

# “INSTIGRID”

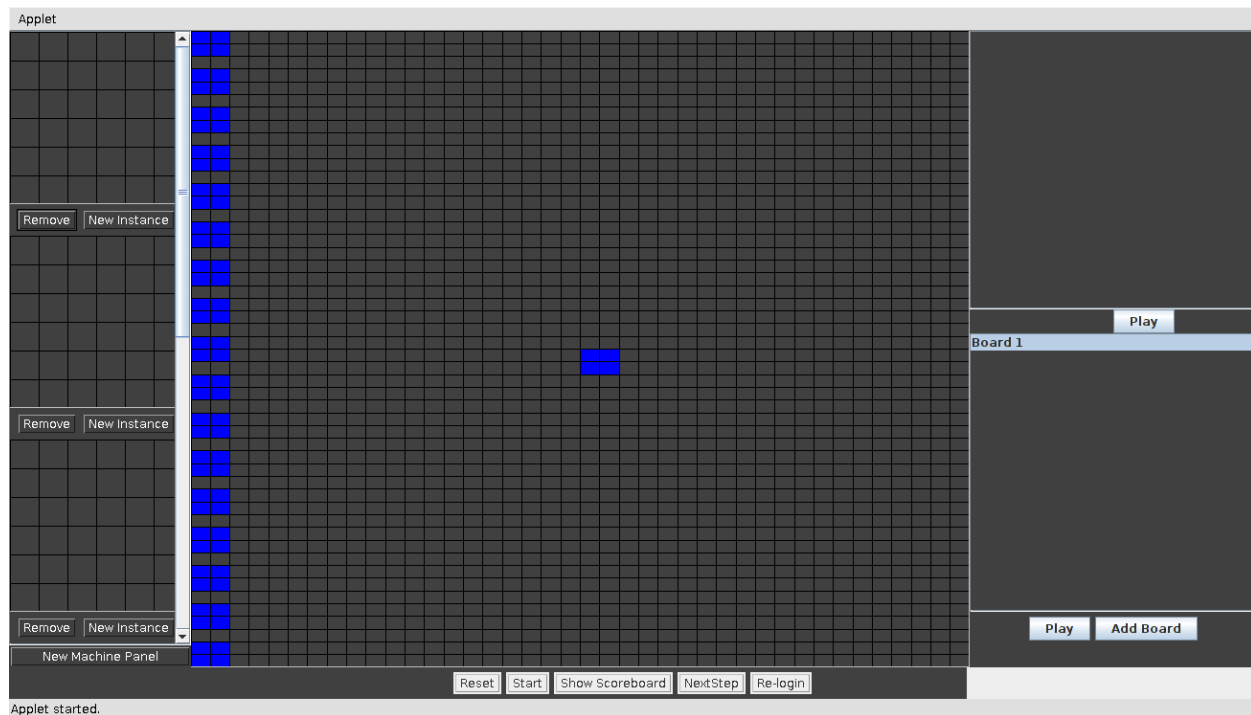
## About the game:

Hello everybody! Welcome to the event ‘INSTIGRID’ where you will pit your brains against your peers on a platform devised by a genius (John Conway) and well studied by mathematicians around the world. Yes, it is the Conway’s Game of Life. Computer scientists call it a cellular automaton. It is essentially an infinite 2D grid of square cells, each of which can have two states (dead/alive, active/inactive, ...) Every cell interacts with its eight neighbors, which are the cells that are horizontally, vertically, or diagonally adjacent. At each step in time, the following transitions occur:

1. Any live cell with fewer than two live neighbours dies, as if caused by under-population.
2. Any live cell with two or three live neighbours lives on to the next generation.
3. Any live cell with more than three live neighbours dies, as if by overcrowding.
4. Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.

The initial pattern constitutes the *seed* of the system. The first generation is created by applying the above rules simultaneously to every cell in the seed—births and deaths occur simultaneously, and the discrete moment at which this happens is sometimes called a *tick* (in other words, each generation is a pure function of the preceding one). The rules continue to be applied repeatedly to create further generations.

