Algoritmos y Programación I

Cátedra Diego Essaya

Práctica Alan

Trabajo Práctico III

Alumnos:

Guerra, Lucas
Legajo: 104096

Patrone, Florencia
Legajo: 102863

Corrector
Martín Margonari

Fecha de entrega : 03 de Junio del 2019

Fractales

Consigna

Implementar un programa que permita generar imágenes fractales, mediante un algoritmo basado en sistemas-L, una simulación de gráficos tortuga y el formato de imágenes estándar SVG.

Referencias

Gráficos Tortuga

Es un método para generar imágenes usando un cursor (la tortuga) relativo a unas coordenadas cartesianas.

La tortuga tiene tres atributos:

- **Posición:** en el plano (que puede expresado en coordenadas cartesianas con dos números reales (x , y)). La tortuga comienza en la posición (0, 0).
- **Orientación:** Expresada como un ángulo.
- **Pluma:** teniendo atributos como color, ancho y un indicador de pluma arriba y abajo. La tortuga comienza con la pluma abajo.

La tortuga se mueve con comandos relativos a su posición, como **adelante 10** y **gira a la izquierda 90°**. Si la pluma está "abajo", la tortuga dibujará una línea recta cada vez que avance.

En nuestra implementación, la tortuga deberá ser capaz de responder, al menos, a los siguientes comandos:

- **adelante(n):** Avanza n unidades en la dirección actual. Si la pluma está abajo, dibuja una línea recta entre las posiciones inicial y final..
- **derecha(** α**)** , **izquierda(** α**)**: Gira según el ángulo α a la derecha o izquierda.
- **pluma _arriba()** , **pluma _abajo()**: Asigna el estado de la pluma. 92

Sistemas-L

Un sistema-L o un sistema de Lindenmayer es un conjunto de símbolos y reglas que tiene una naturaleza recursiva y que permite, entre otras cosas, generar imágenes fractales.

Un sistema-L está formado por:

- **Alfabeto:** Conjunto de todos los símbolos válidos.
- **Axioma:** Una cadena de símbolos del alfabeto que define el estado inicial del sistema.
- **Reglas de transformación:** Cada regla de transformaciń consiste de una cadena predecesora y una cadena sucesora..

El formato SVG

Scalable Vector Graphics o SVG es un formato de archivo basado en XML que permite codificar imágenes.

Un ejemplo de una imagen simple codificada en formato SVG es:

```
<svg viewBox="0 0 300 200" xmlns="http://www.w3.org/2000/svg">
  <rect width="300" height="200" fill="red" />
  <circle cx="150" cy="100" r="80" fill="green" />
  <text x="150" y="125" font-size="60" text-anchor="middle" fill="white">SVG</text>
</svg>
```

Captura 1: Ejemplo imágen SVG.

El programa

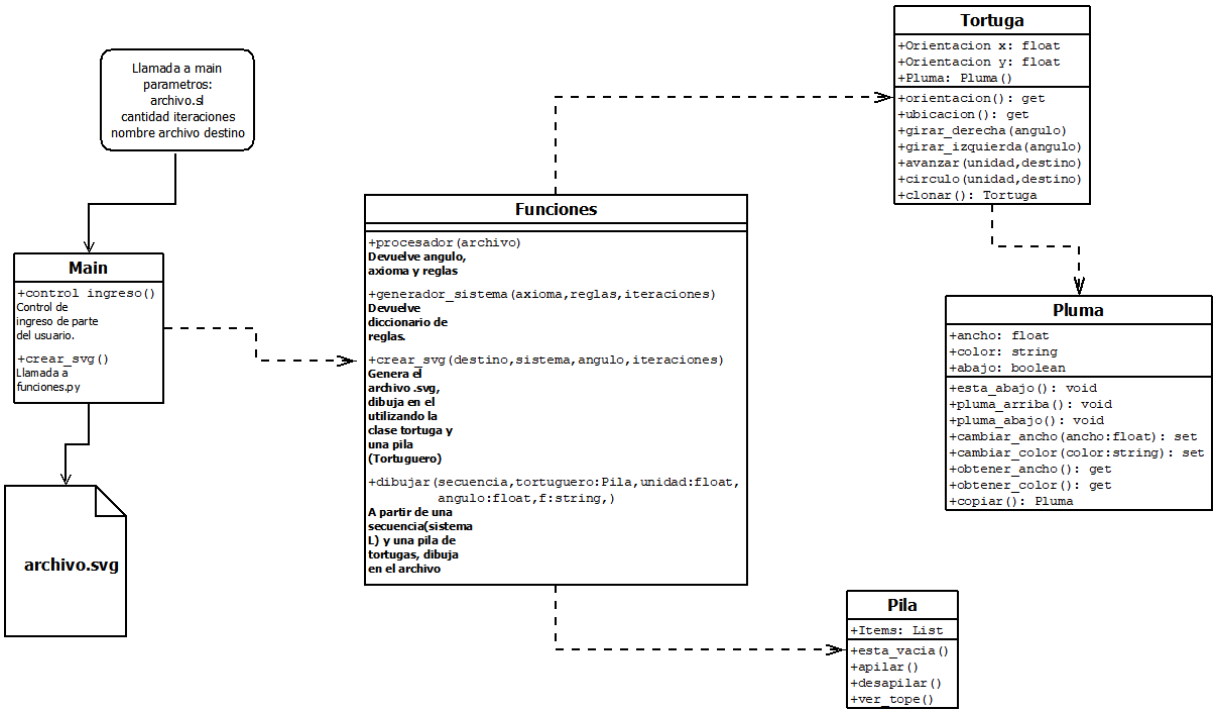
El programa debe recibir tres parámetros por la línea de comandos (recordar sys.argv):

- El nombre del archivo que contiene la descripción del sistema-L.
- La cantidad de iteraciones a procesar.
- El nombre del archivo SVG a escribir.

