ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

**Факультет информатики, математики и компьютерных наук**

**Программа подготовки бакалавров по направлению   
09.03.04 «Программная инженерия»**

*Крайнов Дмитрий Сергеевич*

**КУРСОВАЯ РАБОТА**

«Создание простого графического редактора»

|  |  |
| --- | --- |
|  | Научный руководитель  Старший преподаватель  С. В. Кряжев |

Нижний Новгород, 2017

Содержание

1. Постановка задачи 2

2. Описание основной структуры 3

[3.](#_Toc445601199)Описание функционала 3

[4. Листинг программы.](#_Toc445601213) 4

[5. Результаты.](#_Toc445601213) 5

1. **Постановка задачи.**

Разработать простой графический редактор на языке C++, который должен включать в себя следующие функции:

1. Создание рисунка:

а) в режиме ручной прорисовки (карандаш);

б) с использованием панели инструментов (примитивов):

- прямоугольник

- эллипс

- линия

- ломаная линия

с возможностью их перемещения, изменения их размеров и установкой границ с определённой толщиной (для прямоугольника и эллипса).

2. Манипулирование рисунком:

а) выделение фрагментов рисунка с последующим перемещением;

б) масштабирование изображения;

в) копирование/вырезание/удаление фрагмента;

г) заливка отдельных частей рисунка цветом;

д) пипетка (установка цвета как у пикселя, по координатам которого было произведено нажатие)

е) изменение размеров изображения;

ж) включение/выключение сглаживания.

3. Ввод в изображение текста:

а) Выбор шрифта;

б) Выбор символов (курсив, подчёркивание).

4. Работа с цветами

а) Создание своей палитры цветов.

5. Работа с внешними устройствами

а) Сохранение рисунка в виде файла стандартного формата (pcx, bmp, tif, gif, jpg, png и др.);

б) Открытие файла в редакторе для редактирования или просмотра.

Инструментарий для разработки – Qt

Приложение ориентировано на платформу Windows.

1. **Описание основной структуры.**

Основой приложения является класс (виджет) ImageArea в котором реализованы общие обработчики событий. Он отображается пользователю вместе с полотном для рисования, в роли которого выступает объект встроенного в Qt класса QImage.

Каждый инструмент выделен в свой класс:

* Прямоугольник - RectangleInstr
* Эллипс - EllipseInstr
* Линия – LineInstr
* Ломаная линия - PolygonChainInstr
* Заливка - FillInstr
* Карандаш – PencilInstr
* Масштабирование - ZoomInstr
* Пипетка - PipetteInstr
* Текст – TextInstr
* Вырезать фрагмент – CutInstr

Все классы наследуются от CommonInstr в котором описываются общие для всех инструментов методы. Указатели на объекты этих классов занесены в массив по соответствующем им индексам.

При нажатии клавишей мыши по полотну событие сначала обрабатывается в основном классе ImageArea, после вызывает обработчик выбранного инструмента, который в свою очередь вносит изменения на самом полотне.

1. **Описание функционала.**

Класс CommonInstr имеет следующий вид:

class CommonInstr : public QWidget

{

Q\_OBJECT

public:

explicit CommonInstr(QWidget \*parent = 0);

virtual void *use*() = 0;

virtual void *mousePress*(QMouseEvent\*) = 0;

virtual void *mouseMove*(QMouseEvent \*) = 0;

virtual void *mouseRelease*(QMouseEvent\*) = 0;

void setStartPoint(QPoint);

void setEndPoint(QPoint);

QPoint getStartPoint();

QPoint getEndPoint();

protected:

QPoint start,end;

ImageArea\* imageArea;

};

Как было написано выше, все остальные инструменты наследуются от него. Большинство из них для выполнения своей задачи используют класс QPainter, который предоставляет возможность низкоуровневого рисования на виджетах и других устройствах (drawRect(…), drawEllipse(…), drawLine(…), и др.).

1. Создание рисунка:

а) в режиме ручной прорисовки (карандаш):

Данный инструмент закрашивает по одному пикселю по отслеживаемому пути движения мыши, но так как при быстром движении мыши событие не отлавливает каждый пиксель, то инструмент соединяет их линиями с помощью

painter.drawLine(QLine(start,end)), где start – предыдущая точка из события движения мышью, end – текущая.

б) Использование примитивов:

Для рисования примитивов создаётся слой над основным полотном, на котором рисуется сам объект. Это облегчает в дальнейшем перемещение и изменение размеров фигуры, а именно, для выполнения задачи достаточно работать только с фигурой, не затрагивая основное изображение.

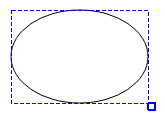
Для удобного перемещения фигура обводится тонкой пунктирной границей, а для изменения размера имеется небольшой квадрат за пределами фигуры.

Рис. 1

Инструмент «Ломаная линия» отслеживает нажатые левой клавишей мыши точки и соединяет их линиями. Нажатие правой клавишей останавливает процесс рисования. С помощью данного инструмента можно рисовать собственный многоугольники (замкнутые ломаные).

Метод use() в классе RectangleInstr:

void RectangleInstr::*use*()

{

imageArea->setChangeFlag(true); // Для отображения изменений

// Присваивание основному изображению копию, которая была сохранена

// до применения метода

\*(imageArea->getImage()) = \*(imageArea->getImageCopy());

// Рисование на новом слое

QPainter painter(imageArea->getPartOfImage());

// Проверка состояния флага сглаживания

painter.setRenderHint(QPainter::Antialiasing,imageArea->isAntialiasing());

painter.setPen(imageArea->getPen());

painter.setBrush(imageArea->getBrush());

painter.drawRect(QRect(start,end));

painter.end();

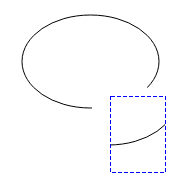
imageArea->update();

}

2. Манипулирование рисунком:

а) выделение фрагментов рисунка с последующим перемещением:

Выделенный фрагмент рисуется на новом слое, а на его старом месте на основном полотне остаётся белый прямоугольник. Дальнейшее перемещение реализовано так же, как и у примитивов.

Рис. 2

б) масштабирование изображения:

Данный инструмент реализован с помощью следующего метода:

QImage QImage::scaled(const QSize &size, Qt::AspectRatioMode aspectRatioMode = Qt::IgnoreAspectRatio, Qt::TransformationMode transformMode = Qt::FastTransformation) const

Он позволяет масштабировать изображение, соблюдая пропорции при aspectRatioMode = [Qt::KeepAspectRatio](http://doc.qt.io/qt-5/qt.html#AspectRatioMode-enum), и возвращает копию полученного объекта. Чтобы «не портить» изображение, создаётся его копия imageCopyForZoom = image->scaled(scaledSize,Qt::KeepAspectRatio)

, и при изменении масштаба пользователю отображается именно она, при этом дальнейшее рисование продолжается на основном рисунке.

в) копирование/вырезание/удаление фрагмента: