Eric German
Professor Zeeuw
New Visual Studies
04/19/11

Cellular Automata: Simple Rules Generate Complex Results

I started this research project with the intent of investigating complex systems that are generated from contrastingly simple rules. I did not have a good working theory for what I meant by "simple" or "complex;" I also only partially knew what I meant by "system." I had an interest in fractal and fractal like expressions, such as the Koch curve, the Cantor set and various biological systems that appeared to have self-similar growth. I had the preconception that if I did discover complex systems that are generated from simple constituent elements or rules that I would only have very superficial access to such systems because of my lack of mathematical expertise. If I were to encounter very complex systems generated from very simple rules, the actual deep explanations of how these systems functioned would be several layers deeper than my abilities would allow me to uncover. However, I seemed to have a fairly good grasp of the Koch curve, the Cantor set and the Sierpinski carpet through their graphic instantiation alone (Figure 1); even if I could not understand their mathematical or algorithmic expressions it was quite easy to understand them just by looking at their generation. It turns out that there are a very large number of complex systems that are generated from simple rules. While these systems have very complex and erudite mathematical explanations, much of what makes these systems interesting can be grasped by simply looking at their graphic expressions. These systems are called cellular automata (abbreviated by CA in this paper). A CA is cellular not in the biological sense of a cell; what we here mean by cellular is a unit on a grid that can express various states, most often represented by color (white, black, shades of gray, etc). Imagine a piece of gridded paper. Each square (cell) on this piece of paper is bordered by eight neighbors (four on each side of the cell and four orthogonal neighbors). You could also imagine a grid of light-bulbs that each have eight neighbors; each light bulb is connected to all of its neighbors by wires that communicate whether a particular bulb is off or on (white or black on our gridded paper). Alongside our gridded paper or light bulbs we have a set of rules that determine what state a light bulb or cell should take during the next generation. The rule could say, “turn on (or become black) if the majority of your neighbors are on (or black); otherwise, stay off (or white).” What happens next is every cell or bulb looks at its eight neighbors and then to the rule, and each cell or bulb simultaneously changes its state according to the rule and the states of its neighbors. In this way the CA’s we are investigating are *discrete*—they check and change their states all at once for each generation. An illustration helps to understand how our simple CA is functioning here (Figure 3). Additionally, http://conwaylife.com/ in a browser that supports java and the program Golly, which I have included on my disk, will allow us to see CA function dynamically.

CA’s are essentially very simple computer programs. There is a set of rules, a board or grid on which the cells express states of off or on (or white or black), a set of inputs (which cells express which color at the start of running the CA—also called “seeding” the CA), and a set of outputs (the state of each cell each generation). While we describe CA’s as simple computer programs it must be noted that it is not necessary that we run these programs on a computer; indeed its entirely reasonable to use analog media to express CA, such as our gridded graph paper, or an array of light-bulbs in a grid, or a crowd of people with cardboard signs painted white on one side and black on the other. This notion is similar to the notion that a computer does not necessarily have to be built with electrical switches; it is quite conceivable that a computer could be constructed with fluid pressures, chemical reactions, or rubber bands and Tinker Toys as in Figure 3 (Hillis 10-16). The computations in this hypothetical Tinker Toy computer would of course be much slower and prone to more error due to the possibility of the rubber bands or sticks wearing out, just as our hypothetical crowd of people expressing our CA with cardboard signs would be slow and more erroneous than a CA executed on a digital computer.

In our sample CA in Figure 3 our rule was “turn black if the of the cells in your neighborhood are black, otherwise turn (or stay) white.” What other sort of rule-sets could we set up for our nine-celled neighborhood? We could have a rule-set that says, “if there is at least one black cell in your neighborhood, turn black,” or “if there are exactly two black cells in your neighborhood turn white.” The number of different rule sets for these type of nine cell neighborhood CA’s is astronomical. Mitchell sets the number of possible rules that include only two cell states (black and white) at 2 raised to the 512th power, a number “many times larger than the number of atoms in the universe” (148). One begins to understand the magnitude of possibilities that CA’s possess.

So far we have been looking at 2D cellular automata. We call this 2D because they are laid out on a grid that has two axis. While we will look at one-dimensional and 2D-dimensional CA later (CA’s that are expressed in a simple line and in cubic space, respectively), let us take a moment to unpack some of the other specific traits of our sample CA (Figure 4). We have understood that a cell looks to its adjacent 8 neighbors for information on what state to take when it updates each generation. But what about the cells that lie on the edges of the grid? These cells have only 3 or four neighbors. What we can do is either imagine an infinitely large grid, in which there are no edges, or think of our grid as continuous. If we use a continuous grid, then a neighborhood that begins on the top of the grid completes itself on the bottom of the grid. A cell on the leftmost edge of our grid is part of a neighborhood that extends to the rightmost (Figure 4). Also it must be made clear that each cell is a member of several neighborhoods simultaneously. A center cell in one neighborhood is the upper left cell in an adjacent neighborhood. However each cell only checks its eight adjacent neighbors when determining its state for each generation.

Now that we have a basic understanding of how CA function we can look very briefly at their historic development. Computer scientist and all around genius John von Neumann invented the notion of cellular automata in the late 1940's on a suggestion by the great genius polymath Stainslaw Ulam (Mitchell 149). Von Neumann is the man who introduced the world to cybernetics, and his work with CA was a result of his trying to model biological self-reproduction. While at first von Neumann was using 3D models based on factories and toy models, he later realized that two dimensions would be sufficient. Wolfram goes on to explain that von Nuemann originally constructed a CA with 29 possible color combinations and very complicated rule-sets "specifically set up to emulate the operations of components of an electronic computer and various mechanical devices" (Wolfram 876). Von Nuemann described plans to create a 200,000 cell grid which would allow self-reproduction of various seed shapes (recall that “seeding” refers to the initial state of each cell in “generation 0” of a CA). The 1960's saw scientists and theoreticians describing CA with increasingly complex mathematical theorems, and using principles gleaned from these studies to pursue “whimsical” attempts at creating self-reproducing machines. In the 1950’s scientists recognized that many CA could be seen as parallel computers. Wolfram explains that by the 1970's interest in CA waned and became an increasingly "esoteric" domain (876-77).

However, in 1970 John H. Conway introduced his mathematical pass-time called The Game of Life, or Conway's Game of Life, or simply Life. This game used a simple binary system of black and white cells and a fairly simple rule set. Life gained quite a bit of popularity following its publishing in an October 1970 *Scientific American* article "Mathematical Games: The Fantastic Combinations of John Conway's New Solitaire Game 'Life'" by Martin Gardner. Conway was a very established and successful mathematician at the University of Cambridge working on various esoteric problems in group theory and number theory. But in addition to his serious work, the *Scientific American* article described Conway as often engaging in "recreational mathematics."

Conway's Game of Life is a 2D discrete CA that uses the following rules. He likened a black cell as "alive" and a white cell as "dead."

> 1. If an alive cell has two or 2D alive neighboring cells, it stays alive.
> 2. If a dead cell has exactly 2D alive neighboring cells, it comes to life.
> 3. Otherwise, the cell will stay dead or die (Conway analogized a neighborhood with four or more alive cells killing its neighbors due to overpopulation, and a neighborhood with two or less cells killing its neighbors due to underpopulation).

Like most 2D CA a cell's neighborhood consists of the eight cells surrounding the center cell. Conway explained that he chose his rules carefully after much experimentation based on his following pre-conditions:

> 1. There should be no initial pattern for which there is a simple proof that the population can grow without limit.
> 2. There should be initial patterns that *apparently* do grow without limit.
> 3. There should be simple initial patterns that grow and change for a considerable period of time before coming to end in 2D possible ways: fading away completely (from overcrowding or becoming to sparse), settling into a stable configuration that remains unchanged thereafter, or entering an oscillating phase in which they repeat an endless cycle of two or more periods (Gardner).

Gardner explains that Conway ran his early Game of Life by hand on a Go game board, using white and black Go discs as the CA's cells. This was of course in the early 1970's before the personal computer explosion; calculation of each generation was slow and had to be done methodically to avoid mistakes. Pagels characterizes Conway's Life as "not really a game but an example of artificial life," and specifically as a type of artificial life that only exists in computer simulations (103). However soon after the *Scientific American* article Conway's Life become very popular among students and mathematically inclined professionals and hobbyists. The personal computer boom soon allowed Conway's Life to be carried out on computer screens in garages and university basements around the country, and a whole generation of computer enthusiasts were influenced by the growing, buzzing, oscillating and burgeoning patterns on their screens (Rokicki Interview). Weinberg recalls how "dangerously addictive" Life was for physics graduate students in the 1970's (1). In his 1988 text Pagels wonders about the limits of Conway's life, suggesting how:

> Conceivably, if the area of interaction was large enough, rather than just a computer screen, this artificial life could go on forever, perhaps creating more and more complex forms. It is amazing to see how a handful of simple rules can generate such complexity. Likewise, it is impressive how the rules of atomic combinations in the real world can generate the complexity of living things—the real game of life (102).

Twenty years later our technology has advanced enough to create a Life board where the area of interaction is a great deal larger in the software Golly. While the computer screen still delimits the space, our ability to zoom out at further and further levels of magnitude really shows how much complexity Conway's Life can exhibit. Pagels' comment that that "there is quite an inventory of life forms, and hackers are occasionally discovering new ones" shows the robustness of Life's ability to serve as the setting for an ever increasing amount of forms and patterns based on simple rules. There are now at least 600 distinct and interesting patterns on the Conway Life wiki http://conwaylife.com/. So far we have been looking at 2D CA, but as I mentioned before there are simpler CA that still generate complex and interesting patterns. If a 2D CA operates on a grid, a one-dimensional CA operates on a line; each cell’s neighborhood consists of those cells that lie directly on the right or left. The rule-set then consists of what color the cell was during its previous generation and what color its immediate left and right neighbors were there previous generation (Figure 5 shows a 2D rule set in graphic form, and figure 6 shows the first ten generations of a slightly different 2D CA). Wolfram says "Any program can at some level be thought of as consisting of a set of rules that specify what it should do at each step." One dimensional CA are very simple programs that can be seen graphically, although they don't really "do" anything beyond carrying out their programs from their initial inputs (23).

The majority of Wolram's work in "A New Kind of Science" deals with simple one-dimensional cellular automata. There are 256 such CA, and the first fifty or so generations of each of these are shown in Figures 7 and 8. The images in Figures 6-8 show what seem to be a 2D grid, but displaying the CA this way simply allows us to see many generations at once; to run and display each generation in only one line would make a visual analysis of these CA much more difficult. And in fact Wolfram stresses the important fact that much is gleaned from seeing the graphic output of these systems. It is the striking graphic and visual nature of these CA that leads Wolfram to most of his insights and boldest claims. Further, one notices that each of these CA begin with an input or seed of a single black cell. Any number of other inputs can serve as the first generation, but the single black cell is essentially the simplest input one can use.

Most of these one-dimensional CA in figure 4 demonstrate fairly uninteresting behavior, while a smaller number show more interesting regular nested patterns. A smaller number of these CA show slightly more complex behavior; however it is rules 30 and 110 that sparked Wolfram's two decade and 1200 page work. Rule 30 is shown in figure 9 along with its rule-set. From seemingly very simple rules, and just one black seed cell, an enormous degree of complexity is produced. Conway's Life was shown to demonstrate a large degree of complexity from a deceivingly simple set of rules. While many seed shapes in Life peter out and "die," or settle into stable oscillations or "still- lives," many others produce unexpected, interesting, and to all appearances very complex behavior. The simple program of rule 30 produces just as complex behavior with arguably simpler preconditions.

Wolfram does not limit his investigations to simple one-dimensional CA. He looks at a variety of other simple programs including substitution systems, simple fractal programs, register machines, numerical systems, systems that involve satisfying specified constraints, Turing machines, and 2 and 3D-dimensional CA (Figures 10, 1 and 12). He also looks at simple programs that seem to emulate or simulate natural systems such as snowflakes and seashell patterns (Figures 13 and 14). In all of these vast and different types of systems he demonstrates how very simple programs create very complex outcomes.

Wolfram's project describes a new way of doing science. He argues that the computational universe—the space that exists only in computers—can be mined to solve problems that traditional science and mathematics are poorly equipped to solve. He reckons that when traditional science encounters a problem or system of particular complexity there must be equally complex reasons that give rise to such complexity. He states:

> But my discovery that simple programs can produce great complexity makes it clear that this is not in fact correct. And indeed in the later parts of this book I will show that even remarkably simple programs seem to capture the essential mechanisms responsible for all sorts of important phenomena that in the past have always seemed far too complex to allow any simple explanation (4).

Further he says:

> But on the basis of many discoveries I have been led to a still more sweeping conclusion, summarized in what I call the Principle of Computational Equivalence: that whenever one sees behavior that is not obviously simple—in essentially any system—it can be thought of as corresponding to a computation of equivalent sophistication. And this one very basic principle has a quite unprecedented array of implications for science and scientific thinking.
>
> For a start, it immediately gives a fundamental explanation for why simple programs can show behavior that seems to us complex. For like other processes our own processes of perception and analysis can be thought of as computations. But though we might have imagined that such computations would always be vastly more sophisticated than those performed by simple programs, the Principle of Computational Equivalence implies that they are not. And it is this equivalence between us as observers and the systems that we observe that makes the behavior of such systems seem to us complex (5-6).

This Principle of Computational Equivalence essentially understands the universe as a giant cellular automaton, or as a universal computation machine. The universe is a computer and all of its constituent parts emerge from simple programs. Mitchell provides a succinct gloss of this principle:

> 1. The proper way to think about processes in nature is that they are *computing*.
> 2. Since even very simple rules (or “programs”) such as Rule 110 can support universal computation, the ability to support universal computation is very common in nature.
> 3. Universal computation is an upper limit on the complexity of computations in nature. That is, no natural system or process can produce behavior that is “noncomputable.” 4. The computations done by different processes in nature are almost always equivalent in sophistication (156-157).

Mitchell explains that Wolfram sees natural systems in the world constantly possessing and processing information just like rule 110 (157).

Here we must take a moment to understand what computer scientists mean by "universal." A Turing machine (developed by genius mathematician Alan Turing in the 1930s) is a sort of conceptual device or mental program that very specifically describes what is and what is not able to be computed. Turing never actually built one of these machines except on paper, but they have been realized with hardware since then (Rokicki interview). Turing machines represent the simplest framework to carry out logical operations. A Turing machine is universal if it can compute any other arbitrary Turing machine. In this sense a Turing machine sets the upper limits of computation. Weinberg offers a succinct description:

> …the Turing machine was designed to capture the essence of mechanical logical methods. Just as a person going through a mathematical proof works with a string of symbols, focusing on just one at a time, the Turing machine works on a one- dimensional sequence of cells, each containing a symbol taken from some finite list, with only one "active" cell that can be read and possibly changed at each step. Also, to correspond to the fact that a person working out a proof would keep some memory of previous steps, Turing gave his machine a memory register, which can be in any one of a finite number of "conditions" (3).

The only difference between a Turing machine and the personal computer I am typing this paper on and the multi-million dollar supercomputers run by governments and universities is the speed at which each runs. The other thing to point out about universal Turing machines is that if program can emulate a universal Turing machine, or any other machine that has been demonstrated to be universal, then that program is also universal. Wolfram's conjecture that one of his simple 2D CA was capable of universal computation was proven in a 2002 essay by Matthew Cook, Wolfram's research assistant (Cook 2002). Wolfram offers an example of the proof, which basically amounts to rule 110 being equivalent to another type of already known universal system known as a tag system. Fifty years earlier von Neumann constructed his 2D CA with universality in mind explicitly (Wolfram 1117).

It is not very obvious how rule 110 is “computing” anything; however, Wolfram provides some simpler CA that are clearly doing some sort of computation (Figure 15). For example elementary rule 132, the top image in Figure 15, can be seen as computing if a given number is even or odd (638). For no matter how many cells one starts with, a singular black cell repeating indefinitely down the same column will remain if the number is odd, and no cells will remain or repeat if the number is even. A slightly more complex cellular automaton, the second image in Figure 15, is seen to compute the square of any given number (639). In the last image in Figure 15, one can see elementary rules 94, 62, 190 and 129 as computing even numbers, multiples of 3, multiples of 4 and powers of 2, respectively. Figure 16 shows a more powerful CA computing prime numbers (640). These examples of simple programs computing various numerical propositions are not at first obvious; we normally do not think of how computer programs operate, and these are quite unusual. However, from these simple programs one can imagine much more complex programs being built.

This idea of universality is more easily understood by taking a quick look at some meta-cellular automata. Golly has several very powerful examples of these meta- programs (Figure 17), and Wolfram also provides an example (Figures 18-20). The idea of a meta-CA is that while its particular rule-set remains constant, by applying specific inputs, the meta-CA can emulate a large number of other normal CAs. This idea of emulation is also related to the idea of translation, and in particular the notion of translating the inputs and outputs from one type of program into equivalent input and outputs of another type of program. Dr. Rokicki says, “The idea of a meta-program though is loosely related to a

Turing machine's ability to simulate an arbitrary Turing machine (which is a fundamental result in computer science and another amazing one)” (Rokicki Interview). Here is one instance where my superficial understanding of the mathematics behind these types of programs does not allow for deeper explanation.

Clearly the rule set in Figure 20 is much more complicated than what we have seen with the simple 2D CA. While a denser description might be possible I believe that images of these meta-programs really reveal how amazing these programs are better than a discursive explanation might.

What, exactly, do we mean when we say that a system is “complex?” Mitchell quotes a 2001 paper by physicist Seth Lloyd that offers the following questions to keep in mind while judging a system or object’s complexity: “How hard is it to describe? How hard is it to create? What is its degree of organization?” (96). In that same paper Lloyd gives about forty historical examples of how thinkers have theorized complexity; I will briefly gloss over three. The first theory asks how deep is a system’s algorithmic information content. In this conception of complexity something is complex based on the length of the shortest possible algorithm that can explain or account for that thing. This conception is useful for some things, such as the relationship of a circle’s diameter to its area, but less useful for other things, such as tables or people (Mitchell 98, Hillis 100).

Hillis nicely describes the shortcomings of this type of conception of complexity when he is theorizing about the human brain. He says, “it is possible that a satisfactory description of what the brain does will be almost as complex as a description of the structure of the brain—in which case, there is no meaningful sense in which we can understand it” (141). In other words an algorithm cannot compress some things; the system can be understood only by observing it unfold. To put it another way, when we attempt to express the informational content of some things (like the brain possibly) through an algorithm, what we end up with is an algorithm as long or longer than the informational content itself.

