Визуализация игры жизнь средствами Python
ЛАБОРАТОРНАЯ РАБОТА

http://www.kimrt.ru
Онлайн-курс
Суперкомпьютерные технологии в задачах моделирования

Игра «Жизнь» – клеточный автомат, разработанный британским математиком Джоном Хортоном Конвеем в 1970 году. Клеточный автомат, непосредственно развивающийся на сетке (теоретически бесконечного размера), состоящей из квадратов-боксов, называемых ячейками, которые имеют бинарное состояние (1 для живых и 0 для мертвых). Ячейки развиваются с течением времени в соответствии с их соседством (каждая ячейка имеет 8 соседних ячеек), что изменяет сетку на каждом этапе эволюции, называемом генерацией [1].

Для визуализации игры «Жизнь» потребуется установка фреймворка GUI tkinter [2] и дальнейшее его использование.

Листинг 1. Визуализация игры «Жизни» на Python

```
from tkinter import *
from tkinter.filedialog import askopenfilename
import tkinter.messagebox as mb

import win32gui
from PIL import Image, ImageGrab

import numpy as np
import scipy.signal

# ГЛАВНЫЕ ПЕРЕМЕННЫЕ
repeat = False
shape = 30
width = 300
height = 300
distance = 10
speed = 500
matrix = []
image_number = 0


# СОСТОЯНИЯ КЛЕТОК
def alive(x, y): # ЖИВАЯ
    global canvas
```

```
        x, y = x * distance, y * distance
        # СВЕТЛО-СЕРОЕ, ЗАЛИВАЕМ ЧЕРНЫМ
        canvas.create_rectangle(x, y, x + distance, y + distance,
fill="#000000", outline="#C6C6C6")


    def dead(x, y): # МЕРТВАЯ
        global canvas
        x, y = x * distance, y * distance
        # СВЕТЛО-СЕРОЕ, ЗАЛИВАЕМ БЕЛЫМ
        canvas.create_rectangle(x, y, x + distance, y + distance,
fill="#FFFFFF", outline="#C6C6C6")


    # ФУНКЦИЯ ОБНОВЛЕНИЯ СЕТКИ
    def updategrid(grid):
        global matrix
        # ИТЕРАТОР, ЧТОБЫ ИЗБЕЖАТЬ ЦИКЛОВ [3]
        iterator = np.nditer(grid, flags=['multi_index'])
        while not iterator.finished:
            # ЕСЛИ 0 - КЛЕТКА МЕРТВАЯ
               if grid[iterator.multi_index] == 0:
                dead(iterator.multi_index[0],
iterator.multi_index[1]) # multi_index[0] - x, multi_index[0] - У
            else: # ИНАЧЕ ОНА ЖИВАЯ
                alive(iterator.multi_index[0],
iterator.multi_index[1])
            iterator.iternext()

    # ПРАВИЛА ИГРЫ
    def rules(current, neighbours):
       # если клетка жива, но соседей меньше 2, значит она умирает
        if current == 1 and neighbours < 2:
            return 0
# если клетка жива и соседей у нее 2 или 3, то она остается живой
        elif current == 1 and neighbours in (2, 3):
            return 1
# если клетка жива, но соседей больше 3, то она умирает от
#перенаселения
        elif current == 1 and neighbours > 3:
            return 0
        # если клетка мертва, но соседей 3, то она оживает
        elif current == 0 and neighbours == 3:
            return 1
        # в других случаях она умирает
        else:
```

```
            return 0

    # ПЕРЕКЛЮЧАТЕЛЬ КНОПКИ ВКЛ/ВЫКЛ
    def switcher():
        global repeat
        global PlayButton
        # ЕСЛИ ПОКА КНОПКА ВЫКЛЮЧЕНА, НО ОНА ГОРИТ ЗЕЛЕНЫМ
        if repeat:
            PlayButton.config(background='LimeGreen')
            repeat = False
        # ЕСЛИ ВЫПОЛНЕНИЕ ВКЛЮЧЕНО, ТО КНОПКА СТАНОВИТСЯ КРАСНОЙ,
ЧТОБЫ ЕЕ ВЫКЛЮЧИТЬ
        else:
            PlayButton.config(background='red4')
            repeat = True
            step()

    # ШАГ ПРОГРАММЫ
    def step():
        global matrix
        global root
        global repeat
        global speed

        speed = slider.get() # ИЗВЛЕКАЕМ ЗНАЧЕНИЕ СО ШКАЛЫ
        # ИСПОЛЬЗУЕМ СВЕРТКУ
        grid  =  scipy.signal.convolve2d(matrix,  np.ones((3,  3)),
mode='same') - matrix

    # ВЫПОЛНЯЕМ ФУНКЦИЮ VERTORIZE, ЧТОБЫ НА ВЫХОДЕ ПОЛУЧИТЬ МАССИВ
