Правительство Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Кафедра «Компьютерной безопасности»

Отчёт к лабораторной работе №2

по дисциплине

«Языки программирования»

Работу выполнил

Студент группы СКБ223 _____ Д. О. Демешко
подпись, дата

Работу проверил _____ С. А. Булгаков
подпись, дата

Москва 2023

Содержание

Постановка задачи

Разработать программу с использованием библиотеки Qt. В окне программы реализовать возможность добавления геометрической фигуры посредством контекстного меню. Реализовать перемещение фигуры в рамках окна при перетаскивании ее курсором мыши. При нажатии на фигуру правой кнопкой мыши выводить контекстное меню позволяющее повернуть или изменить размер фигуры.

Для реализации фигуры использовать класс QWidget и возможности класса QPainter.

Примечание: не использовать QGraphicsScene и QPainterPath.

1. Алгоритм решения задачи

Для решения поставленной задачи были разработаны классы Widget и GeomFigure, которые позволяют добавлять геометрическую фигуру на виджет, перемещать ее, поворачивать и менять ее размер.

2. Решение задачи

Были разработаны 2 класса: класс Widget и класс GeomFigure, разработанные на основе класса QWidget.

2.1 Класс Widget

Класс состоит из конструктора класса Widget, деструктора и слота addGeomFigure.

2.1.1 Конструктор класса Widget

Конструктор устанавливает минимальный размер с помощью setMinimumSize, создает экземпляр QMenu с действием QAction - “Add geometric figure” и соединяет сигнал QAction::triggered со слотом addGeomFigure с помощью connect. Также в конструкторе устанавливается политика контекстного меню, которая позволяет отслеживать пользовательский запрос на контекстное меню.

2.1.2 Деструктор класса Widget

Не делает ничего дополнительного, кроме как базовый деструктор класса QWidget.

2.1.3 Слот addGeomFigure

Данный слот создает экземпляр класса GeomFigure и выводит его на виджет с помощью show.

2.2 Класс GeomFigure

Класс состоит из конструктора, protected методов paintEvent, mousePressEvent, mouseMoveEvent, private слотов showContextMenu, rotate, resize. В поля этого класса входят координаты x, y, угол поворота фигуры angle и размер фигуры scale_f.

2.2.1 Конструктор класса

В конструкторе устанавливается минимальный размер фигуры(для того чтобы она не обрезалась при выходе за границы), политика контекстного меню, устанавливается соединение между сигналом customContextMenuRequested и слотом showContextMenu. Также устанавливается изначальный угол поворота фигуры в 0 градусов, и размер фигуры 1(коэффициент размерности).

2.2.2 Метод paintEvent

Создается объект класса QPainter для рисования на виджете, текущее преобразование QTransform, вычисляется центр виджета center и центр координат устанавливается в центре виджета(во избежание повреждения фигуры). Производится поворот фигуры на угол angle и изменение размера в соответствие со scale_f. Также в данном методе рисуется сама фигура и производится преобразование.

2.2.3 Метод mousePressEvent

В данном методе сохраняются координаты начальной точки нажатия мыши.

2.2.4 Метод mouseMoveEvent

В данном методе вычисляются разницы между текущими координатами мыши и сохраненными, после чего фигура перемещается на соответствующее расстояние

2.2.5 Слот showContextMenu

В данном слоте создается экземпляр класса QMenu для контекстного меню, действия rotate и resize, устанавливаются соединения между действиями и соответствующими слотами. Контекстное меню отображается в позиции pos, преобразованной в глобальные координаты с помощью mapToGlobal(pos).

2.2.6 Слот rotate

В данном слоте отображается диалоговое окно QInputDialog, в котором необходимо ввести угол поворота фигуры в градусах. При нажатии ОК, переменная angle устанавливается равной введенному пользователем значению, а виджет обновится с помощью update.

2.2.7 Слот resize

В данном слоте отображается диалоговое окно QInputDialog, в котором необходимо ввести размер фигуры. При нажатии ОК, переменная scale_f устанавливается равной введенному пользователем значению, а виджет обновится с помощью update.

3.Тестирование программы

Вид окна с контекстным меню для добавления фигуры представлен на рисунке 1.

