Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

IC-1801 Taller de Programación

Prof. Mauricio Avilés

Proyecto Programado 2 - Fractales

Motivación

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 recursivos. El objetivo de este proyecto programado es pasar del mundo de la matemática abstracta hacia la estética visual a través de la representación en pantalla de diferentes fractales.

Software a desarrollar

Su trabajo consiste en implementar funciones recursivas y una iterativa para generar gráficos por computadora de los siguientes fractales:

1. Hexágono de Sierpinski
2. Círculos de Ravel
3. Curva de Satie
4. Estrella de Debussy
5. Curva del Dragón
6. Fractal de Chopin

El trabajo no consiste solamente en programar los fractales tal y como se muestran en este documento. Cada fractal debe hacer uso de colores, figuras rellenas y cualquier otra variación que se desee con el objetivo de generar una presentación única y creativa del fractal. Se espera un grado de creatividad de nivel universitario, utilizar simplemente colores elegidos de forma aleatoria no cuenta como trabajo creativo. La creatividad será evaluada.

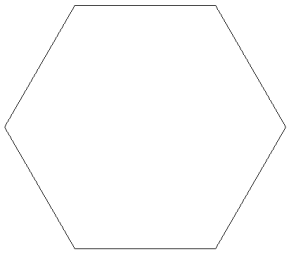
Algunos de estos fractales son muy conocidos y es fácil encontrar código ya implementado en Internet. **NO COPIE** código de ningún lugar. Cualquier copia que se detecte en el código provocará que sea calificado con nota de cero y se reporte el caso a la dirección de la Escuela de Computación.

Hexágono de Sierpinski

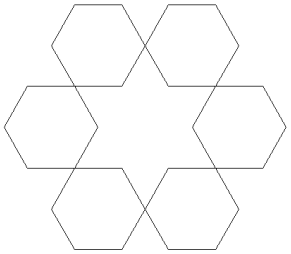
Este fractal sigue la misma idea de los dos fractales anteriores, pero sustituyendo la figura geométrica base, que en este caso es un hexágono. Si la profundidad es 1, se dibuja un hexágono corriente, si no, se dibuja un hexágono (recursivamente) de un tercio del tamaño y se mueve la posición al siguiente vértice del hexágono. Nótese que en este fractal no existe una línea que una los vértices de los hexágonos grandes, si no que queda un espacio vacío.

Debe crear una función recursiva que dibuje el fractal, que utilice como mínimo parámetros para profundidad y tamaño del lado. Puede invocarse como **hexagonoSierpinski(produnfidad, lado)**. A continuación, se muestran algunos ejemplos del resultado que debe mostrar la función utilizando diferentes profundidades.

