Smalltalk: Conway’s Game of Life

Brian Heinold

Mount St. Mary’s University

February 16, 2018
The “game” consists of a grid of cells. Each cell is either on (alive) or off (dead).
A cell has 8 neighbors (above, below, left, right, and diagonals).
Rules

1. Live cells stay alive if 2 or 3 of their neighbors are alive.
2. Live cells with under 2 live neighbors die of loneliness.
3. Live cells with over 3 live neighbors die of overcrowding.
4. Dead cells come to life if exactly 3 neighbors are alive.
1. Live cells stay alive if 2 or 3 of their neighbors are alive.
1. Live cells stay alive if 2 or 3 of their neighbors are alive.
2. Live cells with under 2 live neighbors die of loneliness.
1. Live cells stay alive if 2 or 3 of their neighbors are alive.
2. Live cells with under 2 live neighbors die of loneliness.
3. Live cells with over 3 live neighbors die of overcrowding.
1. Live cells stay alive if 2 or 3 of their neighbors are alive.
2. Live cells with under 2 live neighbors die of loneliness.
3. Live cells with over 3 live neighbors die of overcrowding.
4. Dead cells come to life if exactly 3 neighbors are alive.
A walk-through — five live cells to start
A walk-through — middle cell dies of overcrowding
A walk-through — the other live cells are okay
A walk-through — four new cells come to life!
A walk-through — the next generation
A walk-through — four cells die of overcrowding
A walk-through — the other live cells are okay
A walk-through — four new cells come to life
A walk-through — the third generation
Another walk-through
Another walk-through — outside cells die of loneliness
Another walk-through — middle cell is okay
Another walk-through — two new cells come to life
Another walk-through — outside cells die of loneliness
Another walk-through — middle cell is okay
Another walk-through — two new cells come to life
Another walk-through — back where we started!
Oscillating patterns

- Blinker

- Toad

- Beacon
Throughout this talk, I’ll be switching back and forth between slides and demonstrations.

If you want to try it out for yourself, an online app is here:

https://brianheinold.net/life.html
More oscillators

Pulsar (period 3):

Pentadecathlon (period 15)
## Even more oscillators


<table>
<thead>
<tr>
<th>Period</th>
<th>First discovered</th>
<th>Discoverer</th>
<th>Year of discovery</th>
<th>Smallest known</th>
<th>Minimum # of cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>blinker</td>
<td>John Conway</td>
<td>1970</td>
<td>blinker</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>pulsar</td>
<td>John Conway</td>
<td>1970</td>
<td>tanner</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>pinwheel</td>
<td>Simon Norton</td>
<td>1970</td>
<td>mazing, mold</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>octagon 2</td>
<td>Sol Goodman, Arthur Taber</td>
<td>1971</td>
<td>pseudo-barberpole</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>$rats</td>
<td>David Buckingham</td>
<td>1972</td>
<td>unix</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>burloafermator</td>
<td>David Buckingham</td>
<td>1972</td>
<td>28P7.1, 28P7.2, 28P7.3, burloafermator</td>
<td>28</td>
</tr>
<tr>
<td>8</td>
<td>figure eight</td>
<td>Simon Norton</td>
<td>1970</td>
<td>figure eight</td>
<td>12</td>
</tr>
<tr>
<td>9</td>
<td>worker bee</td>
<td>David Buckingham</td>
<td>1972</td>
<td>29P9</td>
<td>29</td>
</tr>
<tr>
<td>10</td>
<td>42P10.1</td>
<td>David Buckingham</td>
<td>≤1976</td>
<td>24P10</td>
<td>24</td>
</tr>
<tr>
<td>11</td>
<td>38P11.1</td>
<td>David Buckingham</td>
<td>1977</td>
<td>rattlesnake</td>
<td>33</td>
</tr>
<tr>
<td>12</td>
<td>dinner table</td>
<td>Robert Wainwright</td>
<td>1972</td>
<td>dinner table</td>
<td>33</td>
</tr>
<tr>
<td>13</td>
<td>65P13.1</td>
<td>David Buckingham</td>
<td>1976</td>
<td>34P13</td>
<td>34</td>
</tr>
<tr>
<td>14</td>
<td>tumbler</td>
<td>George Collins, Jr.</td>
<td>1970</td>
<td>tumbler</td>
<td>16</td>
</tr>
<tr>
<td>15</td>
<td>pentadecathlon</td>
<td>John Conway</td>
<td>1970</td>
<td>pentadecathlon</td>
<td>12</td>
</tr>
<tr>
<td>16</td>
<td>two pre-L hasslers</td>
<td>Robert Wainwright</td>
<td>1983</td>
<td>Achim's p16</td>
<td>32</td>
</tr>
<tr>
<td>17</td>
<td>54P17.1</td>
<td>Dean Hickerson</td>
<td>1997</td>
<td>Honey thieves</td>
<td>36</td>
</tr>
<tr>
<td>18</td>
<td>117P18</td>
<td>David Buckingham</td>
<td>≤1991</td>
<td>43P18</td>
<td>43</td>
</tr>
<tr>
<td>19</td>
<td>None found</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>146P20</td>
<td>Noam Elkies</td>
<td>1996</td>
<td>mold on furarolo</td>
<td>30</td>
</tr>
</tbody>
</table>
A still life

This is stable. It will not change unless something affects it from outside.
A still life

This is stable. It will not change unless something affects it from outside.

