ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
УФИМСКИЙ УНИВЕРСИТЕТ НАУКИ И ТЕХНОЛОГИЙ

Кафедра ВМиК

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

по предмету **«Объектно-Ориентированное Программирование»**

Выполнил: студент группы МО-204Б

Исламов Ильнур Фандасович.

Проверил:
доцент каф. ВМиК

Котельников В.А.

Уфа 2025г.

Цель лабораторной работы

Лабораторная работа 3. Часть 1 из 2: «Круги на форме»

В рамках лабораторной работы необходимо разобраться:

- каким образом создавать и использовать собственные классы-контейнеры;
- каким образом организовать взаимодействие между графическим интерфейсом и внутренней структурой данных;
- каким образом реализовать выделение и групповые операции над графическими объектами;
- в каких случаях и каким образом используется событие перерисовки (Paint).

Задание

1. Создание базовых классов

- Создать класс CCircle, представляющий круг с постоянным радиусом.
- Создать собственный класс-контейнер CircleStorage для хранения коллекции объектов CCircle.

2. Реализация графического интерфейса

- Создать форму с областью для рисования (например, QWidget или PaintBox).
- Реализовать обработку изменения размера формы.

3. Взаимодействие с пользователем

- При нажатии левой кнопки мыши на форме создавать новый объект CCircle с координатами точки нажатия и добавлять его в контейнер.
- Реализовать отрисовку всех кругов из контейнера в обработчике события Paint.

4. Система выделения объектов

- Реализовать логику выделения, аналогичную популярным графическим редакторам (например, Microsoft Visio):

    - Одиночное выделение при клике на круг
    - Множественное выделение при клике с зажатой Ctrl
    - Выделение нескольких объектов при клике на область пересечения кругов
- Реализовать удаление всех выделенных объектов при нажатии клавиши Delete.

5. Требования к реализации

- Класс-контейнер должен инкапсулировать внутреннее хранилище (массив или список)
- Работа с контейнером должна осуществляться через единый публичный интерфейс
- Объекты CCircle должны самостоятельно определять:
    - Попадание точки в свою область (для выделения)
    - Логику собственной отрисовки

Лабораторная работа 3. Часть 2 из 2: «MVC»

Цель работы

В рамках лабораторной работы необходимо разобраться:

- каким образом разделять логику приложения согласно паттерну MVC;
- каким образом организовать согласованное отображение данных в нескольких компонентах;
- каким образом реализовать бизнес-правила и валидацию данных;
- каким образом минимизировать количество обновлений интерфейса.

Задание

1. Создание модели

- Создать класс TripleModel, хранящий три целых числа A, B, C в диапазоне 0-100

- Реализовать бизнес-правила:
  - $A \leq B \leq C$
  - При изменении A и C - разрешающее поведение
  - При изменении B - запрещающее или ограничивающее поведение

2. Создание представлений

- Для каждого числа реализовать три типа контролов:
  - Текстовое поле (TextEdit)
  - Числовой счетчик (SpinBox)
  - Ползунок (Slider)
- При изменении значения в любом контроле должны обновляться все остальные

3. Синхронизация данных

- Реализовать механизм уведомлений об изменении модели
- Обеспечить атомарность изменений модели
- Минимизировать количество уведомлений (максимум 1 при любом изменении)

4. Сохранение состояния

- Реализовать сохранение значений между запусками приложения
- Ответственность за сохранение должна лежать на модели

5. Требования к реализации

- Модель должна быть самодостаточной и работать без интерфейса
- Все бизнес-правила должны быть инкапсулированы в модели
- Пределы значений (0-100) должны задаваться в модели
- При запуске приложения - только одно уведомление об изменении
- Обработка некорректного ввода пользователя