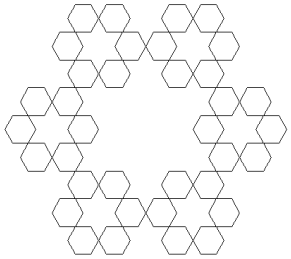
Profundidad=1



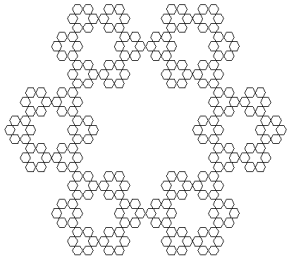
Profundidad=2



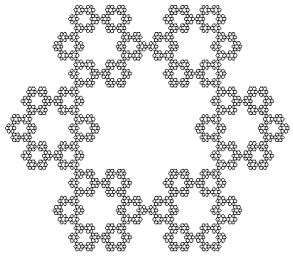
Profundidad=3



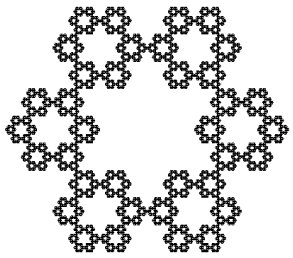
Profundidad=4



Profundidad=5



Profundidad=6

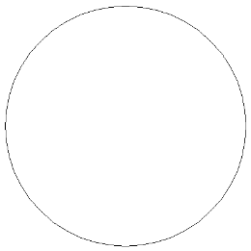


Círculo de Ravel

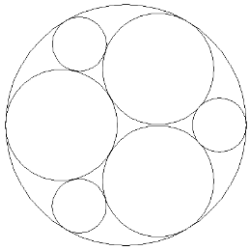
Este fractal consiste de un círculo que contiene seis círculos dentro. Tres de los círculos son más grandes y ocupan el mayor espacio posible dentro del círculo mayor. Los otros tres círculos se ubican en los espacios vacíos entre los círculos grandes, tal y como se ve en la figura con profundidad 2. Este patrón se repite recursivamente en cada uno de los círculos.

Debe crear una función recursiva que dibuje el fractal, que utilice como mínimo parámetros para profundidad y tamaño del radio. Puede invocarse como **ravel(produnfidad, radio)**. A continuación, se muestran algunos ejemplos del resultado que debe mostrar la función utilizando diferentes profundidades. Para este fractal debe investigar el funcionamiento de la función **circle** en Turtle.

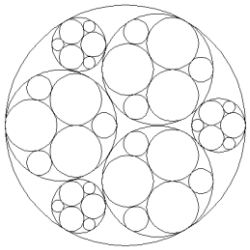
Profundidad=1



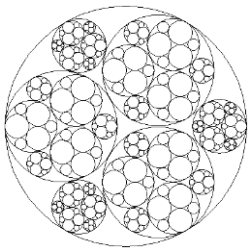
Profundidad=2



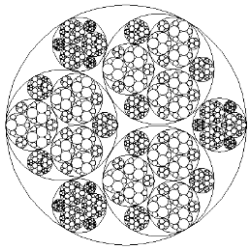
Profundidad=3



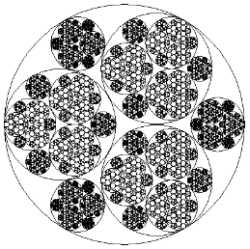
Profundidad=4



Profundidad=5



Profundidad=6



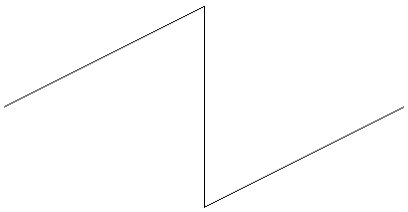
Note como el patrón que forman los círculos grandes se asemeja al patrón producido por el triángulo de Sierpinski.

Curva de Satie

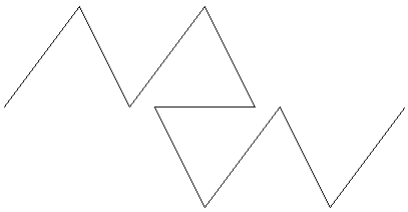
Este es un fractal que sigue una idea similar a la curva de Koch, pero utiliza una figura diferente para dividir cada línea. La base es una línea recta, la cual se divide en 3 líneas, tal y como se muestra en el dibujo que muestra la profundidad=1. La línea vertical es de la mitad del tamaño que separa el inicio y el final. Las otras dos líneas tienen un tamaño que se calcula usando el teorema de Pitágoras. Cada una de estas líneas se sustituye por el mismo patrón en zigzag, dando origen al fractal.

Debe crear una función recursiva que dibuje el fractal, que utilice como mínimo parámetros para profundidad y tamaño del lado. Puede invocarse como **satie(produnfidad, lado)**. A continuación, se muestran algunos ejemplos del resultado que debe mostrar la función utilizando diferentes profundidades.

