated.[1] The smallest known set of Wang tiles that force non-repetition was found by Culik [1996], and contains 13 tiles.[2]

[1] Careful choice of edge types allows one to force all tiles to follow a consistent orientation, so this is really a sub-set of the tile sets considered here.

[2] A somewhat pointless application of Wang tiles to texture synthesis was described by Cohen et al. [2003], in which each tile is decorated with an image who's borders match those of the tiles that may abut it. This can be used to generate infinite, non-repeating texture. But so can a smaller set of tiles, so long as the edge constraints allow some randomness in the choice of tiles. It is easy for a tile set to *allow* non-repeating patterns, the difficulty lies only in *forcing* non-repetition.

If rotation and arbitrarily shaped tiles are allowed, tile sets that force non-repetition containing as few as two tiles are known. Penrose's "kites and darts" are a well known example of this [Grünbaum and Shephard, 1987, pp.531]. It may be possible for a single tile to force non-repetitive tiling, but no tile is currently known that has this property.

Once one realizes that non-repetitive tile sets are possible, one quickly also realizes that there are tile set analogues for all kinds of systems. Wang tile sets have been described that find prime numbers, enumerate the Fibonacci sequence, and simulating Turing machines and cellular automata [Grünbaum and Shephard, 1987, pp.604–606]. The ability of Wang tiles to simulate Turing machines shows that assembly of tile sets is a form of universal computation.

We may then ask whether randomly chosen tile sets commonly lead to complex computations, or whether this is simply a pathological case. This is a question not of existence but of likelihood, and requires computational experiments rather than mathematical deduction, a form of investigation championed by Stephen Wolfram [2002]. Much of what Wolfram has to say is relevant to this chapter, so his theories will now be described in some detail.

Wolfram examined the general behaviour of many discrete systems using computational experiments, though with a particular focus on "cellular automata", a form of causal Markov Random Field. The sheer breadth of Wolfram's investigations show that his observations have some generality, and we may reasonably expect them to apply to tiling patterns.

Wolfram found that an enumeration of the simplest possible rules for a given class of system will commonly contain systems having surprisingly diverse behaviours. He identifies four basic classes behaviour:

1. Repetitive. The system repeats within a finite time.
2. Nested. The system produces a self-similar, fractal pattern.
3. Autoplectic. The system produces pseudo-random output.
4. Complex. The system produces a diversity of behaviours. The particle-like entities called "gliders", with complex interaction rules, often occur in such systems.[3]

Systems with more complex rules chosen at random were also found to

[3]In a cellular automaton, a glider is a pattern that repeats after some number of time steps, offset some number of squares from its original position.

display these same classes of behaviour. The complexity of the rules of a system appeared to have little effect on the complexity of their behaviour.

Wolfram is of the opinion that simple constraint systems rarely produce interesting results. This is certainly true for the particular family of constraint systems he investigated. He also argues that, although constraint systems can be used to describe limiting states of causal processes as time goes to infinity, such causal processes may take an unreasonably long time to find such states, making constraint systems a poor approximation to observed phenomena. For this reason, Wolfram largely restricts himself to the study of causal systems. It is the author's opinion that while this is an important insight, writing off non-causal systems completely is excessive.

1.2.2 Procedural texture synthesis

A common form of procedural texture synthesis is to define a function mapping position to color values (or sometimes heights on a bump map). Different textures may be produced by different functions. Regularly repeating textures may be produced by sums of sine waves, or of the position modulo some quantity, or functions thereof. Random textures are commonly built from lattice noise functions. Lattice noise functions interpolate between pseudo-random values on a discrete grid, the pseudo-random values being calculated by applying a hash function to their position (this is effectively just a way to generate low-pass filtered white noise). Such noise functions may be summed over several scales to produce fractal noise, or may be used to perturb the position parameter given to some other function to mimic turbulence. A wide variety of natural looking textures can be built by these means, including smoke, marble, wood, and, used as a height function, mountainous terrain [Ebert et al., 1994].

