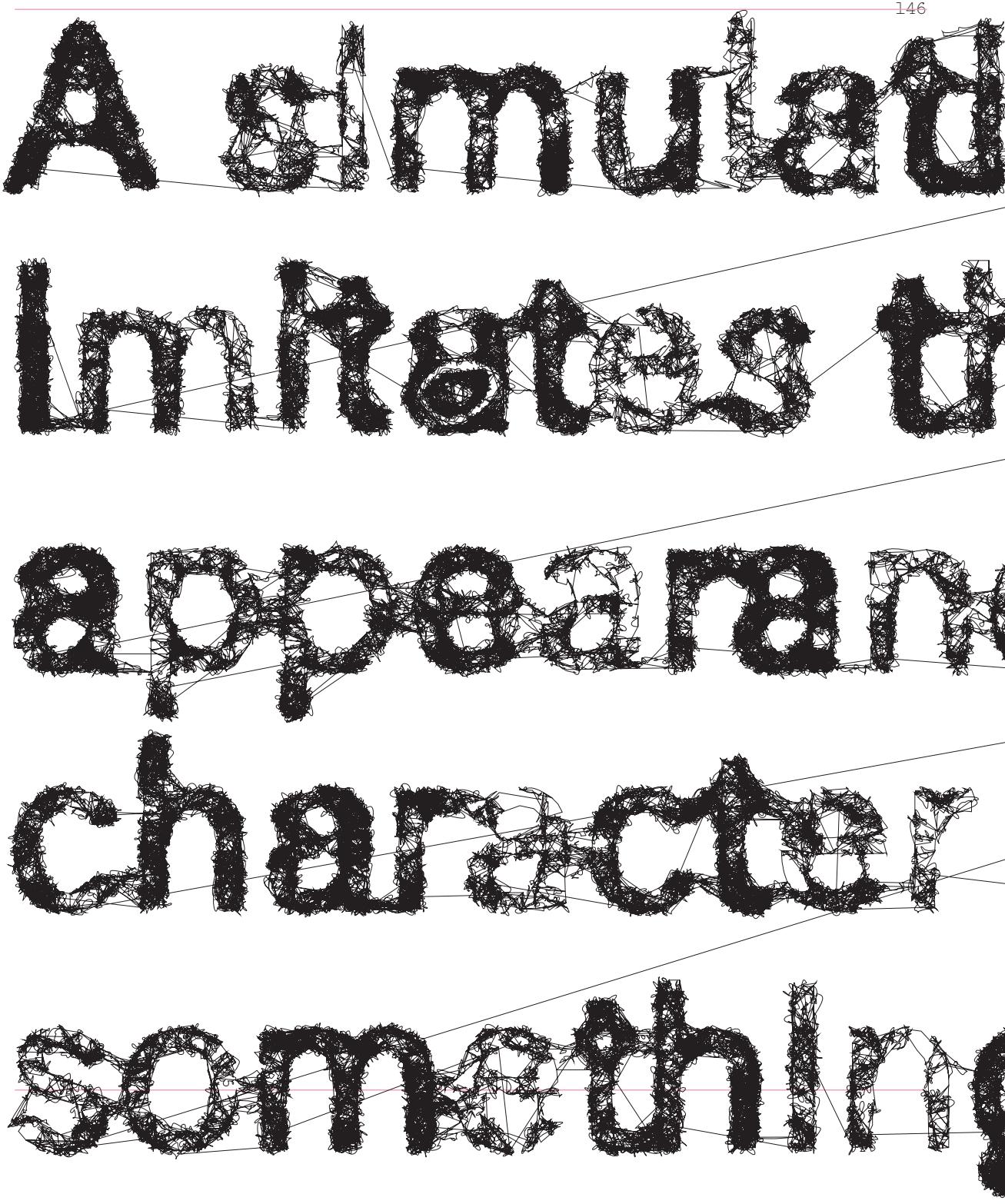
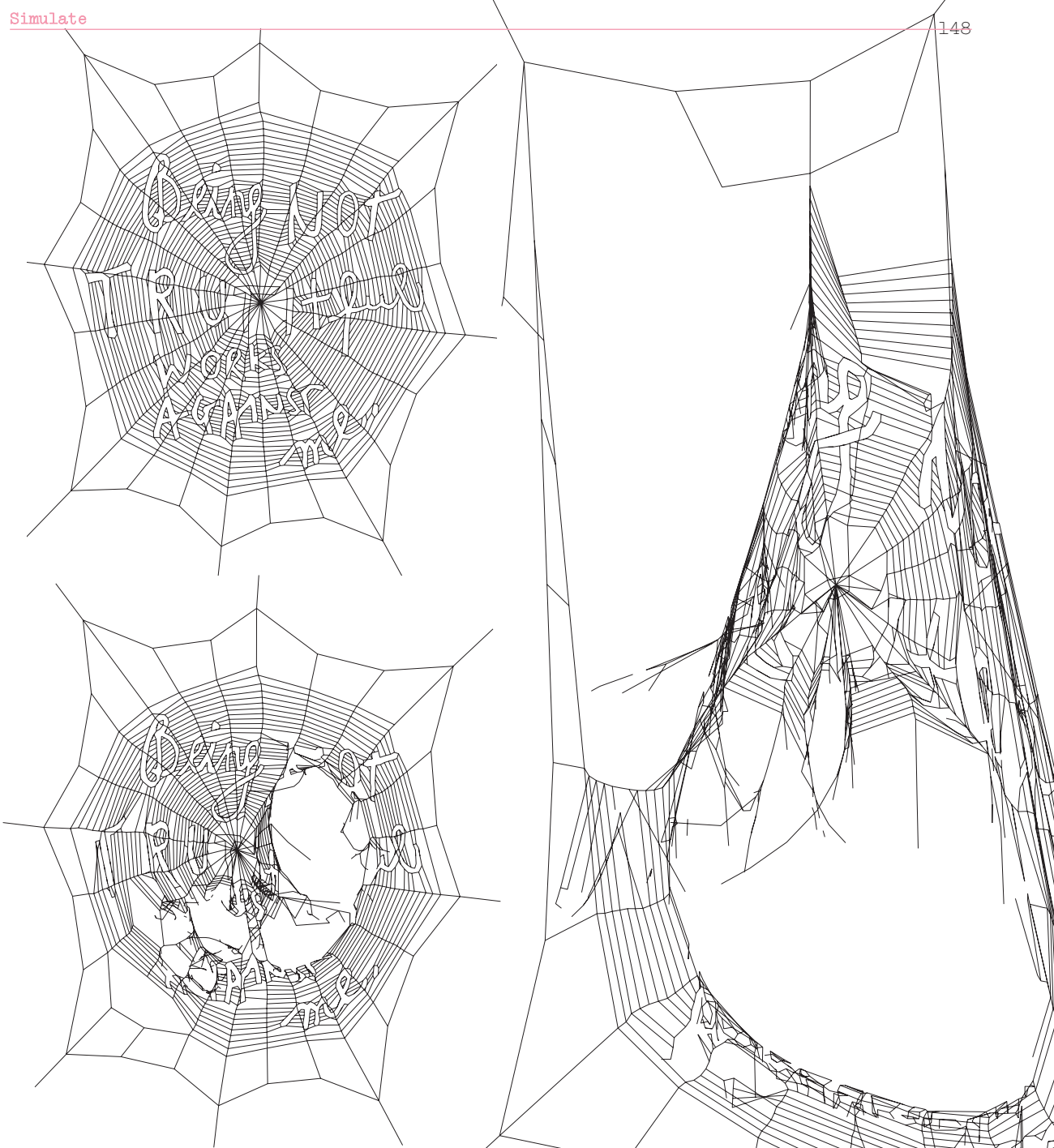

A stimulat
inmates th
appearan
character
something

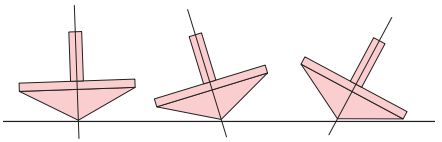
From watching Hollywood movies, we all know that software's ability to simulate the real world is improving. The clothes worn by simulated characters flow like the real thing, the hair on cartoon animals is fluffy and bouncy, and the quality of light in fully rendered scenes is atmospheric and natural. These technical achievements present an interesting aesthetic question. Is modeling the natural world with precision the ultimate goal of software simulation? In some situations, the answer is a definite yes; for example, scientific weather and traffic simulations are only useful when they are realistic. Within other domains, such as design, architecture, and art, high fidelity is less important than the final experience. Bending the rules can create something unexpected and sublime. Simulation can be a precise tool, but also a foundation for something beyond.

