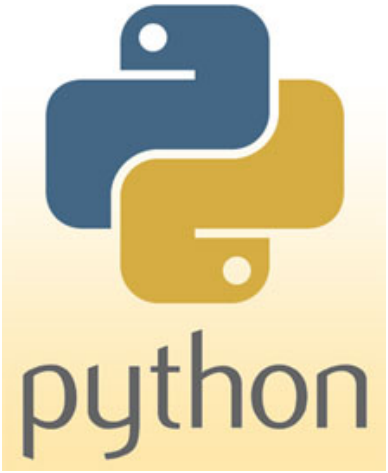
Python: Game of Life

Jan Kamburg

November 8, 2015



The following project is about Conway's Game of Life. It is a simulation which simulates the life of cells based on rules:

Rules for cells which are alive

- Cells with less than two alive neighbours die
- Cells with more than three alive neighbours die
- Cells with two or three alive neighbours stay alive

Rules for cells which are dead

- Cells with three alive neighbours become alive

Because the code is very complex i will explain it in parts.

Structure of the program

The program uses a class for the main part of the program. The program itself gets initialized outside of the class. The class itself has many functions which are there for keeping the simulation running. All of the following functions are inside the MainWindow class. To make it easier to explain i am going to rip it apart.

```
from graphics import *
import random

class MainWindow:

    def __init__(self,W,H):
        Win = GraphWin("Game Of Life", W*7, H*7)
        Win.setCoords(0,0,W+1,H+3)
        self.width = W
        self.height = H
        self.window = Win
        self.colors = [color_rgb(0,0,0),color_rgb(0,255,0)]
        self.debug = color_rgb(0,0,255)
        self.Rounds = 0
        Win.getMouse()
```

First the script has to get started by importing the graphics library and the random library. Then the class is initialized along with some main values.

```
def drawGrid(self,val):
        W = self.width
        H = self.height
        self.GridArray = []
        self.OldArray = []
        self.cellsAlive = 1
        for i in range(W):
            a = []
            for x in range(H):
                life = random.randint(0,val)
                if life == 1:
                    Rect = Rectangle(Point(x+1,i+1),
                                     Point(x+2,i))
                    Rect.setFill(self.colors[1])
                    Rect.draw(self.window)
                    a.append([i,x,1,Rect])
                else:
                    Rect = Rectangle(Point(x+1,i+1),
                                     Point(x+2,i))
                    Rect.setFill(self.colors[0])
                    Rect.draw(self.window)
                    a.append([i,x,0,Rect])

            self.GridArray.append(a)
        self.T1 = Text(Point(self.width/2,self.height+1),
        "Simulations rounds run: " + str(self.Rounds) +
        " | Cells alive: " + str(self.cellsAlive))
        self.T1.draw(self.window)
```

First a few variables get initialised.

Then then a nested for loop is run. The ranges used for the loops are the witdh and the height of the cell colony.

Then a random value is created. The variable val is pased by the function and determins how many cells should be alive when we first start the programm. When the cell is initialized it is drawn. Then i am populating the the arrays. Every cell has its own array. It contains the position, the status and the graphics library object of the cell. Finally a text field is drawn at the top which is saying how many simlulation rounds were already run and how many cells are alive.

Before i can start explaining the update function i first have to start with the initializing update function.

```
def isAlive(self):
        CountA = 0
        for i in range(len(self.GridArray)):
            for x in range(len(self.GridArray[i])):
                CountA = CountA+self.GridArray[i][x][2]

        self.cellsAlive = CountA

    def startSim(self):
        self.Rounds = 0
        while self.cellsAlive > 0:
            self.T1.setText("Simulations rounds run: " +
str(self.Rounds) + " | Cells alive: "
+ str(self.cellsAlive))
            self.isAlive()
            self.OldArray = self.GridArray
            self.Rounds = self.Rounds + 1
            for i in range(len(self.GridArray)):
                self.check(self.GridArray[i],
self.OldArray[i])
```

The isAlive() function updates the self.cellsAlive variable. This is used to run the programm without errors. Finally the startSim() function starts the update function which i will explain later. The while loop runs as long as cells are living. The self.GridArray consists of multiple arrays. Its length is the height of the colony. And then for each of the elements in self.GridArray. But before that i am making a copy of self.GridArray for comaprism and i am naming it self.OldArray. The function self.check() is the update function.

```
def check(self,a,old):
        for i in range(len(a)):
            Count = 0
            Obj = a[i]
            VRow = a[i][0]
            HRow = a[i][1]
            #print VRow,HRow
            for w in range(-1,2):
                for h in range(-1,2):
                    if not VRow+w < 0 and not VRow+w >
self.height-1 and not HRow+h < 0 and not
HRow+h > self.width-1:
                        C = self.OldArray[VRow+w][HRow+h]
                        if Obj[3] != C[3]:
                            if C[2] == 1:
                                Count = Count+1


            Alive = old[i][2]
            if Alive == 1:
                if Count < 2 or Count > 3:
                    Obj[3].setFill(self.colors[0])
                    Obj[2] = 0
                else:
                    if Count == 2 or Count == 3:
                        Obj[3].setFill(self.colors[1])
                        Obj[2] = 1

            if Alive == 0:
                if Count == 3:
                    Obj[3].setFill(self.colors[1])
                    Obj[2] = 1
```

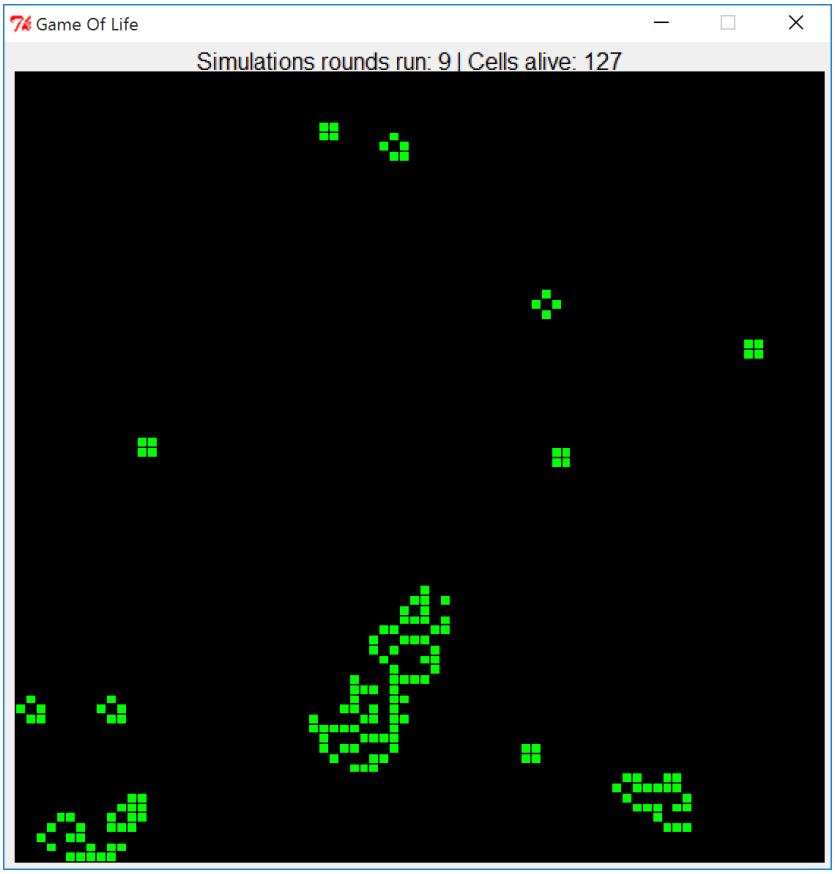
The check function is the most important function in this programm. It checks every cell and changes its status. Because in the startSim() function we pass an array we first need to get every element inside of this array. That we are doing with a for loop. Because we want to count the neighbour cells we set a count value to 0 and the object cell we are cheking is the i element of a. HRow and VRow are the positions of the cell. Then i two for loops form -1 to 2 the programm is checking the sourounding cells. If the calculated coordiates are not outside of our colony and the cell is not our cell we are checking from we are adding its life status to the Count value. After all sourrounding cells are checked we defining the state of the cell according to our rules mentioned at the beginning of this document.

Now that the code in the class is finished i am going to explain the part outside of the class.

```
Start = input("Please state start value")
Window = MainWindow(80,80)
Window.drawGrid(Start)
Window.startSim()
```

The start value is a number which ou can define. It will say how many percent of the cells will be alive at the beginning. The Window is the MainWindow we are creating. The numbers we are passing in are is the size of the colony. With Window.drawGrid(Start) we are calling the drawGrid() function inside the class. And right after that we are starting the simulation.

THE FINAL OUTCOME



SOURCES

"Conway's Game of Life." Wikipedia. Wikimedia Foundation, n.d. Web. 08 Nov. 2015.