Simulación Juego de la Vida

Alumno: Monroy Martos Elioth
Profesor: Genaro Juárez Martínez
Materia: Computing Selected Topics
Grupo: 3CM8

19 de septiembre de 2018

Índice

1. Introducción

El juego de la vida fue desarrollado por John Horton Conway, quien fuera un matemático estadounidense que trabajaba en la Universidad de Cambridge. Él desarrollo un "juego" al cual llamaba vida, debido a su parecido con la forma en que las sociedades de organismos vivos se levantan y caen.[1]

Este juego se considera como un simulador, ya que se asemeja a la vida real. Originalmente, se planteó como un juego de mesa, pero con el pasar de los años fue usado en otras ramas (como la computación) debido a las posibilidades que este juego brinda.

La idea básica del juego, es iniciar con una configuración simple de organismos vivientes, cada uno asignado a una celda dentro de un tablero (el cual se considera un plano infinito), para así observar como está cambia según se aplican las leyes genéticas de Conway, las cuales determinan el nacimiento, muerte o supervivencia de cada organismo. Estás reglas son tres:

1. Supervivencia: Cada organismo con dos o tres vecinos vivos sobrevivirá a la siguiente generación.

2. Muerte: Cada organismo con cuatro o más vecinos muere por sobrepoblación, así mismo cada organismo con solo un vecino o ninguno morirá por aislamiento.

3. Nacimiento: En cada celda vacía que este rodeada por exactamente tres vecinos, nacerá un organismo. [1]

Es importante señalar que cada muerte, nacimiento o supervivencia debe ser simultaneo durante cada salto de generación. Para realizar la evaluación de cada una de las celdas, estás son divididas a su vez en grupos de 9 celdas, la célula que se evaluará constituye el centro del ahora cuadrado. Dentro de este cuadrado, son aplicadas las reglas ya descritas anteriormente.

El programa que ha sido desarrollado para está actividad, simula este juego. Son dados como parámetros el total de la población y la probabilidad de que existan organismos vivos en la misma. Y con base a esto se realizada la simulación del juego de la vida aplicando las reglas de Conway. Además se gráfica el histórico de la cantidad de organismos vivos que han existido durante cada una de las generaciones.

2. Desarrollo

La actividad fue dividida en dos secciones, una encargada de realizar la simulación del juego de la vida y otra encargada de realizar la gráfica histórica de organismos vivos. Ambos programas fueron elaborados en Python 3.7, usando la librería tkinter y matlibplot respectivamente.

Simulación del juego de la vida El siguiente archivo "gol.py" muestra la implementación realizada del juego de la vida, los parámetros que recibe este programa es el tamaño del tablero y la probabilidad de organismos vivientes en el mismo. Ambos parámetros son enviados mediante la terminal como números enteros.

