Documentación
FractGraph
V. 1.0.

Ramírez Ortega Angelo
2017080055
D'Ambrosio Soza Guilliano
2017158561

Escuela de Ingeniería en Computación
Estructuras de Datos
Profesor: Jose Dolores Navas Su
Instituto Tecnológico de Costa Rica
Jueves 22 de septiembre del 2017

Indice

Resumen Ejecutivo

A continuación se abarcaran todos los temas relacionados con el programa FractGraph. Se propuso encontrar una manera de graficar fractales, los cuales son figuras que se crean con un patrón predeterminado, sin importar la cantidad de iteraciones en las que se realice se obtendrá la misma figura con la diferencia que entre más iteraciones se define mejor la figura. Se desarrolló en el ambiente de programación codeblocks con la librería wxWidgets. La razón principal por la cual se realizó este proyecto fue con fines estudiantiles, para ayudar a la comprensión de interfaz grafica en C++ y la implementación de listas con iteraciones y poder obtener puntos de rectas a partir de diferencias de ángulos entre rectas. La solución implementada se explica más adelante. Se logró con éxito cada aspecto requerido para esta programación y además se logró implementar ciertas partes adicionales que se explicaran a fondo a continuación. Las recomendaciones generales son investigar lo necesario para implementar interfaz gráfica y cómo graficar líneas con tiempo para sufrir inconvenientes con la fecha de entrega. Además practicar el código limpio y evitar el desperdicio de recursos valiosos como la memoria.

Introducción

Este proyecto nace a raíz de un proyecto programado con la necesidad de graficar fractales por medio de una interfaz gráfica. Un fractal es una figura cuya estructura se repite a diferentes escalas dependiendo de la cantidad de iteraciones que se le apliquen al crearla. Los fractales fueron propuesto por Benoît Mandelbrot en 1975, su palabra proviene del latín *fractus* que significa fracturado. Los fractales tienen una forma irregular, lo cual los hace imposibles de ser descritos en términos geométricos tradicionales, su forma está compuesta por copias más pequeñas de su misma figura. Debido a esto último los fractales pueden ser creados con

recursividad o iteraciones. En este proyecto se fomenta la autodidaxia ya que había mucha investigación por hacer sobre temas que no fueron vistos en clase. También el proyecto busca fomentar el trabajo en equipo de los participantes. Se incluyen temas como recursividad, iteraciones, listas, interfaz gráfica y cálculo de puntos en un plano. Lo más importante de esto fue el cálculo de puntos y la interfaz gráfica. El cálculo de puntos fue primordial ya que cada fractal se genera de diferente manera, por lo que se tenía que calcular los puntos con rotaciones de 45, 90 y 60 grados. La interfaz fue necesaria ya que una vez calculados los puntos de las rectas, estas se tenían que dibujar.

Presentacion y Analisis del Problema

Problema

El problema principal de del proyecto es cómo graficar los fractales, ya que las listas y la manera en la que se generan ya vienen dadas en el enunciado del programa. A continuación se examinan situaciones más específicas.

- **Cómo dibujar líneas en interfaz gráfica:**

Aprender a dibujar líneas con una librería de interfaz de CodeBlocks es un gran desafío ya que la información no es tan amplia al respecto y ademas es dificil de encontrar. El dibujo de líneas aunque parece fácil, conllevan la creación de diferentes clases y además implementacion de metodos de la librería utilizada para la interfaz que son bastante propensos a producir errores, por lo que se necesita una depuración de errores amplia para que se dibujen las líneas de manera correcta.Para esto se tiene que encontrar la librería, descargara al ordenador y además importarla al ambiente de programación, lo que a su vez lleva un proceso ya que se debe instalar por medio de la terminal y hacer que el compilador de C++ lo pueda interpretar de manera correcta para poder así compilar y correr el proyecto representa todo un desafío.

- Cálculo de puntos en un plano:

Para poder graficar los fractales además de las listas, se necesita un método para poder interpretarlas e imprimirlas en forma de líneas en un plano(ventana). Este método era primordial ya que sin él no se podrían graficar los fractales. El problema principal era cómo obtener los puntos de las rectas que se iban a dibujar con ángulos de 45,90 y 60 grados. Esto es una mayor complicación ya que dependiendo de la dirección la recta su rotación va a generar números negativos, positivos o puede generar un mal cálculo por lo cual no hay una fórmula general que aplica a todas las rectas. Por lo tanto se deben hacer muchas validaciones para que se calcule de manera correcta la longitud del segmento y la dirección de tal segmento.

