I wrote a program to play Conway's Game of Life. The game is a cellular automaton in which the starting position is a grid of cells, with some cells alive and others dead. A living cell survives to the next stage iff it has exactly 2 or 3 living neighbours, and a dead cell comes alive iff it has exactly 3 living neighbours.

Below are the functions I wrote to determine whether a cell lives or dies in the next stage. Here I am using a 38x38 grid, with a "permanently dead" border around the outside.

```
cellneighbours[array_, row_, column_] :=
 If[(row ≠ 1 ∧ row ≠ 40 ∧ column ≠ 1 ∧ column ≠ 40),
  {array[[row - 1]][[column - 1]], array[[row - 1]][[column]],
   array[[row - 1]][[column + 1]], array[[row]][[column - 1]],
   array[[row]][[column + 1]], array[[row + 1]][[column - 1]],
   array[[row + 1]][[column]], array[[row + 1]][[column + 1]]}, {}]

countlivenbrs[array_, row_, column_] := Count[cellneighbours[array, row, column], 1]

cellstep[array_, row_, column_] :=
 If[(((array[[row]][[column]] == 0 ∧ countlivenbrs[array, row, column] == 3) ∨
     (array[[row]][[column]] == 1 ∧ countlivenbrs[array, row, column] == 3) ∨
     (array[[row]][[column]] == 1 ∧ countlivenbrs[array, row, column] == 2)),
  1,
  0]
```

The function below, given an array, returns the array that occurs one stage later.

```
arraystep[array_] := Table[Table[cellstep[array, i, j], {j, 40}], {i, 40}]
```

The main function, below, uses the DynamicModule function, and iterates the process up to 1000 stages. You can change the 0.2 to any number you like to pause for that many seconds between each step.

```
gameoflife[array_] :=
 DynamicModule[{x = array}, For[k = 1, k < 1000, k++, x = arraystep[x];
   test = x;
   Pause[0.2]]]
```

Here are some fun examples of oscillators. To run one of them, set the variable "test" equal to it, and execute the "gameoflife" function below. If you execute this notebook as is, the Clock will run in the Dynamic window below.

The Glider moves across the screen and then forms a stable square when it hits an edge.

```
ReplacePart[ConstantArray[0, {40, 40}],
  {{10, 9} → 1, {10, 10} → 1, {10, 11} → 1, {9, 11} → 1, {8, 10} → 1}];
```

The Clock simply oscillates back and forth with a period of 2.

```
test = ReplacePart[ConstantArray[0, {40, 40}],
    {{20, 19} → 1, {21, 20} → 1, {22, 20} → 1, {20, 21} → 1, {19, 21} → 1, {21, 22} → 1}];
```

The Sideways Tumbler.

```
ReplacePart[ConstantArray[0, {40, 40}],
  {{16, 20} → 1, {16, 19} → 1, {17, 18} → 1,
   {18, 19} → 1, {19, 20} → 1, {18, 21} → 1, {18, 22} → 1, {19, 22} → 1,
   {24, 20} → 1, {24, 19} → 1, {23, 18} → 1, {22, 19} → 1,
   {21, 20} → 1, {22, 21} → 1, {22, 22} → 1, {21, 22} → 1}];
```

If we start with a 10-cell row, it evolves after two steps into an oscillator. It will continue to oscillate from there, with period 15, but it will never revert to the original 10-cell row.

```
ReplacePart[ConstantArray[0, {40, 40}],
  {{20, 16} → 1, {20, 17} → 1, {20, 18} → 1, {20, 19} → 1, {20, 20} → 1,
   {20, 21} → 1, {20, 22} → 1, {20, 23} → 1, {20, 24} → 1, {20, 25} → 1}];
```

The Pulsar, although large, only has a period of 3.

```
ReplacePart[ConstantArray[0, {40, 40}],
  {{14, 16} → 1, {14, 17} → 1, {14, 18} → 1, {14, 22} → 1, {14, 23} → 1, {14, 24} → 1,
   {16, 14} → 1, {16, 19} → 1, {16, 21} → 1, {16, 26} → 1,
   {17, 14} → 1, {17, 19} → 1, {17, 21} → 1, {17, 26} → 1,
   {18, 14} → 1, {18, 19} → 1, {18, 21} → 1, {18, 26} → 1,
   {19, 16} → 1, {19, 17} → 1, {19, 18} → 1, {19, 22} → 1, {19, 23} → 1, {19, 24} → 1,
   {21, 16} → 1, {21, 17} → 1, {21, 18} → 1, {21, 22} → 1, {21, 23} → 1, {21, 24} → 1,
   {22, 14} → 1, {22, 19} → 1, {22, 21} → 1, {22, 26} → 1,
   {23, 14} → 1, {23, 19} → 1, {23, 21} → 1, {23, 26} → 1,
   {24, 14} → 1, {24, 19} → 1, {24, 21} → 1, {24, 26} → 1,
   {26, 16} → 1, {26, 17} → 1, {26, 18} → 1, {26, 22} → 1, {26, 23} → 1, {26, 24} → 1
  }];
```

You can also create your own starting pattern by writing it as an array and setting “test” equal to it! When you’re ready, you can execute the function below...

```
gameoflife[test];
```

...and watch the game run in the Dynamic window below!

```
Dynamic[ArrayPlot[test, Mesh → True]]
```

ArrayPlot[test, Mesh → True]