40
RANDOMNESS

GAMES OF CHANCE
RANDOMNESS BEFORE COMPUTING
RANDOMNESS COMES TO COMPUTING
COMPUTATIONAL RANDOMNESS IN THE ARTS
THE COMMODORE 64 RND FUNCTION

An essential element of `10 PRINT` is randomness; the program could not produce its mesmerizing visual effect without it. This randomness comes by way of `RND`, a standard function in BASIC. `RND` has been part of the BASIC lexicon since the language's early days at Dartmouth. What the function does is easily characterized, yet behind those three letters lie decades, even centuries, of a history bound up in mathematics, art, and less abstract realms of culture. This chapter explores randomness in computing and beyond. The role of randomness in games, literature, and the arts is considered, as are the origins of random number generation in modern mathematics, engineering, and computer science. Also discussed is the significance of "pseudorandomness"—the production of random-like values that may appear at first to be some sad, failed attempt at randomness, but which is useful and even desirable in many cases. The chapter argues that the maze pattern of `10 PRINT` is entwined with a complex history of aesthetic and utilitarian coin flips and other calculations of chance.

Since a random occurrence is "hap," the root of happy, it might seem that "random" would have a happy etymology. But this is not so. In centuries past, before the philosophers and mathematicians in the Age of Enlightenment sought to rationalize chance, randomness was a nightmare. Likely ancestors of the word "random" are found in Anglo-Norman, Old French, and Middle French and include *randoun, raundun, raundoun, randon, randun,* and *rendon*—words signifying speed, impulsiveness, and violence. These early forms are found beginning around the twelfth century and probably derive from *randir,* to run fast or gallop ("random, n., adv., and adj." 2011). Bumper stickers implore drivers to "practice random acts of kindness," but only because people in our culture fear random acts of violence so much that this phrase has become ingrained and can be punned upon—and at a deeper level, perhaps, because the speed and violence of other vehicles are to be feared. While in recent days it might be harmless to encounter "a random" sitting in the computer lab exploring a system at random, a "random encounter" centuries ago was more likely to resemble a random encounter in *Dungeons & Dragons*: a figure hurtling on horseback through a village, delivering death and destruction.

Only recently have the meanings of the word "random" coalesced around science and statistics. The history of this word is strewn with obsolete meanings: the degree of elevation of a gun that maximizes its range; the direction of a metallic vein in a mine; the sloping board on the top

of a compositor's frame where newly arranged pages are stored before printing. These particular randoms kill opponents, create wealth, or help assemble texts. The `RND` command in `10 PRINT` selects one of two graphical characters—a kind of textual composition that recalls the last of these meanings of random. `10 PRINT`'s random is a flip or flop, a symbol like a slash forward or backward (but fortunately less fearsome than the horseman's random slash). The program splays each random figure across the screen using the `PRINT` command, another echo of the printing press and a legacy of the early days of BASIC, when `PRINT` literally meant putting ink on paper. Although `RND` on the Commodore 64 may seem remote from these early meanings of "random," there are, beneath the surface, connections to speed, violence, devastation, and even printing.

GAMES OF CHANCE

Life itself is full of randomness and the inexplicable, and it is no small wonder that children and adults alike consciously incorporate chance into their daily lives, as if to tame it. Games of chance are one of the four fundamental categories of games that all humans play, according to the French cultural historian Roger Caillois. Whereas *agon* are competitive games dependent upon skill, games of *mimicry* are imaginative, and *ilinx* are games causing disorder and loss of control, the *alea* are games of chance. Craps, roulette, the lottery—these are some of the games in this category, ones with unpredictable outcomes. Taken from the Latin name for dice games, *alea* "negates work, patience, experience, and qualifications" (Caillois 2003, 17) so that everything depends on luck. In Latin, the *āleātor* is a gambler; in French, *aléatoire* is the mathematical term for random.

The Appeal of the Random

In his *Arcades Project* on nineteenth-century Paris, Walter Benjamin devotes an entire section to dice games and gambling, a curious assemblage of notes and excerpts from sources ranging from Casanova to Friedrich Engels. "Gambling," Anatole France is quoted as saying, "is a hand-to-hand encounter with Fate" (Benjamin 1999, 498 [O4A]). Every spin of the roulette wheel is an opportunity to show that fate smiles upon the player.

Fortunes rise and fall in the blink of an eye, the roll of the die, or the cut of the cards. Every gambler knows this, accepts it, and even relishes it.

The allure of gambling—and more generally, the allure of chance in all games—rests on uncertainty. Uncertainty is so compelling that even otherwise skill-based games usually incorporate formal elements of chance, such as the coin toss at the beginning of a football game. As Katie Salen and Eric Zimmerman put it, uncertainty "is a key component of meaningful play" (2004, 174). Once the outcome of a game is known, the game becomes meaningless. Incorporating chance into the game helps delay the moment when the outcome will become obvious.

Consider the case of George Hurstwood in Theodore Dreiser's *Sister Carrie*, first published in 1900. Driven by "visions of a big stake," Hurstwood visits a poker room:

> Hurstwood watched awhile, and then, seeing an interesting game, joined in. As before, it went easy for awhile, he winning a few times and cheering up, losing a few pots and growing more interested and determined on that account. At last the fascinating game took a strong hold on him. He enjoyed its risks and ventured on a trifling hand to bluff the company and secure a fair stake. (Dreiser 1981, 374)

What is intriguing about Dreiser's account is that it is only when Hurstwood's good fortune wavers that his interest in the game grows and he begins to enjoy it. Losing a few hands makes a winning streak that much more thrilling. "A series of lucky rolls gives me more pleasure than a man who does not gamble can have over a period of several years," Edouard Gourdon avers in one sexually charged extract in the *The Arcades Project*. "These joys," he continues, "vivid and scorching as lightning, are too rapid-fire to become distasteful, and too diverse to become boring. I live a hundred lives in one" (Benjamin 1999, 498 [O4A]).

Unlike the early, purely malevolent associations of randomness described in the beginning of this chapter, randomness here involves the masochistic interplay between pleasure and pain. There is also a monumental compression of time: a hundred lives in one. Anatole France calls gambling "the art of producing in a second the changes that Destiny ordinarily effects only in the course of many hours or even many years" (Benjamin 1999, 498 [O4A]). Benjamin himself declares that "the greater the component of

chance in a game, the more speedily it elapses" (512 [O12A,2]). Waiting, boredom, monotony—these frustrations disappear as "time spills from his [the gambler's] every pore" (107 [D3,4]).

Forms of Randomness

Perhaps Benjamin describes games of chance with a bit more whimsy than is useful for critical discussion of the role of randomness in culture. Although words like randomness, chance, and uncertainty may be casually interchanged, not all forms of chance are actually the same. To highlight distinctions between various forms of chance, consider the anthropologist Thomas Malaby's account of gambling in a small Greek city on the island of Crete—an appropriate site of exploration, given *alea*'s Greek etymology. Malaby's goal is to use gambling as a "lens through which to explore how social actors confront uncertainty in . . . key areas of their lives" (2003, 7). How do people account for the unaccountable? How do we deal with the unpredictable? And what are the sources of indeterminacy in our lives?

