FRACTAL L-SYSTEM

Presenta

Cristian David Mora Saenz

Docente

Segundo Fidel Puerto Garavito

Asignatura

Diseño de Algoritmos

NRC: 7487

Corporación Universitaria Minuto de Dios

Facultad de Ingeniería

Bogotá D.C, Colombia Mayo 31 de 2020.

Librería “Turtle”

Para este ejercicio conocido como “Peano Gosper Curve”, se encarga de generar figuras, que reproducen la visión de un cubo, pero su figura final siempre es la misma, lo que lo hace un fractal estático, toda la figura se forma implementando trazos diagonales, esto se realiza con la implementación de la librería “turtle” y con un sistema Lindenmayer (L-System).

¿Cómo funciona un fractal de tipo L-System?

Para este ejercicio cada uno de los compañeros debía elegir un tipo de fractal, para presentarlo como examen final, en este caso he elegido un fractal de tipo L-System.

Este tipo de fractales es un sistema de re-escritura de cadenas que puede ser usado para generar fractales de dimensión entre 1 y 2. Los fractales de sistema de Lindenmayer (L-System), deben contar con las siguientes características para realizar su función:

- ***Un alfabeto*** Un conjunto de símbolos que el Sistema L va a utilizar.
- ***Un axioma*** La cadena original que se utiliza en la generación.
- ***Un conjunto de reglas de producción*** estas reglas indica como debe ser reemplazado cada uno de los símbolos en la siguiente iteración.

Debido a que vamos a utilizar la librería “Turtle” para graficar y los Sistemas L para representar los fractales, necesitamos entender de qué manera podemos relacionarlos.Debido a que la librería “Turtle” sólo recibirá los comandos mencionados arriba, debemos asignar un símbolo a cada uno, que en conjunto representarán el alfabeto del Sistema Lindenmayer

- ***F:*** Mover hacia adelante (del inglés Forwards).
- **+*:*** Girar a la derecha.
- ***-:*** Girar a la izquierda.

Ejemplo del resultado del fractal:

Código y explicación

```
import turtle
from math import cos, sin, radians

def create_l_system(iters, axiom, rules):
    start_string = axiom
    if iters == 0:
        return axiom
    end_string = ""
    for _ in range(iters):
        end_string = "".join(rules[i] if i in rules else i for i in start_string)
        start_string = end_string

    return end_string
```

Como se observa en la imagen anterior, se importa la librería “turtle” explicada anteriormente, al igual que las funciones coseno, seno y radian de la librería math, la función que se denomina “create_l_system” define el funcionamiento del fractal, entre estos tiene como parámetros:

- **Iteraciones**: Que representa el número de veces que se forma el fractal.
- **Axioma:** Que funciona generar la primera iteración del fractal.
- **Las reglas:** Que más adelante serán utilizadas para determinar los movimientos del fractal.

Luego de esto se define la función que graficara paso a paso cada trazo del fractal L-System:

```
def draw_l_system(t, instructions, angle, distance):
    steps = len([i for i in instructions if i in "FB"])
    step = 1 / steps
    i = 0
    for cmd in instructions:
        if cmd == 'F':
            t.forward(distance)
        elif cmd == 'B':
            t.backward(distance)
        elif cmd == '+':
            t.right(angle)
        elif cmd == '-':
            t.left(angle)
```

Como se puede observar, la función denominada “draw_l_system” tiene como parámetros:

- **T**: Es la letra que representará más adelante el tablero o la interfaz en la que se mueve el fractal
- **Instrucciones**: Que son las letras que representan si el fractal debe avanzar hacia arriba o abajo y los signos de positivo y negativo, que representan los movimientos hacia la izquierda o derecha.
- **Ángulo:** Como el nombre lo indica es el ángulo en el que se mueve cada instrucción.
- **Distancia:** Es la distancia que recorre cada trazo para no encontrarse con otro mientras se forma la figura.