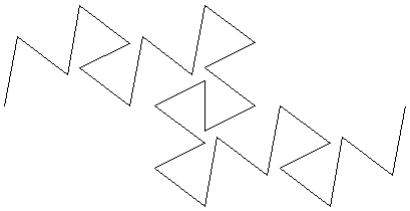
Profundidad=1



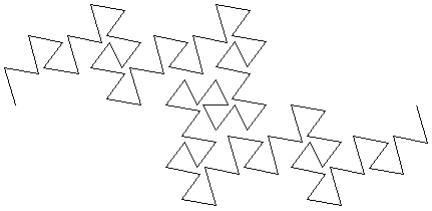
Profundidad=2



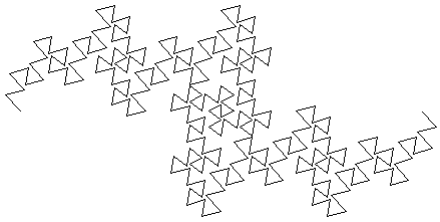
Profundidad=3



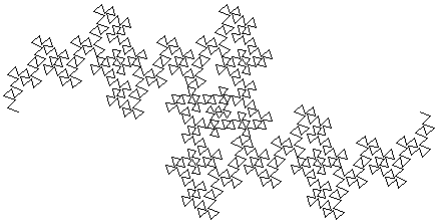
Profundidad=4



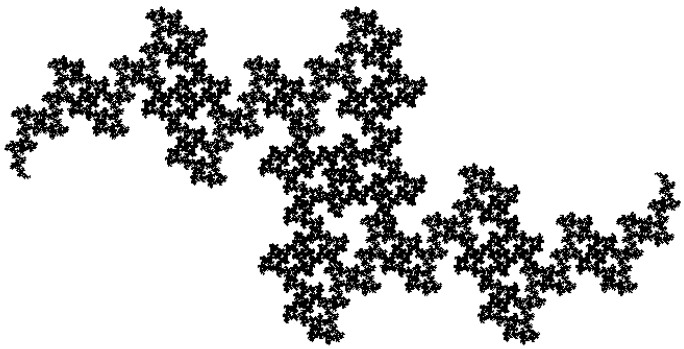
Profundidad=5



Profundidad=6



Con mayor cantidad de iteraciones el fractal toma una forma similar a la siguiente:

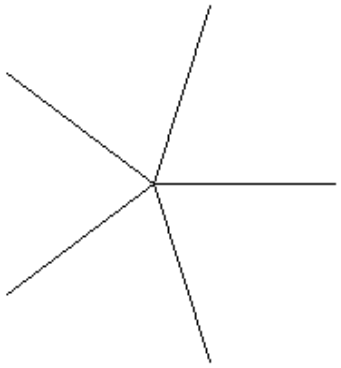


Estrella de Debussy

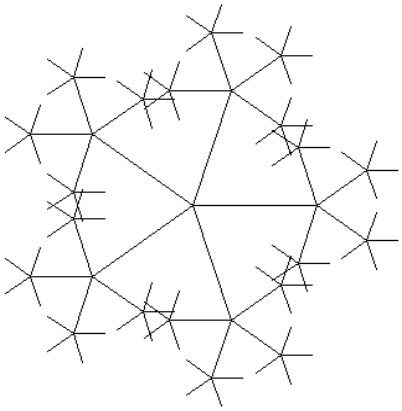
Este fractal es más generalizado que los demás. Consiste en dibujar líneas que formen una estrella o asterisco. La cantidad de picos de la estrella está dada por un parámetro n. Si n es igual a 4, entonces la estrella tiene cuatro picos y cada uno de ellos tendrá estrellas del mismo tipo. Las estrellas "hijas" tendrán un pico menos que la estrella principal, esto es porque cada una nace de un pico de la estrella principal y éste cuenta como uno de sus picos. También requiere otro parámetro que es la relación de tamaño entre una estrella y sus estrellas "hijas". Si la relación es 0.5, entonces las estrellas hijas de una estrella van a ser de la mitad del tamaño que la estrella anterior.

Debe crear una función recursiva que dibuje el fractal, que utilice parámetros para la profundidad, el tamaño del pico, la cantidad de picos y la relación entre el tamaño de la estrella y sus estrellas hijas. Puede invocarse como **debussy(profundidad, lado, n, relacion)**. A continuación, se muestran algunos ejemplos con diferentes valores de parámetros.