These letters simulate an idiosyncratic human hand. They were created by drawing curves, from a given point to one of its nearest neighboring points, rather than directly to the next logical point. This demonstrates the computer's ability to use ordered and systematic processes to produce the opposite result.



Being Not Truthful, by Ralph Ammer and Stefan Sagmeister, 2006
A virtual spider web with the sentence "Being not truthful works against me" is woven into it and projected on a wall. As viewers pass by, the web is torn apart, sticking to the viewer's shadow, only to be reconstructed a short time later. The fragility of the web "serves as a metaphor for the vulnerability of the maxim and the effort to perpetuate it."

Every simulation has three parts: variables, a system, and a state. A variable is a value that represents a component of the simulation. A system is a description of how the variables interact. The state of the system is the values of the variables at any given time. This becomes clearer through an example: a simulation of a spinning toy top. The first step is to realize there are an infinite number of variables for simulating anything. Beyond the most significant aspects of any object or system, there is an infinite amount of miniscule details that may have profound effects. For example, no two tops are totally identical. How do we measure the affect of a small dent that changes the rotation ever so slightly? How do we measure the difference in behavior when the top is new, rather than old and worn? To make a software simulation, it's necessary to select a finite number of variables from an infinite number of options. For a simple top simulation, perhaps the revolutions per minute (RPM), the angle, the diameter, and the height are sufficient. (For simplicity, let's ignore the fact that a person needs to give the top its initial energy.) All software simulations run as a series of time steps. The simulation starts at the first step, then the values are recalculated for the second step, and so on. In the simulation of the top, the state is composed of the values of the four variables at a specific step. The RPM and angle change constantly until the top comes to a stop, but the diameter and height are constant.



As friction on the bottom of the top causes it to slow down, the angle changes so it tilts closer and closer to the ground. Eventually, at a certain ratio of speed to angle, the top falls over.

This simulation becomes more interesting and complicated when we add another system: a simulation of a world. What happens when the top hits a wall while spinning? Does it fall over? What happens when it moves from a smooth to a rough surface? Does it slow down? You can start to imagine how complicated a software simulation can become.

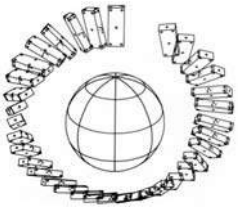
Though simulations often make use of parameterization, these two techniques have some important differences. Whereas parameterization allows for precise, top-down control of form; simulation involves bottom-up mechanisms that make it very difficult to predict how a particular system will behave. Each iteration will have differing results, which may not be obvious from the description of the system. Simulation is the creation of the possibility of form. Whether designed to reproduce the natural world or to generate novel and unexpected forms, it is this very open-ended quality that makes simulation such a powerful technique.

Transformation of a Necklace Dome, by MOS Architects, 2008
Created for the Buckminster Fuller: Starting with the Universe retrospective at the Whitney Museum of American Art, this piece represents a dialog with Fuller's necklace dome. The piece was generated in a simulated physical environment to create a structure whose "connections are flexible locally, but within an overly redundant network of forces they perform as a rigid structure." It is hung from the ceiling and constructed from over 5,000 aluminum rods whose lengths diminish as they reach the ground, creating a structure that "is more rigid at the bottom and more flexible at the top."

MODELING PHYSICAL SYSTEMS

To gain a deeper understanding of the world around us often requires rigorous experimentation and testing. In the sciences, the theory-to-experiment cycle can take years or decades. Often, the system we want to test is too large, too complex, too distant, or the time scales are too large to effectively test a hypothesis. Computers and code have offered a new way to explore these and other systems, and to test our current knowledge. For example, the first computer-generated film simulated a satellite orbiting the Earth. It demonstrated that a satellite can be stabilized so that one side will always face Earth. The film, titled A Two-Gyro Gravity-Gradient Altitude Control System, was made by Edward Zajac at Bell Labs in 1961.

[1] David Owen, "The Anti-Gravity Men," The New Yorker, June 25, 2007.



Much closer to home, simulations in the field of physics that focus on gravity and the interactions between objects have become a part of everyday life. Many of the most successful video games feature the realism of the worlds they portray. With such accurate worlds, it is possible to create games with truly lifelike environments, which, in turn, allow game developers to loosen some restrictions on what the character can do. Rather than restrict how the player can get from point A to point B, these games give the player the freedom to find other routes and use objects in the environment in new ways to achieve the goal at hand.

The ability to simulate physics has had a substantial impact on architecture and engineering. These tools allow engineers to simulate how forces and loads move through virtual structures. Based on the outcome, the model is modified and the simulation is run again until an optimal combination of structure and material is found. The CCTV Tower in Beijing, designed by Rem Koolhaas's firm Office for Metropolitan Architecture (OMA) and built by the engineering firm Arup, went through years of detailed simulation to ensure that the building could stand and survive an earthquake while maintaining its unique shape. The simulations, created by Arup, tested how every part of the building would react in the event of a large earthquake so that adjustments could be made to materials or design. The building was constructed in two parts, and Arup was able to calculate the time of year (late autumn) and time of day (five o'clock in the morning) when conditions would be ideal for connecting the two halves.[1]

Simulations can also become a laboratory for testing new ideas. For example, a Japanese supercomputer built in 2002 called Earth Simulator was designed to imitate weather systems. The Earth Simulator allowed researchers to enter real-time climate data, then speed up the clock and see the cumulative effects hundreds of years in the future. These complex simulations work to expand our understanding by providing a sandbox for testing the effects of different actions on the system. Building such a simulation requires a meticulous description of how things work and interrelate. If there is an error in the assumptions, the simulation will look strange or behave incorrectly, which in turn requires more testing and experimentation.

Will Wright's SimCity is one of the most popular games of all time. The objective of the game is to create and manage a successful city. To do so, the player can control zoning and taxes, construct buildings, roads, and railways, and respond to natural disasters. Constructing a large city takes time, and players often work on the same city for years. SimCity lets the player experiment with a staggeringly complex system, going so far as to offer scenarios based on real-world cities in times of crisis. Playing a game as complex as SimCity requires the player to become intimately familiar with the assumptions underlying the simulation. To be successful, he or she must put aside all preconceived ideas about public policy and learn what actions are rewarded in the game. Some have argued that SimCity encodes certain biases toward policies like rail transportation and environmentalism. In the context of a game purporting to be

A Two-Gyro Gravity-Gradient Altitude Control System, by Edward Zajac, 1961
Zajac created this computer-generated film at Bell Labs to explore options for stabilizing a communications satellite so that one side always faces Earth.

The Incredible Machine, by Sierra, 1992
The Incredible Machine is a video game based on Rube Goldberg contraptions. The player must arrange a set of odd objects—such as mice in cages, cannons, or conveyor belts—to create a machine capable of achieving a goal such as lighting a candle. The game uses a simple physics-based engine to simulate interactions between the machine's parts, as well as with environmental effects like air pressure and gravity.

Persönliches Wagnis (top), Unabweisbare Gefahr Nr. 6 (bottom), by Gerhard Mantz, 2009 Mantz began making digital 3-D models as prototypes of his sculptures, but he pushed further into creating software-based landscapes as metaphors for psychological states. Simulated light, atmosphere, and water are used to express emotions.

an accurate simulation of the real world, such biases, intentional or not, can have a powerful effect on the beliefs and habits of the players.

