**SVG File Editor**

**Кратко описаниe:**

**Поддържани команди:**

close Затваря текущо отворения файл.

save Запазва всички промени в съответните им файлове.

saveas <file> Запазва всички промени в избрана от потребителя локация. За да работи коректно трабва папкта да е съществуваща, иначе не се създава. Но файл се създава.

Help Принтира наличните команди и информация за тях.

еxit Излиза от програмата.

print Принтира всички налични фигри.

create Създава нова фигура по дадени:

<име на фигура> <съответни параметри за фигурата>

erase <n> Изтрива n-та фигура.

translate vertical:<value> horizontal:<value> [n] Транслира [n]-та фигура или всичките ако не е посочено [n].

within <option> Принтира всички фигури които са изцяло вътре във фигурата <option>.

За момнета това са Кръг и Правоъгълник.

clear Изчиства екрана от всичко.

**Описание на архитектурата:**

**Класове:**

Engine В него е основната функция Run(), от която започва изпълнението на програмата. Тя чете входа и го предава на съответната функция на controller компонентата.

Controller Съдържа всичката функционалност на програмата. Всяка потребителска команда има съответна функция в този клас. Тук имаме и списък със всички работни файлове(не със физическите файлове а техните съдържания). Също така указател към последно отворения работен файл.

WorkingFile Той се занимава с четенето и записването на файлове. Извличане на svg елементите, както и създаването на обектите.

Shape Това е базов клас който всяка фигура наследява. Това позволява да се добавят още фигури без да се наруши работата на програмата. Единствените промени се правят в WorkinFile класа, като там просто трябва да се добави създаването на съответния обект в CreateObject() метода. Също така тук са и общите методи за обекти. ToStringToFile(), ToStringToPrint(), WithinCircle(), WithinRectangle().

Circle Наследник на Shape. Описва характеристиките на кръга.

Rectangle Наследник на Shape. Описва характеристиките на правоъгълника.

**Допълнителни структури:**

typename <class T>

struct block

{

std::vector<T> data;

int id;

}

Струрктурата е основна при разделянето на svg тага от останалите елементи на файла. С нейна помощ се поддържа съответната позиция на елементите във файла.

С пример ще се изясни използването.

Пример:

Имаме примерен файл със съдържанието:

<?xml version="1.0" standalone="no"?>

<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"

  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg>

  <rect x="5" y="5" width="10" height="10" fill="green" />

  <circle cx="5" cy="5" r="10" fill="blue" />

  <rect x="100" y="60" width="10" height="10" fill="red" />

</svg>

Идеята е след като се използва функията

DataExtraction(std::vector<block<std::string>>& file, std::string& path);

От посочния [path] да се прочете информацията и да се запише във file като колекция от block<string> елементи. Видът на файл ще е сления:

{

{

data: {

“NOTSVGSTART”

}

id: -1

},

{

data: {

“<?xml version="1.0" standalone="no"?>”,

“<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"”,

“"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">”

}

Id: 0

},

{

data: {

“NOTSVGSTART”

}

Id: -1

},

{

data: {

“SVGSTART”

}

Id: -1

},

{

data: {

“ <rect x="5" y="5" width="10" height="10" fill="green" />”,

“<circle cx="5" cy="5" r="10" fill="blue" />”,

“ <rect x="100" y="60" width="10" height="10" fill="red" />”

}

Id: 1

},

{

data: {

“SVGEND”

}

id : -1

}

}

По този начин се прави разделяне къде има част която не е svg и която е. За удобство на работа се използват така наречените сепариращи блокове които служат за разделянето на различните по вид тагове. Съответно “SVGSTART” индикира начало на такъв таг докато “SVGEND” индикира край. Аналогично с “NOTSVGSTART” и “NOTSVGSTART” които пък показват начало и край на други тагове. Правим разлика дали даден block е сепариращ блок или такъв с конкретна информация по id-то. Ако е id = -1 то тогава блока е сепариращ. Ако е друго число то съотвества на позицията на тоци блок в целия файл.

Останалата функционалност се реализира на база тази структура. Поради тази причина има места в които които може да се получи множествено влакане на цикли за достъпане на елементи.

**Архитектурна структура:**

Engine e основен и задвижва останалите. Използва Controller който държи информация за всичко случващо се и той на свой ред използва WorkingFile. Вече WorkingFile използва дефакто обектите фигури и се занимава с тяхното четене от файл, създаване, синхронизиране и записване обратно във файл.

**Забележки и подобрение:**

Безспорно това не е най-добрия начин за решаване на задачата, но предоставя един интересен подход за работа. Има случаи които не са изтествани при работа с командите и следователно са възможни неочаквани резултати. Не са покрити всички начини за работа с svg файлове. Поддържат се само две фигури. И като цяло работата с файла е успешна ако svg файла е във вид като примера по горе.