Ход выполнения лабораторной работы

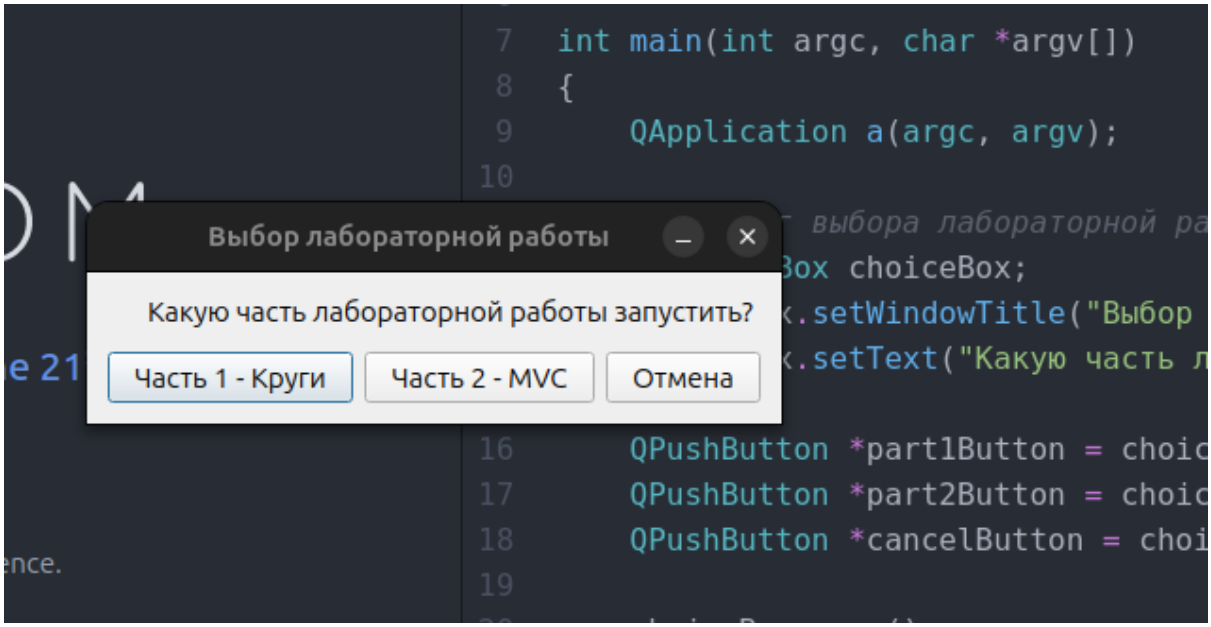


Рис .1 Начальное меню. Выбор: Часть1, Часть2, Отмена.