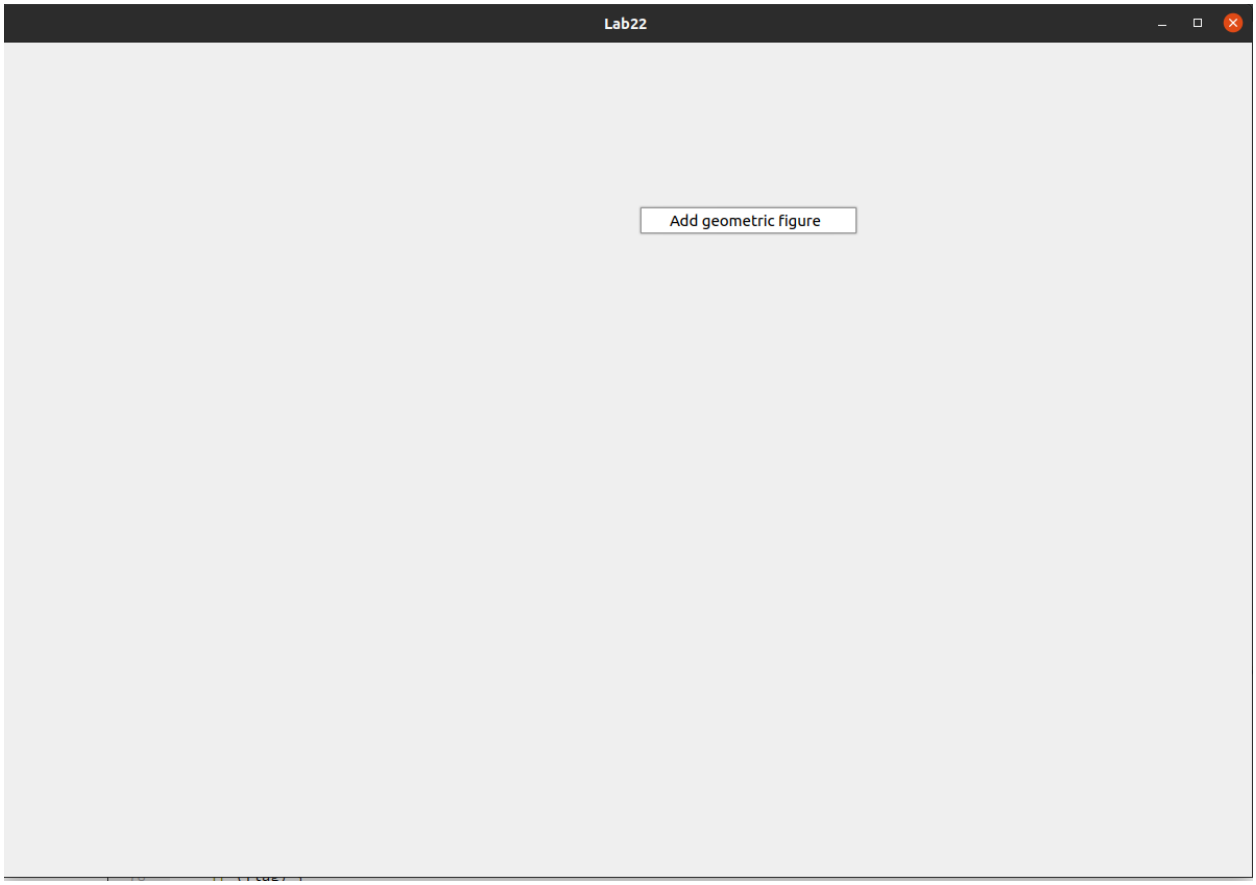

Рисунок 1

На рисунке 2 представлена нарисованная фигура.

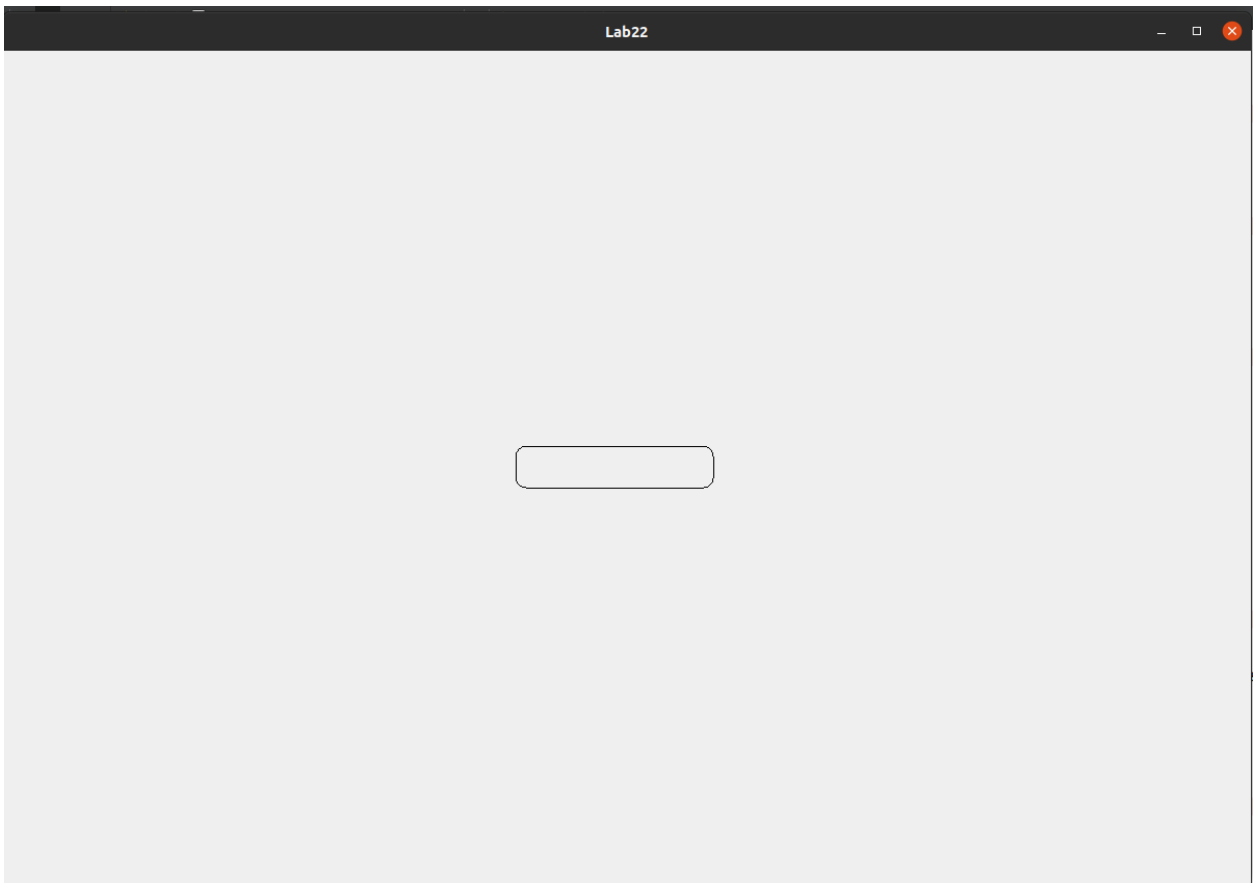

Рисунок 2

На рисунке 3 представлено контекстное меню, выпадающее при нажатии правой клавишей на фигуру.