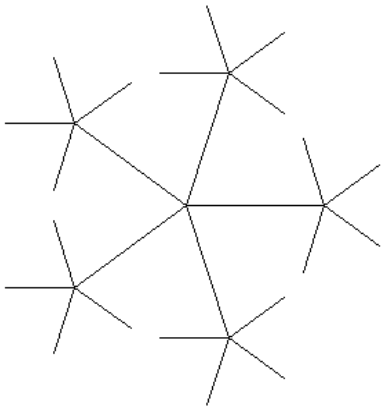
Uno de 5 picos con profundidad 1: debussy(1, 100, 5, 0.5)



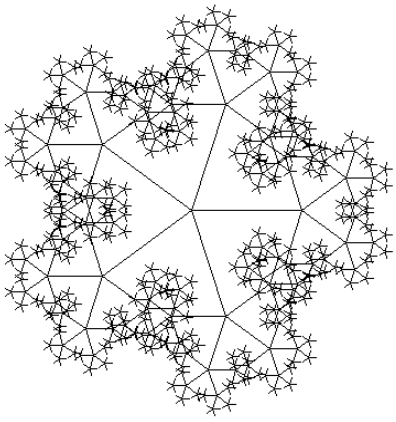
Aumentando profundidad: debussy(3, 100, 5, 0.5)



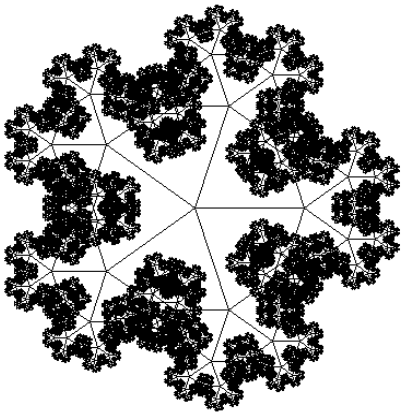
Aumentando profundidad: debussy(2, 100, 5, 0.5)



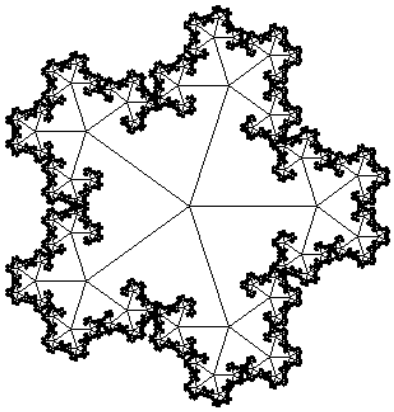
Aumentando profundidad: debussy(5, 100, 5, 0.5)



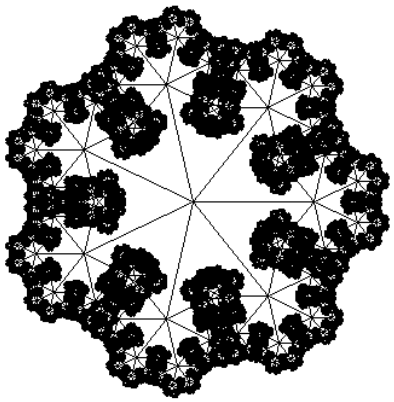
Aumentando profundidad: debussy(8, 100, 5, 0.5)



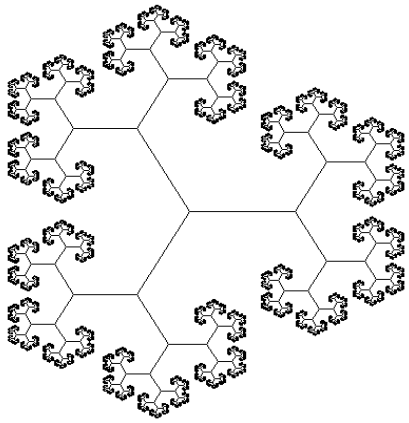
Mismo que el anterior, pero cambiando la relación a 0.4. Note como los picos de las estrellas se empequeñecen más rápido: debussy(8, 100, 5, 0.4)