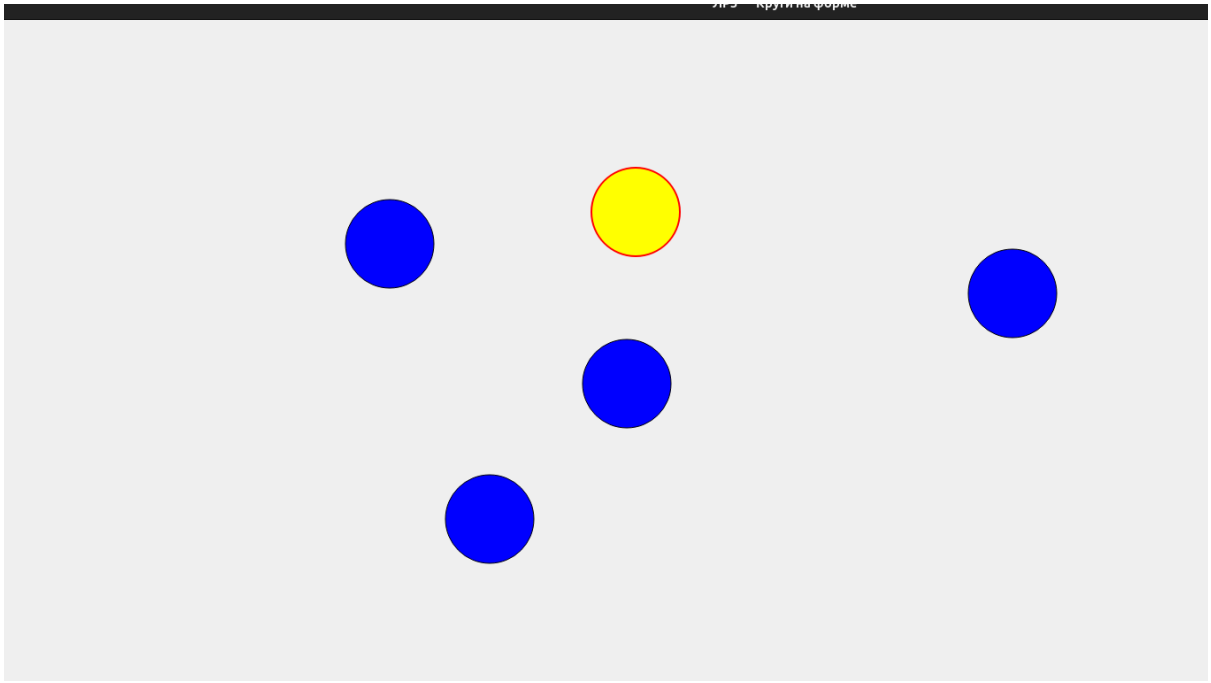


Рис .2 Окно первой части. ЛКМ - СОЗДАТЬ КРУГ, ПРИ ПОВТОРНОМ НАЖАТИИ КРУГ ВЫДЕЛЯЕТСЯ ЖЕЛТЫМ ЦВЕТОМ. CTRL-ЛКМ - МНОЖЕСТВЕННОЕ ВЫДЕЛЕНИЕ

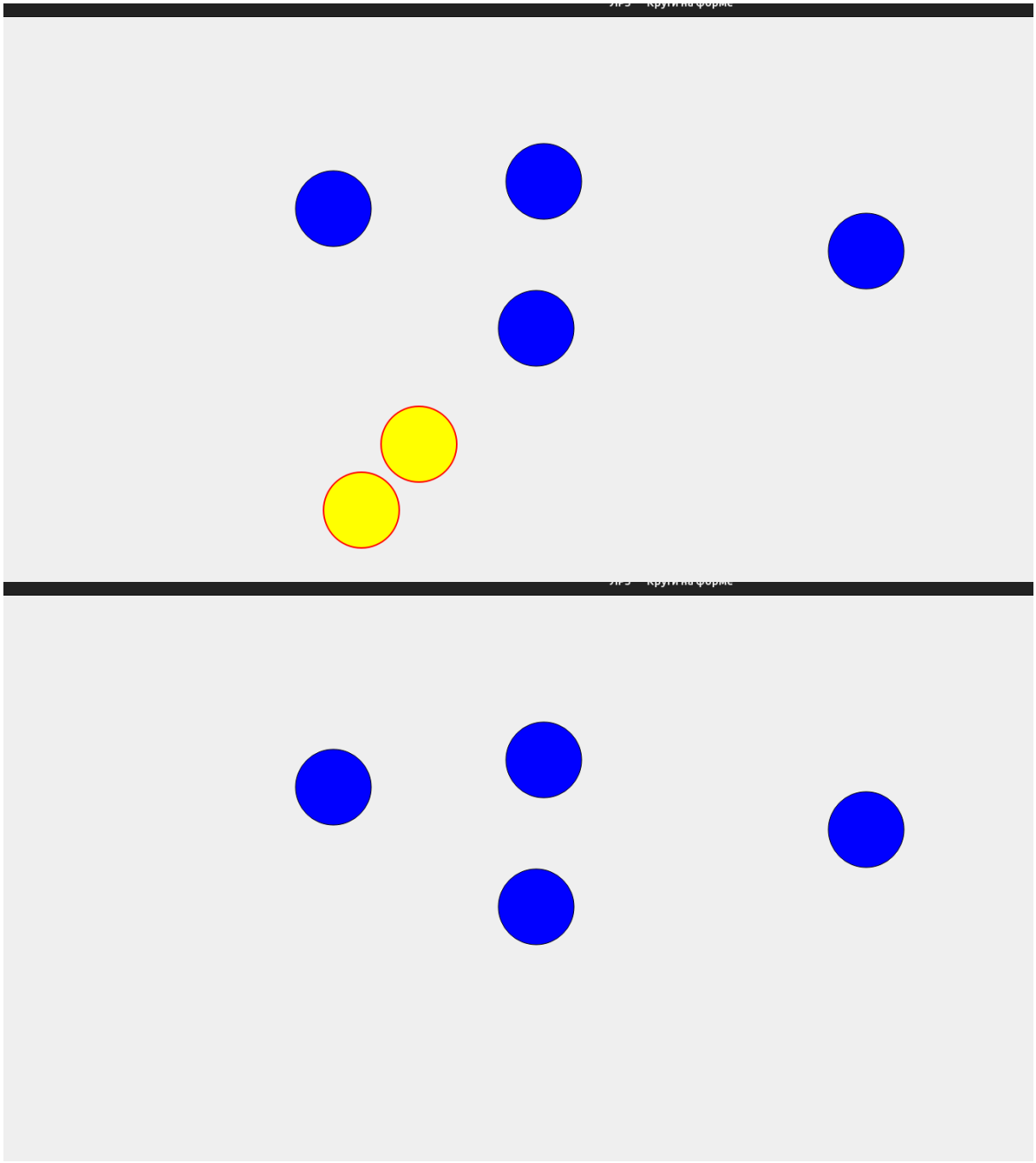


Рис .3 При нажатии delete выделенные круги удаляются.

