МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное
образовательное учреждение высшего образования
**«Балтийский государственный технический университет «ВОЕНМЕХ»
им. Д.Ф. Устинова»
(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова»)**

Факультет О (шифр) Естественнонаучный (наименование)

Кафедра О7 (шифр) Информационные системы и программная инженерия (наименование)

Дисциплина Визуальное программирование

Лабораторная работа №4

«Работа с событиями и графическим представлением в Qt»

ВЫПОЛНИЛ студент группы И596

Орехов. Р. В. .
Фамилия И.О.

ПРЕПОДАВАТЕЛЬ

Васюков В.М. .
Фамилия И.О.

САНКТ-ПЕТЕРБУРГ
2022 г.

ЗАДАНИЕ

Общая постановка задачи.

В данной работе необходимо написать программу в соответствие с индивидуальным вариантом. Все варианты описывают набор объектов для графической сцены (QGraphicsScene). Если в варианте не сказано иного, то должны использоваться объекты трёх разных классов, унаследованных от QGraphicsItem. У каждого из этих классов должны быть переопределена функция paint.

В зависимости от варианта либо для каждого из классов должна быть переопределена функция обработки указанного в варианте события, либо должен быть создан класс для фильтрации событий и подключён к объектам.

Необходимо реализовать возможность выбора типа объекта для добавления на сцену, добавление неограниченного числа объектов любого из указанных в задании классов на сцену, удаление объекта (в зависимости от варианта либо через список, либо через обработку события), выбор и перемещение объекта. Если в качестве объекта используется стандартный виджет, то он должен выполнять указанную в варианте функцию.

Некоторые варианты заданий могут противоречить написанному выше, в этом случае приоритет у написанного в варианте.

Варианты заданий (номер варианта 10).

Написать программу, позволяющую разместить на графической сцене произвольное число объектов, следующих трёх типов:

1. Текст, предварительно введённый пользователем (для каждого объекта свой);
2. Квадрат одного из трёх цветов: синий кадетский, тёмный лосось, помидор по выбору пользователя;
3. Изображение одного из трёх греческих богов: Зевса, Посейдона или Аида.

Должно быть предусмотрено перемещение добавляемых объектов путём перетаскивания их левой кнопкой мыши.

Также должен быть представлен список из всех добавленных элементов, с возможностью выбора и удаления любого из них, а также изменения бога, текста или цвета.

С помощью фильтра событий, должна быть добавлена возможность выбирать объект в списке при нажатии на него правой кнопкой мыши на сцене.

ЭТАПЫ ВЫПОЛНЕНИЯ РАБОТЫ

Текст файла main.cpp:

```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[]) {
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

Текст файла mainwindow.h:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QAction>
#include <QLabel>
#include <QLineEdit>
#include <QListWidget>
#include <QComboBox>
#include <QHBoxLayout>
#include <QPushButton>
#include <QList>
#include <QDebug>

#include <QGraphicsView>
#include <QGraphicsScene>

#include "square.h"
#include "god.h"
#include "text.h"

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

struct NODE {
    int type;
    SQUARE * square;
    GOD * god;
    TEXT * text;
};

class MainWindow : public QMainWindow {
    Q_OBJECT

    QWidget * centralwidget;

    QGridLayout * gridLayout;

    QVBoxLayout * verticalLayoutLeft;

    QHBoxLayout * horizontalLayoutText;
    QLabel * labelText;
    QLineEdit * lineEdit;

    QHBoxLayout * horizontalLayoutSquare;
```

```
    QLabel * labelSquare;
    QComboBox * comboBoxSquare;

    QHBoxLayout * horizontalLayoutGod;
    QLabel * labelGod;
    QComboBox * comboBoxGod;

    QHBoxLayout *horizontalLayoutButton1;
    QPushButton *addText;
    QPushButton *addSquare;

    QHBoxLayout *horizontalLayoutButton2;
    QPushButton *addGod;
    QPushButton *deleteObject;

    QListWidget * listWidget;

    QGraphicsScene * scene;
    QGraphicsView * graphicView;

    QStatusBar * statusbar;

public:
    MainWindow(QWidget * parent = nullptr);
    ~MainWindow();

private slots:
    void addSquare_clicked();
    void addGod_clicked();
    void addText_clicked();
    void deleteObject_clicked();

    void comboBoxGod_activated();
    void comboBoxSquare_activated();
    void lineEdit_textEdited();

    void scene_selectionChanged();
    void listWidget_itemClicked();

private:
    QList <NODE> objectList;
    QStringList stringList;

    int lastIndex;

    Ui::MainWindow * ui;

    void clearWidgetList();
    void viewWidgetList();
};

#endif // MAINWINDOW_H
```

Текст файла mainwindow.cpp:

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent): QMainWindow(parent), ui(new
Ui::MainWindow) {
    ui->setupUi(this);
```