- Encontrar la librería correcta:

Una parte muy importante de todo el proyecto es escoger una librería o librerías que sean útiles y eficaces para los propósitos que se tienen. Si no se escoge la librería más óptima entonces el proyecto puede alargar en duración y volverse más complicado. Una buena librería facilita todo el proceso de graficación e interfaz con el usuario además de facilitar las operaciones necesarias para lograr el fractal.

- Creación de listas con iteraciones:

Para poder usar una lista que tuviera un tamaño máximo definido por la cantidad de elementos en ellas no se podía usar una simple lista. Este es uno de los grandes problemas ya que los ArrayList no sirven en este caso ya que dependiendo del fractal y de las iteraciones el tamaño requerido de la lista puede ser mayor que el tamaño máximo definido por el programador y esto generaría problemas, en el caso de poner un límite de tamaño muy grande entonces tambien se tendrian problemas de malgasto de recursos de memoria. Por lo tanto se necesita implementar un tipo diferente de dato.

- Interfaz con el usuario:

Una parte esencial de cualquier programa o aplicación es la interfaz con el usuario ya que este es el que va a hacer uso del proyecto que se desarrolle. Por lo tanto se

tiene que tener mucho cuidado al desarrollar interfaz. Debe procurarse que esta sea intuitiva al usuario, obviando la existencia del manual de usuario, el usuario prefiere aplicaciones que le sean fácil de usar. La interfaz debe ser simple y atractiva a la vista, las interfaces muy cargadas de elementos o muy complejas tienden a ser cansadas a la vista lo que provoca que el usuario no se sienta cómodo. Además se debe tener un consenso de cómo se dará implementación a esta, por ejemplo, si el usuario escoge las iteraciones, si el usuario escribe el fractal a dibujar o presiona un botón con el nombre del fractal, también si la interfaz se despliega en una ventana o en varias a la vez.

Solución al Problema

Las soluciones se dieron a cada problema específico por lo tanto las soluciones a los problemas planteados anteriormente se dan a continuación:

- Cómo dibujar líneas en interfaz gráfica:

Para poder implementar el uso de líneas en una interfaz gráfica se tuvieron que hacer varias cosas. Primero se creó una ventana en la cual se iba a desplegar el dibujo del fractal linea por linea. En la librería utilizada para poder dibujar líneas se usaba el comando DrawLine, anteriormente para poder dibujar eso se tenía que crear un objeto DC, el cual nos da el contexto del dispositivo, lo que nos permite dibujar en pixeles, además de esto se debe crear un lápiz, posteriormente a es objeto definir el color y el ancho en pixeles para dibujar. El método Drawline usaba dos pares ordenados de coordenadas del plano de la ventana para dibujar la línea. El primer par ordenado era el punto de inicio de la recta y el segundo par ordenado es el punto final de la recta. Esto resulto muy util ya que solo fue necesario crear un método para calcular puntos con base en rotaciones de líneas para poder obtener los argumentos de la recta por dibujar. Fue bastante elaborado poder desplegar las líneas sin errores ya que tienen que estar en una clase específica en el proyecto junto con la ventana para poder implementarlas. Se logró implementar líneas de diferentes colores para así dar la oportunidad al usuario el color del fractal a dibujar aunque no era uno de los requisitos del proyecto.

- Cálculo de punto en un plano:

Como se mencionó antes para poder graficar cada una de las líneas del fractal se necesitaba saber el par ordenado de coordenadas exacto del inicio y el final de cada línea. Para esto se implementó un método dentro de la clase Grafico, donde para cada fractal, dependiendo del ángulo anterior y el ángulo relativo al que se desea desplazar, se calculan los puntos. Para cada rotación se hizo un método diferente ya que cada rotación tiene un comportamiento diferente. Esto se logró utilizando trigonometría y validación de casos basada en posiciones relativas y absolutas, donde la posición relativa es el ángulo de destino y la absoluta es la dirección en la que se encuentra “apuntando” un punto

- Encontrar la librería correcta:

Para este proyecto se utilizó la librería wxWidgets. Esta librería permite la creación de ventanas en la cual se despliega la interfaz y también la creación de líneas en dicha ventana. Esta librería facilitó mucho el proceso de interfaz gráfica ya que tenía métodos fáciles de usar y muy útiles para nuestros propósitos.

