Документация

за проект по ООП

Тема 2: Работа със SVG файлове

Изготвила: Виктория Стилиянова Минчева

ФН:45702

1. Увод
   1. Описание и идея на проекта

В рамките на този проект трябва да се разработи приложение, което работи със файлове във *Scalable Vector Graphics (SVG)* формат*,* или в превод “мащабируема векторна графика”. Приложението трябва да може да зарежда фигури от файл, да извършва върху тях дадени операции, след което да може да записва промените обратно на диска.

Програмата трябва да може да поддържа поне три от основните фигури, които *SVG*форматът съдържа и да бъде възможно изпълнение на дадени команди върху тях. За повече информация за това кои са базовите фигури, вижте<https://www.w3.org/TR/SVG/shapes.html>.

* 1. Цел и задачи наразработката

Програмата има за цел да работи по обработката, промяната и съхранението на данни във файл, като междувременно се покаже добра работа и разбиране на ООП практиките в C++.

Като за начало трябва да се помисли как най-оптимално да се разпредели задачата на по-малки задачи с цел по-лесно изпълнение, но е важно и да се вземе под внимание как те ще могат да бъдат свързани заедно в бъдещето изпълнение на проекта. Дизайнът на приложението трябва да е такъв, че да позволява лесно опериране с командите, обвързани със съответните фигури.

Да се осъществят следните команди, задължителни за изпълнението на проекта:

*open* (отваряне на съотвения файл, където се съдържа информацията за фигурите), *close* (затваряне на файла), *create* (създаване на нова фигура, подадена от конзолата), *print* (принтиране, или още изкарване, на информацията за всяка фигура, съдържаща се в съответния файл), *erase* (изтриване на фигура, подадена чрез нейния номер във файла), *within* (при подадена произволна фигура да се покаже дали в нея се съдържа фигура/и, пазена/и във файла, с който работим), *translate* (да транслираме, т.е. да изместим дадените x и y координати с определени стойности, дадена фигура или всички), *save* (да бъдат запазени промените във файла), *save as* (промените да бъдат запазени в нов файл и под ново име), *help* (да помага при възнокнал проблем или да дава носоки за изпълнението на приложението), *exit* (да прекратява изпълнението на програмата).

Всяка фигура съдържа различни по брой и вид параметри, затова е нужно създаването на класове за всяка една от тях и трябва да се помисли дали фигурите имат нещо общо помежду си.

1.3. Структура на документацията

Описание на задачата и проблемите, които следва да бъдат засегнати с цел изпълнение на поставената задача. Разяснение на моята реализация на програмата и решенията, които съм взимала по време на писане на кода. (избрана архитектура, описание на на член-данните и член-функциите на класовете и начинът им на използване). Също така идеи за бъдещи подобрения.

1. Преглед на предметната област

Изпълнението на задачата има за цел да улесни цялостната работа с *SVG* файлове – от създаването на фигура и записването и във файл, до изтриването ѝ.

Цялата информация за фигурите се пази във файл с подходящо разширение (.svg) и той е ключов за изпълнението на задачата. Четенето от файл и писането в него са основата на всяка една операция и затова е важно как се работи с него.

Като за начало се потребителя трябва да въведе една от позволените команди, като всяка една от тях си има свои специфики, а ако той не е наясно с тях може да потърси насоки за това кои са командите, които програмата поддържа, и съответните им изисквания.

След като бъде въведена в конзолата валидна операция, се използва предимно алгоритъм за токенизация (*Тokenization*), който позволява по-лесно да се определи коя операция трябва да се изпълни, като разделя всяка част от командата на токени(отделни части) и гледа дали първата дума съвпада с името на възможните операции. А след което, ако операцията сама по себе си го изисква, се изполва още веднъж токенизация, но този път с цел да се придобие подадената информация за по-нататъшното изпъление на операцията.

Тук е важно да се уточни, че всяка една операция работи с всяка от шестте основни фигури, поддържани от SVG. Спрямо фигурата програмата знае в какъв формат да изведе нужната информация за нея, как да създаде нова от съответния тип, как да провери дали се съдържа в друга фигура и как да промени своите координати.

1. Проектиране

Реализацията на задачата е изградена съгласно добрите практики на обектно-ориентирания стил.

За тази цел програмата разполага и с общо девет класа, където се срещат полиморфизъм, наследяване, енкапсулация:

* Команда(1): Command;
* Фигура(1): Shape;
* За всяка отделна фигура(6): кръг (*Circles*), правоъгълник (*Rectangles*), линия (*StraightLines*), елипса (*Ellipses*), *polyline* (*Polylines*), *polygon* (*Polygons*);
* Точка(1): Point.

Изпълнението на задачата тръгва от *main()* функцията, където при правилно подадена команда веднага се озоваваме в първия клас – „ Команда“.

Класът *Command* има една от най-важните роли в този проект. Чрез него се управлява цялостното изпълнение на програмата. Всички базови фунции, които тя поддържа, се изпълняват там и спрямо съответнатта функция информацията се препраща към класа на някоя фигура за по-нататъшното използване на подадената информация. Освен това, там се съдържат и много помощни функции, които помагат за изпълнението на главните такива.

Всяка от базовите фигури в проекта се намира в отделен клас, който наследява класа *Shape* (фигура), където се намира общото поле на всяка фигура и това е цветът (color), както и виртуалните функции (полиморфизъм) за извеждане на информация за съответната фигура в конзолата и тази за пренасяне на информация от конзолата във файла.