Every live cell has 3 live neighbors; every dead cell has 2 live neighbors.
More still lifes

- Tub
- Beehive
- Lake 2
More still lifes


<table>
<thead>
<tr>
<th>Live cells</th>
<th>Strict still lifes</th>
<th>Pseudo still lifes</th>
<th>Quasi still lifes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Count (A019473)</td>
<td>Examples</td>
<td>List</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>block, tub</td>
<td>Full list</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>boat</td>
<td>Full list</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>beehive, ship</td>
<td>Full list</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>eater 1, loaf</td>
<td>Full list</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>canoes, pond</td>
<td>Full list</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>hat, integral sign</td>
<td>Full list</td>
</tr>
<tr>
<td>10</td>
<td>25</td>
<td>boat-lie, loop</td>
<td>Full list</td>
</tr>
<tr>
<td>11</td>
<td>46</td>
<td>elevenor</td>
<td>Full list</td>
</tr>
<tr>
<td>12</td>
<td>121</td>
<td>honeycomb, table on table</td>
<td>Partial list</td>
</tr>
<tr>
<td>13</td>
<td>240</td>
<td>sesquihat</td>
<td>Partial list</td>
</tr>
<tr>
<td>14</td>
<td>619</td>
<td>fourteen, paperclip</td>
<td>Partial list</td>
</tr>
<tr>
<td>15</td>
<td>1,363</td>
<td>moose antlers</td>
<td>Partial list</td>
</tr>
<tr>
<td>16</td>
<td>3,266</td>
<td>bi-cap, scorpion</td>
<td>Partial list</td>
</tr>
<tr>
<td>17</td>
<td>7,773</td>
<td>twin hat</td>
<td>Partial list</td>
</tr>
<tr>
<td>18</td>
<td>19,044</td>
<td>dead spark coil</td>
<td>Partial list</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This pattern “glides”. It oscillates, but moves as it oscillates. It is extremely important.
Glider moving
Glider moving
Glider moving
Glider moving
Glider gun

This pattern generates gliders.
This pattern generates gliders.

It shows that infinite growth is possible (starting from a finite number of cells, it is possible to generate an unlimited amount).
There are various other types of gliding patterns, like this one, called a *lightweight spaceship*:
The R-Pentomino below is an example of a “Methuselah pattern”.

![R-Pentomino Diagram]
The R-Pentomino below is an example of a “Methuselah pattern”.

The terminology comes from the fact that it takes a long time before it settles into a steady-state (1103 generations).
History

Invented by John Horton Conway in late 1960s.
History

Invented by John Horton Conway in late 1960s.

Some fun things he is known for from his Wikipedia entry: Surreal numbers, Monstrous moonshine, Doomsday algorithm, Look-and-say sequence, Icosians, Free will theorem.
Initially the game was run by hand on tables with counter pieces and necklace shells in the University of Cambridge math common room.
Initially the game was run by hand on tables with counter pieces and necklace shells in the University of Cambridge math common room.

Article published in Martin Gardner’s *Scientific American* column about it in 1970.
Initially the game was run by hand on tables with counter pieces and necklace shells in the University of Cambridge math common room.

Article published in Martin Gardner’s *Scientific American* column about it in 1970.

After that, it really took off, with people all over writing computer programs for it.
Initially the game was run by hand on tables with counter pieces and necklace shells in the University of Cambridge math common room.

Article published in Martin Gardner’s *Scientific American* column about it in 1970.

After that, it really took off, with people all over writing computer programs for it.

Various papers and were written, informal journals created, and 50 years later people are still discovering new things about it.
A one-line program for the Game of Life

From the Wikipedia page on the APL programming language

```apl
life←{↑1 ν.∧3 4=+/,-1 0 1.θ1 0 1.Φcω)
```
A one-line program for the Game of Life

From the Wikipedia page on the APL programming language

\[
\text{life}←{↑1 \ o v. \land 3 \ 4=+/,¬1 \ 0 \ 1. \ Θ¬1 \ 0 \ 1. \ Φ<ω}
\]

More reasonably, most Mount CS majors could probably write a Game of Life program in no more than a few hours in their favorite programming language.
It’s fun to start with a random board and see what evolves.
Putting glider guns together

If you put place glider guns in the right locations you can get interesting effects, like these gliders that cancel each other out.
You can also block a glider gun.
Putting these two ideas together, you can build logic gates, like this AND gate:
Putting these two ideas together, you can build logic gates, like this AND gate:

Removing either or both of the two blockers will cause the gliders from A or B to wipe out the glider stream from the source.
Creating a computer

- We can build logic gates.

[Video Link: https://youtu.be/8unMqSp0bFY]
Creating a computer

- We can build logic gates.

- We can also use these ideas to build things that remember state (memory).
Creating a computer

- We can build logic gates.

- We can also use these ideas to build things that remember state (memory).

- Put all this together, and we can build an actual working computer in Life:

  https://youtu.be/8unMqSp0bFY
Life can solve anything a computer can do.

So in theory, Life can do anything that a computer can do.
Life can solve anything a computer can

So in theory, Life can do anything that a computer can do (such as a program to play the Milton Bradley board game)
Life can solve anything a computer can

So in theory, Life can do anything that a computer can do (such as a program to play the Milton Bradley board game)

In computer science, people say that Life is Turing Complete.
Fun to change around the rules for when cells turn on and off and see how things change.
Fun to change around the rules for when cells turn on and off and see how things change.

Life is an example of a *cellular automaton*. 
Generalizations

- Fun to change around the rules for when cells turn on and off and see how things change.

- Life is an example of a *cellular automaton*.

- Cellular automata serve as models for physical phenomena like crystallization.
Fun to change around the rules for when cells turn on and off and see how things change.

Life is an example of a cellular automaton.

Cellular automata serve as models for physical phenomena like crystallization.

They are used in simulations of things like traffic flow, spread of a disease, spread of information, and much more.
In whatever time we have left, let’s look at some more demonstrations.

Image credits:

*Life* board game: enjoyingthegameoflife.blogspot.com