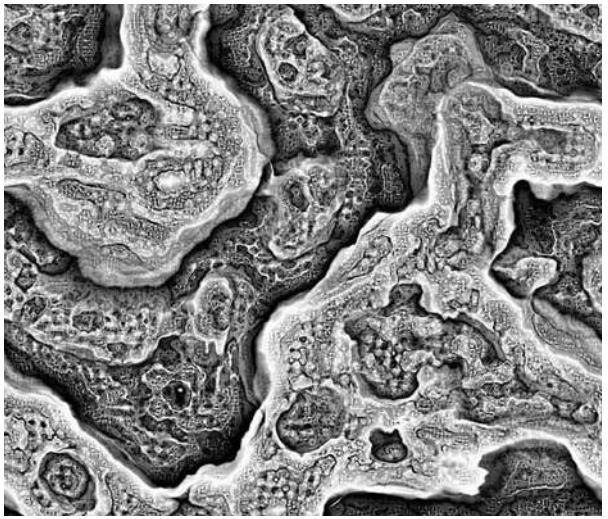
The complex processes behind many fascinating natural phenomenon have been described in equations and procedures, making them ideal for exploration in code. Fractals are a perfect example of a naturally occurring phenomenon that has been explored in depth using computers. In 1978, while working for Boeing, computer graphics researcher Loren Carpenter first started using fractal geometry to create complex and realistic artificial landscapes from simple and elegant processes. First, a rough landscape is created out of large triangles, then each triangle is broken into smaller triangles, and each smaller triangle is broken down again, and the process repeats recursively until a realistic terrain emerges. Carpenter went on to use these techniques to create a fully computer-generated sequence of an alien planet for the 1982 film Star Trek II: The Wrath of Khan.

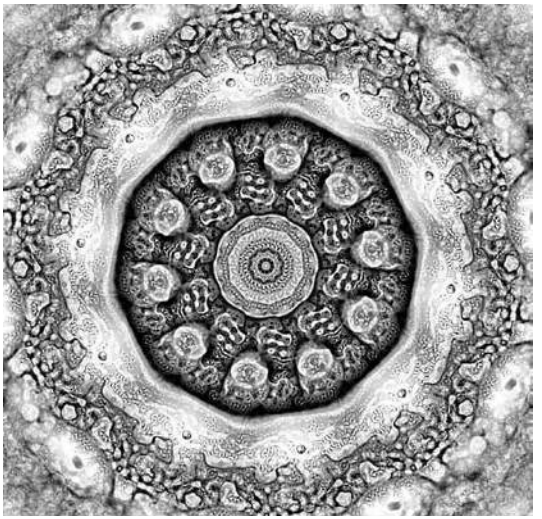
One of the most fascinating biological phenomena to be discovered and described in mathematical terms is the reaction-diffusion system. In 1952 Alan Turing, one of the pioneers of computer science, turned his attention to the problem of morphogenesis: how a fertilized egg turns into a fully formed animal. Turing's novel approach described the problem in terms more familiar to physics than biology. He described the problem as one of symmetry breaking, and eventually developed equations for some of the problem's simpler aspects. He was, however, limited to mathematical expressions that would be computable using only a pencil and paper, and he speculated that perhaps one day a computer would be capable of solving the problem using more complex mathematics. Twenty years later, scientists Hans Meinhardt and Alfred Gierer fulfilled this prophecy and used the computer to continue and expand on Turing's work. Reaction-diffusion systems are incredible in their ability to create a wide variety of patterns, from butterfly wings to zebra spots. The relative simplicity of the system, in which one chemical compound activates another, which in turn inhibits the first, adds to this allure.

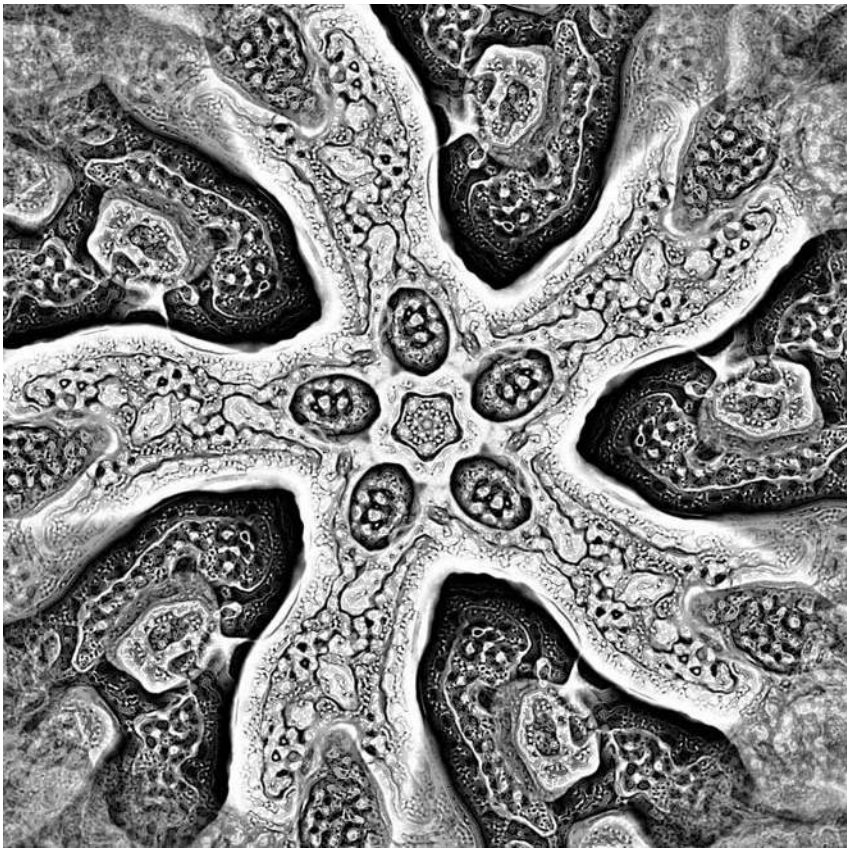
SIMULATE

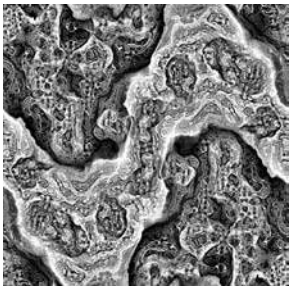
SimCity 2000, by Maxis, 1993
Players design, build, and manage cities by determining every detail of planning and urban development. SimCity, released in 1989, was the first in a long line of simulation games leading up to the release of The Sims in 2000, which simulated the day-to-day life of suburbanites. Will Wright later released Spore in 2008, in which the player controls the development of a species, from a microorganism to a space traveler.

Print magazine cover, by Karsten Schmidt of PostSpectacular, 2008
A reaction-diffusion system was used to create the cover for the August 2008 issue of Print magazine. A 3-D form was made using code; it was then fabricated using a 3-D printer.









MSRSTP (Multi-Scale Radially Symmetric Turing Patterns), by Jonathan McCabe, 2009

McCabe created imagery reminiscent of biological structures by employing several reaction-diffusion systems.

Nexi,
by the Personal Robotics Group at MIT Media Lab, UMASS Amherst, and Xitome Design, 2007
This MDS (Mobile/Dexterous/Social) robot is equipped with a face capable of expressing a wide range of emotions that reveal its inner state.

Micro.Adam and Micro.Eva, by Julius Popp, 2002
A pair of wall-mounted robots try to learn how to rotate by moving their inner actuators and changing their centers of gravity. Popp's aim is "to find self-adapting algorithms that feel and learn the robot's body behavior. The two robots are 'born' with the same 'empty' programs, which then must form themselves according to the specific body characteristics of each robot."

ARTIFICIAL INTELLIGENCE

The history of artificial intelligence (AI) is a tortured one. The mid-1950s saw an explosion of research and speculation on the possibility of creating artificially intelligent machines. The development of the computer was a central catalyst, pushing AI research forward at a breakneck pace. At first, problems were being tackled so quickly that AI pioneer Marvin Minsky was quoted as saying, "Within a generation…the problem of creating 'artificial intelligence' will substantially be solved."[2] By the 1970s, however, progress in AI had slowed and funding for large, expensive projects was getting scarce. Nevertheless, progress has continued, and AI software is common in fields as diverse as banking and medicine.

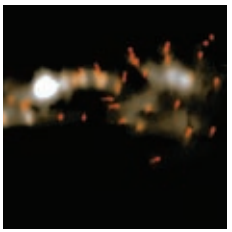
The goals of AI are varied. Some researchers imagine fully sentient beings capable of anything we might term intelligent, while others are concerned with specific activities such as game playing, planning, pattern finding, and social interaction. Robotics researcher Rodney Brooks broke new ground in the 1980s by focusing on behavior-based systems, what he calls Cambrian intelligence. Brooks's robots are constructed of multiple layers of behavior-driven processes, with higher-level layers having some control over inputs on the lower layers and vice versa. Brooks calls this subsumption architecture, and it has had a large impact on robotics and AI in general. For example, one layer might be concerned with avoiding obstacles, on top of that could be a layer that directs the robot toward a goal, and on top of that might be a layer that is interested in faces. When the highest level finds a face, it tells the lower layer where it wants to go, which in turns is responsible for the actual moving, all the while the lowest layer is actively looking out for obstacles in the robot's path.

