Three Rules, Infinite Complexity:

What Conway's Game of Life Teaches Us About the Universe

Mustafa Güzel

April 2026

A Game of Life and Death

What if three simple rules were enough to build a computer?

That sounds impossible at first. The Game of Life is just a grid of black and white squares. Each square is either alive or dead. There is no randomness. No hidden trick. And yet, from this tiny system, you can get movement, memory, and logic.

This is the Game of Life.

In 1970, the British mathematician John Horton Conway looked for the simplest set of rules that could still create surprising behaviour. He was not trying to make something useful. He wanted to answer a harder and more interesting question: how much complexity can come from something simple?

His answer was surprising. With only three rules, a grid of cells can create patterns that stay still, patterns that repeat, and patterns that seem to move across the board. Later, mathematicians found something even stranger: this system can even perform computation.

Conway died in April 2020, but the world he created is still alive. People still study the Game of Life because it shows something beautiful and strange. Very simple rules can create a world that is hard to predict and impossible to forget.

In this essay, I want to show why.

The Rules

The Game of Life takes place on an infinite grid of square cells. Each cell can be in one of two states: *alive* or *dead*. Every cell has eight neighbours: the cells next to it horizontally, vertically, and diagonally. Time moves one step at a time, and at each step, all cells update at the same time according to three rules:

1. **Birth.** A dead cell with exactly 3 alive neighbours becomes alive.
2. **Survival.** An alive cell with 2 or 3 alive neighbours stays alive.
3. **Death.** In all other cases, the cell dies or stays dead. With fewer than 2 alive neighbours, it dies from isolation. With more than 3, it dies from overcrowding.

That is the whole game. There are no players, no choices, and no luck. You choose the starting pattern by deciding which cells begin alive. After that, the rules do everything. Everything that follows is determined by that initial state.

This matters because the Game of Life has no randomness at all. It is not like tossing a coin or shuffling cards. If you start with the same pattern twice, you will get the same result twice. Always. This will matter later.

You might also ask how many starting patterns are possible. On an $n \times n$ grid, there are n^2 cells, and each one can be either alive or dead. That gives

$$2^{n^2}$$

possible configurations. Even for a modest 10×10 grid, this becomes $2^{100} \approx 1.27 \times 10^{30}$, which is larger than the estimated number of grains of sand on Earth.

This number is worth looking at more closely. The number of possible starting patterns grows extremely fast, and that is one reason the Game of Life can produce so much variety. On a 3×3 grid, there are only $2^9 = 512$ possible states, which is still small enough to check by hand. On a 4×4 grid, the number jumps to $2^{16} = 65{,}536$. On a 5×5 grid, it becomes $2^{25} = 33{,}554{,}432$. By the time we reach a 10×10 grid, there are about 1.27×10^{30} possible starting configurations.

$$3 \times 3 \to 512, \quad 4 \times 4 \to 65{,}536, \quad 5 \times 5 \to 33{,}554{,}432, \quad 10 \times 10 \to 1.27 \times 10^{30}.$$

So even very small grids already allow an enormous number of possibilities. This helps explain why the Game of Life feels so rich. The rules themselves never change, but the starting pattern can change in countless ways. That is what gives the system so much room to produce different kinds of behaviour.

Why the blinker has period 2

One of the simplest interesting patterns is the *blinker*, a horizontal row of three alive cells. It is worth checking carefully what happens, because this is the first place where the rules create a repeating pattern.

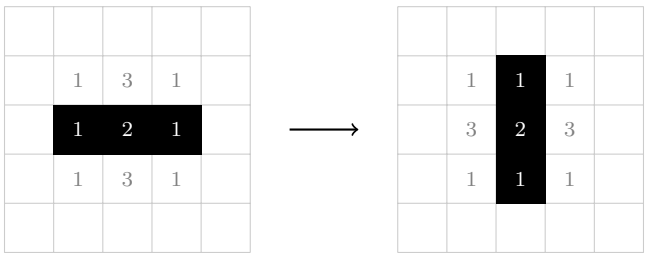


Figure 1: Neighbour counts for the blinker. Numbers show how many alive neighbours each nearby cell has.

In Step 0, the middle alive cell has 2 alive neighbours, so it survives. The two alive cells

at the ends each have only 1 alive neighbour, so they die. At the same time, the dead cell directly above the middle and the dead cell directly below it each have exactly 3 alive neighbours, so they become alive.

So after one step, the horizontal row turns into a vertical one.

Now the same reasoning applies again. In the vertical configuration, the middle cell survives, the two end cells die, and two new cells are born on the left and right. After one more step, the pattern returns to its original horizontal form.

So the blinker repeats every 2 steps. This makes it the simplest oscillator in the Game of Life.

Emergence

At first, it is easy to think that three simple rules will only create simple results. But the Game of Life quickly proves otherwise. Some patterns stay still. Some repeat. Some seem to move. And some can keep producing new structure forever.

Still Lifes

The simplest patterns are the ones that never change.

Claim. *The block, a* 2×2 *square of alive cells, is a still life: after one step, it remains unchanged.*

Proof. Consider the block on a 4×4 grid. Each of its four alive cells has exactly 3 alive neighbours:

1	2	2	1
2	**3**	**3**	2
2	**3**	**3**	2
1	2	2	1

Figure 2: The block with neighbour counts. Each alive cell has 3 neighbours, and no nearby dead cell has exactly 3.

Since each alive cell has 3 neighbours, all four survive by the survival rule. Also, every dead cell around the block has at most 2 alive neighbours, so no new cell is born. That means the pattern after one step is exactly the same as the pattern before.

So the block is a still life.

Oscillators

Some patterns do change, but only in a cycle. We proved in the previous section that the blinker has period 2. More complicated oscillators also exist. The *pulsar*, for example, has 48 cells and period 3. In fact, people have found oscillators with every period greater than

1.

Spaceships

Then comes the real surprise. The *glider* is a pattern of only 5 cells that, over 4 steps, returns to its original shape but shifted one cell diagonally. Then the process repeats, so the pattern seems to move across the grid forever.

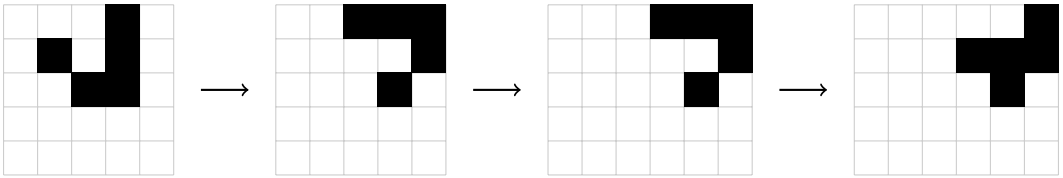


Figure 3: The glider at generations $t = 0, 1, 2, 4$. After 4 steps it returns to the same shape, shifted one cell right and one cell down.

What makes this so strange is that nothing in the rules says anything about movement. Cells do not move from one place to another. They only live, die, or appear. And yet the pattern moves. Direction and motion show up at the level of the whole pattern, even though they are not written directly into the rules. This is emergence.

Claim. *The glider travels at speed $c/4$, where $c = 1$ cell per generation is the maximum possible speed in the Game of Life.*

Proof. In the Game of Life, information can spread only through neighbour interactions. Since each cell affects only the cells next to it, no signal can travel more than 1 cell in a single generation. This gives a maximum possible speed of $c = 1$ cell per generation.

Now look at the glider. After 4 generations, it returns to the same shape, but shifted one cell to the right and one cell down. So along each axis, it moves 1 cell in 4 generations. That is why its speed is written as

$$\frac{1 \text{ cell}}{4 \text{ generations}} = \frac{c}{4}.$$

So the glider moves at one quarter of the maximum possible speed.

Glider Guns

At this point, a natural question appears: can a finite pattern keep producing new structure forever? In 1970, Conway offered a \$50 prize to anyone who could answer this question. Very soon, Bill Gosper found the answer. He discovered the *Gosper glider gun*, a configuration of 36 cells that emits a new glider every 30 generations.

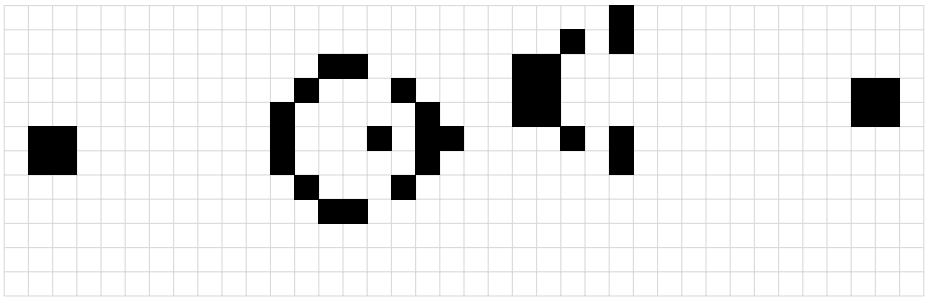


Figure 4: The Gosper glider gun: 36 cells that emit a new glider every 30 generations, producing unbounded growth.

