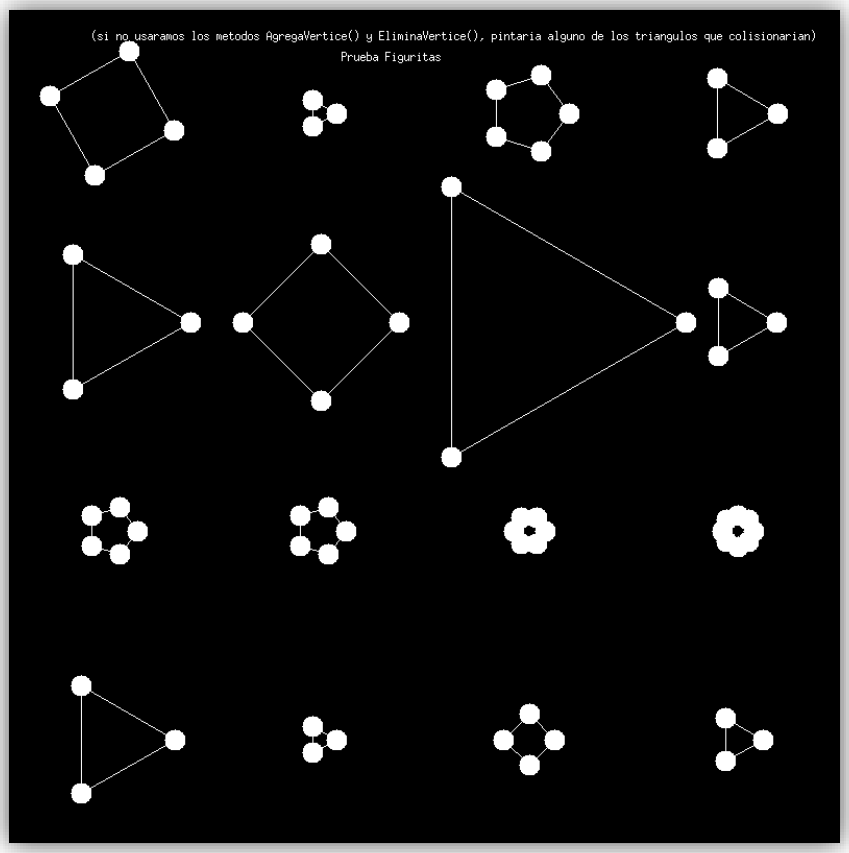
Metodología de la programación

PROYECTO FINAL PRÁCTICAS (PRIMERA ENTREGA)



Ejemplo de prueba figuras.cpp

Iván Cortón da Silva 15426623V - 1ºD

Índice:

Descripción de la práctica

Durante el desarrollo de la primera entrega de la práctica final, se nos encomendó en un primer momento, la prueba del main aportado por el profesor en el que podíamos comprobar el funcionamiento del módulo **miniwin** junto a un struct Punto formado por dos datos de tipo **float** como son “x” e “y” definiendo así las coordenadas en la ventana generada por **miniwin**.

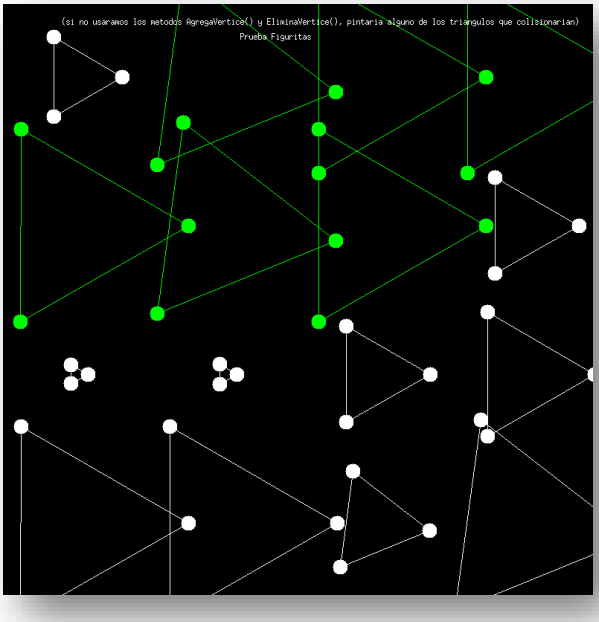
Una vez comprobado el uso de las funciones usadas para dibujar y realizar operaciones sobre nuestros puntos, se nos encargó la implementación de la clase Punto2D y del módulo pintar, donde se pretendía que, análogamente al main aportado por el profesor, se pudiera pintar en la ventana generada a través del módulo **miniwin**.