```
centralwidget = new QWidget(this);

gridLayout = new QGridLayout(centralwidget);
verticalLayoutLeft = new QVBoxLayout(centralwidget);

horizontalLayoutText = new QHBoxLayout(centralwidget);
labelText = new QLabel(centralwidget);
labelText->setMinimumSize(QSize(80, 0));
labelText->setMaximumSize(QSize(80, 16777215));
labelText->setText("Текст");
lineEdit = new QLineEdit(centralwidget);
horizontalLayoutText->addWidget(labelText);
horizontalLayoutText->addWidget(lineEdit);
verticalLayoutLeft->addLayout(horizontalLayoutText);

horizontalLayoutSquare = new QHBoxLayout(centralwidget);
labelSquare = new QLabel(centralwidget);
labelSquare->setMinimumSize(QSize(80, 0));
labelSquare->setMaximumSize(QSize(80, 16777215));
labelSquare->setText("Цвет квадрата");
comboBoxSquare = new QComboBox(centralwidget);
comboBoxSquare->addItem("Синий кадетский");
comboBoxSquare->addItem("Темный лосось");
comboBoxSquare->addItem("Помидор");
horizontalLayoutSquare->addWidget(labelSquare);
horizontalLayoutSquare->addWidget(comboBoxSquare);
verticalLayoutLeft->addLayout(horizontalLayoutSquare);

horizontalLayoutGod = new QHBoxLayout(centralwidget);
labelGod = new QLabel(centralwidget);
labelGod->setMinimumSize(QSize(80, 0));
labelGod->setMaximumSize(QSize(80, 16777215));
labelGod->setText("Бог");
comboBoxGod = new QComboBox(centralwidget);
comboBoxGod->addItem("Зевс");
comboBoxGod->addItem("Посейдон");
comboBoxGod->addItem("Аид");
horizontalLayoutGod->addWidget(labelGod);
horizontalLayoutGod->addWidget(comboBoxGod);
verticalLayoutLeft->addLayout(horizontalLayoutGod);

horizontalLayoutButton1 = new QHBoxLayout(centralwidget);
addText = new QPushButton(centralwidget);
addText->setText("Добавить текст");
addSquare = new QPushButton(centralwidget);
addSquare->setText("Добавить квадрат");
horizontalLayoutButton1->addWidget(addText);
horizontalLayoutButton1->addWidget(addSquare);
verticalLayoutLeft->addLayout(horizontalLayoutButton1);

horizontalLayoutButton2 = new QHBoxLayout(centralwidget);
addGod = new QPushButton(centralwidget);
addGod->setText("Добавить бога");
deleteObject = new QPushButton(centralwidget);
deleteObject->setText("Удалить объект");
horizontalLayoutButton2->addWidget(addGod);
horizontalLayoutButton2->addWidget(deleteObject);
verticalLayoutLeft->addLayout(horizontalLayoutButton2);

listWidget = new QListWidget(centralwidget);
verticalLayoutLeft->addWidget(listWidget);

scene = new QGraphicsScene(centralwidget);
```

```
    scene->setSceneRect(0, 0, 550, 550);
    graphicView = new QGraphicsView(centralwidget);
    graphicView->sceneRect();
    graphicView->setScene(scene);
    graphicView->setRenderHint(QPainter::Antialiasing);
    graphicView->setCacheMode(QGraphicsView::CacheBackground);
    graphicView-
>setViewportUpdateMode(QGraphicsView::BoundingRectViewportUpdate);

    gridLayout->addLayout(verticalLayoutLeft, 1, 0, 1, 1);
    gridLayout->addWidget(graphicView, 1, 1, 1, 1);
    gridLayout->setColumnStretch(0, 2);
    gridLayout->setColumnStretch(1, 5);

    statusbar = new QStatusBar(this);

    this->setMinimumSize(QSize(800, 600));
    this->setMaximumSize(QSize(800, 600));
    this->setCentralWidget(centralwidget);
    this->setStatusBar(statusbar);

    connect(addSquare, SIGNAL(clicked()), this, SLOT(addSquare_clicked()));
    connect(addGod, SIGNAL(clicked()), this, SLOT(addGod_clicked()));
    connect(addText, SIGNAL(clicked()), this, SLOT(addText_clicked()));
    connect(deleteObject, SIGNAL(clicked()), this,
SLOT(deleteObject_clicked()));

    connect(listWidget, SIGNAL(itemClicked(QListWidgetItem*)), this,
SLOT(listWidget_itemClicked()));

    connect(comboBoxGod, SIGNAL(activated(int)), this,
SLOT(comboBoxGod_activated()));
    connect(comboBoxSquare, SIGNAL(activated(int)), this,
SLOT(comboBoxSquare_activated()));
    connect(lineEdit, SIGNAL(textEdited(const QString &)), this,
SLOT(lineEdit_textEdited()));

    connect(scene, SIGNAL(selectionChanged()), this,
SLOT(scene_selectionChanged()));
}

MainWindow::~MainWindow() {
    delete ui;

    for(int i = 0; i < objectList.count(); i++)
        switch(objectList.at(i).type) {
            case 0:
                delete(objectList.at(i).text);
                break;

            case 1:
                delete(objectList.at(i).square);
                break;

            case 2:
                delete(objectList.at(i).god);
                break;
        }
}

void MainWindow::clearWidgetList() {
    listWidget->clear();
}
```

```
void MainWindow::viewWidgetList() {
    clearWidgetList();
    listWidget->addItems(stringList);
    if(lastIndex > -1 && lastIndex < stringList.count()) listWidget-
>setCurrentRow(lastIndex);
}

void MainWindow::addText_clicked() {
    TEXT * temp = new TEXT();

