Variant B

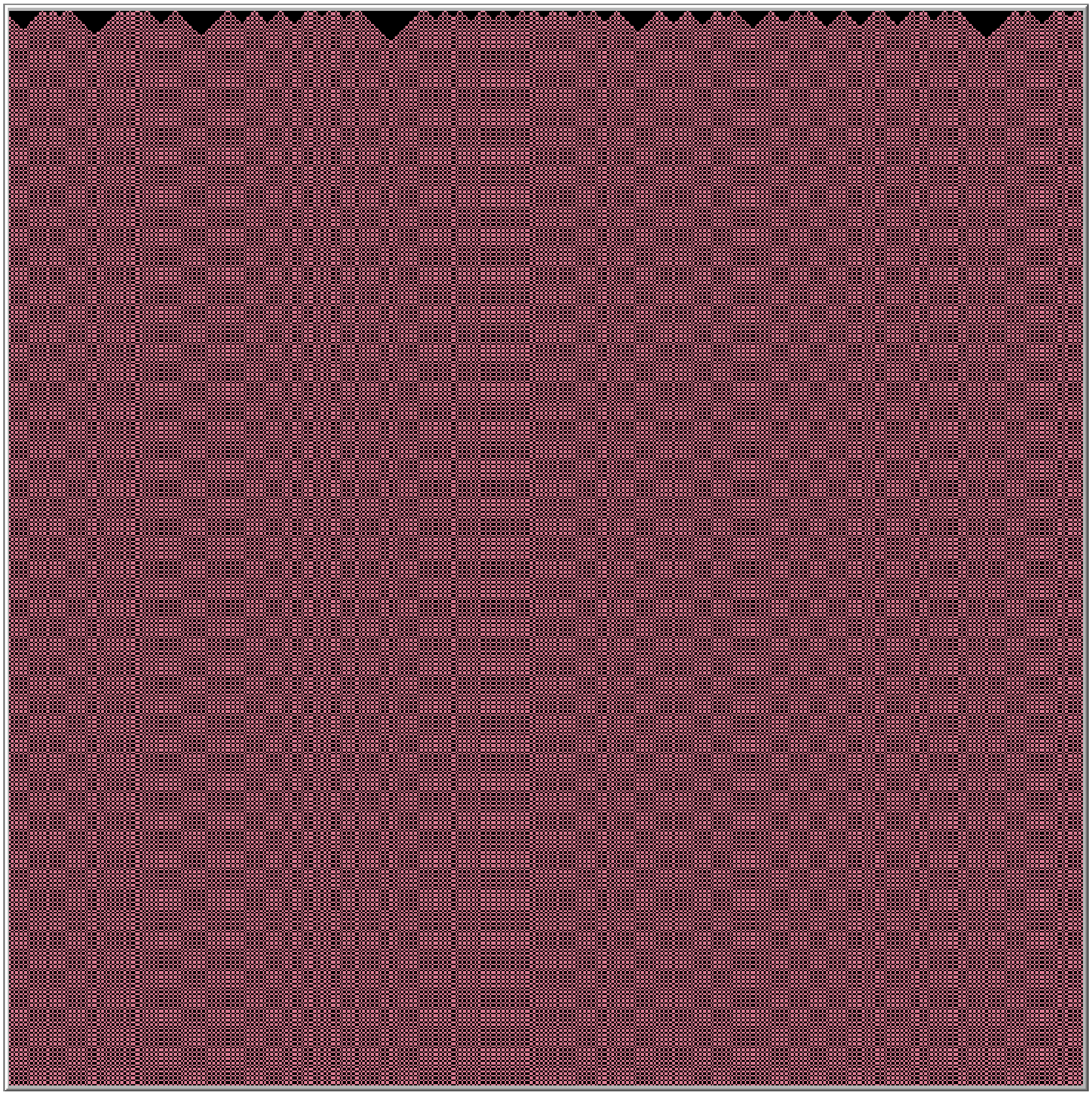
Task 1

Variant b: rules 50, 54, 57, 58, 60, 97

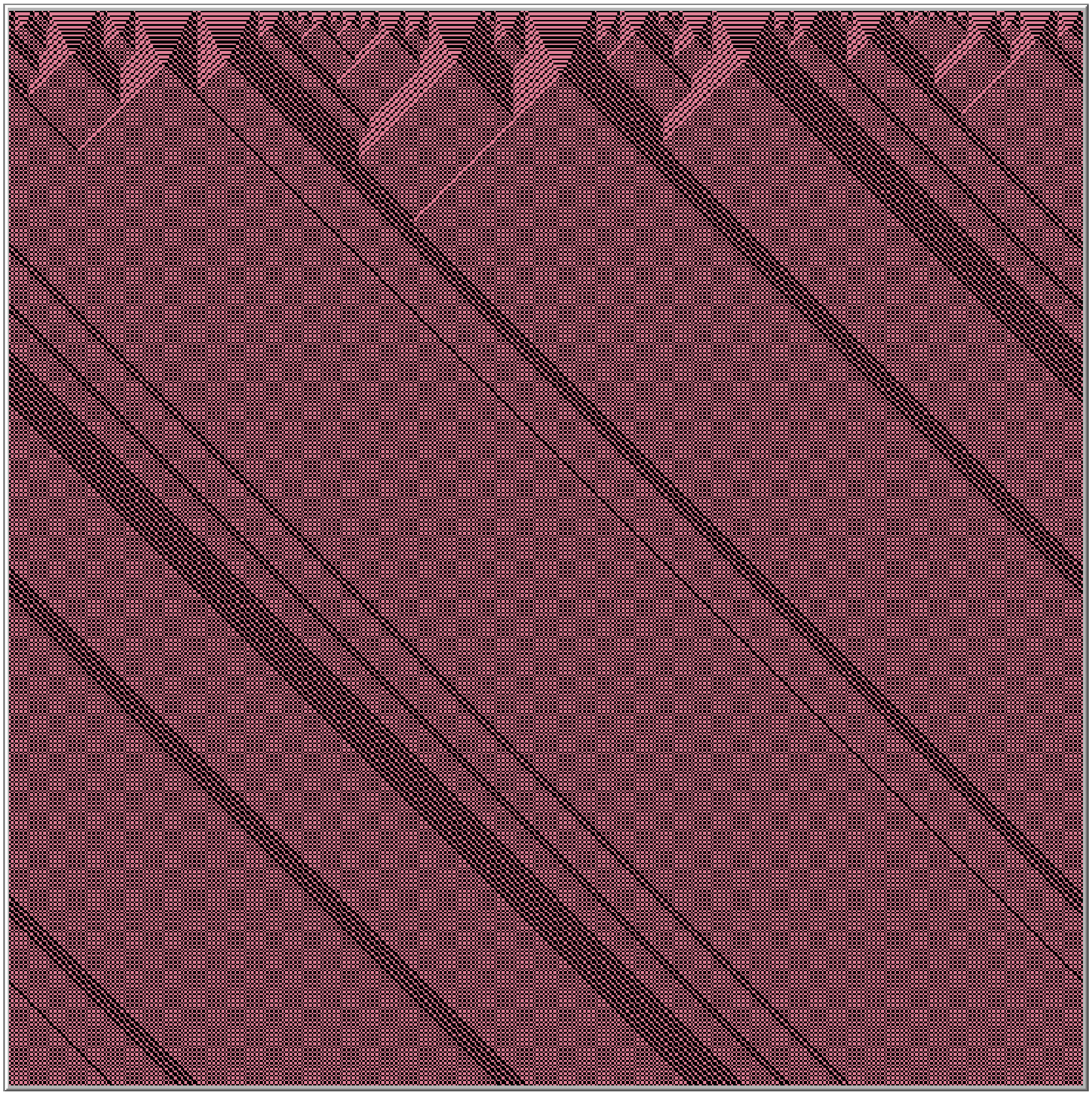
2.

Class 1: none of our variant
Class 1 converges to a uniform state - all cells end up being alive (or dead).