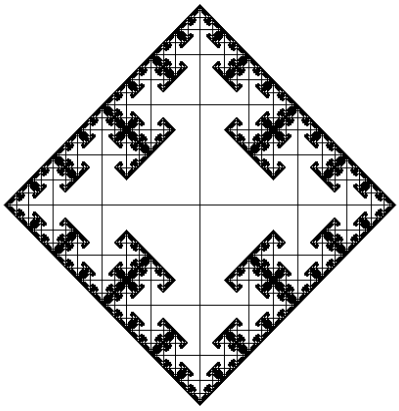
Cambiando la cantidad de picos a 7: debussy(7, 100, 7, 0.4)



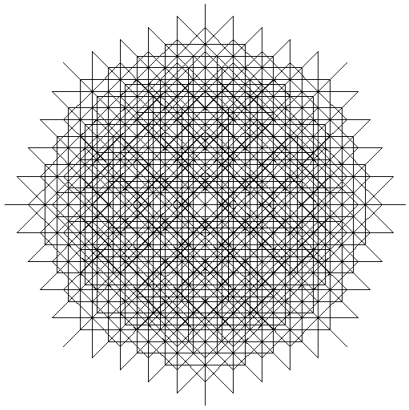
Con un n menor se puede tener mayor profundidad sin afectar el desempeño: debussy(14, 100, 3, 0.6)



Con cuatro ramas: debussy(9, 100, 4, 0.5)

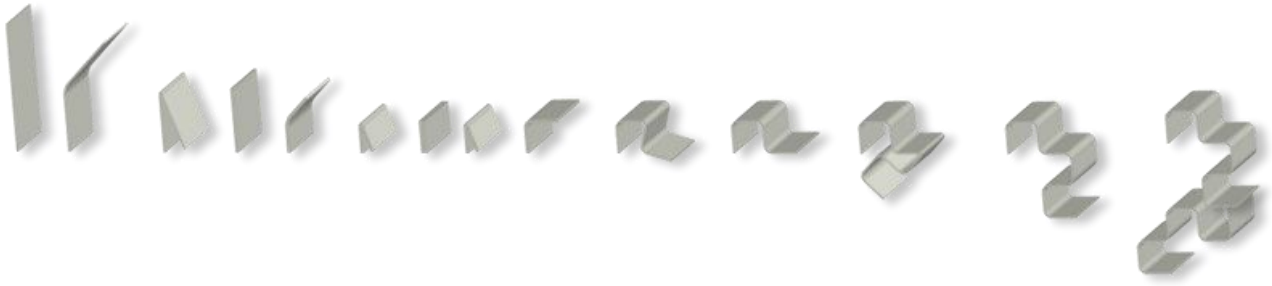


La relación no tiene porqué ser menor que uno, puede ser incluso mayor: debussy(5, 50, 8, 1)



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.



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]