Malaby presents a useful framework for understanding indeterminacy based on four categories. The first category is *formal indeterminacy*, or what is commonly referred to as chance. This is any form of random allotment, which often can be understood and modeled through statistical methods. Malaby argues that the ascendancy of statistical thinking in the social sciences has so skewed our conception of indeterminacy in gambling (in particular) and in our lives (in general) that formal indeterminacy has become a stand-in for other types of indeterminacies. The second category is *social indeterminacy*, the impossibility of knowing or understanding someone else's point of view or intentions. A bluff is a type of social indeterminacy. The third category is *performative indeterminacy*, that is, the unreliability of one's own or of another's actions, say a fumble in football game or misreading the information in plain view on a chessboard. Finally, the fourth category Malaby describes, *cosmological indeterminacy,* refers to skepticism about the fairness and legitimacy of the rules of the game in the first place at a local, institutional, or cosmological level. Suspicion that a game is rigged, for example, is concern about cosmological indeterminacy (Malaby 2003, 15–17).

Privileging of the stochastic principles of formal determinacy means that players, scholars, and even programmers dismiss social and performa-

tive indeterminacies altogether. In the case of `10 PRINT`, thinking about social indeterminacy can reveal several new layers of randomness, such as the idiosyncratic line numbers in the 1982 and 1984 versions of the program. Likewise, understanding performative indeterminacies may account for the textual variants of the program, for example, the version that appeared in the online publication *Commodore Free* that will not actually execute as printed (Lord Ronin 2008).

Cosmological indeterminacy is perhaps the most difficult form of indeterminacy to apply to `10 PRINT`. The rise of the scientific method can be seen as one enduring struggle to impose a more rational view upon the world and to abolish cosmological indeterminacy. From Aristotle to Galileo to Newton, classical mechanics defined the universe as an organized system without random actions. Einstein declared that "God does not play dice with the universe." Yet, as a closer examination of randomness on the Commodore 64 will reveal, there is evidence that randomness on this computer—and indeed, on any computer—is fundamentally "rigged" in a way that echoes Malaby's idea of cosmological indeterminacy. Randomness and chance operations are so necessary to daily life, well beyond the realm of games, that randomness itself is framed as fixed, repeatable, and knowable.

RANDOMNESS BEFORE COMPUTING

Just as the different categories of indeterminacy in games are often grouped together and called "chance," so too in the visual arts, music, and other aesthetic practices is the word "chance" used instead of "randomness." In his chapbook *Chance Imagery*, the conceptual artist George Brecht (1966) describes two distinct types of chance operations by which an artist might create a work: "one where the origin of images is unknown because it lies in deeper-than-conscious levels of the mind" and a second "where images derive from mechanical processes not under the artist's control." The first definition describes the work of the Surrealists and Abstract Expressionists, who sought to allow subconscious processes to dictate their work. The second definition is reminiscent of Dada and closer to the typical concept of randomness in computing; it describes the mechanical operations of the artists most directly connected to `10 PRINT`. These two senses are worth noting

because it is difficult to pull on one of the two senses of "chance" without the other one—the unconscious, in this case—at least feeling a tug.

The tension between these two chance operations is captured in William Burroughs's story about a Surrealist rally in the 1920s. Tristan Tzara suggested writing a poem "on the spot by pulling words out of a hat," and as Burroughs tells it, "a riot ensued" and "wrecked the theater." In his version of events, André Breton, the leading Surrealist, expelled Tzara from the group, his purely mechanistic chance operation being an affront to the power and vagaries of the Freudian unconscious (Burroughs 2003). Burroughs is most certainly conflating several events, and the break between Surrealism and Dada had as much to do with a personality clash between Breton and Tzara as with their approaches to art (Brandon 1999, 127). Burroughs himself clearly preferred the anarchic mode of Tzara and famously described a similarly unpredictable mode of composition, the cut-up method, also proposed by Tzara in his 1920 "To Make a Dadaist Poem." Burroughs explains that "one way to do it" is to cut a page in four quarters and then rearrange the sections: "you will find that it says something and something quite definite" (90). Tzara suggests pulling words blindly from a bag. The generative possibilities of this cut-up technique resemble the collage in art and the montage in film, and have become far more mainstream today than Tzara might have imagined in 1920. For instance, Thom Yorke, the lead singer for the band Radiohead, wrote the lyrics to "Kid A" in 1999 by pulling fragments of text out of a top hat.

Chance Operations

Though Yorke employed a type of cut-up method to address severe writer's block, artistic experimentation with randomness in the early part of the twentieth century can be seen as a response to the sterile functionality of rationality and empiricism wrought by the Industrial Age and as a deliberate reaction against World War I. Consider Marcel Duchamp's *Three Standard Stoppages* (1913–1914). According to his description of the piece, Duchamp dropped three meter-long pieces of string from the height of one meter and let gravity and chance dictate the paths of the twisting string downward. Then he adhered the twisted string onto canvas, the shape and length of which he preserved in 1918 in wooden cutouts, creating three new "stoppages" that parodied the supposed rationality of the

meter. When Duchamp described his method in 1914, he observed that the falling thread distorts "itself as it pleases" and the final result becomes "the meter diminished," subverting both the straightness and the length of what commonly goes unquestioned (Duchamp 1975, 141–142). On his use of randomness, Duchamp said, "Pure chance interested me as a way of going against logical reality" (Cabanne 1971, 46).

Duchamp, like the other Dada artists with whom he associated, saw "logical reality" as a failure, epitomized by the horrors of World War I. Satire, absurdity, and the embrace of indeterminacy seemed to the Dadaists to be the most "reasonable" response to modernity. In the words of the Dada artist Jean (Hans) Arp, "Dada wished to destroy the reasonable frauds of men and recover the natural, unreasonable order. Dada wished to replace the logical nonsense of the men of today with an illogical nonsense." To Arp, individual authorship was synonymous with authoritarianism and random elements were used to liberate the work (Motherwell 1989, 266).

The major twentieth-century composer to explore randomness was certainly John Cage, who was strongly influenced by Duchamp. From Cage's point of view, random elements remove individual bias from creation; they may be used to reach beyond the limitations of taste and bias through "chance operations." Cage influenced generations of artists through his compositions as well as through his writing, lectures, and classes. In his text "Experimental Music," Cage wrote, "Those involved with the composition of experimental music find ways and means to remove themselves from the activities of the sounds they make. Some employ chance operations, derived from sources as ancient as the Chinese *Book of Changes*, or as modern as the tables of random numbers used also by physicists in research" (1966, 10).

Cage's method of random composition was to create a system of parameters and then leave the results to circumstance. Cage explained, "This means that each performance of such a piece of music is unique, as interesting to its composer as to others listening. It is easy to see again the parallel with nature, for even with leaves of the same tree, no two are exactly alike" (1996, 11). Random components are used to transform a single composition into a space of potential compositions. Over the decades, Cage used an array of techniques to insert unexpected elements into his compositions. He defines the range of techniques he and his contemporaries used in the 1958 lectures "Composition as Process." There are generally two

methods for using random values in music: to define the work at the time of composition or to allow for variation when the work is performed. The most obvious use of randomness in `10 PRINT` is in the second category as random decisions are made during the program's execution—that is, while the BASIC instructions are performed by the Commodore 64.

