Растерна графика

Връзка към проекта: https://github.com/MLambeva/Raster-graphics.git

1. Увод

1.1 Проектът представлява конзолен редактор на растерни изображения. Той работи с три формата – PPM(Portable PixMap), PBM(Portable BitMap), PGM(Portable GrayMap). Неговата идея е да поддържа работа с различни файлове с тези три формата, да стартира сесии, които са с уникални номера, да прилага различни трансформации върху изображенията и при желание от потребителя да записва крайния резултат.

1.2 Неговата цел е успешно да могат да се правят основни промени върху растерни изображения като промените, които настъпват, се определят от потребителя. В началото потребителят може да види какви команди поддържа програмата, като напише командата „help". Зареждането на изображение/я става с команда „load", което автоматично създава „потребителска сесия", която има уникален номер. В една сесия може да има повече от 1 изображение. След въвеждане на команда „load" се очаква поне едно изображение, което да е коректно, за да се създаде сесия. Трансформациите, направени в рамките на една сесия, се прилагат върху всички изображения. След приключване на обработката потребителят може да запази направените промени върху същия файл, да го запише с друго име или в друга директория, да затвори сесията или да излезе от самата програма без да са се запазили промените. При затваряне на сесията той може да зареди следваща и да изпълнява същите неща както в първоначалната.

1.3 В документацията по-долу съм разяснила по-подробна информация относно:

- Как съм проектирала класовете си и какви член данни и методи съм използвала в тях.
- Как съм реализирала методите в класовете.
- Как мисля, че съм се справила и мнението ми за бъдещо усъвършенстване.

2. Проектиране:

2.1 Проектът се състои от:

- седем файла с разшерение „.h", в които са дефинирани отделните класове;
- осем файла с разширение „.cpp", в седем от които са имплементирани конструктурите и функциите на отделните класове, а деветият е главната функция, в която се реализира проектът.
- В проекта са използвани свойствата на полиморфизма.

2.2 Основните класове в проекта са:

- **Клас Формати**, който се характеризира с:
  - Име на изображението (името е с разширение „ppm", „pbm", „pgm");
  - ASCII номер (P1 – „pbm", P2 – „pgm", P3 – „ppm");
  - Коментар (Започва със символа '#');
  - Ширина (цяло положително число);
  - Височина (цяло положително число);

Клас Формати е базов клас, в който са дефинирани конструктор по подразбиране, виртуален деструктор, виртуални функции, множество от чисто виртуални функции, оператор за изход и три статични функции, които се използват при записване на промените за дадено изображение.

- **Клас „PPM“**, който е наследник на клас Формати, се характеризира с:
  - Максимална стойност (цяло число от 0 - черно до 255 - бяло);
  - Матрица от пиксели като един пиксел се образува като смесица от трите основни цвята – червено, зелено и синьо;
- **Клас „PBM“**, който е наследник на клас Формати, се характеризира с:
  - Матрица от пиксели като един пиксел е или 0 - бяло, или 1 – черно;
- **Клас „PGM“**, който е наследник на клас Формати, се характеризира с:
  - Максимална стойност (цяло число от 0 – черно до 255 - бяло);
  - Матрица от пиксели като един пиксел може да е в порядъка от 0 до 255, но максималната стойност определя най–наситения нюанс на бялото;

И в трите класа - „PPM“, „PBM“ и „PGM“ са дефинирани конструктор по подразбиране, деструктор, виртуалните и чисто виртуалните функции от базовия клас.

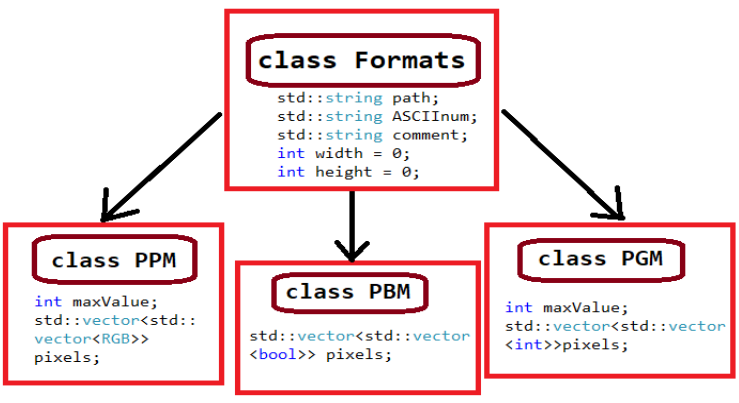
- **Клас „RGB“**, който се характеризира с три цели числа като първия показва наситеност на червения цвят, втория – на зеления и третия – на синия.

