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

Кафедра ВМиК

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

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

В рамках лабораторной работы необходимо разобраться:

- каким образом проектировать и реализовывать сложные иерархии классов;
- каким образом выносить общую функциональность в базовые классы;
- каким образом организовать взаимодействие между графическим интерфейсом и объектной моделью;
- каким образом реализовать стандартные операции редактирования в графических приложениях.

Задание

1. Создание иерархии классов графических объектов

- Разработать базовый класс Shape с общей функциональностью:
  - Позиция, размер, цвет
  - Методы отрисовки, проверки попадания точки, перемещения
  - Контроль границ рабочей области
- Создать классы-наследники для различных фигур:
  - Circle, Rectangle, Square, Ellipse, Triangle, Line
- Вынести в базовый класс максимальное количество общей логики

2. Реализация графического интерфейса

- Создать главное окно с:
  - Меню и панелью инструментов для выбора типа создаваемого объекта
  - Рабочей областью для манипуляций с объектами
- Реализовать обработку изменения размера рабочей области

3. Функциональность создания объектов

- Выбор типа объекта из меню/панели инструментов
- Создание нового объекта в рабочей области (например, по клику мыши)
- Использование контейнера из Л.Р.3 для хранения объектов

4. Система манипуляции объектами

- **Выделение объектов:**
  - Одиночное выделение при клике
  - Множественное выделение с Ctrl
  - Визуальное отображение выделения (рамка, изменение цвета)
- **Управление с клавиатуры:**
  - Перемещение стрелками (← → ↑ ↓)
  - Удаление клавишей Delete
  - Изменение размера с Shift + стрелки
- **Изменение свойств:**
  - Смена цвета через стандартный диалог выбора цвета
  - Контекстное меню для операций с объектами

5. Контроль границ рабочей области

- Реализовать в базовом классе Shape проверку на выход за границы
- Автоматическая коррекция положения/размера при перемещении и изменении
- Запрет операций, приводящих к выходу объекта за пределы области

Ход выполнения лабораторной работы

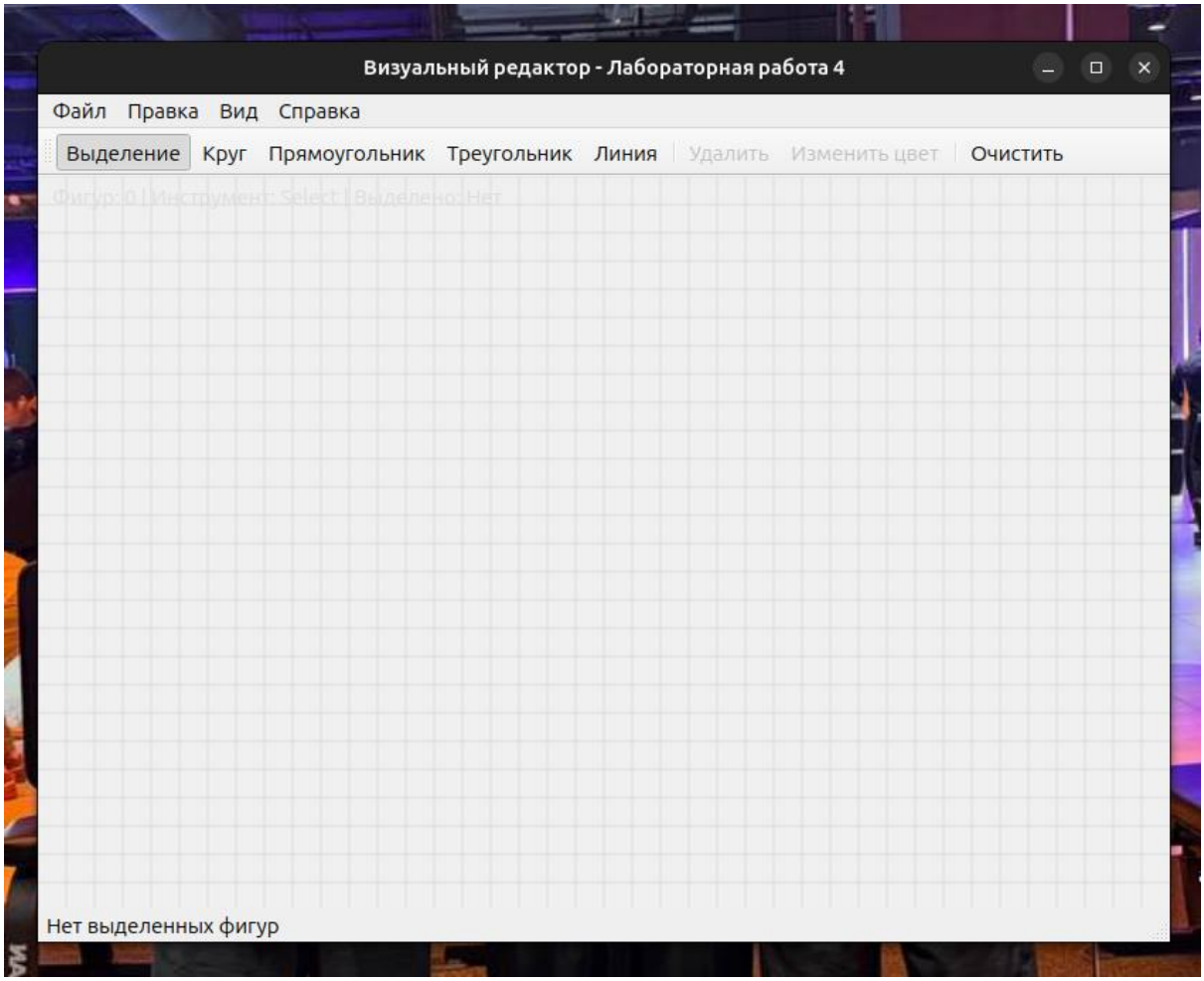


Рис .1 Начальное меню. Визуальный редактор.

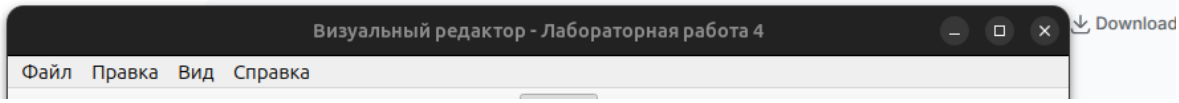


Рис .2 **Файл: Новый** (Ctrl+N) - очищает весь холст. **Выход** (Ctrl+Q) - закрывает программу.

Правка: Удалить (Delete) - удаляет выделенные фигуры **Изменить цвет** (Ctrl+L) - открывает палитру цветов **Выделить все** (Ctrl+A) - выделяет все фигуры **Снять выделение** (Ctrl+D) - снимает все выделения.

Вид: Панель инструментов - показывает/скрывает панель инструментов.