    QString tempStr = lineEdit->text();
    if(tempStr.isEmpty()) return;

    temp->setStr(tempStr);
    scene->addItem(temp);

    NODE node;

    node.type = 0;
    node.text = temp;

    objectList.append(node);
    stringList.append(tempStr);

    lastIndex = -1;
    viewWidgetList();
}

void MainWindow::addSquare_clicked() {
    SQUARE * temp = new SQUARE();

    temp->setColor(comboBoxSquare->currentIndex());
    scene->addItem(temp);

    NODE node;

    node.type = 1;
    node.square = temp;

    objectList.append(node);

    switch(comboBoxSquare->currentIndex()) {
        case 0:
            stringList.append("Cadet blue square");
            break;

        case 1:
            stringList.append("Dark salmon square");
            break;

        case 2:
            stringList.append("Tomato color square");
            break;
    }

    lastIndex = -1;
    viewWidgetList();
}

void MainWindow::addGod_clicked() {
    GOD * temp = new GOD();
```

```
    temp->setColor(comboBoxGod->currentIndex());
    scene->addItem(temp);

    NODE node;

    node.type = 2;
    node.god = temp;

    objectList.append(node);

    switch(comboBoxGod->currentIndex()) {
        case 0:
            stringList.append("Zeus");
            break;

        case 1:
            stringList.append("Poseidon");
            break;

        case 2:
            stringList.append("Hades");
            break;
    }

    lastIndex = -1;
    viewWidgetList();
}

void MainWindow::deleteObject_clicked() {
    if(lastIndex < 0 || lastIndex >= listWidget->count()) return;

    NODE temp = objectList.takeAt(lastIndex);
    switch(temp.type) {
        case 0:
            delete(temp.text);
            break;

        case 1:
            delete(temp.square);
            break;

        case 2:
            delete(temp.god);
            break;
    }

    stringList.removeAt(lastIndex);

    lastIndex = -1;
    viewWidgetList();
}

void MainWindow::comboBoxGod_activated() {
    if(lastIndex < 0 || lastIndex >= listWidget->count()) return;

    NODE temp = objectList.at(lastIndex);
    if(temp.type != 2 || temp.god == NULL) return;

    temp.god->setColor(comboBoxGod->currentIndex());

    switch(comboBoxGod->currentIndex()) {
        case 0:
            stringList.replace(lastIndex, "Zeus");
```

```
            break;

        case 1:
            stringList.replace(lastIndex, "Poseidon");
            break;

        case 2:
            stringList.replace(lastIndex, "Hades");
            break;
    }

    objectList.replace(lastIndex, temp);

    scene->update();
    viewWidgetList();
}

void MainWindow::comboBoxSquare_activated() {
    if(lastIndex < 0 || lastIndex >= listWidget->count()) return;

    NODE temp = objectList.at(lastIndex);
    if(temp.type != 1 || temp.square == NULL) return;

    temp.square->setColor(comboBoxSquare->currentIndex());

    switch(comboBoxSquare->currentIndex()) {
        case 0:
            stringList.replace(lastIndex, "Cadet blue square");
            break;

        case 1:
            stringList.replace(lastIndex, "Dark salmon square");
            break;

        case 2:
            stringList.replace(lastIndex, "Tomato color square");
            break;
    }

    objectList.replace(lastIndex, temp);

    scene->update();
    viewWidgetList();
}

void MainWindow::lineEdit_textEdited() {
    if(lastIndex < 0 || lastIndex >= listWidget->count()) return;

    NODE temp = objectList.at(lastIndex);
    if(temp.type != 0 || temp.text == NULL) return;

    QString tempStr = lineEdit->text();
    if(tempStr.isEmpty()) return;

    temp.text->setStr(tempStr);
    stringList.replace(lastIndex, tempStr);
    objectList.replace(lastIndex, temp);

    scene->update();
    viewWidgetList();
}

void MainWindow::scene_selectionChanged() {
```

```
    QList <QGraphicsItem *> listTemp = scene->selectedItems();
    if(!listTemp.count()) return;

    NODE temp;
    int cur = 0;
    bool flagSel = 0;

    for(cur = 0; cur < objectList.count(); cur++) {
        temp = objectList.at(cur);
        switch(temp.type) {
            case 0:
                if(temp.text->isSelected()) {
                    flagSel = 1;
                    lineEdit->setText(temp.text->getStr());
                }
                break;

            case 1:
                if(temp.square->isSelected()) {
                    flagSel = 1;
                    comboBoxSquare->setCurrentIndex(temp.square->getColor());
                }
                break;

            case 2:
                if(temp.god->isSelected()) {
                    flagSel = 1;
                    comboBoxGod->setCurrentIndex(temp.god->getColor());
                }
                break;
        }

        if(flagSel) break;
    }

    lastIndex = cur;
    listWidget->setCurrentRow(cur);
}

void MainWindow::listWidget_itemClicked() {
    lastIndex = listWidget->currentRow();

    NODE temp = objectList.at(lastIndex);
    int color;
    QString str;

    switch(temp.type) {
        case 0:
            str = temp.text->getStr();
            lineEdit->setText(str);
            break;

        case 1:
            color = temp.square->getColor();
            comboBoxSquare->setCurrentIndex(color);
            break;

        case 2:
            color = temp.god->getColor();
            comboBoxGod->setCurrentIndex(color);
            break;
    }
}
```

Текст файла text.h:

```
#ifndef TEXT_H
#define TEXT_H

#include <QObject>
#include <QGraphicsItem>
#include <QPainter>
#include <QGraphicsSceneMouseEvent>
#include <QCursor>

class TEXT: public QObject, public QGraphicsItem {
    Q_OBJECT

    QString strText;

public:
    explicit TEXT(QObject * parent = 0);
    ~TEXT();

    void setStr(QString);
    QString getStr();

    void mouseMoveEvent(QGraphicsSceneMouseEvent * event);
    void mousePressEvent(QGraphicsSceneMouseEvent * event);
    void mouseReleaseEvent(QGraphicsSceneMouseEvent * event);

signals:

private:
    QRectF boundingRect() const;
    void paint(QPainter *painter, const QStyleOptionGraphicsItem *option,
QWidget *widget);

    void randomPos();

public slots:
};

#endif // TEXT_H
```

Текст файла text.cpp:

```
#include "text.h"

TEXT::TEXT(QObject *parent): QObject(parent), QGraphicsItem() {
    strText = "";
    this->setFlag(ItemIsSelectable);
    randomPos();
}

TEXT::~TEXT() {

}

QRectF TEXT::boundingRect() const {
    return QRectF (0, -10, 100, 10);
}

void TEXT::setStr(QString s) {
    strText = s;
}
```

```
QString TEXT::getStr() {
    return strText;
}

void TEXT::randomPos() {
    this->setPos(mapToScene(rand() % 510, rand() % 510));
}

void TEXT::paint(QPainter *painter, const QStyleOptionGraphicsItem *option,
QWidget *widget) {
    painter->setPen(QColor(0, 0, 0));
    painter->drawText(0, 0, strText);

    Q_UNUSED(option);
    Q_UNUSED(widget);
}

void TEXT::mouseMoveEvent(QGraphicsSceneMouseEvent *event) {
    if (event->buttons() & Qt::LeftButton) {
        if(mapToScene(event->pos()).x() < 510 && mapToScene(event->pos()).x()
> 0)
            this->setX(mapToScene(event->pos()).x());
        if(mapToScene(event->pos()).y() < 550 && mapToScene(event->pos()).y()
> 10)
            this->setY(mapToScene(event->pos()).y());
    }
}

void TEXT::mousePressEvent(QGraphicsSceneMouseEvent *event) {
    if (event->buttons() & Qt::LeftButton)
        this->setCursor(QCursor(Qt::ClosedHandCursor));
    else if (event->buttons() & Qt::RightButton)
        this->setSelected(true);
    Q_UNUSED(event);
}

void TEXT::mouseReleaseEvent(QGraphicsSceneMouseEvent *event) {
    this->setCursor(QCursor(Qt::ArrowCursor));
    this->setSelected(false);
    Q_UNUSED(event);
}
```

Текст файла square.h:

```
#ifndef SQUARE_H
#define SQUARE_H

#include <QObject>
#include <QGraphicsItem>
#include <QPainter>
#include <QGraphicsSceneMouseEvent>
#include <QCursor>

class SQUARE: public QObject, public QGraphicsItem {
    Q_OBJECT

    char color;
    QBrush brush;

public:
    explicit SQUARE(QObject * parent = 0);
```

```
    ~SQUARE();

    void setColor(int);
    int getColor();

    void mouseMoveEvent(QGraphicsSceneMouseEvent * event);
    void mousePressEvent(QGraphicsSceneMouseEvent * event);
    void mouseReleaseEvent(QGraphicsSceneMouseEvent * event);

signals:

private:
    QRectF boundingRect() const;
    void paint(QPainter *painter, const QStyleOptionGraphicsItem *option,
QWidget *widget);

    void randomPos();

public slots:
};

#endif // SQUARE_H
```

Текст файла square.cpp:

```
#include "square.h"

SQUARE::SQUARE(QObject *parent): QObject(parent), QGraphicsItem() {
    color = 0;
    this->setFlag(ItemIsSelectable);
    randomPos();
}

SQUARE::~SQUARE() {

}

QRectF SQUARE::boundingRect() const {
    return QRectF (-40, -40, 80, 80);
}

void SQUARE::setColor(int c) {
    color = c;
}

int SQUARE::getColor() {
    return color;
}

void SQUARE::randomPos() {
    this->setPos(mapToScene(rand() % 510, rand() % 510));
}

void SQUARE::paint(QPainter *painter, const QStyleOptionGraphicsItem *option,
QWidget *widget) {
    painter->setPen(Qt::NoPen);

    switch(color) {
        case 0: {
            QBrush brush(QColor(169, 178, 195));
            painter->setBrush(brush);
            }
```

```
            break;

        case 1: {
            QBrush brush(QColor(255, 140, 105));
            painter->setBrush(brush);
            }
            break;

        default: {
            QBrush brush(QColor(255, 99, 71));
            painter->setBrush(brush);
            }
            break;
    }


    painter->drawRect(-40, -40, 80, 80);