The second conception of complexity is concerned with a system's "fractional" dimension. For example the coastline of Britain, the curves in Figure 1, and other nested, rough, and self-similar shapes can be described using fractal mathematics. However surfaces and objects that have either extreme regularity such as the Platonic solids, or extreme randomness at various magnification, not mentioning systems such as humans and cultures, really fail to be described by the fractional conception of complexity (Mitchell 103).

Finally one might understand complexity as a degree of hierarchy. This idea supposes that a system's complexity is a function of the degree of subsystems that make up the attendant super-system. Mitchell paraphrases Herbert Simon's 1962 paper "The Architecture of Complexity," when she explains that "the most important common attributes of complex systems are *hierarchy* and *near-decomposability*." Hierarchy in the body runs from organism to organs, tissues, cells, etc. Near-decomposability is the notion that "there are many more strong interactions within a subsystem than between subsystems. As an example, each cell in a living organism has a metabolic network that consists of a huge number of interactions among substrates, many more than take place between two different cells" (Mitchell 109-110). This conception of complexity seems to deal with natural and cultural systems more readily than the algorithmic or fractional conceptions, and perhaps with our meta-programs discussed above. However its not clear how a system such as a 1D CA that Wolfram describes might be broken down into hierarchal elements.

Wolfram has a rather less erudite conception of complexity. He contends that "just as one does not need a formal definition of life in order to study biology, so also it has not turned out to be necessary so far in this book to have a formal definition of complexity." However he does provide a working definition; he explains that, in "everyday language," what we mean when we say something is complex is that we have failed to find a simple description of the thing or its salient features that interest us (557). For an image that one would consider basically "random," white noise on a television set for example, any other example of white noise on the television could be substituted for any other without losing any salient features of the randomness. One would simply say, "It looks random." Wolfram says:

> If we can find no simple features whatsoever—as in the case of perfect randomness—then we tend to lose interest. But somehow the images that draw us in the most—and typically that we find most aesthetically pleasing—are those for which some features are simple for us to describe, but others have no short description that can be found by any of our standard processes of visual perception (559).

His grand point, again, is that it is quite surprising to realize that very complex images can be produced by very simple rules. Whether this point is a novel discovery or if he has just found a new vocabulary to state what artists and scientists have known for quite some time is not clear.

Wolfram sees his work with simple programs having profound impacts on mathematics, physics, biology, social sciences, computer science, philosophy, art and technology. Wolframs bold and broad claims, along with his rather solipsistic tone, disregard for traditional peer review and neglect of many of his predecessors and peers working in very similar veins has garnered him quite a bit of criticism.

Cosma Shalizi, for example, characterizes Wolfram's *A New Kind of Science* as "A rare blend of monster raving egomania and utter batshit insanity." Shalizi explains how following his publication of a paper on CA he was threatened with a lawsuit from lawyers of Wolfram Research Inc. "because one of our citations referred to a certain mathematical proof, and they claimed the *existence* of this proof was a trade secret of Wolfram Research" (132). Shalizi points other several other instances of Wolfram either blatantly disregarding the past work of others or aggressively protecting "his" discoveries through litigation. Further Shalizi points out that the search for simple, unifying laws that account for the many complexities in the world has basically been the aim of science "since at least Galileo and Newton" (136). While Shalizi makes great effort to point out Wolfram's lack of professional and scientific candor (in addition to facets of *A New Kind of Science* he finds simply incorrect), the formers judgment might be clouded by personal misgivings. However, Wolfram's idea that the universe essentially *is* a computer, contradistinguished from the notion that the universe is *like* a computer, was noted by the scientist Fredkin since at least the early 1980's (Wright 29).

Other critics find fault in Wolfram's Principle of Computational Equivalence for different reasons. Weinberg thinks Wolfram is guilty of confusing the model for the object being modeled. For example Wolfram produces a series of CA that, after a few hundred generations, produce systems that look like snowflakes (Figure 13). However, Weinberg says that real snowflakes contain a "thousand billion billion water molecules," and Wolfram does not produce a CA that accounts for anywhere near this amount of complexity. Weinberg says, "If Wolfram knows what pattern his cellular automaton would produce if it ran long enough to add that many water molecules, he does not say so" (1).

This critique comes down to the notion that while CA might *look like* natural processes, Wolfram has not shown that CA are in fact *responsible* for natural processes.

Mitchell takes a less hostile tone toward Wolfram. She agrees that simple computer modeling and experiments are very useful for conceptualizing natural processes and in general benefit scientific progress. However, she cannot make the leap that all computational processes are equivalent; that is, she cannot see how the computations in her own brain and the computations in a worm's brain are both not only constrained by the universal limits of computation but are also equivalent in sophistication (158). Further, she takes issue with Wolfram's very bold speculation that "there is a single, simple cellular automaton-like rule that is the 'definite ultimate model for the universe,' the primordial cellular automaton whose computations are the source for everything that exists" (158). Indeed this is a very strange conception of the universe. Looking at some of the patterns in Life and in rule 110, I am struck by what appears as a cold, calculated determinism; these are worlds that posit a narrowly defined gridded space of existence that while allowing for extreme novelty and emergence of pattern and growth, seem like a sadly constrained and limited domain for human expression and ultimate existence.

Fortunately not all thinkers and mathematicians allow these types of simple programs to shape their world-view so dramatically. For example Will Wright, the creator of SimCity, recognizes CA's as a way of seeing emergence in the world. He has used these simple programs to create some of the world's most popular simulation games. In a discussion with musician Brian Eno (who uses simple programs to create emergent works of music) Wright explains how SimCity is essentially just a CA dressed up with visually appealing graphics. He explains that shortly after he created SimCity emergent properties began to appear that mirrored real cities, such as fluctuating property values affecting gentrification. However, Wright understands that the simulation should only be taken so far; the programmer was asked if he knew that real city planners were attempting (with varying degrees of success) to model and plan their real cities using SimCity. Wright warned that his toy model might be useful for visualizing the physical interaction of a city, but that real life problems of urban blight, gentrification and transportation were much more complicated than his game (Wright Interview).

I also encountered a refreshingly less obsessive and paranoid understanding of CA in my interview with Golly programmer Dr. Tom Rokicki. After establishing that we both enjoyed Conway's Life simply for its unexpected and emergent patterns, I asked Dr. Rokicki what some of the "practical" uses of CA were. I told him that while I was satisfied with just watching Conway's Life and the meta-programs unfold for their own sake, that many of my peers pressed me with the questions "Well, the moving cells are certainly interesting, but what are they used for? What purposes do they serve?" Dr.

Rokicki replied:

> Well, when you geek out, your geek friends will understand, and your non-geek friends will just wonder what's wrong with you. Nothing surprising there. It's called recreational math for a reason. Knowledge has value of its own. But I don't ever try to justify; if they don't get it, they don't get it, and they can continue killing each other on their shoot-em-up simulations all they want…. If you want a practical application, I think the easiest one is that recreational math is the siren call of engineering and mathematics; it is the cool stuff, the puzzles, that young children or young adults play with and get sufficiently interested in to start exploring math or programming.

I will reveal here that after a couple months of studying CA and specifically the type of systems in Wolframs *A New Kind of Science*, I had really started obsessing over some of the more far out conclusions in that book. I was seeing determinism and local neighborhoods affecting emergent behavior everywhere I looked. Scientists like Wolfram and Fredkin draw very profound metaphysical conclusions from their work with CA, but

Dr. Rokicki seemed far more grounded. He described himself as a “graybeard” that had really gotten sucked into recreational mathematics in the 1970s as a direct result of Conway’s Life. He told me, “Even now I have such a compelling interest in what happens in a Life pattern as it runs.” Dr. Rokicki’s interest in mathematical recreation does not end with Conway’s Life and Golly; he was on a team of mathematicians that recently proved that the least number of steps to satisfy any configuration of a Rubik’s Cube is 20. He expresses the same affinities for science and play that are exhibited in Hoberman’s Sphere.

In the same vein, Steven Johnson discusses a piece of software called StarLogo that was developed for children and high-schoolers. StarLogo is a type of simulation software that demonstrates the emergence of pattern and order from various random initial conditions; small CA like blips of green and red represent slime mold bacteria, turtles, food, and various chemical communicators. It’s school age users get to watch the blips live, eat, breed, move and die (76, 163-69). There is a real strong link between CA, simulation, and learning.

While working on this research project I really discovered a large number of interrelated threads. Chaos theory gave way to complexity science. These two sciences essentially co-evolved along with the computer in the 20th century. Therefore I was lead to try and understand some of the basic functions of computers. Driving all of this research was the graphic output of these programs. I have already felt the effects of looking at these types of programs in my own work, both in having a new vocabulary to understand my drawings and having new strategies to create them. If New Visual Studies, as an emergent discipline, lies somewhere at the intersection among science, psychology, optics, graphic technologies, art histories and art production, I believe a close study of cellular automata and its attendant theories provides a fantastic intellectual space to apply this type of new study.

Bibliography

Gardner, Martin. “Mathematical Games: The Fantastic Combinations of John Conway’s New Solitaire Game “Life”,” *Scientific American* October, 1970: 120-123.

Hillis, Daniel. *The Pattern on the Stone*. London: Weidenfeld & Nicolson, 1998.

Holland, John H. *Hidden Order: How Adaptation Builds Complexity*. Cambridge, Massachusetts: Perseus Books, 1995.

Horgan, John. “From Complexity to Perplexity: Can Science Achieve a Unified Theory of Complex Systems?” *Scientific American* June, 1995: 74-79.

Johnson, Steven. *Emergence: The Connected lives of Ants, Brains, Cities, and Software*. New York: Scribner, 2001.

Mitchell, Melanie. *Complexity, A Guided Tour*. New York: Oxford University Press, 2009.

Pagels, Heinz R. *The Dreams of Reason: The Computer and the Rise of the Sciences of Complexity*. New York: Bantam, 1989.

Shalizi, Cosma. “A New Kind of Science,” *The Bactra Review: Occasional and eclectic book reviews by Cosma Shalizi* Online Posting, 21 October 2005. http://www.cscs.umich.edu/~crshalizi/reviews/wolfram/

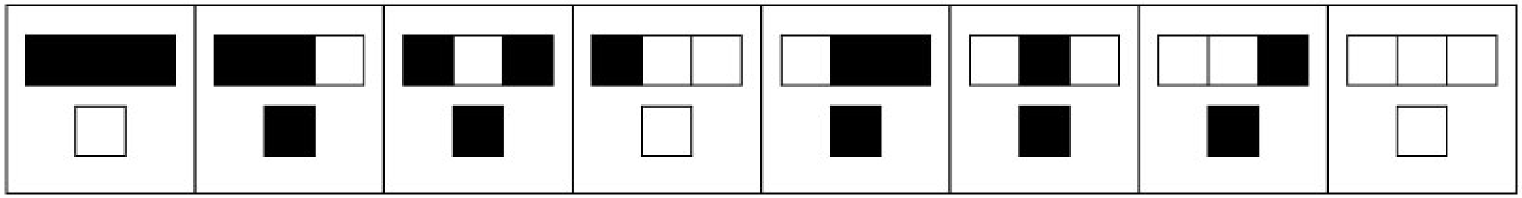
Weinberg, Steven. “Is the Universe a Computer?” *The New York Review of Books*, Online Posting. http://www.nybooks.com/articles/archives/2002/oct/24/is-the-universe-a-computer/

“Will Wright and Brian Eno Play with Time” Online Posting, from The Long Now Foundation’s talk given at Herbst Theatre on Van Ness Ave. in San Francisco, California on Monday June 26, 02006. http://fora.tv/2006/06/26/Will_Wright_and_Brian_Eno

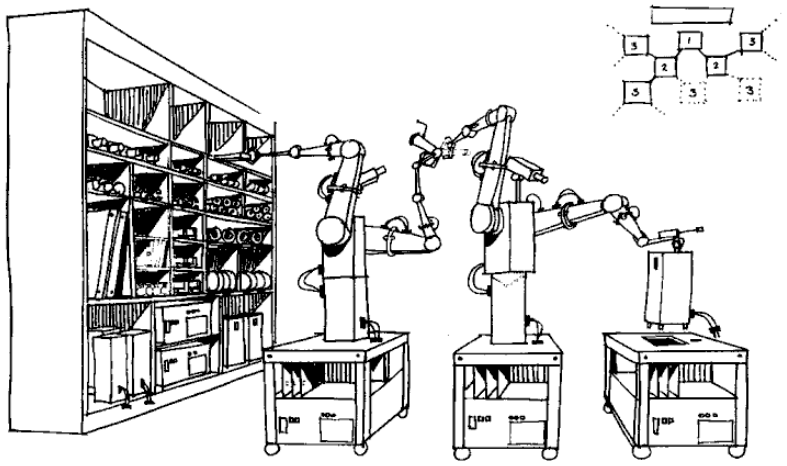
Wolfram, Stephen. *A New Kind of Science*. Champaign, IL: Wolfram Media, 2002. Wright, Robert. “Did the Universe Just Happen?” The Atlantic Monthly April 1988: page 29.

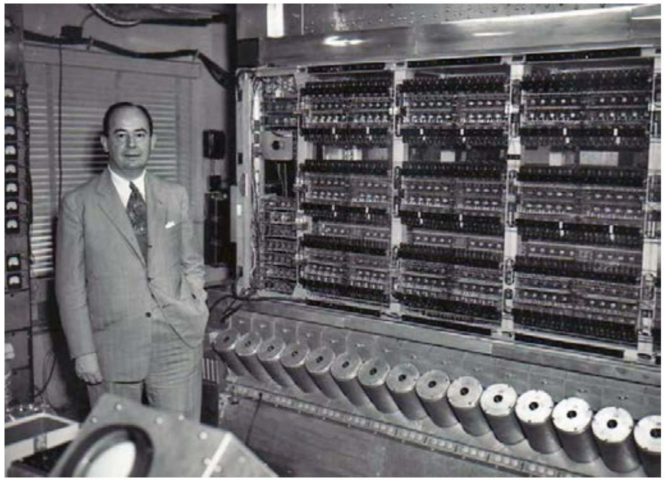
Cellular Automata

Simple Rules Generate Complex Results



Cellular Automata





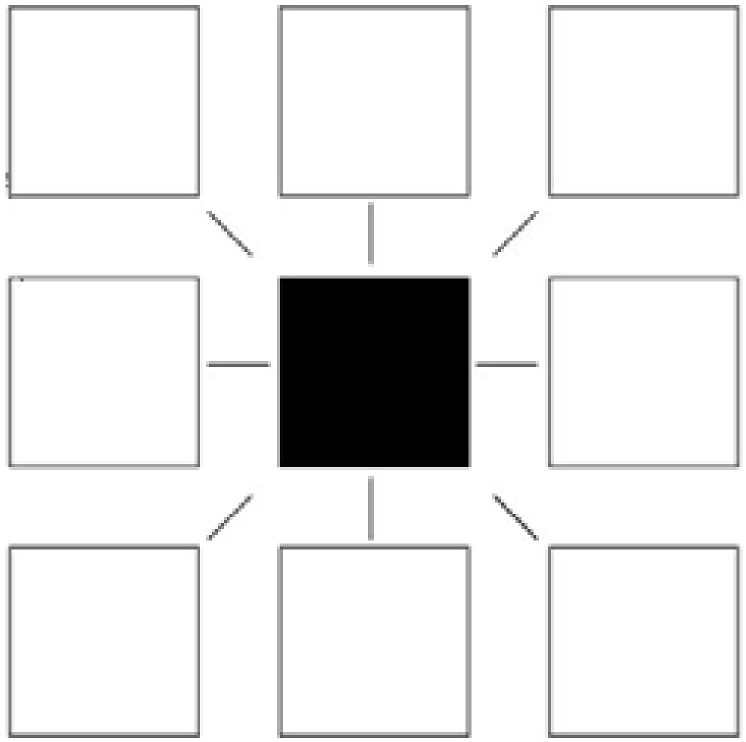
John Von Neumann

John Horton Conway

Stephen Wolfram

Cellular Automata

A cell’s neighborhood is defined by the eight closest adjacent cells, including cells that touch with one side or with one corner.



Cellular Automata

A simple set of rules is created, determining how each generation of the automata is generated. The first generation is arbitralirily seeded with a number of black cells.

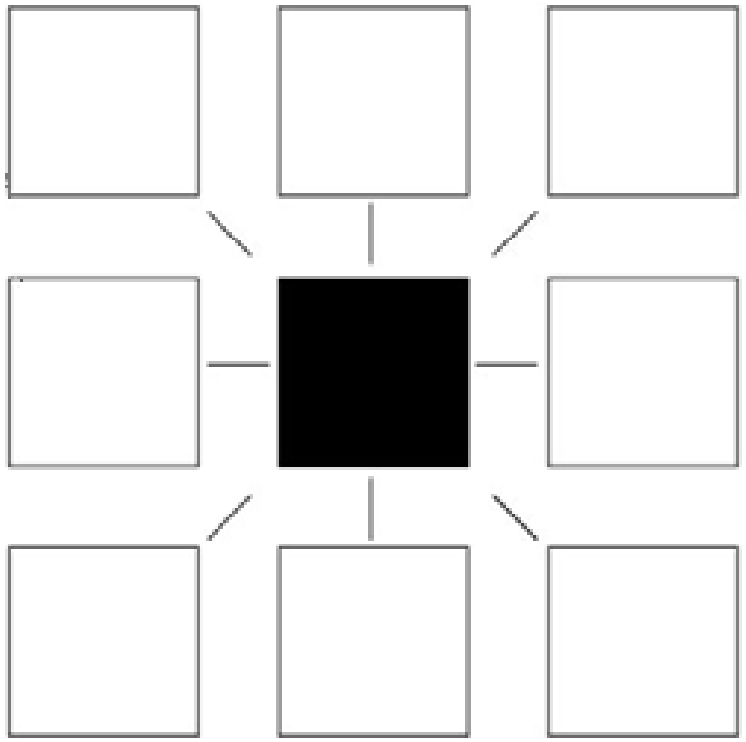
Cells can only be black or white (on/off, alive/dead) in these simplest automata.

The rules state whether a cell turns white, turns black, or remains the same.