Una vez ejecutado el programa, al usuario se le mostrá en pantalla el tablero con los organismos vivientes y las celdas vacías, así mismo se le permitirá configurar el color para los mismos. Además de tener la posibilidad de poder ingresar las reglas de supervivencia, muerte y nacimiento.

```
from tkinter import *
from numpy import random
from time import sleep
from sys import argv
from tkinter.colorchooser import *

reglas=[2,3,3,3]
tam=int(argv[2])

master = Tk()
master.title("Game of Life")

numero_vivos=0
numero_muertos=0
generacion=0

color_vivo="#000fff000"
color_muerto="#ff0000"

continuar=False

canvas = Canvas(master, width=1280, height=800, scrollregion
    =(0,0,1500,1200))
label_vivos=Label(master, font='Helvetica 10')
label_muertos=Label(master, font='Helvetica 10')
label_generacion=Label(master, font='Helvetica 10')
```

```
input_reglas=Entry(master)
input_reglas.insert(8,"2,3,3,3")

# hbar=Scrollbar(master,orient=VERTICAL)
# hbar.pack(side=RIGHT,fill=Y)
# hbar.config(command=canvas.yview)

def pausa():
  global continuar
  continuar=not continuar
  if continuar:
    boton_pausa.configure(text="Pausar")
  else:
    boton_pausa.configure(text="Continuar")

def actualizarLabels():
  label_vivos.configure(text="Número de vivos: "+str(
   numero_vivos))
  label_muertos.configure(text="Número de muertos: "+str(
   numero_muertos))
  label_generacion.configure(text="Generación: "+str(generacion)
   )

def cambiar(event):
  print("Me estas presionando")
  obj=event.widget.find_closest(event.x, event.y)
  tag=int(canvas.gettags(obj)[0])
  global numero_vivos
  global numero_muertos
  print(tag)
  if tag==1:#Si esta vivo entonces lo matamos
    canvas.itemconfig(obj, fill=color_muerto,tags='0')
    print("Aumentaron muertos")
    numero_vivos-=1
    numero_muertos+=1
    actualizarLabels()
  else:#Si esta muerto entonces vive
    canvas.itemconfig(obj, fill=color_vivo,tags='1')
    print("Aumentaron vivos")
    numero_vivos+=1
    numero_muertos-=1
    actualizarLabels()

def cambiar_color_vivo():
  color=askcolor()
```

```
68    global color_vivo
69    color_vivo=color[1]
70    label_color_vivos.configure(bg=color_vivo)
71    actualizarColor()
72  def cambiar_color_muerto():
73    color=askcolor()
74    global color_muerto
75    color_muerto=color[1]
76    label_color_muertos.configure(bg=color_muerto)
77    actualizarColor()
78  def actualizarColor():
79    for i in range(tam):
80      for j in range(tam):
81        if int(canvas.gettags(array[i][j])[0])==1:
82          canvas.itemconfigure(array[i][j], fill=color_vivo)
83        else:
84          canvas.itemconfigure(array[i][j], fill=color_muerto)
85
86
87  def evaluar(por_modificar,i,j):
88    global numero_vivos
89    global numero_muertos
90    vecindad=[]
91    estado=int(canvas.gettags(array[i][j])[0])
92    if i==0:
93      vecindad.append(int(canvas.gettags(array[tam-1][j])[0]))#
      superior
94      vecindad.append(int(canvas.gettags(array[i+1][j])[0]))#
      inferior
95      if j==0:
96        vecindad.append(int(canvas.gettags(array[tam-1][tam-1])
      [0]))#superior izquierdo
97        vecindad.append(int(canvas.gettags(array[tam-1][j+1])[0]))
      #superior derecho
98        vecindad.append(int(canvas.gettags(array[i][tam-1])[0]))#
      izquierdo
99        vecindad.append(int(canvas.gettags(array[i][j+1])[0]))#
      derecho
100       vecindad.append(int(canvas.gettags(array[i+1][tam-1])[0]))
      #inferior izquierdo
101       vecindad.append(int(canvas.gettags(array[i+1][j+1])[0]))#
      inferior derecho
102     elif j==tam-1:
103       vecindad.append(int(canvas.gettags(array[tam-1][j-1])[0]))
      #superior izquierdo
```

```
104         vecindad.append(int(canvas.gettags(array[tam-1][0])[0]))#
    superior derecho
105         vecindad.append(int(canvas.gettags(array[i][j-1])[0]))#
    izquierdo
106         vecindad.append(int(canvas.gettags(array[i][0])[0]))#
    derecho
107         vecindad.append(int(canvas.gettags(array[i+1][j-1])[0]))#
    inferior izquierdo
108         vecindad.append(int(canvas.gettags(array[i+1][0])[0]))#
    inferior derecho
109       else:
110         vecindad.append(int(canvas.gettags(array[tam-1][j-1])[0]))
    #superior izquierdo
111         vecindad.append(int(canvas.gettags(array[tam-1][j+1])[0]))
    #superior derecho
112         vecindad.append(int(canvas.gettags(array[i][j-1])[0]))#
    izquierdo
113         vecindad.append(int(canvas.gettags(array[i][j+1])[0]))#
    derecho
114         vecindad.append(int(canvas.gettags(array[i+1][j-1])[0]))#
    inferior izquierdo
115         vecindad.append(int(canvas.gettags(array[i+1][j+1])[0]))#
    inferior derecho
116     elif i==tam-1:
117       vecindad.append(int(canvas.gettags(array[i-1][j])[0]))#
    superior
118       vecindad.append(int(canvas.gettags(array[0][j])[0]))#
    inferior
119       if j==0:
120         vecindad.append(int(canvas.gettags(array[i-1][tam-1])[0]))
    #superior izquierdo
121         vecindad.append(int(canvas.gettags(array[i-1][j+1])[0]))#
    superior derecho
122         vecindad.append(int(canvas.gettags(array[i][tam-1])[0]))#
    izquierdo
123         vecindad.append(int(canvas.gettags(array[i][j+1])[0]))#
    derecho
124         vecindad.append(int(canvas.gettags(array[0][tam-1])[0]))#
    inferior izquierdo
125         vecindad.append(int(canvas.gettags(array[0][j+1])[0]))#
    inferior derecho
126       elif j==tam-1:
127         vecindad.append(int(canvas.gettags(array[i-1][j-1])[0]))#
    superior izquierdo
128         vecindad.append(int(canvas.gettags(array[i-1][0])[0]))#
```

```
    superior derecho
129       vecindad.append(int(canvas.gettags(array[i][j-1])[0]))#
    izquierdo
130       vecindad.append(int(canvas.gettags(array[i][0])[0]))#
    derecho
131       vecindad.append(int(canvas.gettags(array[0][j-1])[0]))#
    inferior izquierdo
132       vecindad.append(int(canvas.gettags(array[0][0])[0]))#
