Nome: Rafael Montagna Copes Matrícula: 19100930

Barnsley Fern (Samambaia de Barnsley)

Para esse trabalho resolvi implementar a Barnsley Fern, que é um fractal em forma de samambaia nomeado em homenagem ao matemático Michael Barnsley que o primeiramente o descreveu em seu livro *Fractals Everywhere*. Para a interface é usado Python, e para os cálculos é usado C. As informações detalhadas de como executar estão no Readme.md no Github.

Construções da Barnsley Fern:

Esse fractal é construído usando 4 transformações de matrizes:

$$f_1(x,y) = \begin{bmatrix} 0.00 & 0.00 \\ 0.00 & 0.16 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$f_2(x,y) = \begin{bmatrix} 0.85 & 0.04 \\ -0.04 & 0.85 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.00 \\ 1.60 \end{bmatrix}$$

$$f_3(x,y) = \begin{bmatrix} 0.20 & -0.26 \\ 0.23 & 0.22 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.00 \\ 1.60 \end{bmatrix}$$

$$f_4(x,y) = \begin{bmatrix} -0.15 & 0.28 \\ 0.26 & 0.24 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.00 \\ 0.44 \end{bmatrix}$$

Essas transformações são aplicadas em sequência, o ‘x’ e ‘y’ de uma alimenta a próxima iteração.

O algoritmo decide qual transformação utilizar de acordo com probabilidades definidas para cada uma. Nessa tabela, o ‘p’ descreve essas probabilidades:

w	a	b	c	d	e	f	p	Portion generated
f_1	0	0	0	0.16	0	0	0.01	Stem
f_2	0.85	0.04	−0.04	0.85	0	1.60	0.85	Successively smaller leaflets
f_3	0.20	−0.26	0.23	0.22	0	1.60	0.07	Largest left-hand leaflet
f_4	−0.15	0.28	0.26	0.24	0	0.44	0.07	Largest right-hand leaflet

O primeiro ponto se localiza na origem ($x_0 = 0$, $y_0 = 0$).

Os próximos pontos são calculados com as transformações e suas probabilidades:

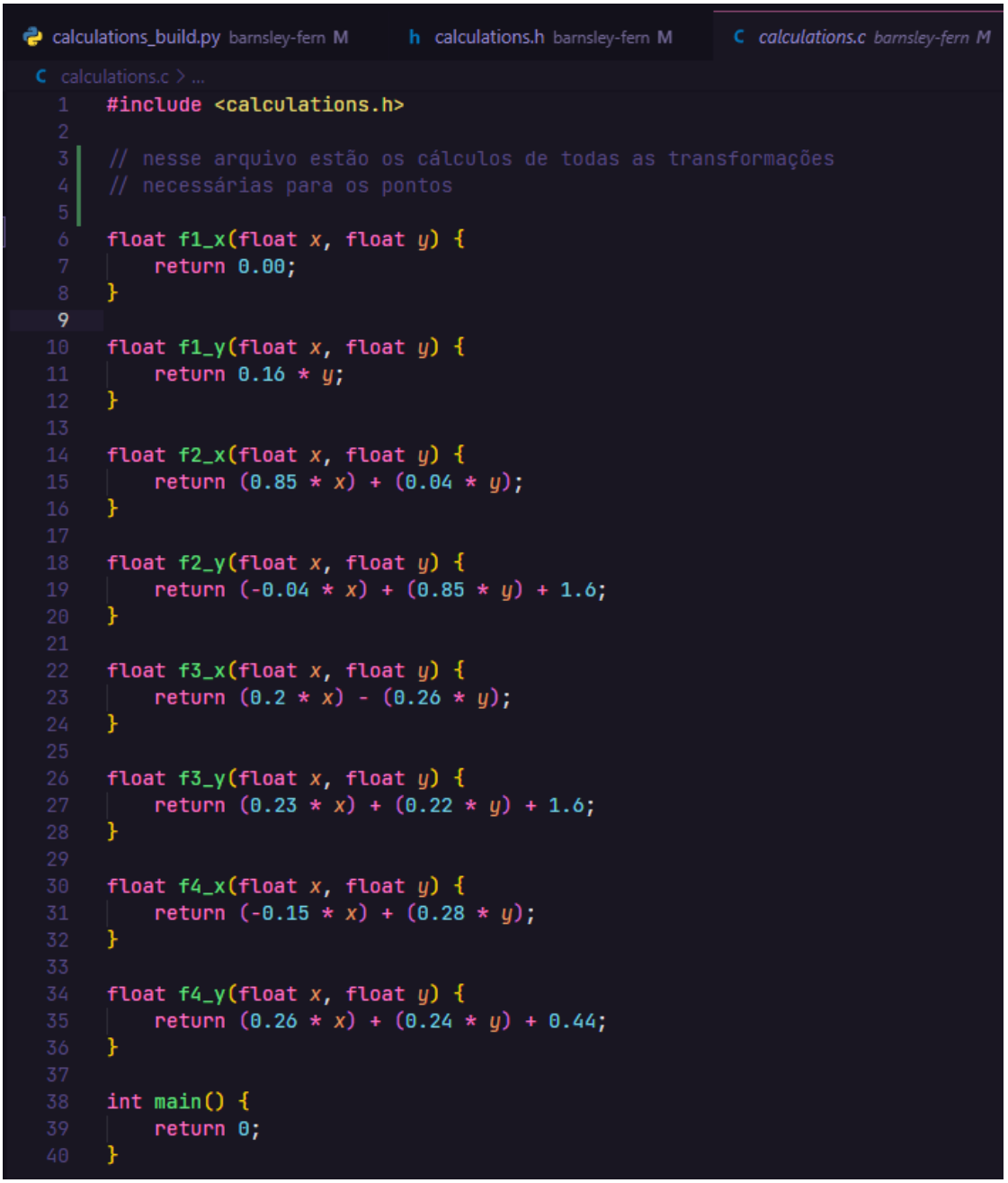
f_1:
$x_{n+1} = 0$
$y_{n+1} = 0.16\ y_n$.