    Q_UNUSED(option);
    Q_UNUSED(widget);
}

void SQUARE::mouseMoveEvent(QGraphicsSceneMouseEvent *event) {
    if (event->buttons() & Qt::LeftButton) {
        if(mapToScene(event->pos()).x() < 550 && mapToScene(event->pos()).x()
> 0)
            this->setX(mapToScene(event->pos()).x());
        if(mapToScene(event->pos()).y() < 550 && mapToScene(event->pos()).y()
> 0)
            this->setY(mapToScene(event->pos()).y());
    }
}

void SQUARE::mousePressEvent(QGraphicsSceneMouseEvent *event) {
    if (event->buttons() & Qt::LeftButton)
        this->setCursor(QCursor(Qt::ClosedHandCursor));
    else if (event->buttons() & Qt::RightButton)
        this->setSelected(true);
    Q_UNUSED(event);
}

void SQUARE::mouseReleaseEvent(QGraphicsSceneMouseEvent *event) {
    this->setCursor(QCursor(Qt::ArrowCursor));
    this->setSelected(false);
    Q_UNUSED(event);
}
```

Текст файла god.h:

```
#ifndef GOD_H
#define GOD_H

#include <QObject>
#include <QGraphicsItem>
#include <QPainter>
#include <QGraphicsSceneMouseEvent>
#include <QCursor>

class GOD: public QObject, public QGraphicsItem {
    Q_OBJECT

    char color;
```

```
public:
    explicit GOD(QObject * parent = 0);
    ~GOD();

    void setColor(int);
    int getColor();

    void mouseMoveEvent(QGraphicsSceneMouseEvent * event);
    void mousePressEvent(QGraphicsSceneMouseEvent * event);
    void mouseReleaseEvent(QGraphicsSceneMouseEvent * event);

signals:

private:
    QPixmap img;

    QRectF boundingRect() const;
    void paint(QPainter *painter, const QStyleOptionGraphicsItem *option,
QWidget *widget);

    void randomPos();

public slots:
};

#endif // GOD_H
```

Текст файла god.cpp:

```
#include "god.h"

GOD::GOD(QObject *parent): QObject(parent), QGraphicsItem() {
    color = 0;
    this->setFlag(ItemIsSelectable);
    randomPos();
}

GOD::~GOD() {

}

void GOD::randomPos() {
    this->setPos(mapToScene(rand() % 510, rand() % 510));
}

QRectF GOD::boundingRect() const {
    return QRectF (-40, -40, 80, 80);
}

void GOD::setColor(int c) {
    color = c;
    switch(color) {
        case 0: {
            QImage i(":/res/zeus.png");
            img = QPixmap::fromImage(i.scaled(80, 80));
            } break;

        case 1: {
            QImage i(":/res/poseidon.png");
            img = QPixmap::fromImage(i.scaled(80, 80));
            } break;
```

```
        default: {
            QImage i(":/res/hades.png");
            img = QPixmap::fromImage(i.scaled(80, 80));
            } break;
    }
}

int GOD::getColor() {
    return color;
}

void GOD::paint(QPainter *painter, const QStyleOptionGraphicsItem *option,
QWidget *widget) {
    painter->setPen(Qt::NoPen);

    painter->drawPixmap(-40, -40, 80, 80, img, 0, 0, 80, 80);

    Q_UNUSED(option);
    Q_UNUSED(widget);
}

void GOD::mouseMoveEvent(QGraphicsSceneMouseEvent *event) {
    if (event->buttons() & Qt::LeftButton) {
        if(mapToScene(event->pos()).x() < 550 && mapToScene(event->pos()).x()
> 0)
            this->setX(mapToScene(event->pos()).x());
        if(mapToScene(event->pos()).y() < 550 && mapToScene(event->pos()).y()
> 0)
            this->setY(mapToScene(event->pos()).y());
    }
}

void GOD::mousePressEvent(QGraphicsSceneMouseEvent *event) {
    if (event->buttons() & Qt::LeftButton)
        this->setCursor(QCursor(Qt::ClosedHandCursor));
    else if (event->buttons() & Qt::RightButton)
        this->setSelected(true);
    Q_UNUSED(event);
}