Справка: О программе - информация о программе и управлении

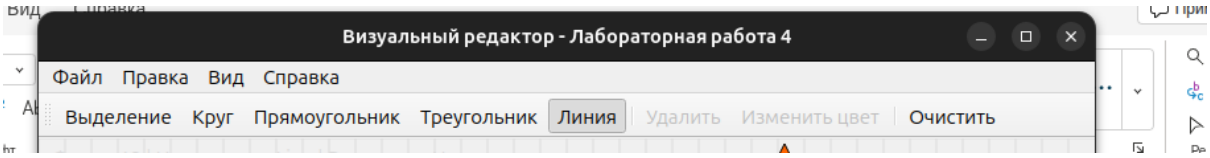


Рис .3 **ПАНЕЛЬ ИНСТРУМЕНТОВ**

Кнопки инструментов: Выделение (S) - выбор и манипуляция фигурами. **Круг** (C) - создание кругов при клике. **Прямоугольник** (R) - создание прямоугольников **Треугольник** (T) - создание треугольников. **Линия** (L) - создание линий. **Кнопки действий: Удалить** - удаляет выделенные фигуры. **Изменить цвет** - открывает диалог выбора цвета. **Очистить** - полностью очищает холст.

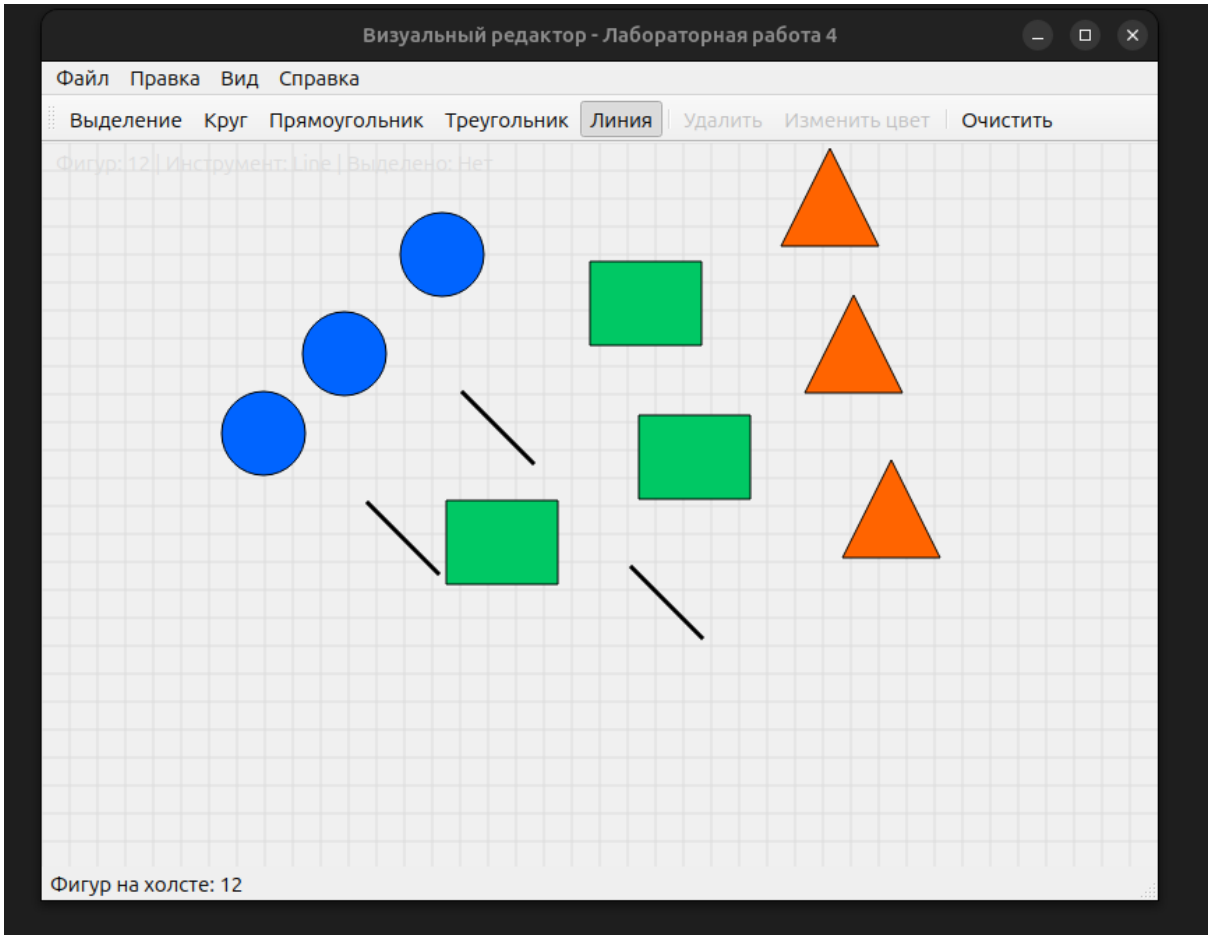
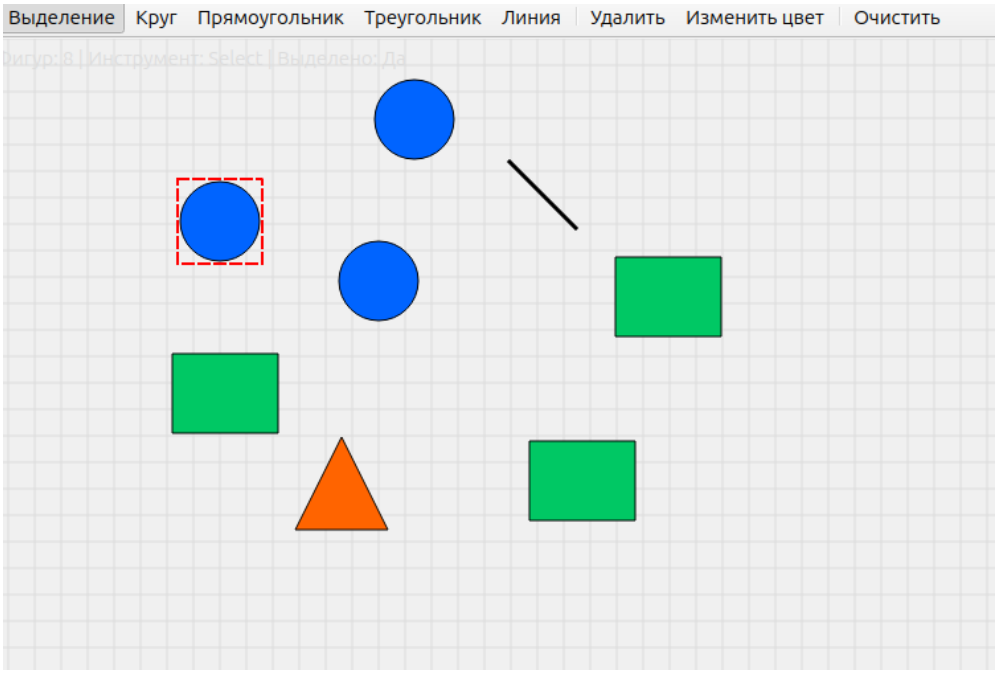


