Conway's Game of Life

Melissa Gymrek

May 2010

Introduction

The Game of Life is a cellular-automaton, zero player game, developed by John Conway in 1970. The game is played on an infinite grid of square cells, and its evolution is only determined by its initial state.

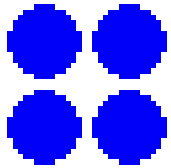
The rules of the game are simple, and describe the evolution of the grid:

- **Birth**: a cell that is dead at time t will be alive at time $t+1$ if exactly 3 of its eight neighbors were alive at time t.
- **Death**: a cell can die by:
  - **Overcrowding**: if a cell is alive at time $t+1$ and 4 or more of its neighbors are also alive at time t, the cell will be dead at time $t+1$.
  - **Exposure**: If a live cell at time t has only 1 live neighbor or no live neighbors, it will be dead at time $t+1$.
- **Survival**: a cell survives from time t to time $t+1$ if and only if 2 or 3 of its neighbors are alive at time t.

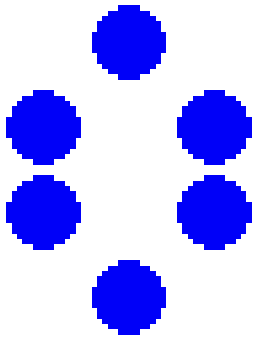
Starting from the initial configuration, these rules are applied, and the game board evolves, playing the game by itself!
This might seem like a rather boring game at first, but there are many remarkable facts about this game. Today we will see the types of "life-forms" we can create with this game, whether we can tell if a game of Life will go on infinitely, and see how a game of Life can be used to solve any computational problem a computer can solve.

Using this simple game's rules, we can create many different types of "life-forms."

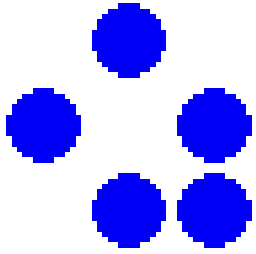
- **Still Life**: a stable, finite and nonempty pattern. Examples include various shapes of ponds, and other patterns shown below:
- Block:



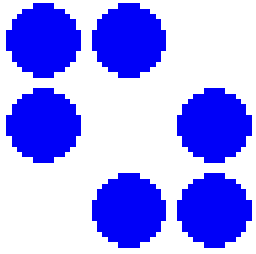
- Beehive:



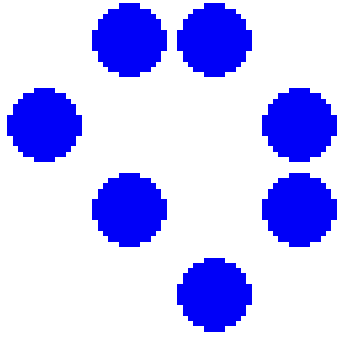
- Boat:



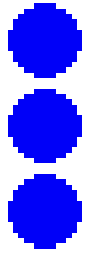
- Ship:



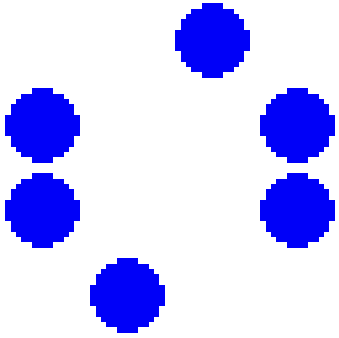
- Loaf:



- **Periodic Life Forms/Oscillators**: the following life-forms oscillate periodically
  Period 2:
  Blinker:

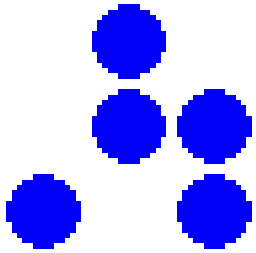


  Toad:



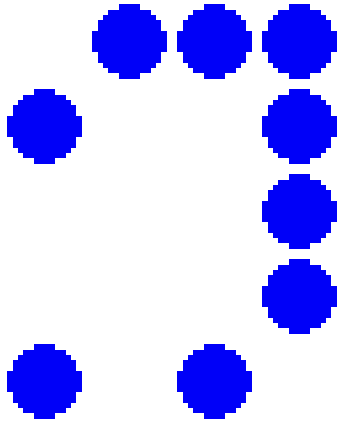
Oscillators of many more periods exist:

- **Glider** A glider is a simple 5-cell pattern that repeats itself every 4 generations, but is offset diagonally by one cell. The smallest and most common spaceship. The glider is going to be extremely important when we discuss applications of the Game of Life. The glider is said to travel with speed $\frac{c}{4}$.

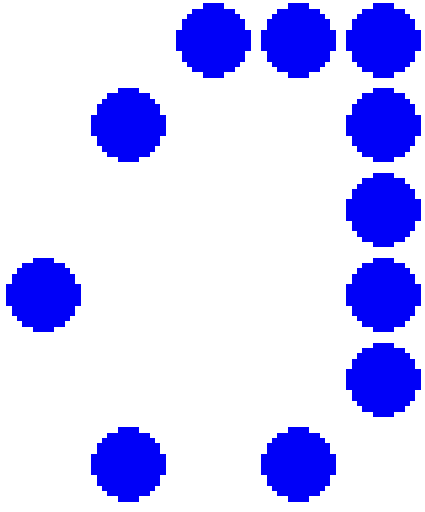


- **Spaceship** A pattern that moves across the game board. (originally Conway found the Light, Medium, and Heavy Weight Space Ship)

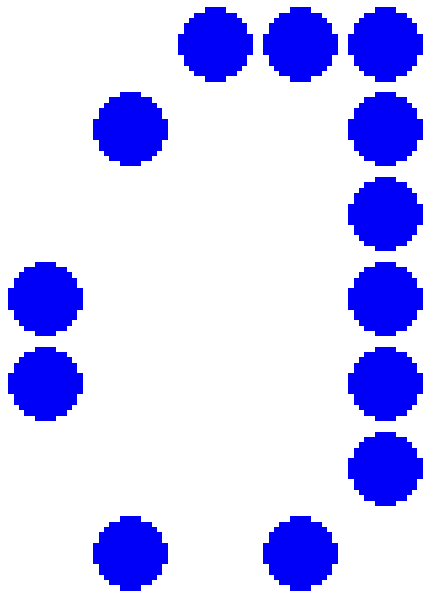
- Light Weight Space Ship (LWSS) (speed $\frac{c}{2}$):



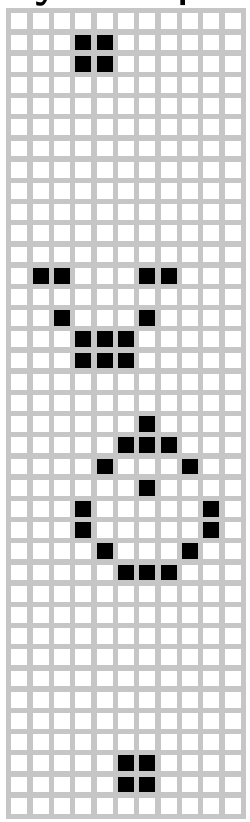
- Medium Weight Space Ship (MWSS):



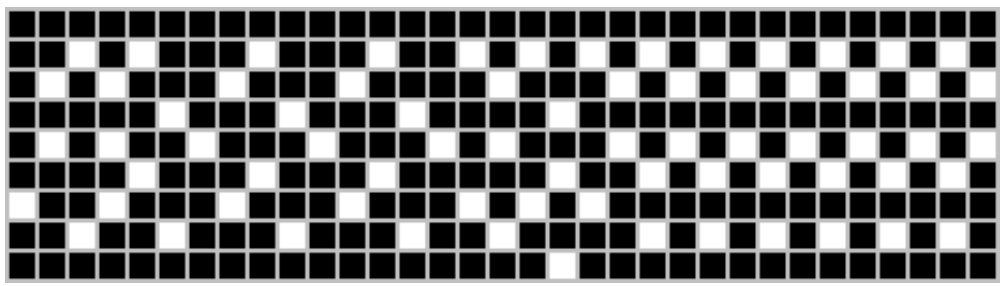
- Heavy Weight Space Ship (HWSS):



- **Guns** A gun periodically shoots out moving patterns. We will look soon at the **Glider Gun**, the first known gun developed by Gosper at MIT.



- **Garden of Eden** A pattern that can only exist as initial pattern. In other words, no parent could possibly produce the pattern.



It is not easy to tell from looking at an initial Life pattern exactly how it will evolve. For instance, in looking at what happens to a straight line of n live cells as a start configuration:

- $n = 1, 2$: fades immediately
- $n = 3$: Blinker
- $n = 4$: becomes a Beehive at time 2
- $n = 5$: traffic lights at time 6
- $n = 6$: fades at $t = 12$
- $n = 7$: makes a symmetric display before terminating in the Honey Farm (see picture in Winning Ways)
- $n = 8$: gives 4 blocks and 4 beehives
- $n = 9$: makes two sets of traffic lights
- $n = 10$: turns into pentadecathlon, with life cycle of 15
- $n = 11$: becomes two blinkers
- $n = 12$: makes two beehives
- $n = 13$: turns into two blinkers

- $n = 14, 15$: vanish completely
- $n = 16, 17$: becomes 4 blocks
- $n = 18, 19$: vanish completely
- $n = 20$: makes 2 blocks