Which of Wolfram's classes can be produced in this way (or mimicked sufficiently well to fool the human eye)? Repetitive or nested texture is straightforward to produce. Random textures that mimic autoplectic phenomena are also straightforward to produce using lattice noise functions. However creation of patterns that imitate textures from Wolfram's complex class is not computationally feasible, as these contain long range correlations that are readily discerned visually, and the texture function would need to recompute these correlations for every point. The inability of this type of procedural texture synthesis to produce complex class textures is a severe limitation, as these textures are often aesthetically interesting.

Note also that such procedural textures are typically thought of as covering the entire plane (or three dimensional space), and any given sample of texture is merely a section cut or carved from this. There is no easy way to

match the details of one texture to another at a boundary. This is a serious drawback, textures in nature often merge into one another in interesting ways (for example the boundary of growth of lichen on rock depending on the texure of the rock, or the stem of a plant extending into a leaf in a branching pattern).

More generally, a procedural texture may use some complex pre-calculation before calculating the value of each pixel. Such generalized procedures *are* able to generate textures belonging to Wolfram's complex class. Examples of such pre-calculations are finding a Voronoi diagram (in order to produce a texture containing cells), or executing some number of iterations of a cellular automaton (such as a reaction diffusion system, to produce spots or stripes [Turk, 1991]). They may also be used to produce textures that blend one into the other [Zhang et al., 2003].

1.3 A tile assembler algorithm

Whether a set of tiles can be assembled into a consistent pattern is undecidable, and within a finite area is NP-complete (see Wolfram [2002, pp. 942]). There is therefore no algorithm that can *guarantee* to assemble an arbitrary set of tiles within a reasonable time. However, an algorithm can be devised that handles many common cases.

Assembly only within a finite area is attempted. The definition of the problem allows us to say that tile placements must conform to a regular grid, and only a finite set of locations need be considered. The algorithm is as follows:

1. Place a randomly chosen tile in the center of the grid.
2. Make a list of empty locations in the grid with abutting non-empty edges. If there are no such locations, halt. Otherwise, if there are any sites where only either one or zero types of tile could be added, restrict the list to just these sites. From the list, choose the location closest to the center of the assemblage.
3. If there is no tile that fits at that location, or if it can be determined that for any tile that might be added the assemblage will become non-completable (see next section), perform backtracking. That is, remove some number of tiles from the assemblage in the reverse order to which they were added (see the section after next).
4. Otherwise choose a tile at random from the remaining possibilities, and put it at the location.

5. Go to step 2.

An implementation is available at
`http://www.logarithmic.net/pfh/ghost-diagrams`. [**todo: And on CD accompanying thesis.**]

1.3.1 Determining if a partial assemblage cannot be completed

An empty cell will be part of a region of empty cells. Whether such a region can ever be filled is completely determined by the edges abutting it. To make use of this fact, a hash-table is kept of region boundaries known not to be completable. If adding a certain tile will create a region that is in this hash-table, adding that tile will make the assemblage non-completable.

If it is determined that no tile can be added at some location, the region that contains that location is added to the hash-table. This allows progressively larger regions to be added to the hash-table, and also prevents previous failures from being repeated.

1.3.2 How much to backtrack

In step 3, the assembler may need to backtrack by some number of tiles. The problem may be in a tile added quite recently, in which case only a small amount of backtracking would be necessary, or in a tile added much earlier, in which case a large portion of the assemblage may need to be discarded.

In the absence of any procedure by which to determine a good number of tiles to backtrack, the approach taken here was to backtrack by a random amount. As we do not know even the scale on which to operate, the amount chosen is sampled from a power-law distribution (power-law distributions are scale-invariant). This ensures that if a large backtrack is required the assembler will perform that backtrack within a reasonable time (for example, in less than the lifetime of the universe).

The specific distribution used was:

$$P(n) \propto (ax)^{-b} \tag{1.1}$$

where n is the number of tiles to remove ($n > 0$, integer), and the parameters

a and b define the personality of the assembler. The parameters affect how long the assembler will take to finish, and may also bias the result towards the emergence of one or another kind of feature.[4] The parameter values used in this chapter were $a = 2, b = 2$.

1.4 Sample results

1.4.1 Single-tile patterns

All possible tiles in from some simple classes were tried, as shown in Figures 1.2, 1.3, 1.4, and 1.5. Only tiles that could be arranged into a self-consistent pattern are shown. Even with simple tiles, a great variety of forms result.

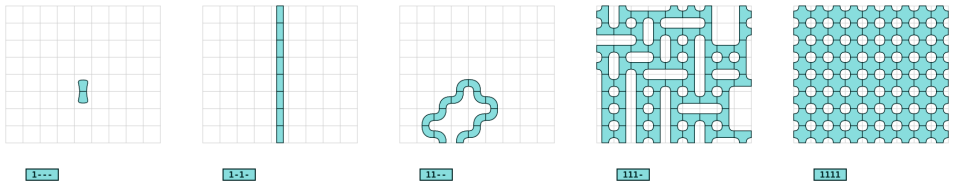

Figure 1.2: Samples of patterns formed by all possible single square tiles having a combination of empty and one type of like-compatible edges.

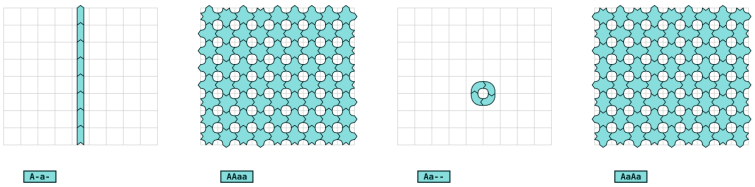

Figure 1.3: Samples of patterns formed by all possible single square tiles having a combination of empty and one type of opposite-compatible edges.

[4]Note that backtracking was also useful in Section **??**. There appears to be a fundamental trade-off between bias and speed, based on the nature and amount of backtracking used. A study of this is beyond the scope of this thesis.

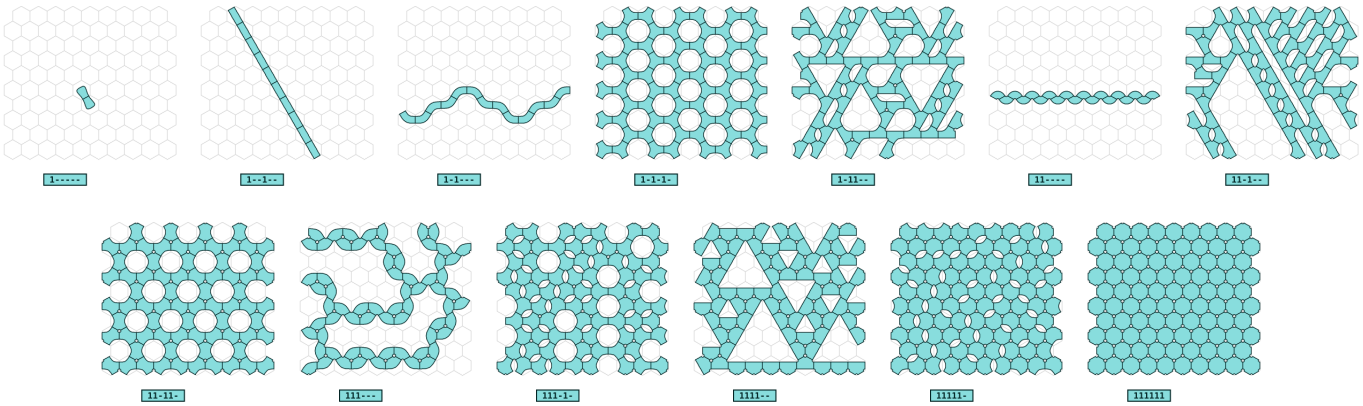

Figure 1.4: Samples of patterns formed by all possible single hexagonal tiles having a combination of empty and one type of like-compatible edges.