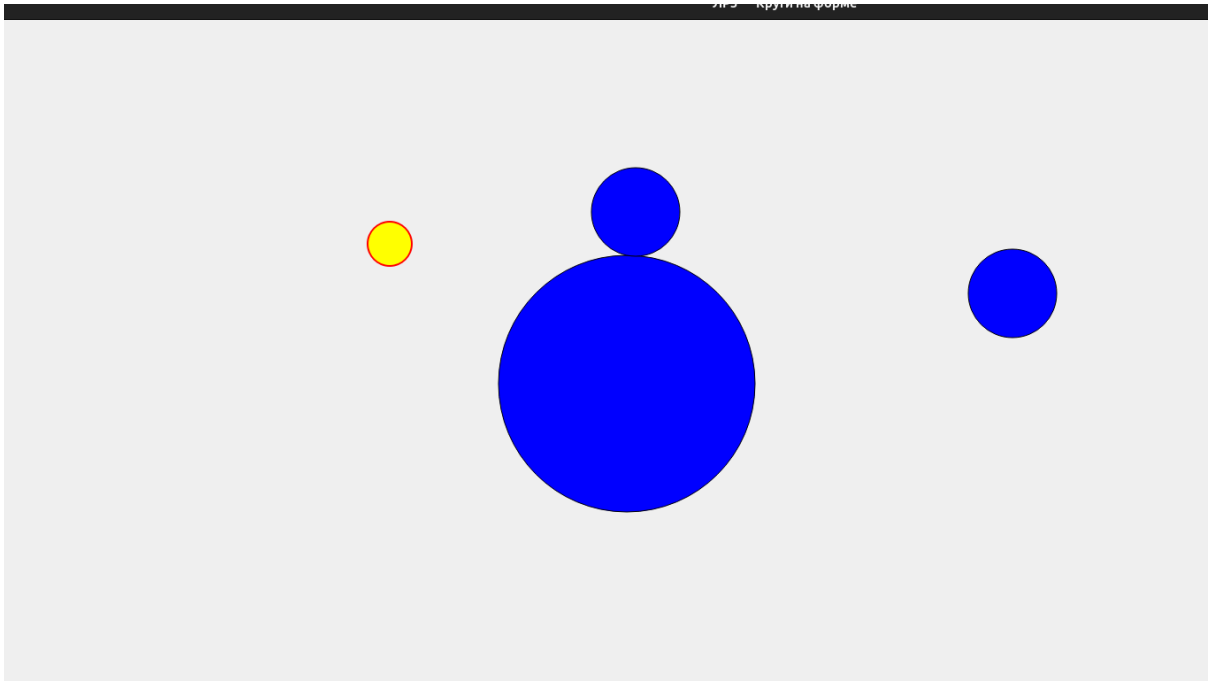
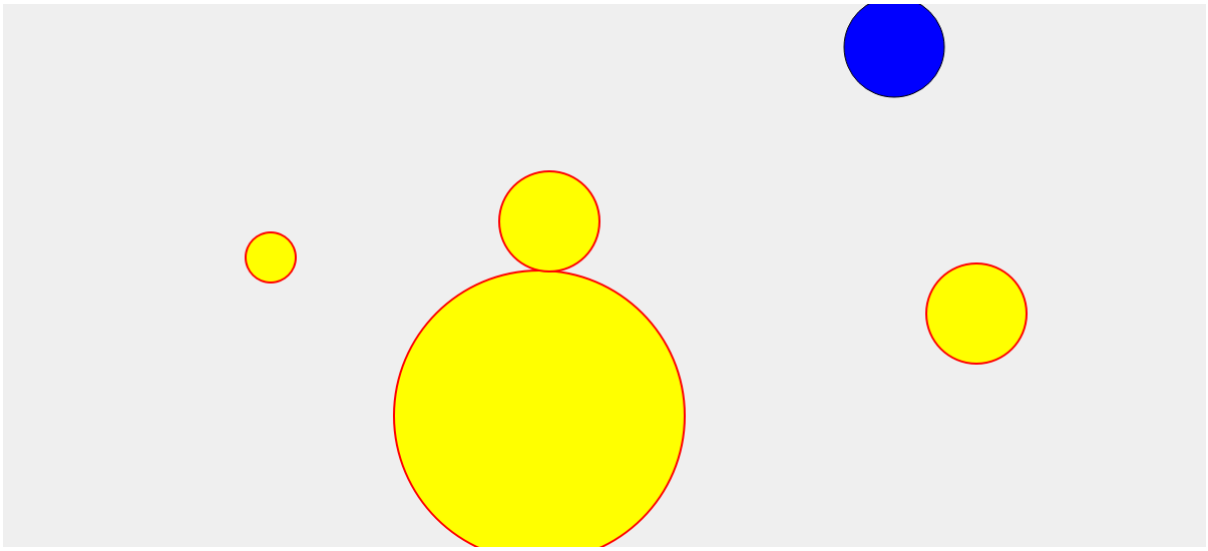


Рис .4 Колесико мыши - изменяет радиус круга под курсором. Прокрутка вверх - увеличивает радиус. Прокрутка вниз - уменьшает радиус. Ограничения: мин. 5px, макс. 200px



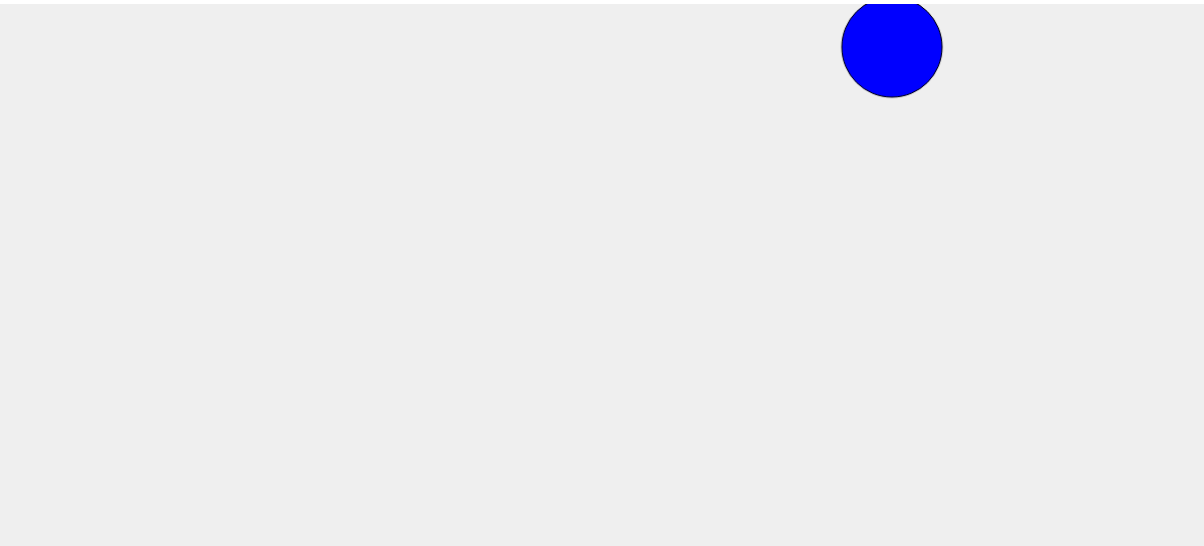


Рис. 5, Рис. 6 Множественное удаление выделенных кругов.

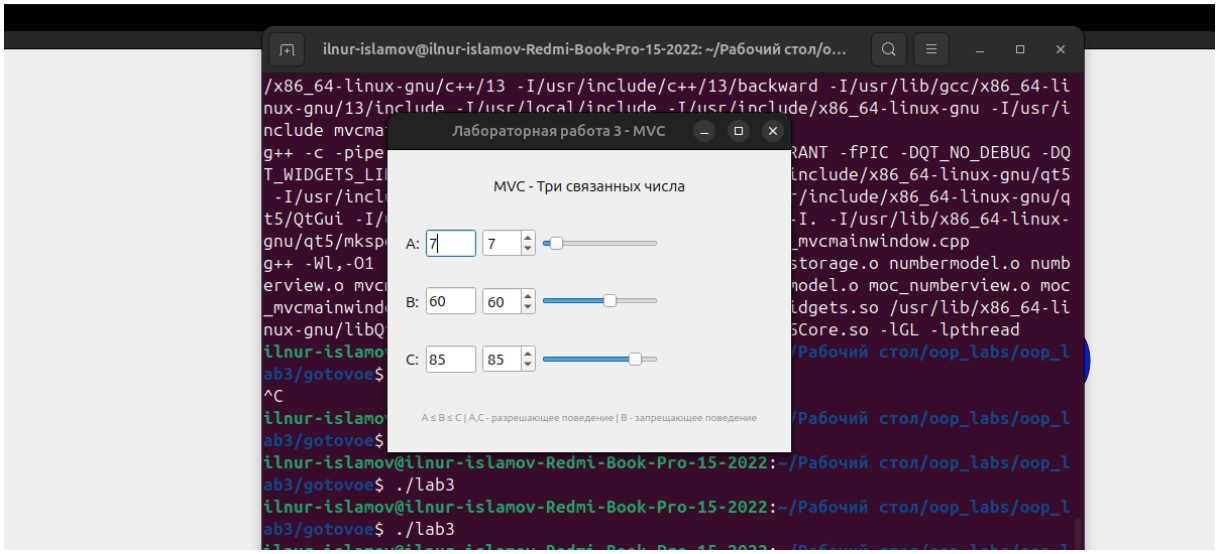


Рис .7 Окно второй части.

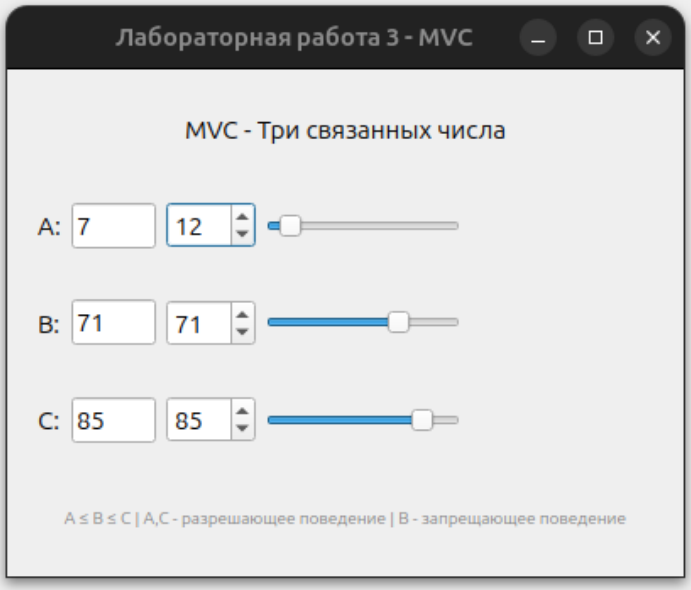


Рис .8 Текстовое поле для прямого ввода чисел. Где A ≤ B ≤ C, A и C - РАЗРЕШАЮЩЕЕ поведение, B - ЗАПРЕЩАЮЩЕЕ поведение

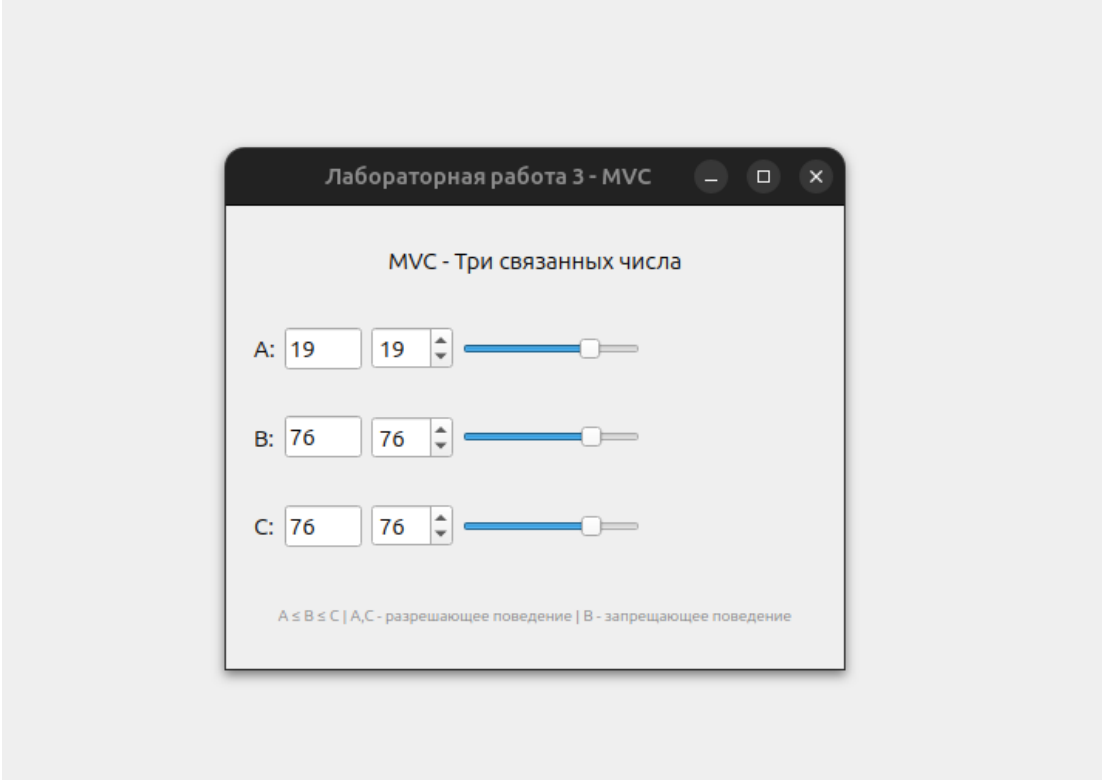


Рис .9 числовая крутилка с кнопками вверх/вниз

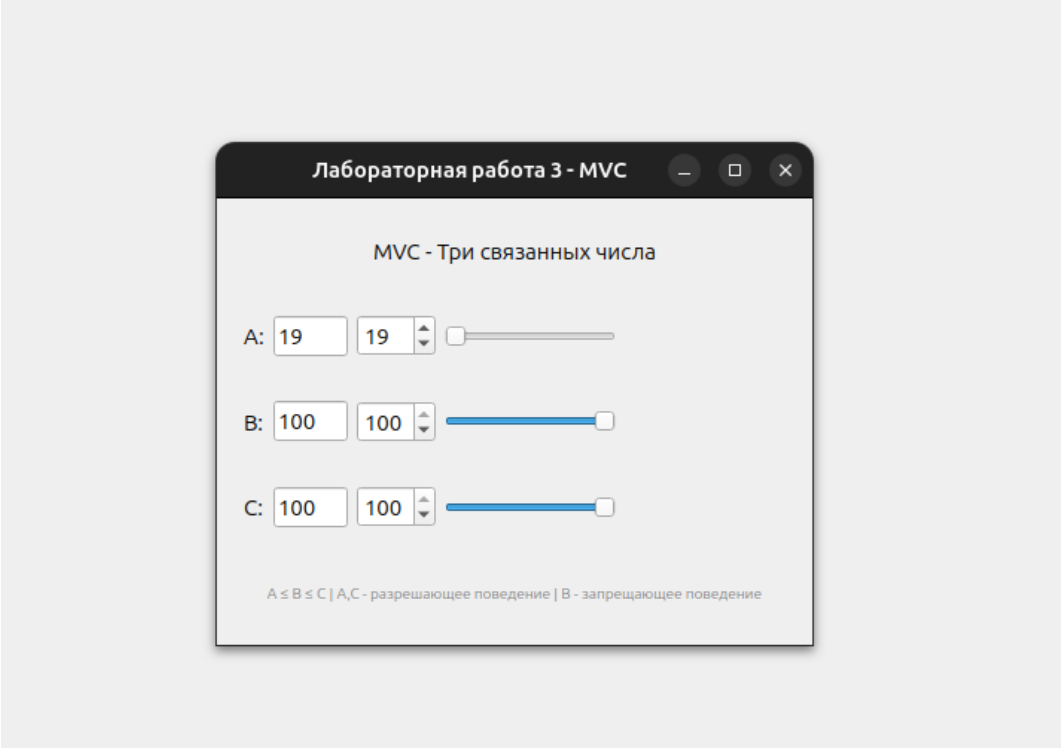


Рис .10 ползунок для визуального управления

Выводы по лабораторной работе

В результате выполнения лабораторной работы были освоены основы объектно-ориентированного программирования на практике через создание двух различных приложений.