    inferior derecho
133     else:
134       vecindad.append(int(canvas.gettags(array[i-1][j-1])[0]))#
    superior izquierdo
135       vecindad.append(int(canvas.gettags(array[i-1][j+1])[0]))#
    superior derecho
136       vecindad.append(int(canvas.gettags(array[i][j-1])[0]))#
    izquierdo
137       vecindad.append(int(canvas.gettags(array[i][j+1])[0]))#
    derecho
138       vecindad.append(int(canvas.gettags(array[0][j-1])[0]))#
    inferior izquierdo
139       vecindad.append(int(canvas.gettags(array[0][j+1])[0]))#
    inferior derecho
140   else:
141     vecindad.append(int(canvas.gettags(array[i-1][j])[0]))#
    superior
142     vecindad.append(int(canvas.gettags(array[i+1][j])[0]))#
    inferior
143     if j==0:
144       vecindad.append(int(canvas.gettags(array[i-1][tam-1])[0]))
    #superior izquierdo
145       vecindad.append(int(canvas.gettags(array[i-1][j+1])[0]))#
    superior derecho
146       vecindad.append(int(canvas.gettags(array[i][tam-1])[0]))#
    izquierdo
147       vecindad.append(int(canvas.gettags(array[i][j+1])[0]))#
    derecho
148       vecindad.append(int(canvas.gettags(array[i+1][tam-1])[0]))
    #inferior izquierdo
149       vecindad.append(int(canvas.gettags(array[i+1][j+1])[0]))#
    inferior derecho
150     elif j==tam-1:
151       vecindad.append(int(canvas.gettags(array[i-1][j-1])[0]))#
    superior izquierdo
152       vecindad.append(int(canvas.gettags(array[i-1][0])[0]))#
    superior derecho
```

```
153         vecindad.append(int(canvas.gettags(array[i][j-1])[0]))#
    izquierdo
154         vecindad.append(int(canvas.gettags(array[i][0])[0]))#
    derecho
155         vecindad.append(int(canvas.gettags(array[i+1][j-1])[0]))#
    inferior izquierdo
156         vecindad.append(int(canvas.gettags(array[i+1][0])[0]))#
    inferior derecho
157       else:
158         vecindad.append(int(canvas.gettags(array[i-1][j-1])[0]))#
    superior izquierdo
159         vecindad.append(int(canvas.gettags(array[i-1][j+1])[0]))#
    superior derecho
160         vecindad.append(int(canvas.gettags(array[i][j-1])[0]))#
    izquierdo
161         vecindad.append(int(canvas.gettags(array[i][j+1])[0]))#
    derecho
162         vecindad.append(int(canvas.gettags(array[i+1][j-1])[0]))#
    inferior izquierdo
163         vecindad.append(int(canvas.gettags(array[i+1][j+1])[0]))#
    inferior derecho
164     cantidad_vivos=vecindad.count(1)
165     cantidad_muertos=vecindad.count(0)
166     # print("Soy:"+str(i)+str(j))
167     # print("Vivos vecindad: "+str(cantidad_vivos))
168     # print("Muertos vecindad: "+str(cantidad_muertos))
169     if estado==1:#Esta vivo
170       # print("Esta vivo")
171       if cantidad_vivos in range(reglas[0],reglas[1]+1):
172         pass
173         #Sigue vivo
174         # print("Seguira vivo")
175       else:
176         # print("Murio")
177         por_modificar.append(array[i][j])
178         numero_vivos-=1
179         numero_muertos+=1
180     else:#Esta muerto
181       # print("Esta muerto")
182       if cantidad_vivos in range(reglas[2],reglas[3]+1):#Revive
183         # print("Revive")
184         por_modificar.append(array[i][j])
185         numero_vivos+=1
186         numero_muertos-=1
187       else:
```

```
188         #Sigue muerto
189         pass
190         # print("Seguira muerto")
191 
192 array=[[0] * tam for i in range(tam)]
193 valores=[[0] * tam for i in range(tam)]
194 
195 for i in range(tam):
196   for j in range(tam):
197     valores[i][j]=random.choice([0,1], 1, p=[1-int(argv[1])/100,
    int(argv[1])/100])[0]
198     array[i][j]=canvas.create_rectangle(200+(j*(650/tam)), 50+(i
    *(650/tam)), 200+(650/tam)+(j*(650/tam)), 50+(650/tam)+(i
    *(650/tam)))
199     if valores[i][j]==1:
200       canvas.itemconfigure(array[i][j], fill=color_vivo, width
    =0, tags=str(valores[i][j]))#Verde
201       numero_vivos+=1
202     else:
203       canvas.itemconfigure(array[i][j], fill=color_muerto, width
    =0, tags=str(valores[i][j]))#Rojo
204       numero_muertos+=1
205     canvas.tag_bind(array[i][j], '<Button-1>', cambiar)
206 
207 label_vivos.configure(text="Número de vivos: "+str(numero_vivos)
    )
208 label_vivos.place(x=20, y=40)
209 label_muertos.configure(text="Número de muertos: "+str(
    numero_muertos))
210 label_muertos.place(x=20, y=60)
211 label_generacion.configure(text="Generación: "+str(generacion))
212 label_generacion.place(x=20, y=80)
213 input_reglas.configure(width=10)
214 input_reglas.place(x=20,y=120)
215 
216 boton_pausa=Button(master,text="Empezar",command=pausa)
217 boton_pausa.place(x=30, y=200)
218 
219 boton_color_vivo=Button(master,text="Color vivo",command=
    cambiar_color_vivo)
220 boton_color_muerto=Button(master,text="Color muerto",command=
    cambiar_color_muerto)
221 label_color_vivos=Label(master,width=2,bg=color_vivo)
222 label_color_muertos=Label(master,width=2,bg=color_muerto)
223 boton_color_vivo.place(x=20,y=400)
```

```
label_color_vivos.place(x=150,y=400)
boton_color_muerto.place(x=20,y=430)
label_color_muertos.place(x=150,y=435)

info = open("historico_unos.txt", "w")
info.write(str(numero_vivos)+","+str(generacion)+"\n")
info.flush()

# canvas.config(xscrollcommand=hbar.set)
# canvas.pack(side=LEFT,expand=True,fill=BOTH)
canvas.pack()
# master.mainloop()

while True:
  master.update_idletasks()
  master.update()
  sleep(.1)
  por_modificar=list()
  if continuar:
    reglas = list(map(int, input_reglas.get().split(",")))
    generacion+=1
    for i in range(tam):
      for j in range(tam):
        evaluar(por_modificar,i,j)
    for i in por_modificar:
      if int(canvas.gettags(i)[0])==1:
        canvas.itemconfig(i, fill=color_muerto,tags='0')
      else:
        canvas.itemconfig(i, fill=color_vivo,tags='1')
    actualizarLabels()
    info.write(str(numero_vivos)+","+str(generacion)+"\n")
    info.flush()
  else:
    pass
```