void GOD::mouseReleaseEvent(QGraphicsSceneMouseEvent *event) {
    this->setCursor(QCursor(Qt::ArrowCursor));
    this->setSelected(false);
    Q_UNUSED(event);
}
```

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ ПРОГРАММЫ

На рисунке 1 представлен интерфейс программы. Слева можно увидеть кнопки добавления объектов на сцену, инструменты для настройки этих объектов и список всех объектов на сцене. Справа же располагается виджет QGraphicsView для просмотра графических элементов.

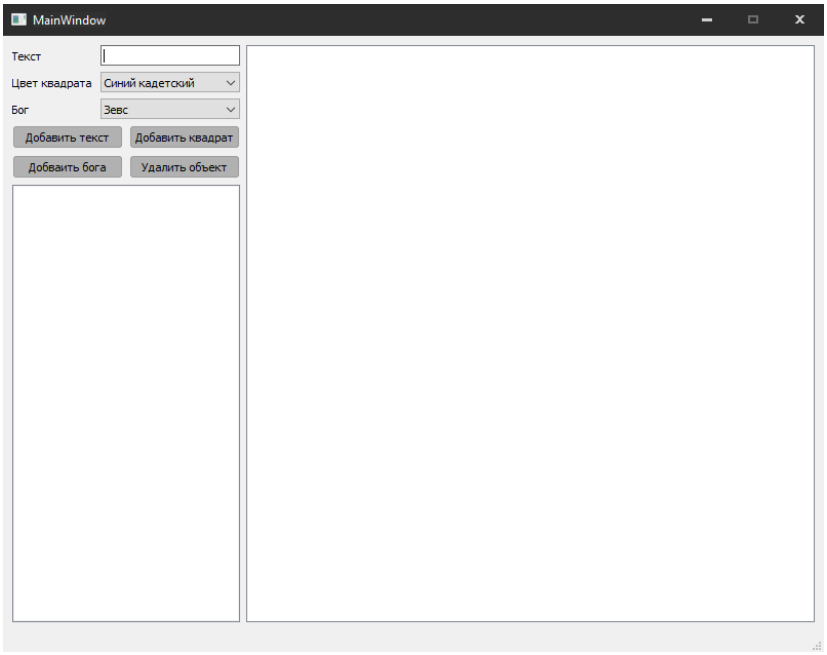

Рисунок 1 – Интерфейс программы.

На рисунке 2 видно, что на сцены были добавлены 3 квадрата разных цветов и 3 разных бога. На рисунке 3 видно, что объекты переместили (перемещение было произведено с помощью левой клавиши мыши).

