Специальность: 09.02.07 Информационные системы и программирование

Проверил:

\_\_\_\_\_\_\_\_\_\_\_ Антонов М.Б.

Дата: «\_\_» \_\_\_\_\_\_\_\_ 2020 г.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

(Университет ИТМО)

факультет среднего профессионального образования

Отчет

о Лабораторной работе №7

по теме: Простейший WebSocket Server чата

по дисциплине: Разработка программных модулей

Выполнила:

студентка группы 2433

Тертень Д. В.

Санкт-Петербург 2020

Содержание

[1 Постановка задачи 3](#_Toc35698558)

[1.1 Цель 3](#_Toc35698559)

[1.2 Задачи 3](#_Toc35698560)

[1.3 Задание 3](#_Toc35698561)

[2 Диаграммы 5](#_Toc35698562)

[3 Исходный код 6](#_Toc35698563)

[4 ЛИСТИНГ ВЫполнения программы 11](#_Toc35698564)

[5 Вывод 12](#_Toc35698565)

1. Постановка задачи
   1. Цель

Познакомиться с основами работы графических сцен и основы рисования в Qt.

* 1. Задачи

1. изучить основы работы с QPaint;
2. Создать простейшее приложение для рисования.
   1. Задание

Требуется создать приложение для рисования по типу кисти Paint мышкой.

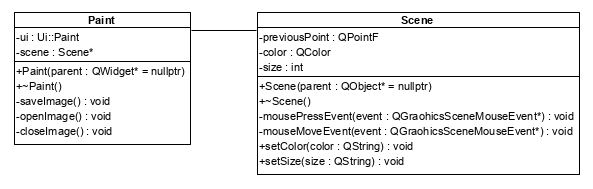
Требуется создать виджет для выбора текущего цвета рисования.

Результат рисования можно сохранять в .bmp файл или любой другой формат изображения, используя QFileDialog.

Можно открыть png изображение и отредактировать его.

* 1. Общее ограничение на лабораторную работу
* использовать разбитие на файлы основной программы (в main.cpp только функция main.cpp);
* документировать весь код используя аннотации Doxygen.
  1. Требования к отчёту
* цели лабораторной работы;
* задачи лабораторной работы;
* выполняемое задание;
* диаграмма(ы) классов;
* диаграммы деятельности (для методов и/или функций больше 10 строк кода (исключая форматирования и пустые строки));
* исходный код программы с комментариями (2 пробела вместо 4, 10pt, моноширинный шрифт);
* листинг выполнения программы;
* исходный код в виде файлов для запуска на произвольном ПК.

1. Диаграммы



*Рисунок 1 – Диаграмма классов*

1. Исходный код

Листинги 2.1-2.6 – исходный код серверной части.

Листинг 2.1 – Код программы

/\*\*

\* @file main.cpp

\* @author Тертень Даниил

\*/

#include "paint.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

Paint w;

w.show();

return a.exec();

}

Листинг 2.2 – Заголовочный файл paint.h

/\*\*

\* @file paint.h

\* @brief window for painting

\* @author Тертень Даниил

\*/

#ifndef PAINT\_H

#define PAINT\_H

#include <QWidget>

#include <QFileDialog>

#include <scene.h>

QT\_BEGIN\_NAMESPACE

namespace Ui { class Paint; }

QT\_END\_NAMESPACE

class Paint : public QWidget

{

Q\_OBJECT

private:

Ui::Paint \*ui;

Scene\* scene;

public:

Paint(QWidget \*parent = nullptr);

~Paint();

private slots:

/\*\*

\* @brief saveImage

\*/

void saveImage();

/\*\*

\* @brief openImage

\*/

void openImage();

/\*\*

\* @brief closeImage

\*/

void closeImage();

};

#endif // PAINT\_H

Листинг 2.3 – Заголовочный файл paint.cpp

/\*\*

\* @file paint.cpp

\* @brief window for painting

\* @author Тертень Даниил

\*/

#include "paint.h"

#include "ui\_paint.h"

#include <QMessageBox>

Paint::Paint(QWidget \*parent)

: QWidget(parent), ui(new Ui::Paint)

{

ui->setupUi(this);

scene = new Scene();

ui->graphicsView->setScene(scene);

QCursor cursor = QCursor(QPixmap(":/Brush.png"));

ui->graphicsView->setCursor(cursor);

connect(ui->color, &QComboBox::currentTextChanged,

scene, &Scene::setColor);

connect(ui->size, &QLineEdit::textChanged,

scene, &Scene::setSize);

connect(ui->save, &QPushButton::clicked,

this, &Paint::saveImage);

connect(ui->open, &QPushButton::clicked,

this, &Paint::openImage);

connect(ui->close, &QPushButton::clicked,

this, &Paint::closeImage);