Рис .4 **ОБЛАСТЬ РИСОВАНИЯ (холст)**

Визуальные элементы: Сетка - светло-серая сетка 20x20px для удобства позиционировани. **Фигуры** - отображаются с заливкой и контуром. **Выделение** - красная пунктирная рамка вокруг выделенных фигур. **Маркеры** - для линий показываются маркеры на концах. **Цвета по**

умолчанию. Круги - синий. Прямоугольники - зеленый. Треугольники - оранжевый. Линии - черный



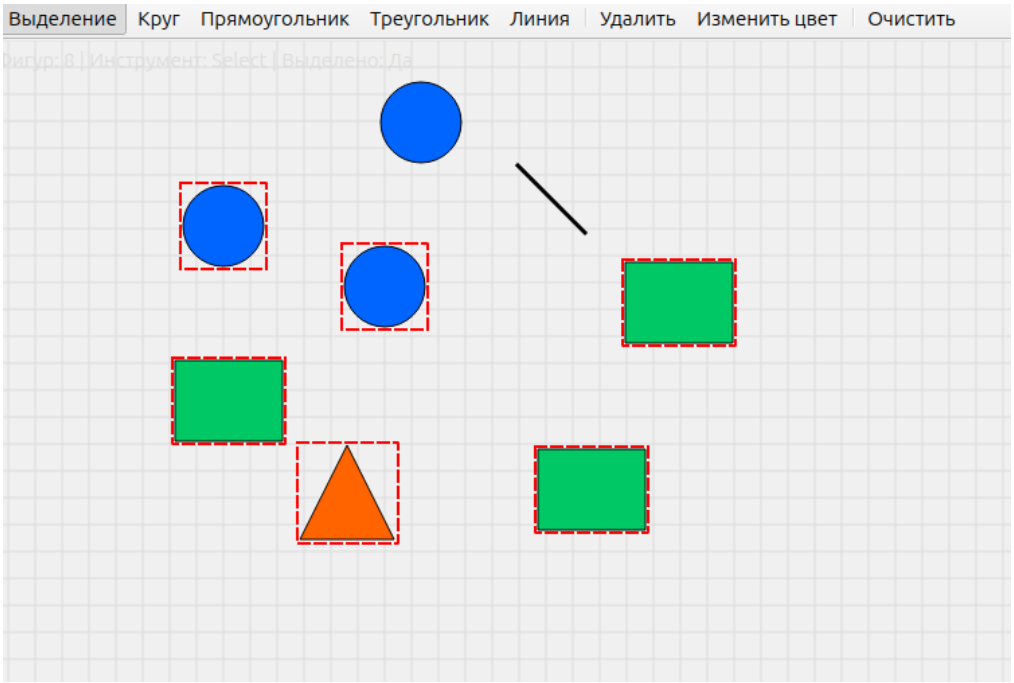


Рис. 5, Рис. 6 Пример одиночного и множественного выделения.

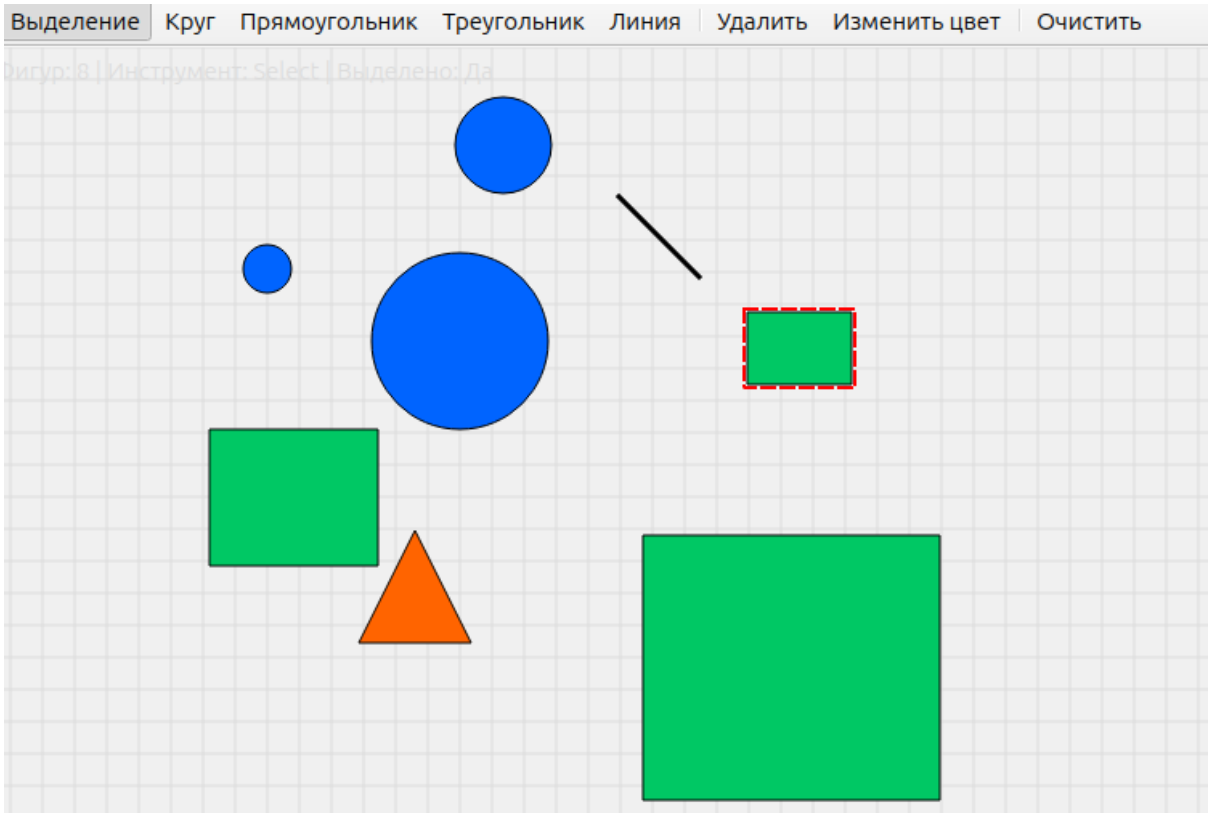
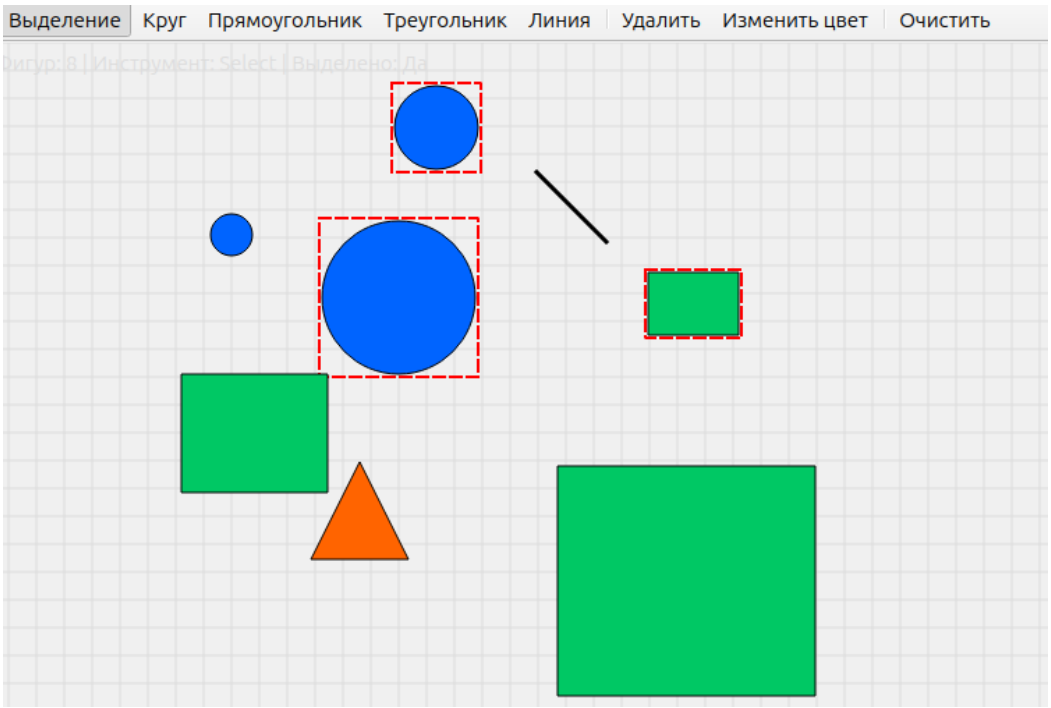