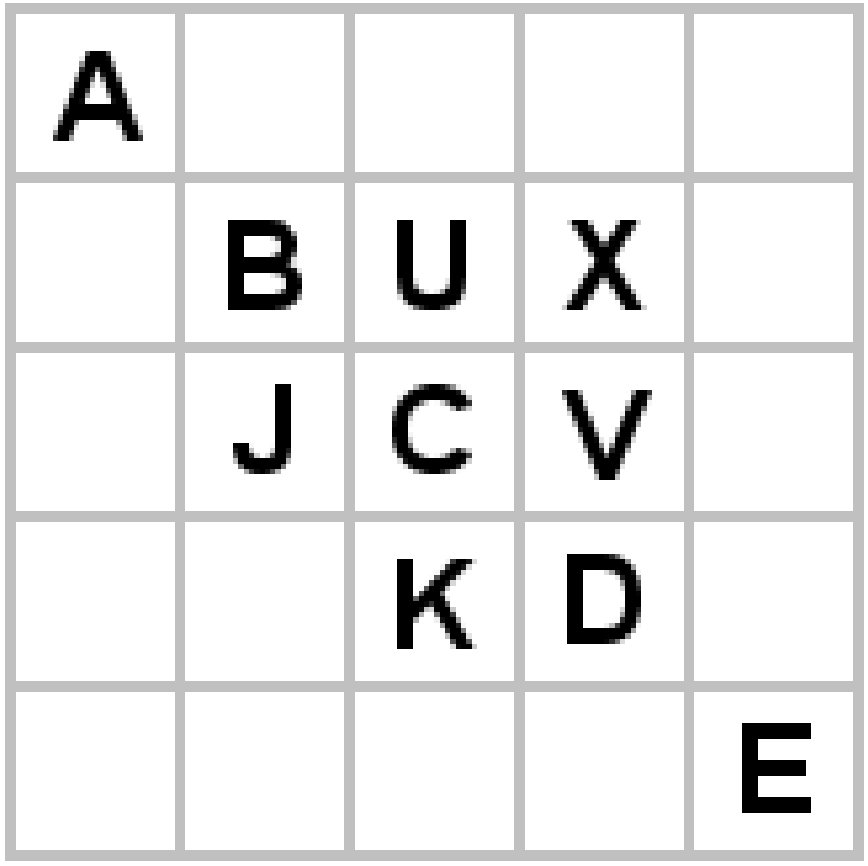
- It is not immediately obvious whether a given initial Life pattern can grow indefinitely, or whether any pattern at all can.
- Conway offered a $50.00 prize to whoever could setlle this question. In 1970 an MIT group headed by R.W. Gosper won the prize by finding the **glider gun** that emits a new glider every 30 generations. Since the gliders are not destroyed, and the gun produces a new glider every 30 generations indefinitely, the pattern grows forever, and thus proves that there can exist initial Life patterns that grow infinitely.

- In the Game of Life, we can define the "speed of light" (c, just as in physics) as the maximum attainable speed of any moving object, a propagation rate of one step (horizontally, vertically, or diagonally) per generation. This is both the maximum rate at which information can travel and the upper bound on the speed of any pattern.
- We can find the speed of two patterns we've seen, the glider and the lightweight spaceship. The glider takes 4 generations to move one cell diagonally, and so has a speed of $\frac{c}{4}$. The light weight spaceship moves one cell orthogonally every other generation, and so has a speed of $\frac{c}{2}$.
- Although the speed of light is defined as one cell per generation, we show here that the maximum attainable speed of any moving object is either $\frac{c}{4}$ diagonally or $\frac{c}{2}$ orthogonally. No spaceships can move faster than our glider or light weight spaceship.

Theorem

No spaceship can travel diagonally faster than $\frac{c}{4}$.

Consider the grid below:

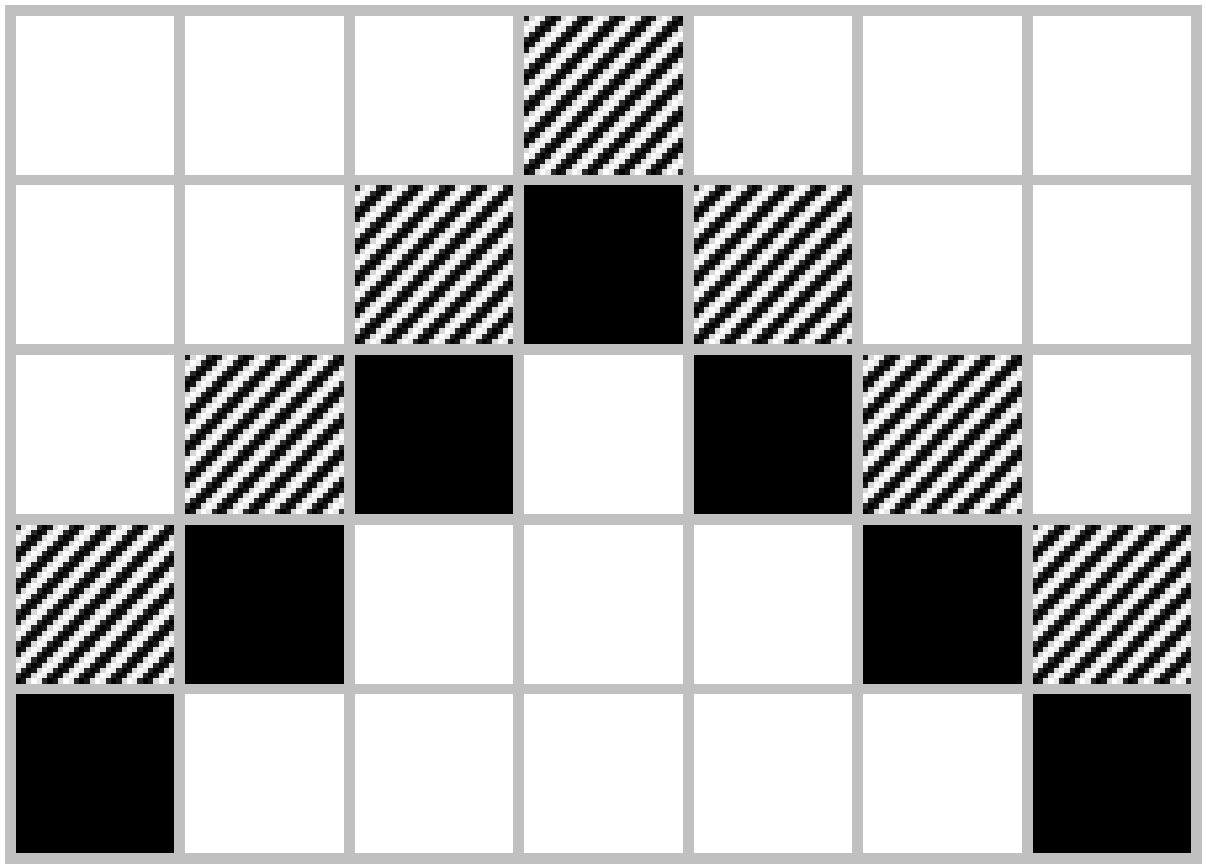


- Say we have a spaceship on and to the left of the diagonal line defined by $ABCDE$ on generation 0. If the spaceship can travel up and to the right faster than $\frac{c}{4}$, then cell X would be alive at generation 2.
- Suppose this is true, and that X is alive at generation 2. Then C, U, and V must be alive at generation 1. Then U and V must have had 3 alive neighbors in generation 0, and so B, C, D, J, and K must have been alive at time 0. Then C must have had at least four live neighbors in generation 0, and so couldn't have survived to generation 1. But we needed C alive at generation 1, and so we have reached a contradiction.
- If the spaceship is behind $ABCDE$ at generation 0, it must be behind UV at generation 2, and so can't travel faster than $\frac{c}{4}$.

Theorem

No spaceship can travel orthogonally faster than $\frac{c}{2}$.

For the orthogonal case consider the following grid:



- If a spaceship is on and below the diagonal lines defined by the solid black squares in generation 0, then we can use the argument above to claim that it must be on or below the lines defined by the striped squares at generation 2. Therefore, it can move at most 1 square forward every 2 generations, and so has maximum speed $\frac{c}{2}$.

One interesting and surprising application of the Game of Life is that we can construct an initial pattern that will generate the prime numbers sequentially. The **primer** below is due to Dick Hickerson, 1991:

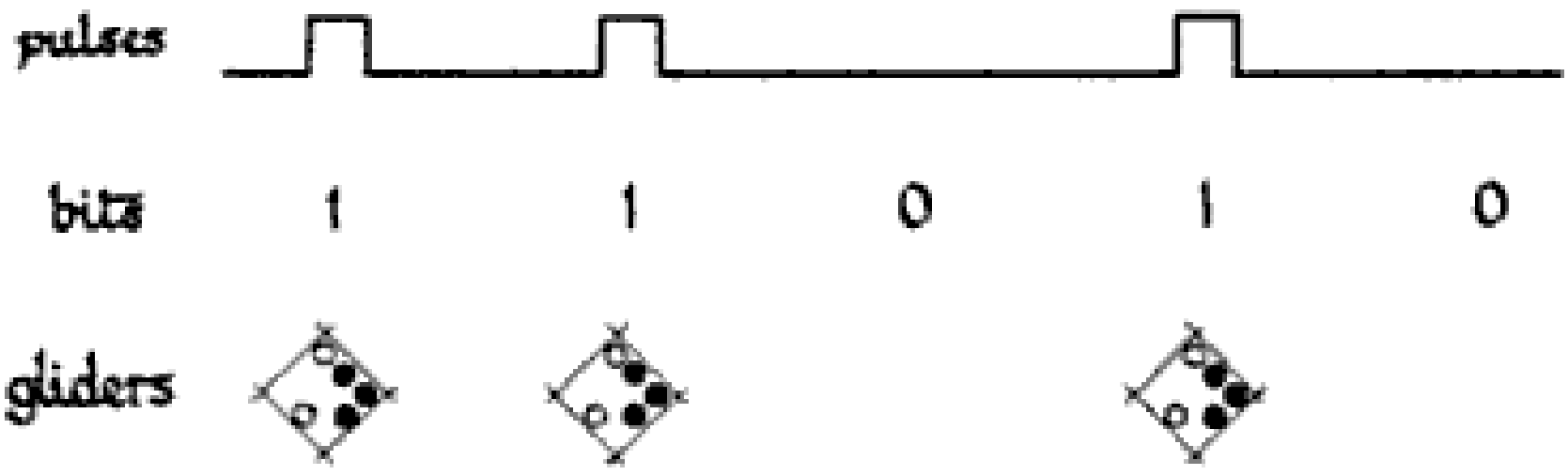
Before describing how the primer works, we describe the **Sieve of Eratosthenes**. The sieve is a simple ancient algorithm for finding all prme numbers up to a specified integer. The algorithm is as follows:

- Create a list of consecutive integers from 1 to n.
- Set $p = 2$ (the first prime)
- Cross out all multiples of p greater than p.
- Find the first number still left that is greater than p.
- Repeat steps 3 and 4 until p^2 is greater than n.
- All he remaining numbers on the list are prime.

- The primer fires a light weight space ship westward, and destroys the spaceships with gliders from a gun that simulates the Sieve of Eratosthenes.
- The light weight spaceship makes it past the left edge of the gun at generation $120N$ if and only if N is a prime number.
- There are extensions of the primer that can be used to generate twin primes, Fermat primes, and several others.

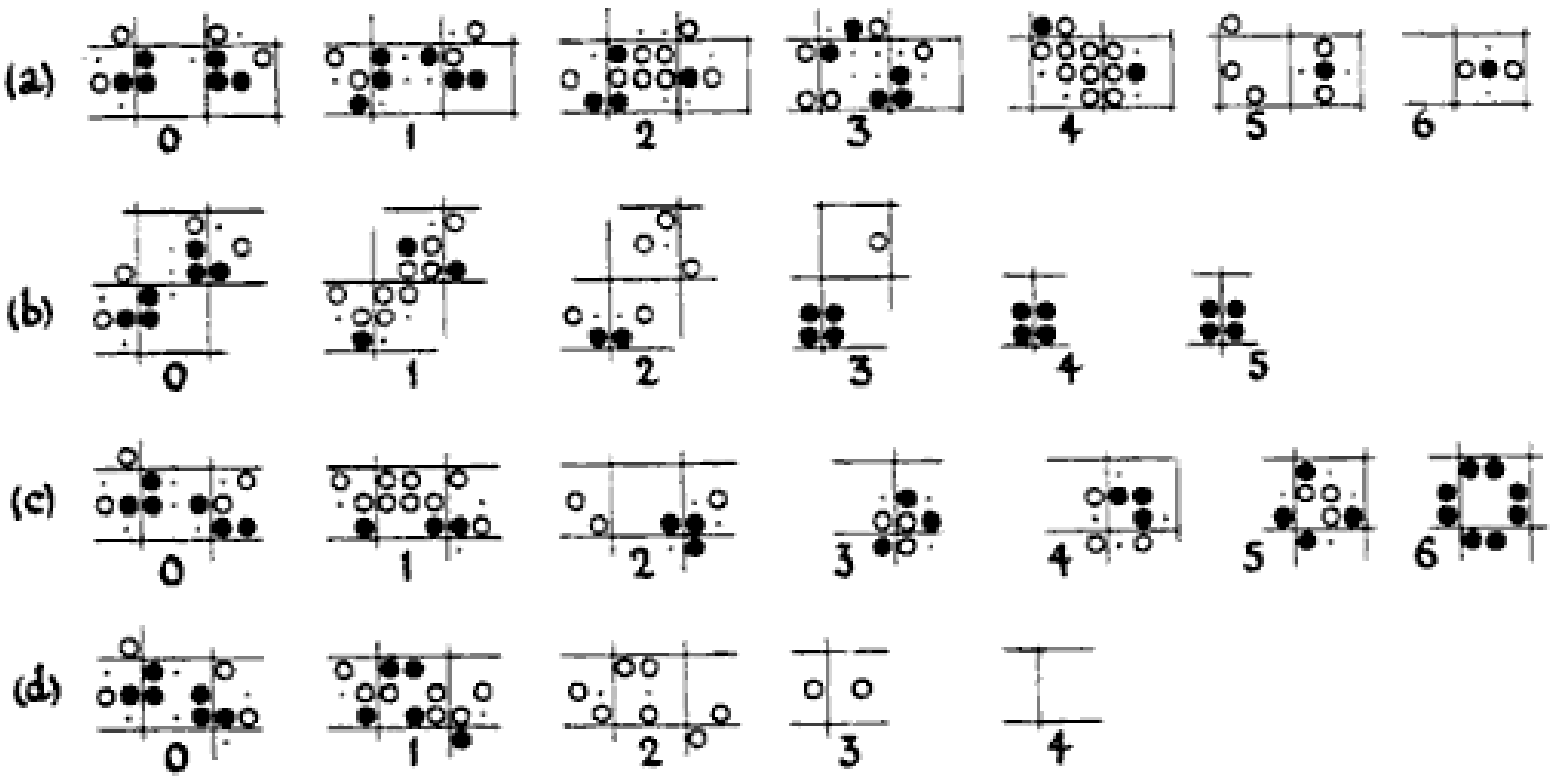
Here we show (glossing over some of the details) how we could use the Game of Life to make functioning logic gates and perform operations on bit streams.

Computers represent information as signals or bit streams. We can use a stream of gliders to represent a signal or bit stream as follows:



Before we get to building our logical parts, let's examine the different reactions we can get when gliders crash into each other. These collisions of gliders will form the building blocks of the reactions that will make up our computer.
We can simluate crashes that result in the following possiblities (we will actually simulate these in class):