Hasta el momento no encontré mayor dificultad que la modularización en clases y del módulo pintar. Por otro lado, tras asegurarnos de que efectivamente la ejecución era prácticamente la misma que la del main del profesor, se nos encargó la tarea de desarrollar una clase **PoliReg**, que serviría para poder construir polígonos regulares inscritos en circunferencias de los que conoceríamos; los vértices pertenecientes a la figura con sus respectivas coordenadas, el radio de la figura regular, el número de vértices de ésta y por último, el punto encontrado en el centro de dicha figura.

En lo referente a esta segunda parte, sí es cierto que me vi algo más apurado, ya que había ciertas características de la clase **PoliReg** que no llegué a poder comprobar hasta haberla desarrollado por completo, junto con su propio main, por lo que fue al final, donde tuve que empezar a corregir todos aquellos errores que fui arrastrando. Dichos errores fueron relativos al mal uso de memoria dinámica (como puede ser la falta de liberaciones), o a aquellas características que debíamos de actualizar tras haber realizado operaciones como pueden ser la generada por el método Mover, el cual requiere que además de mover cada uno de sus vértices debamos de mover el centro de esta figura…

Por otra parte, en cuanto a la cuestión propuesta sobre la implementación de los sets respectivos a cada característica de cada polígono, creo que en este caso, y hasta donde hemos llegado con el desarrollo de la práctica, no tendrían por qué ser del todo necesarios, aunque podrían facilitar algunas operaciones teniendo siempre muy clara la implementación de dichos set, me explico, ya que podríamos haber implementado; un **set_Centro**, para el que necesitaríamos rehacer todos los vértices (incluido el centro) en función del radio, un **set_Radio**, para el que tendríamos que rehacer todos los vértices (sin incluir el centro) en función del radio, un **set_numVertices**, para el que tendríamos que liberar la memoria utilizada para volver a reservar espacio que utilizará nuestro método generador de vértices en función del centro, del radio y del número de vértices.

Por último, y en cuanto a la dificultad planteada para el desarrollo de esta parte de la entrega final, no creo que haya sido exageradamente difícil, aunque creo que igual haber tenido un manual algo más específico sobre los métodos a utilizar del módulo **miniwin** hubiera conseguido ahorrarme mucho tiempo de prueba y error, o quizá el tener un archivo main ya construido que se encargara de proporcionarnos con algo de detallo la estructura para poder implementar correctamente esta parte.

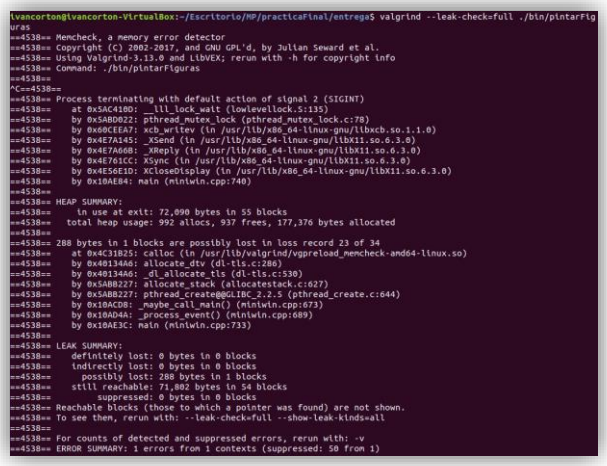


Ejemplo 2 de prueba figuras.cpp

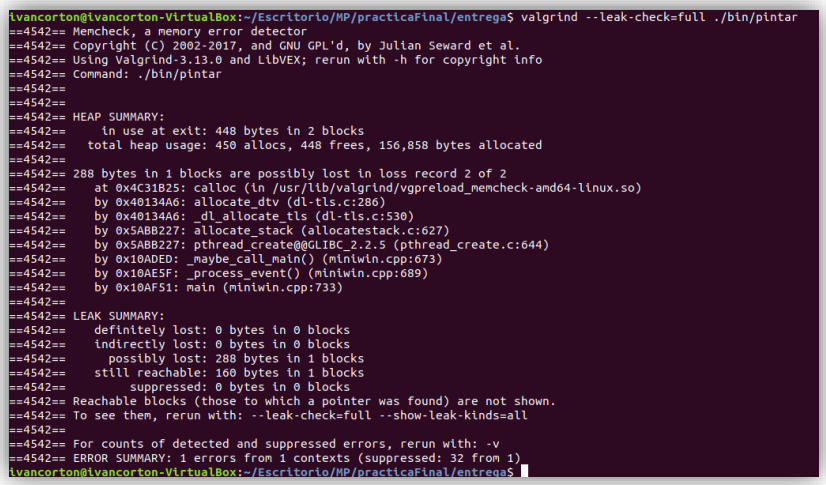
Análisis de Valgrind

- Órdenes utilizadas para obtener las salidas:

$$valgrind --leak-check=full ./bin/pintar$$
$$valgrind --leak-check=full ./bin/pintarFiguras$$



Valgrind sobre main.cpp



Valgrind sobre figuras.cpp