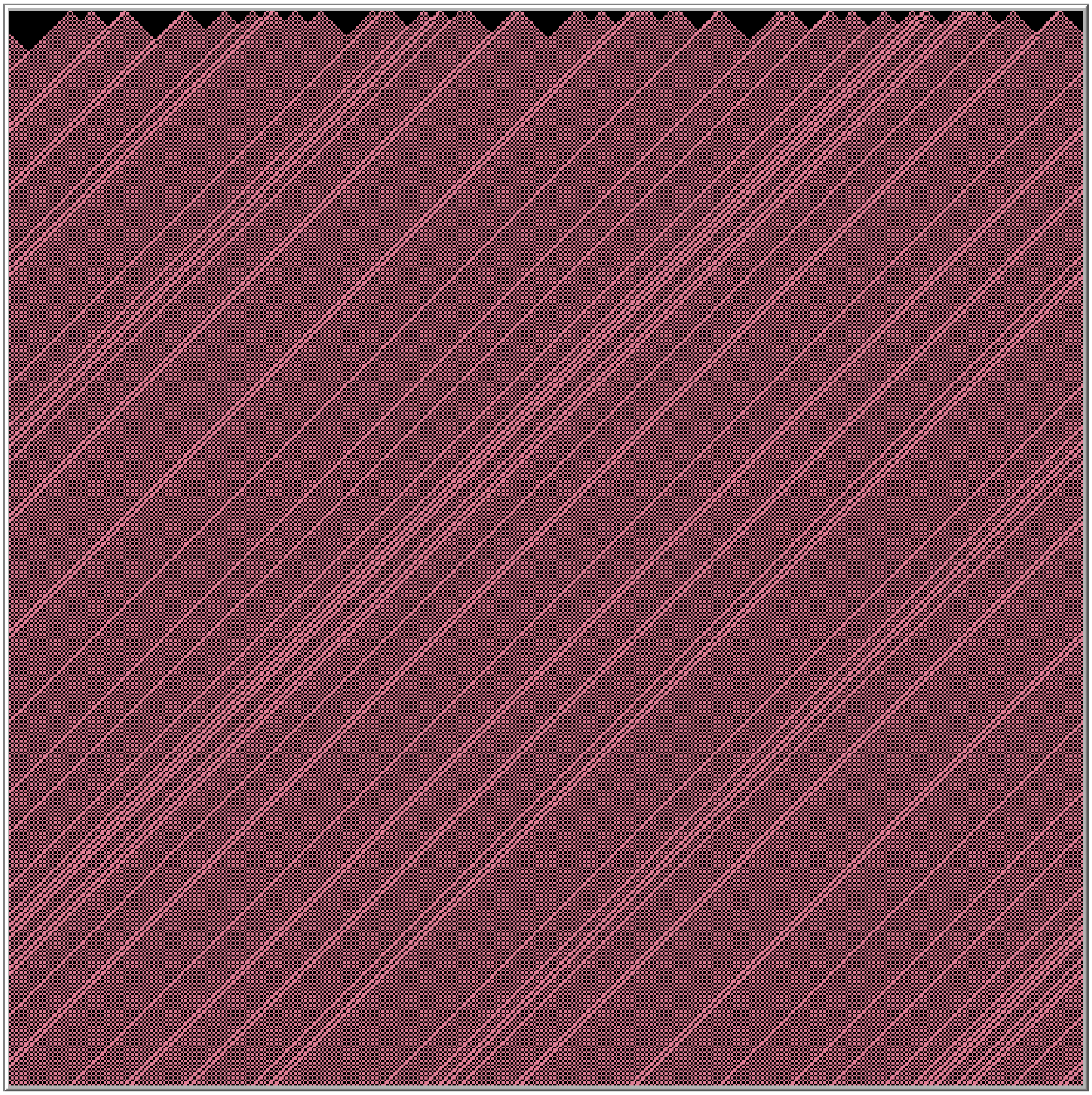
Class 2: 50, 57, 58, 97
Class 2 after a few steps results in periodical structures or nested patterns.



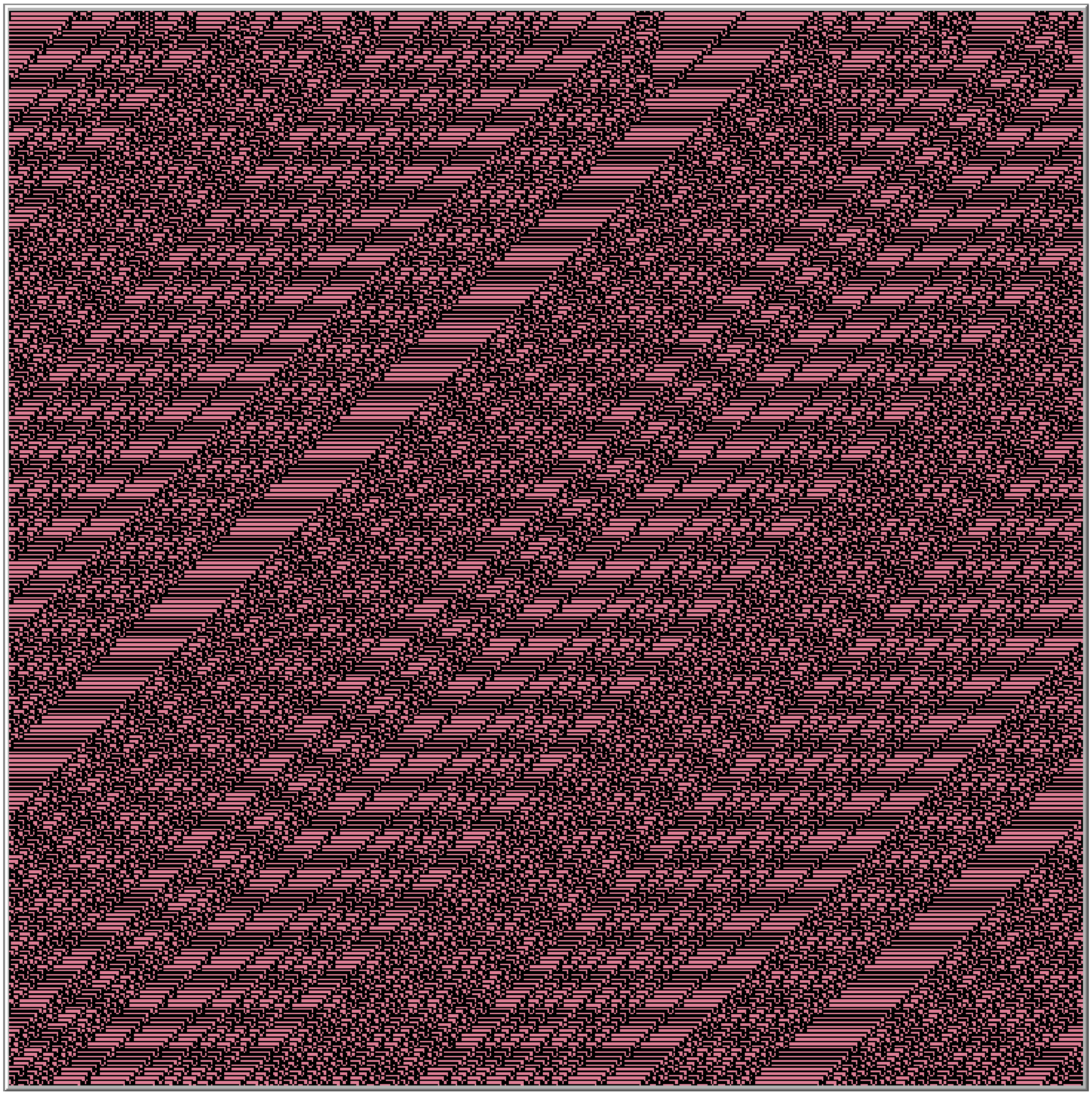
Rule 50 (it has a periodic pattern)