В него са дефинирани конструктор, селектори и мутатори и оператор за изход. Той се използа при формата „PPM“, защото пикселите на този формат може да си ги представим като матрица, в която всеки елемент се състои от 3 цвята, т.е. всеки елемент е от класа „RGB“.

- **Клас Сесия**, който се характеризира с:
  - Уникален идентификационен номер (статична променлива, която си променя стойността);
  - Масив от формати (в една сесия може да има множество от изображения от различен формат);
  - Масив от трансформации (при извършване на някаква промяна върху изображенията от текуща сесия, в този масив се запазват направените промени);

В този клас са дефинирани конструктор, селектори, функция за добавяне на трансформация, функция за генериране на идентификационния номер и оператор за изход.

- **Клас Редактор**, в който са дефинирани две статични променливи, помощни функции и основните команди за работата на редактора.



3. Реализация

3.1 Реализиране на методите в класовете:

- В **клас Формати** са имплементирани:
  - Виртуален дистуктор. Той трябва да се съдържа в класа, поради наличието на чисто виртуални функции.
  - Селектори, с помощта на които достъпваме член данните, които са в “private” частта на класа, извън този клас.
  - Оператор за изход, който извежда информация за всеки формат като се свързва с чисто виртуалната функция „print“ на производните класове.
  - Виртуални функции, които връщат на конзолата съобщения, че дадена команда не е изпълнена. Това са функции, които не са имплементирани във всички класове и когато се извика такава функция, ако в наследения клас тя не е описана, то се извиква функцията от базовия клас.
  - Три статични функции, които проверяват дали дадено число е едноцифрено, двуцифрено или трицифрено. Тези функции се използват, когато записваме информация за изображението във файл, защото при форматите „PPM“ и „PGM“ имаме изискване за разстояние между всеки пиксел, докато при формата „PBM“, на който пикселите са само нули и единици, не е задължително за съществува такова разстояние.
- В **клас „PPM**“ са имплементирани:
  - Конструктор и деструктор.
  - Всички чисто виртуални функции от базовия клас:
  - функцията „load“, която чете информация за изображение от файл по дадено име. При отваряне на потока за четене се извършва проверка за състоянието на потока. Ако то е добро, то първо се прочита ASCII кода на изображението и се проверява дали наистина съответства на разширението. Понеже не всички изображения имат коментари, то ние трябва да сме сигурни какво ще четем и затова прочитаме един символ. Ако този символ е ‘#’, то значи изображението има коментар и трябва да върнем указателя един символ назад, след което прочитаме коментара. Ако няма коментар, то значи трябва да прочетем информация за ширината и височината, максималната стойност, а след това и за пикселите. Пикселите на този формат може да си ги представим като матрица, на която всеки елемент е представен от 3 цвята.
  - функцията „saveas“, чрез която записваме направените промени във файл, като позволява на потребителя да укаже неговия път.
  - функцията „print“, чрез която извеждаме информация за дадения формат.
  - функцията „negative“, която прави цветово обръщане, т.е. от максималната стойност изважда текущите пиксели.
  - функцията „rotation <direction>“, която завърта изображението на 90° в зависимост от зададената посока.
  - функцията „collage“, която създава колаж от две изображения (<image1>, <image2>), които са от един и същ формат и еднаква размерност, в зависимост от зададена посока - хоризонтална или вертикална, като полученото изображение се записва в ново изображение <outimage>.