## Objective of the game:



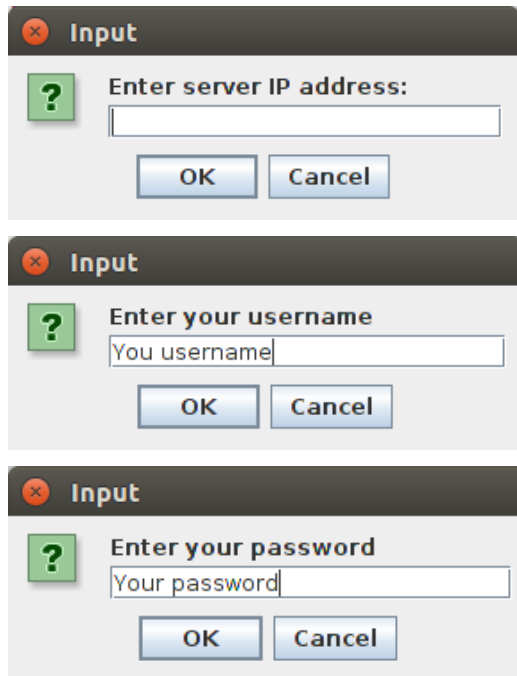
Your initial board contains still-life patterns in the form of 18  $2 \times 2$  square blocks. This complex of 18 square blocks forms your castle.

Now the rules are pretty simple. Here two teams have to play against each other wherein they have to protect their own castle & destroy other team's castle to win. Whichever team first destroys other team's castle wins. Even if one of the  $18 \times 2 \times 2$  square blocks is destroyed, the castle is considered to be destroyed. So hold your fort and besiege your enemy's!

## Gameplay:

### Login:

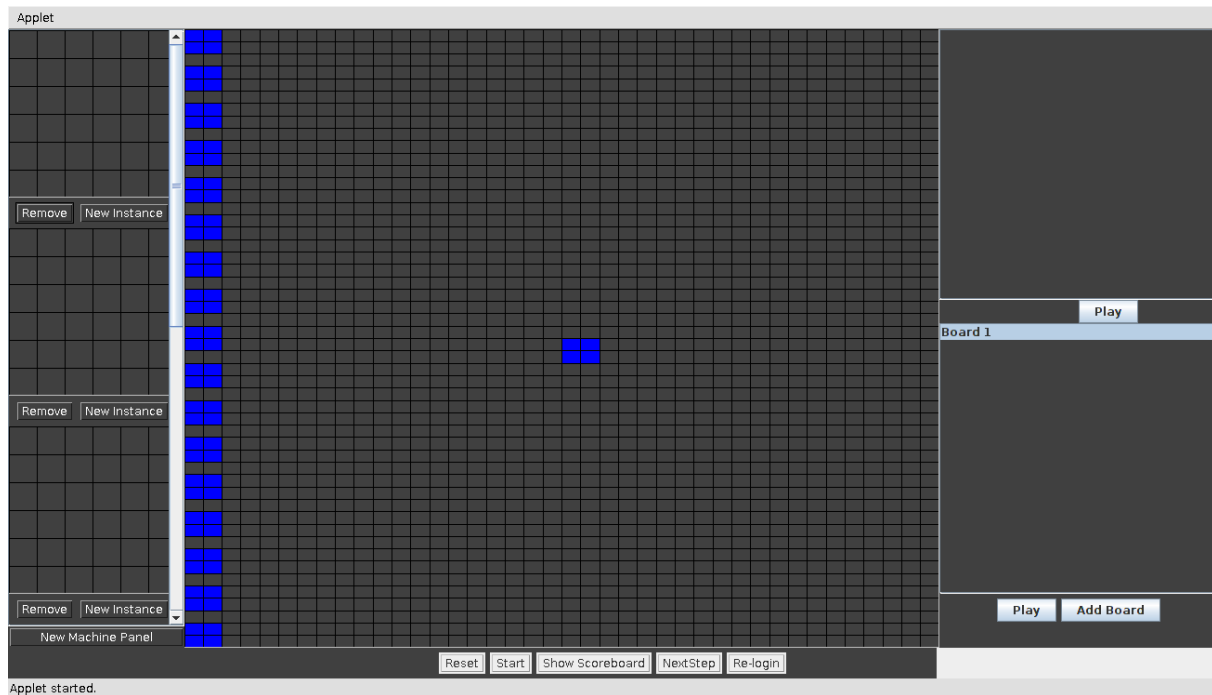
Immediately after the execution of the program, it will prompt for the server IP address, username and password as shown in the figure. Enter the relevant username and password given to you prior to the beginning of the event.