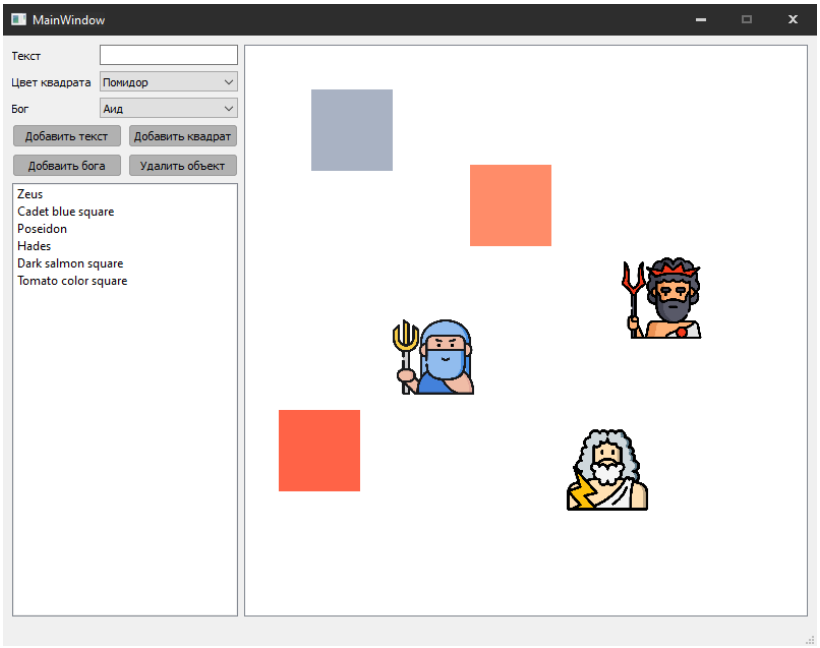

Рисунок 2 – Сцена с объектами.

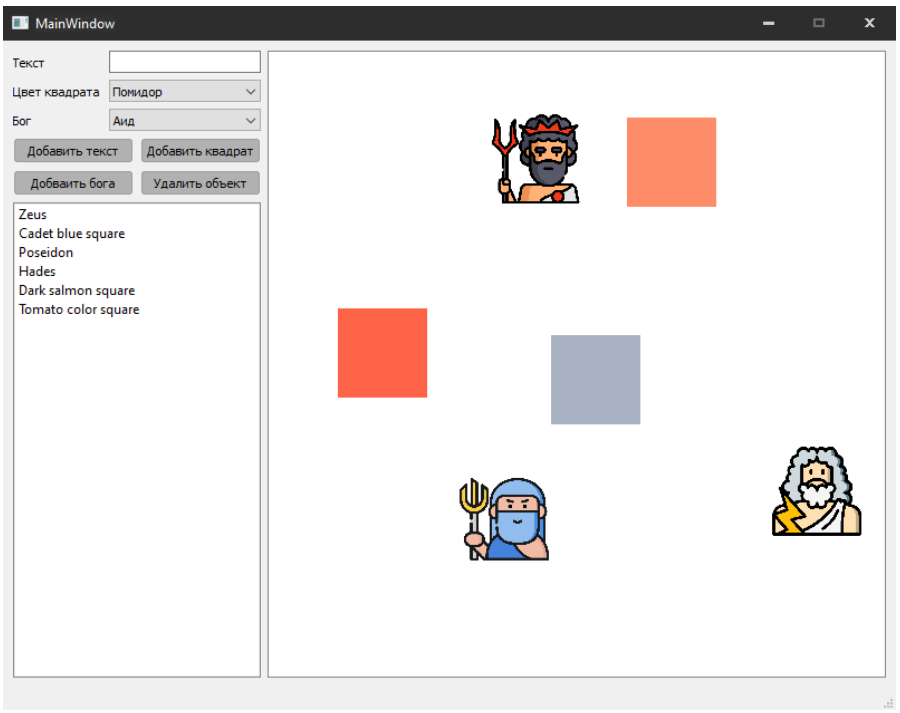

Рисунок 3 – Сцена с перемещенными объектами.

На рисунке 4 показано, что произошло после нажатия правым щелчком на объект с картинкой посейдона, а именно: в списке объектов появилось выделение объекта и в меню с выбором богов поменялось название бога (с аида на посейдона). После такого выделения можно поменять картинку посейдона на другую (например, зевса). Это видно на картинке 5. Так же после замены картинки одного бога на другого поменялось и название объекта в списке объектов. Такие же действия можно проделывать и с объектами, выделенными в списке а не на сцене.