Within two-dimensional visual art, artists also explored mechanical random processes for reasons championed by Cage. The eminent contemporary painter Gerhard Richter provided a simple answer to this method's benefits when he said, "I'm often astonished to find how much better chance is than I am." There are precedents for chance used within visual works dating back to collage works by Arp from 1916, but the two early works most relevant in the discussion of `10 PRINT` are the *Spectrum of Colors Arranged by Chance* collage series (1951) by Ellsworth Kelly and *Random Distribution of 40,000 Squares Using the Odd and Even Numbers of a Telephone Directory* (1961) by François Morellet. These works start with an even grid and fill the grid carefully with elements based on the algorithms developed by the artists. Kelly uses squares of colored paper, placed according to a system he designed. He assigned a number to each color and plotted the numbers on the grid systematically (Malone 2009, 133). Morellet employed a stricter system, reading a series of numbers from the telephone book. He made a grid of 200 vertical and horizontal lines, painting a square blue if its assigned number is even, painting it red if it is odd. In both of these artworks and in `10 PRINT`, the structure of the grid is what makes it possible to focus on the variability created through the random operations.

A Million Random Digits

The need for large batches of random numbers is so acute that there are standardized collections of them. In Deborah Bennett's history of humans' quest for randomness—which she suggests goes as far back as ancient Babylonia (1998, 17)—she highlights one of the earliest and largest sets of random numbers, *A Million Random Digits with 100,000 Normal Deviates* (135). This series of numbers (figure 40.1) was generated in 1947 from "random frequency pulses of an electronic roulette wheel" by the RAND Project, a research and development think tank that would eventually become the RAND Corporation. The 1955 publication of the series in book

form was an important contribution to any study of probability; the book is still in use today. As the forward to the undated online edition of the table notes:

> The tables of random numbers in the book have become a standard reference in engineering and econometrics textbooks and have been widely used in gaming and simulations that employ Monte Carlo trials. Still the largest known source of random digits and normal deviates, the work is routinely used by statisticians, physicists, polltakers, market analysts, lottery administrators, and quality control engineers. (RAND Corporation 1955)

Considering its sophisticated origins and uses, *A Million Random Digits* proposes a surprisingly unscientific method of using the book: "In any use of the table, one should first find a random starting position. A common procedure for doing this is to open the book to an unselected page of the digit table and blindly choose a five-digit number." The RAND report goes on to somewhat ominously explain that its one million random numbers were originally "prepared in connection with analyses done for the United States Air Force." Like so many other advances in computing, randomness, it turns out, is intimately linked to Cold War military strategies. In fact, most of the early work on computer-based random number generation was performed under the auspices of the U.S. Atomic Energy Commission see, for example, Rotenberg's [1960] work in the late 1950s) or the U.S military (see Green, Smith, and Klem's [1959] work at MIT, done with joint support of the U.S. Army, Navy, and Air Force).

RANDOMNESS COMES TO COMPUTING

The `RND` command acts as the algorithmic heart of `10 PRINT`, its flip-flopping beat powering the construction of the maze. The `RND` function is as fully specified as any BASIC keyword, but its output is, by that definition, unpredictable. Mathematicians and computer scientists don't think in terms of predictability, though; rather, the standard mathematical treatment of randomness defines randomness in terms of probability. A random process generates a sequence of values selected from a set of possible values ac-

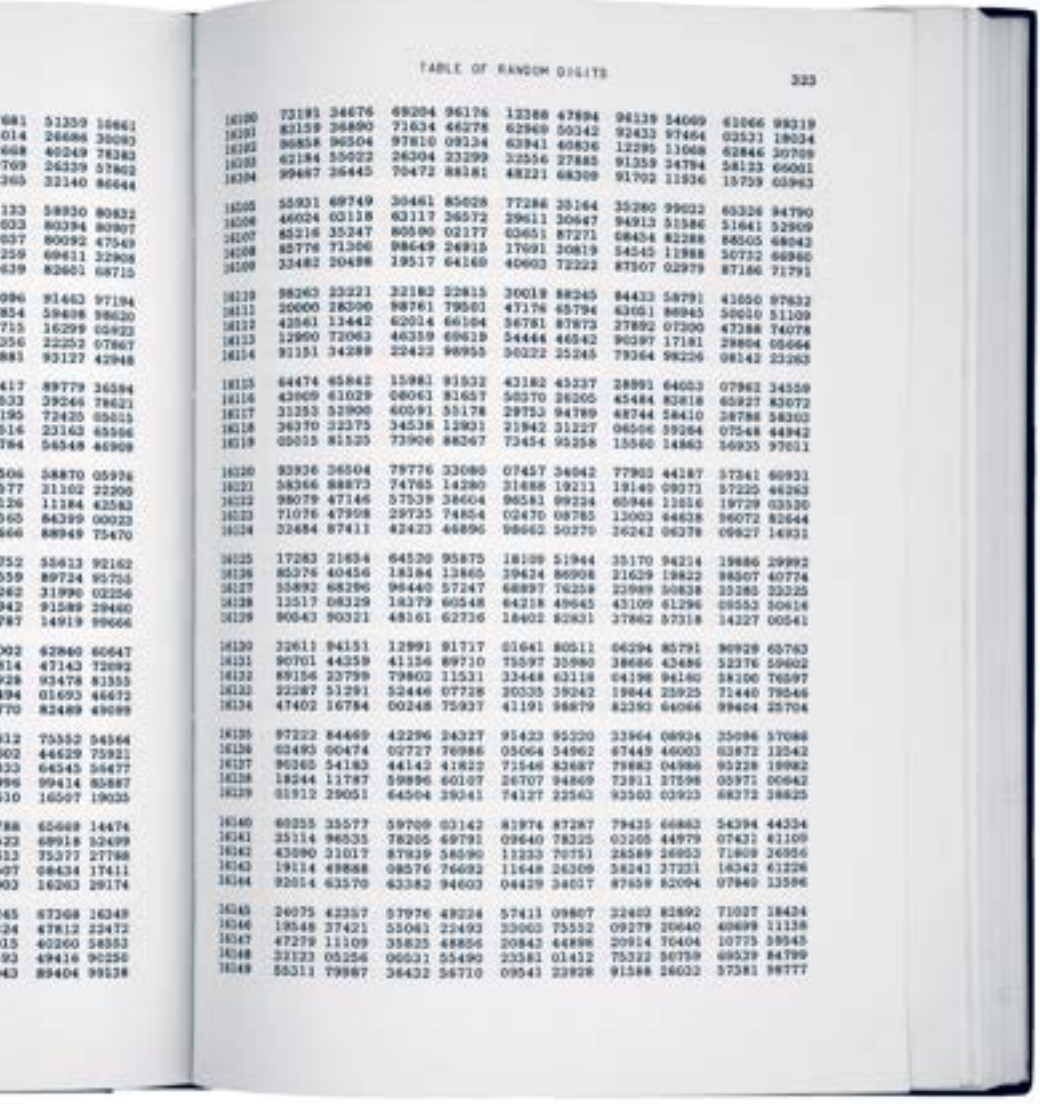

TABLE OF RANDOM DIGITS 323

16100 73191 34676 69204 96176 12388 47894 98139 54069 61066 98319
16101 82159 26890 73634 46278 62969 00342 92433 97464 02531 18034
16102 86858 96504 97810 09134 63941 40836 12295 13068 62846 30709
16103 62184 55022 26304 23299 32556 27885 91359 34794 58123 66001
16104 99467 36445 70472 88181 48221 68309 91702 11936 15759 03963