The figure displays three sequential 'Input' dialog boxes, each with a green question mark icon and 'OK'/'Cancel' buttons.

- First dialog:** Titled 'Input', it prompts 'Enter server IP address:' with an empty text field.
- Second dialog:** Titled 'Input', it prompts 'Enter your username' with a text field containing the placeholder text 'You username'.
- Third dialog:** Titled 'Input', it prompts 'Enter your password' with a text field containing the placeholder text 'Your password'.

After you logged in successfully you can see the configuration of the castle in the formally created 'Board1'.



### Add new boards:

If you want to try a different configuration without editing your current board, you have the option of adding new boards in the applet.

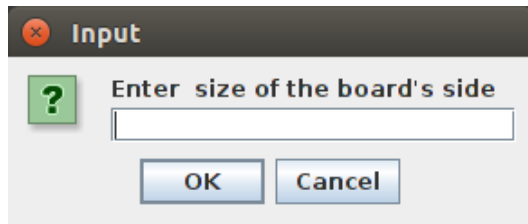
Clicking on the 'Add Board' button creates a new board in the list. Then select the board and click 'Play' for viewing the corresponding Board. The board which you are viewing is the Active board which will be used for your challenge.

### Working with machine library:

It is tedious to have to draw the same machine again and again

In machine library you can create a new instance i.e you can make a machine in the panels and you can use them by clicking on the 'New Instance' button and then again clicking and dragging it on the board you can select the place where you want to place the machine.

If you want to make a machine of your wanted size or if you want to add some more machine to your machine library you can do it by clicking 'New Machine Panel' which will prompt a dialog box as shown in which you have to enter the side of the panel which you want.



**Note:**

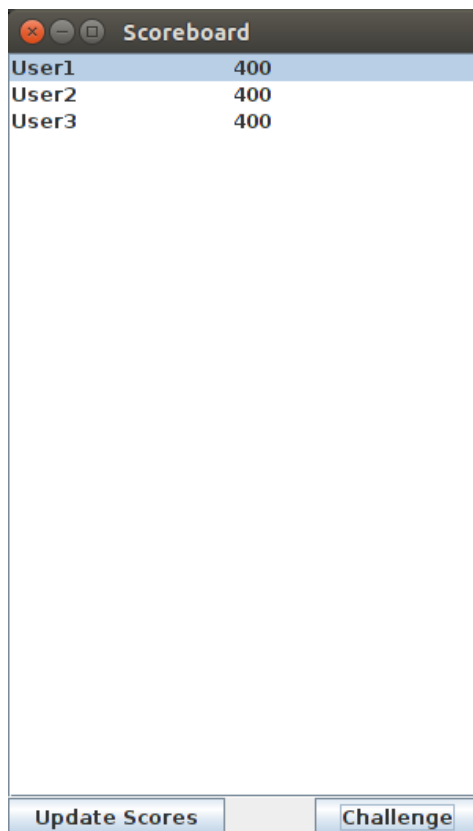
Sometimes, it may happen that the machine panel is elongated. For this, just keep on adding new machine panels which will become normal after two or three additions.

**Warning:**

Don't ever close the tab as you will lose your saved machine library and the saved boards, but the score pertaining to you is the same.

