Документация на курсов проект на тема „Работа с SVG файлове“

Съдържание:

1. Увод
   1. Описание и идея на проекта
   2. Цел и задачи на разработката
2. Някои проблеми, възникнали при разбработката на програмата и тяхното решение
3. Архитектура
   1. Структура на класовете
   2. Файлова структура
4. Тестване
5. Възможности за подобрение

1.Увод

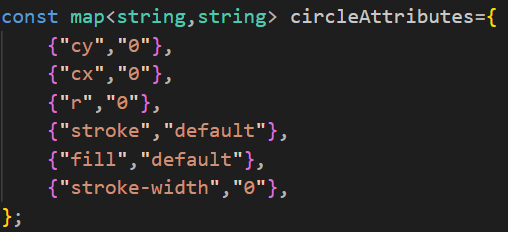
Тази документация има за цел да предостави информация за изградения проект и да покаже трудностите, свързани с неговото създаване и техните решения. Идея на проекта е да се създаде програма за доста просто манипулиране на SVG файлове.

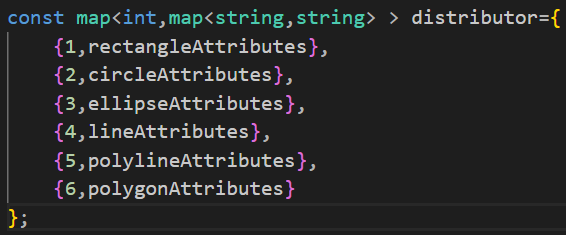
Зададената задача включва имплементация на: четене от даден файл, създаване, премахване и транслация на фигури, записване на информацията в същия или друг файл и откриване на фигури, които се намират в определена област зададена от правоъгълник или окръжност. Също така се изисква поддържане на поне 3 фигури от категорията basic shapes в SVG файловете. В имплементацията на този проект са включени и други функционалности като: поддръжка на всички фигури от категорията basic shapes в SVG файловете и освен тях поддръжка на елемента група, който много усложнява някои други функционалности на проекта. Освен тези са добавени и функционалностите: намиране на лице и периметър на фигура(за фигурите circle, rectangle, ellipse и polygon) и намиране на дължина на фигура(за фигурите line и polyline). Друго изискване, което е изпълнено е да се направи лесно добавянето на нови фигури, като за решението на този проблем се използват различни map-ове от тип string, които се използват като шаблони за въвеждане на данни от командния ред и от файл.

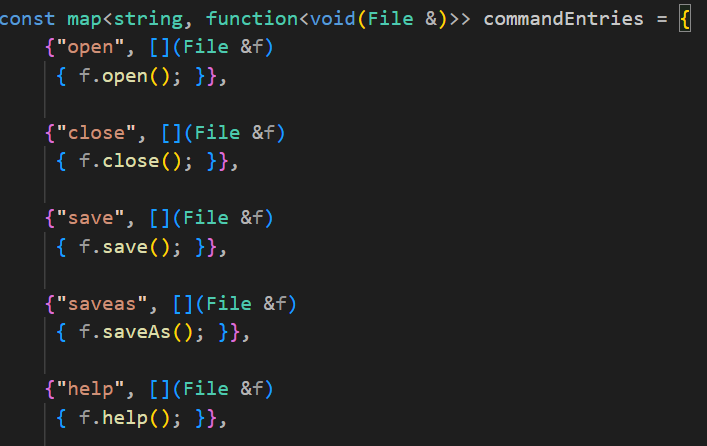
2. Някои проблеми, възникнали при разбработката на програмата и тяхното решение

Споменахме, че добавянето на групи усложнява някой функционалности на програмата. Такива функционалности са: намирането на правоъгълната област, в която се намират всички фигури в тази група, четенето от файл и правилното превръщане във вложени обекти, изтриването на конкретна фигура. Също така използването на групи означава, че е нужно създаването на виртуален copy constructor и деструктор на групата, който да освобождава паметта. Друго затруднение, което групите носят, е нуждата от рекурсивно обхождане и понякога правене на неща в обратния ход на рекурсията. Пример за това е изтриването на фигура, за което е нужно не само да освободим паметта за дадения обект, а и да го премахнем от вектора, в който тя се е намирала. За тази цел рекурсията е написана да връща pair<Group,BasicShape\*>, като първата част е групата в която се намира фигурата. Варианта, в който трием фигурата още от групата беше сменен поради нуждата от рекурсия на няколко места, което доведе до отделянето и в отделна функция, която само връща фигурата и групата,а не ги променя.

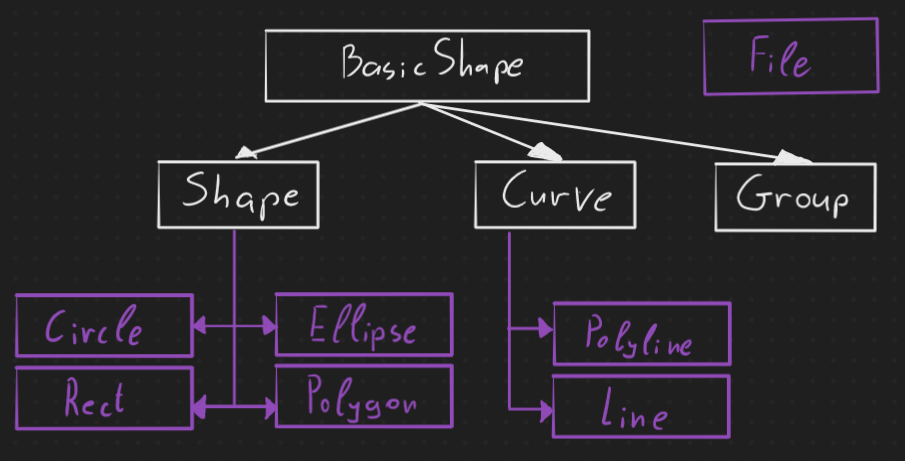
Друг проблем беше да се направи лесно добавянето на поддръжка за нови фигури. Този проблем беше решен чрез добавянето на map<string,string > :

