Instituto Tecnológico de Costa Rica

Bachillerato de Ingeniería en Computación

IC-2001 Estructuras de Datos - Prof. Mauricio Avilés

Proyecto Programado I - Fractales

Introducción

Un fractal es una figura geométrica cuya estructura básica se repite a diferentes escalas. El término fue propuesto por el matemático polaco Benoît Mandelbrot en 1975, y proviene del latín *fractus* que significa fracturado. Los fractales son demasiado irregulares para ser descritos en términos geométricos tradicionales y también son autosimilares, su forma está compuesta por copias más pequeñas de la misma figura. Esta última característica hace posible que los fractales puedan ser generados por medio de procedimientos en naturaleza recursivos. El objetivo de este proyecto programado es utilizar estructuras de datos lineales para facilitar la representación gráfica en pantalla de diferentes fractales.

Generación de fractales

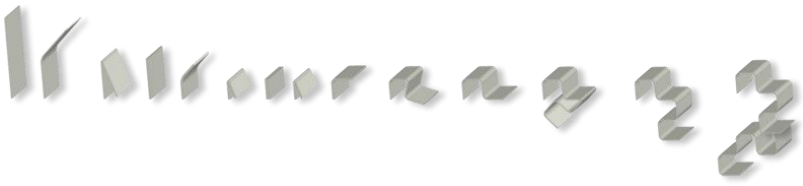
Los tres tipos de fractales que se cubren en este proyecto son conocidos como "curvas", esto quiere decir que pueden ser dibujados a partir de una línea continua que puede tener ángulos.

1. Curva del dragón
2. Curva de punta de flecha de Sierpinski
3. Curva de Lévy C

A continuación se presenta una descripción de cada uno de estos fractales y cómo generarlos.

Curva del dragón

Este fractal se construye a partir de segmentos de igual tamaño y ángulos de 90 grados. Esta figura sigue el mismo patrón que se obtiene al doblar repetidas veces por la mitad una tira de papel e interpretando cada doblez como un ángulo de 90 grados.



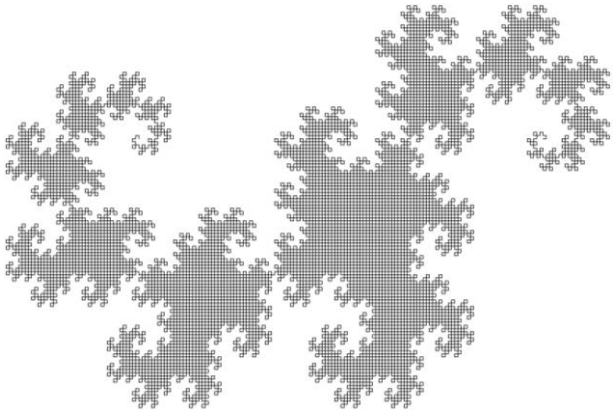


Ilustración 1 http://en.wikipedia.org/wiki/Dragon_curve

La curva del dragón puede crearse dibujando segmentos del mismo tamaño, pero realizando un giro de 90 grados hacia la izquierda o hacia la derecha entre ellos. La curva más simple contiene solamente un giro hacia la derecha. Si se representan los ángulos encontrados dentro de la curva como una lista con la dirección del ángulo, la curva más simple sería:

[D]

Para generar la siguiente iteración de la curva, se van a agregar nuevos ángulos entre cada uno de los existentes en la lista. Incluso al inicio y al final. Estos ángulos nuevos en la lista se agregan uno hacia la

izquierda y otro hacia la derecha, empezando hacia la derecha. La segunda iteración sería (en negrita el giro de la iteración anterior):

[D, **D**, I]

Siguiendo este mismo patrón, la siguiente iteración es (en negrita los elementos de la iteración anterior):

[D, **D**, I, **D**, D, **I**, I]

Como último ejemplo tenemos la cuarta iteración:

[D, **D,** I, **D,** D, **I,** I, **D,** D, **D,** I, **I,** D, **I**, I]

Si se toma esta lista y se recorre dibujando un segmento seguido de un ángulo de 90 grados en la dirección indicada por cada elemento, se obtendrá la representación gráfica de la curva del dragón.

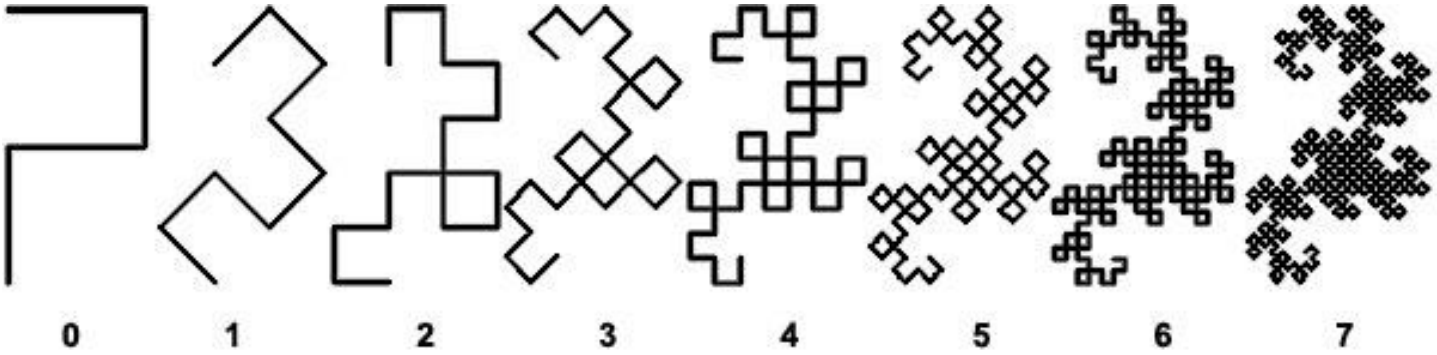


Ilustración 2 http://fractalfoundation.org/OFC/OFC-2-3.html

Curva de punta de flecha Sierpinski

Esta curva tiene una apariencia similar y un límite idéntico al triángulo de Sierpinski, el cual consiste en un triángulo equilátero que está dividido en cuatro triángulos equiláteros pequeños. Cada uno de los triángulos ubicados en las esquinas del triángulo principal recibe el mismo tratamiento que el triángulo original, repitiendo el mismo patrón dentro de un área más pequeña.

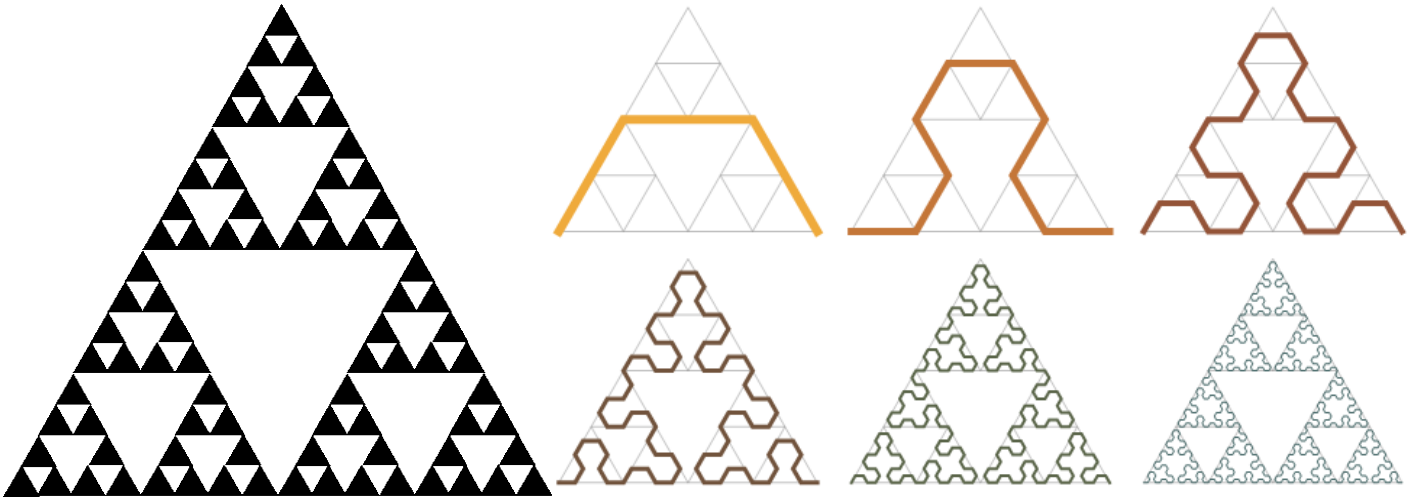


Ilustración 3 http://www.math.unl.edu/~s-bbockel1/dsweb/lesson1/index.php

Al igual que la curva del dragón, este fractal también puede generarse a partir de segmentos del mismo tamaño conectados en ángulos de 60 grados. Si representamos la curva como una lista de ángulos hacia la derecha o hacia la izquierda, la curva más simple sería:

[D, D]

Para generar la siguiente iteración de la curva, se van a insertar parejas de ángulos antes, entre y después de los elementos de la lista anterior. Las parejas de ángulos se alternan hacia la derecha y hacia la izquierda. El orden empieza en la dirección contraria al que se encuentra de primero en la lista anterior. En este caso, se empieza con dos ángulos hacia la izquierda porque el primer ángulo de la iteración pasada es hacia la derecha (en negrita los ángulos de la iteración anterior).

[I, I, **D**, D, D, **D**, I, I]

Si se aplica la misma regla sobre la lista anterior, se obtiene la siguiente iteración del fractal. Dado que el primer elemento de la iteración anterior es un ángulo hacia la izquierda, se empieza hacia la derecha.

[D, D, **I**, I, I, **I**, D, D, **D**, I, I, **D**, D, D, **D**, I, I, **D**, D, D, **I**, I, I, **I**, D, D]

Entre mayor sea la cantidad de iteraciones, más cercano va ser el resultado al triángulo de Sierpinski.

Curva de Lévy C

Este fractal se construye con segmentos del mismo tamaño y ángulos de 45 grados hacia la izquierda o la derecha, pero pueden haber segmentos contiguos sin un ángulo entre ellos. La regla básica es sustituir un segmento por dos segmentos unidos en 90 grados que forman un triángulo recto con el segmento anterior, con éste como hipotenusa.