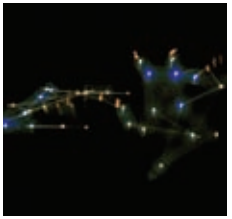
All discussions of AI pose problems of embodiment. These problems concern philosophical questions like the existence of a mind-body split, to more practical ones of how a computer or robot can sense the world around it. When Brooks developed the robot Cog, he designed it to be somewhat human in appearance, with video-camera eyes and a face. Looking at the problem of embodiment face on, he argues, "If we are to build a robot with humanlike intelligence then it must have a humanlike body in order to be able to develop similar sorts of representations."[3] The current work of researcher Cynthia Breazeal and the Personal Robots Group at the MIT Media Lab has taken this research in new directions with the development of the Nexi Mobile/Dexterous/Social (MDS) robot. MDS robots are designed to learn about and engage in human social interaction. Nexi has a small body, which allows it to move freely and pick up objects, but perhaps its most interesting feature is its expressive head and face, which are capable of a wide range of emotions, including sadness, anger, confusion, excitement, and even boredom.

Video games are another significant area of exploration in AI. Even simple games like tic-tac-toe may be used to test novel approaches to creating intelligent behavior. The defeat of chess grandmaster Garry Kasparov to IBM's Deep Blue supercomputer in 1997 received widespread media attention, however, chess is not the only or even the central interest for game-playing AI. Video games designed for consoles and computers are one of the most visible and active areas of AI development. Though the goals are often more restricted than in academic research, games offer an environment to test many key problems: planning, reacting to events in the real world, and interacting with and moving through an environment. For example, AI game characters are expected to look and act like real players; they must execute complex attacking behaviors while protecting themselves from counterattacks and responding to input from the player in a believable way that also drives the narrative forward. The 2005 Monolith

[2] Hubert L. Dreyfus and Stuart E. Dreyfus with Tom Athanasiou, Mind Over Machine: The Power of Human Intuition and Expertise in the Era of the Computer (New York: Free Press, 1986), 78.

[3] Rodney Brooks, "Cog Project Overview," 2000, http://www.ai.mit.edu/projects/humanoid-robotics-group/cog/overview.html.





Loops,
by The OpenEnded Group, 2008
Loops is a real-time generative animation built with motion-capture data from the performances of dance choreographer Merce Cunningham. The software uses a complex action-selection mechanism to interpret the motion data to create stunning visual imagery. Each point chooses how to interact with nearby points, how to move, and how to draw itself. The creators explain, "Our basic idea was to make the points that construct the hands of Loops 'autonomous'—so that these points are, to a limited degree, live 'creatures.'"

AARON,
by Harold Cohen
1973–present
The AARON software creates original drawings based on the rules encoded in it. Over the last thirty years, the software has advanced from drawing basic geometric forms to human figures, from black-and-white drawings to color prints.

seed "g" designed by hand and given to the program

designed by the Letter Spirit program using seed letters "b,c,e,f,g"

designed by hand by Douglas Hofstadter to be in the same style as the "g"

Letter Spirit, by Gary McGraw and John Rehling with Douglas R. Hofstadter, 1996
Letter Spirit used a two-by-seven grid of lines to create letters. Given a few seed letters, the program attempts to generate the entire alphabet in the same style as the seeds.

Productions game F.E.A.R. was notable for the sophisticated AI control of the game's characters. Opponents can spontaneously act as a team, providing cover fire for one another and moving around to flank the player. A different approach to game AI is found in Façade, created by Procedural Arts in 2005. In this game, the player is a guest invited to dinner by a married couple. As the evening progresses, tension in the couple's relationship forces the player to make choices that have a direct effect on the behaviors and lives of the characters. Unlike a simple branching structure, like a classic Choose Your Own Adventure book, the characters in the game change their behaviors based on the actions of the player, thus affecting the moment-to-moment flow of the narrative.

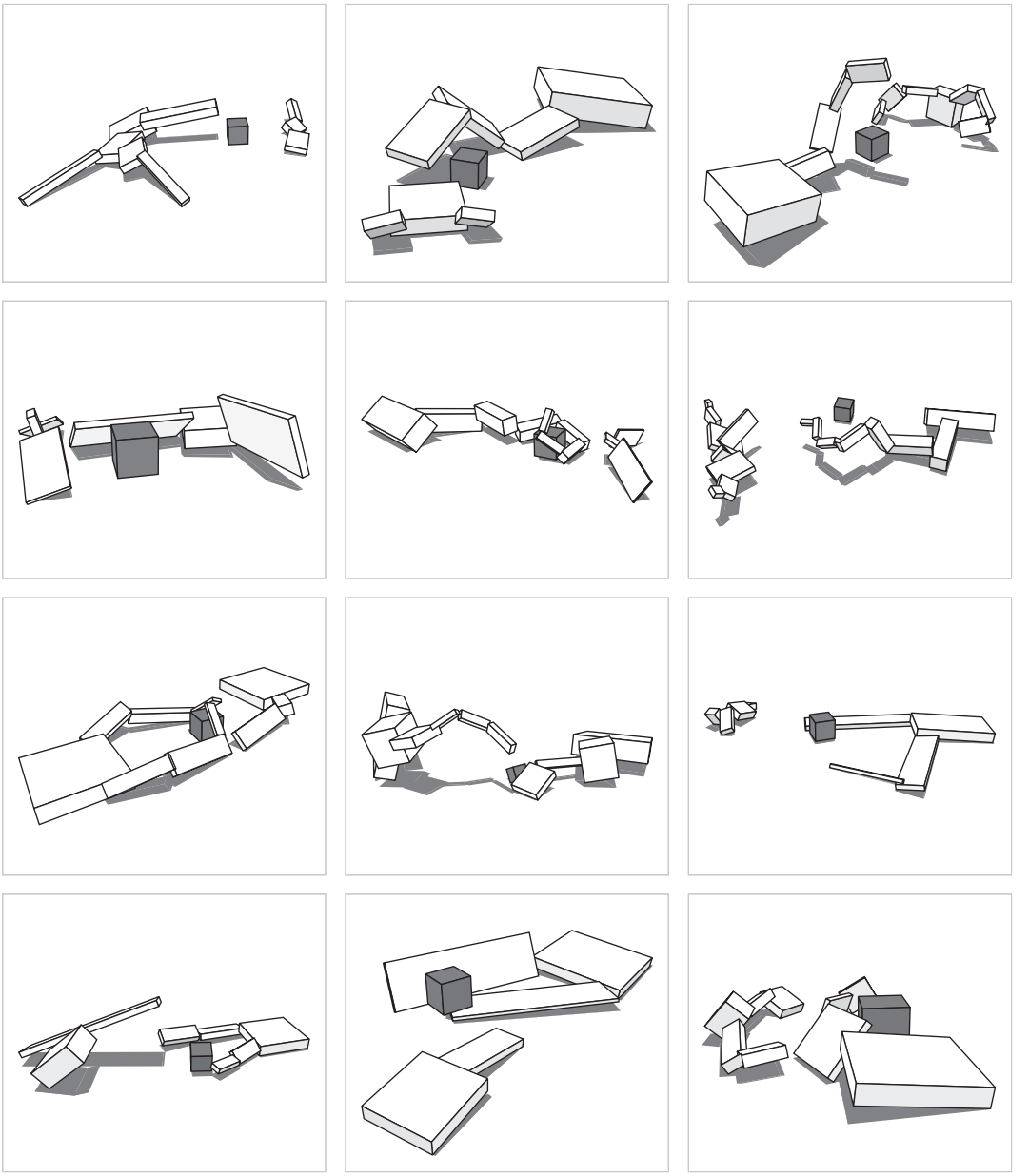
Since the first days of AI, researchers have searched for ways to imbue computers with the spark of creativity. In contrast to systems focused on reasoning and deduction, these projects explore strategies for creation. Programs such as Douglas Lenat's Automated Mathematician, written in 1977, tried to generate mathematical theorems from basic principles, while in 1983 William Chamberlain and Thomas Etter's Racter program was made famous when it was credited with writing the book The Policeman's Beard Is Half Constructed.