Чрез този мап задаваме и default стойностите на дадената фигура, като за всяка стойност само презаписваме информацията в map-a. По този начин покриваме всички случаи на подадена информация. Подадената информация може да бъде каквото и да е, също така програмата разглежда само тези стойности, ако има параметър с друго наименование, тоще бъде записано в map-a, но не и в фигурата Circle, а чрез distributor:

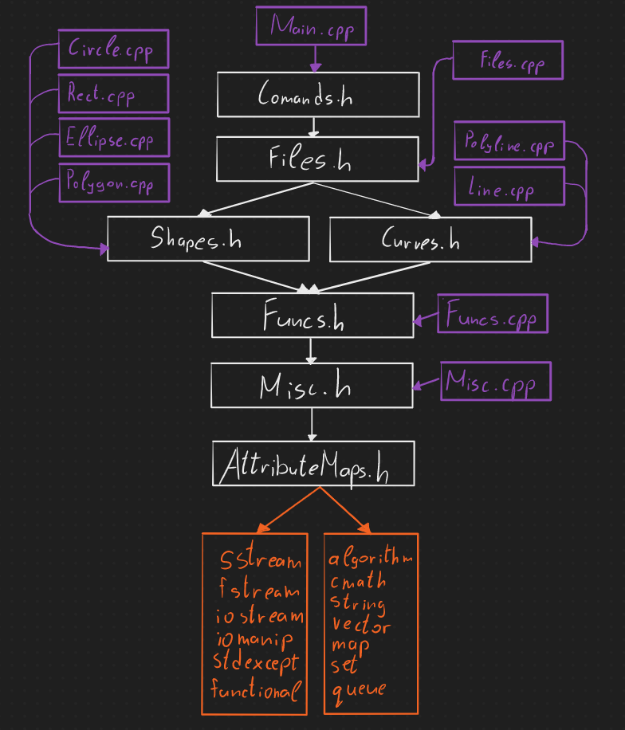
можем да не докосваме кода. Всяко число е shapeID-то на фигурите(всяка фигура получава такова при създаване). Като използваме и двете неща в комбинация, получаваме добавяне на четене на нова фигура само чрез добавяне на нов map и добавянето му в distributor(и писането на функция fillArguments() за дадената фигура). Освен това по този начин игнорираме излишните параметри и работи за всякакви конфигурации от поддържаните.

3.Архитектура

3.1. Структура на класовете

В програмата има 2 основни класа. Единият е File, чрез който можем да създаваме обекти от този тип и да ги използваме за манипулация на 1 файл. В този клас има методи за всяка функционалност на програмата. Общото между тях е, че всичките са void и са без параметри. Точно тази еднаквост ни позволява и да ги запишем в map, който за даден стринг изпълнява дадената функция. По този начин командите от командния ред, които не се разпознават от map-a, ще бъдат игнорирани.

Другият основен клас в програмата е BasicShapе, който представя всички фигури, включително и групите. Тъй като фигурите се различават съществено в това, че някои от тях имат лице и периметър, а другите имат само дължина са въведени още 2 абстрактни класa: Curve и Shape. Тези класове наследяват BasicShape, а техните наследници са съответно на Curve: Line и Polyline и на Shape: Circle, Rect, Ellipse и Polygon. Групите се наследяват директно от BasicShape и в себе си съдържат vector<BasicShape\*>.

3.2. Файлова структура

Файловата структура e опсисана чрез схемата отдясно. В нея с лилаво са отбелязани .cpp файловете на програмата, с бяло - .h файловете, а с оранжево - отбелязани стандартните С++ библиотеки, които се използват в програмата. Идеята на структурата е смислено различният код да бъде разделен в различни файлове. В AttributeMaps.h имаме дефинирани мапове, които се използват от по-горните файлове. В Misc.h и Misc.cpp са дефинирани и имплементирани базовите класове, техните невиртуални фунцкии и структури, които се използват в цялата програма. Funcs.h и Funcs.cpp са файлове в които са дефинирани и имплементирани помощни функции(основно операции със string). В Shapes.h и Curves.h има дефиниции на фигурите, а в съответните .cpp файлове има имплементация на техните методи. Files.h и Files.cpp са файловете, в които е дефиниран и имплементиран класа File. В Commands.h са инициализирани map-овете, които спрямо даден стринг извикват определен метод на обект от тип File и Factory-то, което спрямо string връща обект от искания тип.

4.Тестване

Примерни сценарии за тестване:

1)

open <path to>\Tests\test2.svg

translate horizontal:50 vertical:100

print

translate 7 horizontal:0 vertical:-50

erase 2

create rect 10 10 50 50 transparent blue 8

saveas <path to project>\Tests\result.svg

close

open <path to project>\Tests\result.svg

print

save

exit

2)

open <path to project>\Tests\test3.svg

print

create rectangle -1000 -1000 10 20 yellow

print

within rectangle 30 30 210 80

within circle 0 0 278

erase 11

erase 100

print

translate vertical=10 horizontal=100

saveas <path to project>\Tests\test4.svg

translate vertical=-10 horizontal=-100

print

save

exit

По-подробни примери има в ExampleScenarios.txt

5.Възможност за подобрение

Идеи:

1. Създаване на групи
2. Създаване на елементи във групи
3. Визуализация на текущия файл
4. Оптимизация на някои от алгоритмите
5. Добавяне на path елементи