Conway's Game of Life

Ilya Michurin

November 12, 2015

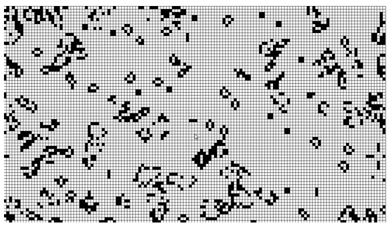
Figure 1: This is an example of the Game of Life

Main Menu - Program Code

1) What is Conway's Game of Life

Conway's Game of Life, also known simply as Life, is a cellular automaton devised by the British mathematician John Horton Conway in 1970.

The "game" is a zero-player game, meaning that its evolution is determined by its initial state, requiring no further input. One interacts with the Game of Life by creating an initial configuration and observing how it evolves or, for advanced players, by creating patterns with particular properties.

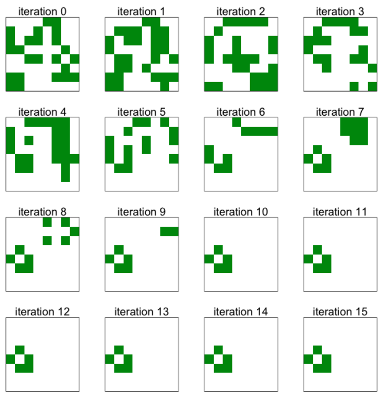

Figure 2: In this picture 12 section are presented of the Game of Life, highlighted squares are the cells that live.

2) 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
```

After reading a code and copying it into your PYTHON, please BE CAREFUL!!!! The reason is the grid number, put the most lucky number for your laptop/PC in the way not to burn it. XD

```
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


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):
```

The number that you put will be your grid size, if you put 6 or 5, your grid will be 6x6 or 5x5, so pick the good one

It all depends on a size, if you do not know settings of your computer, it will take more time to process the whole code, don't even try to enjoy the time of processing the grid of 20x20

```
            A[i][j]=B[i][j]

def drawArray(A,a,window):
    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()

N=int(input("Enter the grid dimension  "))
win = GraphWin()
win.setCoords(-1,-1,N+1,N+1)
grid=empty(N)
grid2=empty(N)
circles=gen2Dgraphic(N)
fill(grid,0.3)

while True:
    drawArray(grid,circles,win)
    update(grid,grid2)
    push(grid2,grid)
```

3) Rules of the Conway's Game of Life

The universe of the Game of Life is an infinite two-dimensional orthogonal grid of square cells, each of which is in one of two possible states, alive or dead. Every cell interacts with its eight neighbours, 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 two or three live neighbours lives on to the next generation.

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

3)In any other situation cell dies.

We showed you our version of work that does not work properly, but it works. There is one problem that we could not how to solve the problem with putting changes of the new state in one window

So, by the creating a first erray, we crate the "Old State" or "original" one where we put cell in random positions, after the creating, by the graph library, all lines and circles=cells, we started to create a new array, which is also random, and that will follow all the rules of Game of Life, after by using again graphics library, we get a ready "New State" of cells which applied all the rules.]4) How our program does it's job

So, by the creating a first erray, we crate the "Old State" or "original" one where we put cell in random positions, after the creating, by the graph library, all lines and circles=cells, we started to create a new array, which is also random, and that will follow all the rules of Game of Life, after by using again graphics library, we get a ready "New State" of cells which applied all the rules.

```
As the result, we got

continuous creation of "New States" which you can stop
only by killing the program

Because we couldn't put changing of live cells
in one state
```