МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Нижегородский государственный университет им. Н.И. Лобачевского»
Национальный исследовательский университет

Институт информационных технологий, математики и механики
Кафедра программной инженерии

ОТЧЕТ

по дисциплине «Разработка мобильных приложений»

«Основы конструирования интерфейсов пользователя»

Выполнил:
студент группы 381706-1
Денисов В. Л.

Проверил:
доцент кафедры программной инженерии
Борисов Н. А.

Нижний Новгород
2020.

Содержание

1 Цели

Целью данной лабораторной работы является освоение базовых навыков построения пользовательских интерфейсов, позиционирования, отрисовки и перемещения элементов. А также изучение подхода к реализации анимации элементов, созданию диалогов и взаимодействию с ними.

2 Постановка задачи

1. Создать новый проект со стандартной заготовкой приложения.
2. Нарисовать 3 квадрата красного, зелёного и синего цветов следующим образом:

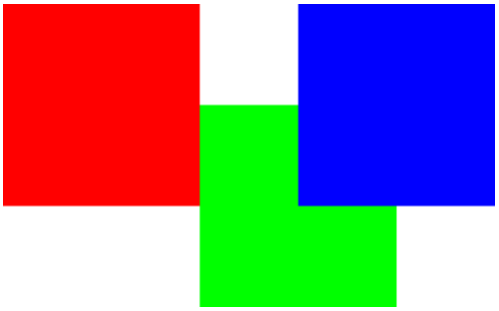

Рисунок 1 Взаимное расположение 3-х квадратов.

3. Поместить текст “Квадрат” белого цвета по центру синего квадрата.
4. Нарисовать 5 квадратов с использованием Column и Row следующим образом:

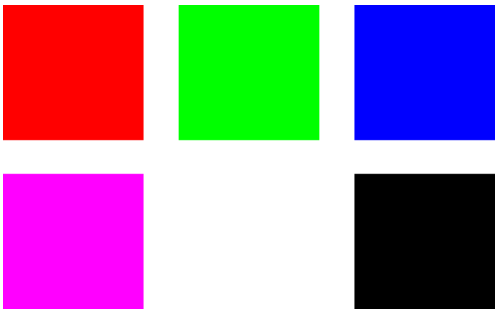

Рисунок 2 Взаимное расположение 5-ти квадратов.

5. Нарисовать те же 5 квадратов с использованием Grid.
6. Сделать из квадрата “A” повернутый по часовой стрелке прямоугольник “B” с использованием объектов Translate, Scale и Rotation.
7. Нарисовать квадрат и анимировать его перемещение вниз с увеличением его размера.
8. Реализовать диалог с двумя текстовыми полями, в которые вводятся числа. После нажатия на кнопку “Подтвердить” в консоль выводится сумма чисел. Для преобразования строк к числам использовать функцию *parseInt(“42”).* Валидацией и обработкой ошибок можно пренебречь.

3 Решение поставленных задач

1. Создадим проект со стандартной заготовкой приложения, где файлом главной страницы приложения будет являться *FirstPage.qml*, а файл для диалога, который потребуется создать будет *SumDialog.qml*.

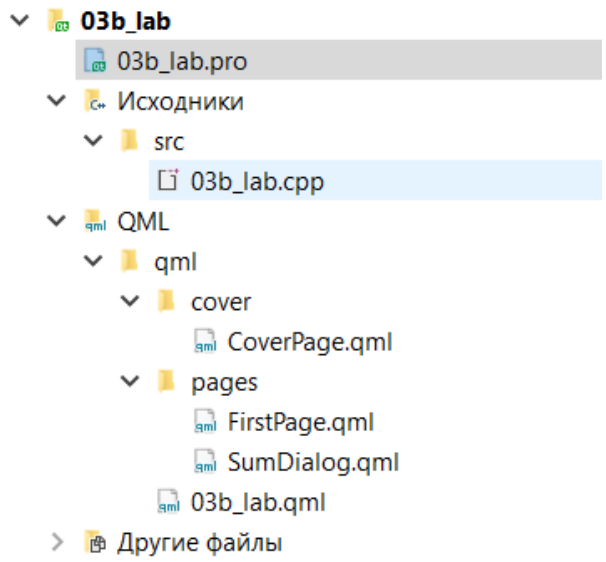

Рисунок 3 Структура проекта.

2. Разместим на странице объект *Column*, в котором будут находиться все задания, за исключением диалога. Внутри него создаем вложенную структуру *Rectangle*, которые будут отображаться друг за другом: от «отца» к «сыну». Для задания размеров, положения, цвета квадрата используем параметры *width, height, x, y, color*.

Рисунок 4 Реализация заданий 2 и 3.