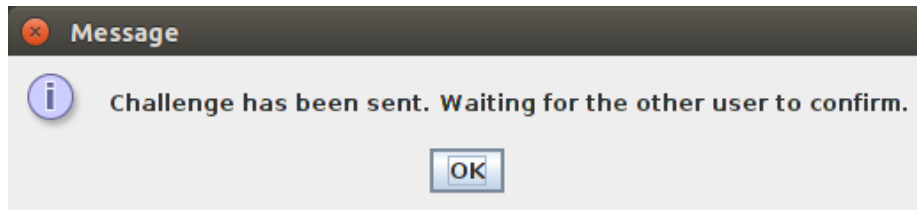
Challenging:

One can challenge the other by clicking on the 'Show scoreboard which prompts up a box as shown

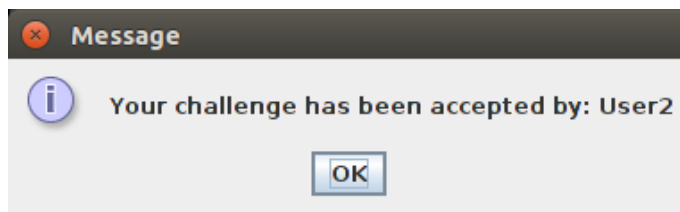


For updating the scores list click on the 'Update scores' button by which you can estimate you position of yours and others.

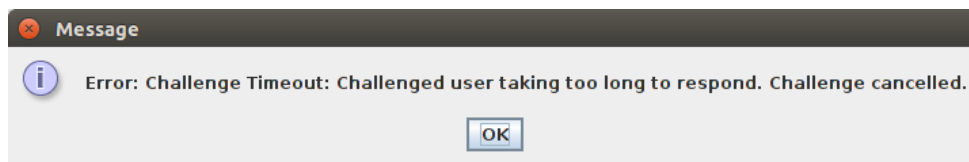
If you want to challenge a team, just select the team and click on 'Challenge' option which will send a request to play with the team which if he accepts the battle begins and outputs the result immediately.



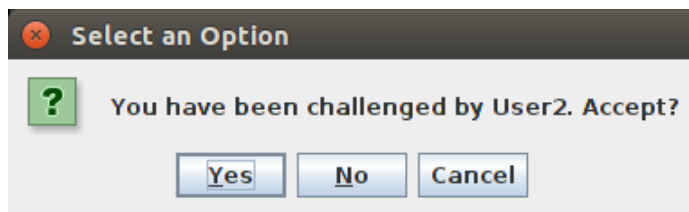
Depending of the challenged person's response you will get a pop up of acceptance or rejection as shown



If the challenged person take too much time(i.e more than 15 seconds) to decide for the challenge an error will pop out saying



If any one challenges you will get a dialog box as shown below and proceed as you wish to do.



### Simulations process:

If you want to view the simulation of your challenge, select the challenge which you have completed and click on 'play'. The board which you have used for the challenge appears and then on clicking start you can see the simulation of the challenge.

If you want to review the challenge from the beginning click and 'Reset' and then start again. You can also view the simulation step by step for analyzing your game by clicking the 'Next step' button .

### Re-login:

In case the connection drops, press the re-login button and enter the credentials again. In such cases, you will generally be prompted for the IP address again. Also note that, if you close the window, all data of the boards will be lost. So please do not close the window.

### Scoring:

Initially your score starts with a value of 0.0. The scoring is such that use of fewer number of active cells is encouraged. For each match we calculate the ratio of the number of cells which are active during the start of simulation. This ratio is added or subtracted to the score after the match for the winner and loser respectively. If the person with the fewer (relatively) active cells wins, the  $\max\{y/x, x/y\}$  is taken and (otherwise) if the winner uses greater (relatively) number of cells the  $\min\{y/x, x/y\}$  is taken .

Here's something interesting and useful for innovator. If a team makes innovative machines or creates ways to use them they will be awarded 2 percent extra of their final score for each innovative idea upto 5 ideas.