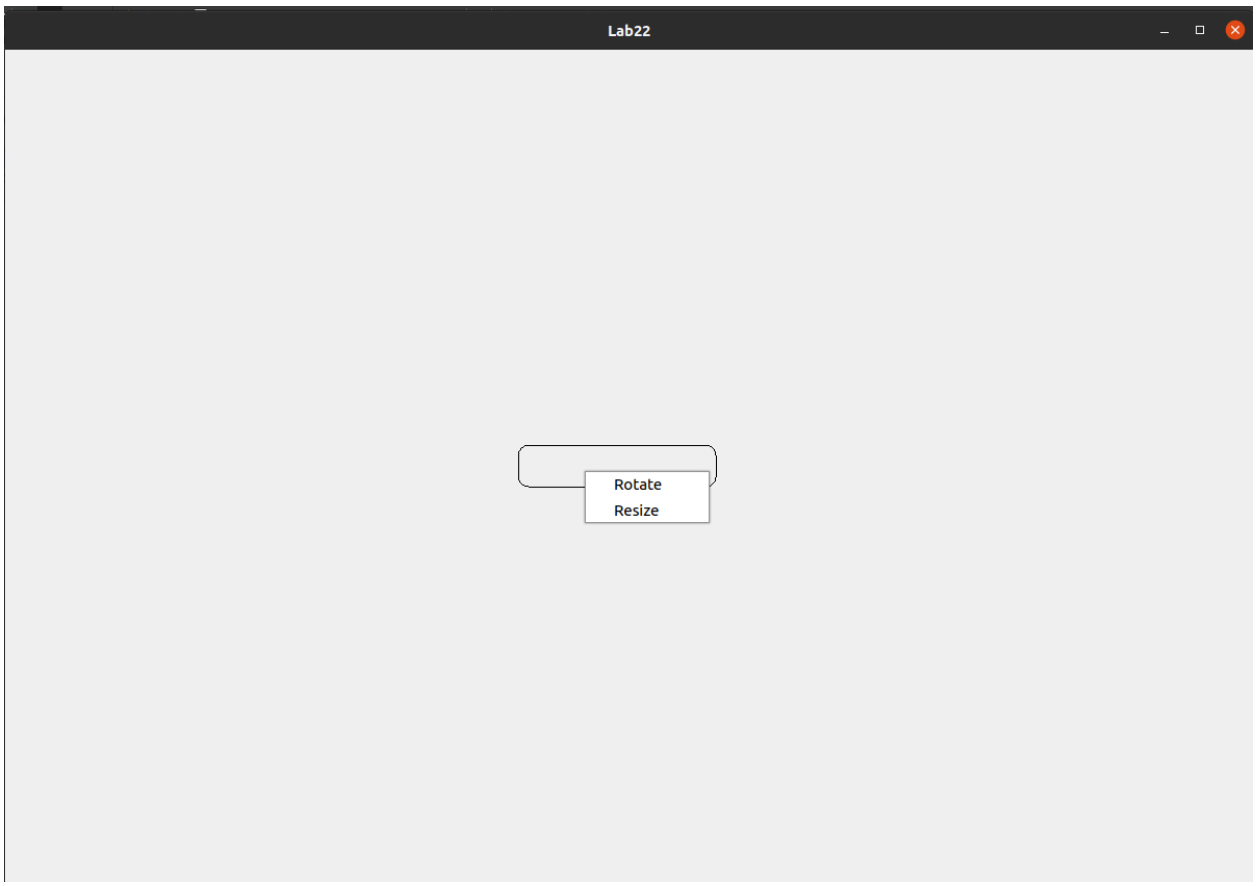

Рисунок 3

На рисунке 4 представлено диалоговое окно ввода угла поворота.