Perhaps the most famous piece of creative software is Harold Cohen's AARON. Cohen programmed AARON to create drawings, which have been displayed at numerous museums including the Tate Britain in London and the San Francisco Museum of Modern Art. He has worked on AARON since 1973, continually adding features and expanding its capabilities. The software has progressed from black-and-white outlined shapes to scenes of people posing and dancing surrounded by colorful vegetation. AARON is fully automatic and produces images with no human interaction. Cohen recently added reflectivity to AARON so that the program monitors the image as it's created and alters how it applies its coloring rules based on what it finds. The consistent style seen in AARON's images suggests that the rules used to compose, choose colors, and evaluate its drawings are an increasingly accurate encoding of Cohen's own ideas about composition.

A different approach to the problem of artificial creativity is found in the work of the computer and cognitive scientist Douglas R. Hofstadter. In the late 1970s, Hofstadter set out to model the mechanisms of creativity in a computer. His Fluid Analogies Research Group created software to solve seemingly trivial problems, such as finding the rule for a number sequence and solving anagram puzzles. In contrast to traditional computational or AI techniques, Hofstadter's team tried to mimic their own mental processes of solving these puzzles; the software was designed to repeat the mental back-and-forth of coming up with ideas and trying them out until the solution is found. One program, Copycat, tried to find novel analogies between sequences of letters, and use those analogies to generate new sequences. This work eventually led to the project Letter Spirit. Hofstadter explains:

> The Letter Spirit project is an attempt to model central aspects of human creativity on a computer. It is based on the belief that creativity is an automatic outcome of the existence of sufficiently flexible and context-sensitive concepts....The specific locus of Letter Spirit is the creative act of artistic letter-design. The aim is to model how the 26 lowercase letters of the roman alphabet can be rendered in many different but internally coherent styles. Starting with one or more seed letters representing the beginnings of a style, the program will attempt to create the rest of the alphabet in such a way that all 26 letters share that same style, or spirit.[4]

[4] Douglas R. Hofstadter, Fluid Concepts & Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought (New York: Basic Books, 1996), 407.



Evolved Virtual Creatures, by Karl Sims, 1994
Evolved virtual creatures compete for the possession of a cube within this simulated world. The winner of each round of the competition receives a higher score, giving it the ability to survive and reproduce.

ARTIFICIAL LIFE AND GENETIC ALGORITHMS

Artificial life (a-life) has different goals from artificial intelligence. While AI focuses on higher-level cognitive functions, a-life imitates biological phenomena to simulate reflexes, behaviors, and evolution. The goals of a-life are to simulate life as it exists and to explore new categories of life. In his book Artificial Life: The Quest for a New Creation, Steven Levy describes the context:

> Artificial life…is devoted to the creation and study of lifelike organisms and systems built by humans. The stuff of this life is nonorganic matter, and its essence is information: computers are the kilns from which these new organisms emerge. Just as medical scientists have managed to tinker with life in vitro, the biologists and computer scientists of a-life hope to create life in silico.[5]

Since its origins in the early 1980s, a-life researchers have developed simulations based on many kinds of flora and fauna, ranging from cells to trees to insects to mammals. Within the visual arts, these simulations are the foundation for new ways to generate form. Instead of creating form directly through drawing or sculpting, it can be grown or evolved using a-life techniques.

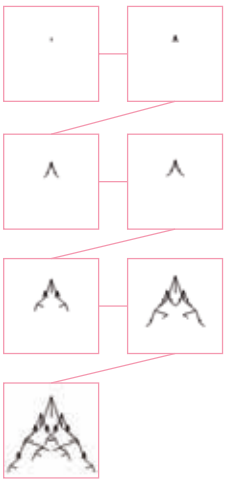
Related to a-life research, a genetic algorithm (GA) is a software process that simulates evolution by creating and changing an artificial genome. As in real genetics, an artificial genome is modified through crossbreeding and mutation. Crossbreeding (or sexual reproduction) uses a combination of genes from two parents to form a unique child. Parts of the genome of each parent are used to create a new child with a mix of both. Mutation affects a gene in a single creature, and this change is transmitted to the next generation. In nature, mutation can result from a copying error or radiation, among other events. In a-life, mutation is simulated by changing one or more characters of the simulated genome. The primary advantage of evolution within a software environment is the ability to go through thousands of generations (or more) within seconds rather than thousands of years.

The potential of genetic algorithms to evolve visual form was pioneered by the evolutionary biologist Richard Dawkins, through software he wrote for his book The Blind Watchmaker: Why the Evidence of Evolution Reveals a Universe without Design, published in 1996. He created the software to demonstrate his belief that the diversity of life on Earth is the result of a process of evolution involving the gradual accumulation of small changes. His software experiment produced results that far exceeded his expectations. He wrote, "Nothing in my biologist's intuition, nothing in my 20 years' experience of programming computers, and nothing in my wildest dreams, prepared me for what actually emerged on the screen."[6] Using only mutation and a simple genome of nine genes (or parameters), Dawkins generated shapes, one step at a time, that emerged to look remarkably like complex organisms (trees, insects, amphibians, mammals) from the starting point of a single pixel. In the words of author and Wired magazine cofounder Kevin Kelly, he demonstrated visually that, "While random selection and aimless wandering would never produce a coherent design, cumulative selection (the Method) could."[7] Dawkins called the software's design space Biomorph Land. He acknowledges that every biomorph form in this land already exists mathematically as a sequence of variables within a finite set, but he insists that the experience of evolving form within the software is a creative act, because the space is so large and the search must be directed rather than random. Although biomorphs are visually simple, the project opened the door for other researchers, including William Latham and Karl Sims, to create more fully realized visual projects.

[5] Steven Levy, Artificial Life: The Quest for a New Creation (New York: Pantheon Books, 1992), 5.

[6] Richard Dawkins, The Blind Watchmaker: Why the Evidence of Evolution Reveals a Universe without Design (New York: Norton, 1996), 59.

[7] Kevin Kelly, Out of Control: The New Biology of Machines, Social Systems & the Economic World (Reading, MA: Addison-Wesley, 1994), 266.

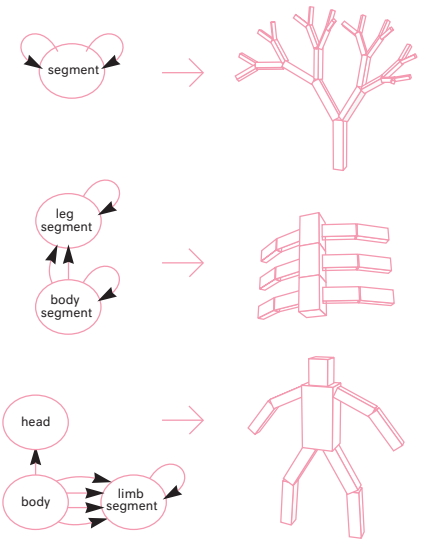


Crossover and mutation
Crossover uses part of the genome from each parent, and mutation is a random change to one or more elements of the genome. Together, they simulate biological evolution.

Biomorph, by Richard Dawkins, 1986
Starting from a single pixel in the upper-left, the form at the bottom evolved through many generations, each with only a single genetic mutation.

Morphogenesis Series # 4, by Jon McCormack, 2002
Using an evolutionary process, McCormack explored the growth and development of plantlike structures, based on native Australian species. Unlike natural selection, McCormack influenced the selection to present a personal interpretation of nature's process.

Karl Sims' Evolved Virtual Creatures, which he created in 1994, was a breakthrough in a-life. The project is so convincing that it remains relevant over fifteen years later. Through software, Sims was able to simulate the evolution of creatures to perform different tasks, such as swimming, walking, jumping, and following. Eventually he added the ability for two creatures to compete against one another to control possession of a cube. He did this by simulating the basic physical properties of gravity, friction, and collision, and then building creatures that could evolve at amazingly fast rates through crossbreeding and mutation. The genotype of his creatures is a directed graph, and the phenotype is a hierarchy of 3-D parts.