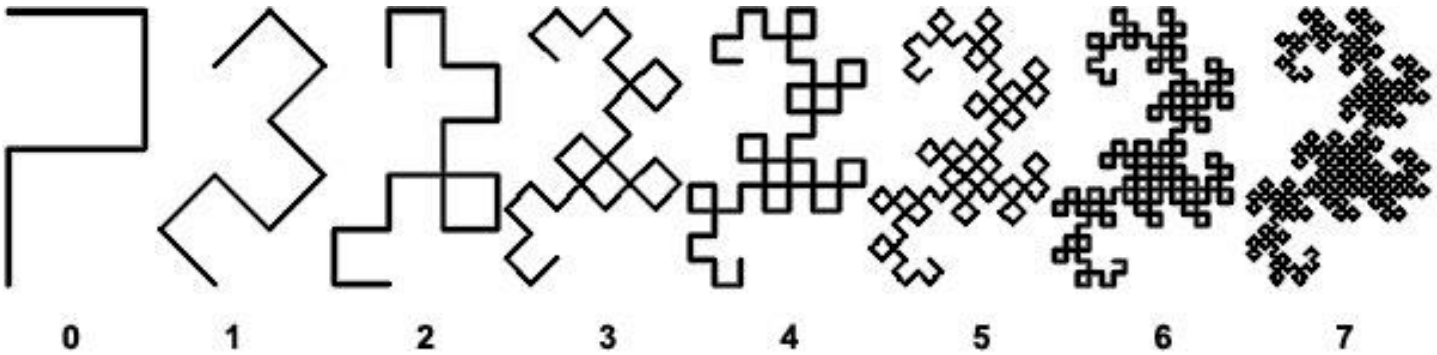
Siguiendo este mismo patrón, la siguiente iteración es (en negrita los elementos de la iteración anterior):

[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.



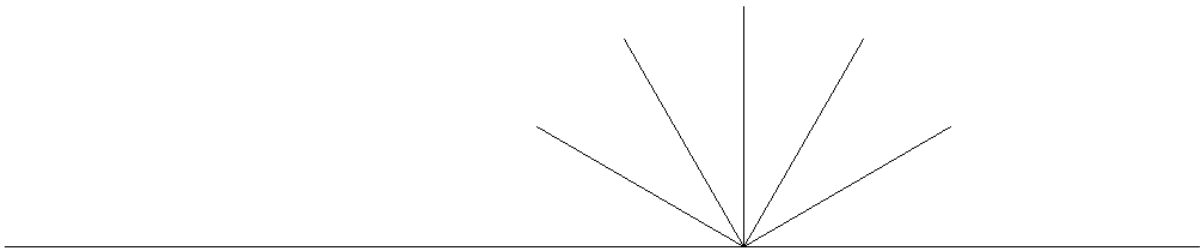
Debe crear una función que dibuje el fractal, que utilice como mínimo parámetros para profundidad y tamaño del lado. Puede invocarse como **dragon(produnfidad, lado)**. Esta función puede programarse usando iteración. Puede dividir el problema en dos partes, una función que se encargue de generar la lista con las instrucciones a ejecutar, y otra función que reciba una lista de instrucciones como parámetro y la dibuje en pantalla.

Fractal de Chopin

Este fractal utiliza valores obtenidos de la secuencia de Fibonacci para definir la cantidad de repeticiones y la distribución de tamaños. Primero, es importante aclarar el concepto de proporción aurea, ya que se utiliza su valor para generar el fractal. La proporción áurea es el valor al que se aproxima la secuencia de Fibonacci cuando se divide un valor de la secuencia entre el valor anterior, es decir: Fib(n) / Fib(n-1). Cuando n es grande, esta proporción converge a un número irracional, aproximadamente 1.61803398875. El número específico que vamos a utilizar es el inverso de dicha proporción, que es 0.61803398875, aproximadamente. Queda a criterio suyo investigar un poco más sobre la proporción áurea, es un tema interesante y un valor muy utilizado en arquitectura, arte y otras disciplinas.