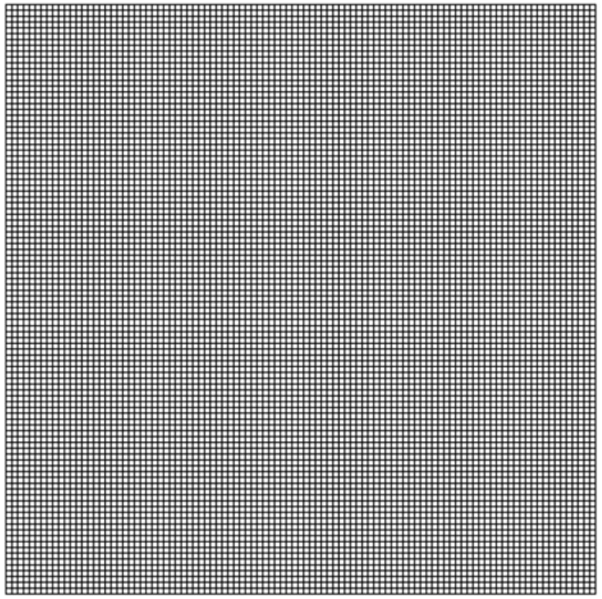
Each cell checks its neighbors and its own state at the same time, then adjusts its color for the next generation. All cells change colors concurrently, one generation at a time.

For example our rule set could be “turn black if the majority of cells in your neighborhood are black, otherwise turn white.”

A ridiculously large number of possible rule sets exist for this type of 2-dimensional cellular automata (2^{512}, or 1.3 x $10^{154)}$.



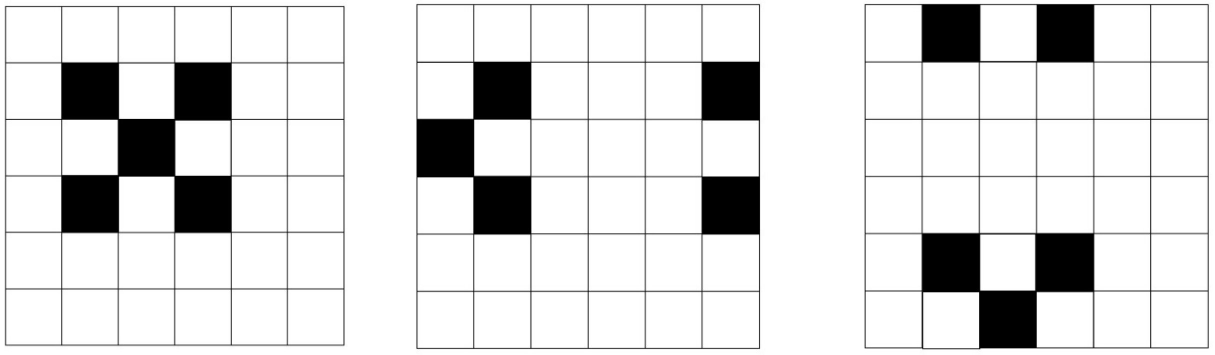
Cellular Automata



In theory the grid is extended infinitely. For most of our purposes we can simply use an arbitrarily large grid.

If space doesn't allow for an infinite grid, we can have the grid be "circular."

A 3x3 neighborhood starting on the left edge of the grid finishes on the right side of the grid. A neighborhood starting on the top of the grid extends to the bottom.



Cellular Automata

A simple set of rules is created, determining how each generation of the automata is generated. The first generation is arbitrarily seeded with a number of black cells.

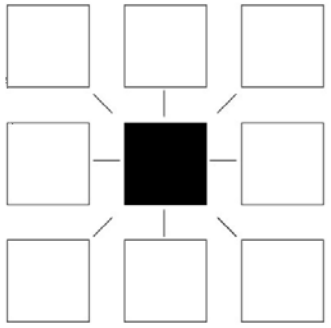
Cells can only be black or white (on/off, alive/dead) in these simplest automata.

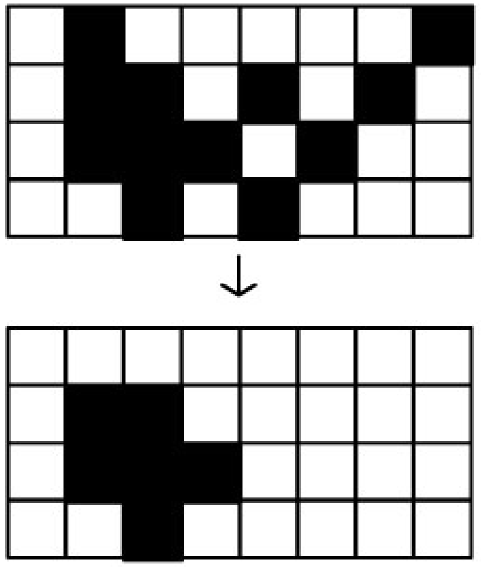
The rules state whether a cell turns white, turns black, or remains the same.

Each cell checks its neighbors and its own state at the same time, then adjusts its color for the next generation. All cells change colors concurrently, one generation at a time.

For example our rule set could be "turn black if the majority of cells in your neighborhood are black, otherwise turn white."

A ridiculously large number of possible rule sets exist for this type of 2-dimensional cellular automata (2^{512}, or 1.3×10^{154}).





Two generations of a cellular automata with the rule being "take on whichever state is a majority in my local neighborhood." The first state was arbitrarily chosen.

Conway’s Game of Life

Conway’s Game of Life Rules:

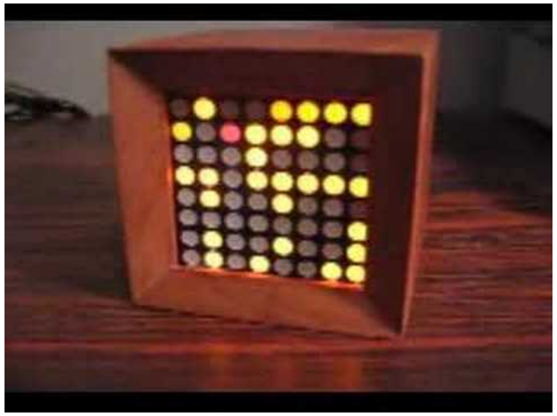
- If an alive cell has two or three alive neighboring cells, it stays alive (turns black).
- If a dead cell (white) has exactly three alive neighboring cells, it comes to life.
- Otherwise, the cell will stay dead or die (analogous to over or under population).

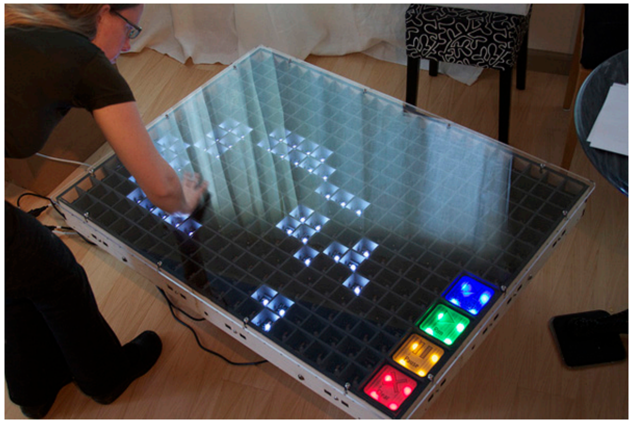
Conway originally worked out this game on a Go board, methodically updating each generation while checking for errors.

Conway’s Game of Life

Conway’s Game of Life Rules:

- If an alive cell has two or three alive neighboring cells, it stays alive (turns black).
- If a dead cell (white) has exactly three alive neighboring cells, it comes to life.
- Otherwise, the cell will stay dead or die (analogous to over or under population).



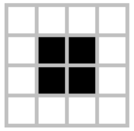


Conway's Game of Life

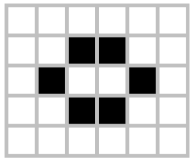
Conway's Game of Life Rules:

- If an alive cell has two or three alive neighboring cells, it stays alive (turns black).
- If a dead cell (white) has exactly three alive neighboring cells, it comes to life.
- Otherwise, the cell will stay dead or die (analogous to over or under population).

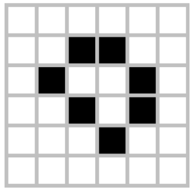
Block



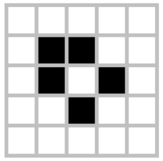
Beehive



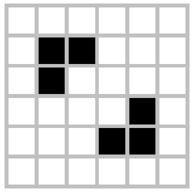
Loaf



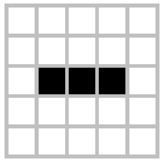
Boat



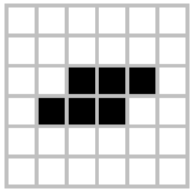
Beacon



Blinker



Toad

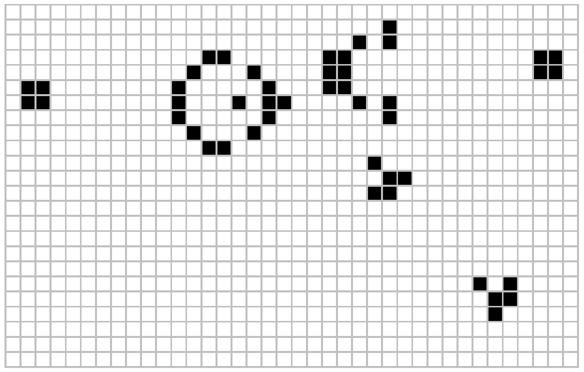


Conway’s Game of Life

Conway’s Game of Life Rules:

- If an alive cell has two or three alive neighboring cells, it stays alive (turns black).
- If a dead cell (white) has exactly three alive neighboring cells, it comes to life.
- Otherwise, the cell will stay dead or die (analogous to over or under population).

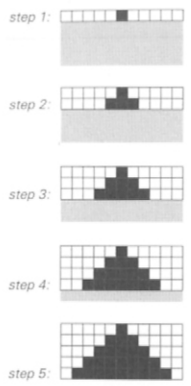
Conway believed that there would be no initial arrangements of cells that would lead to unrestrained growth and offered a $50 prize to anyone who could demonstrate such an arrangement. The “Gosper Glider Gun” took the prize that same year.



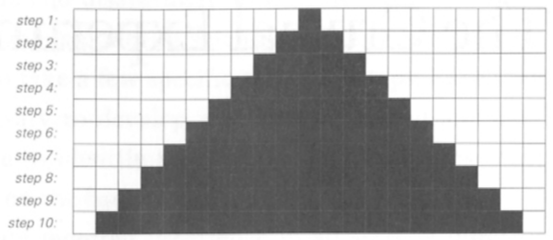
It turns out there are many arrangements of cells that offer unrestrained growth, complex period oscillations (blinkers), and “methuselahs” that generate complex patterns but then die out or settle into fixed states of no change.

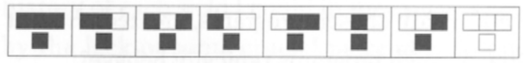
One Dimensional Elementary Automata

The possibilities for two dimensional automata are virtually without end, especially if one begins to add additional states. For example one could have several different colors instead of black and white. However, we can also look at even simpler cellular automata that have just one dimension.



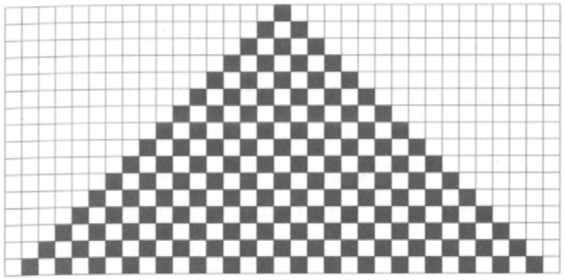
A visual representation of the behavior of a cellular automaton, with each row of cells corresponding to one step. At the first step the cell in the center is black and all other cells are white. Then on each successive step, a particular cell is made black whenever it or either of its neighbors were black on the step before. As the picture shows, this leads to a simple expanding pattern uniformly filled with black.

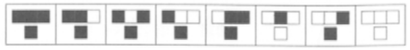




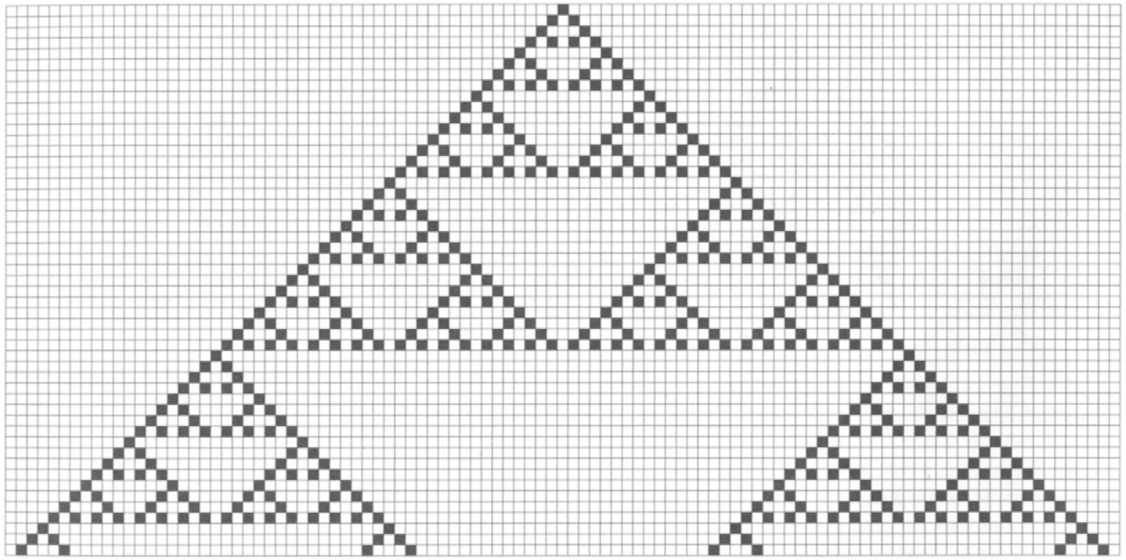
A representation of the rule for the cellular automaton shown above. The top row in each box gives one of the possible combinations of colors for a cell and its immediate neighbors. The bottom row then specifies what color the center cell should be on the next step in each of these cases. In the numbering scheme described in Chapter 3, this is cellular automaton rule 254.

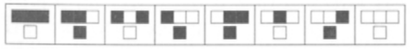
One Dimensional Elementary Automata



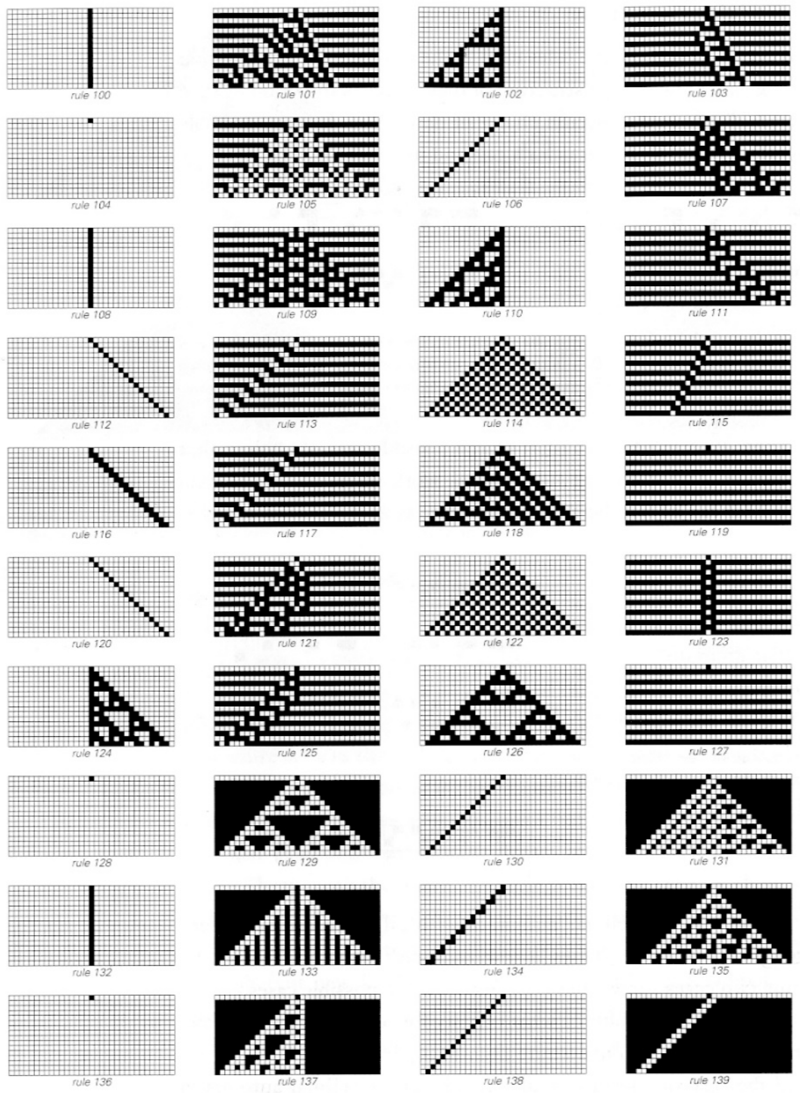


A cellular automaton with a slightly different rule. The rule makes a particular cell black if either of its neighbors was black on the step before, and makes the cell white if both its neighbors were white. Starting from a single black cell, this rule leads to a checkerboard pattern. In the numbering scheme of Chapter 3, this is cellular automaton rule 250.

