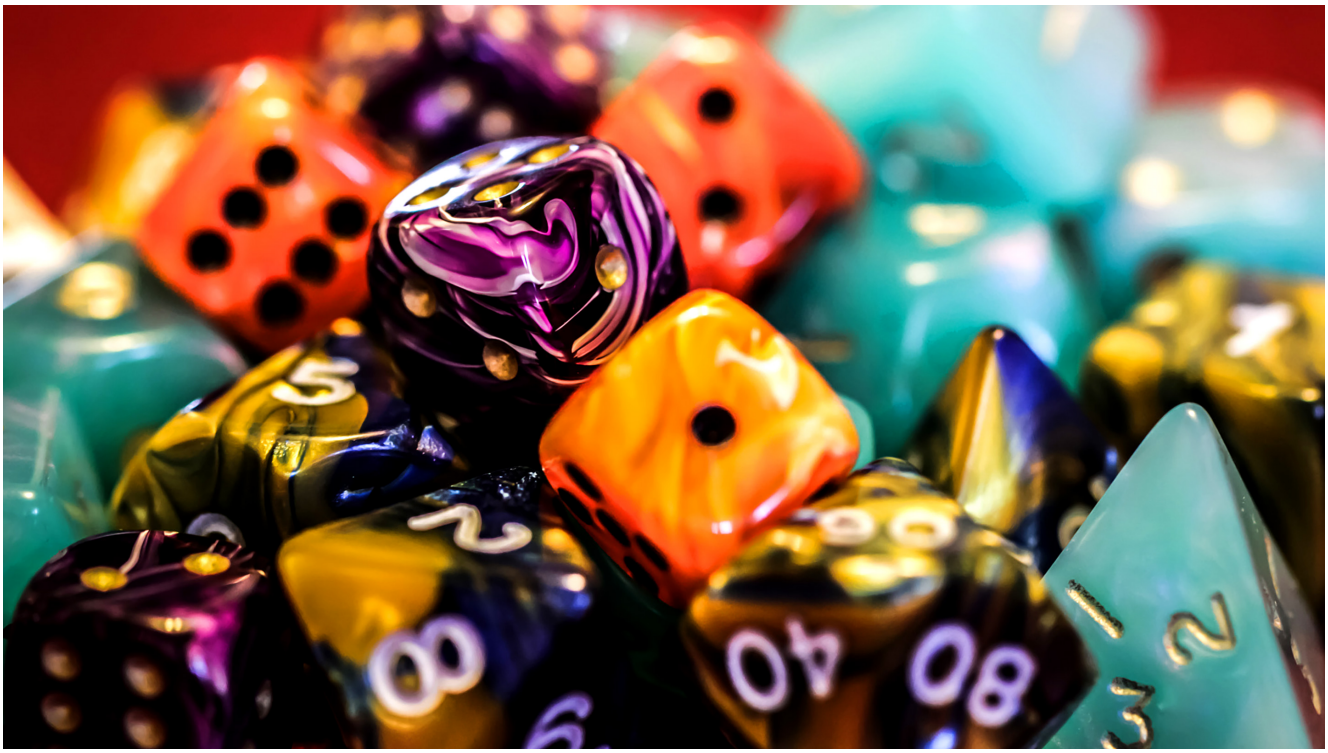
Quanta magazine

How and Why Computers Roll Loaded Dice

By Stephen Ornes

July 8, 2020

Researchers are one step closer to injecting probability into deterministic machines.



Simulating the roll of loaded dice has long been a problem for computers.

Justin Mathews / Wikimedia Commons

Here's a deceptively simple exercise: Come up with a random phone number. Seven digits in a sequence, chosen so that every digit is equally likely, and so that your choice of one digit doesn't affect the next. Odds are, you can't. (But don't take my word for it: Studies dating back to the 1950s reveal how mathematically nonrandom we are, even if we don't recognize it.)