Rule 57 (it is chaotic at first but after a few steps we can see a pattern as diagonal lines)



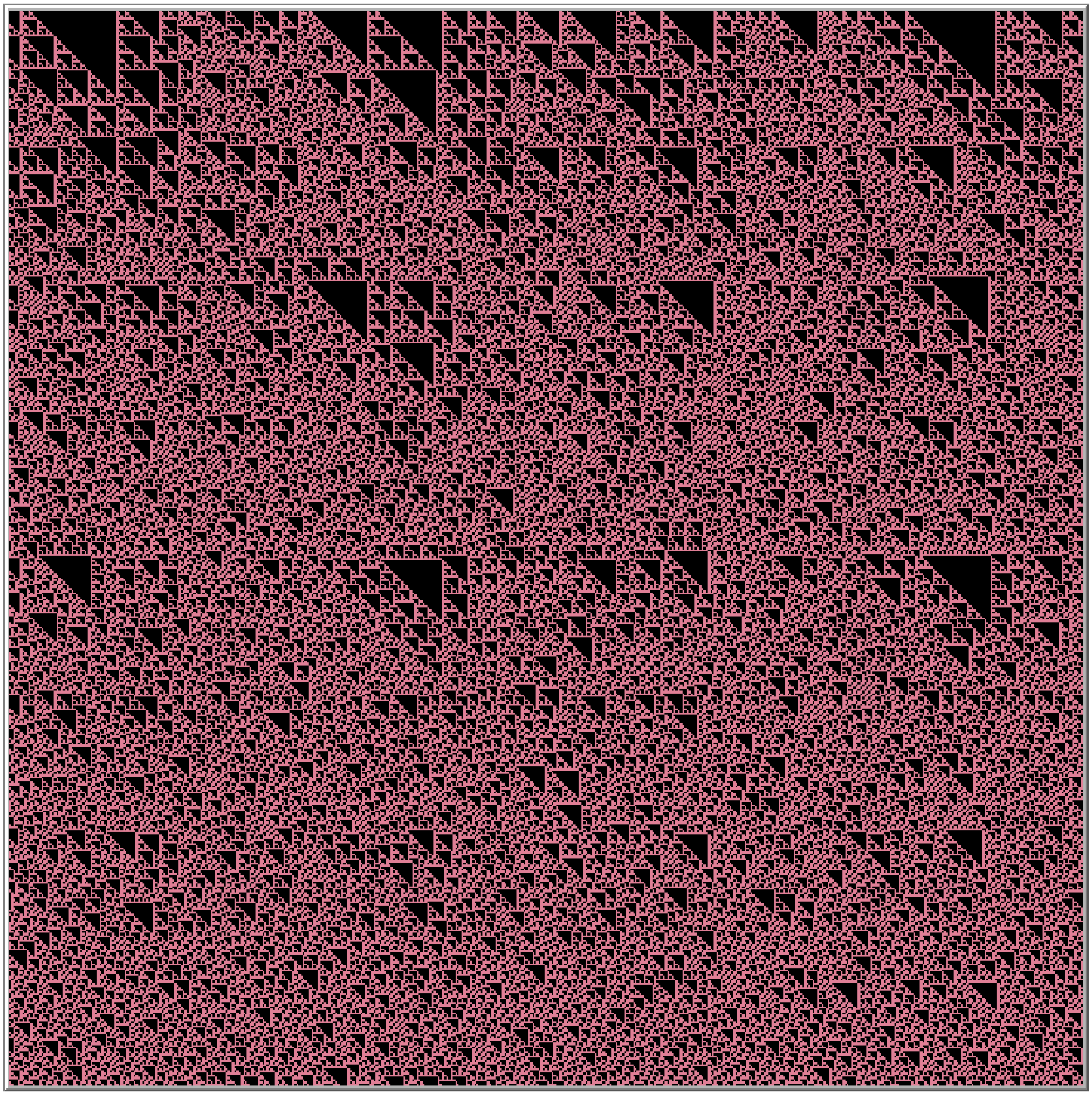
Rule 58 (it clearly has a pattern as diagonal lines)