Във всеки един клас, посветен на определена фигура, се съдържат подобни една на друга функции, но с нужните специфики на всяка фигура. Основните, които задължително присъстват, са: създаването, транслирането, проверката за местоположение в друга фигура, токенизацията на подадените в конзолата параметри и виртуални функции за принтиране на информация (overloaded virtual functions).

Кръгът, правоъгълникът, линията и елипсата са с фиксирани полета, които никога не се променят и са задължителни, така например съдаването на една от тези фигури става невъзможно без един от компонентите и.

За разлика от тях обаче, *polyline* и *polygon* могат да приемат различен брой точки и тук идва функцията на класа Point. Тези две фигури се състоят изцяло от точки с нефиксиран брой в кооординатната система, но с едно ограничение – не могат да бъдат създадени, ако имат по-малко от три точки, и както е споменато по-горе те също наследяват цвета си от класа Shape.

1. Реализация, тестване

В *main* функцията се приемат заявки за отделните команди, които програмата може да изпълнява, като при грешно въведена команда ще се наложи да въведем наново исканата от нас команда или да потърсим помощната такава *help*, тя ще ни даде насоки за съществуващите команди и как да ги използваме.

Преди изпълнението на почти всяка команда (с изключение на отваряне на файл - *open*, и излизане от програмата - *exit*) се проверява дали има отворен съществуващ вече файл, т.е. командата open е задължително да бъде първа, тъй като без нея няма да можем да достъпим останалите, понеже няма да имаме нужната информация. След отварянето на файла вече можем да използваме всяка една команда от наличните.

Да кажем, че искаме да проверим на първо време какви фигури се съдържат в нашия файл. За целта ползваме командата *print*, която си взима само тези редове във файла, които съдържат информация за фигура, а другите биват пропуснати. От там според фигурата се извиква различна *print*-функция (защото всички имат ралични полета), намираща се в съоветния клас. Също така дава на всяка фигура пореден ID номер.

Следваща евентуална стъпка е да си направим нов *shape*. За да се случи това използваме *create*, последвано от вида фигура и нейните спецификации, които при нужда можем да намерим в *help*. Също както при *print*, за всяка фигура си има отделна функция, към която ще бъдем препратени, за да се създаде във файла новата фигура. Логично тя отива най-отдолу като последна, но тъй като в C++ не е позволено да се пише в средата на файл, се копира цялата информация във временен файл, след което се създава новата фигура и накрая се добавя затварящия *tag* (в нашия случай – „</svg>“). Преименува се временния файл с оригиналното име на файла и се трие стария.

Ако пък искаме да изтрием фигура, ползваме *erase* и номера на фигурата, която искаме да изтрием. Използваме същата тактика като при *create*: нов временен файл, копиране на информацията от оригиналния, пропускаме само тази фигура, която искаме да изтрием, преименуваме временния файл с името на оригиналния и той е готов да се изтрие. Така вече си имаме файл без нежеланата фигура.

Следваща команда – *within*, следвана от произволна фигура, която не е задължително да бъде от вече наличните във файла. Важно е да се каже, че можем да проверяваме само в рамките на кръг и правоъгълник и с правилни спецификации, като изпускаме цвета. Тази команда ще ни върне всички фигури от нашия файл, които напълно се съдържат дадения кръг или правоъгълник и техните спецификации. В помощ на командата се притичват няколко фунции, които проверяват важните точки от фигурите във файла дали се намират в даената фигура и ако всички са вътре, то съответната фигура се намира изцяло вътре и би се изписала на екрана.

Последната команда свързана с фигурите е *translate*. Тя има задачата да транслира една или всички фигура/и, т.е. да промени x и y координатите като тук е много важно всичко да бъде въведено правилно, иначе командата няма да работи. Но ако объркаме нещо, програмата си казва какво точно сме объркали и как да го напишем правилно. Тази команда също ползва няколко помощни функции, които и помагат да определи с колко да се измести вертикала и с колко хоризонтала, и също така спрямо това дали искаме да транслираме една или всички фигури, си има отделни функции.

Останаха командите *save*, *save* as и *exit*. *Save* запазва промените по файла. *Save as* пък от своя страна запазва промените, но в нов файл и под ново име. А *exit* прекратява изпълнението на програмата.

В проекта няма задължителен ред, по който да се изпълняват операциите, но както беше споменато по-горе, функцията *open* трябва да е първа, за да можем да достъпим информацията, с която можем да боравим.

Предложението ми за изпълнение на програмата би било:

* open <fileName>;
* help;
* print;
* create <shapeName> <shapeSpecifics>;
* print //за да се види, че успешно е била създадена фигурата;
* translate <shapeNumber> vertical=<number> horizontal=<number>;
* print //да се види промяната;
* within <shape>;
* erase <shapeNumber>;
* print //да се види, че я няма;
* save;
* save as;
* close;
* exit.

5. Заключение

Считам, че откъм изпълнение на задачата, съм покрила всички поставени изисквания и задачи, спазвайки добрите ООП практики. Разбира се, винаги има възможност за подобрения.

За в бъдеще бих искала да поработя отново над формата, така че да се създават „истински“ *SVG* файлове, които да могат да визуализират реално подадените фигури. И евентуално повече използване на абстрактни функции.

Използвана литература:

<https://en.wikipedia.org/wiki/Scalable_Vector_Graphics>

<https://www.w3.org/TR/SVG/shapes.html>

<https://bg.wikipedia.org/wiki/SVG>

GitHub: <https://github.com/ViktoriaMincheva/SVGProject>