**Note:** Here, the  $x, y$  denote the number of active cells used by each user.


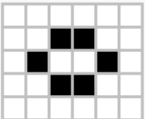
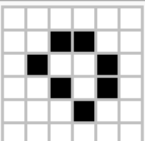
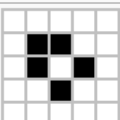
The judgement made by the judge(s) is final.

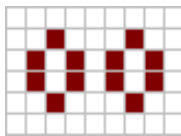
### Machinery:-

With the above set of rules we can form various machines (or patterns) which behave in different ways. Like:-

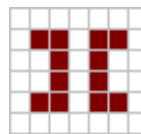
- 1) **Still-life machines**- are patterns which don't change their shape in the next generation.

Eg. the 2\*2 square block & beehive.....

Still lifes	
Block	
Beehive	
Loaf	
Boat	



A pseudo still life in Conway's game of life.  
Removing one of the islands will not affect the stability of the other.

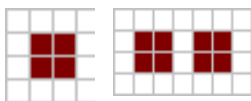


A strict still life in Conway's game of life.  
Each island depends on the other for stability.

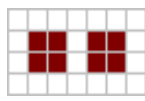
## Common examples:-

### Blocks

The most common still life (i.e. that most likely to be generated from a random initial state) is the **block**. A pair of blocks placed side-by-side (or **bi-block**) is the simplest pseudo still life. Blocks are used as components in many complex devices.



Block

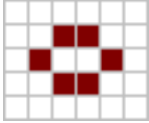


Bi-block



## Hives

The second most common still life is the **hive** (or **beehive**).<sup>[3]</sup> Hives are frequently created in (non-interacting) sets of four, in a formation known as a **honey farm**.



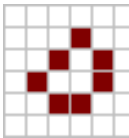
Hive



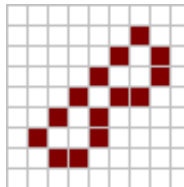
Honey farm

## Loaves

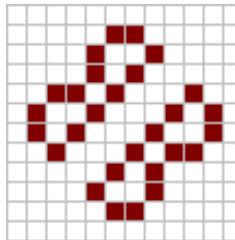
The third most common still life is the **loaf**.<sup>[3]</sup> Loaves are often found together in a pairing known as a **bi-loaf**. Bi-loaves themselves are often created in a further (non-interacting) pairing known as a **bakery**.



Loaf



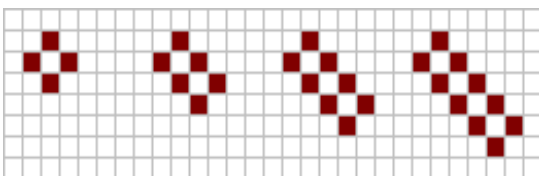
Bi-loaf



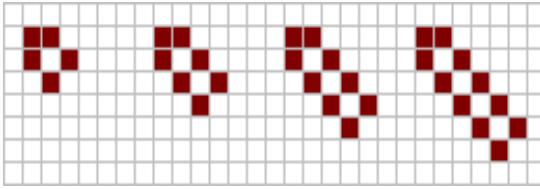
Bakery

## Tubs, barges, boats and ships

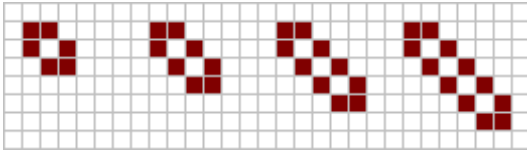
A **tub** consists of four live cells placed in a diamond shape around a central dead cell. Placing an extra live cell diagonally to the central cell gives another still life, known as a **boat**. Placing a further live cell on the opposite side gives yet another still life, known as a **ship**. A tub, a boat or a ship can be extended by adding a pair of live cells, to give a **barge**, a **long-boat** or a **long-ship** respectively. This extension can be repeated indefinitely, to give arbitrarily large structures.



From left: tub, barge, long-barge, etc...