3. Текст на синем квадрате располагается при помощи элемента *Label* с привязкой по якорю к центру этого квадрата.

4. Для реализации этого пункта размещаем последовательно несколько контейнеров *Row*, каждый из которых содержит в себе несколько квадратов, заданных при помощи *Rectangle* по принципу из п.2.

Рисунок 5 Использование контейнера Row.

5. Расположить квадраты аналогично пункту 4 при помощи контейнера *Grid* можно путем использования специальных параметров *columns* и *rows* – число столбцов и строк соответственно.

Рисунок 6 Использование контейнера Grid.

6. Для удобства и наглядности создаем контейнер *Grid* с одной строкой и двумя столбцами, в котором размещаем исходный квадрат и трансформированный. Все операции выполняются при помощи параметра *transform*, в котором устанавливаются *Scale* (масштабирование), *Rotation* (поворот), *Translate* (перемещение).

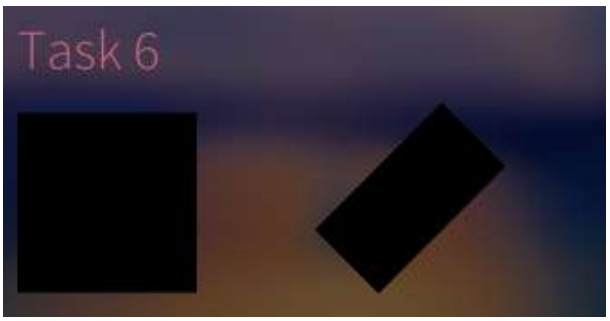

Рисунок 7 Использование параметра transform.

7. Анимацию квадрата путем изменения его размера и перемещения вверх-вниз делаем с помощью *SequentialAnimation*, размещенного внутри *Rectangle*, задающего квадрат. Блок последовательной анимации, в свою очередь, содержит 2 параллельных *ParallelAnimation*: одна перемещает квадрат вниз и увеличивает размер, затем другая выполняет обратные действия.

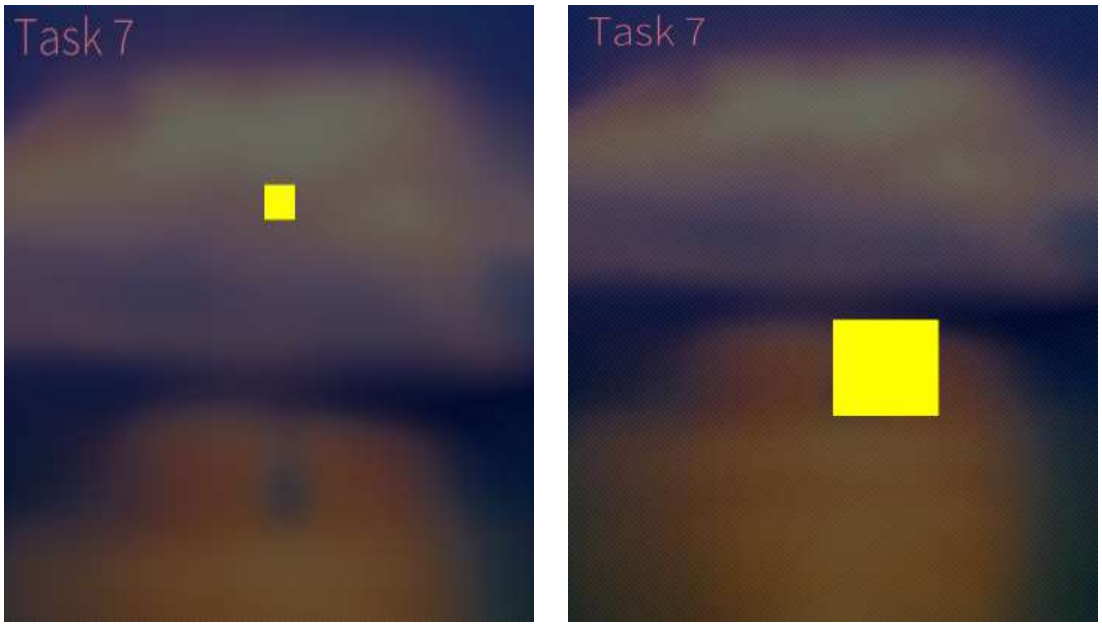

Рисунок 8 Процесс анимации.

8. Элемент выпадающего меню *PullDownMenu* для вызова диалога, который требуется реализовать, размещаем в *SilicaFlickable*. Это требуется принципом построения приложений на SailfishOS. Вызываемый диалог содержит 2 текстовых поля, но данные вводимые туда распознаются через функцию *parceInt()* – обрабатываем только целые числа. Результат суммирования логгируем на консоль.