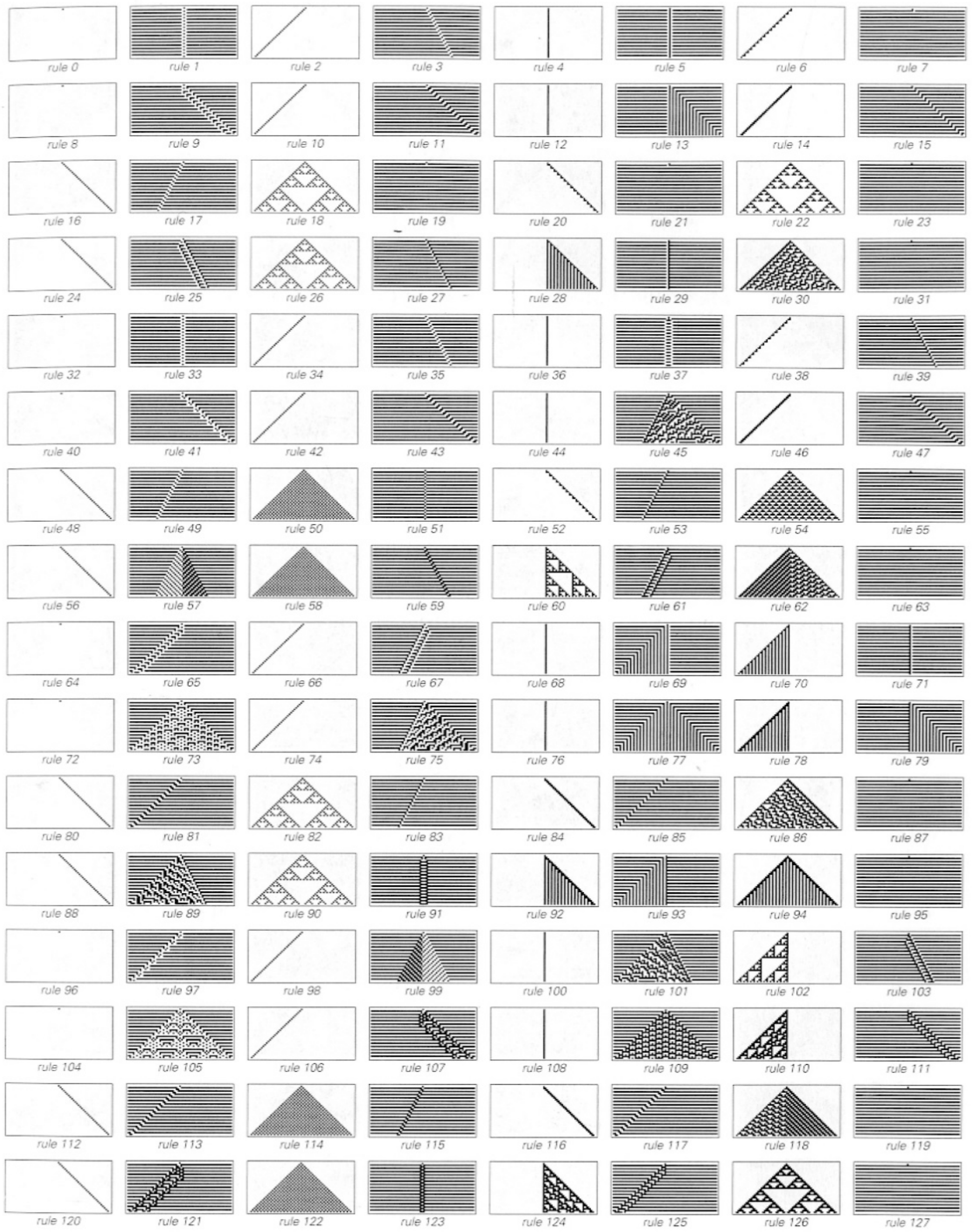


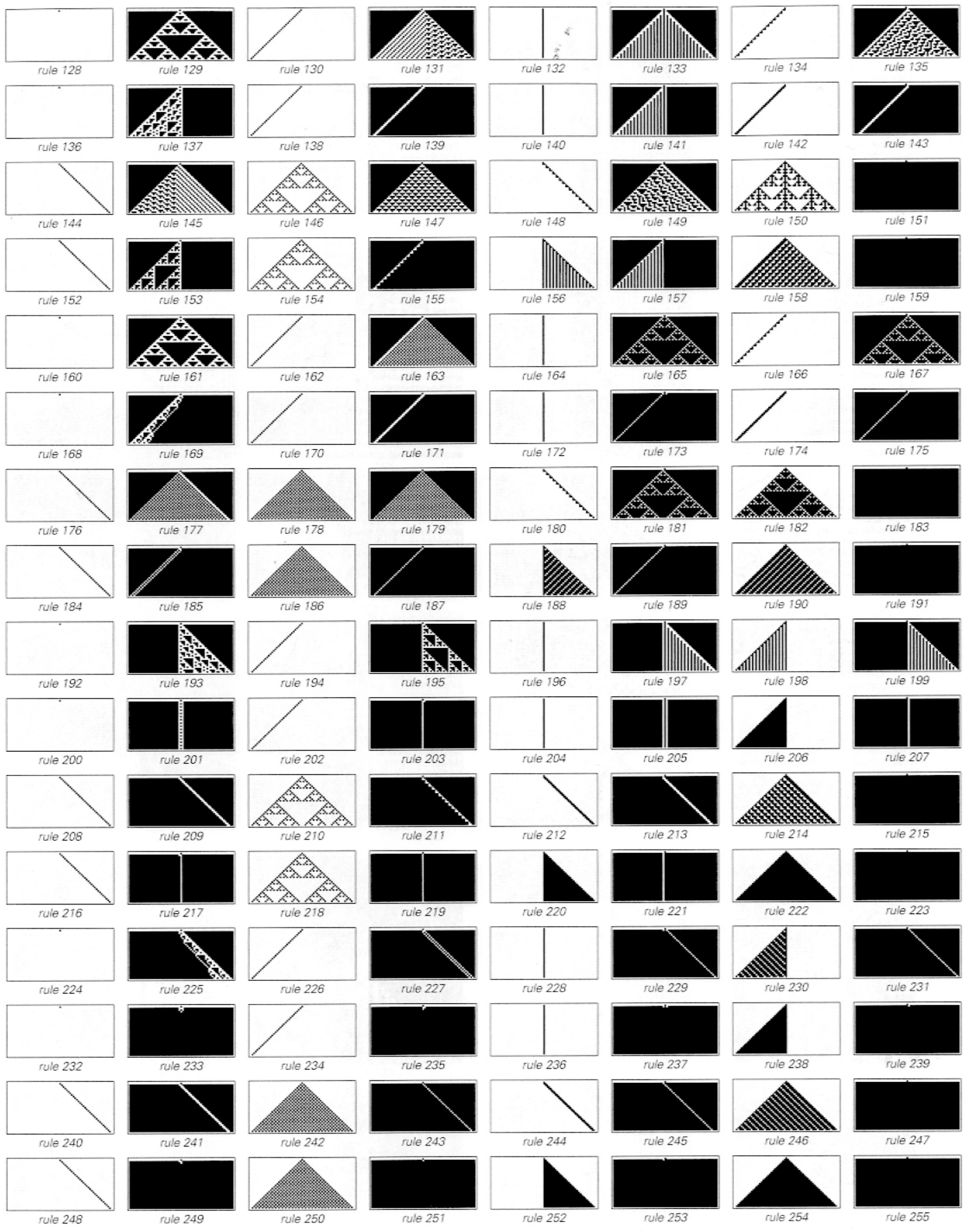


A cellular automaton that produces an intricate nested pattern. The rule in this case is that a cell should be black whenever one or the other, but not both, of its neighbors were black on the step before. Even though the rule is very simple, the picture shows that the overall pattern obtained over the course of 50 steps starting from a single black cell is not so simple. The particular rule used here can be described by the formula $a_i' = Mod[a_{i-1} + a_{i+1}, 2]$. In the numbering scheme of Chapter 3, it is cellular automaton rule 90.

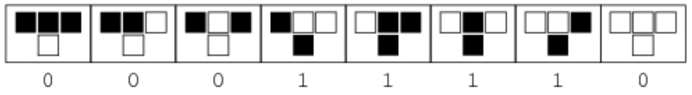


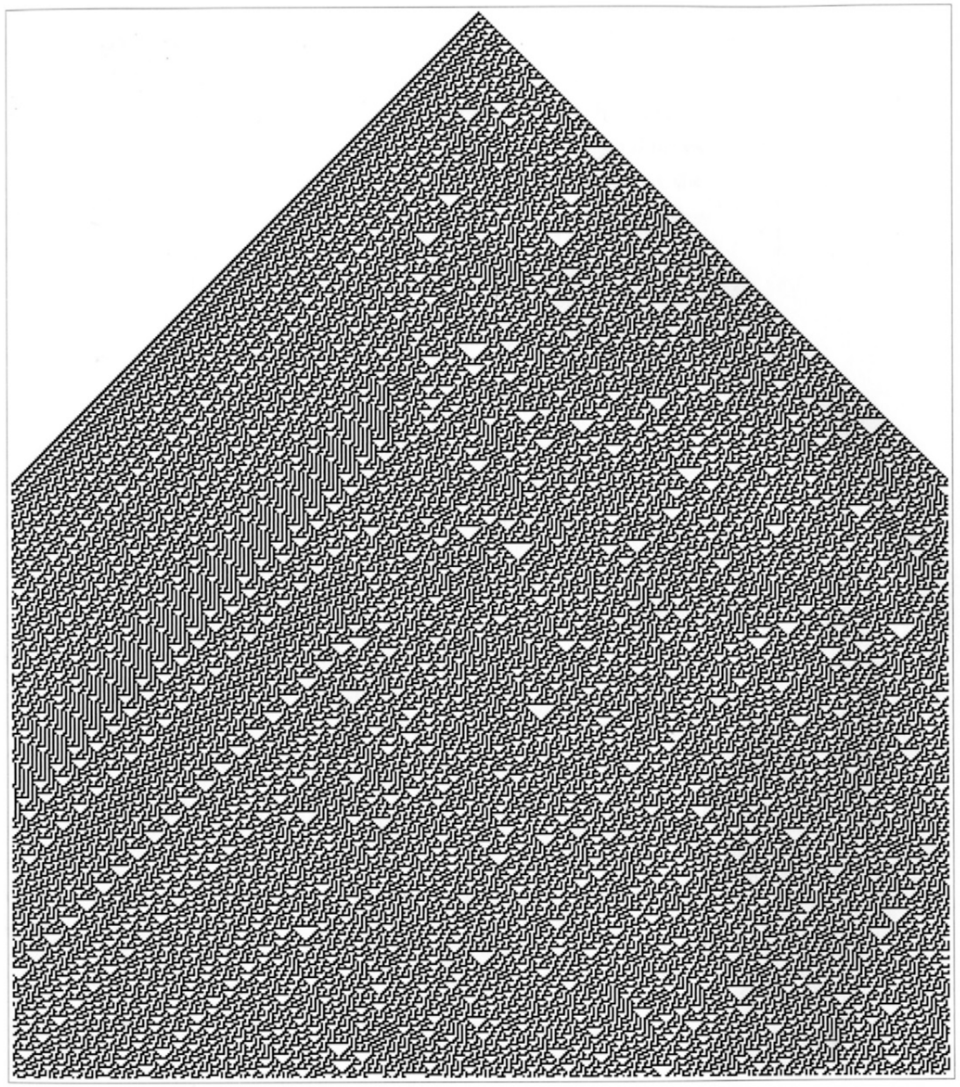
Evolution of cellular automata with a sequence of different possible rules, starting in all cases from a single black cell.


rule 0
rule 1
rule 2
rule 3
rule 4
rule 5
rule 6
rule 7
rule 8
rule 9
rule 10
rule 11
rule 12
rule 13
rule 14
rule 15
rule 16
rule 17
rule 18
rule 19
rule 20
rule 21
rule 22
rule 23
rule 24
rule 25
rule 26
rule 27
rule 28
rule 29
rule 30
rule 31
rule 32
rule 33
rule 34
rule 35
rule 36
rule 37
rule 38
rule 39
rule 40
rule 41
rule 42
rule 43
rule 44
rule 45
rule 46
rule 47
rule 48
rule 49
rule 50
rule 51
rule 52
rule 53
rule 54
rule 55
rule 56
rule 57
rule 58
rule 59
rule 60
rule 61
rule 62
rule 63
rule 64
rule 65
rule 66
rule 67
rule 68
rule 69
rule 70
rule 71
rule 72
rule 73
rule 74
rule 75
rule 76
rule 77
rule 78
rule 79
rule 80
rule 81
rule 82
rule 83
rule 84
rule 85
rule 86
rule 87
rule 88
rule 89
rule 90
rule 91
rule 92
rule 93
rule 94
rule 95
rule 96
rule 97
rule 98
rule 99
rule 100
rule 101
rule 102
rule 103
rule 104
rule 105
rule 106
rule 107
rule 108
rule 109
rule 110
rule 111
rule 112
rule 113
rule 114
rule 115
rule 116
rule 117
rule 118
rule 119
rule 120
rule 121
rule 122
rule 123
rule 124
rule 125
rule 126
rule 127

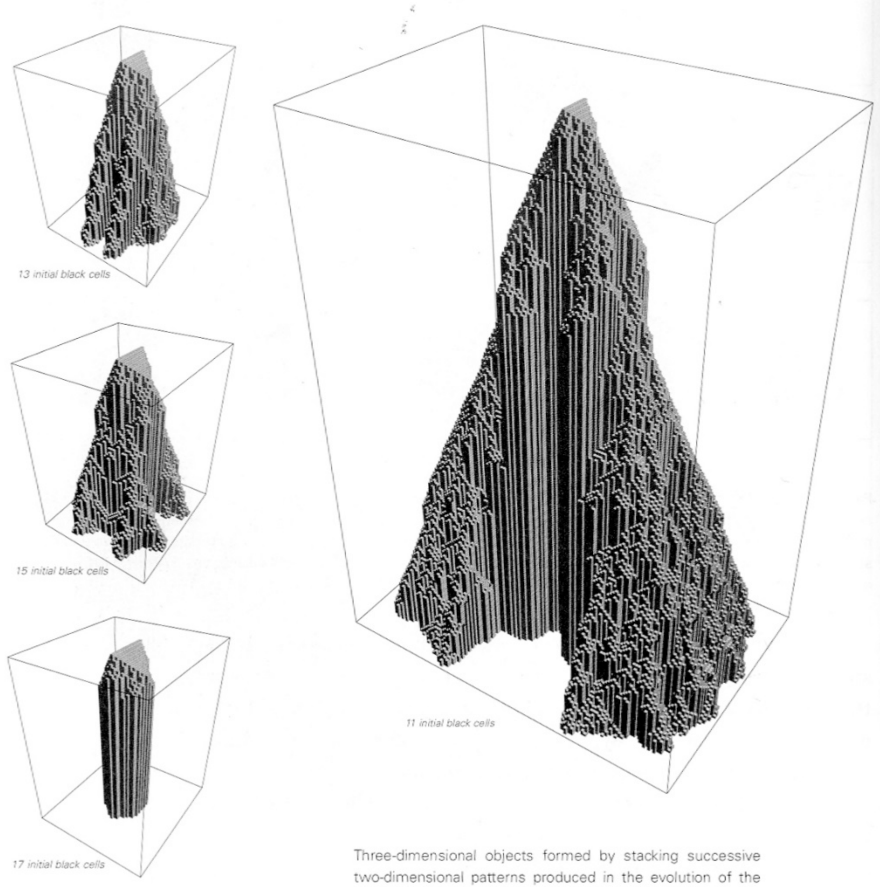

rule 128
rule 129
rule 130
rule 131
rule 132
rule 133
rule 134
rule 135
rule 136
rule 137
rule 138
rule 139
rule 140
rule 141
rule 142
rule 143
rule 144
rule 145
rule 146
rule 147
rule 148
rule 149
rule 150
rule 151
rule 152
rule 153
rule 154
rule 155
rule 156
rule 157
rule 158
rule 159
rule 160
rule 161
rule 162
rule 163
rule 164
rule 165
rule 166
rule 167
rule 168
rule 169
rule 170
rule 171
rule 172
rule 173
rule 174
rule 175
rule 176
rule 177
rule 178
rule 179
rule 180
rule 181
rule 182
rule 183
rule 184
rule 185
rule 186
rule 187
rule 188
rule 189
rule 190
rule 191
rule 192
rule 193
rule 194
rule 195
rule 196
rule 197
rule 198
rule 199
rule 200
rule 201
rule 202
rule 203
rule 204
rule 205
rule 206
rule 207
rule 208
rule 209
rule 210
rule 211
rule 212
rule 213
rule 214
rule 215
rule 216
rule 217
rule 218
rule 219
rule 220
rule 221
rule 222
rule 223
rule 224
rule 225
rule 226
rule 227
rule 228
rule 229
rule 230
rule 231
rule 232
rule 233
rule 234
rule 235
rule 236
rule 237
rule 238
rule 239
rule 240
rule 241
rule 242
rule 243
rule 244
rule 245
rule 246
rule 247
rule 248
rule 249
rule 250
rule 251
rule 252
rule 253
rule 254
rule 255

rule 30



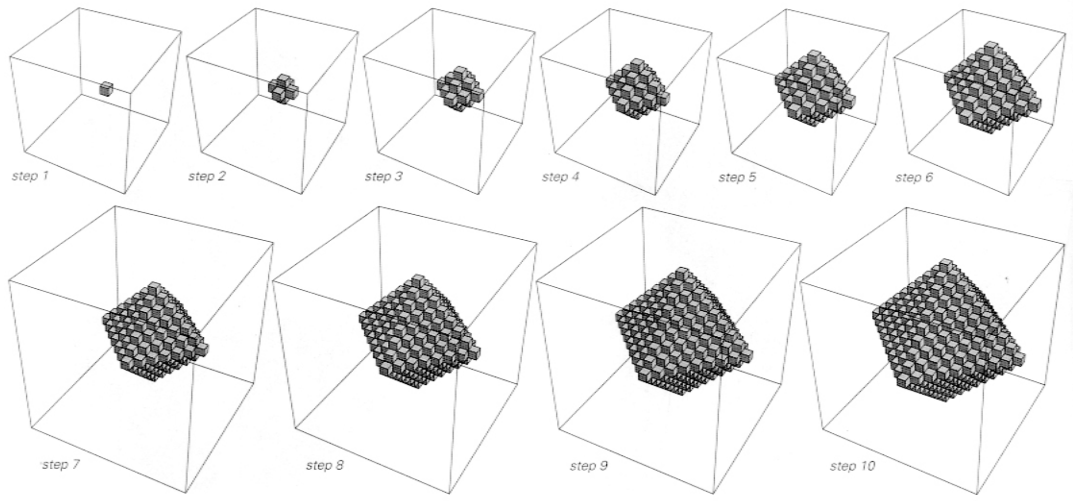

Five hundred steps in the evolution of the rule 30 cellular automaton from page 27. The pattern produced continues to expand on both left and right, but only the part that fits across the page is shown here. The asymmetry between the left and right-hand sides is a direct consequence of asymmetry that exists in the particular underlying cellular automaton rule used.