Por ejemplo:

```
$ python tp3.py arbol.sl 3 arbol.svg
```

Diseño



Captura 2: Diseño UML.

Main

Se encarga de controlar los parámetros ingresados por el usuario y realizar las llamadas a funciones principales. Utilizamos un **try** para manejar los errores que puedan surgir.

Funciones

- **generador _sistema()**: decidimos utilizar un diccionario para almacenar las reglas de cambio del sistema, ya que cada símbolo tiene una traduccián única, por lo tanto no habrá claves repetidas. Además independientemente del tamaño del diccionario, al acceder a sus elementos, en particular en la función '**generador _sistema**', lo hará en tiempo constante. Funciona con recursión

- **crear _svg()** utilizamos una pila para guardar las tortugas, dado que sólo puede haber una tortuga activa a la vez, y esta debe ser siempre la última agregada, por lo que esta estructura, al permitirnos acceder en tiempo constante al último elemento añadido, era la más adecuada.

- **dibujar()** : es invocada desde **crear _svg()**, se encarga de apilar las nuevas tortugas e invocar sus métodos dependiendo del comando que lea desde la secuencia. Funciona con recursión

Clase Pluma

Modela el comportamiento de la pluma que poseen las tortugas del modulo **turtle** de Python, la misma controla entre otras cosas el ancho de las líneas y su color.

Algunos de los métodos implementados en la clase:

- **pluma _arriba()**, **pluma _abajo()**: Cambian el estado del atributo **abajo** de la pluma.
- **copiar()** : Nos devuelve una copia del estado actual de la **pluma**, especialmente útil al momento de clonar una **tortuga**.

Clase Tortuga

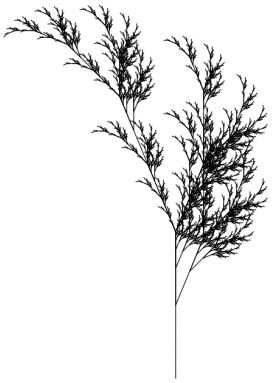
Simula el comportamiento de las tortugas del modulo **turtle** de Python, implementando operaciones como **avanzar(n)** , **girar _derecha(α)** entre otras .

Como atributos, la tortuga tiene una **ubicacion** (coordenadas x e y), una **Pluma** y una **orientación**, ésta última nos permite saber en todo momento a donde apunta la tortuga, sobretodo luego de girar en reiteradas ocasiones.

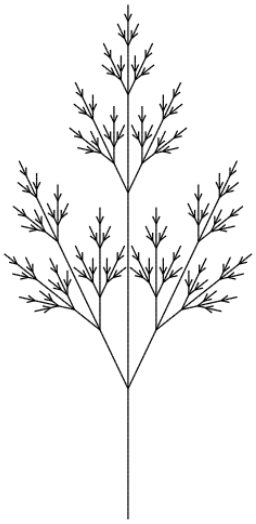
Algunos de los métodos implementados en la clase:

- **girar _derecha(α)**, **girar _izquierda(α)**: Cambian el estado de la **orientacion** de la tortuga, girándola hacia la derecha o izquierda un ángulo α .
- **avanzar(n)** : Actualiza la **ubicación** de la **tortuga**, desplazándola **n** posiciones en sus coordenadas x e y, en caso de que la **pluma** se encuentre abajo, la función también se encarga de escribir en el archivo.
- **circulo(n)** : Si la **pluma** esta abajo dibuja un cículo con centro en las coordenacas x e y de la **ubicación** y radio el **ancho de la pluma**, toma también el color de la pluma.
- **clonar( )** : Nos devuelve una nueva tortuga con los atributos de la **tortuga anterior (ubicacion, orientacion y estado de pluma)**, éste método es imprescindible a la hora de **apilar** y **desapilar** tortugas en nuestro tortuguero.

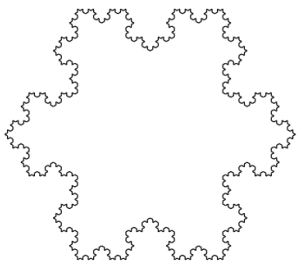
Fractales generados



Captura 3: Ejemplo del enunciado.



Captura 4: arbol2.sl .



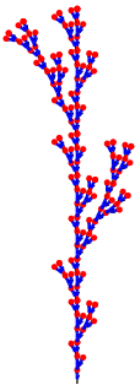
Captura 5: nieve.sl .

EXTRAS

Para la implementación de los **extras**, utilizamos los métodos de la clase **Pluma** para modificar los atributos **ancho** y **color** de la pluma que posee cada tortuga.

También agregamos a la clase **Tortuga** el método **circulo(n)** descrito anteriormente .

Por último se ajustaron los **ifs** de la función **dibujar()** para que aceptara los comandos **(a, b, 1, 2 y L)**.



Captura 6: Ejemplo del enunciado, sección extras.

Informe escrito con LaTeX