16105 55931 69749 30461 85028 77286 30164 35280 99022 65326 94790
16106 46024 03118 63117 26572 29611 30967 94913 51586 51641 52909
16107 85216 35247 80590 02177 63651 87271 58454 82288 86505 68043
16108 85776 71306 98649 26915 17691 30819 54545 11988 50732 66980
16109 33482 20498 19517 64160 40603 72222 87507 02979 87186 71791

16110 98263 23221 32182 22815 30019 88245 84433 58791 41050 97832
16111 20000 28300 98761 79501 47176 65794 63051 86945 50010 51109
16112 42561 13442 62014 66104 36781 87873 27892 07300 47288 74078
16113 12990 72063 46359 69619 54444 46542 90297 17181 29804 05664
16114 91351 34289 22422 98955 30322 25245 79364 98226 98142 23263

16115 64474 65842 15981 91532 43182 45237 28891 64003 07962 34559
16116 43069 61029 08061 81657 58370 26205 45484 83818 65927 83072
16117 31253 52900 60091 55178 29753 94789 48744 58410 38788 58303
16118 36370 32275 34538 12931 21942 31227 06506 59284 07548 44942
16119 09015 81525 73906 88267 73454 95258 15560 14863 36935 97011

16120 93936 36504 79776 33080 07457 36042 77903 44187 37361 80931
16121 58366 88873 74765 14280 31488 19211 19140 09371 57225 46263
16122 98079 47146 57539 38604 96581 99224 60946 13016 19729 03530
16123 71076 47908 29735 74854 02470 08785 13003 64638 96072 82644
16124 22484 87411 42423 46896 98062 50270 26242 06378 09827 14931

16125 17283 21654 64520 95875 18109 51944 35170 94214 19886 29992
16126 85376 40456 18184 13865 39424 86908 21629 19822 98507 40774
16127 53892 68296 98440 57247 68897 76259 33989 50838 33385 33225
16128 13517 08329 18379 60548 64218 49645 43109 61296 09553 30416
16129 90043 90321 48161 62736 18402 82831 37862 57318 14227 00541

16130 32611 94151 12991 91717 01641 80511 06294 85791 90929 65763
16131 90701 44259 41156 89710 75597 33980 38686 43486 52376 59602
16132 89156 22799 79802 11531 33448 63118 04198 94160 38100 76097
16133 22287 51291 52446 07726 20035 39242 19844 25925 71440 79546
16134 47402 16784 00248 75937 41191 98879 82393 64066 89404 25704

16135 97222 84469 42296 24327 91423 95220 33964 08924 35096 57088
16136 03493 00474 02727 76986 05064 54962 67449 46003 03872 12542
16137 96365 54183 44143 41822 71546 83687 79883 04986 95228 19982
16138 18244 11787 89896 60107 26707 94869 72811 27598 05971 00942
16139 01912 29051 64504 39341 74127 22563 93503 03923 68272 38825

16140 60255 35577 59709 03142 81974 87287 79435 66863 54394 44334
16141 35114 96535 78205 69791 09640 78325 03205 44979 07431 41109
16142 43090 31017 87939 58090 11233 70751 28589 26853 71808 36956
16143 19114 49888 08576 76692 11648 26309 38243 37231 16342 61226
16144 92014 63570 63382 94603 04429 34017 87659 82094 07840 13586

16145 26075 42357 57976 49224 57411 09807 22403 82892 71027 18424
16146 19548 37421 55061 22493 33003 75552 09279 20640 40699 11138
16147 47279 11109 35825 48856 20842 64898 20914 70404 10775 59545
16148 33133 05256 00031 55490 23581 01412 75322 30759 69529 84799
16149 55311 79987 36432 56710 09543 23928 91588 26032 57381 98777

Figure 40.1

A Million Random Digits with 100,000 Normal Deviates was published in 1955 by the RAND Corporation and was the largest list of random values yet published. It was necessary for RAND to execute their research without repeating values from previously published, smaller number tables.

cording to a probability distribution. In the case of a discrete distribution (heads or tails, for instance), the distribution explains how much weight is on each possible outcome—how likely that value is to appear.

If, for example, one draws a single card from a thoroughly shuffled deck, the probability distribution from which this draw is done is uniform: it is equally likely that any particular card will be chosen. Similarly, random numbers are typically defined as numbers drawn from a uniform distribution over all possible numbers in some range. A difficulty with this definition is that the randomness of a number is defined in terms of that range. Given a number such as 42, it is impossible to tell how random a selection it was. To determine randomness without knowing the means of generation, one must consider a sequence of numbers; knowing the range in which the numbers are supposed to lie or, more generally, the distribution from which they are supposed to be drawn, is also essential.

Digital computers are deterministic devices—the next state of the machine is determined entirely by the current state of the machine. Thus, computer-based random number generators are more technically described as pseudorandom number generators. The somewhat dismissive-sounding "pseudo" refers to the fact that a deterministic process (a computer program) is being used to generate sequences of numbers that appear to be uniformly distributed. This works well in practice for sequences that aren't astronomically long. But eventually, for long enough sequences, the deterministic nature of a pseudorandom number generator will be unmasked, in that eventually statistical properties of the generated sequence will start diverging from those of a true random process. In an extremely long sequence, for example, a true random process will generate the same number many times in a row. A version of `10 PRINT` running using a true random process will eventually generate the regular image in figure 40.4 (and the image in figure 40.5, and every other possible pattern), while the pseudorandom number generator in the Commodore 64 will not. Tests for long runs are one of the many statistical tests used to judge the quality of pseudorandom number generators.

An obvious question to ask about randomness is why a computer would need to implement it in any form. Chance might produce stunning poetry, breathtaking art, uncanny music, and compelling games, but what is its role in the sciences? Why provide a calculating machine with the ability to generate random numbers in the first place? Certainly, one stereo-

type of computing is that it is done exactly, repeatedly, with perfect precision and accuracy. Computers are commonly thought to order the world, to sift through reams of data and then model possible outcomes, possible futures, providing certain—and deterministic—answers. Yet a function to generate random numbers was present in the first Dartmouth BASIC. Every version of BASIC since then has had one or more ways to create random numbers. Nearly every contemporary programming language, including Python, Perl, Java, JavaScript and C++, has a built-in way to generate randomness.

Quite simply, the answer to this puzzle is that randomness is necessary for any statistical endeavor, any simulation that involves unknown variables. Practically *everything* involves unknown variables: the meteorological conditions at a rocket launch site, the flow of air under a bomber's wings, and the spread of an infectious disease. Additionally, there is the movement and halting of traffic, the cost of bread, and the drip of water from the kitchen faucet. Forecasting any of these phenomena requires reckoning with uncertainty, which in turn requires a pool of random numbers. Furthermore, one or two random numbers are not enough. Large-scale statistical calculations or simulations require large batches of random numbers.