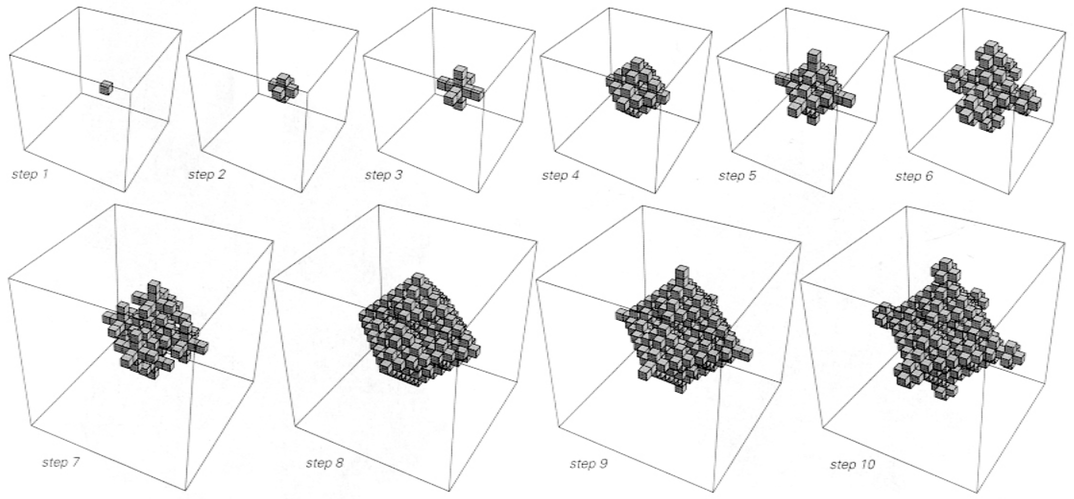
Two-Dimensional Cellular Automata Projected in Three Dimensions

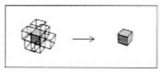


Three-dimensional objects formed by stacking successive two-dimensional patterns produced in the evolution of the cellular automaton from the previous page. The large picture on the right shows 200 steps of evolution.

Three-Dimensional
Cellular Automata

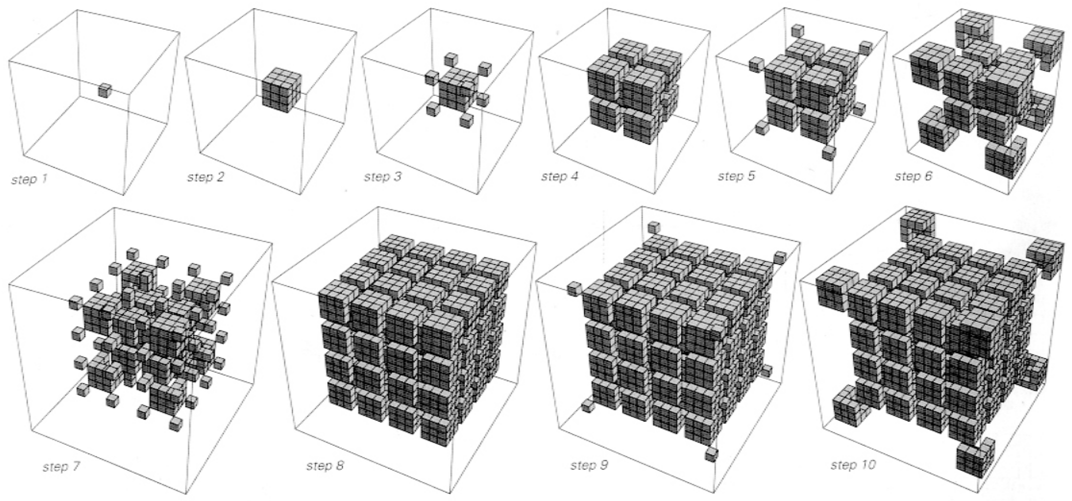


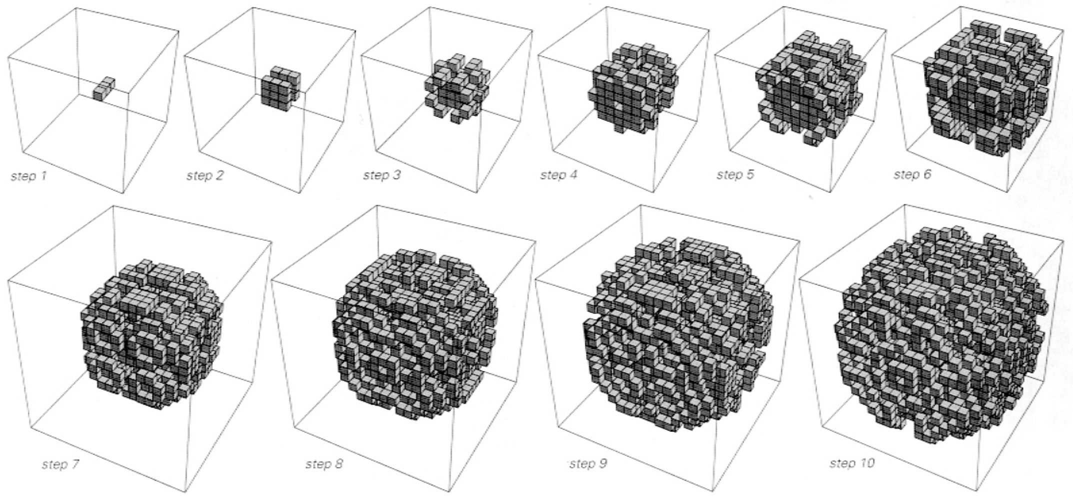


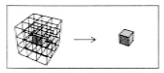


Examples of three-dimensional cellular automata. In the top set of pictures, the rule specifies that a cell should become black whenever any of the six neighbors with which it shares a face were black on the step before. In the bottom pictures, the rule specifies that a cell should become black only when exactly one of its six neighbors was black on the step before. In both cases, the initial condition contains a single black cell. In the top pictures, the limiting shape obtained is a regular octahedron. In the bottom pictures, it is a nested pattern analogous to the two-dimensional one on page 171.

Three-Dimensional Cellular Automata

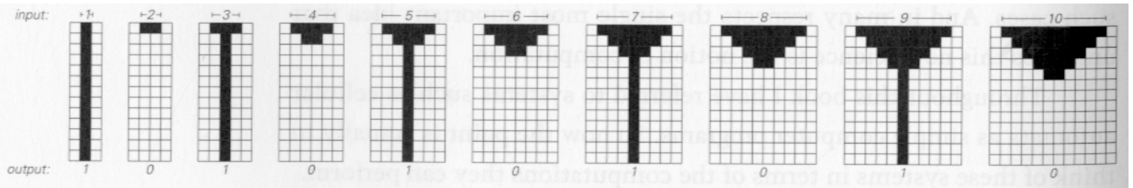


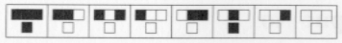




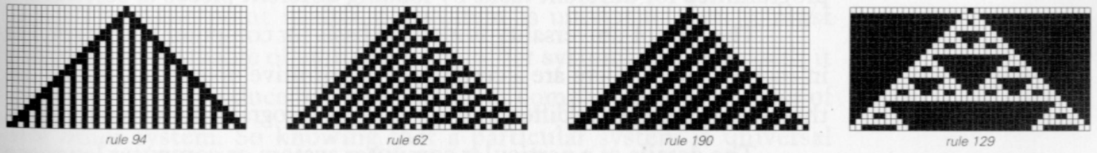
Further examples of three-dimensional cellular automata, but now with rules that depend on all 26 neighbors that share either a face or a corner with a particular cell. In the top pictures, the rule specifies that a cell should become black when exactly one of its 26 neighbors was black on the step before. In the bottom pictures, the rule specifies that a cell should become black only when exactly two of its 26 neighbors were black on the step before. In the top pictures, the initial condition contains a single black cell; in the bottom pictures, it contains a line of three black cells.

Simple Computations

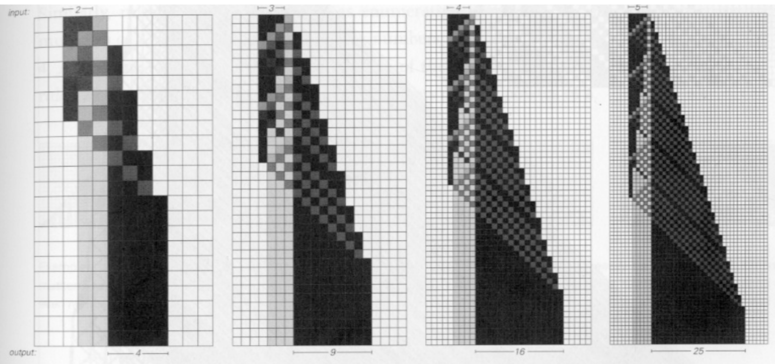




A simple cellular automaton whose evolution effectively computes the remainder after division of a number by 2. Starting from a row of *n* black cells, 0 black cells survive if *n* is even, and 1 black cell survives if *n* is odd. The cellular automaton follows elementary rule 132, as shown on the left.

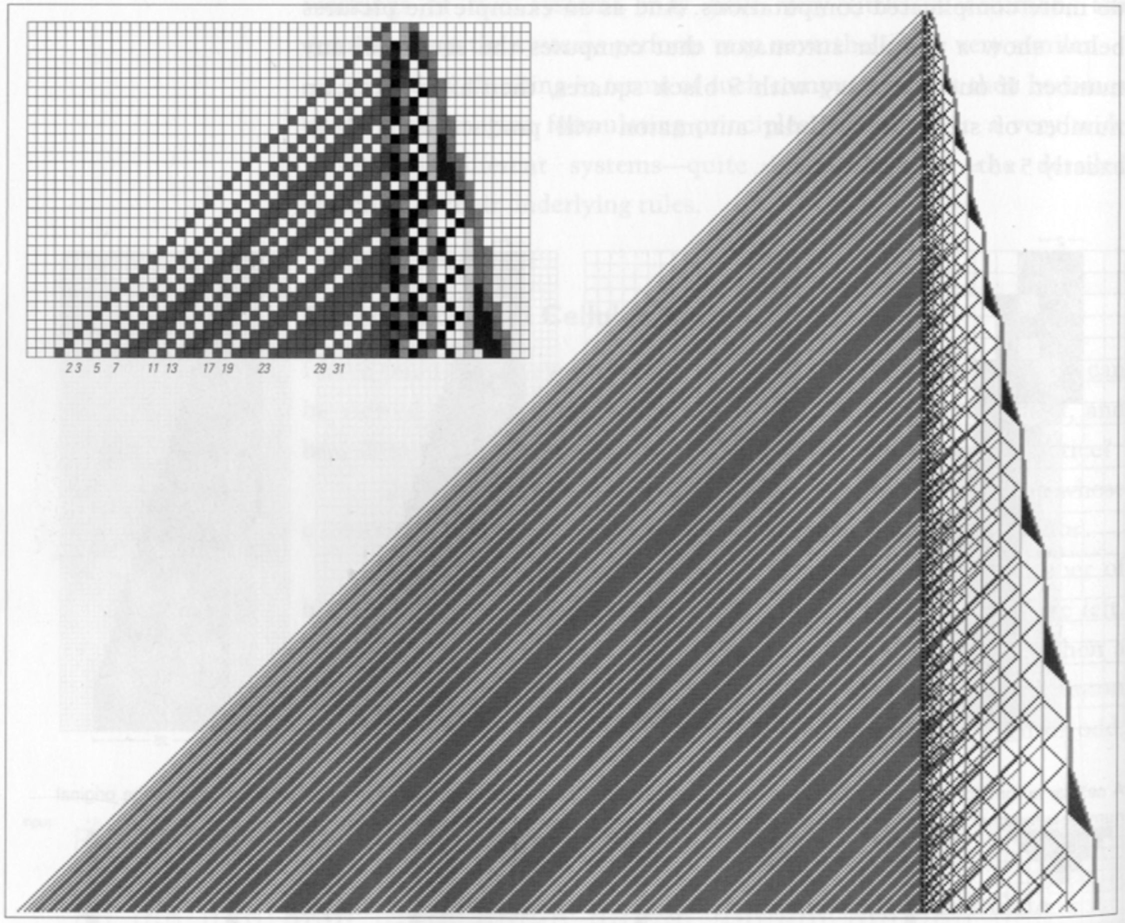


Examples of simple cellular automata whose evolution corresponds to computations that can easily be described in traditional mathematical terms. In analogy to the previous page, the positions of white cells at the bottom of the rule 94 picture correspond to even numbers, on the left in rule 62 to multiples of 3, in rule 190 to multiples of 4, and in the center column of rule 129 to powers of 2.



A cellular automaton that computes the square of any number. The cellular automaton effectively works by adding the original number *n* together *n* times. The underlying rule used here involves eight possible colors for each cell.

A slightly more complex computation

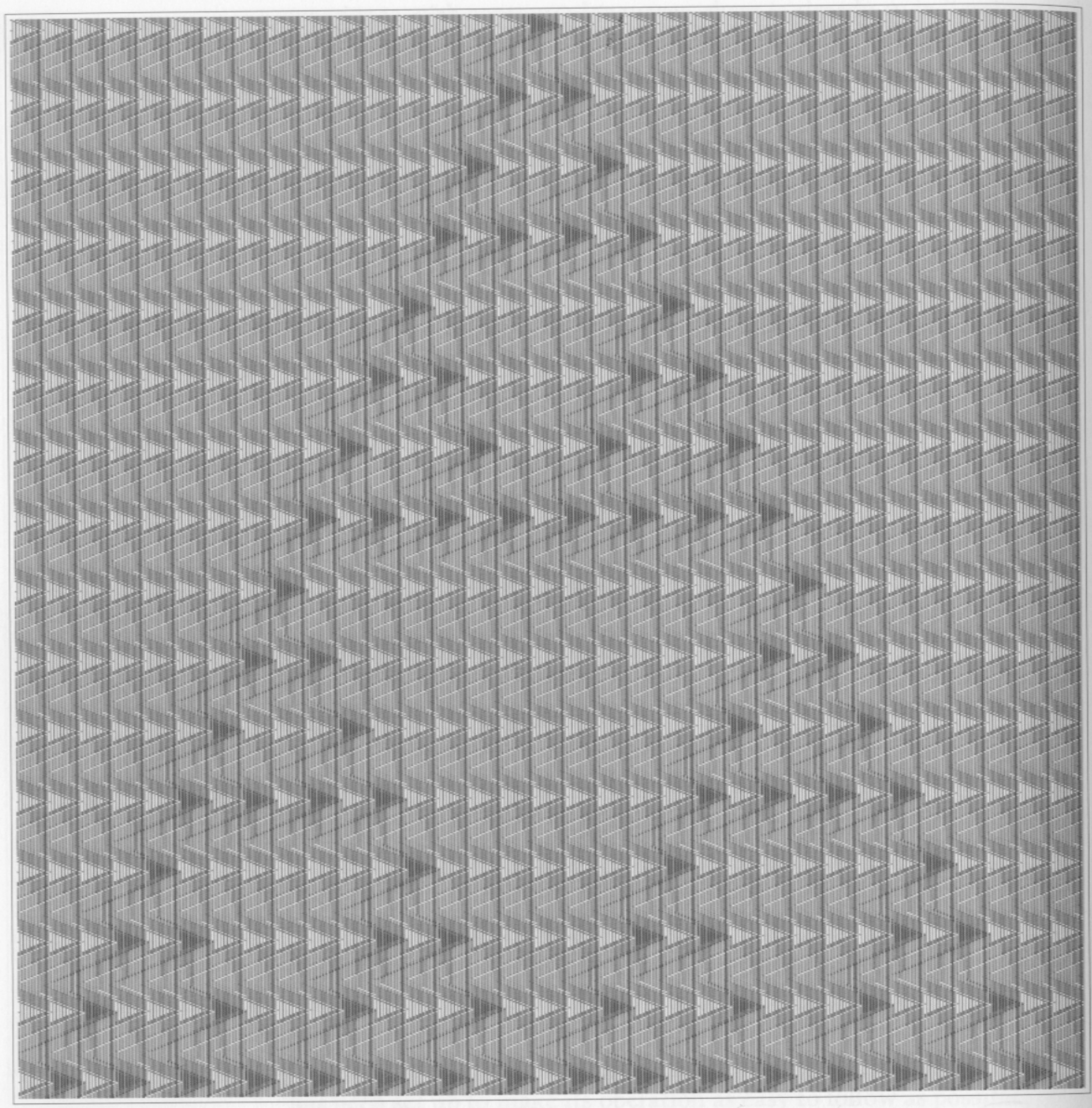


A cellular automaton constructed to compute the prime numbers. The system generates a dark gray stripe on the left at all positions that correspond to any product of numbers other than 1. White gaps then remain at positions that correspond to the prime numbers 2, 3, 5, 7, 11, 13, 17, etc. The cellular automaton effectively does its computation using the standard sieve of Eratosthenes method. The structures on the right bounce backwards and forwards with repetition periods corresponding to successive odd numbers. Once in each period they produce a gray stripe which propagates to the left, so that in the end there is a gray stripe corresponding to every multiple of every number. The rule for the cellular automaton shown here involves 16 possible colors for each cell.

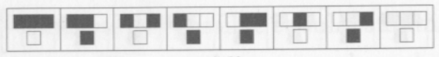
One Dimensional

Universal Cellular

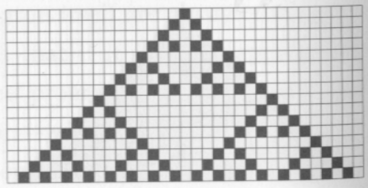
Automata



The universal cellular automaton emulating elementary rule 90. The underlying rules for the universal cellular automaton are exactly the same as on the previous page. But each block in the initial conditions now contains a representation of rule 90 rather than rule 254.