Don't take it to heart. Computers don't generate randomness well, either. They're not supposed to: Computer software and hardware run on Boolean logic, not probability. "The culture of computing is centered on determinism," said Vikash Mansinghka, who runs the Probabilistic Computing Project at the Massachusetts Institute of Technology, "and that shows up at pretty much every level."

But computer scientists want programs that can handle randomness because sometimes that's what a problem requires. Over the years, some have developed sleek new algorithms that, while they don't themselves generate random numbers, offer clever and efficient ways to use and manipulate randomness. One of the most recent efforts comes from Mansinghka's group at MIT, which

will present an algorithm called Fast Loaded Dice Roller, or FLDR, at the online International Conference on Artificial Intelligence and Statistics this August.

Simply put, FLDR uses a random sequence to perfectly simulate the rolling of a loaded die. (Or a weighted coin, or a rigged card deck.) "It shows how to change a perfectly random coin toss into a biased coin toss," said the mathematician Peter Bierhorst of the University of New Orleans. Bierhorst did not contribute to the research but describes the premise behind FLDR's success as "compelling."

FLDR won't give you an edge in Monopoly or boost your odds at Las Vegas craps tables. But its creators say it suggests a way to integrate random numbers into software and hardware, which are built to be deterministic. Incorporating this kind of uncertainty will help machines make humanlike predictions and better simulate phenomena that rely on probability like the climate or financial markets.

Randomness is a trickier concept than it seems. In sequences of random numbers, where there's no discernible pattern, every digit has the same probability of showing up. Pi itself is not a random number, for example, but the digits of pi are believed to be random in this way (what mathematicians describe as "normal"): Every integer from 0 to 9 appears about the same number of times.

Even though you can find a "random number generator" in Google, pure randomness is not what you get. Today's processors, search engines and password generators use "pseudorandom" approaches that are close enough for most purposes. They're generated according to complicated formulas, but in theory if you knew the formula, you could likely predict the sequence.

Scientists have tried to build actual randomness into machines for at least 80 years. (Before then, random numbers came from old standbys like rolled dice, numbered balls picked from a well-mixed bag or other mechanical exercises. In 1927, a statistician sampled census data to produce a table of 40,000 random digits.)

Vikash Mansinghka of the Massachusetts Institute of Technology says the new Fast Loaded Dice Roller could help scientists incorporate probability into deterministic computers.

Allana Taranto / Ars Magna Photography

Early sources of random numbers were physical machines. The first random number-generating machine was the brainchild of the British statisticians Maurice George Kendall and Bernard Babington Smith, who described it in 1938. The machine was placed in a dark room, where it spun a disk divided into 10 equal wedge-shaped pieces labeled 0 to 9. When someone tapped a key at erratic intervals, a brief neon flash or spark would illuminate the disk so that it appeared to freeze, with only one number visible.

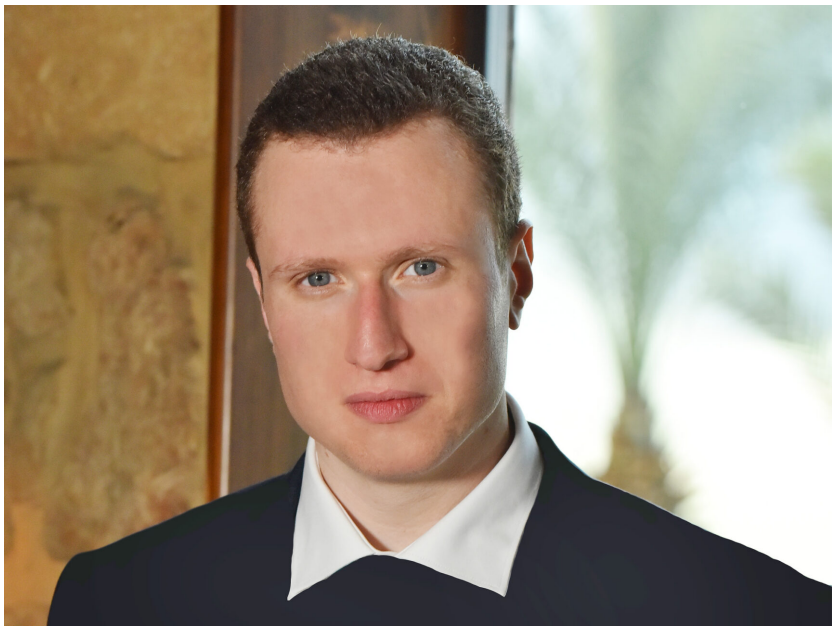
A later machine, called Ernie, used signal noise in neon tubes instead; starting in 1957 it chose the winning numbers in a British lottery.