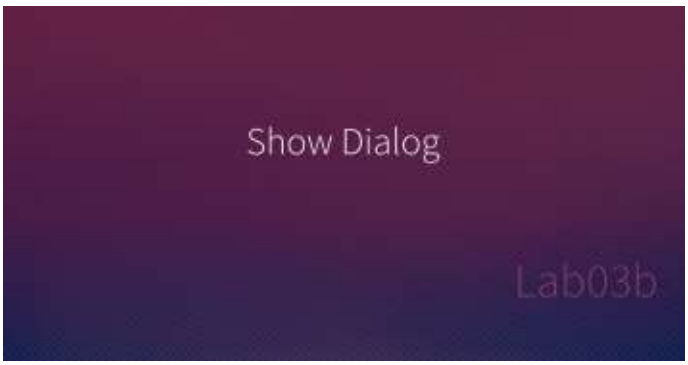

Рисунок 9 Использование PullDownMenu.

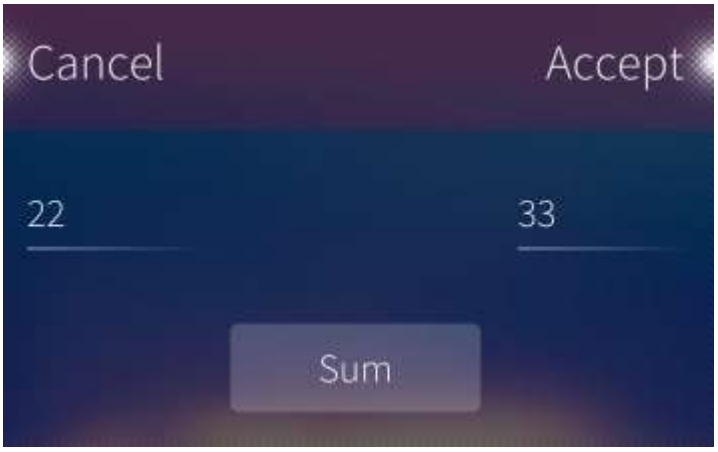

Рисунок 10 Использование диалога.

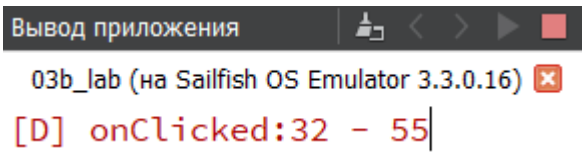

Рисунок 11 Лог, отображаемый на консоль.

4 Приложение

4.1 Файл FirstPage.qml:

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    id: page
    allowedOrientations: Orientation.All

    // To enable PullDownMenu, place our content in a SilicaFlickable
    SilicaFlickable {
        anchors.fill: parent
        contentHeight: column.height
        PullDownMenu {
            MenuItem {
                text: qsTr("Show Dialog")
                onClicked: pageStack.push(Qt.resolvedUrl("SumDialog.qml"))
            }
        }

        Column {
            id: column
            width: page.width
            spacing: Theme.paddingLarge

            property int recSize: 200

            PageHeader {
                title: qsTr("Lab03b")
            }

            // Tasks 2, 3
            Label {
                x: Theme.horizontalPageMargin
                text: qsTr("Tasks 2, 3")
                color: Theme.secondaryHighlightColor
                font.pixelSize: Theme.fontSizeExtraLarge
            }
            Rectangle {
                id: squareRed
                color: "red"
                width: column.recSize
                height: column.recSize
                x: column.recSize / 2
                Rectangle {
                    id: squareGreen
                    color: "green"
                    width: column.recSize
                    height: column.recSize
                    x: column.recSize
                    y: column.recSize / 2
                    Rectangle {
                        id: squareBlue
                        color: "blue"
                        width: column.recSize
                        height: column.recSize
                        x: column.recSize / 2
                        y: -column.recSize / 2
                        Label {
                            text: qsTr("Квадрат")
```

```
                        color: "white"
                        anchors.centerIn: squareBlue
                    }
                }
            }
        }

        // Task separator
        Rectangle { width: column.recSize; height: column.recSize/2;
color: "transparent" }

        // Task 4
        Label {
            x: Theme.horizontalPageMargin
            text: qsTr("Task 4 - Rows")
            color: Theme.secondaryHighlightColor
            font.pixelSize: Theme.fontSizeExtraLarge
        }
        Row {
            spacing: Theme.paddingLarge
            x: Theme.paddingLarge

