# *Conway's Game of Life Project*

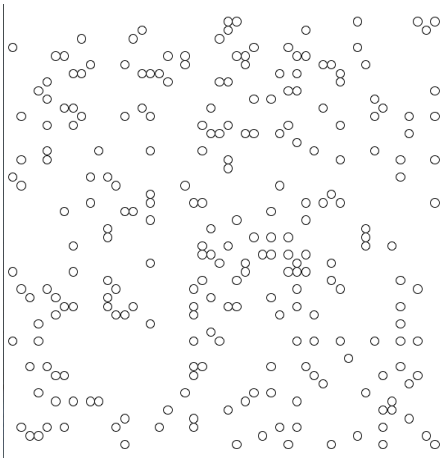
*Chiu Zi Xin*

*November 5, 2015*

Conway's Game of Life is a game play by nobody. It is work by few rules and it will evolves from the initial configuration, we observe the patterns and how it evolves.


Figure 1: There are many names of this game...

## IMAGES



This is the first loop, lives generate as the percentage we want.

This is the third loop, patterns start to form, such as: "still life", "oscillators" or "spaceships"

## THE PYTHON CODE

```
import random
from graphics import *
def empty(N):
    a=[]
    for i in range(N):
        b=[]
        for j in range(N):
            b=b+[0]
        a=a+[b]
    return a
def fill(a,p):
    N=len(a)
    for i in range(N):
        for j in range(N):
            if random.uniform(0,1)<p:
                a[i][j]=1
def update(A,B):
    N=len(A)
    for i in range(N):
        for j in range(N):
            neigh=A[(i-1)%N][(j-1)%N]+
                   A[(i-1)%N][(j)]+
                  A[(i-1)%N][(j+1)%N]+
                   A[i][(j-1)%N]+
                   A[i][(j+1)%N]+
                   A[(i+1)%N][(j-1)%N]+
                  A[(i+1)%N][(j)]+
                  A[(i+1)%N][(j+1)%N]
            if A[i][j]==0:
                if neigh==3:
                    B[i][j]=1
                else:
                    B[i][j]=0
            else:
                if neigh==2 or neigh==3:
                    B[i][j]=1
                else:
                    B[i][j]=0
```

This function creates an NxN array filled with zeros.

This function fills the array a with a portion p of live .

This function updates the array A by the rule and generates array B.

```
def gen2Dgraphic(N):
    a=[]
    for i in range(N):
        b=[]
        for j in range(N):
            b=b+[Circle(Point(i,j),.49)]
        a=a+[b]
    return a
def push(B,A):
    N=len(A)
    for i in range(N):
        for j in range(N):
            A[i][j]=B[i][j]
def slider(a,x,y):
    a[0+x][0+y]=1
    a[0+x][1+y]=1
    a[2+x][1+y]=1
    a[0+x][2+y]=1
    a[1+x][2+y]=1
def slider2(a,x,y):
    a[0+x][0+y]=1
    a[1+x][0+y]=1
    a[2+x][0+y]=1
    a[0+x][1+y]=1
    a[1+x][2+y]=1
def drawArray(A,a,window):
#A is the array of 0,1 values representing the state of
the game
#a is an array of Circle objects
#window is the GraphWin in which we will draw the circles
    N=len(A)
    for i in range(N):
        for j in range(N):
            if A[i][j]==1:
                a[i][j].undraw()
                a[i][j].draw(window)
            if A[i][j]==0:
                a[i][j].undraw()
```

This function is the config of the circles.

This function pushes the array B back to array A to prepare for the next update.

These two functions create a pattern that will slide to one direction.

This function will draw the living cells on the window.

```
N=50
win = GraphWin("Title",600,600)
win.setCoords(-1,-1,N+1,N+1)
grid=empty(N)
grid2=empty(N)
circles=gen2Dgraphic(N)
#fill(grid,0.1)
#for i in range(10):
    #slider(grid,5*i,5*i)
    #slider2(grid,5*i,5*1)
while True:
    drawArray(grid,circles,win)
    update(grid,grid2)
    push(grid2,grid)
```

Things here are the values of the variable.

## PROSE [1]

[1]

The Game of Life, also known simply as Life, is a cellular automaton devised by the British mathematician John Horton Conway in 1970.
The rule of the game of life is:

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 over-population.

4. Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.

A cellular automaton is a discrete model studied in computability theory, mathematics, physics, complexity science, theoretical biology and microstructure modeling. Cellular automata are also called cellular spaces, tessellation automata, homogeneous structures, cellular structures, tessellation structures, and iterative arrays.
Wolfram, in A New Kind of Science and several papers dating from the mid-1980s, defined four classes into which cellular automata and several other simple computational models can be divided depending on their behavior.

Class 1: Nearly all initial patterns evolve quickly into a stable, homogeneous state. Any randomness in the initial pattern disappears.

Class 2: Nearly all initial patterns evolve quickly into stable or oscillating structures. Some of the randomness in the initial pattern may filter out, but some remains. Local changes to the initial pattern tend to remain local.

Class 3: Nearly all initial patterns evolve in a pseudo-random or chaotic manner. Any stable structures that appear are quickly destroyed by the surrounding noise. Local changes to the initial pattern tend to spread indefinitely.

Class 4: Nearly all initial patterns evolve into structures that interact in complex and interesting ways, with the formation of local structures that are able to survive for long periods of time.[33] Class 2 type stable or oscillating structures may be the eventual outcome, but the number of steps required to reach this state may be very large, even when the initial pattern is relatively simple. Local changes to the initial pattern may spread indefinitely. Wolfram has conjectured that many, if not all class 4 cellular automata are capable of universal computation. This has been proven for Rule 110 and Conway's game of Life.

## REFERENCES

[1] 11 2015. URL https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life#Origins.