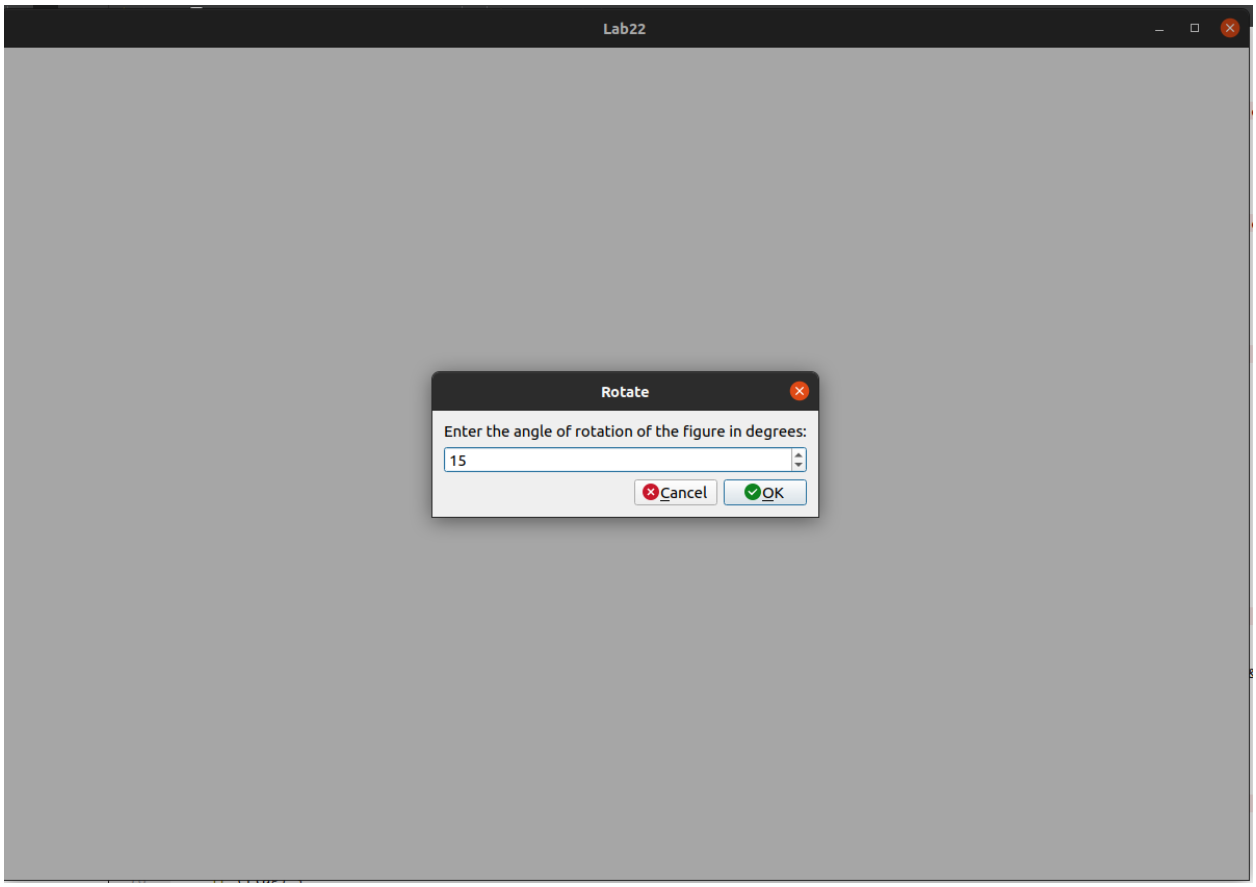

Рисунок 4

На рисунке 5 представлено диалоговое окно ввода размера фигуры.