            Rectangle { width: column.recSize; height: column.recSize;
color: "red" }
            Rectangle { width: column.recSize; height: column.recSize;
color: "green" }
            Rectangle { width: column.recSize; height: column.recSize;
color: "blue" }
        }
        Row {
            spacing: Theme.paddingLarge
            x: Theme.paddingLarge

            Rectangle { width: column.recSize; height: column.recSize;
color: "magenta" }
            Rectangle { width: column.recSize; height: column.recSize;
color: "transparent" }
            Rectangle { width: column.recSize; height: column.recSize;
color: "black" }
        }

        // Task 5
        Label {
            x: Theme.horizontalPageMargin
            text: qsTr("Task 5 - Grid")
            color: Theme.secondaryHighlightColor
            font.pixelSize: Theme.fontSizeExtraLarge
        }
        Grid {
            spacing: Theme.paddingLarge
            x: Theme.paddingLarge
            columns: 3
            rows: 2

            Rectangle { width: column.recSize; height: column.recSize;
color: "red" }
            Rectangle { width: column.recSize; height: column.recSize;
color: "green" }
            Rectangle { width: column.recSize; height: column.recSize;
color: "blue" }
            Rectangle { width: column.recSize; height: column.recSize;
color: "magenta" }
```

```
                Rectangle { width: column.recSize; height: column.recSize;
color: "transparent" }
                Rectangle { width: column.recSize; height: column.recSize;
color: "black" }
            }

            // Task 6
            Label {
                x: Theme.horizontalPageMargin
                text: qsTr("Task 6")
                color: Theme.secondaryHighlightColor
                font.pixelSize: Theme.fontSizeExtraLarge
            }
            Grid {
                spacing: Theme.paddingLarge
                x: Theme.paddingLarge
                columns: 2
                rows: 1
                Rectangle {
                    id: square
                    width: column.recSize
                    height: column.recSize
                    color: "black"
                }
                Rectangle {
                    width: square.width
                    height: square.height
                    color: "black"
                    transform: [
                        Scale { yScale: 0.5 },
                        Rotation { angle: -45 },
                        Translate { x: square.width/2; y:  square.height/2 +
30 }

                    ]
                }
            }

            // Task 7
            Label {
                x: Theme.horizontalPageMargin
                text: qsTr("Task 7")
                color: Theme.secondaryHighlightColor
                font.pixelSize: Theme.fontSizeExtraLarge
            }
            Rectangle {
                id: borderRec
                width: parent.width
                height: 3 * column.recSize
                color: "transparent"
                property int duration: 2500
                Rectangle {
                    id: movingRec
                    x: borderRec.width / 2
                    y: 0
                    width: 0
                    height: 0
                    color: "yellow"
                    SequentialAnimation {
                        loops: Animation.Infinite
                        running: true
                        // Increase the size of the square and move down.
                        ParallelAnimation {
                            PropertyAnimation {
```

```
                                target: movingRec
                                property: "y"
                                from: 0
                                to: 500
                                duration: borderRec.duration
                            }
                            PropertyAnimation {
                                target: movingRec
                                property: "width"
                                from: 0
                                to: column.recSize
                                duration: borderRec.duration
                            }
                            PropertyAnimation {
                                target: movingRec
                                property: "height"
                                from: 0
                                to: column.recSize
                                duration: borderRec.duration
                            }
                        }
                        // Decrease the size of the square and move up.
                        ParallelAnimation {
                            PropertyAnimation {
                                target: movingRec
                                property: "y"
                                from: 500
                                to: 0
                                duration: borderRec.duration
                            }
                            PropertyAnimation {
                                target: movingRec
                                property: "width"
                                from: column.recSize
                                to: 0
                                duration: borderRec.duration
                            }
                            PropertyAnimation {
                                target: movingRec
                                property: "height"
                                from: column.recSize
                                to: 0
                                duration: borderRec.duration
                            }
                        }
                    }
                }
            }

            Rectangle { height: column.recSize; width: column.recSize; color:
"transparent" }
        }
    }
}
```

4.2 Файл SumDialog.qml:

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Dialog {
    Column {
```

```
        anchors.fill: parent
        id: column
        width: parent.width
        spacing: Theme.paddingMedium
        DialogHeader {
            acceptText : "Accept"
            cancelText : "Cancel"
        }
        Row {
            width: parent.width
            spacing: 250

            TextField {
                id: f1
                width: parent.width/3
            }
            TextField {
                id: f2
                width: parent.width/3
            }
        }
        Button {
            width: 250
            anchors.horizontalCenter: parent.horizontalCenter
            text: "Sum"
            onClicked: {
                console.log(parseInt(f1.text) + parseInt(f2.text))
            }
        }
    }
}
```

5 Используемая литература

1. Документация Qt – https://doc.qt.io/qt-5/qmake-project-files.html