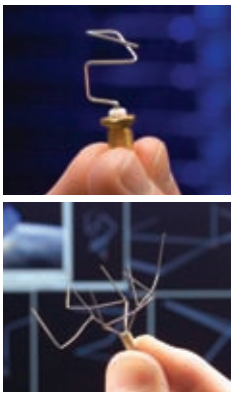
At each generation, some creatures were allowed to survive and reproduce based on their fitness, how well they performed the current task. For example, how well did they swim, walk, jump, or follow? To communicate the results, Sims produced an appealing series of short animations that featured the creatures in action. The most fascinating aspect of the project was the diversity of strategies the creatures developed to meet their goals. Despite the primitive geometry and limited joints, they had the essence of living, self-motivated animals. GAs are an emerging technique for solving real-world design problems that are well suited to optimization. The X-band antenna developed by NASA in 2004 is a clear example. Designing an antenna by hand is time and labor intensive, and therefore expensive. In contrast, software can be used to design a better antenna in less time. NASA describes the process:

> Our approach has been to encode an antenna structure into a genome and use a GA to evolve an antenna that best meets the desired antenna performance as defined in a fitness function. Antenna evaluations are performed by first converting a genotype into an antenna structure, and then simulating this antenna using the Numerical Electromagnetic Code (NEC) antenna simulation software.[8]

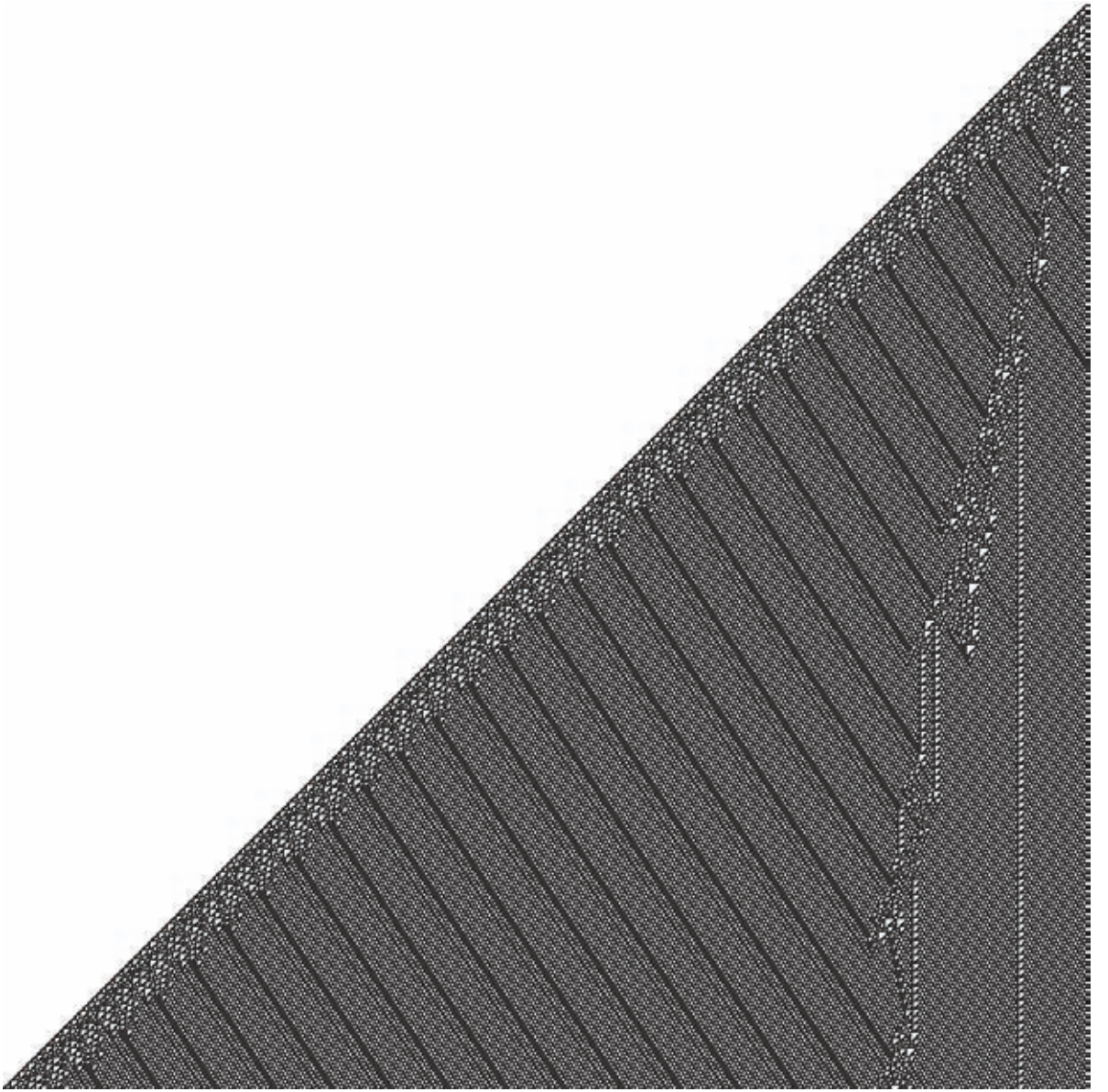
NASA claims that generated designs have the potential to outperform those designed by expert engineers. Additionally, the generated antennae often have radically different forms compared to those designed by hand. In general, GAs have the potential to find unique forms that more traditional design approaches overlook.

GAs and evolutionary thinking have played an important role in contemporary architectural theory and practice. John Frazer's 1995 book An Evolutionary Architecture presents the genetic algorithm as a technique for creating novel forms that bridge the form-function divide. Theorist Karl Chu has explored the intersection of computer and genetic code as an avenue for exploring possible futures.

[8] "NASA Evolvable Systems Group, Automated Antenna Design," http://ti.arc.nasa.gov/projects/esg/research/antenna.htm.



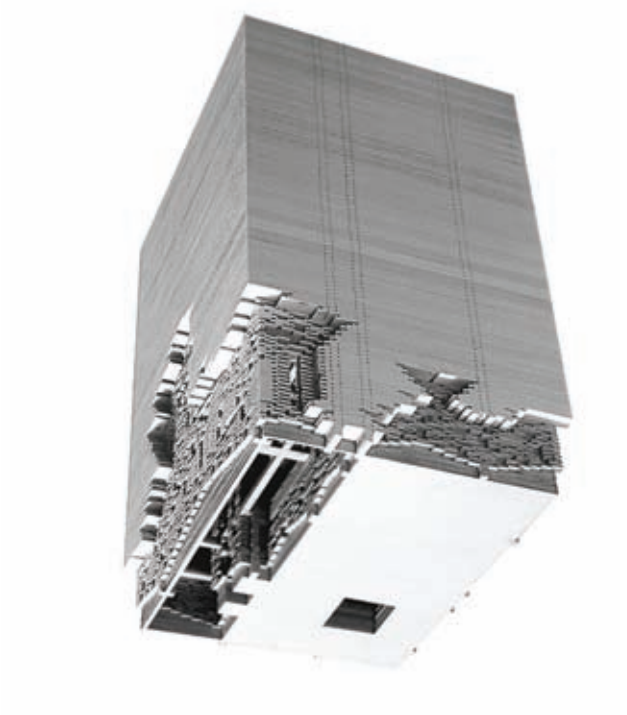
X-Band Antenna, by NASA, 2004
This unexpected antenna design was developed using a genetic algorithm. The fitness function was determined based on performance criteria, and each generation was tested with a software simulator.