- Виртуалните функции:
- „grayscale“, която променя пикселите да са в сив нюанс. Формулата, която съм използвала за променяне на пикселите, съм взела от „tutorialspoint.com“.
- „monochrome“, която променя пикселите да са черни или бели. Тя работи по следния алгоритъм:

1. Променяме изабражението чрез „grayscale“, за да стане само с сиви нюанси.
2. Избираме си някакво число, което ще бъде граница. В случая съм си избрала 123.
3. Ако червеният пиксел е >= 123, то пикселите стават {255, 255, 255}, т.е. бели. Ако червеният пиксел е <123, то пикселите стават {0, 0, 0}, т.е. черни.

Тези две функции се прилагат само ако изображението е цветно.

- функциите „undoGrayscale“ и „undoMonochrome“, които премахват направени промени.

➢ В **клас „PBM**“ са имплементирани:

- Конструктор и деструктор.
- Всички чисто виртуални функции от базовия клас:
- функцията „load“ чете информация за изображение от файл по дадено име. При отваряне на потока за четене се извършва проверка за състоянието на потока. Ако то е добро, то първо се прочита ASCII кода на изображението и се проверява дали наистина съответства на разширението. Понеже не всички изображения имат коментари, то ние трябва да сме сигурни какво ще четем и затова прочитаме един символ. Ако този символ е ‘#’, то значи изображението има коментар и трябва да върнем указателя един символ назад, след което прочитаме коментара. Ако няма коментар, то значи трябва да прочетем информация за ширината и височината, а след това и за пикселите. Пикселите на този формат може да си ги представим като матрица, на която всеки елемент е от булев тип.
- функцията „saveas“, чрез която записваме направените промени във файл, като позволява на потребителя да укаже неговия път. При записване този формат няма ограничение дали ще има празно място между всеки символ, тъй като се състои само от нули и единици.
- функцията „print“, чрез която извеждаме информация за дадения формат.
- функцията „negative“, която прави цветово обръщане, т.е. от 1 изважда текущия пиксел (0 или 1).
- функцията „rotation <direction>“, която завърта изображението на 90° в зависимост от зададената посока.
- функцията „collage“, която създава колаж от две изображения (<image1>, <image2>), които са от един и същ формат и еднаква размерност в зависимост от зададена посока - хоризонтална или вертикална, като полученото изображение се записва в ново изображение <outimage>.

➢ В **клас „PGM**“ са имплементирани:

- Конструктор и деструктор.
- Всички чисто виртуални функции от базовия клас:
- функцията „load“ чете информация за изображение от файл по дадено име. При отваряне на потока за четене се извършва проверка за състоянието на потока. Ако то е добро, то първо се прочита ASCII кода на изображението и се проверява дали наистина съответства на разширението. Понеже не всички

изображения имат коментари, то ние трябва да сме сигурни какво ще четем и затова прочитаме един символ. Ако този символ е ‘#’, то значи изображението има коментар и трябва да върнем указателя един символ назад, след което прочитаме коментара. Ако няма коментар, то значи трябва да прочетем информация за ширината и височината, за максималната стойност, а след това и за пикселите. Пикселите на този формат може да си ги представим като матрица, на която всеки елемент е цяло неотрицателно число.

- функцията „saveas“, чрез която записваме направените промени във файл, като позволява на потребителя да укаже неговия път. При записване е задължително изискване между всеки пиксел да има поне едно празно място.
- функцията „print“, чрез която извеждаме информация за дадения формат.
- функцията „negative“, която прави цветово обръщане, т.е. от максималната стойност изважда текущия пиксел.
- функцията „rotation <direction>“, която завърта изображението на 90° в зависимост от зададената посока.
- функцията „collage“, която създава колаж от две изображения (<image1>, <image2>), които са от един и същ формат и еднаква размерност в зависимост от зададена посока - хоризонтална или вертикална, като полученото изображение се записва в ново изображение <outimage>.

➢ В **клас „RGB“** са имплементирани:

- Селектори и мутатори, за да можем да достъпваме и променяме пикселите на изображение извън този клас.
- Оператор за изход, в който се грижим да има разстояние между всеки пиксел.