- Creación de listas con iteraciones:

Ya que el uso de ArrayList no era eficiente ni óptimo para este tipo de proyecto, se implementó el uso de LinkedList que a su vez utiliza nodos para poder ser implementada. Este tipo de lista se basa en nodos con punteros, tiene un head(cabeza) y un tail(cola) que ambos son nulos y son el fin y el inicio de la lista respectivamente. Los nodos están conectados mediante punteros, cada nodo tiene un puntero que apunta al siguiente nodo, head y tail también son punteros. Tail no apunta a nada y head apunta al primer elemento de la lista. Con este tipo de lista se pueden agregar elementos sin importar el tamaño máximo ya que no tiene, la lista puede crecer sin tener que preocuparse por el tamaño máximo, el uso de nodos también facilitó el agregar y navegar la lista.

- **Interfaz con el usuario**:

Para la interfaz se optó por una interfaz simple e intuitiva para el usuario, ya que no era necesario utilizar complejas interfaces debido a las pocas funciones de este proyecto. Se utilizan dos ventanas, una principal en la cual el usuario escoge el color en código RGB y la cantidad de iteraciones que quiere para el fractal. Cada fractal tiene un máximo de iteraciones para evitar que la computadora demore mucho en producir el fractal ya que con cada iteración el crecimiento de la lista es mayor. Luego de esto el usuario escoge el fractal a dibujar. Una ventana secundaria en la cual se despliega el fractal seleccionado con las iteraciones y el color de líneas deseado.

Análisis de Implementación

Todos los requerimientos del proyecto se lograron llevar a cabo. Cada fractal se logró graficar de manera exitosa ya que se le dedico mucho tiempo y con anticipación a cada uno de los fractales, además de que se le dio la investigación adecuada a cada aspecto de la graficación, es decir, a cómo dibujar las líneas, cómo calcular los puntos en el plano y como crear cada lista de cada fractal con las iteraciones deseadas. También se logró implementar funcionalidades adicionales al proyecto. Estas son la escogencia del color del fractal y la escogencia de la cantidad de iteraciones que desea para el fractal. Estas funciones adicionales se lograron gracias a la anticipación con la cual se terminó el proyecto inicial, al esfuerzo puesto en este durante semanas de trabajo y adicionalmente una gran iniciativa por los creadores para dar mayor uso y flexibilidad a su proyecto.

Conclusión

- Los fractales son figuras con comportamientos irregulares dependiendo de la cantidad de iteraciones. Con cada iteración la lista de cada fractal aumenta de tamaño considerablemente por lo que se debe tener cuidado de no utilizar muchas iteraciones para no causar que el programa sea muy lento o que se “pegue”.
- Los triángulos trigonométricos y los teoremas de pitágoras son muy útiles para calcular la distancia entre dos puntos con diferencia de ángulos, también son indispensables para saber la localización de dichos puntos. Esto implica la conversión a radianes y la multiplicación de la longitud del segmento por cada unidad de desplazamiento relativo dada ya sea por el seno o el coseno dependiendo del contexto específico.
- La librería utilizada afecta mucho al resultado del programa, cuando se selecciona una librería útil para el propósito se facilita mucho el proceso de creación de código o interfaz. Al contrario el no usar librerías o usar una incorrecta ocasiona que todo el proceso sea más tedioso o no se pueda lograr. La librería wxWidgets resultó indispensable en este proyecto ya que facilitó la escritura del código de la interfaz y la graficación de líneas además de que sus métodos llevaron a un código más limpio y eficiente.
- Las validaciones son primordiales en todo proyecto programado. Las validaciones están presentes a lo largo de este proyecto, nos ayuda a saber que lo que el usuario está haciendo es válido. Para obtener los colores deseados primero se valida que estos estén en el rango correcto y que además sean números, para las iteraciones se verifica que estén en el rango posible, sin estas validaciones entonces el usuario podría ocasionar un error de ejecución en el programa. Otras validaciones como el tamaño de cada segmento para la cantidad de iteraciones deseada también son importantes ya que sino la visión del fractal se distorsiona por la magnitud de este. La validación más importante es el análisis del contexto actual de graficación,

donde se tiene que tomar en cuenta de donde se viene y hacia dónde se va, utilizando ángulos para obtener pares ordenados en un plano bidimensional.

