

1. Цель и задачи

Цель: познакомиться с основами работы графических сцен и основы рисования в Qt.

Задачи:

* изучить основы работы с QPaint;
* создать простейшее приложение для рисования.

1. Задание

Общее ограничение на лабораторную работу:

* использовать разбитие на файлы основной программы (в main.cpp только функция main.cpp);
* документировать весь код используя аннотации Doxygen.

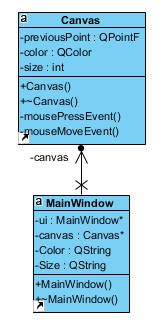
Требуется создать приложение для рисования по типу кисти Paint мышкой.

Требуется создать виджет для выбора текущего цвета рисования.

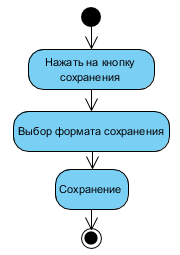
Результат рисования можно сохранять в .bmp файл или любой другой формат изображения используя QFileDialog.

Можно открыть png изображение и отредактировать его.

1. ДИАГРАММЫ



*Рисунок 1 – диаграмма классов*

**

*Рисунок 2 – диаграмма деятельности сохранения*

1. КОд программы

Основной выполняемый файл main.cpp:

#include "mainwindow.h"

#include <QApplication>

/\*\*

\* @file main.cpp

\* @author Oreshnikov Egor

\*/

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

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

Файл mainwindow.cpp:

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "QTextStream"

#include "QMessageBox"

/\*\*

\* @file mainwindow.cpp

\* @author Oreshnikov Egor

\*/

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

color();

}

MainWindow::~MainWindow()

{

delete ui;

}

/\*\*

\* @brief сохранить файл

\* @author Oreshnikov Egor

\*/

void MainWindow::on\_Save\_clicked()

{

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"));

}

/\*\*

\* @brief открытие или создание нового файла

\* @author Oreshnikov Egor

\*/

void MainWindow::on\_Open\_clicked()

{

QString fileName;

QMessageBox::StandardButton reply;

reply = QMessageBox::question(this, "Открыть", "Создать новый файл?",

QMessageBox::Yes|QMessageBox::No);

if (reply == QMessageBox::Yes) {

QString path = QCoreApplication::applicationDirPath();

fileName = path+"NewCanvas.jpg";

QFile mFile(path+"NewCanvas.jpg");

mFile.open(QIODevicCje::WriteOnly);

}

else

{

fileName = QFileDialog::getOpenFileName

(this, tr("Открыть файл"),"",

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

}

Canvas\* sceneImage = new Canvas;

sceneImage->addPixmap(QPixmap(fileName));

ui->graphicsView->setScene(sceneImage);

sceneImage->setSize(ui->Size->text());

sceneImage->setColor(ui->ColorPen->text());

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

sceneImage, &Canvas::setSize);

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

sceneImage, &Canvas::setColor);

}

/\*\*

\* @brief установка размера

\* @author Oreshnikov Egor

\*/

void MainWindow::setSize(){

canvas->setSize(ui->Size->text());

}

/\*\*

\* @brief закрыть файл

\* @author Oreshnikov Egor

\*/

void MainWindow::on\_Close\_clicked()

{

QMessageBox::StandardButton reply;

reply = QMessageBox::question(this, "Закрыть", "Закрыть без сохранение?",

QMessageBox::Yes|QMessageBox::No);

if (reply == QMessageBox::Yes)

{

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

}

}

Файл canvas.cpp:

#include "canvas.h"

#include "mainwindow.h"

#include <QMessageBox>

/\*\*

\* @file canvas.cpp

\* @author Oreshnikov Egor

\*/

Canvas::Canvas(QObject \*parent) : QGraphicsScene(parent){

this->setSize("1");

color = QColor(Qt::green);

}

Canvas::~Canvas(){}

void Canvas::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 Canvas::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 Canvas::setColor(QString color){

this->color = color;

}

void Canvas::setSize(QString size){

if(size.toInt()>0)

this->size = size.toInt();

else{

QMessageBox::information(NULL,QObject::tr("Ошибка"), tr("Размер не может равняться 0"));

return; }

}

Файл canvas.h:

#ifndef CANVAS\_H

#define CANVAS\_H

#include <QGraphicsScene>

#include <QGraphicsSceneMouseEvent>

class Canvas : public QGraphicsScene

{

Q\_OBJECT

private:

QPointF previousPoint;

QColor color;

int size;

public:

explicit Canvas(QObject \*parent = nullptr);

~Canvas();

private:

void mousePressEvent(QGraphicsSceneMouseEvent \*event);

void mouseMoveEvent(QGraphicsSceneMouseEvent \*event);

public slots:

void setColor(QString color);

void setSize (QString size);

};

#endif // CANVAS\_H

Файл mainwindow.h:

/\*\*

\* @file mainwindow.h

\* @author Oreshnikov Egor

\*/

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QFileDialog>

#include <canvas.h>

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private slots:

void on\_Open\_clicked();

void on\_Save\_clicked();

void on\_Close\_clicked();

void setSize();

private:

Ui::MainWindow \*ui;

Canvas\* canvas;

QString Color = "Black";

QString Size = "1";

};

#endif // MAINWINDOW\_H

В качестве результата работы программы представлен скриншот выходных файлов на рисунке 3.



*Рисунок 3 – результат работы программы*

# ВЫВОД

В процессе выполнения лабораторной работы были получены навыки работы с QtPaint.