Рис. 7 Пример изменение размера Зажать Ctrl и использовать:
Ctrl + "+" - увеличить на 5px
Ctrl + "-" - уменьшить на 5px



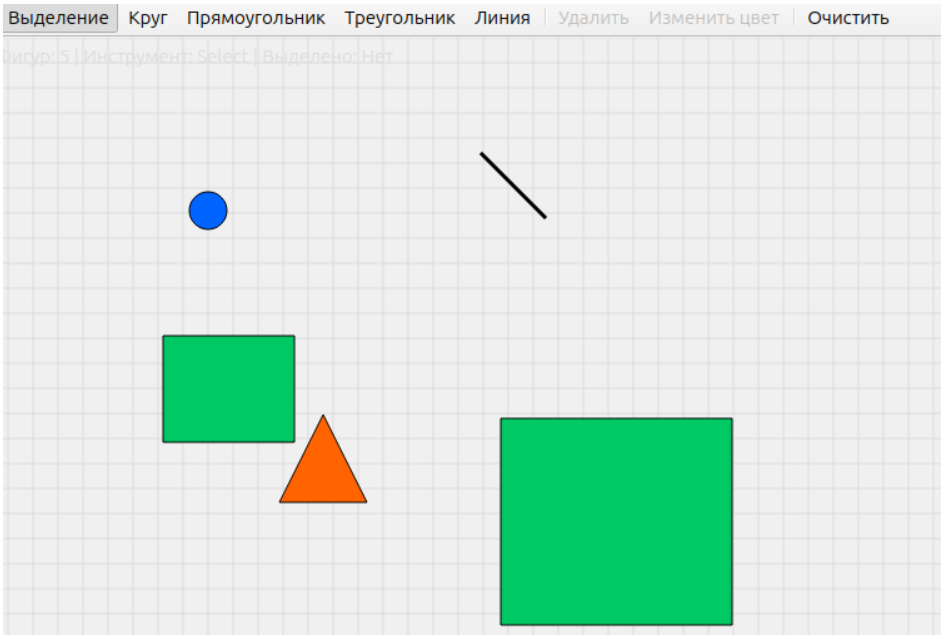
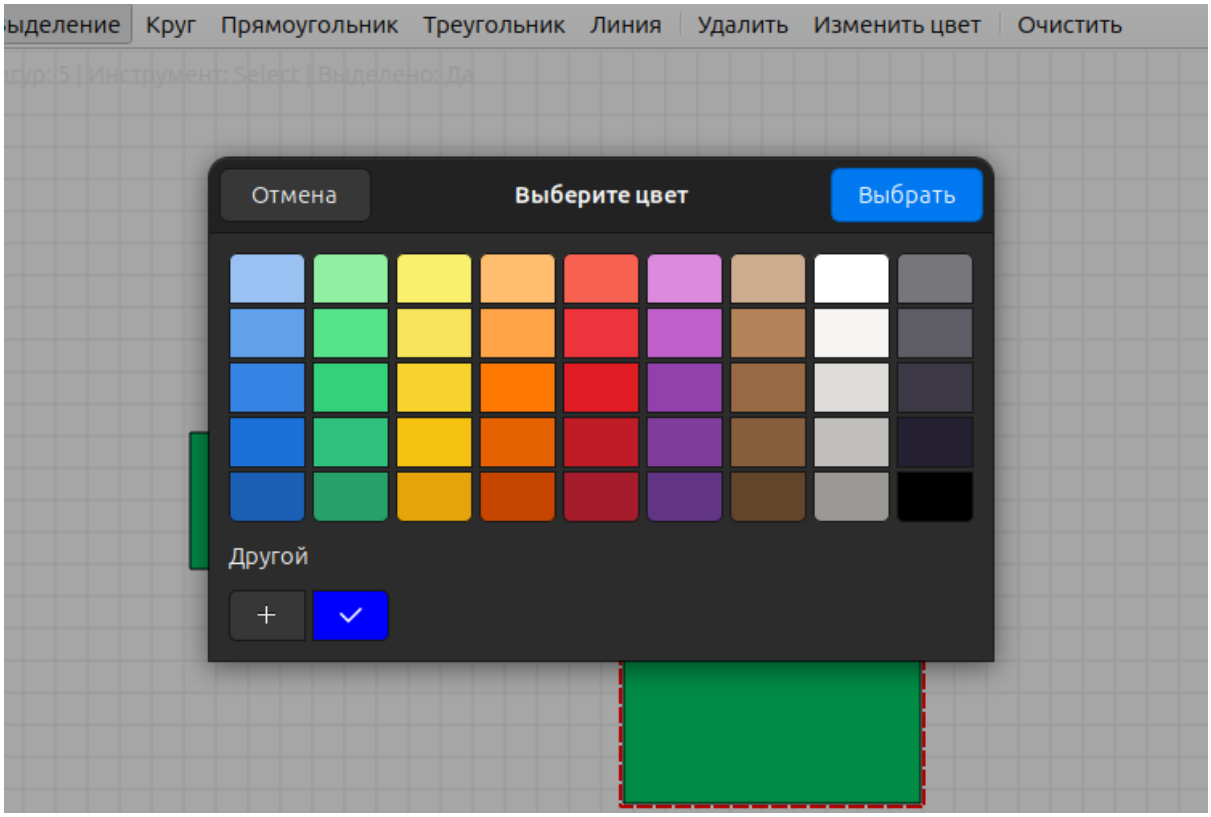