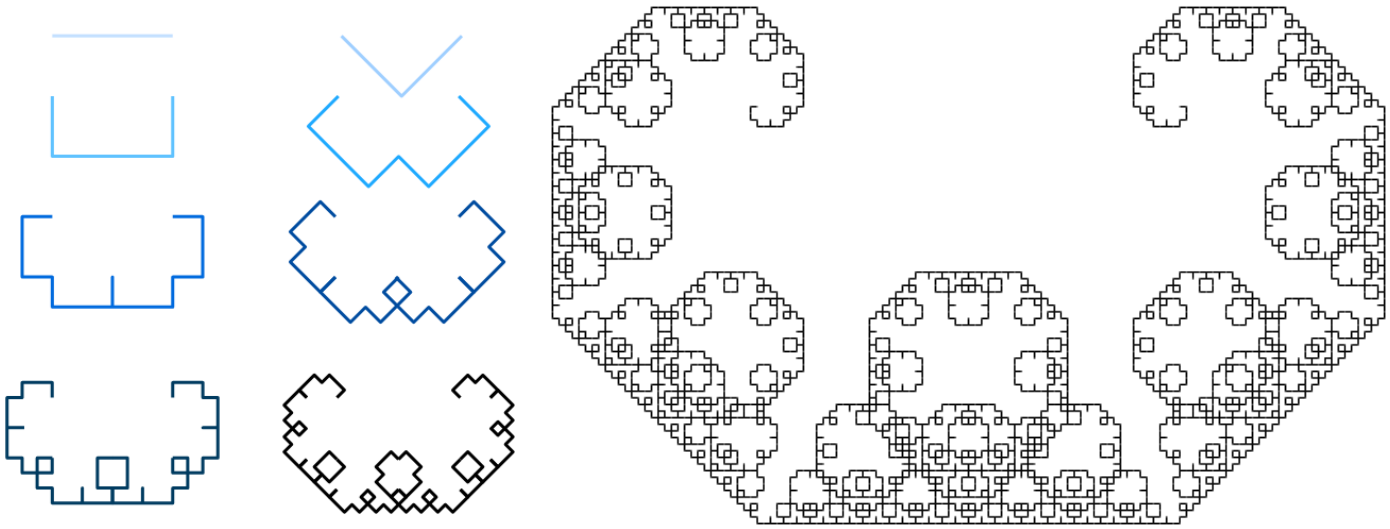


Ilustración 4 https://en.wikipedia.org/wiki/L%C3%A9vy_C_curve

Este fractal también puede representarse por medio de una lista, con la diferencia de que es necesario almacenar los ángulos y también los segmentos, ya que debido a la regla pueden quedar varios ángulos contiguos sin dibujar algún segmento. Si utilizamos la letra I para indicar ángulo de 45 grados hacia la izquierda, D para ángulo de 45 hacia la derecha y A para indicar el avance o la presencia de un segmento, entonces la única regla que se necesita es iniciar con la lista [A] y aplicar sucesivamente la sustitución de las ocurrencias de A por la lista [D, A, I, I, A, D].

Como se mencionó, se iniciaría con la lista que contiene sólo un segmento:

[A]

La segunda iteración consiste en sustituir las ocurrencias de A en la lista por los elementos [D, A, I, I, A, D], lo que nos da como resultado

[D, A, I, I, A, D]

Si se vuelve a aplicar la misma regla para generar la siguiente iteración, se obtiene el siguiente resultado:

[D, D, A, I, I, A, D, I, I, D, A, I, I, A, D, D]

Y en una tercera iteración:

[D, D, D, A, I, I, A, D, I, I, D, A, I, I, A, D, D, I, I, D, D, A, I, I, A, D, I, I, D, A, I, I, A, D, D, D]

Software a desarrollar

Su trabajo consiste en implementar un programa de C++ que utilice las estructuras lineales creadas en el curso para generar las listas con los "movimientos" necesarios de cada fractal y que interprete las mismas para lograr una representación en pantalla de los fractales.

El programa debe solicitar al usuario cuál de los fractales quiere representar en pantalla y proceder a la graficación. Finalizada la representación del fractal elegido en pantalla, el programa espera por una tecla del usuario y regresa a las opciones iniciales donde puede elegir algún otro fractal o salir.

Los problemas principales son los siguientes:

1. Creación de una estructura lineal tipo lista que permita ser recorrida y realizar inserciones y borrados.
2. Generación de las listas de ángulos y segmentos para cada uno de los fractales.
3. Creación de una clase que se encargue de interpretar las listas generadas en el punto anterior y representarlas en pantalla.

Este proyecto cuenta con un grado de libertad para la representación gráfica de los fractales, siéntase libre de experimentar con diferentes colores para los segmentos, fondos de pantalla y cualquier otra mejora gráfica que considere interesante.

