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»

«Интерфейс пользователя как граф состояний»

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 изучение способов создания пользовательского интерфейса с конфигурируемыми состояниями, реализация анимированных переходов при смене состояний и создание собственных QML-компонентов.

2 Постановка задачи

1. Создать приложение, отображающее светофор. На экране должно присутствовать 3 разноцветных сигнала, которые загораются и гаснут в том же порядке, что и сигналы светофора. Сделать автоматическую смену состояний.
2. Доработать задание 1 так, чтобы во время зеленого сигнала светофора из одного конца экрана в другой плавно двигалась иконка человечка.
3. Создать приложение, отображающее строку текста вверху экрана. При нажатии на текст он должен плавно перемещаться вниз экрана, поворачивать на 180 градусов и менять цвет. Когда нажатие прекращается, он должен так же плавно возвращаться в исходное положение.
4. Выделить сигналы светофора из задания 1 в отдельный компонент и использовать его.
5. Создать QML компонент со свойством по умолчанию, который берет значение свойства text любого объявленного внутри него объекта и создает Button с тем же текстом. Добавить возможность задавать цвет кнопки при объявлении компонента.
6. Создать приложение-секундомер. На экране должны отображаться значения часов, минут и секунд. Секундомер запускается по сигналу кнопки, при повторном нажатии секундомер останавливается. Для отображения часов, минут и секунд использовать собственные QML компоненты.
7. Добавить обработчик сигналов PageStack, подсчитывающий количество добавленных и удаленных страниц в PageStack.

3 Решение поставленных задач

Создадим проект со стандартной заготовкой приложения, где файлом главной страницы приложения будет являться *FirstPage.qml*, для каждого отдельного задания из постановки задачи тоже будет свой файл.

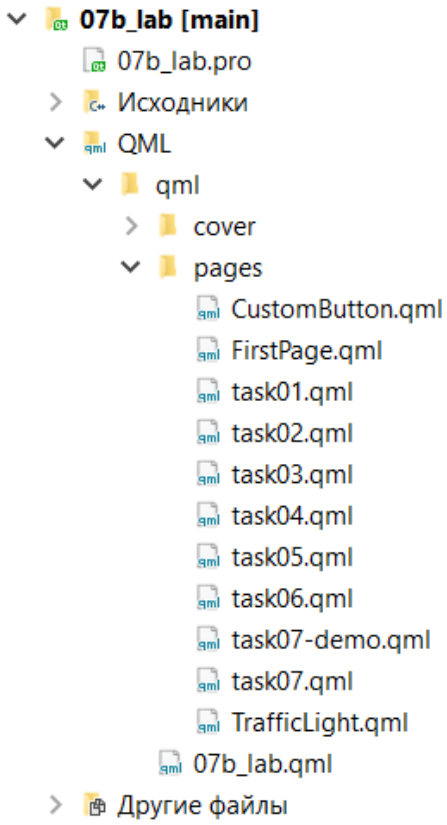

Рисунок 1 Структура проекта.

Вызов демонстрации каждого задания будем производить путем нажатия соответствующей кнопки, расположенной на *FirstPage*. Страница с выбранным заданием будет отправлена в *pageStack*.

1. Для создания светофора, имеющего 3 разноцветных сигнала, которые загораются и гаснут в том же порядке, что и сигналы настоящего светофора используем следующий подход. Создадим *Rectangle*, (*id: traffic_light*) представляющий собой непосредственно сам светофор, внутри которого расположим еще 3 *Rectangle* – каждый из них будет отвечать за свой сигнал светофора.

   Для отображения различных сигналов используем атрибут *states* у внешнего *Rectangle* (*id: traffic_light*) – в атрибуте опишем массив возможных состояний, содержащий элементы *State*, каждый из которых будет изменять свойство цвета у одного или нескольких сигналов.

Переходы между состояниями State выполним путём задания атрибута *transitions* у внешнего *Rectangle* (*id: traffic_light*) – в атрибуте опишем массив переходов, содержащий элементы *Transition*, каждый из которых содержит описание последовательных шагов в анимации для смены соответствующих состояний.

Для автоматической смены сигналов реализуем функцию *change_state*(), которая по текущему состоянию светофора определяет, к какому следует перейти.