John von Neumann was the first to propose the idea of harnessing a computer to generate random numbers (Knuth 1969, 3). It was around 1946 and von Neumann was fresh off the Manhattan Project and soon to begin his lead work on the hydrogen bomb. Seeking a way to statistically model each stage of the fission process, von Neumann and his colleague Stanislaw Ulam first relied on the Monte Carlo method to generate tables of random numbers. These tables, however, soon grew too large to be stored on computers (Bennett 1998, 138–139). Von Neumann's solution was to design a computer program to produce random numbers on the fly, using the middle-square method. It worked by squaring an initial number, called the seed, and extracting the middle digits; this number was then squared again, and the middle digits provided a new random number (von Neumann 1961). Because each number is a function of the one before it, the sequence, as Donald Knuth explains, "isn't random, but it appears to be" (3)—that is, it is "pseudorandom."

GRAPHING RANDOM MAZES

Randomness has enabled the construction of mazes for decades. These mazes are not grown in a careful arrangement of hedgerows, or built amid the mossy walls of Cretan dungeons. Instead, they are typically graphs, mathematical objects consisting of a set of nodes (also called vertices), pairs of which may be connected with a link (also called an edge). Graphs, or networks, don't need to have any particular geometry. They are simply nodes linked to other nodes, and they can be drawn on paper in many different ways that are correct representations.

Consider, however, a piece of graph paper, blank white except for a regular grid of pale blue lines. Each point where two lines cross can be taken to represent a node, while the lines between these points can define links. This construction, based on a lattice, is a special kind of graph called a grid graph. Using a pencil and tracing only along the pale blue demarcations, how does one draw a maze whose links (hallways) connect all of the nodes (rooms) to each other?

Graph theory, a field of mathematics, offers a number of methods for producing random mazes of this kind. The most well-known approaches are algorithms for calculating a minimal spanning tree, a graph in which all links are connected and with only one simple path between any two points. (Minimum spanning trees are found to solve problems in various domains, from phone networks to demographic analysis.) Because they lack cycles—there is exactly one path between any two nodes—the mazes produced by such trees are called "perfect mazes." Spanning solutions are not always mazes in the multicursal sense; they don't need to have forking paths. For example, on a grid graph, it's possible to create a minimal spanning tree using a single line, winding back and forth on a labyrinthine path until the page is filled. Of the myriad spanning solutions to a piece of graph paper, however, the vast majority of them *are* branching mazes. Thus, selecting a solution at random can be a good way to produce different mazes. A straightforward maze-generation technique involves adding random values (or weights) to all the links in the grid graph, then employing an algorithm to find a minimum spanning tree and thus generate a maze. Depending on the algorithm used, the resulting mazes may reflect different aesthetics, for instance, having different proportions of shorter and longer paths.

Significant minimum spanning tree algorithms were pioneered by Czech mathematicians in the early twentieth century (Otakar Borůvka in 1926; Vojtěch Jarník in 1930) and independently rediscovered many times thereafter, including decades

later by computer scientists writing in English (e.g., Sollin in 1965). Two of the most well-known maze-generating algorithms in graph theory today are Joseph Kruskal's and Robert Clay Prim's. Both algorithms were published in 1957—although Prim's was a rediscovery of Jarník's and was in turn rediscovered by Dutch computer scientist Edsger W. Dijkstra, famous opponent of `GOTO`, in 1959 (Foltin 2011, 15). Both are greedy algorithms, which means that they choose the best link to take at every turn. Kruskal's algorithm chooses across the entire graph, while Prim's algorithm builds up a connected path. These algorithms can be modeled with paper and pencil, but computational randomization allows them to rapidly generate a plethora of maze forms, thanks to the interaction of the regularity of the grid, the deterministic algorithm, and the random weighting of links.

COMPUTATIONAL RANDOMNESS IN THE ARTS

To those interested in randomness and expressive culture, perhaps the most intriguing element of Donald Knuth's magisterial discussion of random numbers appears in a footnote. Knuth recalls a CBS television documentary in 1960 called "The Thinking Machine" which featured "two Western-style playlets" written by a computer (Knuth 1969, 158–160). In fact, three playlets were acted out on national television that day in October 1960, generated by a TX-0 computer housed at MIT's Electronics Systems Laboratory. SAGA II, the script-writing program behind the mini Westerns, took programmers Douglas Ross and Harrison Morse two months to develop and consisted of 5,500 instructions (Pfeiffer 1962, 130–138). The key to SAGA II was its thirty "switches," which made "various alternative or branching paths" possible (136). "Among other things," Pfeiffer observed, "the robber may go to the window and look out and then go to the table, or he may go to the table directly. You cannot tell in advance which one of these alternatives the program will select, because it does the equivalent of rolling a pair of dice" (136).

Even before the SAGA II playlets, there were other literary experiments with randomness and computers. Noah Wardrip-Fruin identifies the

British computer scientist Christopher Strachey as the creator of the first work of electronic literature, a series of "love letters" generated by the Ferranti Mark I computer at Manchester University in 1952 (Wardrip-Fruin 2005). Affectionately known as M.U.C., the Manchester University Computer could produce the evocative love letters at a pace of one per minute, for hours on end, without producing a duplicate. The "trick" is, as Strachey put it, the two model sentences (e.g., "My *adjective noun adverb verb* your *adjective noun*" and "You are my *adjective noun*") in which the nouns, adjectives, and adverbs are randomly selected from a list of words Strachey had culled from Roget's *Thesaurus*. Adverbs and adjectives randomly drop out of the sentence as well, and the computer randomly alternates the two sentences. On the whole, Strachey is dismissive of his foray into the literary use of computers, using the example of the love letters simply to illustrate his point that simple rules can generate diverse and unexpected results (Strachey 1954, 29–30). Nonetheless, a decade before Raymond Queneau's landmark combinatory work *One Hundred Thousand Billion Poems,* Strachey had unwittingly laid the foundation for the combinatory method of composition by computer, a use of randomness that would grow more central to literature and the arts in the following decades.

Other significant early works involving random recombination had more visible connection to literary tradition and artistic movements. The 1959 "Stochastic Texts" of Theo Lutz combined texts from Franz Kafka with logical operations to produce "EVERY CASTLE IS FREE. NOT EVERY FARMER IS LARGE" among other statements (Lutz 1959/2005). In the next decade, Fluxus artist Alison Knowles and James Tenney, a programmer who worked in FORTRAN, devised *A House of Dust.* The program's output combines a regular stanza form and repetition with random variation in vocabulary, and was printed on a scroll of line printer paper for a 1968 chapbook publication (Pearson 2011, 194–203). More than a decade later, Jackson Mac Low made use of the venerable book *A Million Random Digits* to devise "Converging Stanzas," which were randomly populated with words from the 1930 850-word *Basic English Word List* (Mac Low 2009, 236). This poet's "Sade Suit" similarly used playing cards and *A Million Random Digits* to rewrite the work of Marquis de Sade (46).