Рис. 8, Рис. 9 Пример удаления фигур. 1. Выделить фигуру(ы)
2. Нажать Delete или кнопку "Удалить"



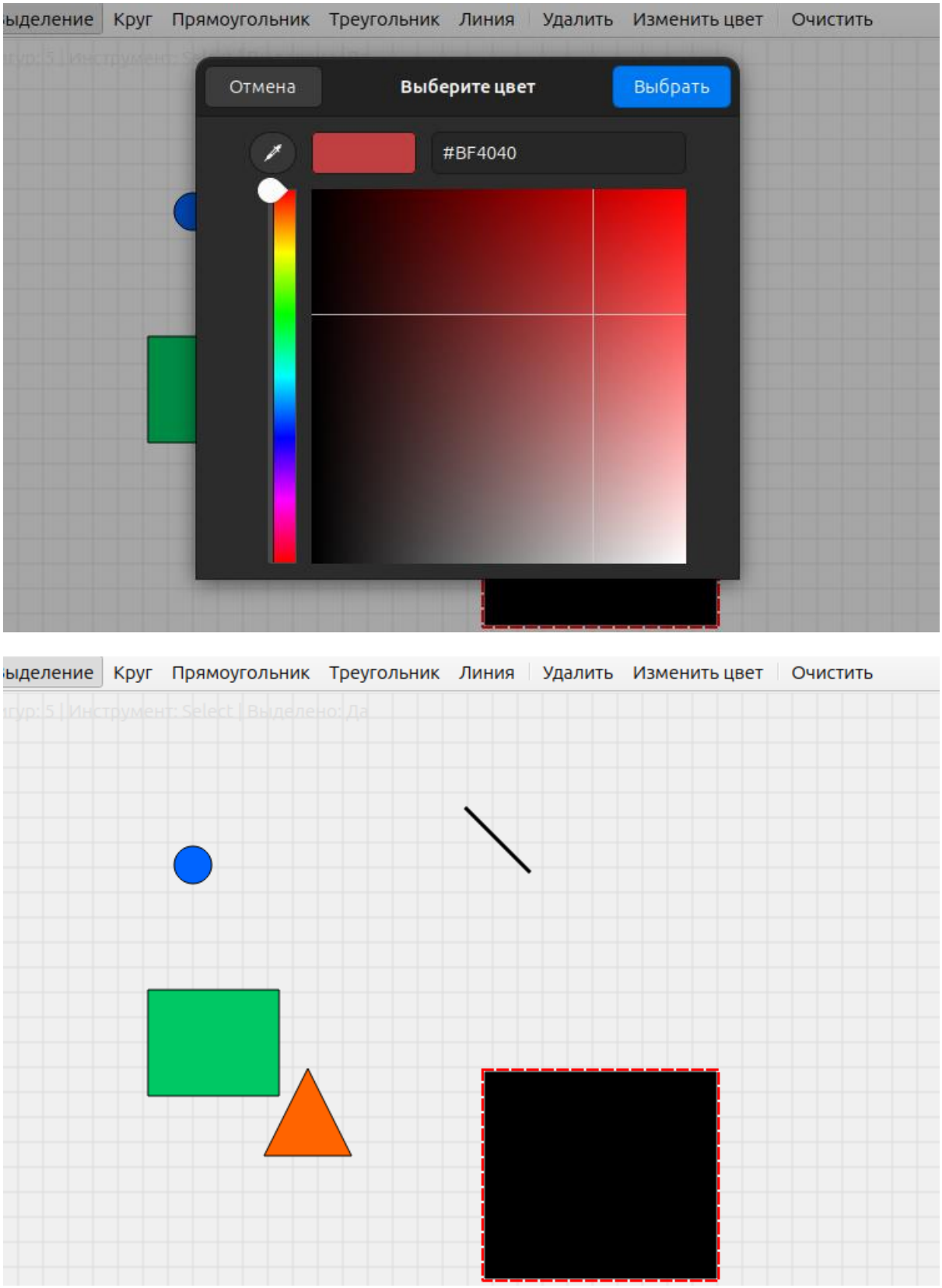


Рис. 10, Рис. 11, Рис. 12 Пример изменения цвета фигуры. Выбор цвета. Или можно воспользоваться палитрой.

1. Выделить фигуру(ы) 2. Нажать "Изменить цвет" или Ctrl+L
3. Выбрать цвет в диалоговом окне 4. Нажать OK