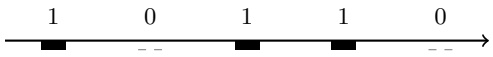
Its population keeps growing over time, because every 30 generations it sends out another 5-cell glider. This was a turning point. The Game of Life was no longer just producing small, interesting patterns. It could now create an endless stream of moving ones. And once gliders could be produced in a controlled way, they could be used as signals. That is the idea behind the next section.

A Computer Made of Life

The glider gun changes the whole picture. Once a pattern can keep producing gliders in a steady way, those gliders stop looking like a small curiosity. They start to look like signals.

Gliders as Bits

A glider moves in a straight line at a constant speed: one cell diagonally every 4 generations. A glider gun can send out a steady stream of them. Now imagine two possibilities in that stream: a glider is *present*, or a glider is *absent*. We can call presence 1 and absence 0. In other words, the Game of Life can carry binary information across the grid.



A stream of gliders encodes binary: `1 0 1 1 0`

Figure 5: Glider streams as binary signals: present = 1, absent = 0.

Collisions as Logic Gates

The next question is what happens when two glider streams meet. Depending on their timing and angle, different things can happen. The gliders might destroy each other, change direction, or create other patterns. This is not just random mess. With careful design, these collisions can be controlled.

And once they can be controlled, they can be used to perform logic.

A logic gate is one of the basic parts of a computer. Three important examples are:

- **AND** means the output is 1 only if both inputs are 1.

- **OR** means the output is 1 if at least one input is 1.
- **NOT** flips the input, so 1 becomes 0 and 0 becomes 1.

Researchers have shown that all of these gates can be built from glider collisions in the Game of Life.

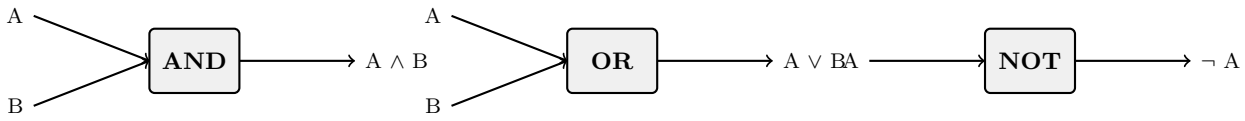


Figure 6: Logic gates implemented by glider collisions.

A Concrete Example: Binary Addition

To see this more clearly, let us look at a very simple computation: adding two single binary digits. In binary, $1 + 1 = 10$, which is 2 in decimal. This means the result has two parts: a *sum* bit and a *carry* bit.

The full table looks like this:

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

If you read the table carefully, you can see two familiar logic operations inside it. The sum bit is 1 exactly when the two inputs are different, so it matches XOR. The carry bit is 1 only when both inputs are 1, so it matches AND. In symbols,

$$\text{Sum} = A \oplus B, \qquad \text{Carry} = A \wedge B.$$

We already know that AND, OR, and NOT gates can be built from glider collisions. And XOR can be built from those same gates:

$$A \oplus B = (A \vee B) \wedge \neg(A \wedge B).$$

So a binary adder can also be built inside the Game of Life. Two glider streams go in, representing the two input bits, and the resulting collisions produce two output streams: one for the sum and one for the carry.

Once you can do this for one pair of bits, you can chain many copies together and add larger binary numbers as well. That is exactly the basic idea behind addition in an ordinary computer. The difference is that here the wires are streams of gliders, and the gates are made from carefully arranged collisions.

Universal Computation

This is the point where the Game of Life stops feeling like a simple toy. In computer science, we know that any computation can be built from logic gates. Your phone, your laptop, and every large computer all rely on huge collections of simple logical operations.

So if the Game of Life can build AND, OR, and NOT gates, then it can also build arbitrary computation. In the language of computer science, it is *Turing complete*.