Early Experiments in Computational Art

The 1960s were a time of radical experimentation with randomness in the visual arts. Even though computers were available at that point for the exploration of chance operations, they were used in a very limited way because it was difficult to gain access to the machines, and there was a general distrust of computer technology in the arts. The `10 PRINT` program is remarkable because it was created later, when these barriers were far fewer. The Commodore 64 was relatively inexpensive and accessible. The public image of the computer was changing from a machine that supported technocracies to a tool for self-empowerment and creativity. Before personal computers, calculating machines could only be found in universities and research labs and, because of their cost and perceived purpose, they were typically used exclusively for what seemed more serious work, not for creating aesthetic images. When artists did gain access to these machines, it was typically through artists-in-residence programs at companies such as Bell Labs and IBM, and through infrastructures such as Experiments in Art and Technology (E.A.T.) based in New York or the Los Angeles County Museum of Art's Art and Technology initiative. Many of the first aesthetic computer graphics were made not by artists, but by mathematicians and engineers who were curious about other uses to which the machines at their labs could be put.

Within the first years that computer images were made, random processes were explored thoroughly. The first two exhibitions of computer-generated graphics appeared in art galleries in 1965; both shows included pieces that were created using random values. In New York, the works of A. Michael Noll and Bela Julesz, both researchers at Bell Labs, were exhibited at the Howard Wise gallery from April 6–24, 1965, under the title "Computer-Generated Pictures." In Stuttgart, the works of Georg Nees and Frieder Nake were exhibited at the Wendelin Niedlich Gallery from November 5–26, 1965, under the title "Computer-Grafik Programme."

In 1962, Noll published a technical memorandum at Bell Labs entitled "Patterns by 7090," the number referring to the IBM 7090 digital computer. He explained a series of mathematical and programming techniques that use random values to draw "haphazard patterns" to a Carlson 4020 Microfilm Printer. The eight patterns documented in the memo are the basis for his Gaussian Quadratic image that was exhibited in the 1965

exhibition. Noll used existing subroutines of the printer to draw a sequence of lines to connect a series of x- and y-coordinates that he calculated and stored inside an array. The x-coordinates in the array were generated by a custom subroutine he wrote called WNG (White Noise Generator), which produced random values within the range of its parameters, and the y-coordinates were set using a quadratic equation. Through this series of patterns, Noll explored a tension between order and disorder, regularity and random values.

In 1965, Nake created his *Fields of Rectangular Cross Hatchings* series, which succeeds through pairing ordered patterns with random placement (figure 40.2). Nake explained the way random values are used in the images:

> Within a given (arbitrarily chosen) image size, a random number of hatchings were generated. Each one of them was determined by the following random variables: location (x, y), size (a, b), orientation of lines within rectangle (horizontal or vertical), number of lines, pen. So for each rectangle there were seven random numbers determining its details. (Nake 2008)

After the first wave of visual images were created on plotters and microfilm at universities and research labs, a few professional artists independently started to gain access to computers and use them in their practice. The artists with the most success integrating a computer into their work had previously created drawings using formal systems. These artists continue to use computers in their work to this day. Artists who worked seriously with computers in the late 1960s, either individually or with technical collaborators, include Edward Zajec, Lillian Schwartz, Colette Bangert, Stan Vanderbeek, Harold Cohen, Manfred Mohr, and Charles Csuri. All of them employed random numbers in their early works created with software.

Manfred Mohr, for example, started as a jazz musician and later studied art in Paris; he began writing software to create drawings in 1969, at the Meteorological Institute of Paris, during the night after researchers had left for the day. In 1971, Mohr's work was featured in "Une Esthétique Programmée" at the Musée d'Art Moderne de la Ville de Paris (see figure 40.3), the first solo exhibit of artworks created with a computer at a museum. Random values are used extensively in the creation of the work shown.

Figure 40.2
Frieder Nake, *Fields of Rectangular Cross Hatchings, Overlaid by Vertical Lines. 22/10/65 Nr. 2.* Computer drawing, ink on paper, 50 × 44 cm. Collection Etzold, Museum Abteiberg Mönchengladbach. Courtesy of Frieder Nake. ©1965, Frieder Nake.

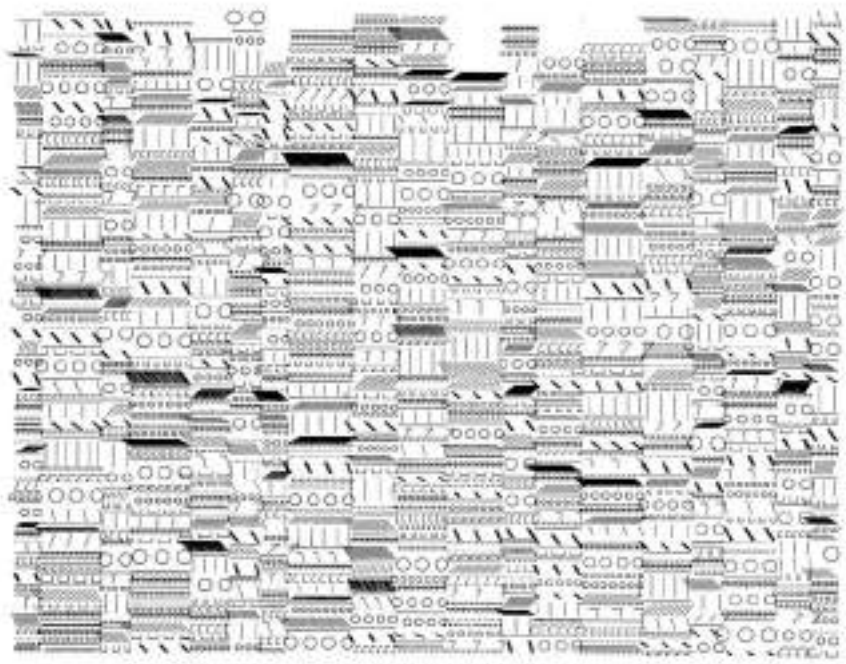

Figure 40.3
Manfred Mohr, *P-071*, 1970. Plotter drawing, ink on paper, 13.75 × 16.5" / 35 × 42 cm. Courtesy of bitforms gallery nyc. ©1970, Manfred Mohr.

Charles Csuri's *Random War* (1967) is an early notable work of computer art to use random values. Like much of Csuri's early computer work and unique in relation to his contemporaries, *Random War* is figurative rather than abstract. This plotter drawing comprises outlined military figures, patterned off of the toy figures of little green army men that were popular at the time. Each figure, named after a real person, is placed randomly on the page and randomly given a status: dead, wounded, or missing. The soldiers of one army are drawn in red, of the other army in black; the name and status of each soldier appear at the top of the drawing. In general terms, Csuri's work comments on the often arbitrary nature of war through both its form and its content; more specifically, with his reliance on random number generation, Csuri gestures toward the days of computers, random numbers, and their inextricable link to the Cold War.

 `10 PRINT CHR$(205.5+RND(1)); : GOTO 10`

Acceptance and Resistance

While the first decade of computer-generated art was well documented in magazines, books, and exhibition catalogues, there are fewer source materials from the 1970s, when public interest veered and the energy needed to publish and exhibit waned. Later in the decade, computer graphics started to make their way into advertising and films. The 1982 film *Tron* is a landmark in the history of computation and aesthetics that pushed graphics to a new aesthetic level and therefore revealed the limitations of computer imagery at that time. *Tron*'s images are purely geometric and cold; they lack the organic qualities of our natural world. Ken Perlin, one of the programmers for the graphics in *Tron,* expressed frustration with the clean look. Later, in 1983, he developed a technique called Perlin Noise to generate organic textures that have a random appearance even though they are fully controllable to allow for careful design. Perlin Noise makes it possible for computer graphics models to have the subtle irregularities of real objects; it is used to create hard surfaces such as rocks and mountains and softer systems like fire and clouds. By the 1990s, it was being used extensively in Hollywood special-effects films and had been incorporated into most off-the-shelf modeling software.