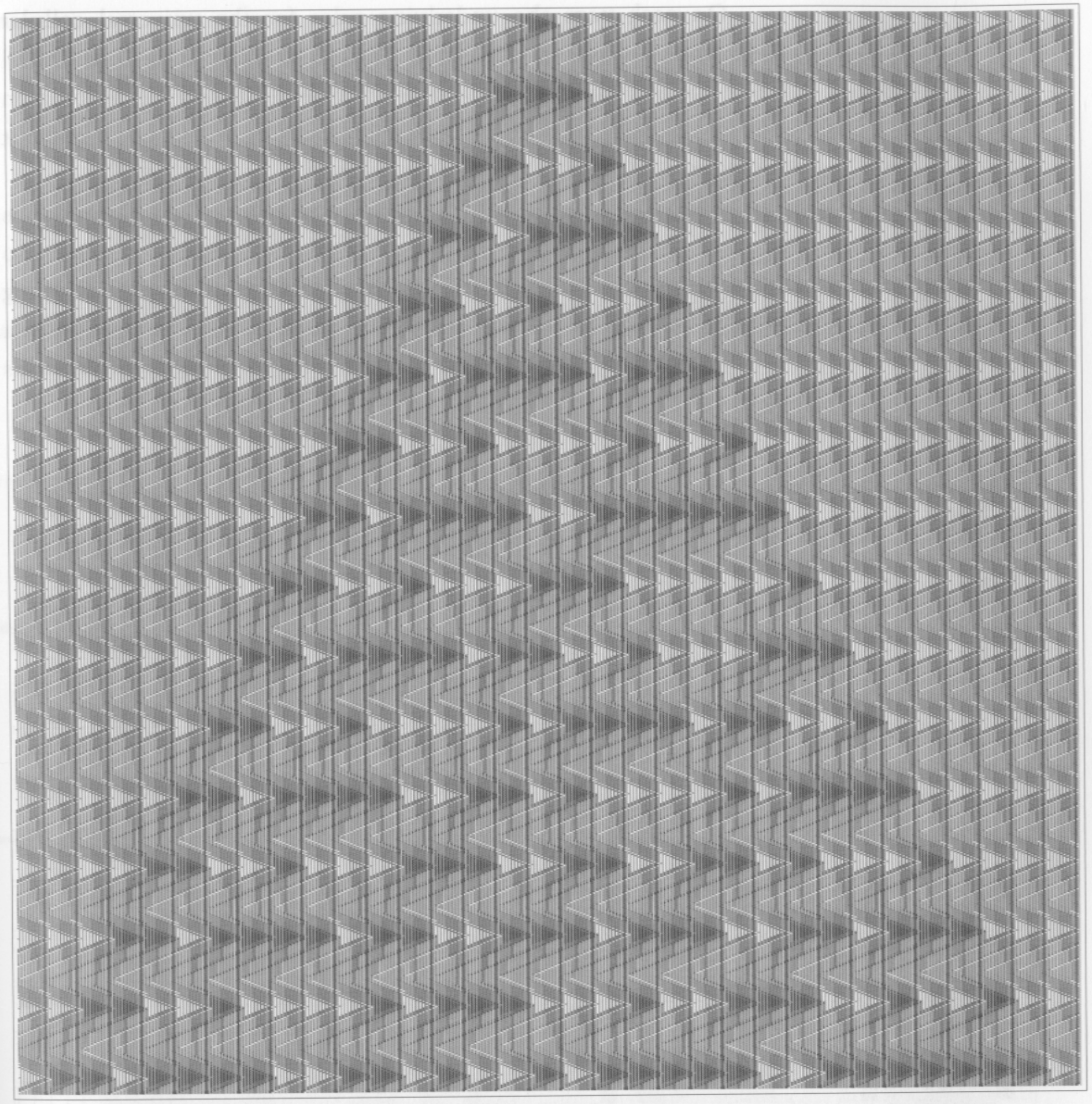
rule 90



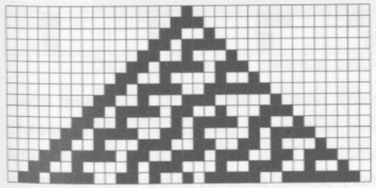
One Dimensional

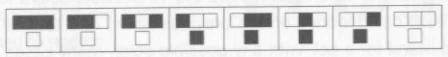
Universal Cellular

Automata

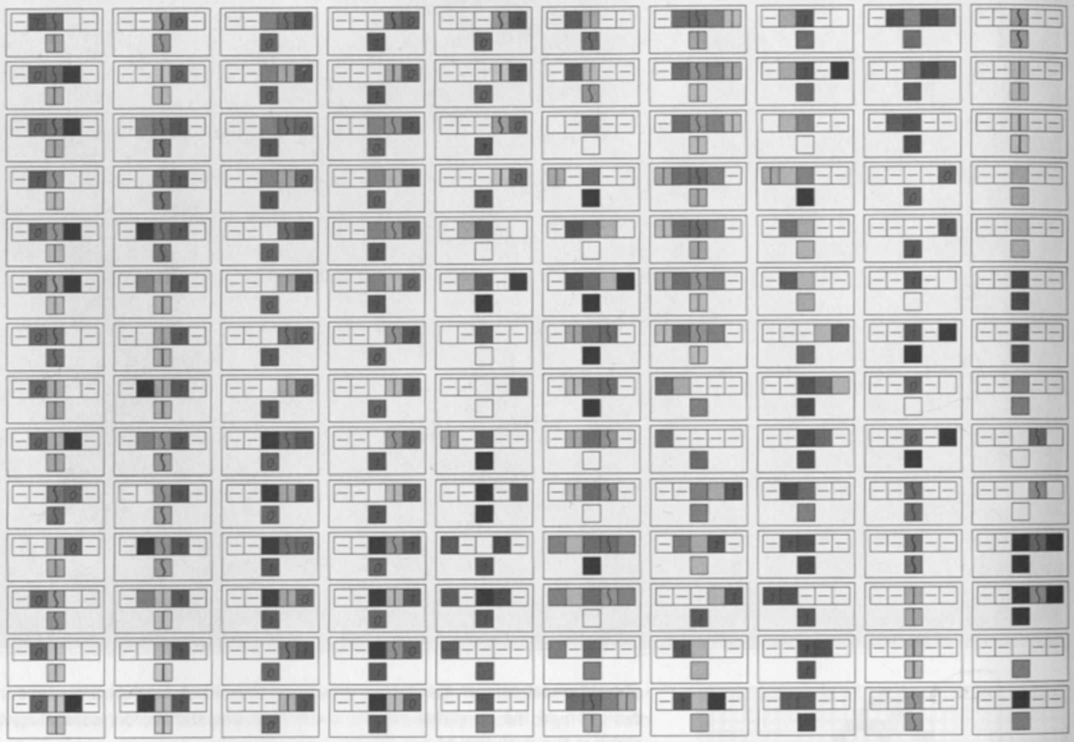


The universal cellular automaton emulating rule 30. A total of 848 steps in the evolution of the universal cellular automaton are shown, corresponding to 16 steps in the evolution of rule 30.





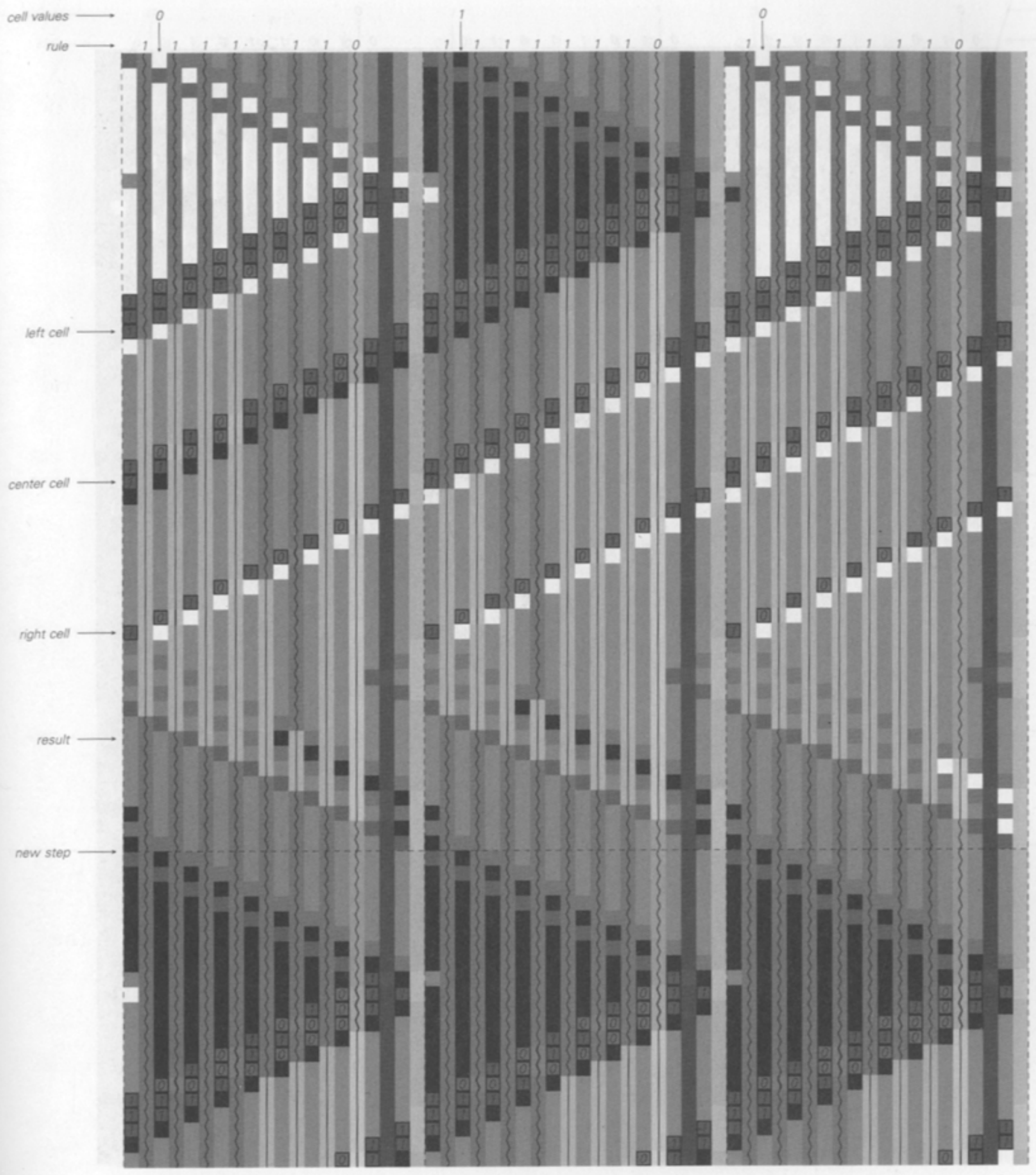
Universal Cellular Automata



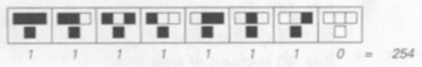
The rules for the universal cellular automaton. There are 19 possible colors for each cell, represented here by 19 different icons. Since the new color of each cell depends on the previous colors of a total of five cells, there are in principle 2,476,099 cases to cover. But by using ⊟ to stand for a cell with any possible color, many cases are combined. Note that the cases shown are in a definite order reading down successive columns, with special cases given before more general ones. With the initial conditions used, there are some combinations of cells that can never occur, and these are not covered in the rules shown.

Universal Cellular

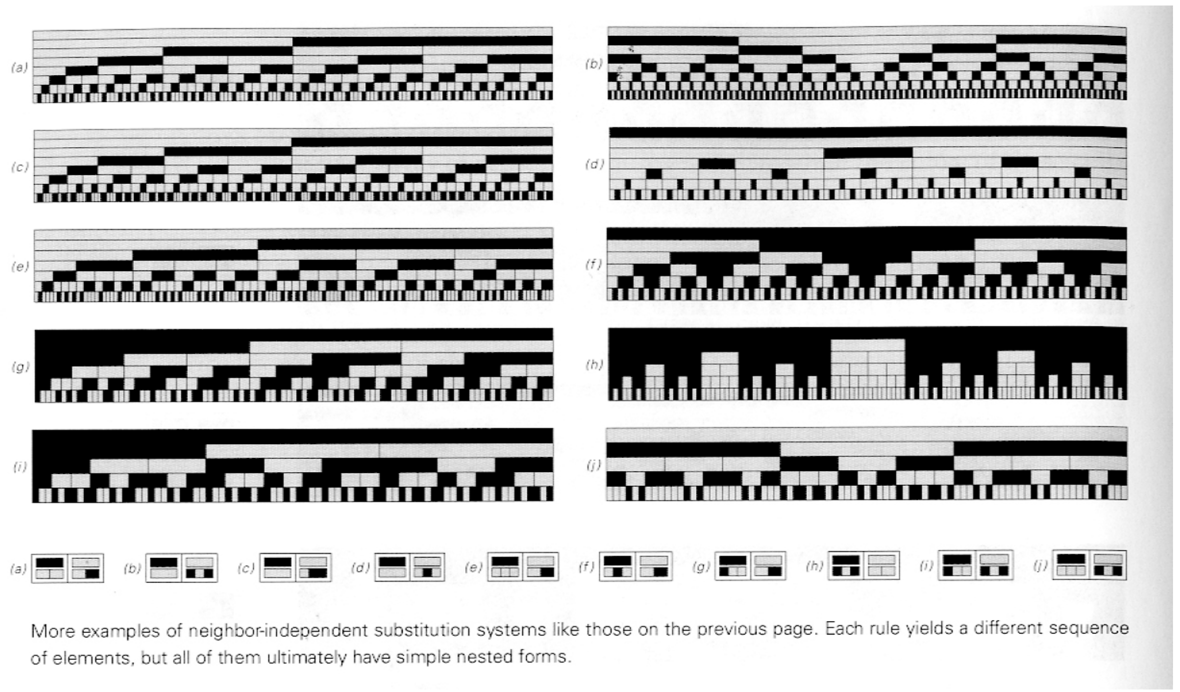
Automata



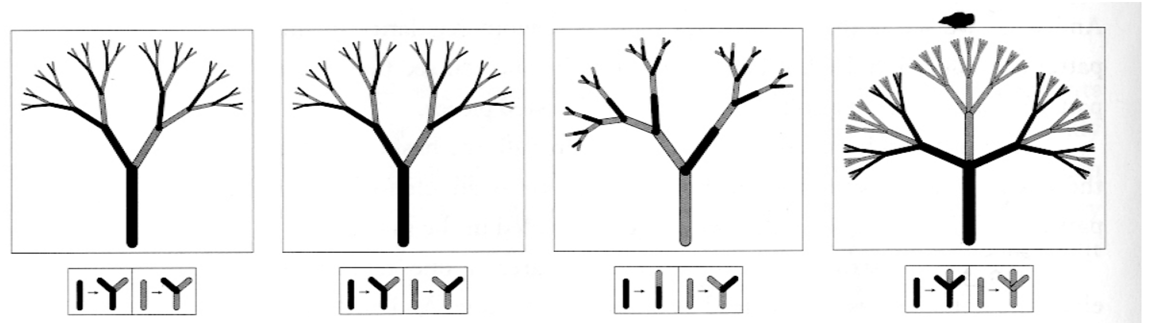
Details of how the universal cellular automaton emulates rule 254. Each of the blocks in the universal cellular automaton represents a single cell in rule 254, and encodes both the current color of the cell and the form of the rule used to update it.



Wolfram Investigates Other Simple Systems

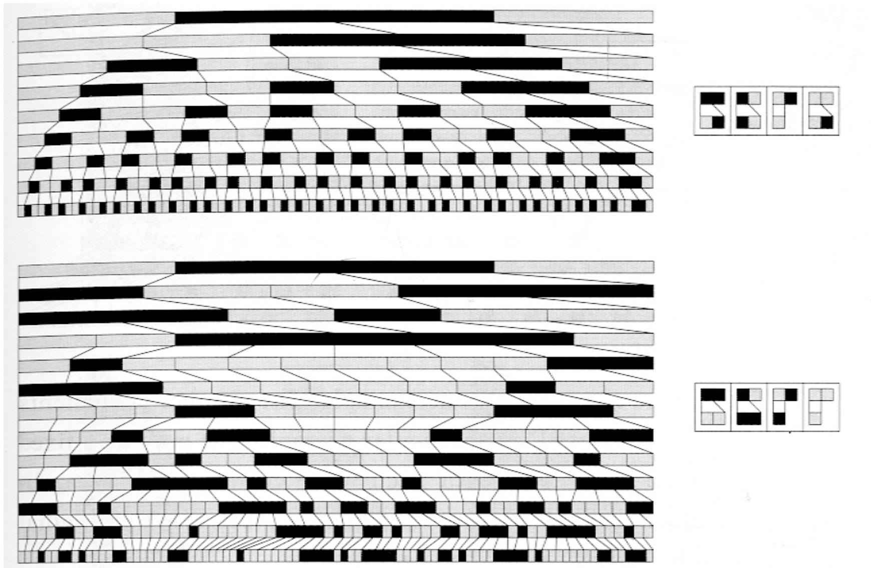


More examples of neighbor-independent substitution systems like those on the previous page. Each rule yields a different sequence of elements, but all of them ultimately have simple nested forms.

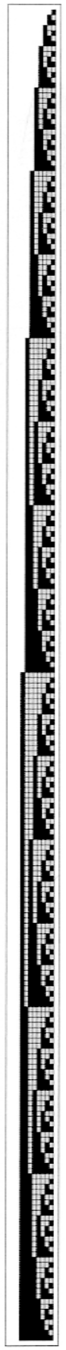


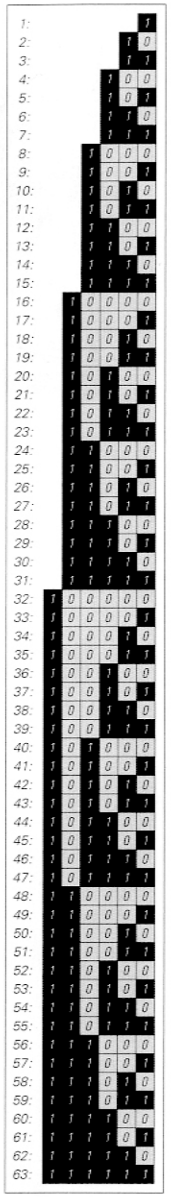
The evolution of the same substitution systems as on the previous page, but now shown in terms of trees. Starting from the trunk at the bottom, the rules specify that at each step every branch of a particular color should split into smaller branches in the same way. The result is that each tree consists of a collection of progressively smaller subtrees with the same structure. On page 400 I will use similar systems to discuss the growth of actual trees and leaves.