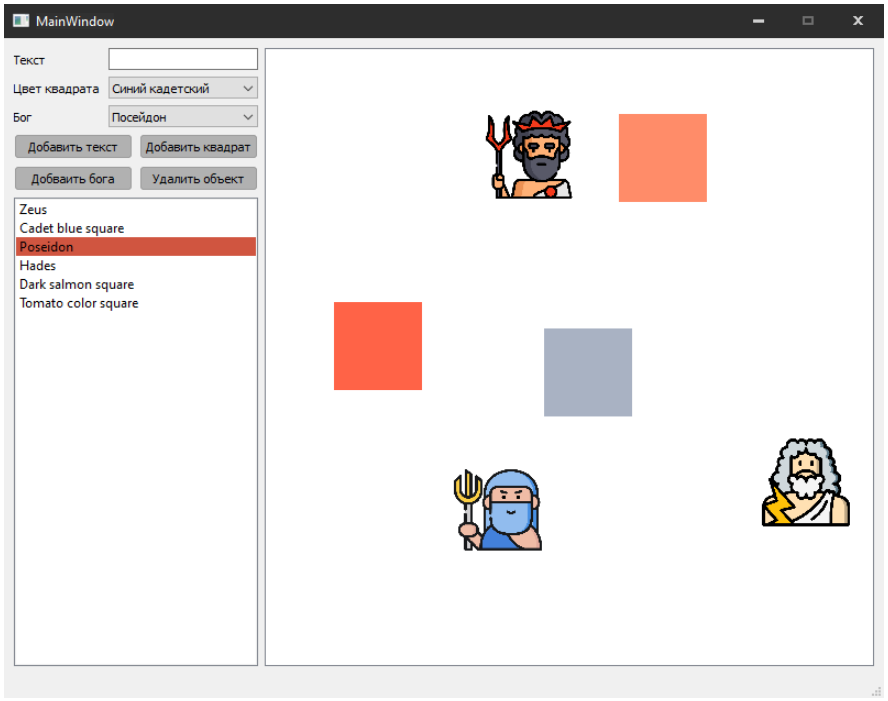

Рисунок 4 – Выделение объекта.

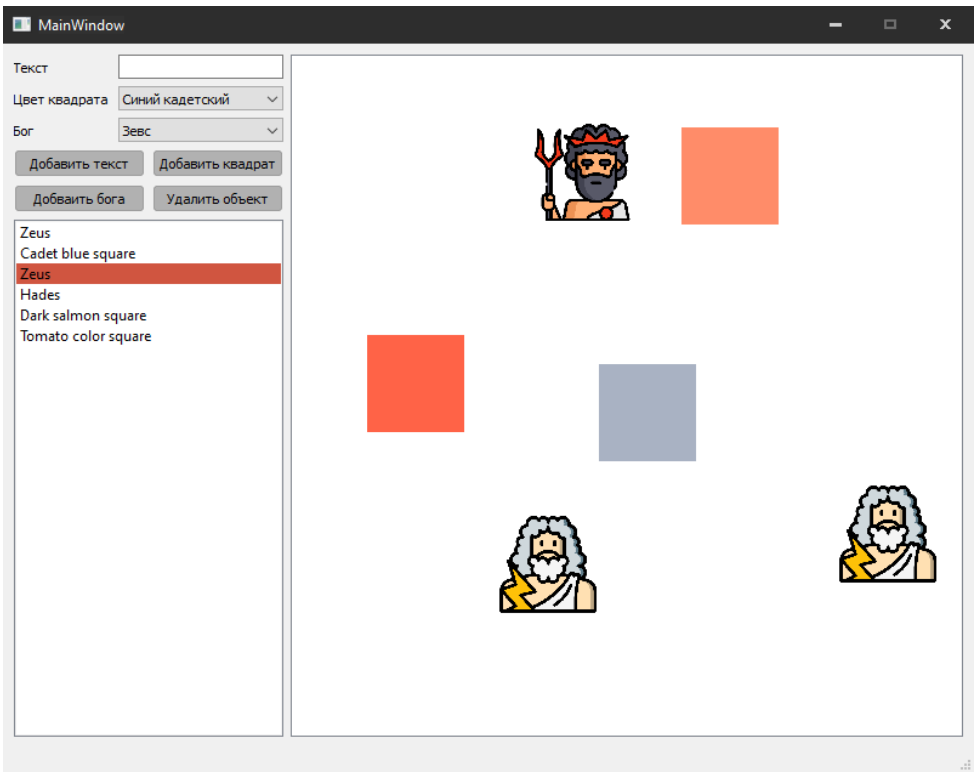

Рисунок 5 – Замена картинки бога.

На рисунке 6 видно, что добавление текста тоже работает.

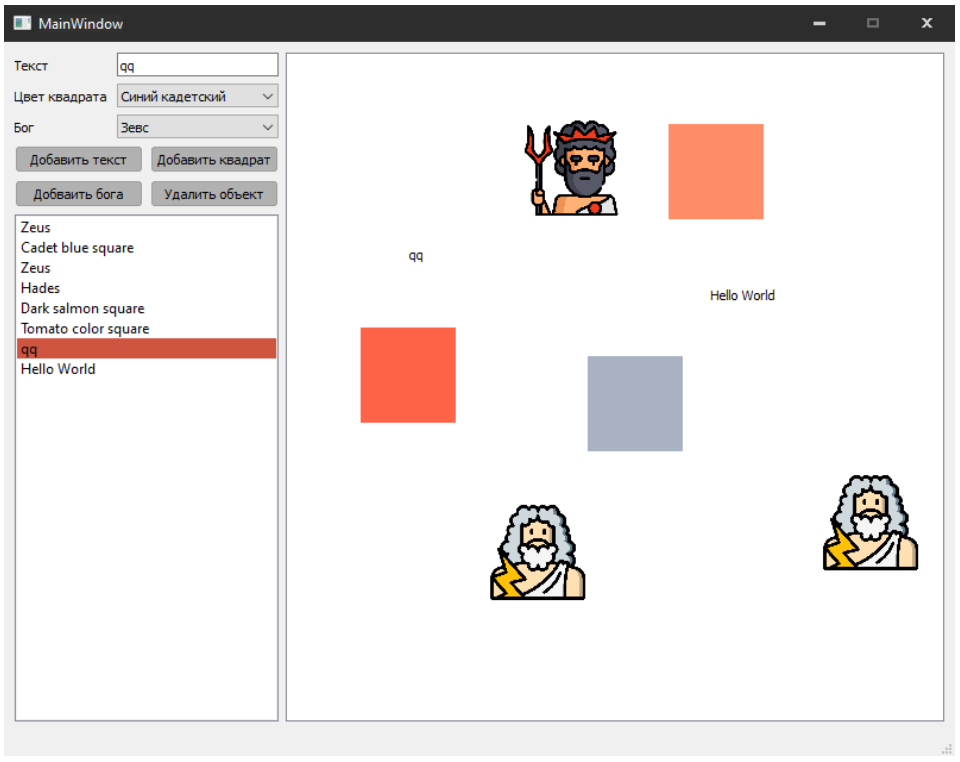

Рисунок 6 – Добавлен текст.

На рисунке 7 видно, что кнопка “удалить объект” тоже работает. Для того, чтобы удалить объект нужно его выделить одним из двух способов и нажать на кнопку.

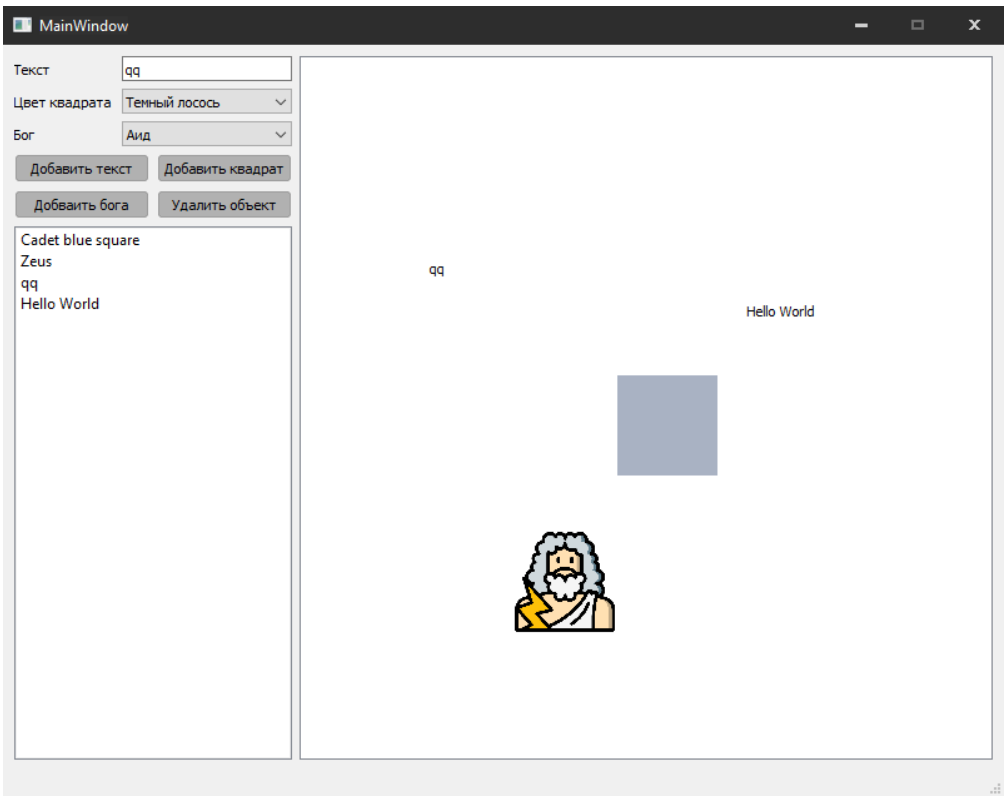

Рисунок 7 – Удалены объекты.

ВЫВОД

Немного разобрались как работать в Qt с QGraphicsItem, QGraphicsScene и QGraphicsView.

Написали программу соответствующую варианту задания.