В первой части ("Круги на форме") были изучены принципы инкапсуляции и композиции - каждый графический объект самостоятельно управляет своим состоянием, отрисовкой и взаимодействием с пользователем. Была реализована система выделения объектов, аналогичная профессиональным графическим редакторам, что позволило понять важность следования принципам единственной ответственности и минимальной связанности.

Во второй части ("MVC") был глубоко освоен архитектурный паттерн Model-View-Controller, что позволило понять преимущества разделения логики данных, представления и управления. Особое внимание было уделено созданию самодостаточной модели, способной функционировать независимо от пользовательского интерфейса, и оптимизации системы уведомлений для предотвращения избыточных обновлений.

Приложение №1

```
#include "mainwindow.h"
#include <QPainter>

#include <QWheelEvent>

void MainWindow::wheelEvent(QWheelEvent *event)
{
   // Получаем координаты мыши
```

```
    int x = event->position().x();
    int y = event->position().y();

    // Проверяем, есть ли круг под курсором
    CCircle* circle = storage.getCircleAt(x, y);
    if (circle) {
        // Определяем направление прокрутки
        int delta = event->angleDelta().y() / 120; // 1 шаг колеса = 120
        double newRadius = circle->getRadius() + delta * 5.0; // шаг 5 пикселей
        if (newRadius < 5) newRadius = 5;       // минимальный размер
        if (newRadius > 200) newRadius = 200;   // максимальный размер
        circle->setRadius(newRadius);
        update();
    }
}

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    //ui(new Ui::MainWindow)
{
    //ui->setupUi(this);
    setWindowTitle("ЛР3 — Круги на форме");
    setMinimumSize(600, 500);
}

MainWindow::~MainWindow()
{
    //delete ui;
}

void MainWindow::paintEvent(QPaintEvent *)
{
    QPainter painter(this);
    storage.drawAll(painter);
}

void MainWindow::mousePressEvent(QMouseEvent *event)
```

```
{
    bool ctrlPressed = event->modifiers() & Qt::ControlModifier;
    bool leftClick = event->button() == Qt::LeftButton;

    if (leftClick)
    {
        CCircle *circle = storage.getCircleAt(event->x(), event->y());

        if (circle)
        {
            if (!ctrlPressed)
                storage.deselectAll();
            circle->setSelected(!circle->isSelected());
        }
        else
        {
            if (!ctrlPressed)
                storage.deselectAll();
            storage.add(CCircle(event->x(), event->y()));
        }
        update();
    }
}

void MainWindow::keyPressEvent(QKeyEvent *event)
{
    if (event->key() == Qt::Key_Delete)
    {
        storage.removeSelected();
        update();
    }
}
```

```
#include "mvcmainwindow.h"
#include #include
```

```
MVCMainWindow::MVCMainWindow(QWidget *parent) :
QMainWindow(parent) { m_model = new NumberModel(this); m_view = new
NumberView(m_model, this);
setCentralWidget(m_view);
setWindowTitle("Лабораторная работа 3 - MVC");
resize(400, 300);

}
MVCMainWindow::~MVCMainWindow() { }

#include "mainwindow.h"
#include "mvcmainwindow.h"
#include <QApplication>
#include <QMessageBox>
#include <QPushButton>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    // Диалог выбора лабораторной работы
    QMessageBox choiceBox;
    choiceBox.setWindowTitle("Выбор лабораторной работы");
    choiceBox.setText("Какую часть лабораторной работы запустить?");

    QPushButton *part1Button = choiceBox.addButton("Часть 1 - Круги",
QMessageBox::ActionRole);
    QPushButton *part2Button = choiceBox.addButton("Часть 2 - MVC",
QMessageBox::ActionRole);
    QPushButton *cancelButton = choiceBox.addButton("Отмена",
QMessageBox::RejectRole);

    choiceBox.exec();

    if (choiceBox.clickedButton() == part1Button) {
        // Запуск твоей части с кругами
```

```
        MainWindow w;
        w.show();
        return a.exec();
    } else if (choiceBox.clickedButton() == part2Button) {
        // Запуск части с MVC
        MVCMainWindow w;
        w.show();
        return a.exec();
    }

    return 0;
}
```