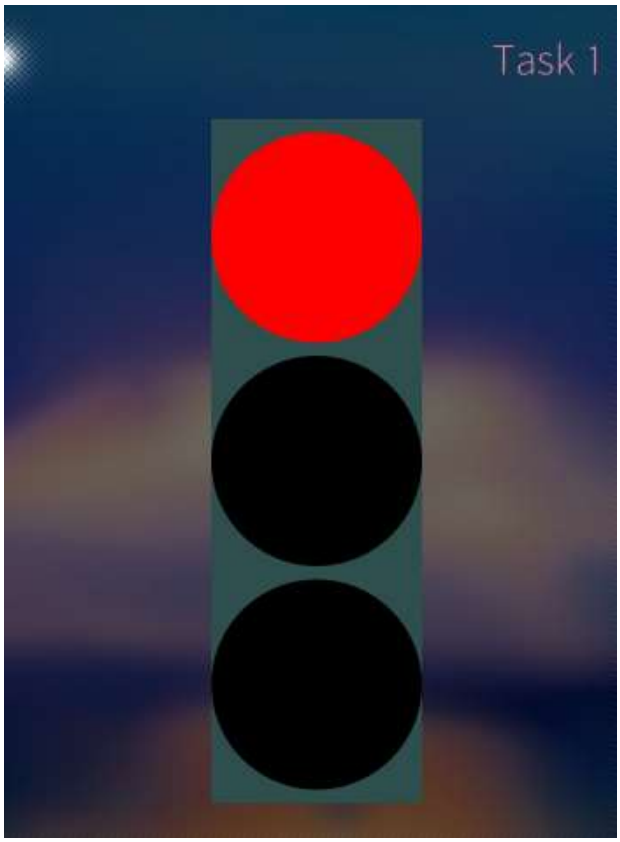

Рисунок 2 Светофор - запрещающий сигнал

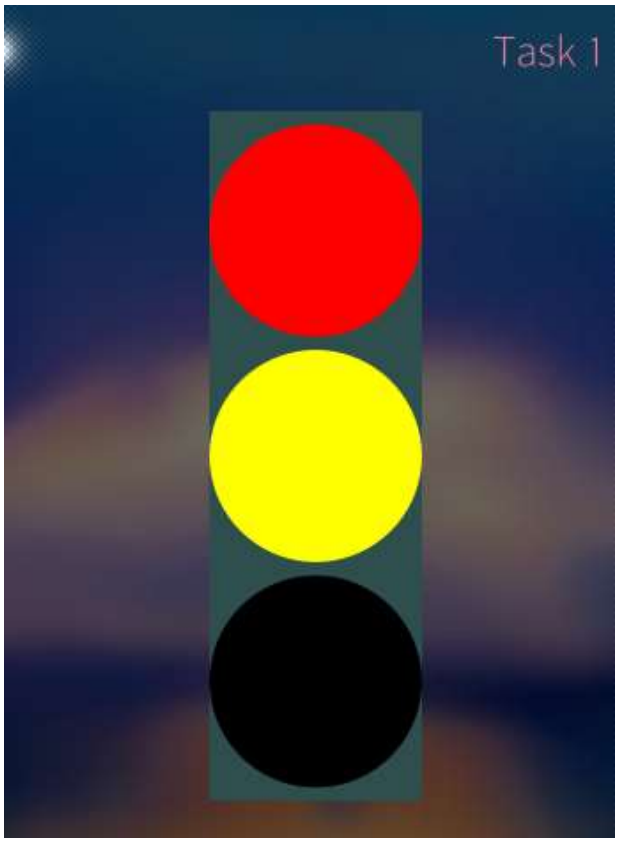

Рисунок 3 Светофор - смена сигналов

2. Перемещения иконки человечка из одного конца экрана в другой во время зеленого сигнала светофора добьёмся за счет небольшого дополнения реализации задания из пункта 1.

   После основного блока светофора, представленного объектом *Rectangle,* (*id: traffic_light*) добавим объект *Image*, который будет отрисовывать иконку человечка. Аналогично реализации пункта 1 задаем атрибуты *states* – человечек справа или слева на экране, *transitions* – анимация перемещения человечка с одной стороны экрана на другую. А также немного дополняем функцию *change_state*() – во время зеленого сигнала изменяем состояние *state* у *Image*.