Този клас използваме в класа „PPM“, за по-лесно представяне на пикселите.

➢ В **клас Сесия** са имплементирани:

- Конструктор по подразбиране.
- Селектори за лесно достъпване на елементи от класа извън него.
- Функция, чрез която лесно добавяме направена трансформация.
- Функция, която генерира идентификационен номер, който започва от 1.
- Оператор за изход, който извежда информация за идентификационния номер, изображенията в сесията и направените транформации върху тях.

Този клас се използва при стартиране на програмата като в една сесия може да има повече от едно изображение. Командите, които се броят за трансформация върху изображения са командите „grayscale“, „monochrome“, „negative“, „rotate <direction>“, „collage direction <image1> <image2> <outimage>“, „add<image>“, като съответно за тях има команда „undo“, чрез която да се премахне последно направената трансформация.

Ако в сесия има само черни бели изображения и се направи опит да си приложи командата „grayscale“ или „monochrome“, то се извежда съобщение, че не е извършена промяна и това не се брои за трансформация. Ако в заредената сесия има поне едно цветно изображение, без значение дали другите са черно-бели, то трансформация се извършва само върху цветното изображение.

При натискане на команда „save/saveas“ трансформациите се занулят и по този начин след като изображението е вече записано, то не може да се използва командата „undo“ и да се върнат промени.

- В **клас Редактор** са имплементирани:
  - Много помощни функции, които са в „private“ частта на класа и се използват само в този клас.
  - Една основна функция, която работи с командния ред на приложението.
  - При стартиране на приложението то работи, докато не се напише командата „exit“, която излиза от програмата.

3.2 Примери:

- Пример 1:

- Създавам сесия с идентификационен номер 1, в която се добавят три изображения, които са коректни.
- Прилага се команда „grayscale“, която се изпълнява само върху изображението от „.ppm“.
- Команда за завъртане наляво.
- Извеждане на информация за текущата сесия.
- Запазване на изображенията.
- Извеждане на информация за текущата сесия като се забелязва, че след запазването, няма вече трансформации.

```
>load firstPPM.ppm secondPGM.pgm thirdPBM.pbm
Session with ID: 1 started!
Image firstPPM.ppm added!
Image secondPGM.pgm added!
Image thirdPBM.pbm added!
>grayscale
Cannot edit image with path: secondPGM.pgm
Cannot edit image with path: thirdPBM.pbm
>rotate left
>session info
Session ID: 1
Name of images in the session: firstPPM.ppm, secondPGM.pgm, thirdPBM.pbm,
Pending transformations: grayscale, rotate left,
>save
Successfully saved firstPPM.ppm
Successfully saved secondPGM.pgm
Successfully saved thirdPBM.pbm
>session info
Session ID: 1
Name of images in the session: firstPPM.ppm, secondPGM.pgm, thirdPBM.pbm,
Pending transformations:
>close
Successfully closed!
>exit
Exiting the program...
```

- Пример 2:

- Зареждам 2 изображения като в първото има интервал.
- Създавам колаж с коректни данни.
- Опитвам се да създам колаж с некоректно име.
- Опитвам се да създам колаж от различни формати.
- Опитвам се да създам колаж с изображения, които не са в сесията.

```
>load feep ppm.ppm thirdPGM.pgm
Session with ID: 1 started!
Image feep ppm.ppm added!
Image thirdPGM.pgm added!
>collage vertical feep ppm.ppm feep ppm.ppm NEWFEEP1.ppm
New collage "NEWFEEP1.ppm" created!
>collage horizontal feepppm.ppm feepppm.ppm NEWFAIL.ppm
Unable to make collage!
>collage vertical feep ppm.ppm thirdPGM.pgm NEWpgm.pgm
Unable to make collage!
>collage horizontal firstPPM.ppm firstPPM.ppm NEWppm.ppm
Unable to make collage!
>exit
Exiting the program...
```

4. Заключение

Смятам, че успях да осъществя първоначалните си цели и идеи. В бъдеще, когато придобия повече познания, този проект може да се доусъвършенства, но с знанията ми до тук, съм способна само на това. 😊