Gráfica de organismos vivos Finalmente, se muestra el archivo "graficar.py" en el cual se realizó la implementación de la gráfica de organismos vivos que existen durante cada una de las generaciones de la simulación del juego de la vida. Los valores a graficar son la cantidad de organismos vivos y la generación a la que corresponden. Estos valores son leídos de un archivo .txt el cual contiene el histórico de organismos vivos, este archivo es actualizado cada que una generación transcurre en el juego de la vida. La gráfica se actualiza automaticamente cada segundo y fue elaborada usando matlibplot.

```
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from matplotlib import style

fig = plt.figure('Historial de unos')
fig.suptitle("Historial de unos")
grafica = fig.add_subplot(1, 1, 1)

def animacion(i):
  info = open("historico_unos.txt", "r").read()
  lineas = info.split("\n")
  xs = []
  ys = []
  for linea in lineas:
    if len(linea) > 1:
      y,x = linea.split(",")
      xs.append(int(x))
      ys.append(int(y))
  # print(ys)
  grafica.clear()
  grafica.set_xlabel('Generación')
  grafica.set_ylabel('Cantidad de unos')
  grafica.plot(xs, ys)

ani=animation.FuncAnimation(fig, animacion, interval=1000)

plt.show()
```

3. Resultados

Para comprobar la funcionalidad del programa, se mostrarán tres patrones de organismos vivientes que el programa simulará para comparar los resultados:

Primer patrón

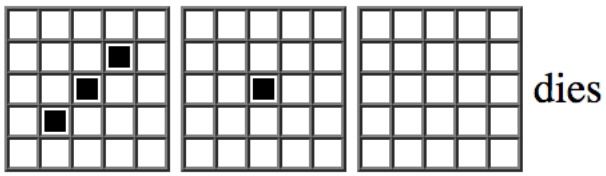

Figura 1: Primer patrón

Resultados obtenidos:

Figura 2: Primer resultado obtenido(1)

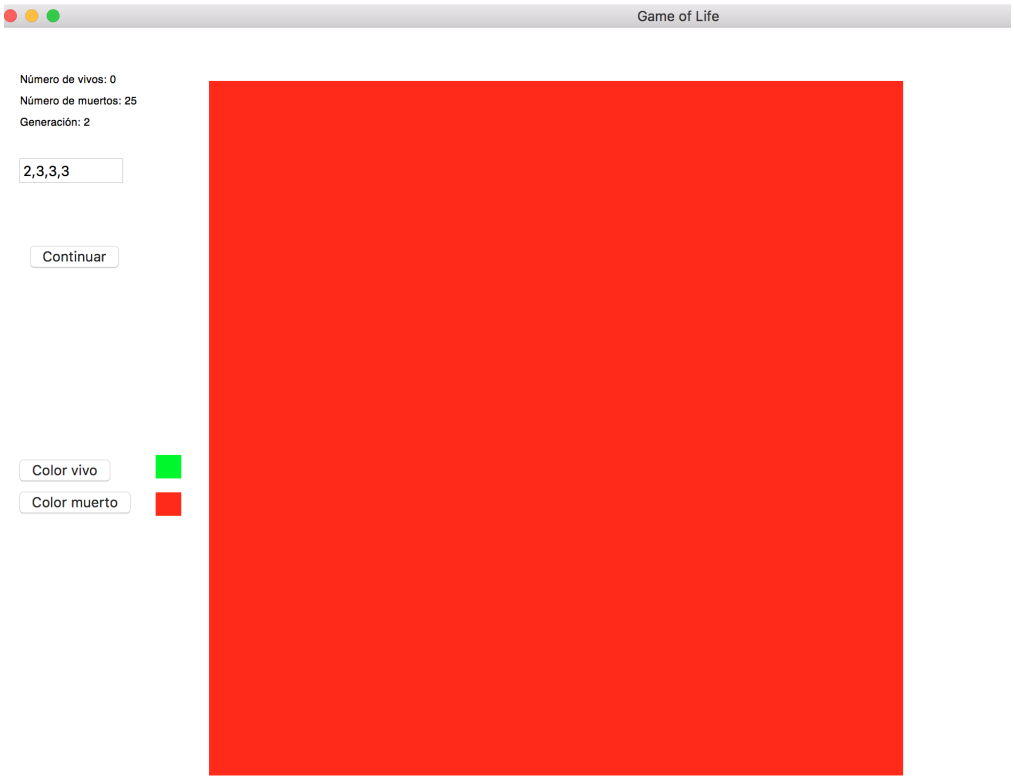

Figura 3: Primer resultado obtenido(2)

Segundo patrón

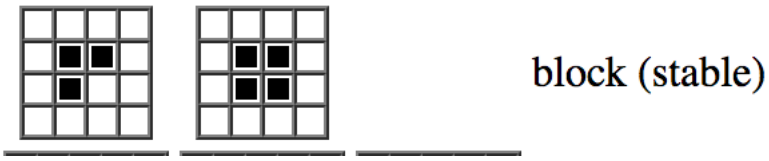

Figura 4: Segundo patrón

Resultados obtenidos:

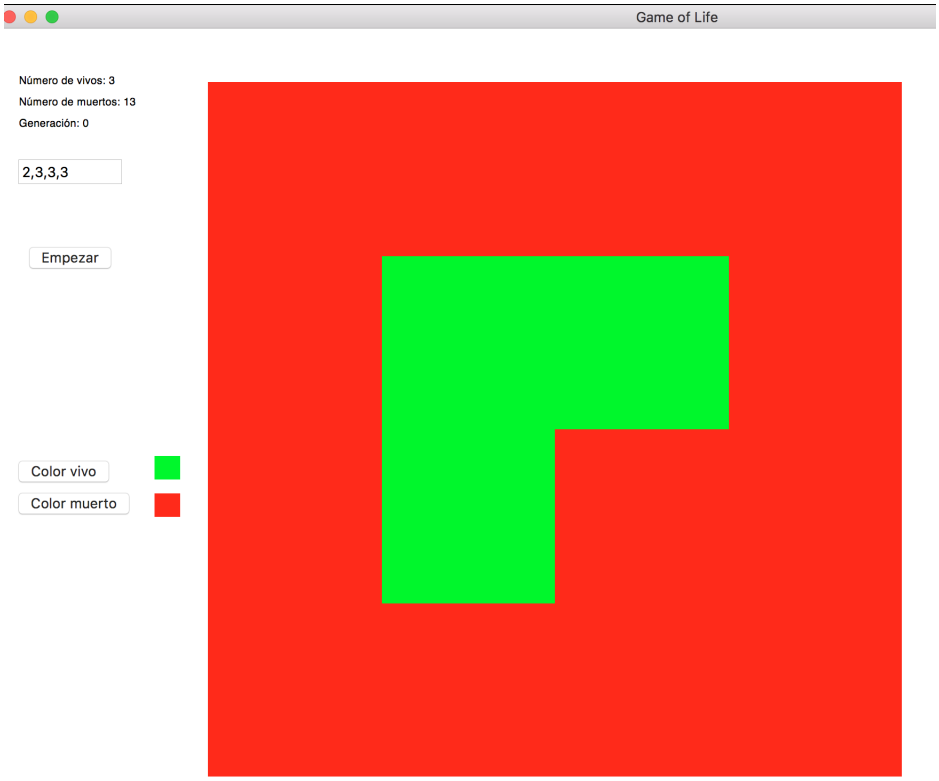

Figura 5: Segundo resultado obtenido(1)

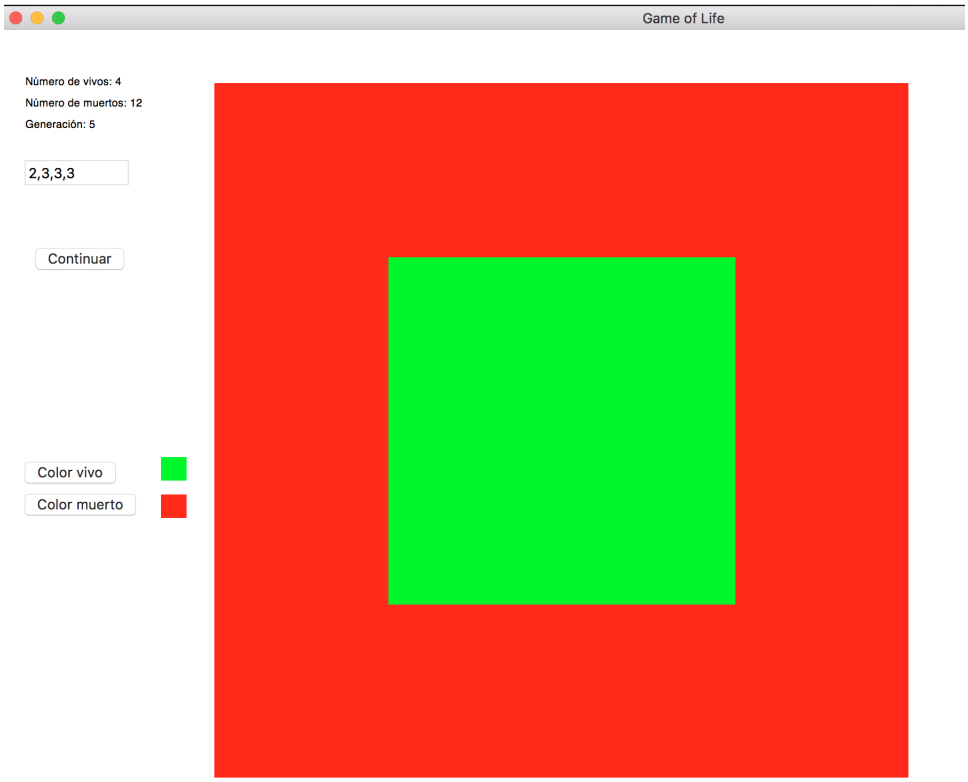

Figura 6: Segundo resultado obtenido(2)

