Приложение за работа с векторни изображения

Изготвил:Даниел Златинов Янев №45801

Глава 1

1. Идеята на този проект е да работи със фигури, които могат да бъдат наредени във векторни изображения. Тези фигури могат да бъдат прочетени от файл или да се създадат ръчно от потребителя, като всяка от тях си има слой, на който лежи(т.е фигурите могат да се припокриват). Върху въпросните фигури могат да бъдат прилагани операции за транслиране, скалриане, групиране, ротация, промяна на слой и запазване на промените обратно на диска.

2. Приложението поддържа четири онсовни фигури. Това са отсечка, окръжност, правоъгълник и полигон(многоъгълник). Всяка една от тези фигури се опрделя от ключови точки и величини , като отсечката с двата си края, окръжността със центъра и радиуса си, правоъгълникът с горната лява точка,ширината и височината си и полгионът със точките на краищата си. Всички фигури имат цвят на контур и цвят на запълване. Целта на приложението е да може да извършва гореспоменатите операции, при нужда да може лесно да бъде добавена поддръжка на нов вид фигура и промените направени от приложението, по време на работа, да могат да бъдат запазени във избран от нас формат и във графичен формат за визуализация.

3. Структурата на документацията е замислена да бъде почти идентична с примерната струкутра, показана във файла с темите на проектите.

Глава 2

1. Във Глава 2, точка 2 дадохме дефиниции за основните фигури, с които ще работи приложението. Алгоритмите, които ще използваме, са: зареждане на изображението от файл, създаване на фигура чрез входни данни от потребителя, принтиране на елементите в изображението, транслация на точки чрез указан вектор,скалиране на величините, чрез указани коефициенти, ротацията завърта елементите чрез указан ъгъл, промяна на слой ще разменя слоевете на различните елементи в изображението и групиране на елементи. Групирането поражда нов елемент(група), в които операциите зададени за него се прилагата върху всички елементи на групата.

Глава 3

1. Архитектурата която приложението ще използва е следната

- Ще имаме струкутара *Point* - представлява наредена двойка от числа с плаваща запетая и функционалности за транслация и сравнение.

- Базов клас *Element* - той е основата на всички елементи в приложението.Абстрактен клас със всички функционалности, почти всички от които дефинирани като чисто виртуални. Функционалонстите които съдържа са: *print, bringForward, sendBackward, translate, scale ,rotate isContained, saveToFormat, saveToSvg, getId, ungroup*. Съдържа число *id*, което указва на кой слой се намира елемента и два обекта от тип *Point*, които указват горният ляв ъгъл и долният десен ъгъл на елемента.

- Базов клас *Shape*, наследяващ *Element* - основният клас за фигурите във изображението. Съдържа в себе си обект от тип *std::vector<Point>*, който представлява ключовите точки на фигурата и два обекта от тип *std::string,* представляващи цвета на контура и цвета на запълването.

- клас *ShapeCreator* - този клас е помощен за създаване на фигура. Той е базов клас за създателите на фигурите наследници на класа *Shape,* като в него се съдържа низ, който показва каква точно фигура ще създава и методите му са чисто виртуални. Функционалностите които съдържа са: *userCreateShape, formatCreateShape, svgCreateShape*. Създаването е от файл или от потребителски вход.

- клас *Line*, наследяващ *Shape* - това е репрезентацията на фигурата отсечка. Той презаписва всички виртуални функции от *Element*, така че операциите да се изпълняват коректно за отсечка. Съдържа две променливи от положителен целочислен тип, които служат за индексиране на първата и втората точка на отсечката.

- клас *LineCreator*, наследяващ *ShapeCreator* - това е помощен клас за създаване на обект от тип *Line*. Той презаписва всички методи от *ShapeCreator* така че създаването на отсечката да е коректно.

- клас *Circle*, наследяващ *Shape* - това е репрезентацията на фигурата окръжност. Той презаписва всички виртуални функции от *Element*, така че операциите да се изпълняват коректно за окръжността. Съдържа една константна статична променлива от целочислен тип, която служи за индексиране на центъра на окръжността и едно число с плаваща запетая, представляващо радиуса на окръжността.

- клас *CircleCreator*, наследяващ *ShapeCreator* - това е помощен клас за създаване на обект от тип *Circle*. Той презаписва всички методи от *ShapeCreator* така че създаването на окръжността да е коректно.

- клас *Rectangle*, наследяващ *Shape* - това е репрезентацията на фигурата правоъгълник. Той презаписва всички виртуални функции от *Element*, така че операциите да се изпълняват коректно за правоъгълника. Съдържа две числа с плаваща запетая, указващи ширината и височината на правоъгълника, както и две положителни целочислени променливи, които служат за индексация на горният ляв и долният десен ъгъл.

- клас *RectangleCreator*, наследяващ *ShapeCreator* - това е помощен клас за създаване на обект от тип *Rectangle*. Той презаписва всички методи от *ShapeCreator* така че създаването на правоъгълника да е коректно.

- клас *Polygon*, наследяващ *Shape* - това е репрезентацията на фигурата полигон. Той презаписва всички виртуални функции от *Element*, така че операциите да се изпълняват коректно за полигона.

- клас *PolygonCreator*, наследяващ *ShapeCreator* - това е помощен клас за създаване на обект от тип *Polygon*. Той презаписва всички методи от *ShapeCreator* така че създаването на полигона да е коректно.

- клас *ShapeFactory* - представлява ООП дизайн фабрика. В него се съдържат всички класове наследяващи *ShapeCreator*. Когато е зададена някаква фигура за създаване, фабриката намира, адекватният създател, които връща новосъздадената фигура.

- клас *Group,* наследяващ *Element* - този елемент се получава от операцията групиране.В него се съдържат всички елементи които се съдържат в указаният обсег от координати. Той презаписва така операцците от *Element*, че се прилагат на всички елементи в него. Може да бъде прекратено групирането. В такъв случай всички елементи се връщат към позицията си в изображението и елемента група се изтрива. Съдържа обект от тип *std::vector<Element *>*, който представлява контейнер от различни елементи. Поддържа се и "група в група".

- клас *Canvas* - това е класът репрезентиращ изображението. В него се съдържат всички текущи елементи и от него могат да бъдат прилагани гореспоменатите операции.Съдържа обект от тип *std::vector<Element *>* със същата логика като групата.

- клас *Interface* - това е клас който изпълнява команди от потребителя, за избор на желаните операции над изображението(*Canvas).*

Глава 4

1. Реализацията на класовете се получава чрез спазването на всички ООП стълбове.Абстракция, капсулация, преизползване и полиморфизъм. Чрез реализираната йерархия, ако е нужно в някакъв момент да се добави поддръжка на нова фигура, това може да се случи лесно, стига да се дефинира клас за новата фигура със съответният и създател(също така да бъде добавен във фабриката).

2. В този проект съм решил да заложа на готовите контейнери *std::vector* и *std::string* от *STL*, което би могло доста да забави програмата, но понеже е замислено да се работи с указатели към вече създадени обекти, вместо да се копират се избягва голяма част от това забавяне(освен при PolygonCreator ,понеже там се използва за четене на ключовите точки от файл).

Глава 5

1. Всяка изискана фунцкионалност е покрита, освен ротацията.

2. За развитие на проекта, би могло да се използват различни контейнери от тези на *STL* за максимална ефективност и също така да бъде реализиран графичен интерфейс в който да се изобразяват фигурите в реално време.