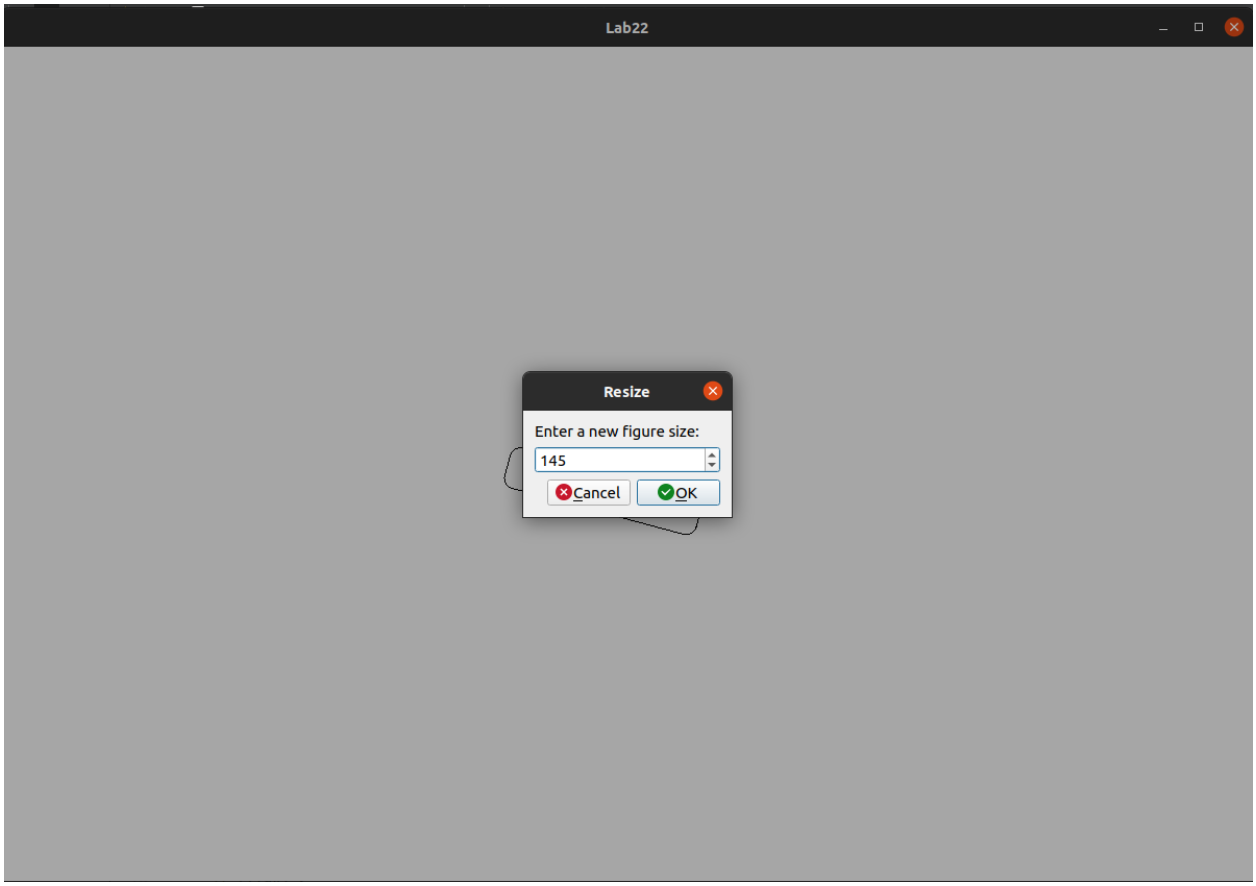

Рисунок 5

На рисунке 6 представлена фигура после преобразований.