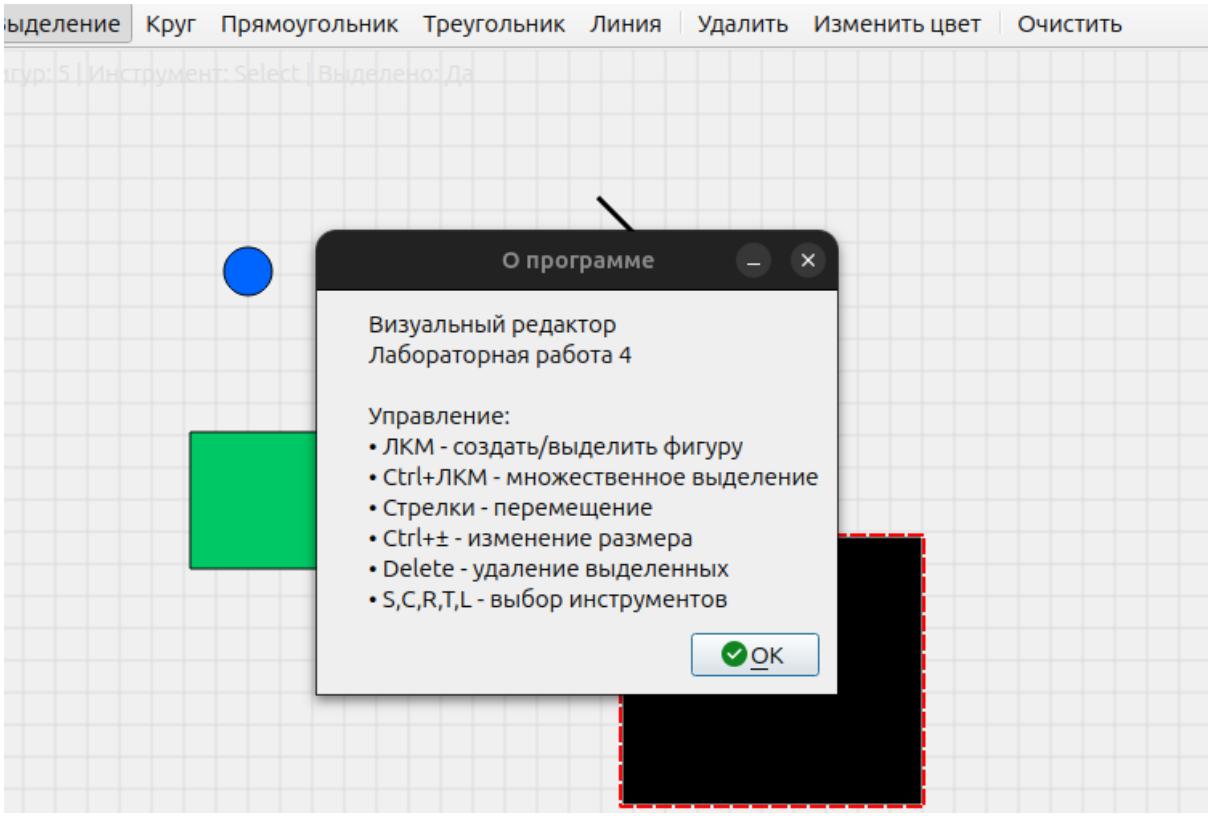


Рис. 13, Пример окна ”О программе”.

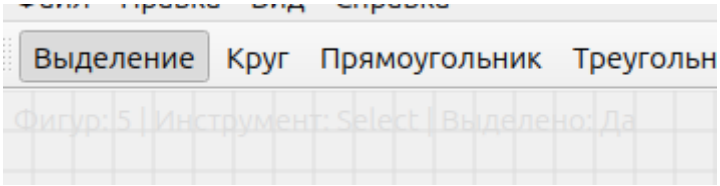


Рис. 14, Информационная панель сверху. **Отображает: Количество фигур** - всего на холсте. **Активный инструмент** - текущий выбранный инструмент. **Статус выделения** - есть ли выделенные фигуры

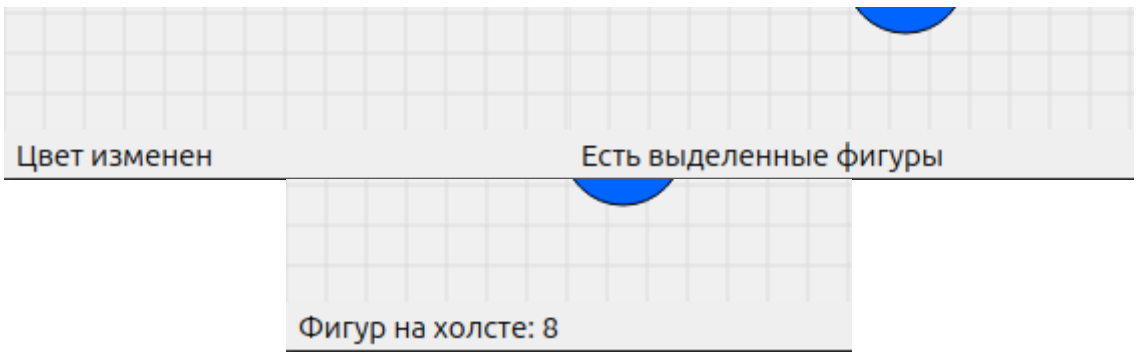


Рис. 15, Рис. 16, Рис. 17 Пример визуального состояние.

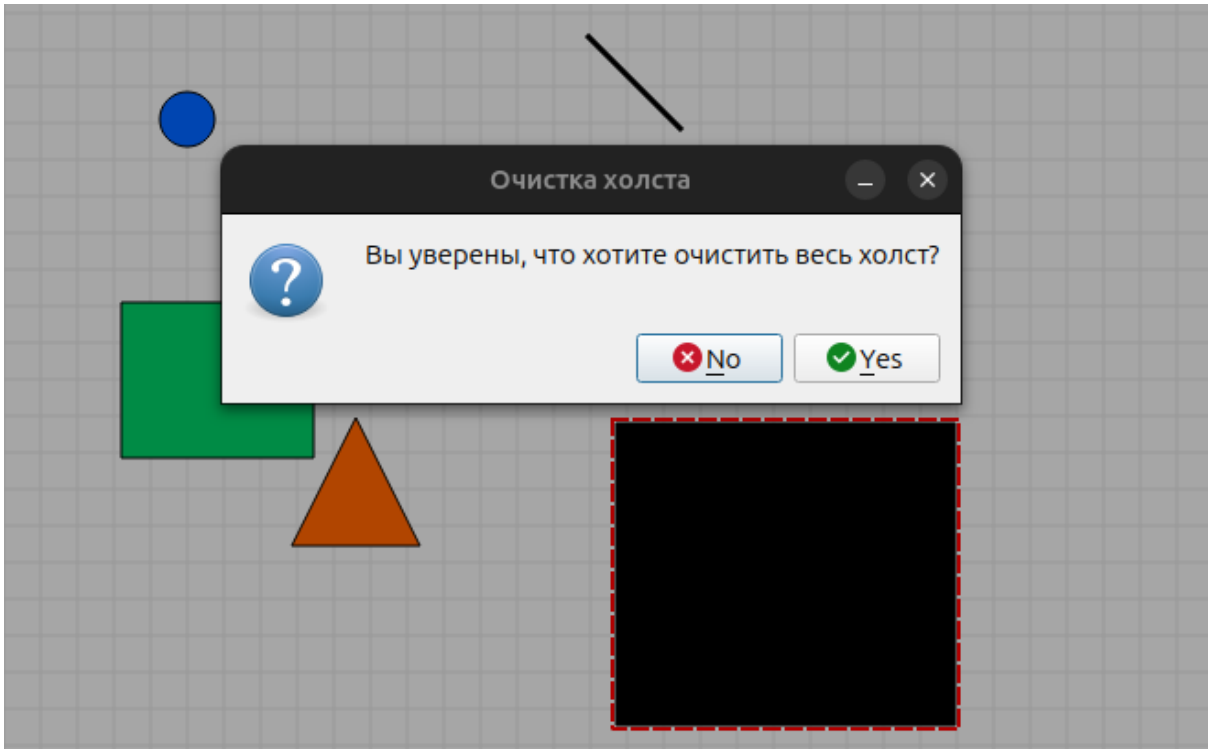


Рис. 18 Рис. 19 Пример использование функции очистки холста.