- (a) Blinker
- (b) Block
- (c) Pond
- (d) Vanishing reaction

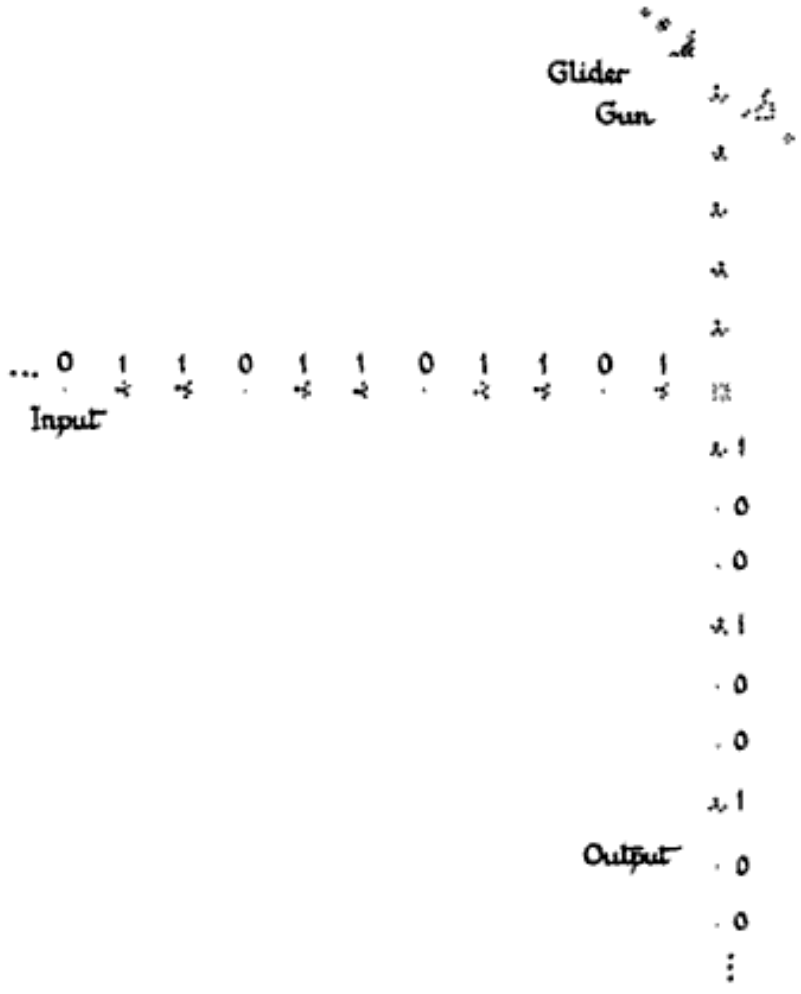

(a)
0
1
2
3
4
5
6
(b)
0
1
2
3
4
5
(c)
0
1
2
3
4
5
6
(d)
0
1
2
3
4

We can use the reactions we learned above to build a logical NOT gate. If we let one stream of gliders be our bit stream, we can "invert" this stream of gliders through the use of a glider gun and vanishing reactions. We can construct and position a glider gun such that every "space" or "0" in our input glider stream allows one glider to escape from the glider gun, while every glider, or "1" in our input stream collides with the glider from the glider gun to make a "0". This is illustrated in the image below:

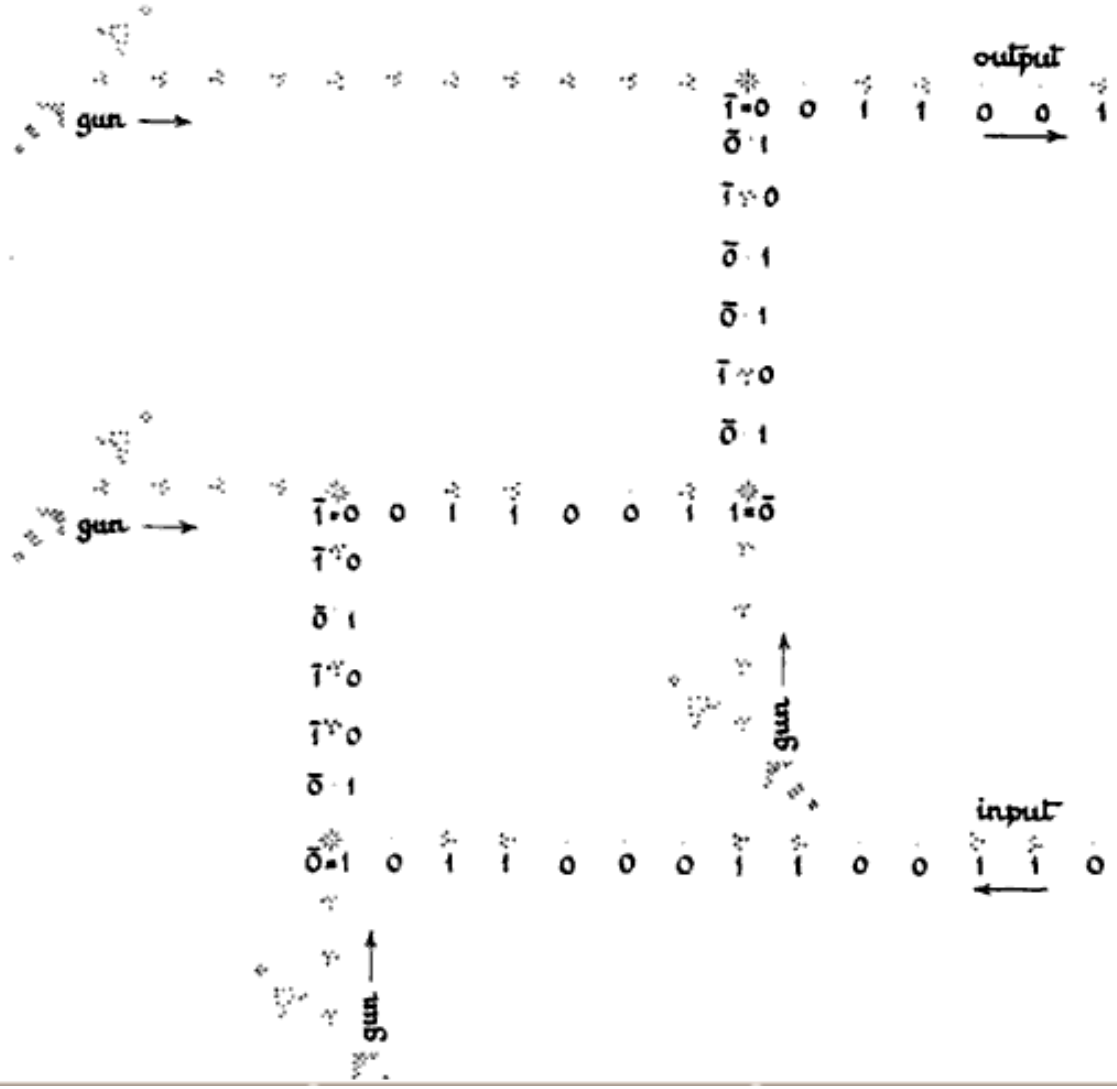
1 1 0 1 1 0 1 1 0 ...