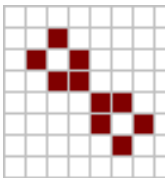


From left: boat, long-boat, etc...

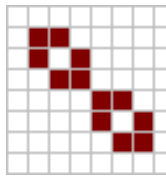


From left: ship, long-ship, etc...

A pair of boats can be combined to give another still life known as the **boat tie** (a pun on bow tie, which it superficially resembles). Similarly, a pair of ships can be combined into a **ship tie**.

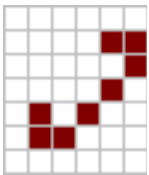


Boat tie

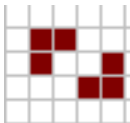


Ship tie

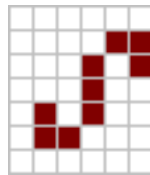
## Miscellaneous



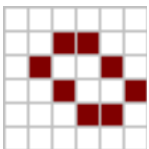
Canoe /  
Sinking  
ship



Carrier /  
Aircraft  
Carrier



Integral  
sign



Mango /  
Cigar



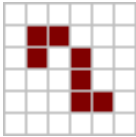
Pond



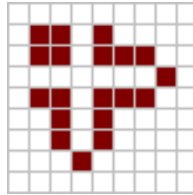
Snake

## Eaters and reflectors

Still lifes can be used to modify or destroy other objects. An **eater** is capable of absorbing a spaceship and returning to its original state after the collision. Many examples exist, with the most notable being the **fish-hook**, which is capable of absorbing several types of spaceship. A similar device is the **reflector**, which alters the direction of an incoming spaceship. Eaters and reflectors are not necessarily still lifes, as the term may also apply to oscillators with similar properties.



The fish-hook, capable of eating a variety of objects.



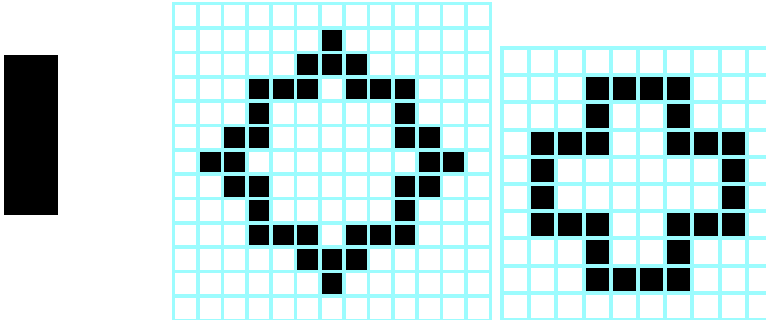
The eater 2, capable of absorbing gliders entering from the upper left, and spaceships from the left

**Oscillators**- these machines return back to their original state after fixed number of generations. Eg. blinker & toad.

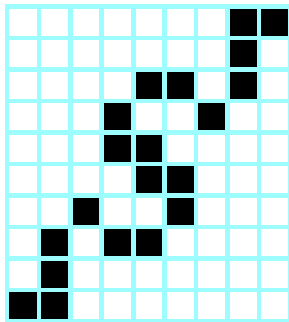
Oscillators	
Blinker (period 2)	A 5x5 grid showing the blinker pattern, a simple oscillator with a period of 2 generations.
Toad (period 2)	A 5x5 grid showing the toad pattern, a simple oscillator with a period of 2 generations.
Beacon (period 2)	A 5x5 grid showing the beacon pattern, a simple oscillator with a period of 2 generations.
Pulsar (period 3)	A 10x10 grid showing the pulsar pattern, a complex oscillator with a period of 3 generations.

The smallest number of generations it takes before the pattern returns to its initial condition is called the *period* of the oscillator. An oscillator with a period of 1 is usually called a still life, as such a pattern never changes. Sometimes, still lifes are not taken to be oscillators. Another common stipulation is that an oscillator must be finite.