Rule 97 (it seems pretty chaotic, but we can see a pattern as periodic diagonal lines)

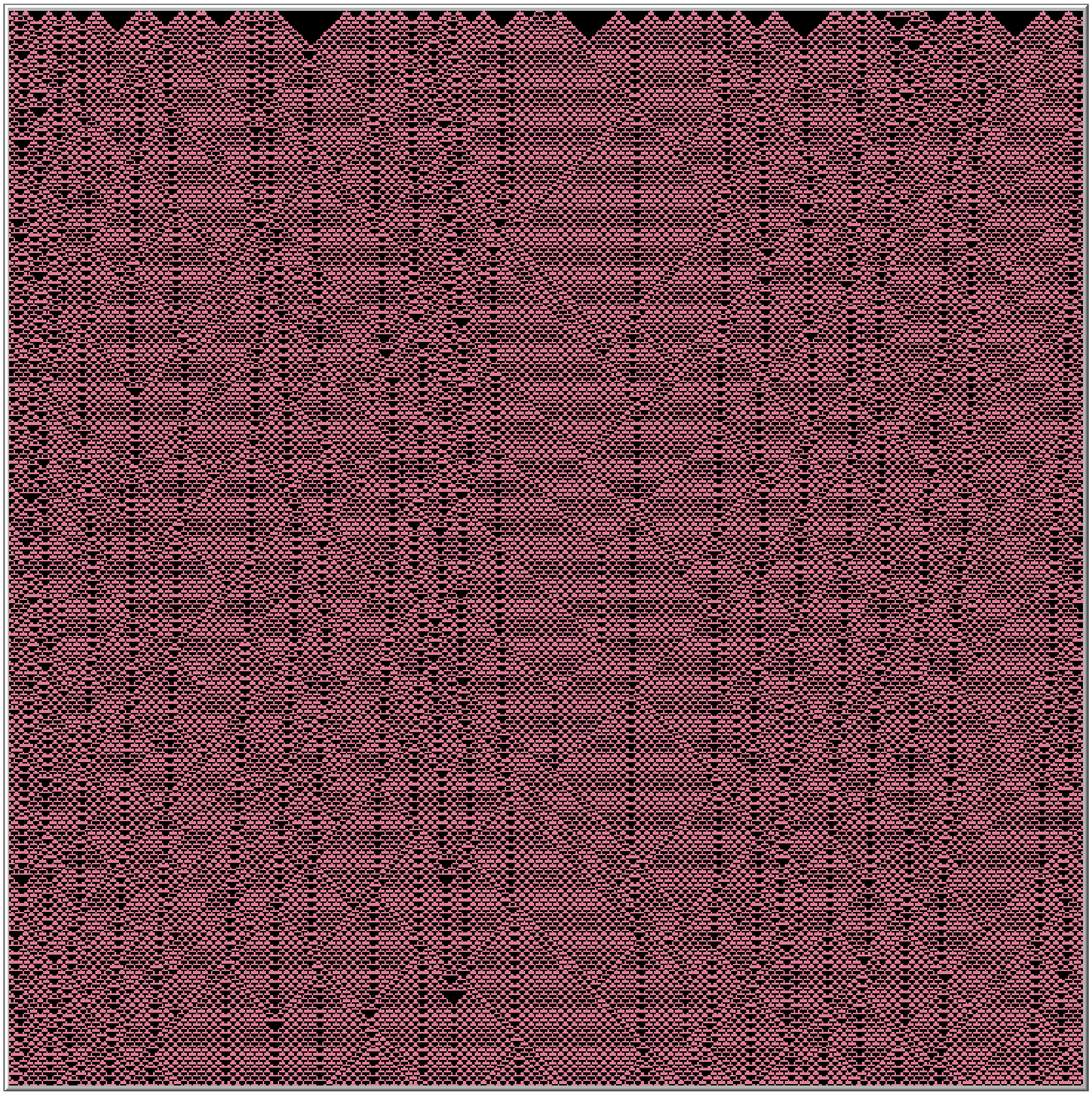
Class 3: 60

Class 3 contains patterns but they are disordered and not periodic.