Recomendaciones para la implementación

Es importante abstraer las principales funcionalidades que se necesitan en el proyecto, para de esta forma separarlas y poder reutilizarlas. A continuación se muestran algunas funcionalidades que podrían abstraerse:

- Método que recibe la cantidad de iteraciones y produce una lista con las instrucciones para dibujar la curva del dragón
- Método que recibe la cantidad de iteraciones y produce una lista con las instrucciones para dibujar la curva de punta de flecha de Sierpinski
- Método que recibe la cantidad de iteraciones y produce una lista con las instrucciones para dibujar la curva de Lévy C.
- Método que reciba una lista con instrucciones y las dibuje en pantalla.

La estrategia para interpretar las listas de instrucciones y dibujarlas en pantalla puede variar, pero una forma válida es crear una clase con un comportamiento similar a la tortuga de turtle graphics, la cual cuenta con una posición actual y una dirección. Al interpretar una instrucción como A (avanzar), entonces se calcula el punto destino donde debe ubicarse después de avanzar y se dibuja una línea entre el punto anterior y el actual.

Si se va a tener sólo una clase que se encargue de interpretar las instrucciones y dibujar en pantalla el resultado, entonces es necesario estandarizar las instrucciones que pueden venir en las listas. La curva del dragón se dibuja con ángulos de 90 grados, la de Sierpinski con ángulos de 60 y la de Lévy C con ángulos de 45. En la curva del dragón y la de Sierpinski no es necesario almacenar cuándo hay un segmento, pues se asume que hay uno antes de cada ángulo, mientras que en la de Lévy C sí es necesario guardar los segmentos pues no siempre hay segmentos entre los diferentes ángulos.

Entonces, una posibilidad es siempre representar la presencia de segmentos (en este ejemplo con la letra A) y los ángulos representarlos con números enteros negativos para ángulos a la izquierda y positivos para ángulos a la derecha.

De esta forma algunas iteraciones de cada uno de los fractales se podrían representar así:

3era iteración dragón: [A, 90, A, 90, A, -90, A, 90, A, 90, A, -90, A, 90, A]

2da iteración Sierpinski: [A, -30, A, -30, A, 30, A, 30, A, 30, A, 30, A, -30, A, -30, A]

3era iteración Lévy C: [45, 45, A, -45, -45, A, 45, -45, -45, 45, A, -45, -45, A, 45, 45]

Al estandarizar la representación de los tres fractales, sólo es necesario programar un método que interprete y represente los fractales en pantalla en vez de tres diferentes.

Documentación

La documentación interna del código debe ser mínima, puede limitarse únicamente a una descripción breve del objetivo de cada clase y método dentro del proyecto. Evite hacer comentarios excesivos. Escriba código claro y conciso, trate de apegarse a los principios de código limpio para el código que escriba. Recuerde que es mucho más importante que su código sea claro y fácil de entender que si es muy eficiente.

En cuanto a la documentación externa, debe entregarse dos documentos en formato PDF:

1. Manual de usuario: sección que explica detalladamente a cualquier usuario cómo ejecutar y utilizar la aplicación. Debe describir el proceso de instalación de cualquier librería que utilice en su proyecto. Debe tener portada.
2. Documentación del proyecto
    a. Portada
    b. Resumen ejecutivo. Descripción breve de todo lo que abarca la documentación. El objetivo de este resumen es captar la atención del lector y motivarlo a aprender más sobre el proyecto. Menos de una página.
    c. Introducción. ¿Por qué se hace el proyecto y qué se incluye? (1-2 págs.)
    d. Presentación y análisis del problema (5+ págs.)
        i. Qué es lo que hay que resolver. Identificar pequeños problemas que deben resolverse en el proyecto.
        ii. Cómo se va resolver el problema. La forma en que se planea resolver el problema.
        iii. Análisis crítico de la implementación. Luego de la implementación, decir qué se logró implementar, lo que faltó y qué cosas se podrían mejorar de lo que se implementó. No sólo mencionar, si no explicar por qué.
    e. Conclusiones: resoluciones puntuales tras el proyecto. Estas deben ser relacionadas con los aspectos técnicos del trabajo únicamente. Si no tiene claro cómo escribir conclusiones para un trabajo académico, consulte al profesor o asistentes. (1-2 págs.)
    f. Recomendaciones: consejos o advertencias que se derivan de las conclusiones. Lecciones aprendidas durante el desarrollo del proyecto. Recomendaciones para personas que tengan que hacer el mismo trabajo. También deben estar orientadas con aspectos técnicos de la tarea programada. Se recomienda hacer una o más recomendaciones por cada conclusión. (1-2 págs.)
    g. Referencias. Deben incluirse en formato APA.