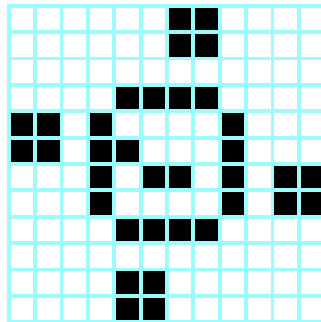
### Examples



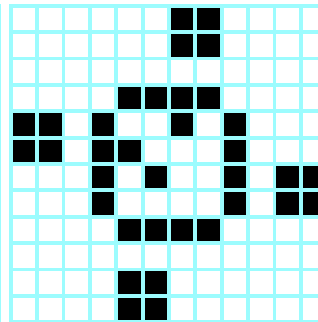
blinker, period 2



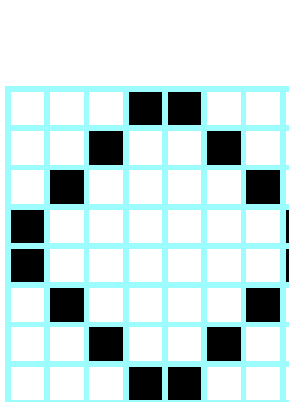
star, period 3



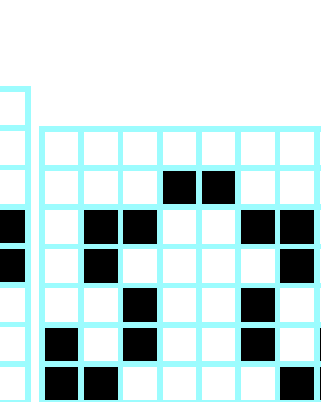
cross, period 3



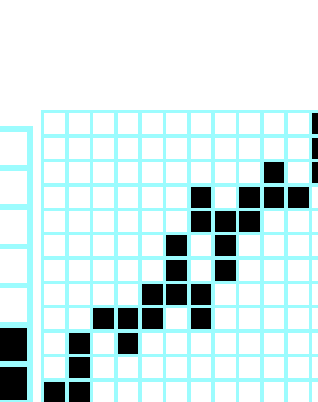
French kiss, period 3



clock 2, period 4



pinwheel, period 4



octagon, period 5

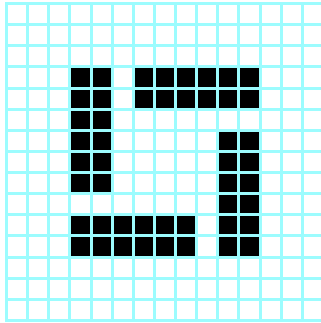


fumarole, period 5

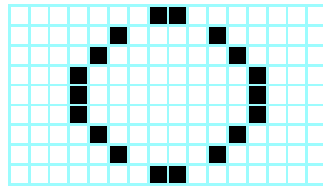


pentoad, period 5



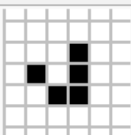
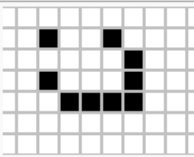


Kok's galaxy, period 8

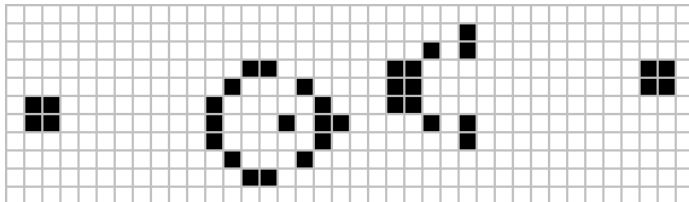


pentadecathlon, period 15

**Spaceship** - these patterns translate across the board in particular direction. Eg. Lightweight spaceship which travels in horizontal direction.. There is another simple spaceship which travels in the diagonal direction. Try to create it on your own.

Spaceships	
Glider	
Lightweight spaceship (LWSS)	

**Glider gun:-**



**Note:-** There are many other types of machines performing various types of interesting functions like some explode into smaller machines, some self destruct etc. Try to build them by trial & error on your boards.