Rule 60 (it has patterns as triangles, but they are chaotic)

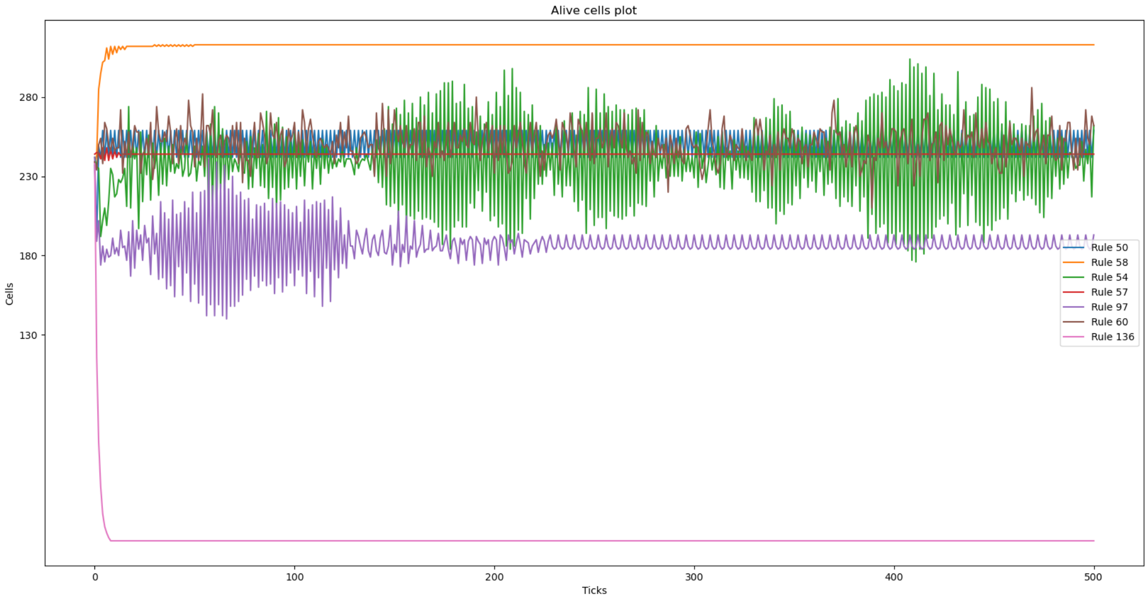
Class 4: 54
Class 4 doesn’t have the exact pattern, on the contrary, it may yield complex aperiodic and localized structures.



Rule 54 (some parts with triangles may look like patterns, but it is not periodic, not random, and not deterministic)

3.

Here is a plot of live cells for each generation of rules listed above.



4.

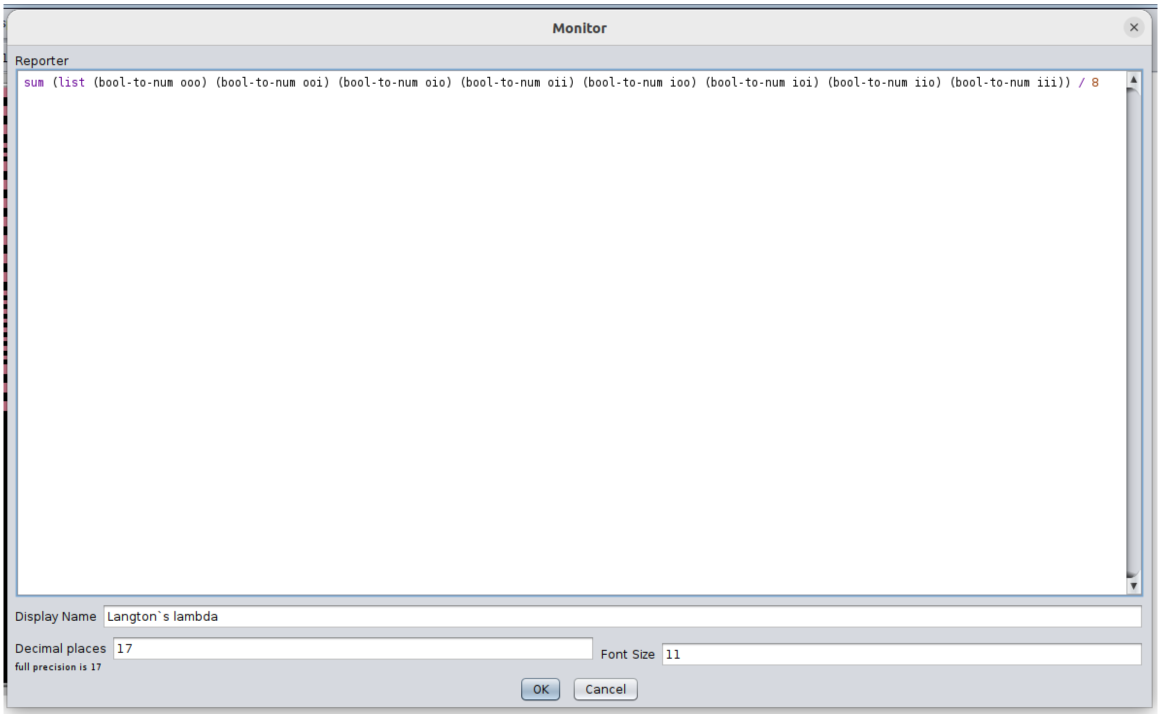
We added rule 136 to display class 1, as no such class was presented in our variant.
As we can see from the plot, some differences between the rules of different classes can be seen. The most obvious one is rule 136 which belongs to class 1, because the cells die very quickly, we can see it from the plot. The next special case is rule 54, the plot of it is quite chaotic, the most unpredictable one, thus implying class 4. Rule 50 is behaving in a stable pattern, which implies similarity to class 2. Rules 57 and 58 look pretty similar to that class too. Rule 97 looks chaotic at the beginning, but stays periodical later, so it looks like class 2. And the last one, rule 60 looks pretty chaotic, but not enough for class 4, so we consider classifying it as class 3.