Forma de trabajo

El proyecto se desarrollará en tríos o parejas. No se permitirá la entrega de proyectos desarrollados de forma individual a menos que exista autorización previa del profesor.

En la medida de lo posible, presente al menos un adelanto de trabajo al profesor o asistentes del curso durante la primera semana y media del proyecto. Esta supervisión le puede ayudar a mejorar el resultado final del proyecto. No dude en consultar cualquier asunto tanto de programación como de elaboración de la documentación.

Los miembros del grupo deben distribuir el trabajo uniformemente, cualquier situación en la que un miembro no esté realizando su parte del trabajo debe ser notificada al profesor inmediatamente y dicha persona puede ser separada del grupo para realizar el trabajo de forma individual.

Evaluación

La tarea tiene un valor de 20% de la nota final, en el rubro de Proyectos Programados.

Desglose de la evaluación de la tarea programada:

Documentación:	20%
Programación:	80%

Recomendaciones adicionales

Pruebe cada funcionalidad individualmente. No implemente grandes secciones del programa sin verificar el funcionamiento por separado de cada una de sus partes. Esto dirige a errores que son más difíciles de encontrar.

Recuerde que el trabajo es en equipos, es indispensable la comunicación y la coordinación entre los miembros del subgrupo.

Comparta el conocimiento con los demás compañeros de grupo y de la carrera, la ciencia de la computación es una disciplina que requiere el traspaso libre de conocimientos. Se logran mejores resultados con la colaboración de todos que con el esfuerzo separado de diferentes personas.

No dude en consultar diferentes fuentes para satisfacer las dudas. Aparte de las búsquedas en internet, asegúrese de exponer sus dudas a sus compañeros, profesor y conocidos que estudien la carrera; en la mayoría de las ocasiones es más provechosa conversación de 10 minutos entre personas que están trabajando en lo mismo que pasar horas buscando la respuesta a una duda de forma individual.

No deje la documentación para el final, es buena práctica ir desarrollándola durante todo el transcurso del proyecto. Recuerde que la documentación debe ser concisa y puntual, por lo que en realidad no toma mucho tiempo al realizarla de esta forma.

Plagios no serán tolerados bajo ninguna circunstancia. Cualquier intento de fraude será evaluado con una nota de cero y se enviará una carta al expediente del estudiante. Siempre escriba su propio código.

Se recomienda la utilización de la biblioteca WinBGIm para la elaboración de la interfaz, debido a su simplicidad de uso. Si se va a utilizar otro tipo de biblioteca debe comunicarlo al profesor y proveer todos los archivos necesarios para su ejecución.

Referencias

Lévy C curve. (2014, August 8). In *Wikipedia, The Free Encyclopedia*. Retrieved 17:27, August 26, 2014, from http://en.wikipedia.org/w/index.php?title=L%C3%A9vy_C_curve&oldid=620432806

Dragon curve. (2014, March 9). In *Wikipedia, The Free Encyclopedia*. Retrieved 17:27, August 26, 2014, from http://en.wikipedia.org/w/index.php?title=Dragon_curve&oldid=598813375

Sierpiński arrowhead curve. (2014, May 5). *In Wikipedia, The Free Encyclopedia*. Retrieved 17:28, August 26, 2014, from http://en.wikipedia.org/w/index.php?title=Sierpi%C5%84ski_arrowhead_curve&oldid=607167543