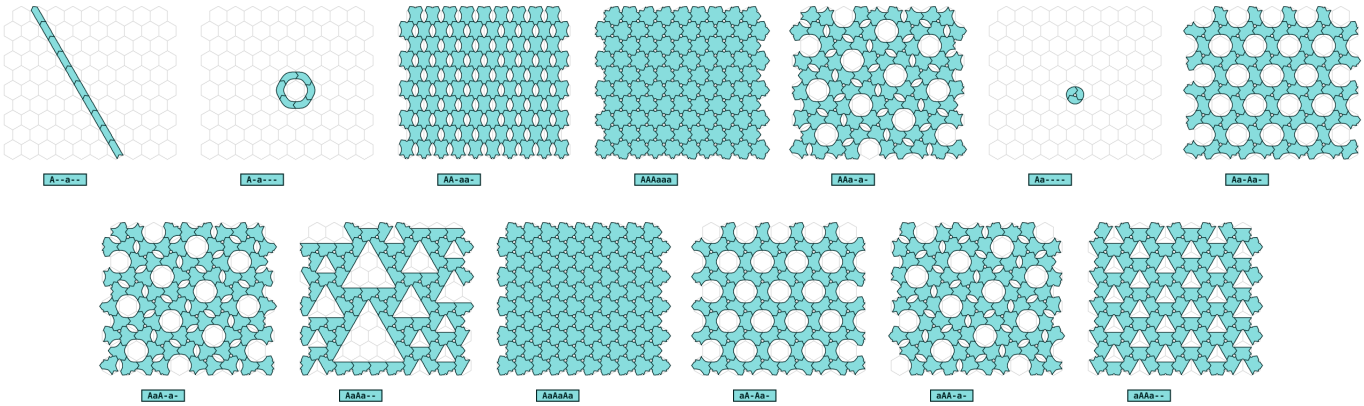

Figure 1.5: Samples of patterns formed by all possible single hexagonal tiles having a combination of empty and one type of opposite-compatible edges.

As might be expected, regular tilings of various kinds can be seen (`1111`, `AAaa`, `AaAa`, `1-1-1-`, `11111`, `AA-aa-`, `AAa-a--`, `Aa-Aa-`, `AaA-a-`, `AaAaAa`, `aAA-a-`, `aAAa--`).

Some tiles form patterns of finite size (`1---`, `Aa--`, `1-----`, `Aa----`, `A-a---`).

`A--a--` and `1--1--` are analogous to simple gliders. If one dimension were to be arbitrarily interpreted as time, these would appear as a single moving tile.

Various forms of randomness can be seen. This is true randomness, not the pseudo-randomness seen in some cellular automata. `11--`, and `1-1---` follow random paths (which may form closed loops). `111---` forms two kinds of

composite units, one with two exposed edges and one with three exposed edges. A random network results, in which the three-edge units are nodes and the two-edge units form random paths between these nodes. `111-` produces rectangles of random size and shape. `1111--` produces triangles of random size and orientation. `AaAa--` produces triangles of random size, but of only one orientation. `AaAa--` turns out to be related to the Sierpinski triangle fractal, as will be seen in the next section.

1.4.2 Further tile sets of interest

Certain tile sets require a great deal of back-tracking on the part of the assembler. These tile sets are often also visually interesting. Two such sets are shown in Figure 1.6. The set on the left yields patterns with interesting local mirror symmetries.

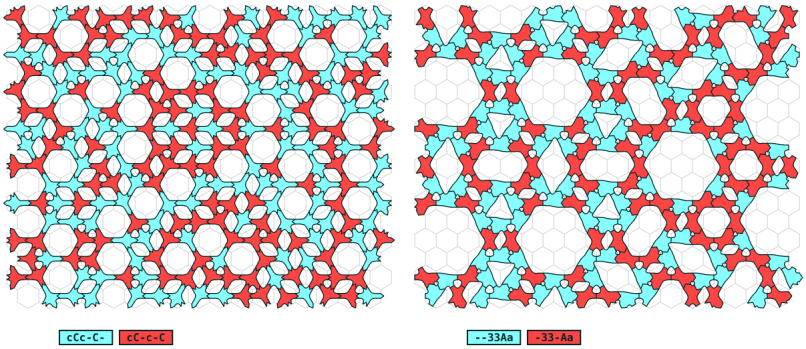

Figure 1.6: Tile sets the assembler has difficulty with.