5-6.

Langton’s Lambda can be calculated by the following formula:

(number of rules in which the new state of the cell is living) / 8

To implement it we used Monitor:



Reporter

```
sum (list (bool-to-num ooo) (bool-to-num ooi) (bool-to-num oio) (bool-to-num oii) (bool-to-num ioo) (bool-to-num ioi) (bool-to-num iio) (bool-to-num iii)) / 8
```

Display Name: Langton`s lambda

Decimal places: 17

full precision is 17

Font Size: 11

APAC is an average percent of alive cells. Here is the code for calculating it after 200th tick:

```
to calculate-APAC-for-one-rule
  let cells-alive count patches with[pycor = row and on? = true]
  let cells-total count patches with[pycor = row]
  let apac (cells-alive / cells-total) * 100
  set all-apac all-apac + apac
end

to calculate-APAC
  set all-apac 0
  repeat 200 [
    go
    calculate-APAC-for-one-rule
  ]
  set average all-apac / 200
end
```

Rule	**Class**	**Langton’s lambda**	**APAC**
50	2	0.375	49.25649 %
54	4	0.5	47.77745 %
57	2	0.5	48.21557 %

58	2	0.5	51.52994 %
60	3	0.5	43.35129 %
97	2	0.375	38.36427 %
136	1	0.25	0.003992 %

Task 2

The figure selection process

We begin by generating random figures in python. The building process is such that we avoid a lot of straggler “dead soon” cells and duplicates. The set of generated coordinates we write into a csv file

The implementation pipeline

Before each BS experiment we have to manually reopen the figure csv by clicking “setup-csv-manual”. Then each run reads the next line from the file. The experiment must be run on one core (to avoid bad data). The BS collects information about the number of living cells. The run is terminated if all cells die or they have reached a steady state or the number of ticks exceeds 1000. The steady state is expressed as the stabilization of the cell population on a given time interval. AKA the difference between min and max cell count on that interval is below a certain threshold.

The results

We performed our analysis in Excel.
The white cells are alive

4-cell-figure

	best max alive	best final	result sum
figure number	3	3	3
result	20.00	12.00	32.00

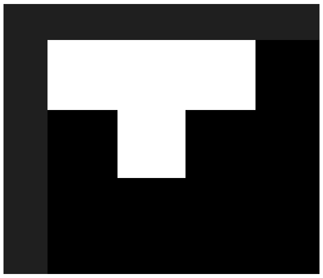


Fig. 3

5-cell-figure

	best max alive	best final	result sum
figure number	5	484	484
result	529.00	400.00	929.00

The figures that yielded comparable results:

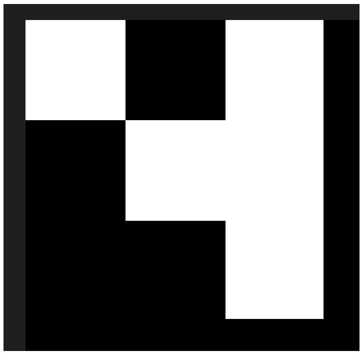


Fig. 484

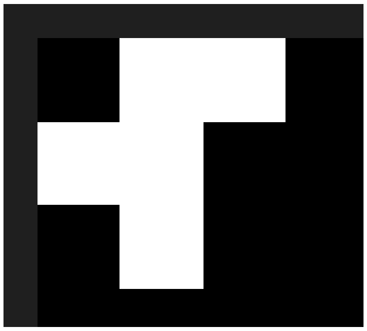
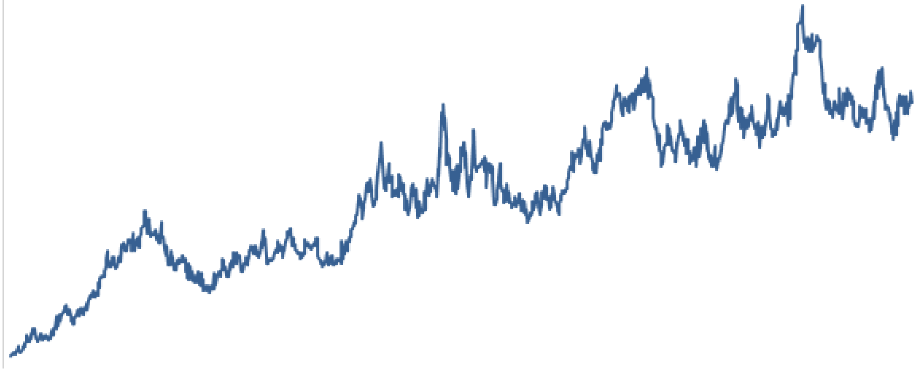
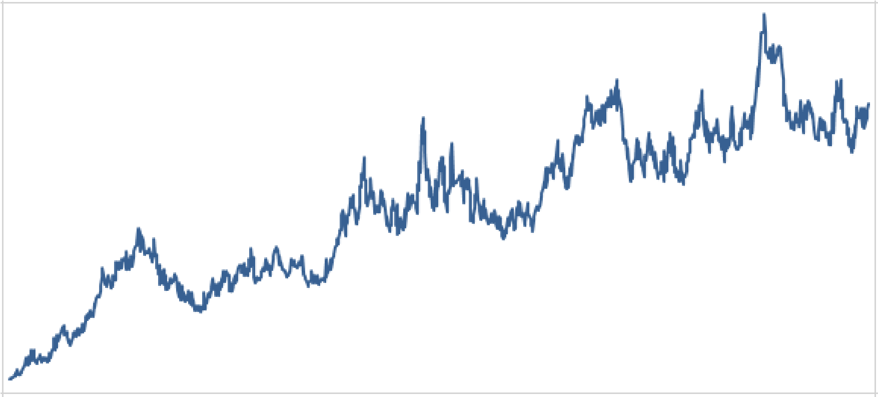


Fig.5

Corresponding graphs:





6-cell-figure

	best max alive	best final	result sum
figure number	33	85	85
result	529.00	400.00	929.00

The figures that yielded comparable results:

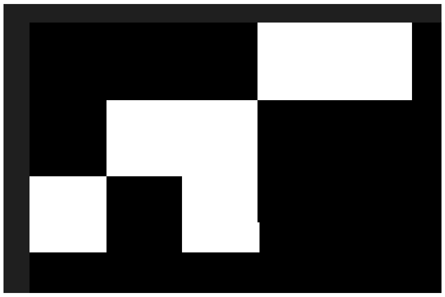


Fig. 85

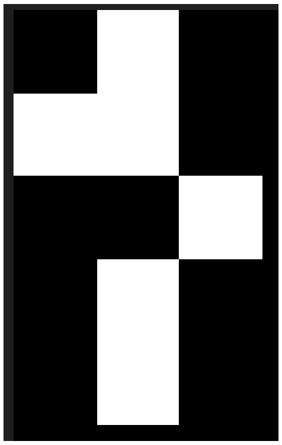


Fig. 33

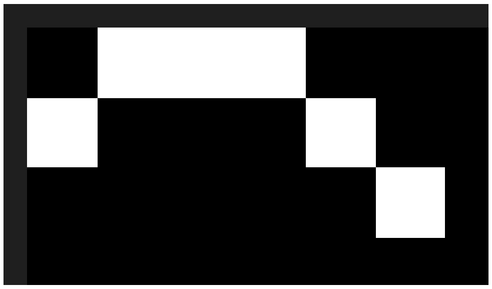


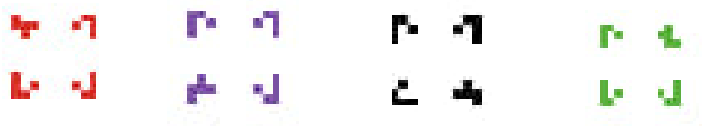
Fig. 201

The discussion

It is surprising that despite having different cell sizes the final sum is the same. Also, there are many different figures that yield the same result and are just variations of pantomino figure. It is probably the cause of some cells dying early and the figures converging to the same one

Task 3

The selected figures



The idea is to send drones in each direction to disrupt other players and ensue chaos, and use 5pantomino, since it’s the best and most compact. Each configuration performed best for its corner in our test runs. Also, the initial rotation of 5pantomino can significantly alter the future state of the game and decide whether we are doomed, or destined to win.