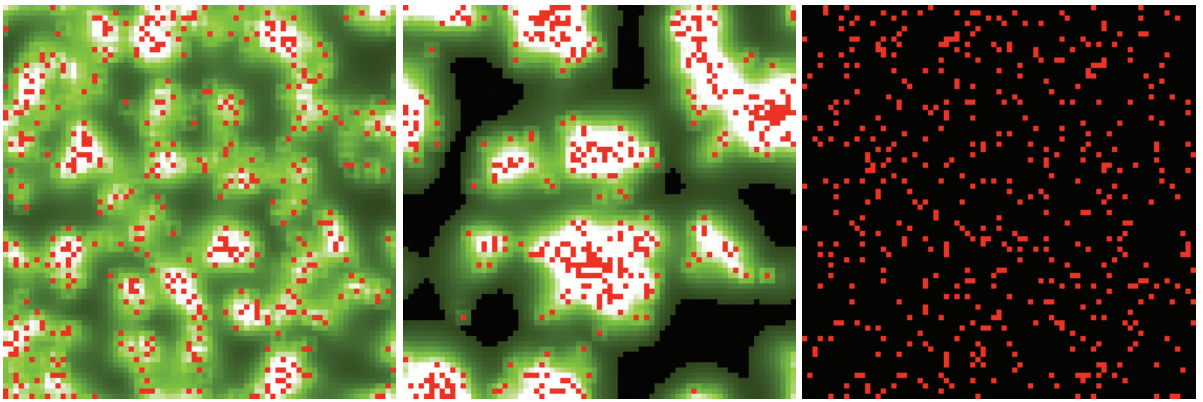
1D Cellular Automata, Rule 110, by Stephen Wolfram, 1983
This set of rules produces what Wolfram categorizes as "class IV behavior." This means that the pattern is neither completely random nor repetitive. Seen as a diagram, the rule for the current pattern is:

111	110	101	100	011	010	001	000	CURRENT PATTERN
↓	↓	↓	↓	↓	↓	↓	↓	
0	1	1	0	1	1	1	0	NEW STATE FOR CENTER CELL

SIMULATION TECHNIQUE
CELLULAR AUTOMATA



A cellular automata (CA) is a grid of cells, with the behavior of each cell defined by a set of rules. CAs are remarkable for their ability to demonstrate unexpectedly complex behavior based on a small group of simple rules. The simplest CAs are of the one-dimensional type invented by Stephen Wolfram in the early 1980s. Each cell is either white or black (also referred to as alive or dead). Over time, the cell becomes white or black by following a few rules. For example, if a cell is white but has one black adjacent cell, it changes to black. Each generation of cells is drawn below the previous generation so one sees the complete history of life and death within a single image. Remarkably, some of the patterns discovered by Wolfram have a striking resemblance to patterns found in nature. CAs have the astounding ability to emulate many properties of living systems within an incredibly constrained set of rules. The most famous CA is John Conway's Game of Life; this 2-D CA has inspired countless projects within the visual arts.



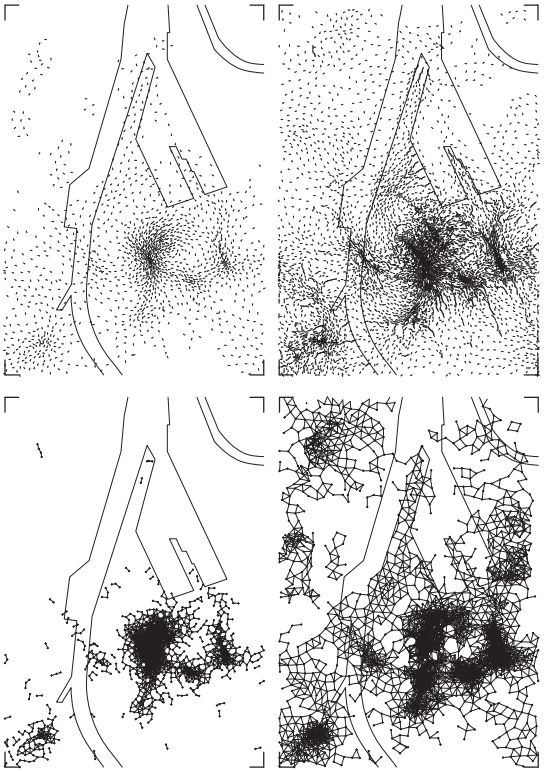
ZZ-IM4 Museum Model, by Mike Silver, 2004
This entry for a competition sponsored by San Jose State University's Museum of Art and Design was designed using AutomasonMP3, a custom software application that generates brick patterns linked to simple concrete structures. AutomasonMP3 also contains a voice synthesizer that enables masons to convey and receive audible block-stacking commands in the field.

StarLogo Slime Mold Aggregation Simulation
This simulation mimics the behavior of slime-mold organisms as they follow pheromones to form groups. Each simulated organism is represented as a red dot, and their pheromones are shown in green.

Aggregation 4, by Andy Lomas, 2005 This form was built from a gradual accumulation of particles on top of an initial surface. In the simulation, millions of particles flow freely until they hit either the initial seed surface or other particles that have been previously deposited.

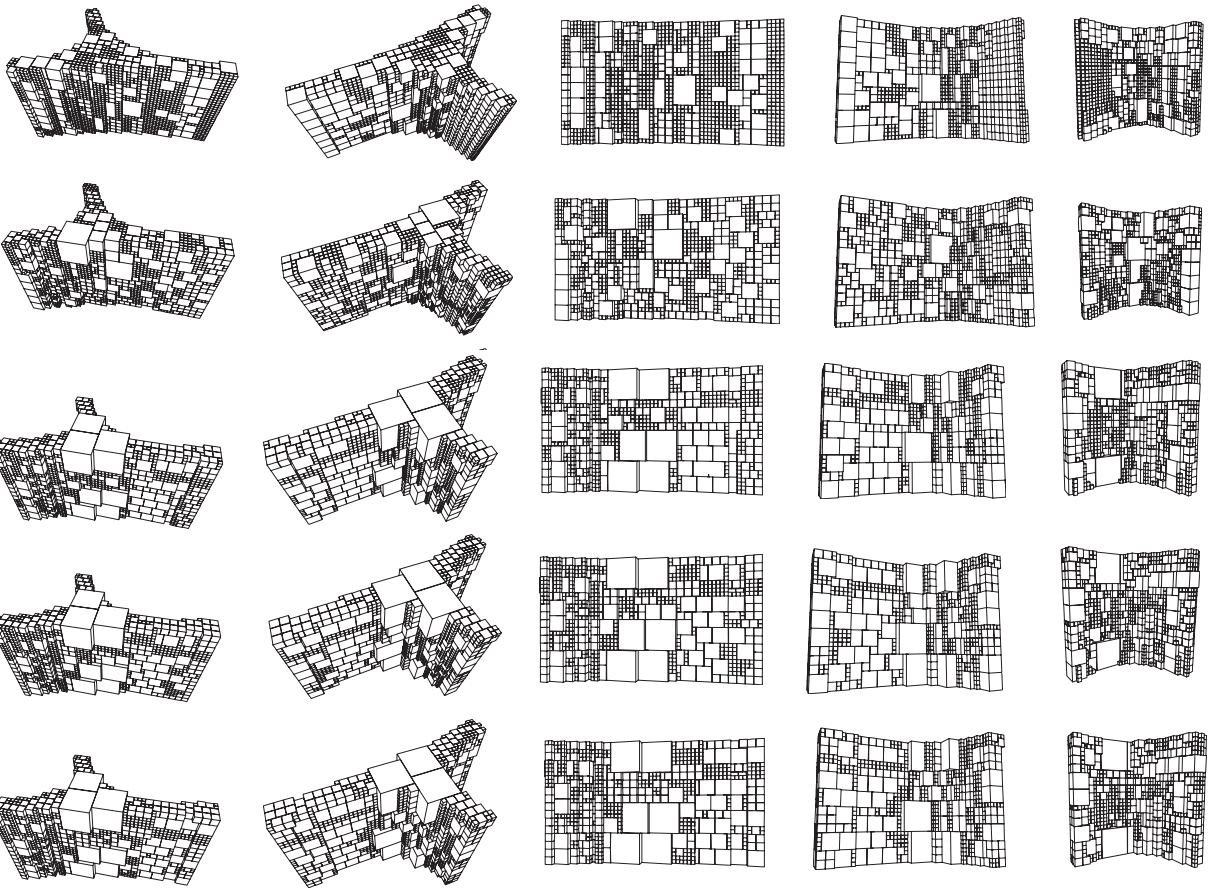
SIMULATION TECHNIQUE
SWARMS

We're all familiar with the surprising phenomenon of a school of fish zipping through the water, turning as a group in a fraction of a second. Software simulations for swarming, flocking, and crowd behavior introduce the ideas of agents to represent each bee, fish, or person. Each agent follows a set of rules that define its behavior in relation to its local environment. For example, in the Boids software, written by Craig Reynolds, each agent follows three clear rules: steer to avoid crowding local flockmates, steer toward the average heading of local flockmates, steer to move toward the average position of local flockmates. The complex swarming patterns seen in the images on this page emerge from these basic rules.



Fox Horror, by Robert Hodgin and Nando Costa, 2007
Using the Boids rules (created by Craig Reynolds) as a foundation, Hodgin wrote a parameterized performance tool for Costa, who then used the tool to choreograph animation and create composites with it using video footage.