Wolfram Investigates Other Simple Systems


Examples of substitution systems whose rules depend not just on the color of an element itself, but also on the color of the element immediately to its right. Rules of this kind cannot readily be interpreted in terms of simple subdivision of one element into several. And as a result, there is no obvious way to choose what size of box should be used to represent each element in the picture. What I do here is simply to divide the whole width of the picture equally among all elements that appear at each step. Note that on every step the rightmost element is always dropped, since no rule is given for how to replace it.

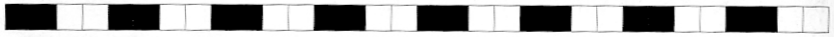
Wolfram Investigates Other Simple Systems



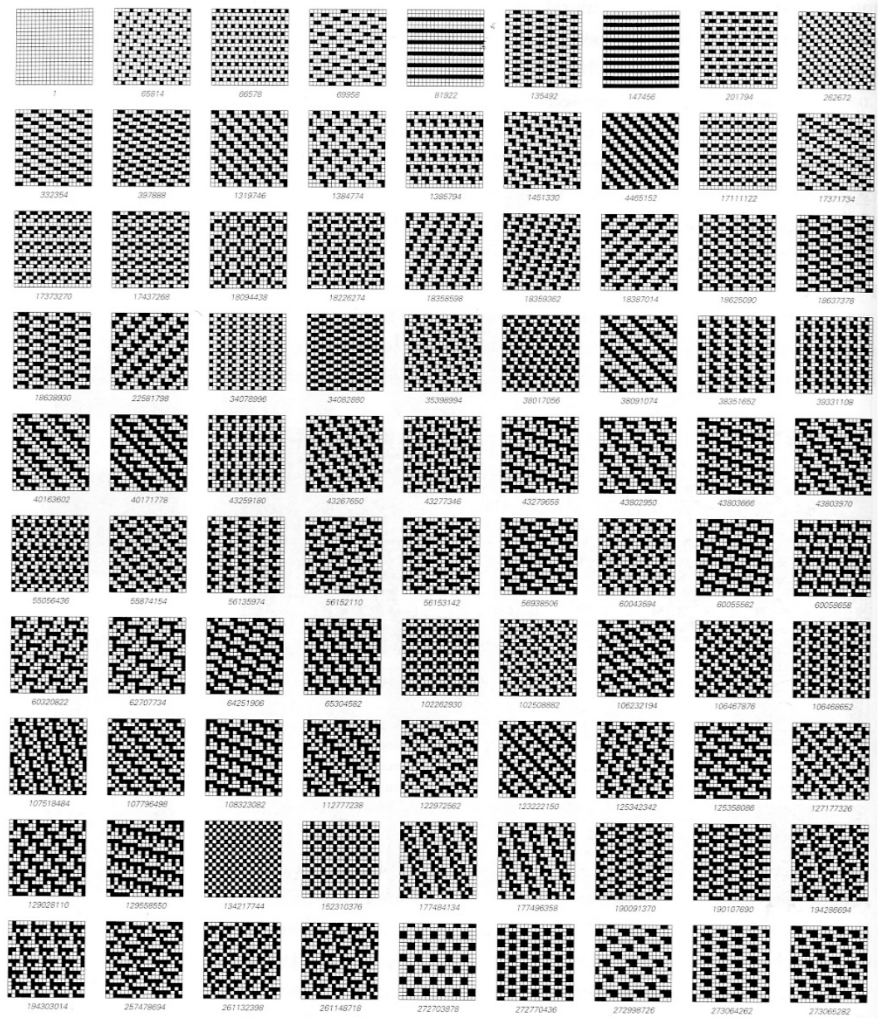


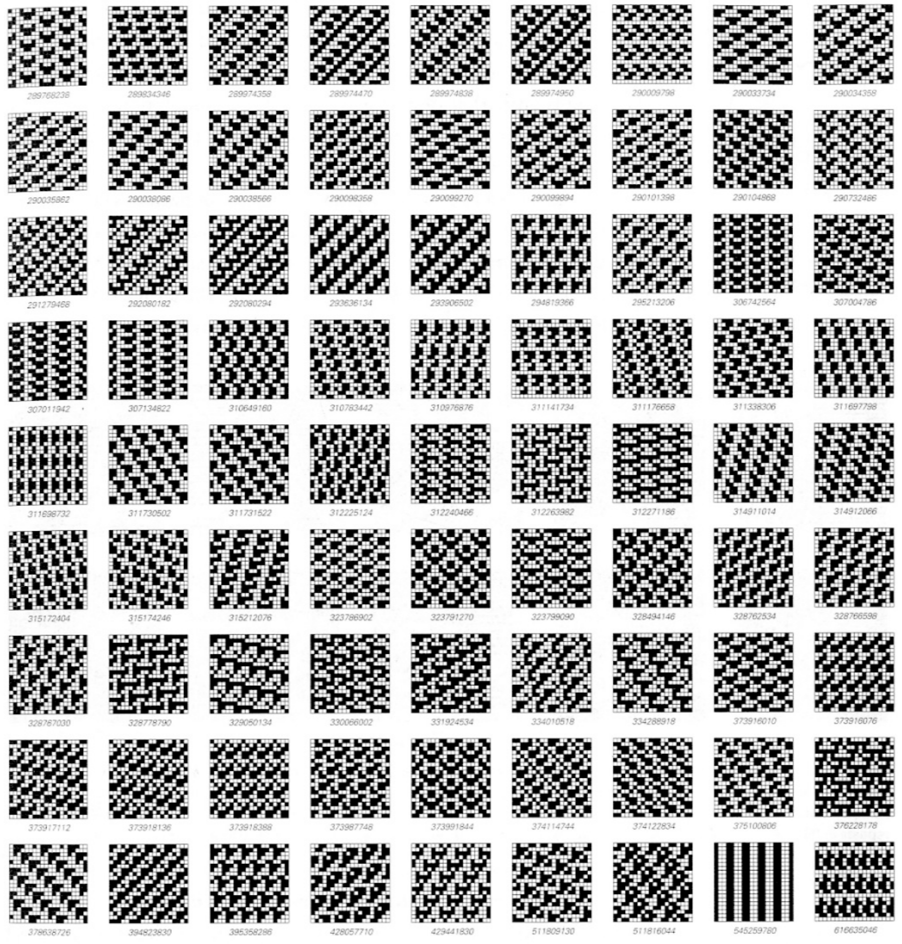
1:						1
2:					1	0
3:					1	1
4:				1	0	0
5:				1	0	1
6:				1	1	0
7:				1	1	1
8:			1	0	0	0
9:			1	0	0	1
10:			1	0	1	0
11:			1	0	1	1
12:			1	1	0	0
13:			1	1	0	1
14:			1	1	1	0
15:			1	1	1	1
16:		1	0	0	0	0
17:		1	0	0	0	1
18:		1	0	0	1	0
19:		1	0	0	1	1
20:		1	0	1	0	0
21:		1	0	1	0	1
22:		1	0	1	1	0
23:		1	0	1	1	1
24:		1	1	0	0	0
25:		1	1	0	0	1
26:		1	1	0	1	0
27:		1	1	0	1	1
28:		1	1	1	0	0
29:		1	1	1	0	1
30:		1	1	1	1	0
31:		1	1	1	1	1
32:	1	0	0	0	0	0
33:	1	0	0	0	0	1
34:	1	0	0	0	1	0
35:	1	0	0	0	1	1
36:	1	0	0	1	0	0
37:	1	0	0	1	0	1
38:	1	0	0	1	1	0
39:	1	0	0	1	1	1
40:	1	0	1	0	0	0
41:	1	0	1	0	0	1
42:	1	0	1	0	1	0
43:	1	0	1	0	1	1
44:	1	0	1	1	0	0
45:	1	0	1	1	0	1
46:	1	0	1	1	1	0
47:	1	0	1	1	1	1
48:	1	1	0	0	0	0
49:	1	1	0	0	0	1
50:	1	1	0	0	1	0
51:	1	1	0	0	1	1
52:	1	1	0	1	0	0
53:	1	1	0	1	0	1
54:	1	1	0	1	1	0
55:	1	1	0	1	1	1
56:	1	1	1	0	0	0
57:	1	1	1	0	0	1
58:	1	1	1	0	1	0
59:	1	1	1	0	1	1
60:	1	1	1	1	0	0
61:	1	1	1	1	0	1
62:	1	1	1	1	1	0
63:	1	1	1	1	1	1

Digit sequences of successive numbers written in base 2. The overall pattern has an intricate nested form.



A system consisting of a line of black and white cells whose form is defined by the constraint that every cell should have exactly one black and one white neighbor. The pattern shown is the only possible one that satisfies this constraint. The idea of implicitly determining the behavior of a system by giving constraints that it must satisfy is common in traditional science and mathematics.



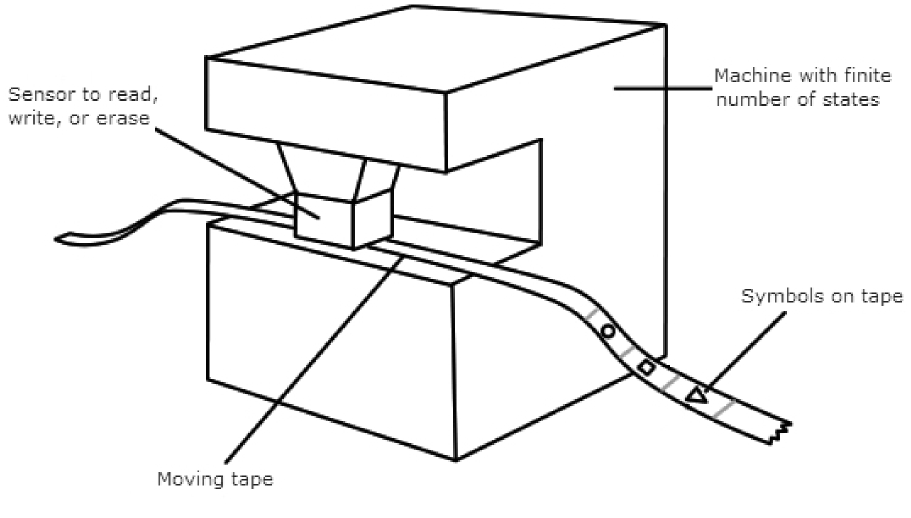


The complete collection of all 171 patterns needed to satisfy constraints of the type shown on the previous page. If none of these 171 patterns satisfy a particular constraint, then it follows that no pattern at all will satisfy the constraint. The patterns are labelled by numbers which specify the minimal constraint which requires the given pattern. Patterns differing by overall reflection, rotation or interchange of black and white are not shown.

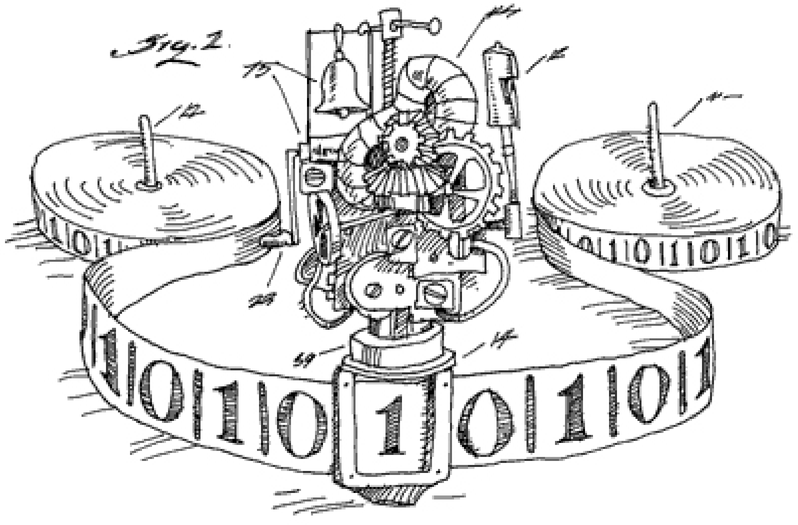
Computational Universality

Turing Machines!

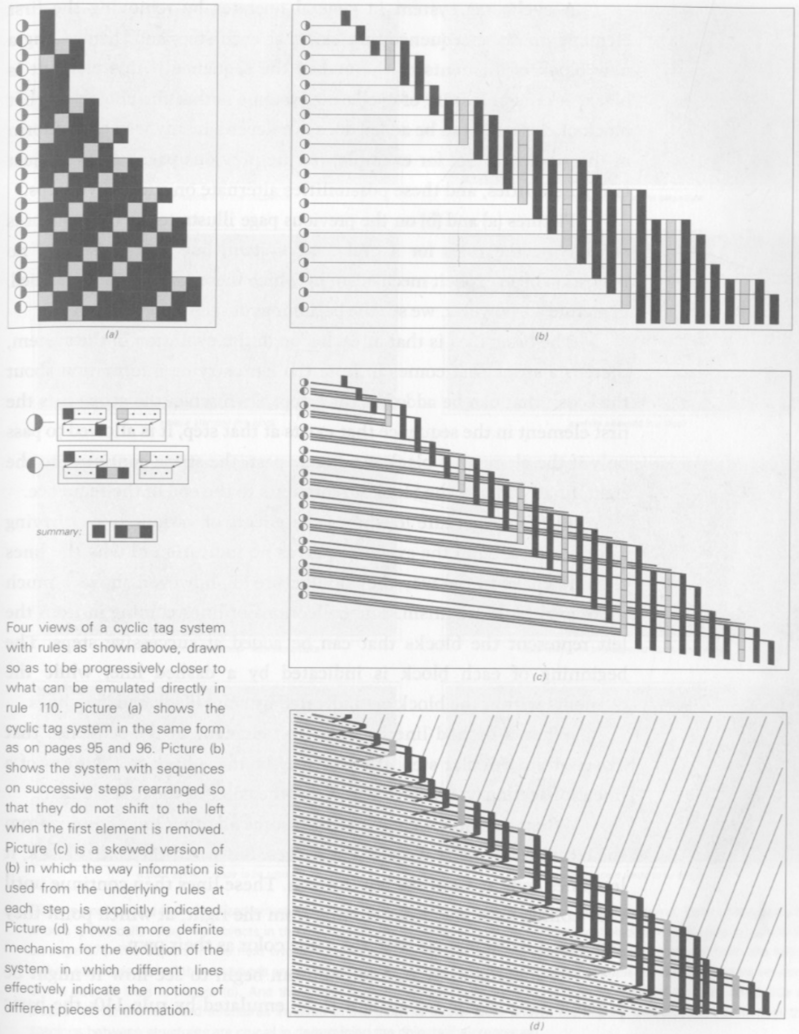
These are a type of mental construct developed by Alan Turing in the 1930’s. They describe how computation functions, including the top and bottom limits of computation. They limit what is computationaly possible. A computer is “universal” if it can Emulate a Universal Turing Machine.


Sensor to read, write, or erase
Machine with finite number of states
Symbols on tape
Moving tape

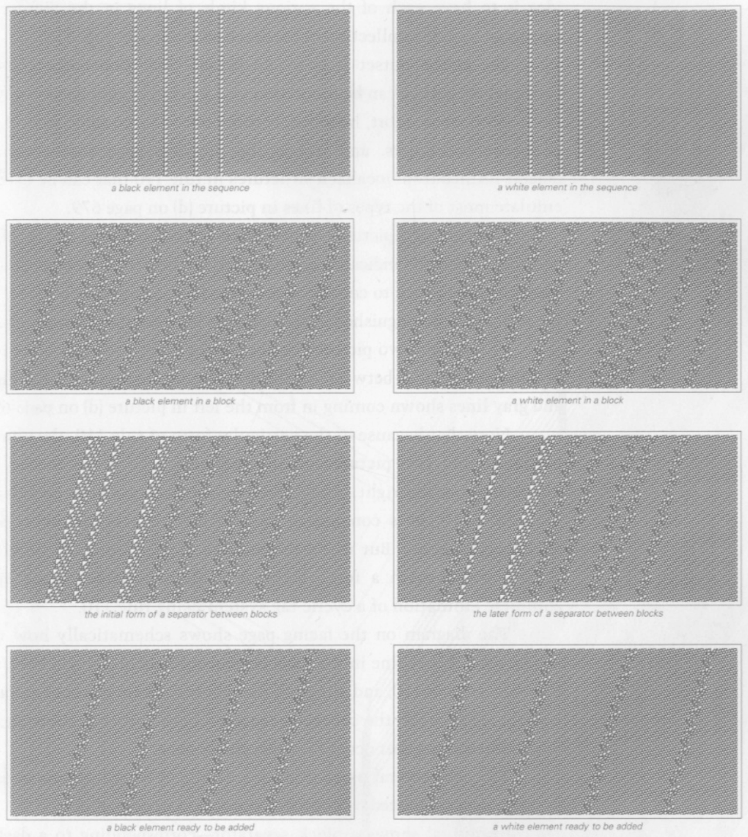
A Turing Machine



Computational Universality

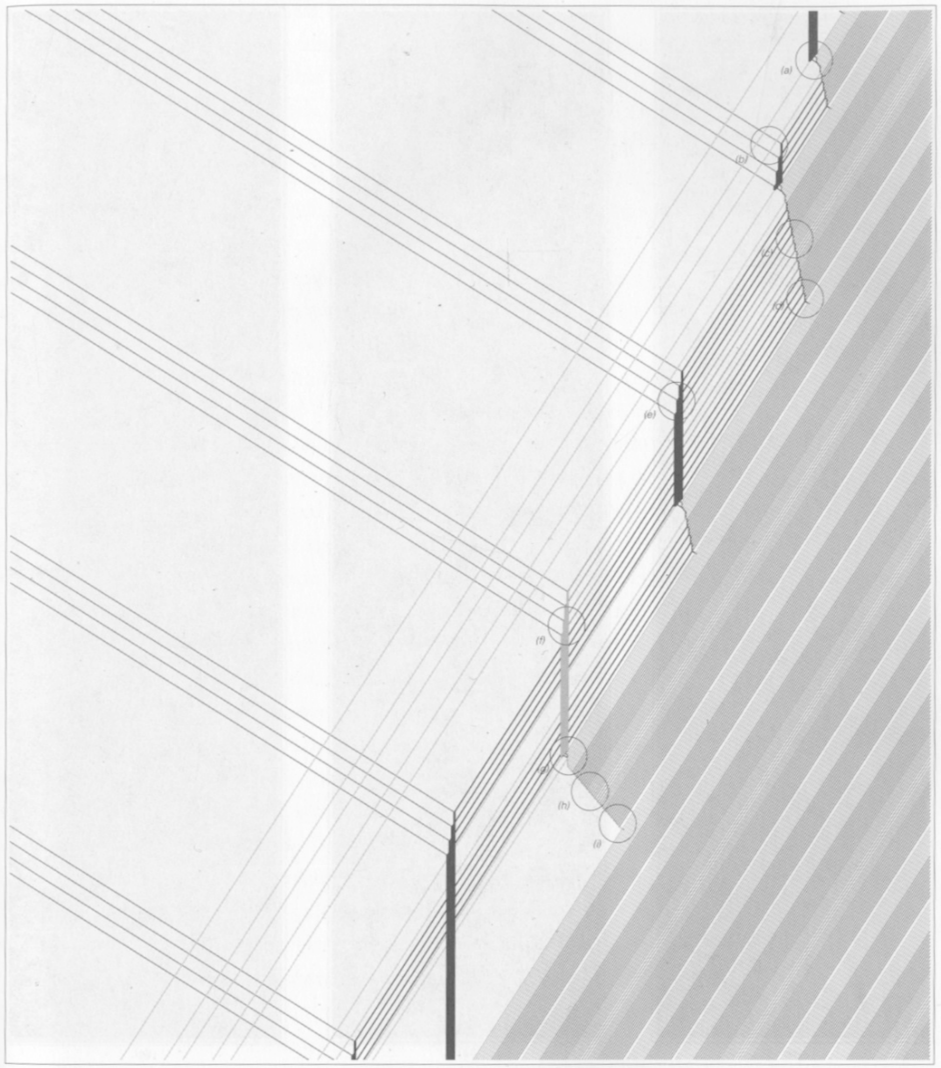


Four views of a cyclic tag system with rules as shown above, drawn so as to be progressively closer to what can be emulated directly in rule 110. Picture (a) shows the cyclic tag system in the same form as on pages 95 and 96. Picture (b) shows the system with sequences on successive steps rearranged so that they do not shift to the left when the first element is removed. Picture (c) is a skewed version of (b) in which the way information is used from the underlying rules at each step is explicitly indicated. Picture (d) shows a more definite mechanism for the evolution of the system in which different lines effectively indicate the motions of different pieces of information.