ЗНАЧЕНИЙ (Т.К. RULES ВОЗВРАЩАЕТ НЕ МАССИВ, А ЛИШЬ САМИ ЗНАЧЕНИЯ)
        rules_func = np.vectorize(rules)
        grid  =  rules_func(matrix,  grid)  #  ВЫЧИСЛЯЕМ  СОСТОЯНИЕ
КЛЕТКИ ПО ПРАВИЛАМ
        grid.astype(int) # ИЗБЕГАЕМ FLOAT'А
        updategrid(grid) # ОБНОВЛЯЕМ СЕТКУ
        matrix = grid # ТЕПЕРЬ НАША ОСНОВНАЯ МАТРИЦА - ЭТО РАНЕЕ
ВЫСЧИТАННАЯ

        #  ЕСЛИ  ВКЛЮЧИЛИ  ЗАПУСК  ПРОГРАММЫ,  ВЫПОЛНЯЕМ  ЕЕ  С
ОПРЕДЕЛЕННОЙ СКОРОСТЬЮ ВОСПРОИЗВЕДЕНИЯ КАРТИНОК
        if repeat:
            root.after(speed, step)
    def savepng():
        global image_number
```

```
        image_number += 1
        HWND = canvas.winfo_id()
        rect = win32gui.GetWindowRect(HWND)
        im = ImageGrab.grab(rect)

        filename = f'step_{image_number}.png'
        im.save(filename, 'png')
        mb.showinfo("Информация", "Успешно!")

    # ЗАГРУЗКА СОСТОЯНИЙ
    def load():
        global matrix
        global width
        global height
        global shape

        filename = askopenfilename(filetypes=[("Text files",
"*.txt")]) # ВЫБОРКА ИЗ ДИАЛОГОВОГО ОКНА
        f = open(filename, "r")
        s = f.read()
        # ТАК КАК СЧИТАННЫЙ ФАЙЛ ТЕПЕРЬ ВМЕСТО МАССИВА ЯВЛЯЕТСЯ
СТРОЧКОЙ, ТО НАМ ТРЕБУЕТСЯ ИЗБАВИТЬСЯ ОТ ВСЕГО
        # КРОМЕ САМИХ ЧИСЕЛ МАССИВА. ДЛЯ ЭТОГО УДАЛЯЕМ ЗНАКИ [] И
ЗНАК НОВОЙ СТРОЧКИ

        shape = int(s.partition('[')[0])

        # ИНИЦИАЛИЗИРУЕМ МАССИВ СОСТОЯНИЙ
        matrix = np.zeros((shape, shape), dtype=int)

        # СЧИТЫВАЕМ РАЗМЕР СЕТКИ С ФАЙЛА
        s = s.partition('[')[1:][1]
        loaded = np.array(s.replace("]",
"").replace("[","").split(' '))
        loaded = [int(line.rstrip()) for line in loaded] # ТАК ЖЕ
НЕ ЗАБЫВАЕМ ПРЕВРАТИТЬ TR --> INT
        loaded = np.reshape(loaded, (shape, shape)) # ИЗ
ОДНОМЕРНОГО МАССИВА ДЕЛАЕМ ДВУМЕРНЫЙ

        updategrid(loaded) # ОБНОВЛЯЕМ СЕТКУ
        matrix = loaded # ТЕПЕРЬ ЭТО НАШ ГЛАВНЫЙ МАССИВ СОСТОЯНИЙ
        f.close()

    # ОТРИСОВКА ОКНА
    root = Tk()
    root.title("Визуализация игры жизнь")
```

```
    root.geometry("303x390")
    root.resizable(0, 0) # ЧТОБЫ НЕЛЬЗЯ БЫЛО МЕНЯТЬ РАЗМЕР
    canvas = Canvas(root, width=300, height=300)
    canvas.pack()

    # ВЕРТИКАЛЬНАЯ ПРОРИСОВКА СЕТКИ
    for x in range(distance, width, distance):
        canvas.create_line(x, 0, x, height, fill="#C6C6C6")
    # ГОРИЗОНТАЛЬНАЯ ПРОРИСОВКА СЕТКИ
    for y in range(distance, height, distance):
        canvas.create_line(0, y, width, y, fill="#C6C6C6")

    # ШКАЛА СКОРОСТИ
    slider = Scale(root, from_=500, to=10, orient=HORIZONTAL,
label='Скорость', length=width)
    slider.pack()
    slider.set(speed) # БУДЕТ ПРИСВАИВАТЬСЯ ПЕРЕМЕННОЙ SPEED

    # МЕНЮ
    menubar = Menu(root)
    file = Menu(menubar, tearoff = 0)
    file.add_command(label="Загрузить", command = load)
    menubar.add_cascade(label = "Файлы", menu = file,
command=load)

    # ИНИЦИАЛИЗАЦИЯ КНОПОК ПРОГРАММЫ
    NextButton = Button(root, text="Далее", command=step,
anchor=W)
    NextButton.pack()
    NextButton = canvas.create_window(width, height + 25,
anchor=SE, window=NextButton)
    PlayButton = Button(root, width=150, background="LimeGreen",
command=switcher, anchor=W)
    PlayButton.pack()

    SaveButton = Button(root, text="Сохранить картинку",
command=savepng, anchor=W)
    SaveButton.pack()
    SaveButton.place(x=130, y=300)

    root.config(menu = menubar)
    mainloop()
```

```
10
[[0 0 0 0 0 1 0 0 1 0]
 [0 0 0 0 1 0 0 0 0 1]
 [0 0 0 0 0 0 0 0 0 1]
 [0 1 0 0 0 0 0 0 0 0]
 [1 0 0 0 1 1 1 1 0 0]
 [1 0 0 1 0 0 1 0 0 0]
 [0 1 1 0 0 1 0 0 0 0]
 [0 1 0 0 0 1 0 0 0 0]
 [0 0 1 1 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0]]
```