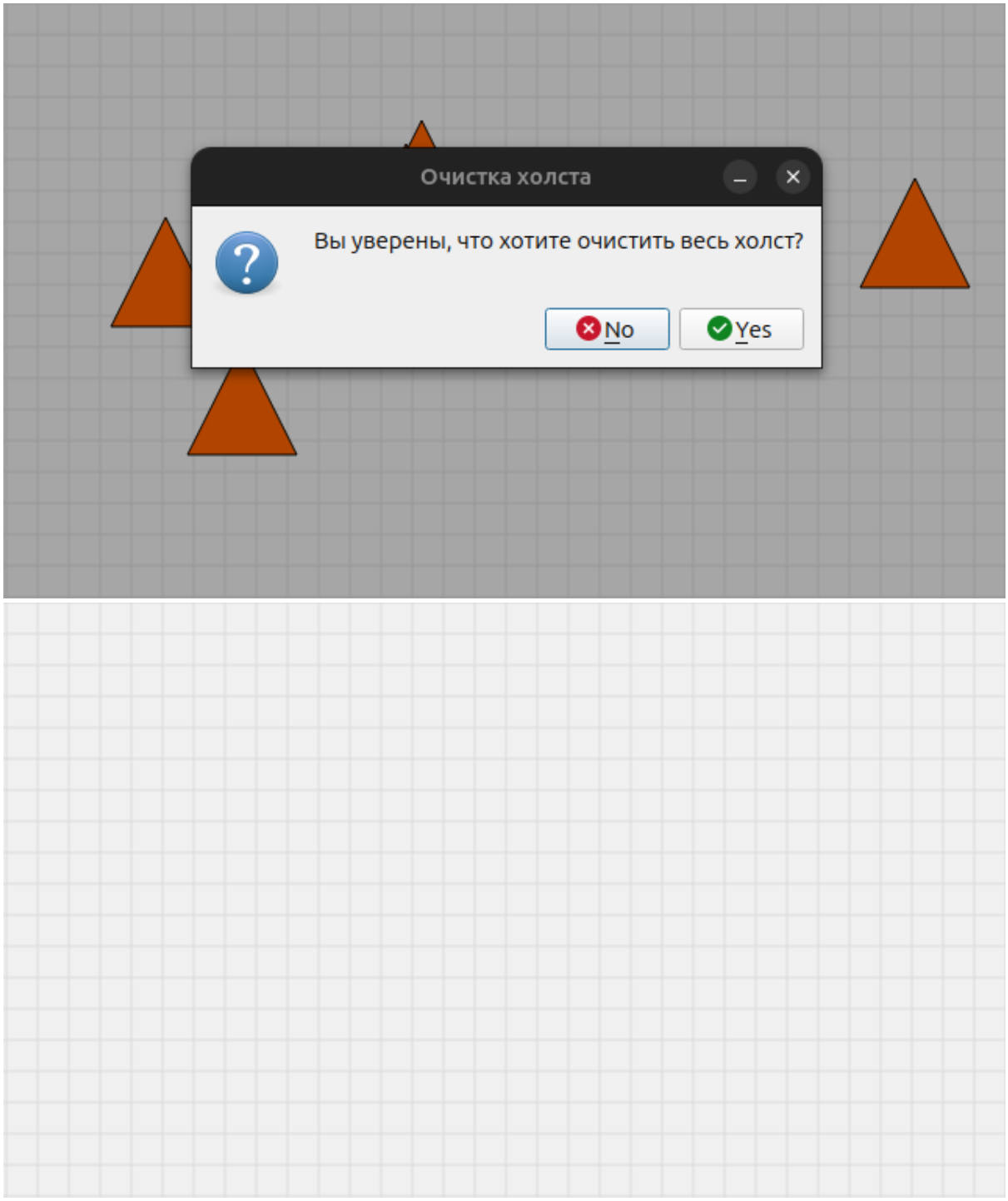


Рис. 20 Рис. 21 Пример использование функции создания нового холста.

Выводы по лабораторной работе

В результате выполнения лабораторной работы был создан полнофункциональный векторный графический редактор, что позволило углубить понимание принципов наследования и полиморфизма в объектно-ориентированном программировании.

Была разработана расширяемая иерархия классов с базовым абстрактным классом BaseShape, что продемонстрировало преимущества вынесения общей функциональности на верхний уровень иерархии. Реализация различных типов фигур (круг, прямоугольник, треугольник, линия) показала практическое применение виртуальных методов и динамического полиморфизма.

Освоены принципы проектирования пользовательских интерфейсов - создана интуитивно понятная система управления с панелью инструментов, контекстным меню и горячими клавишами. Реализовано профессиональное взаимодействие с пользователем через стандартные диалоговые окна и визуальную обратную связь.

Особое внимание было уделено управлению памятью с использованием умных указателей и созданию отказоустойчивой архитектуры, способной обрабатывать различные сценарии пользовательского взаимодействия без нарушения целостности данных.