The addition of one or two tiles to `AaAa--` (Figure 1.4) yields patterns which when finite in size form Sierpinski triangles (Figure 1.7).

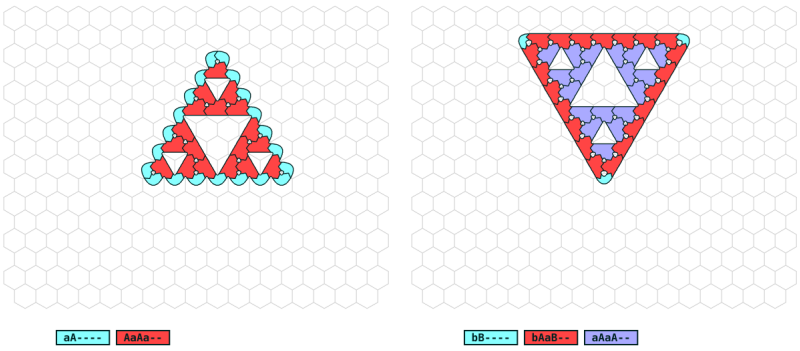

Figure 1.7: Tile sets that yield Sierpinski triangles.

The tile set in Figure 1.8 mimics the structure of DNA.[5] The tiles represent respectively the sugar-phosphate backbone, adenine, thymine, cytosine, and guanine.

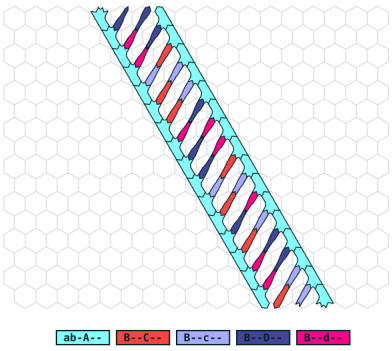

Figure 1.8: Tile set based on DNA.

The tile set on the left in Figure 1.9 is visually reminiscent of a Feynman diagram containing photons, electrons, and positrons. `c--C--` represents electrons and positrons, and `44----` represents photons. Relaxing the requirement that the assemblage be connected, and adding a tile for spontaneous electron-positron creation/annihilation gives rise to "zero-point fluctuations" (tile set on right).

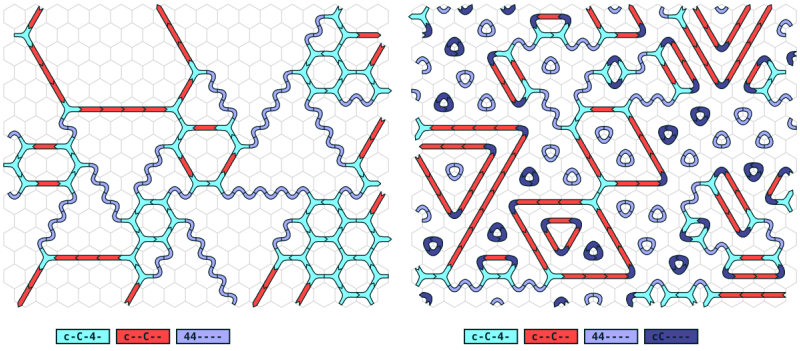

Figure 1.9: Tile sets reminiscent of Feynman diagrams.

Tile sets based on an underlying triangular grid may be simulated using a hexagonal grid (Figure 1.10). This merely requires that every second edge be empty.

[5] This tile set was suggested by Jeff Epler.