f_2:
$x_{n+1} = 0.85\ x_n + 0.04\ y_n$
$y_{n+1} = -0.04\ x_n + 0.85\ y_n + 1.6$.

f_3:
$x_{n+1} = 0.2\ x_n - 0.26\ y_n$
$y_{n+1} = 0.23\ x_n + 0.22\ y_n + 1.6$.

f_4:
$x_{n+1} = -0.15\ x_n + 0.28\ y_n$
$y_{n+1} = 0.26\ x_n + 0.24\ y_n + 0.44$.

Para esses cálculos eu utilizei C, e estão no arquivo calculations.c.




```c
#include <calculations.h>

// nesse arquivo estão os cálculos de todas as transformações
// necessárias para os pontos

float f1_x(float x, float y) {
    return 0.00;
}

float f1_y(float x, float y) {
    return 0.16 * y;
}

float f2_x(float x, float y) {
    return (0.85 * x) + (0.04 * y);
}

float f2_y(float x, float y) {
    return (-0.04 * x) + (0.85 * y) + 1.6;
}

float f3_x(float x, float y) {
    return (0.2 * x) - (0.26 * y);
}

float f3_y(float x, float y) {
    return (0.23 * x) + (0.22 * y) + 1.6;
}

float f4_x(float x, float y) {
    return (-0.15 * x) + (0.28 * y);
}

float f4_y(float x, float y) {
    return (0.26 * x) + (0.24 * y) + 0.44;
}

int main() {
    return 0;
}
```

No arquivo main.py é utilizado essas funções que realizam os cálculos de acordo com suas probabilidades, e eu utilizo a biblioteca ‘matplotlib’ para plotar numa interface.


```python
from random import randint

import matplotlib.pyplot as plt
from _calculations_cffi import ffi, lib
from matplotlib.pyplot import figure

# número de ponto a serem redenizados
POINTS = 50000

def main():
    x = []
    y = []
    x.append(0)
    y.append(0)

    current = 0

    # para cada ponto aplica a função de acordo com sua probabilidade
    for i in range(1, POINTS):
        z = randint(1, 100)

        if z == 1:
            x.append(lib.f1_x(x[current], y[current]))
            y.append(lib.f1_y(x[current], y[current]))

        if z >= 2 and z <= 86:
            x.append(lib.f2_x(x[current], y[current]))
            y.append(lib.f2_y(x[current], y[current]))

        if z >= 87 and z <= 93:
            x.append(lib.f3_x(x[current], y[current]))
            y.append(lib.f3_y(x[current], y[current]))

        if z >= 94 and z <= 100:
            x.append(lib.f4_x(x[current], y[current]))
            y.append(lib.f4_y(x[current], y[current]))

        current = current + 1

    # plota a interface
    with plt.style.context('dark_background'):
        figure(figsize=(8, 10), dpi=80)
        plt.scatter(x, y, s = 0.2, edgecolor ='green')
        plt.grid(False)
        plt.axis('off')
        plt.show()

if __name__ == '__main__':
    main()
```

Como foi feito a linkagem entre os códigos em C e Python:

Foi utilizada a biblioteca CFFI, que tem como objetivo chamar código C de dentro do código Python.

O bind é feito no arquivo calculations_build.py onde a biblioteca é chamada, ela necessita saber do arquivo header(.h) e implementação(.c), ele compila o código e gera o bind em um pacote automaticamente numa pasta Release. Com isso, é possível utilizar as funções C no Python a lib desse pacote. Nesse caso eu chamei esse pacode de _calculations_cffi:

calculations_build.py barnsley-fern M

h calculations.h barnsley-fern M

calculations_build.py > ...

```python
import pathlib

import cffi

# builder que para gerar o bind entre C e Python
ffibuilder = cffi.FFI()

# pega o arquivo header
this_dir = pathlib.Path().absolute()
h_file_name = this_dir / "calculations.h"
with open(h_file_name) as h_file:
    ffibuilder.cdef(h_file.read())

# seta o arquivo header e o ponto .c a se linkar
ffibuilder.set_source(
    "_calculations_cffi",
    '#include "calculations.h"',
    sources=["calculations.c"],
    include_dirs=[this_dir.as_posix()],
    extra_link_args=["-Wl,-rpath,."],
)

# compila o código C e gera a pasta Realise
# com o pacote e as funções
if __name__ == "__main__":
    ffibuilder.compile(verbose=True)

```

Para as funções estão dentro da lib do pacote _calculations_cffi, e podem ser acessadas por ela.

```python
from random import randint

import matplotlib.pyplot as plt
from _calculations_cffi import ffi, lib
from matplotlib.pyplot import figure
```

```python
if z == 1:
    x.append(lib.f1_x(x[current], y[current]))
    y.append(lib.f1_y(x[current], y[current]))
```

Aqui está o fractal gerado através dos códigos:

Links consultados para realizar esse trabalho:

https://en.wikipedia.org/wiki/Barnsley_fern
https://cffi.readthedocs.io/en/latest/overview.html
https://matplotlib.org