Calculating probabilities and results distributions is an important element of board game design. McDie is a node-based Monte Carlo simulator to help designers to rapidly iterate through different game mechanisms to tune them for their purposes.

While designing a board game may appear simple on the surface, it involves integrating a number of disciplines across the arts and sciences, including graphic design, psychology, and math. Not surprisingly, a big part of the required math is probability.

Most games include some random element, whether it is drawing cards, rolling dice, or pulling colored cubes from a bag. And as a designer, developing mechanisms to give the probabilities you want is an important skill.

Many designers don’t explicitly calculate probabilities while designing. They simply test mechanisms repeatedly, get a feel for what the range of outcomes are, and tweak accordingly. Being able to do the math and know exactly what the probabilities are will get you to the same place as testing, and it will get you there faster. However, once you get past basic probability problems, doing the calculations quickly gets past the capability of most designers.

Recently I was testing out a new feature on the powerful Unity framework, a tool for developing video games. A key element for video game graphics is “shaders,” which are responsible for actually rendering the pixels onto the screen and calculating glow, transparency, reflections, and advanced effects. Shaders can do amazing things if used properly. For years, shaders needed to be written in a special computer language. But recently, visual tools have been introduced to allow even novice programmers to design complex shader effects.

They work by placing what are called nodes onto a canvas and connecting them together. Nodes have inputs, where data come in, then they do some processing, and data come out of the node outputs. Those modified data are then piped down the line, until they end at nodes that render the pixels to the screen. Shader nodes can do all kinds of things, such as generating noise, multiplying textures together, and physically deforming geometry. The key is that the gritty details of how each node works internally don’t matter to the user. They just wire up a network to do what they want and let it run.

It occurred to me that a similar visual node-based model could be used to let non-programmers or the math-phobic to create probability models. Certain nodes could generate dice, others could manipulate pools of dice, divide up pools, detect certain patterns, and so on. The model could then be run thousands of times, and the output collected in a histogram. No knowledge of math or coding required.

And so the Monte Carlo Dice Information Explorer—McDie—was born.

A sample network is shown in the first image. This runs two different models at the same time. The first rolls three six-sided dice (3d6), removes the highest, and adds the two remaining dice. The second model rolls two dice (2d6) and adds them. The simulation is run 100,000 times, and the results are plotted to the right. Most people intuitively understand that rolling three dice and removing the highest will lower the average roll, but by how much? McDie makes it easy to see that, while the average drops to 6, the most common result is a 5. In addition, the overall curve is much flatter near the mode than it is in the normal 2d6 distribution.
McDie also allows designers to experiment with more complex models. The second image simulates two players attacking each other by rolling a pool of six-sided dice. A 1, 2, or 3 counts as a block; 4 and 5 are one hit; and a 6 is two hits. In addition, any rolls of 6 also get rerolled and may generate additional hits or blocks or more rerolls. Blocks are then subtracted from hits to determine the net damage. Normally designers would test a system like this by hand, rolling dice and getting a feel for how it is working. Having this model allows for fine-tuning of the parameters to get the desired results.

Another useful part of McDie is exploring very-low-probability events. In certain games, there may be very unlikely combinations that reduce enjoyment, cut games short, or even lock them up. Due to their low frequency, these events may not arise in playtesting, or the designer may know about them but think they are much less likely than they actually are. Exploring these black swan edge cases can be very

Figure 1. Modeling rolling two six-sided dice versus rolling three dice and discarding the highest

Figure 2. A complete model of a dice-based battle system
useful and avoid problems once the games are being played by tens of thousands of people, and even low-probability events have a good chance of happening.

It is also very instructive for designers to learn about how the actual distribution of random results in a game can vary from the expectation. Normally, McDie runs a model 100,000 times. However, it can be set to run any number of iterations. If the dice are rolled 10 or 20 times during a game, it will rarely look like the theoretical distribution. McDie makes it very easy to run 10 times, see the results, and repeat that a bunch of times to compare what actual games may look like. Most designers are surprised at how much a distribution can vary, even when you roll dice a hundred times.

As a designer I’ve also found McDie to be a source of design inspiration. There are now many, many different types of nodes included. And as the number of nodes grows, the number of ways to combine them grows exponentially. I’ve already had many ideas about unique and innovative ways to manipulate pools of dice just from designing new nodes and thinking about how they interact with each other. Having a tool to rapidly explore these ideas is, frankly, fun. I’m currently a tad obsessed with dice combinations that yield bimodal distributions and how those can be used in a game.

For the future I plan to add support for card decks, bag draws, and similar systems where you pull from a fixed pool. This will open up a whole other world of simulation for designers. The goal is also to turn it into an open source project, so others can contribute their ideas as well.

McDie is currently available as a “pay what you want” download (which includes free) at https://gengelstein.itch.io/mcdie. All sales proceeds are donated to New Voices In Gaming, which provides scholarships for designers from underrepresented groups to game design conferences and events. The author is the founder and chair of this organization.

About the Authors
Geoff Engelstein is a board game designer whose latest titles include Super-Skill Pinball and Versailles 1919. He teaches game design at the NYU Game Center and is the author of several books including Building Blocks of Tabletop Game Design and Game Production. He is also the president of Mars International, a product development and manufacturing firm.