Luego tenemos la función con la que se mide la profundidad y longitud de nuestro fractal L-System:

```
def calc_length_height(instructions, angle, correction_angle):
    current_angle = correction_angle
    x_offset = 0
    y_offset = 0
    min_x = 0
    min_y = 0
    max_x = 0
    max_y = 0
    for inst in instructions:
        if inst == "F":
            x_offset += cos(radians(current_angle))
            y_offset += sin(radians(current_angle))
        elif inst == "B":
            x_offset -= cos(radians(current_angle))
            y_offset -= sin(radians(current_angle))
        elif inst == "+":
            current_angle -= angle
        elif inst == "-":
            current_angle += angle
        max_x = max(max_x, x_offset)
        min_x = min(min_x, x_offset)
        max_y = max(max_y, y_offset)
        min_y = min(min_y, y_offset)

    width = abs(max_x) + abs(min_x)
    height = abs(max_y) + abs(min_y)


    return width, height, abs(min_x), abs(min_y)
```

Utilizamos las variable X y Y para representar las posiciones de los trazos, en estos utilizamos las funciones de la librería “math”, esta función utiliza un ciclo FOR y condicionales IF y ELSEIF, para encontrar las posiciones en las que se debe mover cada trazo del fractal, teniendo en cuenta el ancho y alto del mismo.

Por último tenemos una función “main”, para ejecutar nuestro código, dentro de esta se colocan las instrucciones, ángulos, tablero, iteraciones, ancho y alto del fractal para implementar las funciones vistas anteriormente.

```
def main(iterations, axiom, rules, angle, length=None, size=None, correction_angle=0,
        y_offset=None, x_offset=None, offset_angle=None, inverted=False, flip_h=False,
        flip_v=False, width=None, height=None, margin=None, aspect_ratio=None):

    inst = create_l_system(iterations, axiom, rules)

    width_, height_, min_x, min_y = calc_length_height(inst, angle, correction_angle)

    if width_ == 0 and height_ == 0:
        return

    if aspect_ratio is None:
        if 0 in [width_, height_]:
            aspect_ratio = 1
        else:
            aspect_ratio = width_ / height_

    if width is None and height:
        width = height / aspect_ratio

    if height is None and width:
        height = width / aspect_ratio

    if margin is None:
        margin = 35

    if offset_angle is None:
        offset_angle = -90
```

```
    if length is None:
        if width_ > height_:
            length = (width - 2 * margin) / width_
        else:
            length = (height - 2 * margin) / height_

    if width_ * length > width:
        length = (width - 2 * margin) / width_
    elif height_ * length > height:
        length = (height - 2 * margin) / height_

    if x_offset is None:
        if width_ >= height_  and (width - width_) <= width_ - 2 * margin :
            x_offset = -(width / 2 - margin) + min_x * length
        else:
            x_offset = -(width / 2) + (width - width_ * length) / 2 + min_x * length

    if y_offset is None:
        if height_ >= width_ and (height - height_) <= height_ - 2 * margin :
            y_offset = -(height / 2 - margin) + min_y * length
        else:
            y_offset = -(height / 2) + (height - height_ * length) / 2 + min_y * length

    if inverted:
        inst = inst.replace('+', '$')
        inst = inst.replace('-', '+')
        inst = inst.replace('$', '-')
        inst = inst.replace('F', '$')
        inst = inst.replace('B', 'F')
        inst = inst.replace('$', 'B')
```

```
    if flip_h:
        inst = inst.replace('F', '$')
        inst = inst.replace('B', 'F')
        inst = inst.replace('$', 'B')
        y_offset *= -1

    if flip_v:
        inst = inst.replace('+', '$')
        inst = inst.replace('-', '+')
        inst = inst.replace('$', '-')
        y_offset *= -1

    if size is None:
        if length < 3:
            size = 1
        elif length < 12:
            size = 2
        elif length < 25:
            size = 3
        else:
            size = 5

    t = turtle.Turtle()
    wn = turtle.Screen()
    wn.setup(width, height)
```

```
    t.up()
    t.backward(-x_offset)
    t.left(90)
    t.backward(-y_offset)
    t.left(offset_angle)
    t.down()
    t.speed(0)
    t.pensize(size)
    draw_l_system(t, inst, angle, length)
    t.hideturtle()

    wn.exitonclick()

# Global parameters

width = 450

title = "Peano-Gosper-Curve"
axiom = "FX"
rules = {"X":"X+YF++YF-FX--FXFX-YF+", "Y":"-FX+YFYF++YF+FX--FX-Y"}
iterations = 3 # TOP: 6
angle = 60

offset_angle = -90 + 15 * iterations
correction_angle = 15 * iterations

main(iterations, axiom, rules, angle, correction_angle=correction_angle,
    offset_angle=offset_angle, aspect_ratio=1, width=width)
```