}

Paint::~Paint()

{

delete ui;

}

void Paint::saveImage()

{

QString fileName = QFileDialog::getSaveFileName

(this, tr("Сохранить файл"),"",

tr("Images(\* .png \* .jpg)"));

QImage image(ui->graphicsView->scene()->width(),

ui->graphicsView->scene()->height(),

QImage::Format\_RGB32);

image.fill(QColor(Qt::white).rgb());

QPainter painter(&image);

ui->graphicsView->scene()->render(&painter);

painter.end();

if(fileName.indexOf(".png") != -1 || fileName.indexOf(".jpg") != -1)

image.save(fileName);

else

image.save(fileName.append(".png"));

}

void Paint::openImage()

{

QString fileName = QFileDialog::getOpenFileName

(this, tr("Открыть файл"),"",

tr("Images(\* .png \* .jpg)"));

Scene\* sceneImage = new Scene;

sceneImage->addPixmap(QPixmap(fileName));

ui->graphicsView->setScene(sceneImage);

sceneImage->setSize(ui->size->displayText());

sceneImage->setColor(ui->color->currentText());

connect(ui->color, &QComboBox::currentTextChanged,

sceneImage, &Scene::setColor);

connect(ui->size, &QLineEdit::textChanged,

sceneImage, &Scene::setSize);

QMessageBox::information(NULL, QObject::tr("Информация"), tr("Чтобы сохранить нажмите save"));

}

void Paint::closeImage()

{

ui->graphicsView->setScene(this->scene);

}

Листинг 2.4 – Заголовочный файл scene.h

/\*\*

\* @file scene.h

\* @brief scene for painting

\* @author Тертень Даниил

\*/

#ifndef SCENE\_H

#define SCENE\_H

#include <QGraphicsScene>

#include <QGraphicsSceneMouseEvent>

class Scene : public QGraphicsScene

{

Q\_OBJECT

private:

QPointF previousPoint;

QColor color;

int size;

public:

explicit Scene(QObject \*parent = nullptr);

~Scene();

private:

/\*\*

\* @brief mousePressEvent

\* @param event

\*/

void mousePressEvent(QGraphicsSceneMouseEvent \*event);

/\*\*

\* @brief mouseMoveEvent

\* @param event

\*/

void mouseMoveEvent(QGraphicsSceneMouseEvent \*event);

public slots:

/\*\*

\* @brief setColor

\* @param color

\*/

void setColor(QString color);

/\*\*

\* @brief setSize

\* @param size

\*/

void setSize (QString size);

};

#endif // SCENE\_H

Листинг 2.5 – Заголовочный файл scene.cpp

/\*\*

\* @file scene.cpp

\* @brief scene for painting

\* @author Тертень Даниил

\*/

#include "scene.h"

Scene::Scene(QObject \*parent) : QGraphicsScene(parent){}

Scene::~Scene(){}

void Scene::mousePressEvent(QGraphicsSceneMouseEvent \*event)

{

addEllipse(event->scenePos().x()-(size/2),

event->scenePos().y()-(size/2),

size, size, QPen(Qt::NoBrush), QBrush(color));

previousPoint = event->scenePos();

}

void Scene::mouseMoveEvent(QGraphicsSceneMouseEvent \*event)

{

addLine(previousPoint.x(), previousPoint.y(),

event->scenePos().x(), event->scenePos().y(),

QPen(color, size,Qt::SolidLine,Qt::RoundCap));

previousPoint = event->scenePos();

}

void Scene::setColor(QString color)

{

this->color = color;

}

void Scene::setSize(QString size)

{

this->size = size.toInt();

}

Листинг 2.6 – Файл res.qrc

<RCC>

<qresource prefix="/">

<file>brush.png</file>

</qresource>

</RCC>

1. ЛИСТИНГ ВЫполнения программы

На рисунке 2 представлен интерфейс работы программы.

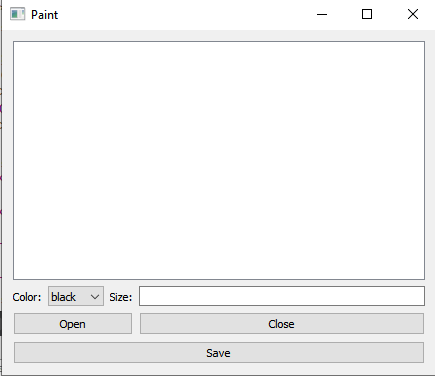


Рисунок 2 – Интерфейс программы

1. Вывод

В ходе выполнения лабораторной работы была написана программа, которая реализует возможность рисования кистью и смены цвета.