being complemented to

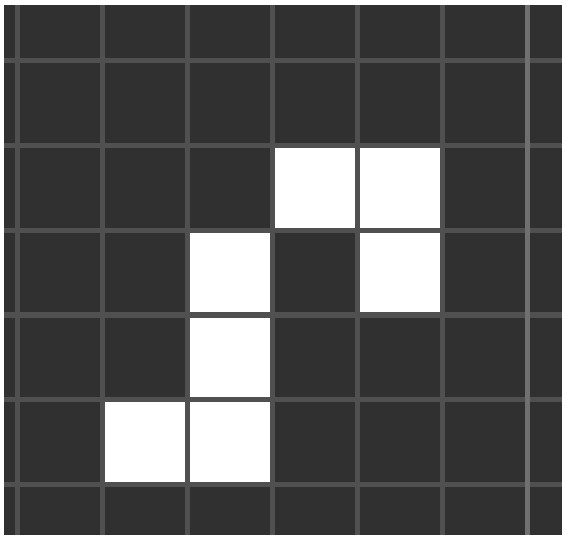
0 0 1 0 0 1 0 0 1 ....



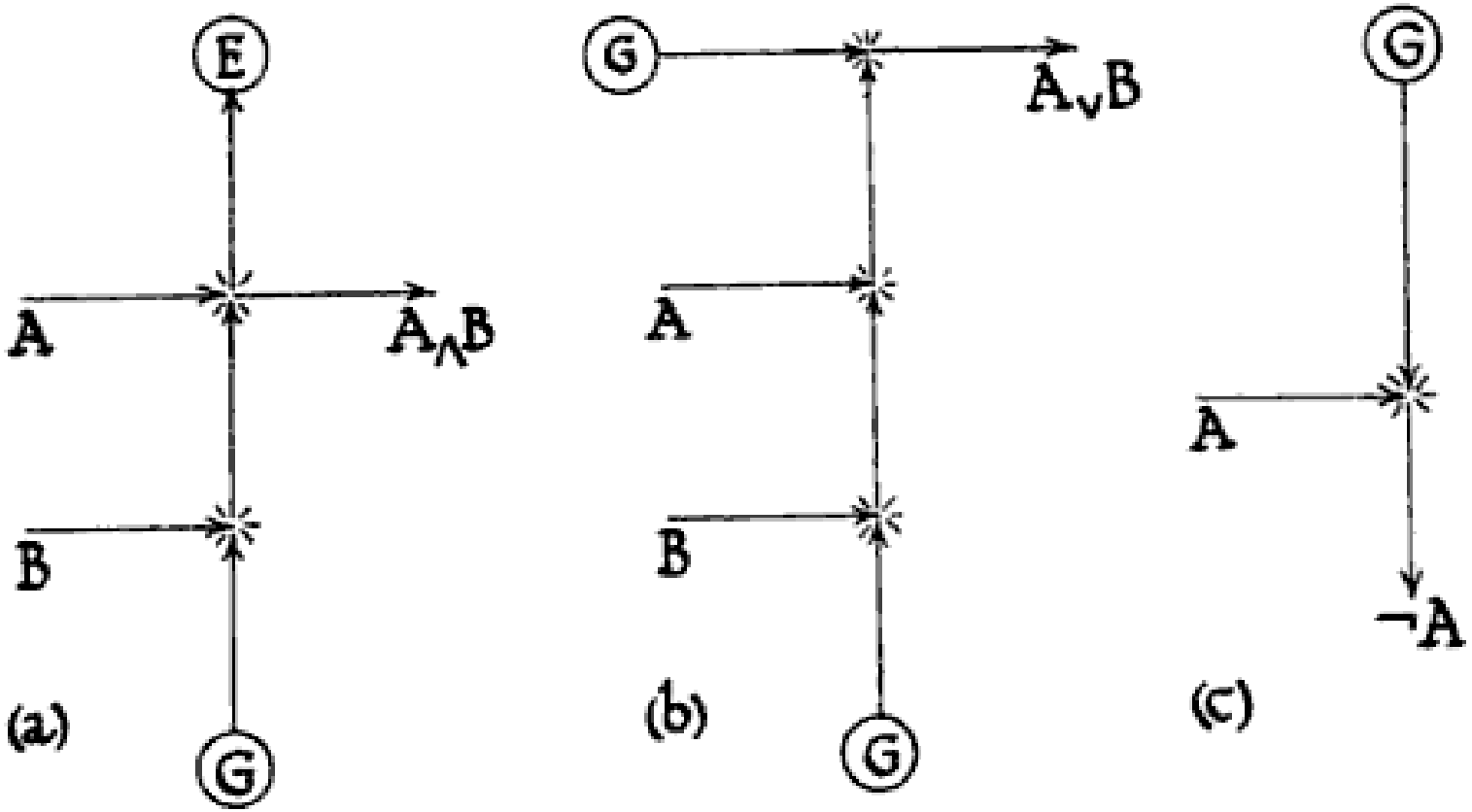
We will not go into details, but for engineering purposes we note that we can manipulate the direction of a glider stream however we want by positioning glider guns to navigate gliders around corners and taking advantage of the different glider-glider reactions we have. From now on we assume we can direct a glider stream to go in whatever direction we want.


output
gun
gun
gun
gun
input

We can also get rid of unwanted glider streams by forming an "eater." The eater is able to devour gliders. (see demo in class)



We also note that we can use gliders as the building blocks for almost all the pieces we will need. For instance, two gliders can collide to make an eater. Gliders can also collide to make blocks and ponds, and therefore a whole glider gun!.


E
A
A∧B
B
(a)
G
G
A∨B
A
B
(b)
G
G
A
¬A
(c)

For the AND gate, we position our two glider streams that we are ANDing perpendicular to the line of fire of a glider gun, with an eater to eat up unwanted gliders. Let us go through all possible cases and see why this gives us an AND gate.

- A = 1, B = 1
  If B's stream has a glider, it will vanish when it collides with the glider from G. When the stream gets to A, A's glider will be able to pass through where the vanishing reaction left a hole, making the value of A and B 1, as expected.
- A = 1, B = 0
  If B's stream has no glider, the glider from the gun will reach a collision point with A's glider and cause a vanishing reaction, making the output of A and B 0, as expected.

- A = 0, B = 1
  If B's stream has a glider, it will vanish when it collides with the glider from G. When the stream gets to A, there will be an empty space, and so the glider stream will continue upwards and get eaten by the eater, E. Therefore the output A and B will be 0, as expected.
- A = 0, B = 0
  If B's stream has no glider, the stream from the glider gun will continue to cross A's path. A also has no glider, so the glider fro the gun continues on to the Eater to be eaten, and the output of A and B is 0, as expected.

For the OR gate, we again position our two input streams perpendicular to the output of the glider stream, and make use of a second glider gun as well. Let us go through all possible cases and see why this gives us an OR gate.

- A = 1, B = 1
  If B's stream has a glider, it will collide with the glider from the gun and create a vanishing reaction. If A has a glider, but there is no space from the incoming glider stream, no glider from the stream will reach the next node, a collision path with another glider gun. Since there is no glider from the incoming stream, the glider from the second gun will reach the output, making the value of A or B 1, as expected.

- $A = 1$, $B = 0$
  If B's stream has no glider, the glider from the gun will continue onward to collide with A, leaving a space to interact with the second glider gun. Therefore, the glider from the second gun will reach the output, making the value of A or B 1, as expected.
- $A = 0$, $B = 1$
  If B's stream has a glider, it will create a vanishing reaction with the stream from G, leaving a space for A. A also has no glider, and therefore there will be no incoming glider to collide with the glider stream from the second gun, and the output will again be 1, as expected.

- $A = 0$, $B = 0$
  If both A and B have no gliders, there is nothing to interrupt the path of the first glider gun. Therefore, gliders from both of the glider guns will collide and vanish, and therefore the make the output, A or B, be 0, as expected.

- Therefore, using these parts, we could construct a slow and primitive, but functioning, computer out of the Game of Life. The Game of Life exhibits a property we call **universality**, meaning that it can theoretically compute anything that can be computed.
- For more details on how exactly we could construct a computer from the Game of Life, see Winning Ways.

Another equivalent statement to saying Life is universal is that we can build a Turing-machine from it. We start by describing Turing Machines. A Turing Machine has:

- A finite set of states
- Rules for transitioning between states
- An infinite sequence of cells (called a "tape")
- A set of symbols describing the possible contents of each cell in the tape
- The ability to read and write the symbol in a single cell
- The ability to move along the tape to access different cells.

Each state transition rule specifies:

- The current state
- The symbol read from the current cell
- The state to move to
- The symbol to write to the current cell
- The direction to move the machine along the tape

- On each time step, the Turing Machine (TM) reads from the current cell on the tape, finds the appropriate transition rule that matches the current state and symbol just read, then changes to the specified new state, writes the specified symbol, and moves the machine in the specified direction.
- Here in class we will go through an example Turing machine calculation (slides 37-54 of Rendell slides)

- The famous **Church-Turing Thesis** states that "everything computable is computable by a Turing Machine." This statement is not proven, but is almost univerally accepted. If it is true, then TM's have power greater than modern day computers.
- We call something **Turing-complete** if it has rules followed in sequence that have the same computational power as a Turing Machine.
- If we can show that the Game of Life is Turing-complete, then we can say that Life is as powerful as any computer, and that we can compute any computational problem using the Game of Life.

Now let us see how we might actually construct a Turing Machine from the building blocks of the Game of Life.
We will go through slides 61-80 of the Rendell notes in class to see the construction. We will also show a demo of the TM in action!

References

- http://www.math.com/students/wonders/life/life.html
- http://www.nathanieljohnston.com/index.php/tag/conways-game-of-life/
- Winning Ways
- Cellular Automata and Turing Universality in the Game of Life by Paul Rendell, presented by David Thue