THE GMAE OF LIFE

Yijun Chen

November 16, 2015

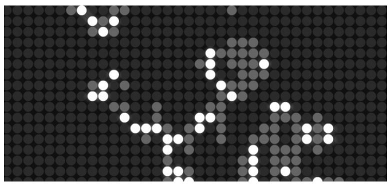
Figure 1: An example of a random state of game of life

INTRODUCTION

The game of life was first devised by a British mathematician John Horton Conway in 1970. It's a cellular automaton. The universe of this game is an infinite two-dimensional grid. No players are needed in this "game", which means its evolution is determined by its initial state.

Rules

[1] In most case, there are two colors on the grid, because we need to distinguish the live cells and the dead cells. Usually we use black to represent the life cells, white for the dead cells. There are three rules in the game of life, and are all pretty simple. The evolution of the grid has to follow the four rules below.

[1]

- When a live cell has fewer than two live neighbors, it dies in the next state(think of the case of underpopulation).
- When a live cell has exactly two or three neighbors, it keeps alive in the next state.
- When a dead cell has exactly three live neighbors, it becomes alive in the next state.
- When a live has more than three neighbors, it dies in the next state(think of the case of overpopulation).

Births and deaths happen simultaneously. The game will continue to apply these rules to create new states until it cannot be changed any further.

Now, I will present a program that can generate a game of life in the "life-form" of "Gliders".

GoL Program

In this program, all live cells are represented by zeros, and they are filled in an NxN grid.

```
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



def gen2Dgraphic(N):
    a=[]
```

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

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

```

#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()

def glider(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 glider2(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[0+x][0+y]=1
```

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

```
def glider3(a,x,y):
    a[1+x][0+y]=1
    a[1+x][2+y]=1
    a[2+x][2+y]=1
    a[0+x][2+y]=1
    a[0+x][1+y]=1

N=50
win = GraphWin()
win.setCoords(-1,-1,N+1,N+1)
grid=empty(N)
grid2=empty(N)
circles=gen2Dgraphic(N)
#fill(grid,0.5)
for i in range(10):
    glider(grid,4*i,5*i)

for i in range(200):
    drawArray(grid,circles,win)
    update(grid,grid2)
    push(grid2,grid)
```

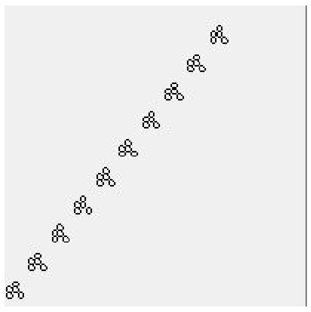
Figure 2: The final state of my game.You can see the gliders are aligned diagonally, marching towards the top right corner.

References

[1] Melissa Gymrek: Conway's Game of Life
`http://web.mit.edu/sp.268/www/2010/lifeSlides.pdf`