Приложение №1

```
#include "mainwindow.h"
#include <QToolBar>
#include <QMenu>
#include <QAction>
#include <QMenuBar>
#include <QStatusBar>
```

```
#include <QColorDialog>
#include <QMessageBox>
#include <QIcon>
#include <QKeySequence>
#include <QDebug>

MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent) { //
Создаем центральный виджет m_shapeWidget = new ShapeWidget(this);
setCentralWidget(m_shapeWidget);
// Создаем интерфейс
createToolBar();
createMenu();

// Подключаем сигналы
connect(m_shapeWidget, &ShapeWidget::selectionChanged,
        this, &MainWindow::onSelectionChanged);
connect(m_shapeWidget, &ShapeWidget::shapesCountChanged,
        this, [this](int count) {
            statusBar()->showMessage(QString("Фигур на холсте:
%1").arg(count));
        });

// Настраиваем окно
setWindowTitle("Визуальный редактор - Лабораторная работа 4");
resize(800, 600);

// Статус бар
statusBar()->showMessage("Готов к работе");

// Обновляем состояние действий
updateActions();

}
MainWindow::~MainWindow() { }
void MainWindow::createToolBar() { m_toolBar =
addToolBar("Инструменты");
```

```
// Действие выбора
m_selectAction = new QAction("Выделение", this);
m_selectAction->setCheckable(true);
m_selectAction->setChecked(true);
m_selectAction->setShortcut(QKeySequence("S"));connect(m_selectAction,
&QAction::triggered, this, [this]() {
m_shapeWidget->setCurrentTool(ShapeWidget::Tool::Select);
});

// Действие круга
m_circleAction = new QAction("Круг", this);
m_circleAction->setCheckable(true);
m_circleAction-
>setShortcut(QKeySequence("C"));connect(m_circleAction,&QAction::triggere
d, this, [this]() {
    m_shapeWidget->setCurrentTool(ShapeWidget::Tool::Circle);
});

// Действие прямоугольника
m_rectangleAction = new QAction("Прямоугольник", this);
m_rectangleAction->setCheckable(true);
m_rectangleAction->setShortcut(QKeySequence("R"));
connect(m_rectangleAction, &QAction::triggered, this, [this]() {
    m_shapeWidget->setCurrentTool(ShapeWidget::Tool::Rectangle);
});

// Действие треугольника
m_triangleAction = new QAction("Треугольник", this);
m_triangleAction->setCheckable(true);
m_triangleAction->setShortcut(QKeySequence("T"));
connect(m_triangleAction, &QAction::triggered, this, [this]() {
    m_shapeWidget->setCurrentTool(ShapeWidget::Tool::Triangle);
});

// Действие линии
m_lineAction = new QAction("Линия", this);
m_lineAction->setCheckable(true);
```

```
m_lineAction->setShortcut(QKeySequence("L"));
connect(m_lineAction, &QAction::triggered, this, [this]() {
    m_shapeWidget->setCurrentTool(ShapeWidget::Tool::Line);
});

// Группа для эксклюзивного выбора инструментов
QActionGroup *toolGroup = new QActionGroup(this);  // ← ОБЪЯВЛЯЕМ
ПЕРЕМЕННУЮ
toolGroup->addAction(m_selectAction);
toolGroup->addAction(m_circleAction);
toolGroup->addAction(m_rectangleAction);
toolGroup->addAction(m_triangleAction);
toolGroup->addAction(m_lineAction);  // ← ДОБАВЛЯЕМ ЛИНИЮ В
ГРУППУ
toolGroup->setExclusive(true);

// Действия управления
m_deleteAction = new QAction("Удалить", this);
m_deleteAction->setShortcut(QKeySequence::Delete);
connect(m_deleteAction, &QAction::triggered, this, [this]() {
    m_shapeWidget->deleteSelectedShapes();
});

m_colorAction = new QAction("Изменить цвет", this);
m_colorAction->setShortcut(QKeySequence("Ctrl+L"));
connect(m_colorAction, &QAction::triggered, this,
&MainWindow::onChangeColor);

m_clearAction = new QAction("Очистить", this);
connect(m_clearAction, &QAction::triggered, this,
&MainWindow::onClearCanvas);

// Добавляем действия на панель инструментов
m_toolBar->addAction(m_selectAction);
m_toolBar->addAction(m_circleAction);
m_toolBar->addAction(m_rectangleAction);
m_toolBar->addAction(m_triangleAction);
```

```
m_toolBar->addAction(m_lineAction);  // ← ДОБАВЛЯЕМ КНОПКУ
ЛИНИИ
m_toolBar->addSeparator();
m_toolBar->addAction(m_deleteAction);
m_toolBar->addAction(m_colorAction);
m_toolBar->addSeparator();
m_toolBar->addAction(m_clearAction);

}
//создание меню void MainWindow::createMenu() { // Меню Файл QMenu
*fileMenu = menuBar()->addMenu("Файл");
QAction *newAction = new QAction("Новый", this);
newAction->setShortcut(QKeySequence::New);
connect(newAction, &QAction::triggered, this,
&MainWindow::onClearCanvas);

QAction *exitAction = new QAction("Выход", this);
exitAction->setShortcut(QKeySequence::Quit);
connect(exitAction, &QAction::triggered, this, &QMainWindow::close);

fileMenu->addAction(newAction);
fileMenu->addSeparator();
fileMenu->addAction(exitAction);

// Меню Правка
QMenu *editMenu = menuBar()->addMenu("Правка");
editMenu->addAction(m_deleteAction);
editMenu->addAction(m_colorAction);
editMenu->addSeparator();

QAction *selectAllAction = new QAction("Выделить все", this);
selectAllAction->setShortcut(QKeySequence::SelectAll);
connect(selectAllAction, &QAction::triggered, this, [this]() {
   statusBar()->showMessage("Выделение всех фигур - в разработке");
});
editMenu->addAction(selectAllAction);
```

```
QAction *deselectAllAction = new QAction("Снять выделение", this);
deselectAllAction->setShortcut(QKeySequence("Ctrl+D"));
connect(deselectAllAction, &QAction::triggered, this, [this]() {
    m_shapeWidget->getStorage().deselectAll();
    m_shapeWidget->update();
    onSelectionChanged();
});
editMenu->addAction(deselectAllAction);

// Меню Вид
QMenu *viewMenu = menuBar()->addMenu("Вид");
viewMenu->addAction(m_toolBar->toggleViewAction());

// Меню Справка
QMenu *helpMenu = menuBar()->addMenu("Справка");


QAction *aboutAction = new QAction("О программе", this);
connect(aboutAction, &QAction::triggered, this, []() {
    QMessageBox::about(nullptr, "О программе",
        "Визуальный редактор\n"
        "Лабораторная работа 4\n\n"
        "Управление:\n"
        "• ЛКМ - создать/выделить фигуру\n"
        "• Ctrl+ЛКМ - множественное выделение\n"
        "• Стрелки - перемещение\n"
        "• Ctrl+± - изменение размера\n"
        "• Delete - удаление выделенных\n"
        "• S,C,R,T,L - выбор инструментов");
});
helpMenu->addAction(aboutAction);

}

void MainWindow::onSelectionChanged() { updateActions();
// Обновляем статус бар
if (m_shapeWidget->getStorage().hasSelectedShapes()) {
```

```
    statusBar()->showMessage("Есть выделенные фигуры");
} else {
    statusBar()->showMessage("Нет выделенных фигур");
}

}
void MainWindow::onChangeColor() { if (!m_shapeWidget-
>getStorage().hasSelectedShapes()) { QMessageBox::information(this,
"Информация", "Сначала выделите фигуры"); return; }
QColor color = QColorDialog::getColor(Qt::blue, this, "Выберите цвет");
if (color.isValid()) {
    m_shapeWidget->changeSelectedColor(color);
    statusBar()->showMessage("Цвет изменен");
}

}
void MainWindow::onClearCanvas() { int result =
QMessageBox::question(this, "Очистка холста", "Вы уверены, что хотите
очистить весь холст?", QMessageBox::Yes | QMessageBox::No);
if (result == QMessageBox::Yes) {
    m_shapeWidget->getStorage().clear();
    m_shapeWidget->update();
    statusBar()->showMessage("Холст очищен");
    emit m_shapeWidget->shapesCountChanged(0);
}

}
void MainWindow::updateActions() { bool hasSelection = m_shapeWidget-
>getStorage().hasSelectedShapes();
m_deleteAction->setEnabled(hasSelection);
m_colorAction->setEnabled(hasSelection);

}
```