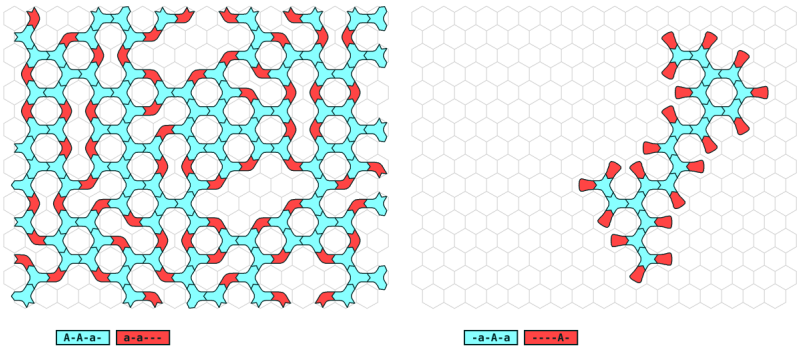

Figure 1.10: Tile sets based on triangles, simulated with hexagons.

If tiles are decorated by drawing lines connecting different edges, the results resemble Celtic knot-work (Figure 1.11).

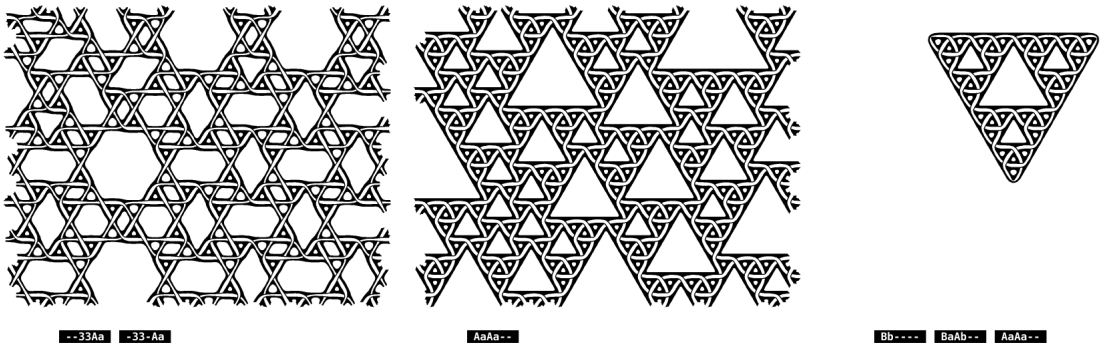

Figure 1.11: Knot-work patterns. The third pattern consists of a single loop.

1.5 k-morphism and crystallization

Grünbaum and Shephard [1987, pp.46–53] describe a number of tiles that allow multiple regular tilings. These are called k-morphisms. An example of this is shown in Figure 1.12. Tile sets were also found that suggest generalizations of this concept. Figure 1.13 shows a tile set that yields either regular tiling or a random network pattern. Figure 1.14 shows a tile set with ∞-morphism.

The assembly process (using the algorithm described earlier) shows behaviour reminiscent crystallization. The assembler initially makes little progress, but eventually produces a sufficient "seed" from which a pattern

crystallizes outwards rapidly. This might also be considered a simple form of auto-catalysis.

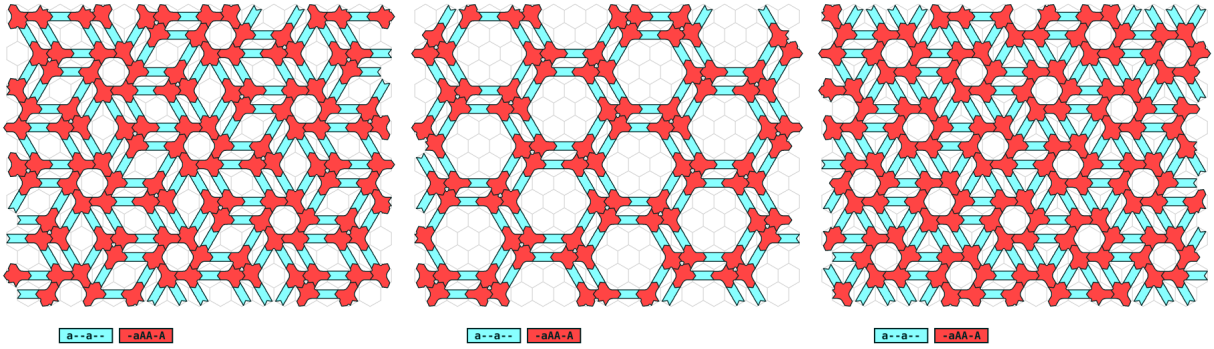

Figure 1.12: A k-morphic tile set. k is at least 3.