Tercer patrón

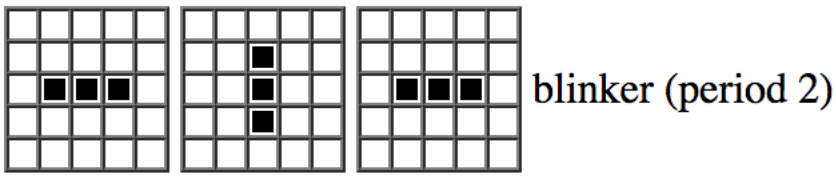

Figura 7: Tercer patrón

Resultados obtenidos:

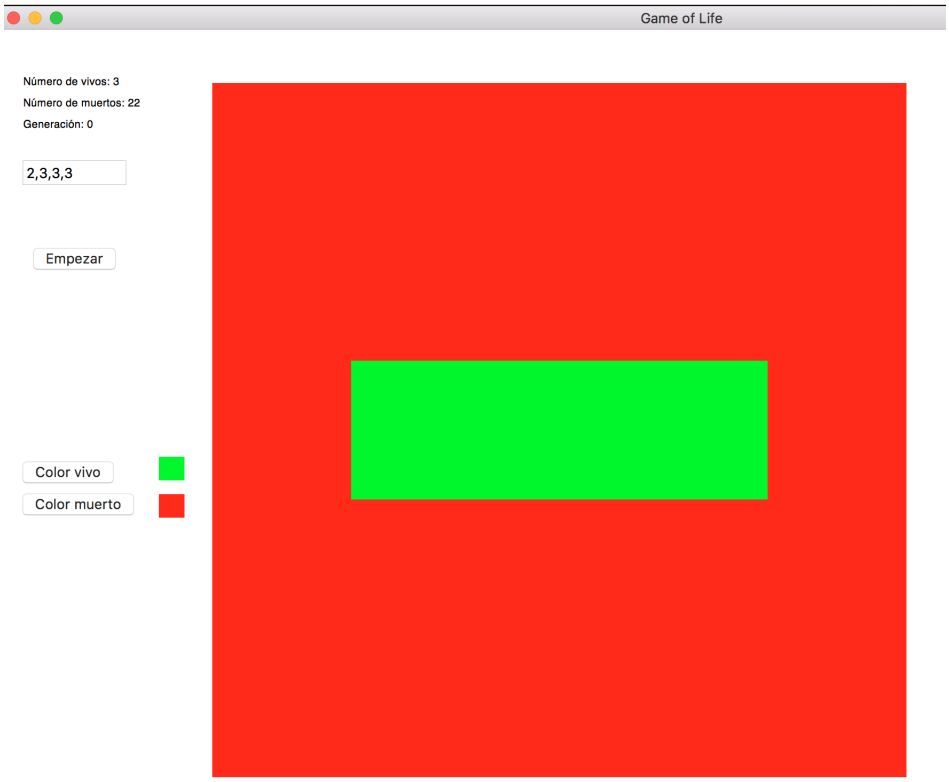

Figura 8: Tercer resultado obtenido(1)