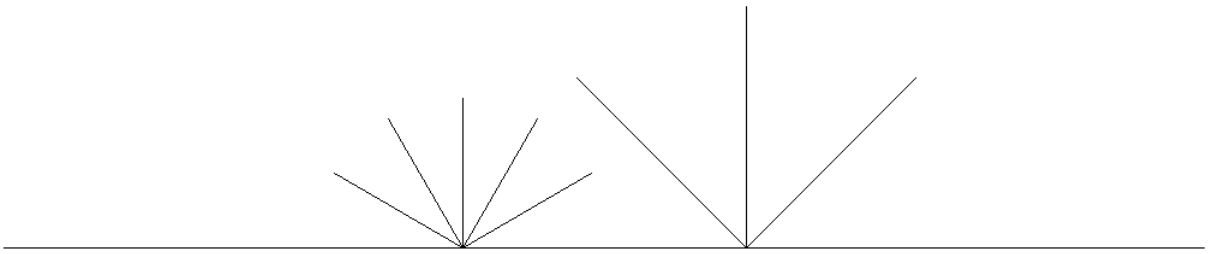
Este fractal consiste en dibujar una línea, digamos de largo 1000. Esta línea va a estar dividida en dos segmentos, la primera mide 1000 * 0.61803398875, la segunda parte mide 1000 * (1 - 0.61803398875). En el punto donde se unen los dos segmentos, se va a dibujar una serie de líneas radiales que repiten el mismo patrón con 1/5 del largo anterior. La cantidad de líneas que se dibujen es escogida aleatoriamente entre los números [2, 3, 5, 8]. El caso base de este fractal no es con profundidad, si no cuando el largo ha alcanzado un mínimo.

El siguiente ejemplo muestra el fractal con un largo de 1000 y un mínimo de 1000.



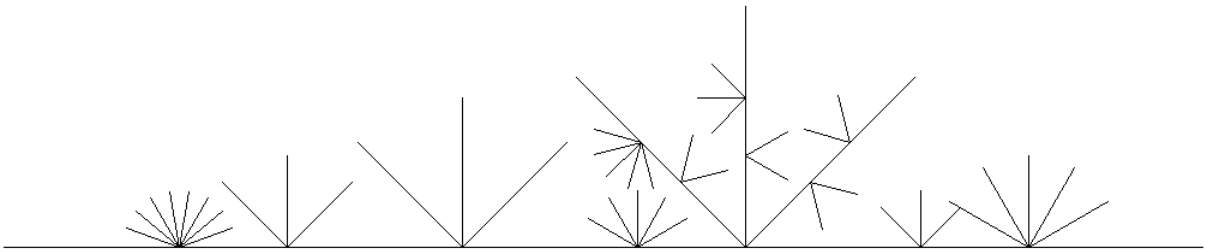
Se dibuja el segmento de la izquierda que mide 1000 * 0.61803398875. En este punto se escoge un número aleatoriamente entre [2, 3, 5, 8]. El que salió fue 5, por lo que se dibujan 5 ramas o rayos, separados equitativamente entre ellos. El largo de los rayos son 1/5 del largo anterior, es decir, 200. Dado que el largo de estas líneas es inferior al mínimo (1000), no repiten el patrón del fractal. Finalmente, se dibuja el segmento de la derecha que mide 1000 * (1 - 0.6180339887).

El siguiente ejemplo muestra el fractal con un largo de 1000 y un mínimo de 500.



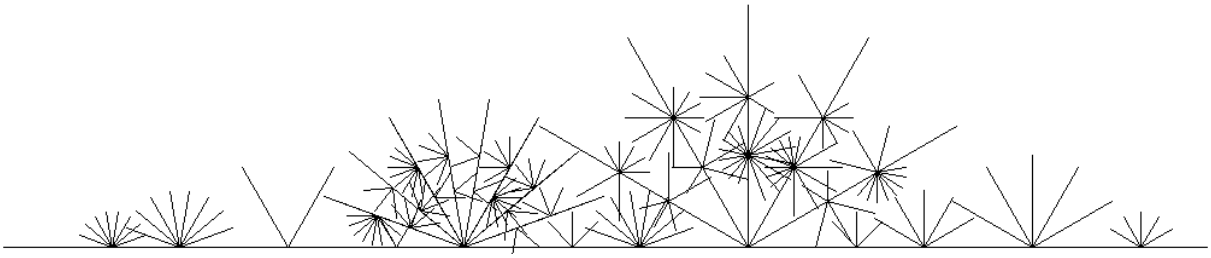
La estructura es muy similar a la del fractal anterior, pero esta vez el número elegido para los rayos que dividen la línea principal es 3. Observe que el primer segmento ahora repite el mismo patrón, esto porque el largo de dicho segmento (618.03398875) no es menor que 500. El segmento de la derecha no repite el patrón del fractal porque su largo es menor que 500.