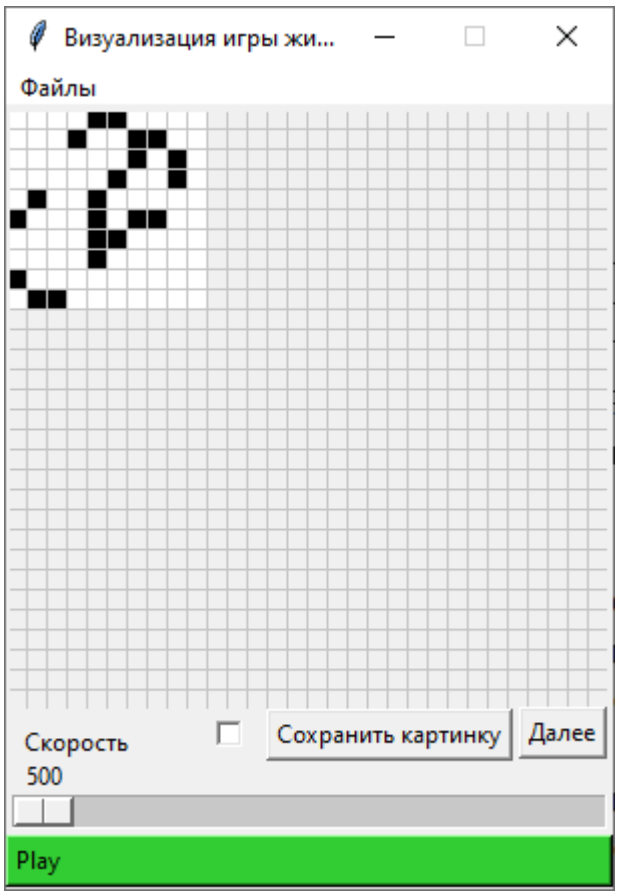

Рис. 1. Текстовый файл данных и программа визуализации

На рис. 1 представлен текстовый файл состояний ячеек. Первая строчка представляет собой число N – размер квадратной решетки. Далее идет массив состояний ячеек. Программа визуализации должна считывать эти данные и прорисовывать. Выполните следующие задания.

Задание 1. Модифицируйте данный выше код визуализации игры «Жизнь», добавив в него возможность сохранения состояний ячеек в текстовый файл на определенном шаге.

Задание 2. Модифицируйте данный выше код визуализации игры «Жизнь», добавив флажок, при активном состоянии которого при воспроизведении картинки сохраняются в ту же папку, где и данные (step_1.png, step_2.png и т.д.).

http://www.kimrt.ru/index/course_stm/0-24

Проект реализуется победителем Конкурса на предоставление грантов преподавателям магистратуры 2020/2021 благотворительной программы «Стипендиальная программа Владимира Потанина» Благотворительного фонда Владимира Потанина.

Источники

1. «Клеточный автомат игра «Жизнь». http://www.kimrt.ru/courses/stm/LabWork/GameOfLife.pdf
2. Tkinter. http://www.kimrt.ru/courses/stm/self_work/Tkinter.pdf
3. Iterating Over Arrays. https://numpy.org/doc/stable/reference/arrays.nditer.html