More recently, for truly random sequences, Bierhorst says, computer scientists have to look to natural phenomena like the cosmic background radiation or the unpredictable behaviors of quantum systems. “There are random processes in nature that can be harnessed that are truly unpredictable,” he said. “The electron that dodges to the left or to the right doesn’t even know what it’s going to do beforehand.”

But randomness can only take you so far. By the late 1940s, physicists began generating random numbers to fit a given probability distribution. Such tools — a theoretical version of rolling a loaded die — could more accurately be used to model real-world situations, like traffic flow or chemical reactions. In 1976, the pioneering computer scientists Donald Knuth and Andrew Yao introduced an algorithm that could use a string of random numbers to produce random samplings of any array of weighted outcomes. “It was the most time-efficient algorithm one could ever arrive at,” said Feras Saad, a doctoral student in Mansinghka’s lab who led the new work on FLDR.

Unfortunately, Saad says the algorithm makes a trade-off that’s well known to computer scientists: Many applications that run quickly use a lot of memory, and applications that use little memory can have a long runtime. Knuth and Yao’s algorithm falls into the first category, making it elegant but too memory-intensive for any practical use.

But last spring, Saad spotted a clever workaround. He realized that when the weights on a die added up to a power of 2, the algorithm was efficient in both time and memory. So for a six-sided die, imagine that the odds of rolling the numbers 1 through 6, respectively, are weighted at 1, 2, 3, 4, 5 and 1. That means the probability of rolling a 1, for example, is 1/16, and the probability of rolling a 2 is 2/16. Since the weights add up to a power of 2 — 16 is 2^4 — the algorithm for this system is efficient in time and memory.



Feras Saad, a doctoral student at the Massachusetts Institute of Technology, helped devise a new algorithm that efficiently and accurately simulates the rolling of a loaded die.

Walid Antar

But imagine that the weights are instead 1, 2, 3, 2, 4, 2, which sum to 14. Since 14 is not a power of 2, the Knuth-Yao algorithm requires significantly more memory. Saad realized he could include an additional weight so that they always summed to a power of 2. In this case, he would simply add another hypothetical face — one with weight 2 — so that the weights added up to 16. If the

algorithm chose one of the original faces, the value was recorded. If it chose the extra face, the value was discarded, and the algorithm was run again.

This is the key to FLDR's efficiency, making it a powerful tool in simulations: The amount of extra memory needed for the additional rolls is minimal compared to the hefty memory usually required in the original.

FLDR makes the Knuth-Yao algorithm efficient and suggests ways to improve a wide swath of applications. Climate change simulations, crop yield predictions, particle behavior simulations, financial market models and even the detecting of underground detonations of nuclear weapons depend on random numbers in weighted probability distributions. Random numbers also form the backbone of cryptography schemes that protect data sent over the internet.

Mansinghka says FLDR may also help researchers find ways to integrate probability into computer processors — the focus of his lab's work at MIT. The algorithm could help inform the design of software libraries and new hardware architectures that can generate random numbers and use them in efficient ways. That would be a dramatic departure from the deterministic Boolean machines we're used to.

"We might be very well positioned to integrate randomness into the building blocks of software and hardware," he said.