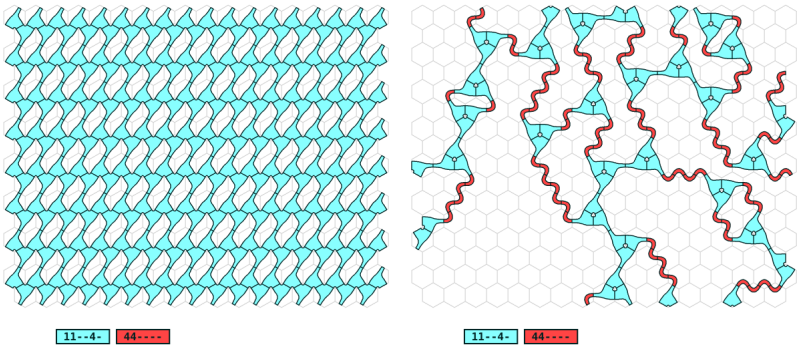

Figure 1.13: A tile set with two mutually exclusive modes.

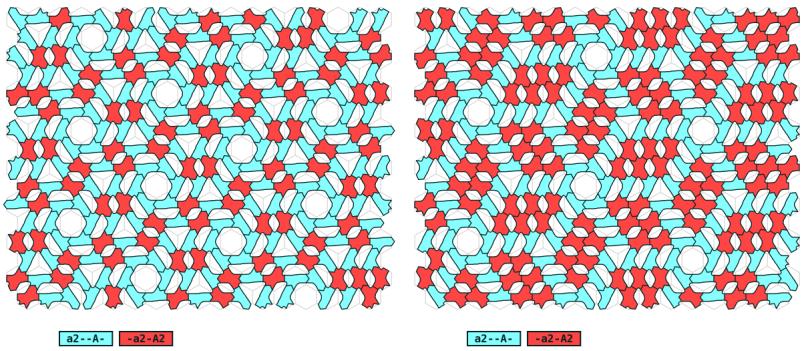

Figure 1.14: An ∞-morphic tile set. The repeating unit can be of arbitrary size.

1.6 Relation to cellular automata

A cellular automaton is a grid of cells in which each cell evolves over time according to simple local rules. By mapping the time dimension into a spatial dimension, appropriately shaped tile pieces can be used to implement cellular automata. This has interesting implications.

1.6.1 Wolfram's Elementary Cellular Automata

Wolfram's "Elementary Cellular Automata" [Wolfram, 2002] are one-dimensional automata in which each cell can be in one of two states, and in which the state of each cell depends only on the previous state of that cell and its two immediate neighbours. Such one-dimensional automata can be mapped into two-dimensional tile sets.

An Elementary Cellular Automaton tile set requires certain crossover pieces so that tiles implementing the rules of the automaton have access to all required input edges. The following tiles suffice: `a-aC-C`, `a-bC-D`, `b-aD-C`, `b-bD-D`.

A rule tile is required for each possible set of inputs, having three input edges as follows:

Input	Edges
000	cac
001	cad
010	cbc
011	cbd
100	dac
101	dad
110	dbc
111	dbd

Each rule tile will also have three output edges that represent the state of the cell, which must be one of:

State	Edges
0	AAA
1	BBB

The rule number of an Elementary Cellular Automaton when written in binary gives the mapping from inputs to states. For example, Rule 110 (Figure 1.15) maps to the tile set: `a-aC-C/a-bC-D/b-aD-C/b-bD-D/cacAAA/dacBBB/cbcBBB/dbcBBB/cadAAA/dadBBB/cbdBBB/dbdAAA`. Rule 110

has been shown by Matthew Cook to be capable of universal computation (see Wolfram [2002, pp. 675–689]).