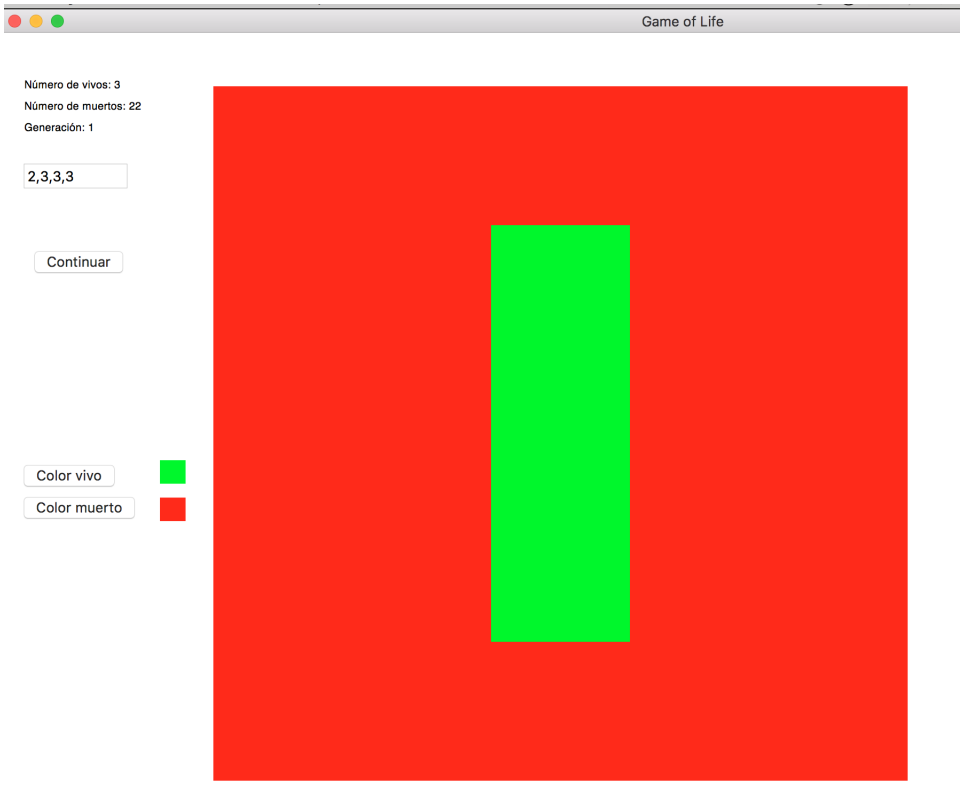

Figura 9: Tercer resultado obtenido(2)

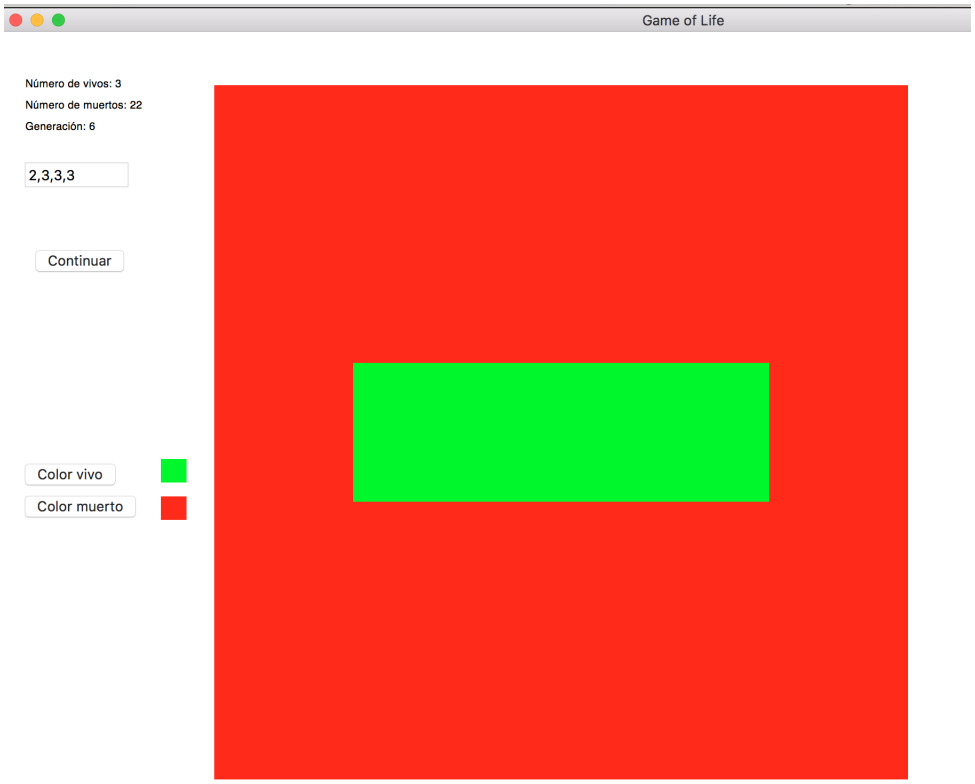

Figura 10: Tercer resultado obtenido(3)

Referencias

[1] M. Gardner, "Mathematical games the fantastic combinations of john conway's new solitaire game life." `https://web.archive.org/web/20090603015231/http://ddi.cs.uni-potsdam.de/HyFISCH/Produzieren/lis_projekt/proj_gamelife/ConwayScientificAmerican.htm`, 1970. [Consultado: 2018-09-19].