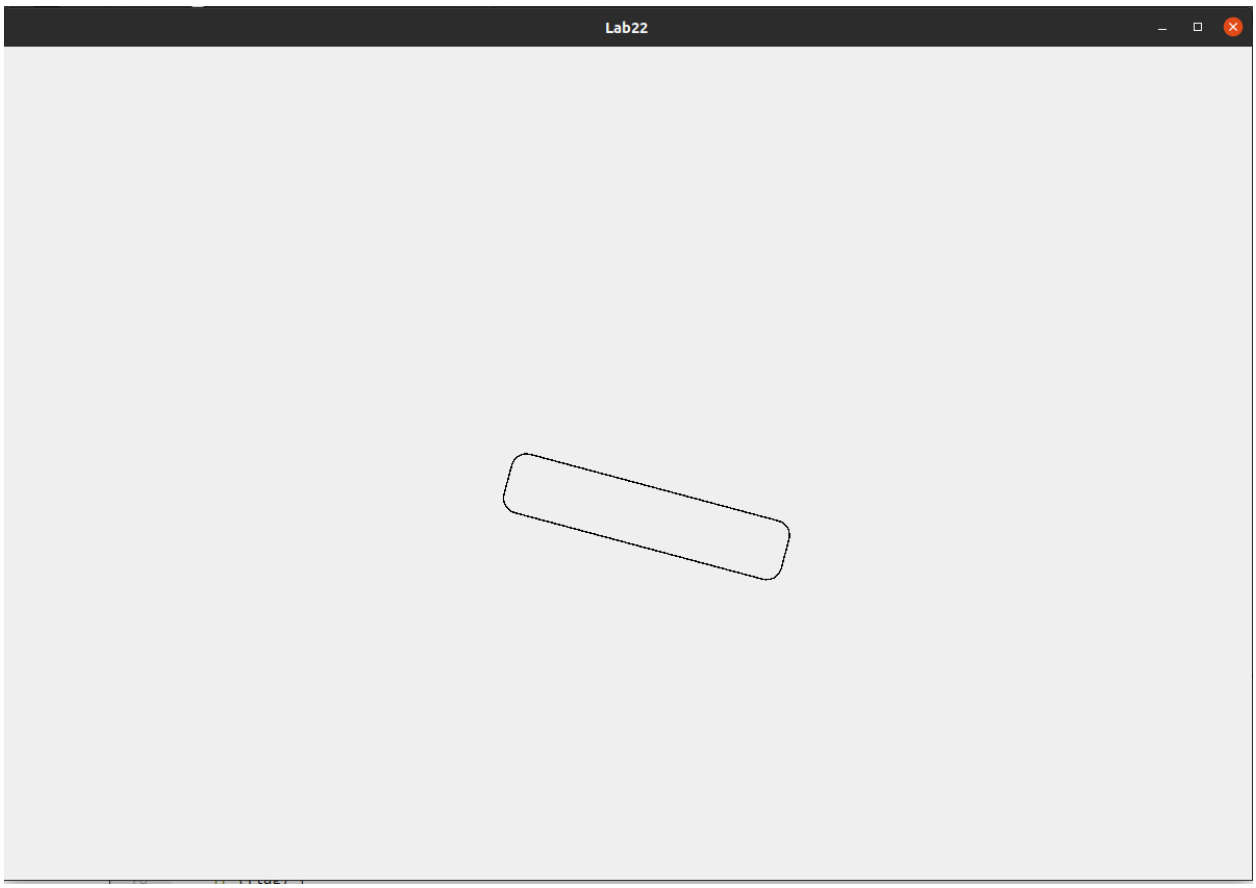

Рисунок 6

4. UML диаграмма

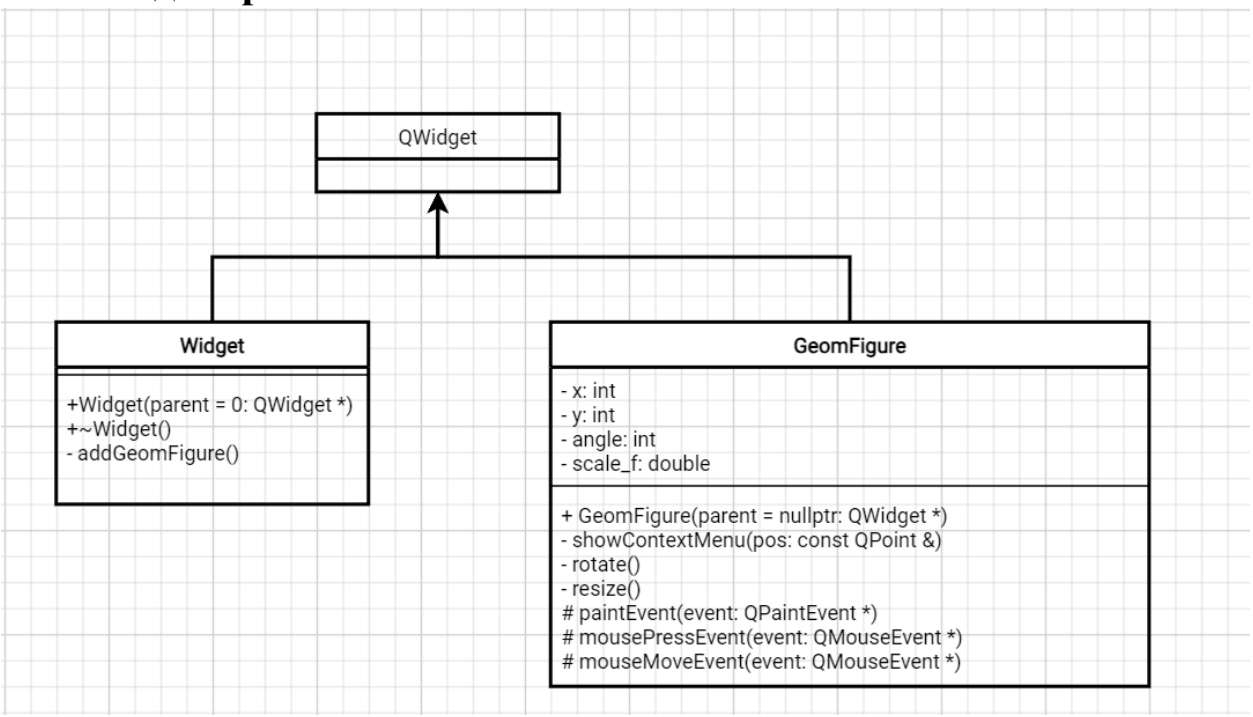

Рисунок 7

Приложение А

А.1 Исходный код main.cpp

```
#include "widget.h"
#include <QApplication>
int main(int argc, char *argv[])
{
   QApplication a(argc, argv);
   Widget w;
   w.show();
   return a.exec();
}
```

А.2 Исходный код widget.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QWidget>
class Widget : public QWidget
{
   Q_OBJECT
public:
   Widget(QWidget *parent = nullptr);
   ~Widget();
private slots:
   void addGeomFigure();
};
#endif // MAINWINDOW_H
```

А.3 Исходный код widget.cpp