Today the most widely known artists to use random values still do so without computers. For example, 2002 Turner Prize winner Keith Tyson designed sculptures not by using a computer to produce random numbers, but by rolling dice. One reason for this sort of reluctance to use computers, certainly, is the stigma surrounding computers in art. As Manfred Mohr remarked in an interview, "I called my work generative art, or occasionally also algorithmic works. The problem was that no-one understood either of these terms, and I was forced—so to speak—to declare my drawings as art from the computer . . . people accused me of degrading art, because I was employing capitalistic instruments of war—computer was a word non grata!" (Mohr 2007, 35). While Mohr was referring to the situation in the 1970s, the aversion to computers in art remains strong today.

More recently, however, as a new generation of visual artists have started to program their work, computed random numbers are playing an increasing role in the visual landscape. The most prominent programming languages used by visual artists have functions for generating random numbers and noise values, as well as for setting the random seed value to allow

RANDOMNESS IN CONTEMPORARY COMPUTING

In the many examples of randomness given here, the random element of the process—whether computational, literary, or aesthetic—is often foregrounded, or at least made very obvious. Randomness is not always visible, however, even though it is often used in ordinary computing tasks. Randomness plays an essential role in the security of networked computers, for instance, and is also a part of popular computer games. Other uses of randomness lie beyond the everyday computing experience, but security, networking, and gaming are a few of the ones that are closest at hand.

When a computer needs to generate a new password for a user, a URL that will let someone reset a password, or a CAPTCHA to keep automated spammers at bay, randomness is invoked. A nonrandom password could easily be predicted, but a random password, URL, or distorted word is much harder to crack through guessing or brute force. Randomness also plays a behind-the-scenes role in protocols such as SSH (Secure Shell) and SSL (Secure Sockets Layer) in a few ways, including the generation of keys for encryption and padding out the rest of a block when a plain-text message is too short to complete it. Without randomness, it would not be possible to complete a secure credit card transaction on the Web, which happens over SSL. Early versions of SSL as implemented in the Netscape browser suffered from being insufficiently random: The seeds for random number generation were the current time, the process ID, and the parent process ID, which were sufficiently predictable to leave the browser vulnerable to attack. Better randomness was the solution to this problem.

Computers using Ethernet—almost all of those that are plugged into wired networks—communicate with one another thanks to randomness, too. All systems on a single local area network send information over the same wire. If two of them start sending on this single wire at the same time, what is known as a "collision" occurs; the data sent is not intelligible to the intended recipients. When a collision happens, the computer that detects the problem sends a jamming signal and tries to restart the transmission. But rather than restarting immediately, the computer chooses at random to start or wait—and the other computer that was trying to send does the same. If there is another collision, the computers either send immediately or wait for one of three intervals. The increasing number of intervals is part of the technique of exponential backoff; the selection of one of these intervals at random

is an essential part of this method of avoiding network congestion.

A typical computer user of the 2010s will encounter randomness in many computer games. Randomness will shuffle the cards in poker or solitaire, for example, and will be invoked to arrange jewels and tiles in casual games. Randomness may also be used to determine the behavior of computer opponents, whether in poker, chess, or a first-person shooter. Some action, arcade-style, open-world, and other types of games incorporate randomness in other ways to determine what happens. Many early games and certain contemporary ones, however, are entirely deterministic. As those who discovered and exploited *Pac-Man* patterns know, that game is deterministic; *Ms. Pac-Man,* in contrast, uses randomness.

Though modern computers have many ways to provide initial values to seed their pseudorandom number generators, when higher levels of randomness are required one of the most reliable methods is to look beyond the computer. External entropy collection means that the random seed cannot be determined by knowing information about the computer's hardware, a common source for seeds inside the computer. In some cases the computer has to turn to a human to become more random, recording data from users mashing the keys on their keyboard or wiggling their mouse around to generate a random key or password. Even more unguessable are inputs from physical systems of sufficient complexity—anything from video of a lava lamp to atmospheric radio distortions can be used to create random numbers for computation. These levels of randomness are now required for demanding applications like high-level cryptography and scientific simulations. With continual increases in processing power, attacks on encryption are becoming easier, and the goal of making random numbers *more* random will be critical for securing society's constant digital transactions.

for the repetition of sequences. With the perspective of time, it seems that aesthetic computational work and random values are intertwined. Writing in 1970, Noll highlights randomness as an essential feature of the computer in relation to the arts:

> The computer is a unique device for the arts since it can function solely as an obedient tool with vast capabilities for controlling complicated and involved processes, but then again, full exploitation of its unique talents for controlled randomness and detailed algorithms could result in an entirely new medium—a creative artistic medium. (Noll 1970, 10)

THE COMMODORE 64 RND FUNCTION

The way that `10 PRINT` invokes the randomness provided by the Commodore 64 is of interest for reasons that will each be explored in turn. First, using randomness is aesthetically necessary in this program; there is no other way to achieve a similar effect. Second, the methods used in Commodore 64 BASIC are historically quite typical of computational approaches to pseudorandomness since the 1950s. Finally, out of several common approaches to randomness available on the Commodore 64, `10 PRINT` uses a very standard method that is well suited to experimentation, debugging, and the production of canonical results, although this method is not without its deficiencies.

`10 PRINT` produces a wrapping series of diagonal lines that alternate between left and right unpredictably. This unpredictability is crucial to producing the impression of a maze. Looking at variations of `10 PRINT` that have regular or no alternation demonstrates the significance of randomness in the program. It's possible to write an even simpler program than `10 PRINT` to draw only the left diagonal to the screen in a regular pattern (figure 40.4):

```
10 PRINT CHR$(205); : GOTO 10
```

This program can be extended by writing the other diagonal character to the right to form a chevron that repeats (figure 40.5):

```
{142}  10 PRINT CHR$(205.5+RND(1)); : GOTO 10
```

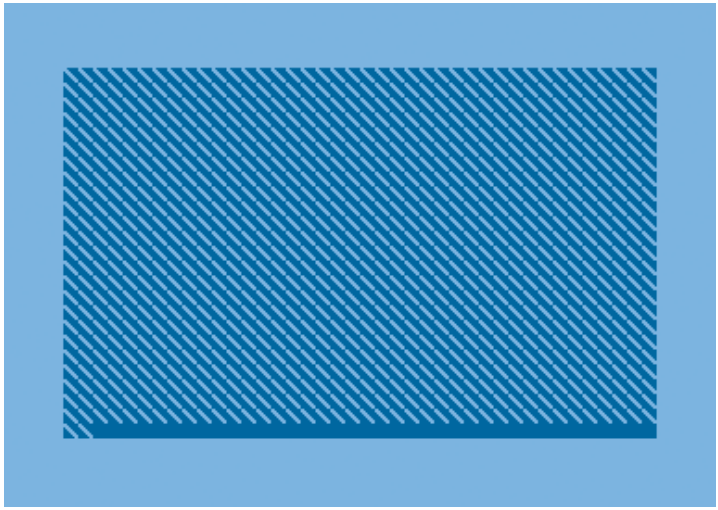

Figure 40.4
Screen capture from `10 PRINT CHR$(205); : GOTO 10`,
a regular repetition of the ╲ character.