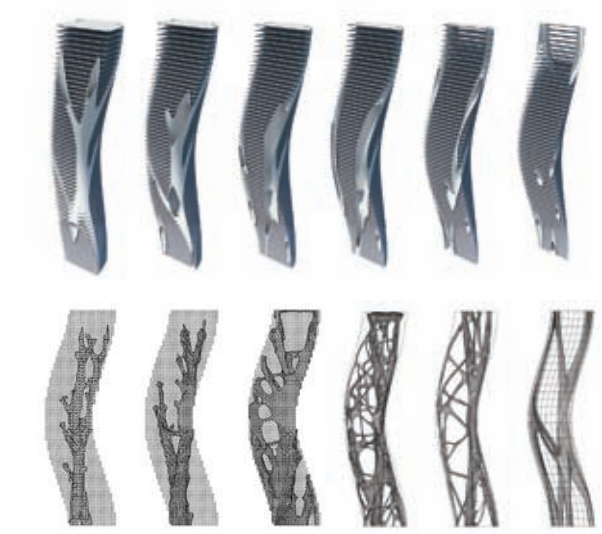
Swarm Urbanism, by Kokkugia, 2008
In an attempt to rethink urban planning, the firm Kokkugia applied the logic of swarms to develop a plan for the Melbourne Docklands. Using swarms in this way, they hope to create a more dynamic plan that is adaptable and better fits the needs of the inhabitants.



The Resolution Wall, by Gramazio & Kohler with the Architecture and Digital Fabrication department at ETH Zürich, 2007

Constructed by a robot, this wall is composed of aerated concrete blocks with a dimension ranging from 5 to 40 centimeters (1.9 to 15.7 inches). Smaller blocks allow for finer detail, but building with larger blocks is faster and therefore more cost effective.

In a student course, a genetic algorithm was developed to evolve a design with a good balance between aesthetic detail, construction time, and structural stability.



SIMULATION TECHNIQUE
UNNATURAL SELECTION

Just as a painter chooses which work to exhibit, or a novelist chooses which sentence to cut, systems that generate form often need some rubric by which to measure success. This can be either implicit (left to the whims of the designer) or explicit (part of the system). Cases where there are clear requirements for performance, structure, or fidelity lend themselves to encoding these requirements directly into the system. In this way, simulations can run autonomously, first generating, then testing, and so on, until an optimal candidate is reached. It is not always possible or desirable, however, to achieve this level of autonomy. Often the designer acts, as William Latham has described it as a "gardener," choosing favorite specimens and culling weeds to encourage the system to grow in a desired direction.

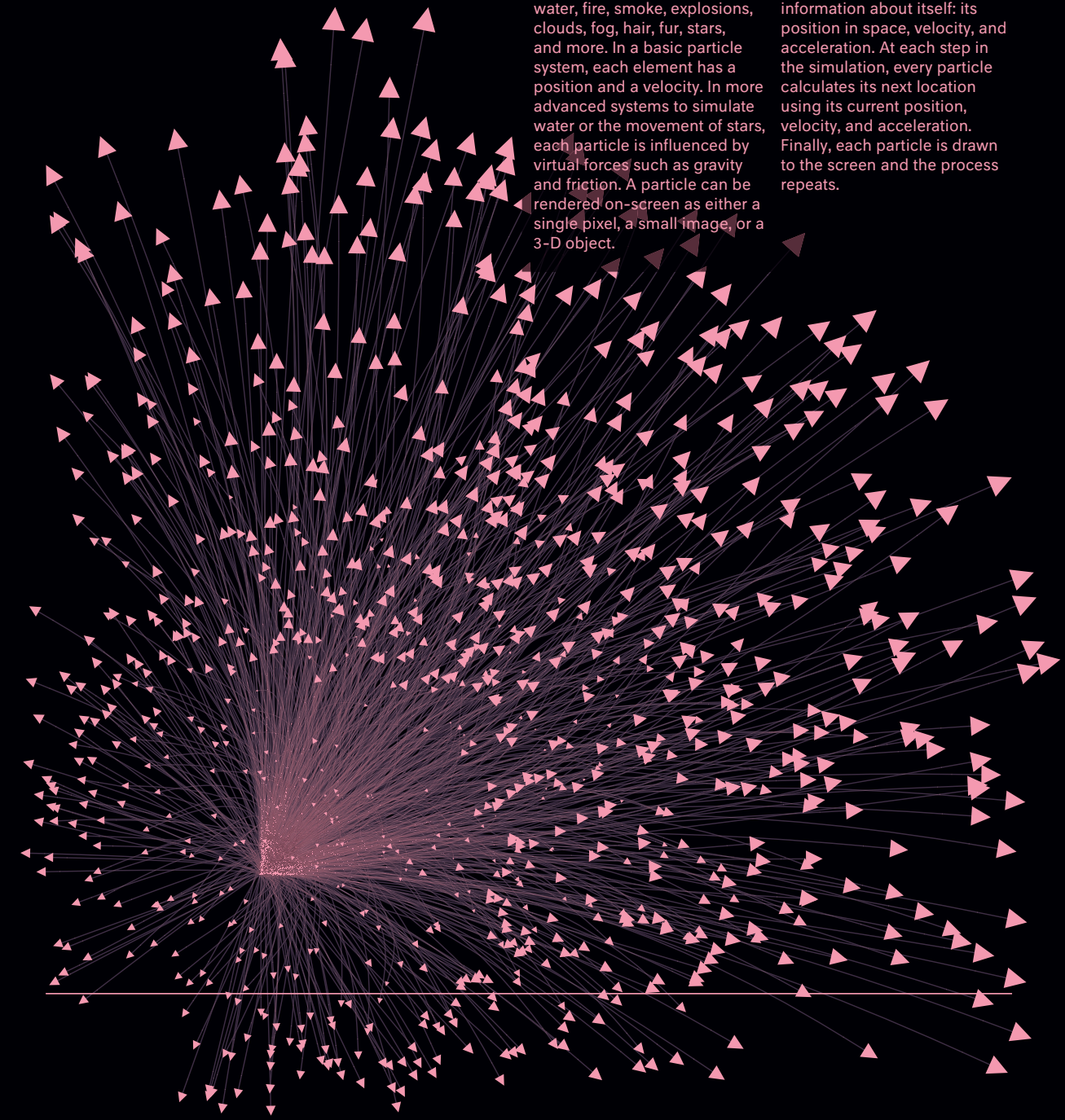
evolutionary computation, by moh architects, 2006
The design of this tower emerged from a process utilizing evolutionary and genetic computing to create a structurally sound form. This technique creates unity of form and structure, and it eliminates the need to post-optimize a design to meet structural demands.

Strandbeest, by Theo Jansen, 1990–present
Jansen has worked for over a decade to create a population of artificial animals to survive unassisted on the beaches of the Netherlands. The skeletons of these creatures are completely constructed of plastic tubing, are powered by wind, and they can secure themselves to the ground in a storm. A genetic algorithm was written to optimize the lengths of the bones.

CODE EXAMPLES
PARTICLES

Particle systems are an essential technique for simulating nature. These systems can replicate the look and behavior of water, fire, smoke, explosions, clouds, fog, hair, fur, stars, and more. In a basic particle system, each element has a position and a velocity. In more advanced systems to simulate water or the movement of stars, each particle is influenced by virtual forces such as gravity and friction. A particle can be rendered on-screen as either a single pixel, a small image, or a 3-D object.

This basic particle system is created by making multiple copies of a simple particle. Each particle stores some information about itself: its position in space, velocity, and acceleration. At each step in the simulation, every particle calculates its next location using its current position, velocity, and acceleration. Finally, each particle is drawn to the screen and the process repeats.



CODE EXAMPLES
DIFFUSION-LIMITED AGGREGATION

Diffusion-limited aggregation (DLA) is a process for generating organic forms from a few simple rules. Particles moving through space, typically in a pattern called a random walk, stick together when they collide. The form is built up over time as more and more particles collide and clump together. The aggregate form often has a complex, branching structure.

The process begins with a fixed seed. Next, virtual particles are created and begin to move through the space. The motion of each particle is created by choosing a new random direction and moving a short distance at each step of the simulation. When the particle moves, it checks to see if it has collided with the seed or another fixed particle. If it collides with either, it stops moving and becomes part of the growing form.

Download the code at http://formandcode.com