Cuando los rayos tienen una longitud mayor que el mínimo, comienzan a repetir el patrón dentro de ellos mismos. El siguiente ejemplo muestra el fractal con un largo de 1000 y un mínimo de 200.



Observe cómo los rayos que están sobre el mínimo repiten el fractal en dos direcciones. También los segmentos horizontales que se encuentran sobre el mínimo repiten el patrón con una cantidad de rayos escogida entre los números [2, 3, 5, 8].

El siguiente ejemplo muestra el fractal con un largo de 1000 y un mínimo de 100.



Fractal con un largo de 1000 y un mínimo de 10.

Fractal con un largo de 1000 y un mínimo de 2.

Debe crear una función recursiva que dibuje el fractal, que utilice como mínimo parámetros el tamaño del lado y el mínimo. Puede invocarse como **chopin(largo, minimo)**.

Documentación

Toda función debe llevar como documentación interna comentarios con lo siguiente:

- Descripción de la función
- Entradas
- Salidas
- Restricciones

En cuanto a la documentación externa, debe entregarse un manual de usuario en formato PDF que explique a cualquier persona cómo utilizar las funciones en el *shell* de Python.

Forma de trabajo

Debido a la cantidad de trabajo involucrada, el proyecto se desarrollará en parejas.

Para hacer dibujos en pantalla se utilizará el paquete **turtle** que viene incluido con el lenguaje Python. Este paquete es una implementación de la geometría de la tortuga, concepto matemático desarrollado por Seymour Papert y Wally Feurzig durante los años 60.

Para importar el módulo se utiliza la sentencia: **import turtle**

Algunos métodos que pueden ser de provecho para la tarea son:

turtle.forward(numpixels)	turtle.backward(numpixels)
turtle.right(degrees)	turtle.left(degrees)
turtle.circle(radius)	turtle.reset()
turtle.pencolor(r,g,b)	turtle.penup()
turtle.pendown()	

Puede encontrar más información sobre **turtle** en http://docs.python.org/release/3.1.3/library/turtle.html

Entrega

El tiempo asignado para la tarea programada es de 2 semanas. Debe entregarse un archivo **archivo ZIP** que contenga dos cosas:

1. **Un solo archivo Python** con todas las funciones requeridas.
2. Un archivo PDF con el manual de usuario

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%

Como se mencionó anteriormente, se tomará en cuenta la creatividad para la representación de los fractales. La utilización de diferentes colores, fondos de pantalla o cualquier otro elemento que le dé un matiz artístico a su trabajo, será considerado para la nota de la programada.

Recomendaciones adicionales

Pruebe cada función individualmente. No implemente todo el programa sin verificar el funcionamiento por separado de cada una de sus funciones. 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.

Referencias

Fractal. (2013, May 8). In *Wikipedia, The Free Encyclopedia*. Retrieved 02:28, May 10, 2013, from http://en.wikipedia.org/w/index.php?title=Fractal&oldid=554087975

Sierpinski triangle. (2013, March 4). In *Wikipedia, The Free Encyclopedia*. Retrieved 02:28, May 10, 2013, from http://en.wikipedia.org/w/index.php?title=Sierpinski_triangle&oldid=542044665

Sierpinski carpet. (2013, February 22). In *Wikipedia, The Free Encyclopedia*. Retrieved 02:28, May 10, 2013, from http://en.wikipedia.org/w/index.php?title=Sierpinski_carpet&oldid=539780666

N-flake. (2015, October 24). In *Wikipedia, The Free Encyclopedia*. Retrieved 22:41, May 1, 2016, from https://en.wikipedia.org/w/index.php?title=N-flake&oldid=687324189

Koch snowflake. (2016, March 31). In *Wikipedia, The Free Encyclopedia*. Retrieved 22:41, May 1, 2016, from https://en.wikipedia.org/w/index.php?title=Koch_snowflake&oldid=712767769