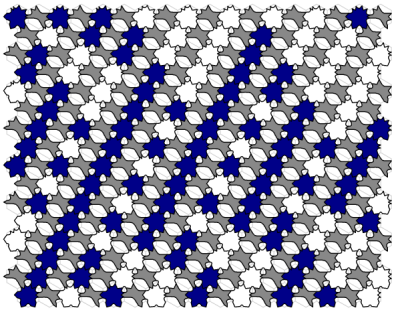

Figure 1.15: Elementary Cellular Automaton Rule 110, as a tile set.

1.6.2 Causality

In the case of tile sets based on cellular automata, a solution may be found without backtracking by selecting pieces in an order corresponding to the automata's time direction. Such tile sets might be described as causal. Furthermore these tile sets are deterministic, in that the anti-causal neighbours of a location fully determine which tile must be placed in it.

That causality is an emergent property of tile sets rather than an assumption (as per cellular automata, and simulation of dynamic systems in general) allows us to think about cases that are "almost" deterministic-causal. There are two obvious ways to break strict deterministic causality:

- We might add several tiles that will match a given previous configuration. The result would be a random cellular automaton.
- We might choose not to include any tile which matches a certain input configuration. This would forbid any past state that would lead inevitably to this configuration. The result would be a tile set that is mostly causal, but with teleological elements.

Combining these two changes would yield a tile set that was "almost" deterministic causal. This would seem to be an interesting generalization. It would be interesting to know if teleological effects would tend to be obscured by random elements on sufficiently large scale in such a tile set.

The importance of this is not clear, and it is possible that it has no importance. The most obvious potential application would be in modelling

quantum-scale phenomena. Entanglement, for example, is straightforward to implement in a near-causal tile-set. The discrete nature of tile-sets is also appealing at this scale. However it is not immediately clear how to account for destructive interference of the wave-function.

1.7 Discussion

A diverse range of patterns were formed from simple tile sets. This is evidence against Wolfram's claim that causality is necessary in order for interesting patterns to be common. Though some tile sets are computationally difficult to assemble, many displaying interesting patterns are not, and these are not strictly those that can be generated in a causal manner. Causal systems are a class of easy-to-assemble tile sets, but there are likely to be other classes that yield patterns of comparable interest with comparable ease of assembly. For example, tile sets whose assembly by the algorithm described above sometimes requires a large backtrack, but on average require total backtracking of the order of the size of the assemblage, can be assembled in linear expected time.

Were the method of texture synthesis described in this chapter applied to decorating a (real or virtual) object, the nature of the tile-set would constrain the resulting decoration. One might even need to slightly redesign the object to suit the tile-set. A certain amount of back-and-forth would be required. It is the author's opinion that objects created using methods that require such back and forth can have greater aesthetic value than objects created using methods that do not. The patterns most difficult to work with may well be the ones of greatest aesthetic value.

This chapter has demonstrated how texture synthesis, in an idealized form, is related to some fundamental properties of computation. The next chapter looks at the implications of this relation for texture synthesis in general.

Bibliography

M. F. Cohen, J. Shade, S. Hiller, and O. Deussen. Wang tiles for image and texture generation. In *Proceedings of SIGGRAPH 2003*, pages 287–294. ACM Press, 2003.

K. Culik. An aperiodic set of 13 wang tiles. *Discrete Mathematics*, 160: 245–251, 1996.

D. S. Ebert, F. K. Musgrave, D. Peachy, K. Perlin, and S. Worley. *Texturing and Modelling: a procedural approach*. Academic Press Inc., 1994.

B. Grünbaum and G. C. Shephard. *Tilings and Patterns*. W. H. Freeman and Company, 1987.

G. Turk. Generating textures on arbitrary surfaces using reaction-diffusion. In *Proceedings of SIGGRAPH 1991*, pages 289–298. ACM Press, 1991.

S. Wolfram. *A New Kind of Science*. Wolfram Media Inc, 2002.

J. Zhang, K. Zhou, L. Velho, B. Guo, and H. Shum. Synthesis of progressively-variant textures on arbitrary surfaces. In *Proceedings of SIGGRAPH 2003*, pages 295–302. ACM Press, 2003.