Recomendaciones

- Se recomienda tomar en cuenta que los fractales no siempre se comportan igual, por ejemplo depende de la cantidad de iteraciones sierpinski se desarrolla hacia arriba o hacia abajo. Tener esto en cuenta a la hora de graficar el fractal es de vital importancia para que el fractal se despliegue de la manera deseada. También los fractales crecen de tamaño puesto que aumentan la cantidad de líneas por fractal por lo tanto se tiene que tomar en cuenta el tamaño de las líneas para que el fractal se pueda ver en la ventana de despliegue. Se tuvo ciertas complicaciones cuando se intentó cambiar el tamaño de las líneas porque se tuvo que trabajar el cálculo de puntos con variables cambiantes por la cantidad de iteraciones. Este no pudo ser calculado por una fórmula sencilla ya que la recursión de los fractales es compleja, entonces se optó por una función que dado un número de iteraciones retorna un valor que es la longitud del segmento
- Se recomienda investigar mucho sobre los triángulos especiales trigonométricos ya que estos sirven de manera idónea para los ángulos de 60 y 45. Estos sirven para poder obtener la posición del siguiente punto cuando se tienen que hacer estas rotaciones con ángulos. Tuvimos ciertas complicaciones a la hora de calcular las rotaciones con ángulos a la izquierda o derecha, ya que para cada posición de recta se tiene que cambiar la fórmula ligeramente para que se despliegue de manera correcta. Es decir, se tiene que tener en cuenta no solo hacia dónde se mueve si no de donde viene también.
- Recomendamos fuertemente para este tipo de proyectos utilizar wxWidgets como librería para los gráficos ya que esta es muy útil y tiene todos los métodos necesarios para poder hacer interfaz y graficar fractales. Sin embargo, también se debe tomar en cuenta que se debe hacer una

investigación extensa sobre cómo implementar cada método ya que cada uno tiene muchas especificaciones para que pueda funcionar de manera correcta. Además de instalarla con anticipación ya que puede ser tedioso lograr instalarlo y que el código compila de forma correcta.

- Recomendamos primero pensar en las validaciones del proyecto para así irlas aplicando conforme se avanza en el proyecto. Es muy difícil y tedioso hacer validaciones una vez escrito todo el código ya que este tiende a dar errores cuando es modificado y además es difícil para cualquier programador volver a leer todo el código y saber donde va cada validación. Es más eficiente escribir el código pensando en estas, así ambos se acoplan de mejor manera. Aplicando así la regla del Boy Scout “Deja el campamento un poco más limpio de lo que lo encontraste”
- El código limpio toma un rol muchisimo mas relevante cuando se trabaja en grupos, ya que, como hay más de una persona revisando y modificando el código, es imperativo poder entender los cambios que la otra persona ha realizado para no perder el valioso tiempo tratando de entender el contexto de la aplicación.

Referencias

[1] (2011, agosto 16). *Common Dialogs* [Online]. Disponible en https://wiki.wxwidgets.org/Writing_Your_First_Application-Common_Dialogs

[2] (2011, abril 23). *Drawing on a Panel with a DC* [Online]. Disponible en https://wiki.wxwidgets.org/Drawing_on_a_panel_with_a_DC

[3] (2012, mayo 11). *Mathematics()* [Online]. Disponible en https://math.stackexchange.com/questions/143932/calculate-point-given-x-y-angle-and-distance

[4] (2017, septiembre 20). *wxTextEntry Class Reference* [Online]. Disponible en http://docs.wxwidgets.org/trunk/classwx_text_entry.html#af234557ffcbfefb3ea45358c4f2a5283

[5] (2012, diciembre 15). *Determine coordinates for an object* [Online]. Disponible en https://stackoverflow.com/questions/13895237/determine-coordinates-for-an-object-at-a-specific-distance-and-angle-from-a-point