This idea goes back to Alan Turing. In 1936, he described an abstract machine, now called a *Turing machine*, that captures the basic idea of computation. If a system can simulate a Turing machine, then it can in principle compute anything that any other computer can compute.

The Game of Life can simulate a Turing machine. That is why it is called Turing complete.

$$\text{Game of Life can simulate a Turing machine} \Rightarrow \text{Game of Life is Turing complete.}$$

This may sound abstract, but the idea is real. In 2010, enthusiasts built a full implementation of the Game of Life inside the Game of Life itself. A system made from three simple rules was able to simulate its own world.

The Unpredictability Paradox

Let us return to an important fact from earlier: the Game of Life is completely deterministic. There are no dice, no coin flips, and no hidden randomness. Once the starting configuration is fixed, every future state is fixed as well.

So it is natural to ask a tempting question. If everything is determined from the start, should there not be a shortcut? Could we find a formula that jumps straight to generation t without simulating all the steps in between?

Surprisingly, the answer is no. And this is not just because we have not found the right trick yet. It is a mathematical limitation.

The Halting Problem

The reason goes back to Alan Turing's work in 1936. Since the Game of Life is Turing complete, it can encode any computer program as a starting configuration. In particular, it can encode the question: does this program eventually halt, or does it run forever?

Turing proved that no algorithm can answer this question in full generality.

Theorem (Turing, 1936). *There is no computable function $H(P)$ that, for every program P, correctly determines whether P halts.*

Proof. Suppose, for contradiction, that such a function H exists. Using H, define a new

program D by the following rule:

$$D = \begin{cases} \text{halt} & \text{if } H(D) = \text{“runs forever”}, \\ \text{run forever} & \text{if } H(D) = \text{“halts”}. \end{cases}$$

Now ask whether D halts.

- If H says that D halts, then by definition D runs forever.
- If H says that D runs forever, then by definition D halts.

Both cases lead to a contradiction. So such a function H cannot exist.

Application to the Game of Life

Since the Game of Life can simulate any program, the Halting Problem applies to it as well. We can encode a program P as an initial configuration C_P such that C_P reaches a stable state if and only if P halts. So if we could always predict the long-term behaviour of C_P without carrying out the simulation, then we could solve the Halting Problem. But the proof above shows that this is impossible.

More formally, let $f^t(C)$ denote the state of a configuration C after t steps. Then there is no general shortcut that computes $f^t(C)$ for every C and every t in substantially fewer than t steps:

$$\nexists\, g : \; g(C,t) = f^t(C) \;\; \forall\, C, t \quad \text{computable in } o(t) \text{ steps.}$$

So for an arbitrary starting pattern, the only reliable way to know what happens after t steps is to let the Game of Life run for those t steps.

This is the paradox at the heart of the Game of Life. The system is fully deterministic, but that does not make it fully predictable. Very simple rules can still lead to behaviour that resists shortcuts. In the end, the only way to know the future is to watch it unfold, one step at a time.

Beyond the Grid

The Game of Life is only a grid of cells. It does not contain real organisms, real motion, or real thought. And yet, from a few simple rules, it creates patterns that stay stable, patterns that move, and patterns that can even carry out computation. That is what makes it so hard to forget.

This is also what makes the game feel bigger than it first appears. It suggests that complexity does not always need complicated rules. Sometimes very simple rules, repeated again and again, are enough to create behaviour that looks far richer than we would expect.

That idea does not only belong to the Game of Life. In science, many of the most complicated things we know come from simple foundations. Physics begins with a small number of basic laws, but from those laws come stars, chemistry, planets, and life. I am

not saying that the universe is literally the Game of Life. But the comparison is hard to ignore.

This is one reason the Game of Life has remained so fascinating for so long. It begins as a small mathematical game, but it quickly turns into something more. It becomes a way to think about emergence, computation, and the limits of prediction.

Conway started with a simple question: how much complexity can come from a very small set of rules? The answer was astonishing. Three rules were enough to create a world with structure, motion, logic, and uncertainty.

For me, that is the deepest lesson of the Game of Life. Simple things do not always stay simple. Sometimes, if you let them run long enough, they build a world.

References

[1] Gardner, M. (1970). Mathematical Games — The fantastic combinations of John Conway's new solitaire game "life." *Scientific American*, 223(4), 120–123.

[2] Turing, A. M. (1936). On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42), 230–265.

[3] Rendell, P. (2016). *Turing Machine Universality of the Game of Life*. Springer.