```
 #include "widget.h"
#include "geometric.h"
#include <QMenu>
#include <QAction>
#include <QInputDialog>
Widget::Widget(QWidget *parent)
```

```
    : QWidget(parent)
{
    setMinimumSize(1200, 800);
    QMenu *contextMenu = new QMenu(this);
    QAction *addFigureAction = new QAction("Add geometric figure", this);
    connect(addFigureAction, &QAction::triggered, this, &Widget::addGeomFigure);
    contextMenu->addAction(addFigureAction);
    setContextMenuPolicy(Qt::CustomContextMenu);
    connect(this, &QWidget::customContextMenuRequested, [this, contextMenu](const QPoint &pos) {
        contextMenu->popup(mapToGlobal(pos));
    });
}
void Widget::addGeomFigure()
{
    GeomFigure *shape = new GeomFigure(this);
    shape->show();
}
Widget::~Widget()
{
}
```

Приложение Б

Б.1 Исходный код geometric.h

```
#ifndef GEOMETRIC_H
#define GEOMETRIC_H
#include <QWidget>
#include <QMouseEvent>
class GeomFigure : public QWidget
{
   Q_OBJECT
   int x,y;
   int angle;
   double scale_f;
public:
   GeomFigure(QWidget *parent = nullptr);
protected:
   void paintEvent(QPaintEvent *event);
   void mousePressEvent(QMouseEvent *event);
   void mouseMoveEvent(QMouseEvent *event);
private slots:
   void showContextMenu(const QPoint &pos);
   void rotate();
   void resize();
};
#endif // GEOMETRIC_H
```

Б.2 Исходный код geometric.cpp

```
#include "geometric.h"
#include <QPainter>
#include <QMenu>
#include <QInputDialog>
#include <QTransform>
GeomFigure::GeomFigure(QWidget *parent) : QWidget(parent)
{
```

```
    setMinimumSize(1200, 1200);
    setContextMenuPolicy(Qt::CustomContextMenu);
    connect(this, &QWidget::customContextMenuRequested, this,
&GeomFigure::showContextMenu);
    angle = 0; scale_f = 1.0;
}
void GeomFigure::paintEvent(QPaintEvent *event)
{
  QPainter p(this);
  QTransform orig = p.transform();
  QPointF center(width()/2, height()/2);
  p.translate(center);
  p.rotate(angle);
  p.scale(scale_f, scale_f);
  p.drawLine(20, 10, 190, 10);
  p.drawLine(200, 20, 200, 40);
  p.drawLine(190, 50, 20, 50);
  p.drawLine(10, 40, 10, 20);
  p.drawArc(10,10,20,20,90*16,90*16);
  p.drawArc(10,30,20,20,180*16,90*16);
  p.drawArc(180,30,20,20,270*16,90*16);
  p.drawArc(180,10,20,20,  0*16,90*16);
  p.setTransform(orig);
  p.resetTransform();
}
void GeomFigure::mousePressEvent(QMouseEvent *event)
{
    x = event->pos().x();
    y = event->pos().y();
}
void GeomFigure::mouseMoveEvent(QMouseEvent *event)
{
```

```
    int delta_x = event->pos().x() - x;
    int delta_y = event->pos().y() - y;
    move(pos().x()+delta_x, pos().y()+delta_y);
}
void GeomFigure::showContextMenu(const QPoint &pos)
{
    QMenu contextMenu(this);
    QAction rotateAction("Rotate", this);
    connect(&rotateAction, &QAction::triggered, this, &GeomFigure::rotate);
    contextMenu.addAction(&rotateAction);
    QAction resizeAction("Resize", this);
    connect(&resizeAction, &QAction::triggered, this, &GeomFigure::resize);
    contextMenu.addAction(&resizeAction);
    contextMenu.exec(mapToGlobal(pos));
}
void GeomFigure::rotate()
{
    bool flag;
    int a = QInputDialog::getInt(this, "Rotate", "Enter the angle of rotation of the figure in degrees:",
0, -360, 360, 1, &flag);
    if (flag) {
        angle = a;
        update();
    }
}
void GeomFigure::resize()
{
    bool flag;
    int size = QInputDialog::getInt(this, "Resize", "Enter a new figure size:", 100, 10, 150, 1,
&flag);
    if (flag) {
        scale_f = static_cast<double>(size) / 100.0;
        update();
```

```
    }
}
```