a black element in the sequence
a white element in the sequence
a black element in a block
a white element in a block
the initial form of a separator between blocks
the later form of a separator between blocks
a black element ready to be added
a white element ready to be added

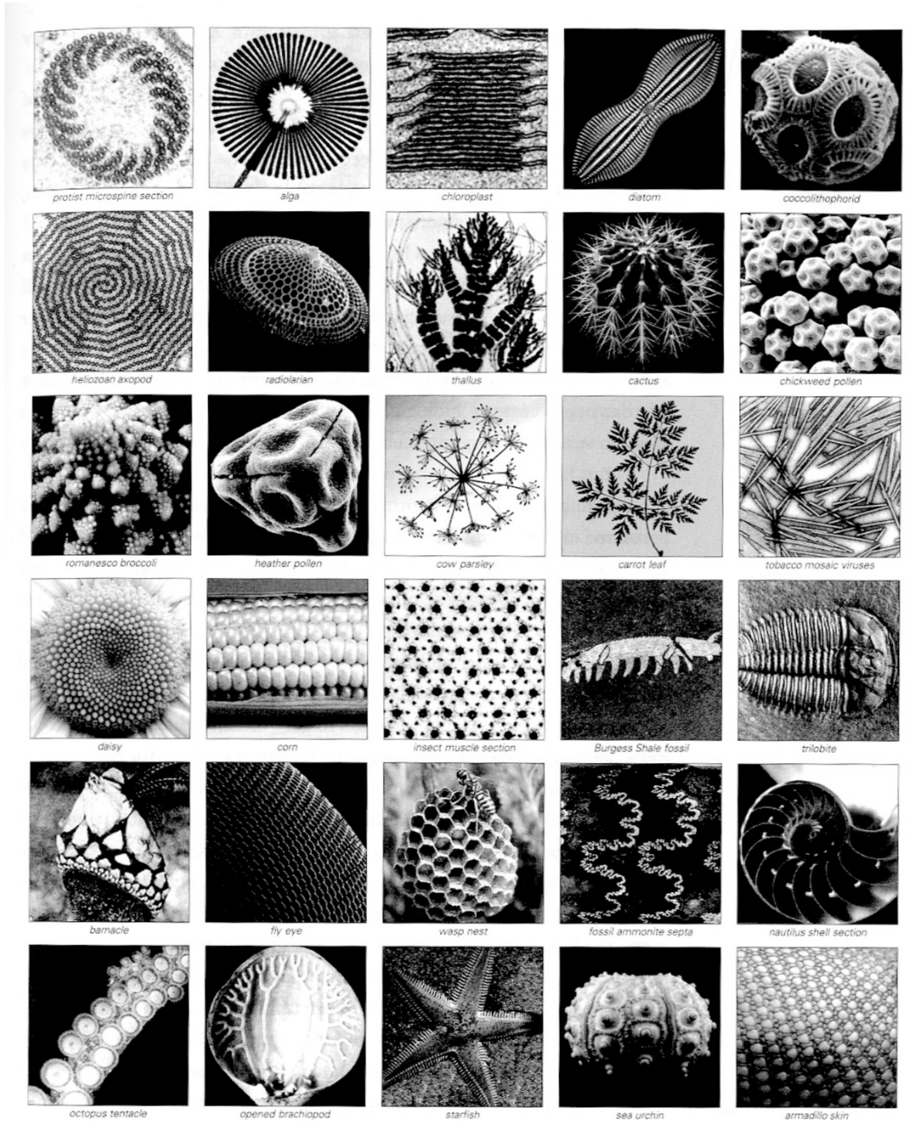
Objects constructed from localized structures in rule 110, used for the emulation of cyclic tag systems. Each of the pictures shown is 500 cells wide. The objects in the top two pictures correspond to the thick vertical black and gray lines in picture (d) on page 679. The objects in the next two pictures correspond to the dark and light gray lines that come in from the left in picture (d). (Note that all the structures are left-right reversed in rule 110.) The third pair of pictures correspond to two versions of the dashed lines in picture (d). And the fourth pair of pictures correspond to right-going lines on the right-hand side of picture (d). All the localized structures involved in the pictures above were shown individually on page 292. Note that the spacings between structures are crucial in determining the objects they represent.

Computational Universality

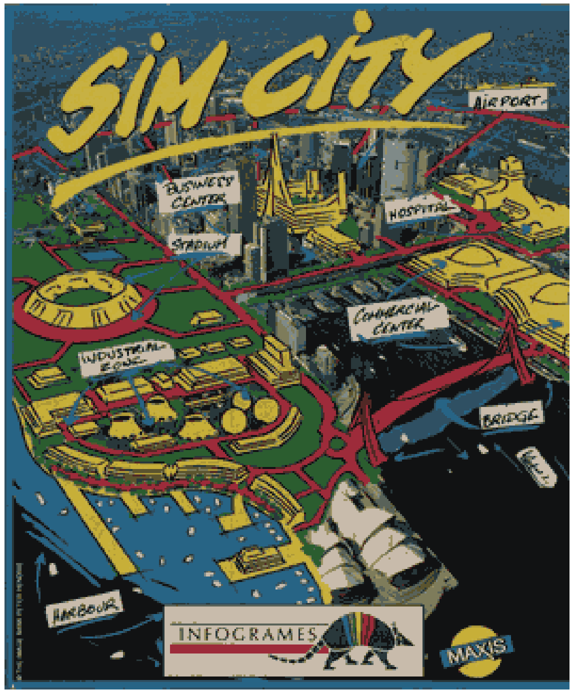


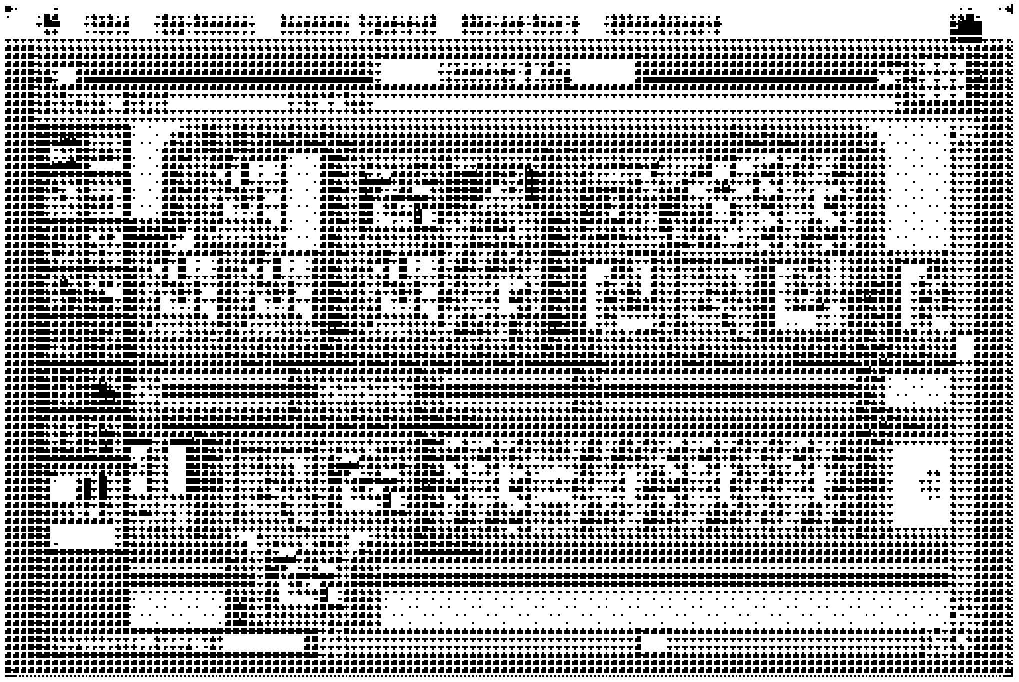
A schematic diagram of how rule 110 can be made to emulate a cyclic tag system. Each line in this diagram corresponds to one localized structure in rule 110. Note that the relative slopes of the structures are reproduced faithfully here, but their spacings are not. Note also that lines shown in different colors here often correspond to the same structure in rule 110.

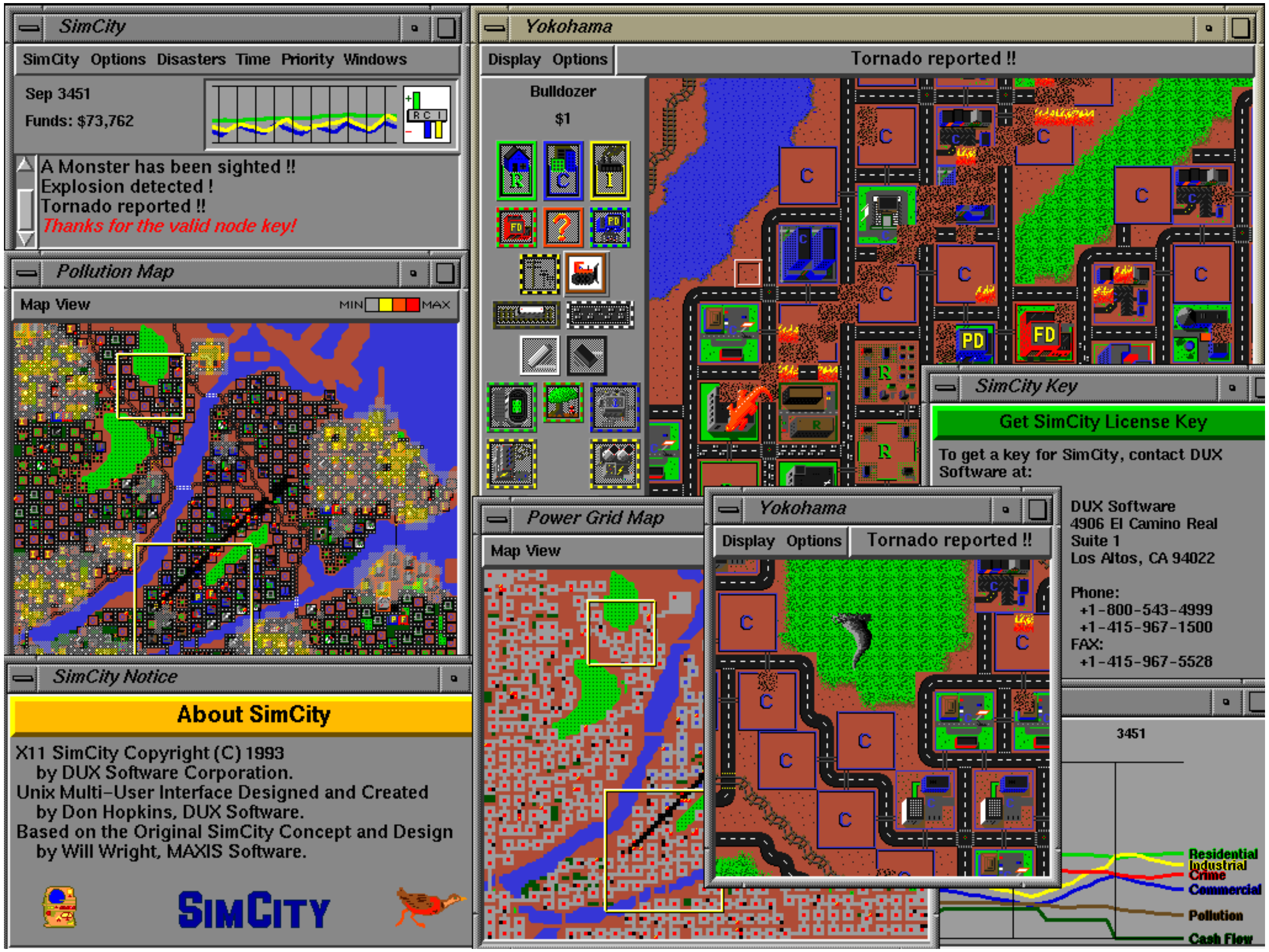
Simulation & Emulation



Simulation & Emulation







A Monster has been sighted !!
Explosion detected !
Tornado reported !!
Thanks for the valid node key!

Tornado reported !!

To get a key for SimCity, contact DUX Software at:

DUX Software
4906 El Camino Real
Suite 1
Los Altos, CA 94022

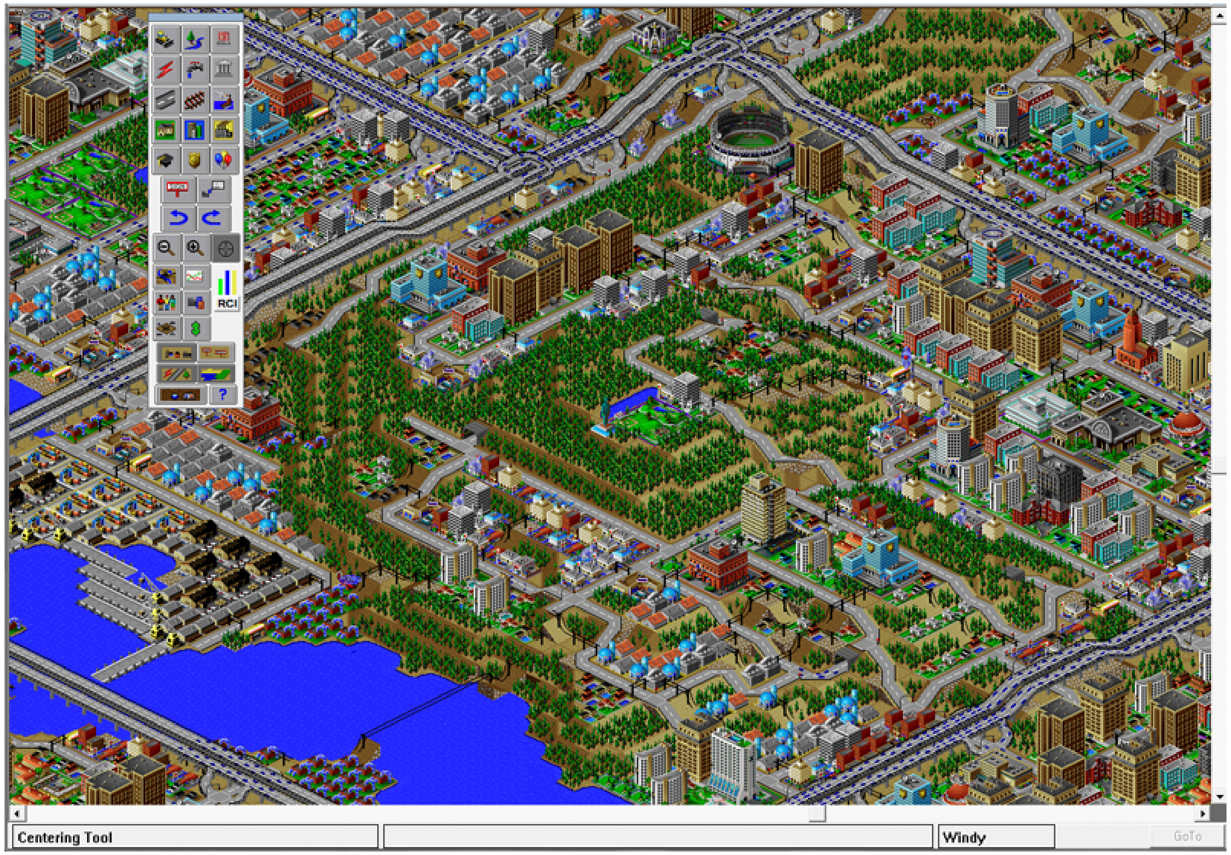
Phone:
+1-800-543-4999
+1-415-967-1500
FAX:
+1-415-967-5528

SimCity Notice

About SimCity

X11 SimCity Copyright (C) 1993
by DUX Software Corporation.
Unix Multi-User Interface Designed and Created
by Don Hopkins, DUX Software.
Based on the Original SimCity Concept and Design
by Will Wright, MAXIS Software.

SIMCITY


RCI
Centering Tool
Windy
GoTo

File
Window
View
Options
Speed
SimAnt - Tutorial Game
You feel much better now!
House View
You feel much better now!
MAP
PAUSE
SCENT
Health
Y B R
Patch
House
Strat
Graphic
MATE
Queens
0
AUTOMATIC
MANUAL
PLACE
BLACK COLONIES 1
RED COLONIES 1

Behavior Control
Forage
86%
Dig
13%
Nurse
00%
y needs more foo