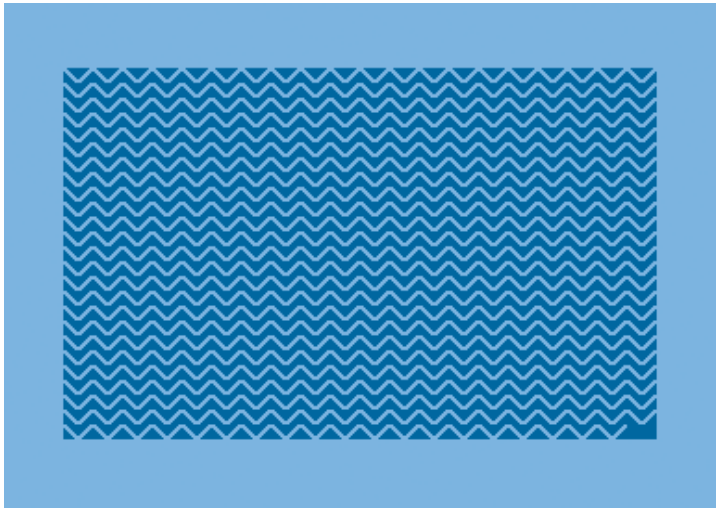

Figure 40.5
Screen capture from `10 PRINT CHR$(205)CHR$(206); : GOTO 10`,
a regular repetition of the ╲ character followed by ╱.

```
10 PRINT CHR$(205)CHR$(206); : GOTO 10
```

The next step in this elaboration is the canonical `10 PRINT`, which draws either the left or right diagonal to the screen based on the result of the random number (figure 40.6):

```
10 PRINT CHR$(205.5+RND(1)); : GOTO 10
```

In `10 PRINT`, random numbers are provided through `RND`, one of ten mathematical functions available in BASIC since the earliest version of the language. As described the original Dartmouth BASIC manual (1964), `RND` produces a "new and different random number" between 0 and 1 "each time it is used in a program" (39). These numbers can then be used to drive unpredictable processes, as in fact they do drive the coin-toss decision between diagonal lines in `10 PRINT` output. A similar process might also determine the direction changes of ghosts in *Ms. Pac-Man* or the way other game elements appear or behave.

`RND` is, like most computational sources of randomness, a pseudorandom number generator. While there may be no apparent pattern between any two numbers, each number is generated based on the previous one using a deterministic process. When the first number is the same, the entire sequence will always be the same. In the case of the Commodore 64, this is particularly important because the same seed, and thus the same first number, is set at startup. So when `RND(1)` is invoked immediately after startup, or before any other invocation of `RND`, it will always produce the same result: 0.185564016. The next invocation will also be the same, no matter what Commodore 64 is used or how long the system has been on. The next invocation—and all others—will also be the same. Since the sequence is deterministic, the pattern produced by the `10 PRINT` program typed in and run as the first program is always the same, on every computer or well-functioning emulator.

When called on *any* positive number, as when `RND(1)` is invoked in `10 PRINT`, `RND` produces the next number in this sequence. `RND(8)`, `RND(128)`, and `RND(.333)` do exactly the same as `RND(1)`. `RND`, however, has two other modes besides the one used in `10 PRINT`. The second is stopwatch-based: when `RND(0)` is called, the clock time since the computer was powered on is used in generating a new seed, meaning

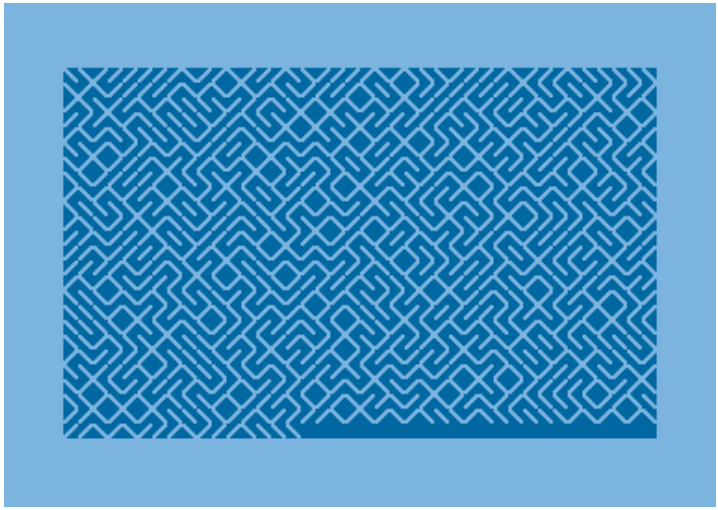

Figure 40.6
Screen capture from `10 PRINT CHR$(205.5+RND(1)); : GOTO 10`, which has a 50/50 chance of writing a ╲ or ╱ at each loop.

that if `RND(0)` replaces `RND(1)`, each run of `10 PRINT` at a different second should generate a different output. After a single call to `RND(0)`, subsequent calls to `RND(1)` will continue generating numbers in that new sequence.

The third mode for RND applies when any negative number is called. A call to `RND(-17)` stores −17 as the seed value for the random number generator, directly, and produces a new number. This negative seeding must be followed by positive calls to the function, such as `RND(1)`, in order to provide a useful sequence. Because negative calls simply set the seed, calling `RND(-1)` repeatedly will always return 0.544630526. For this reason, `10 PRINT` could not be a single-line loop that calls a negative `RND` value; that program would output the same diagonal again and again. A single call to `RND`, however, with any negative number, followed by the rest of the `10 PRINT` program, will generate a unique (and repeatable) `10 PRINT` pattern.

Pseudorandomness, however lacking it may sound, is generally acceptable and in many situations desirable. Engineers running a computer simulation, for example, often have many random variables, but every run

of the simulation needs those variables to have the same values; otherwise the program cannot be tested or the experiment repeated. Pseudorandom number generators are also highly useful in hashing, since they allow data to be distributed widely but also placed in known locations. Similarly, they are useful in cryptography, where it is vital that sequences be repeatable if (and only if) the initial conditions are known.

The *Commodore 64 User's Guide* introduces the concept of randomness using an example that sidesteps the origins of randomness in computing. There is no mention of the hydrogen bomb, computer-generated literature, or prime numbers. Randomness comes into play in the shape of a game when it is necessary to, as the manual puts it, "simulate the throw of dice" (Commodore 1982, 48). This example takes the reader back to preindustrial notions of randomness. Yet, centuries ago, long before Mallarmé provided his assurance that a throw of the dice would not abolish chance, Sir Walter Raleigh wrote of this event as apocalyptic:

> Dead bones shall then be tumbled up and down,
> In every city and in every town.

Fortune's wheel and what Paul Auster called *The Music of Chance* have long been considered a matter of life and death. As `10 PRINT` scrolls its playful, pleasing maze pattern upon the screen, there may be the faintest echo of the dead bones of the dice and the random simulation of the hydrogen bomb. And perhaps, as well, there is the transformation of this grim, military use of randomness into a thing of beauty.