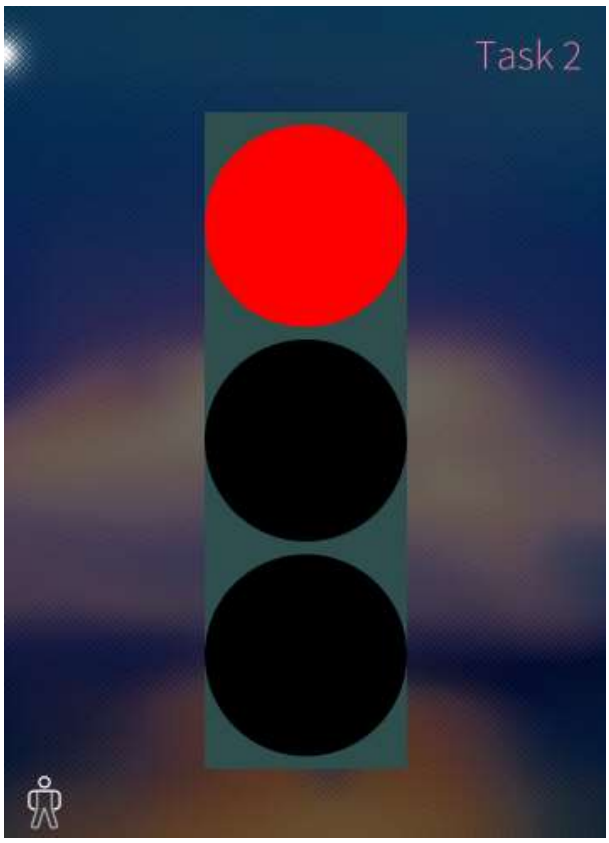

Рисунок 4 Человечек у левой стороны экрана

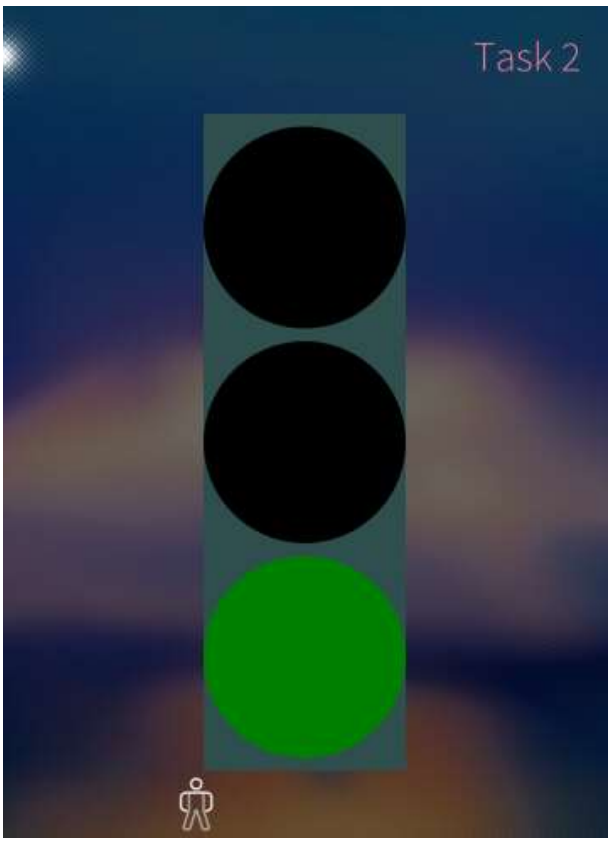

Рисунок 5 Перемещение человечка на другую сторону экрана во время зеленого сигнала светофора

3. Строку текста зададим объектом *Label*. Атрибуту *states* указываем 2 состояния: нормальный и перевернутый (будет изменен цвет текста и его ориентация). Задаем атрибут *transitions* – описание процесса перехода между состояниями (изменение цвета, поворот, перемещение вниз). Помимо всего перечисленного создаем внутри *Label* элемент *MouseArea*, полностью заполняющий его, с двумя обработчиками сигналов: *onPressed* – на нажатие, который изменит текущее состояние *Label* на противоположное, *onReleased* – на прекращение нажатия, аналогично предыдущему.

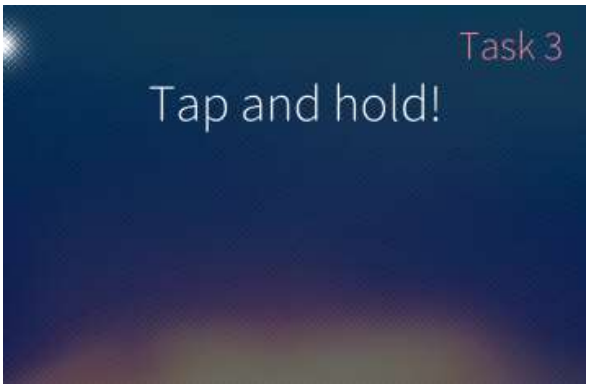

Рисунок 6 Текст в исходном положении

Рисунок 7 Производится нажатие на текст

4. Для выполнения этого задания вынесем весь необходимый функционал для работы светофора в отдельный файл *TrafficLight.qml*. Непосредственно в приложении на странице, демонстрирующей работу этого пункта задания, создаем одноименный элемент *TrafficLight*, который будет предоставлять функционал и внешний вид, описанный в указанном выше файле *TrafficLight.qml.*

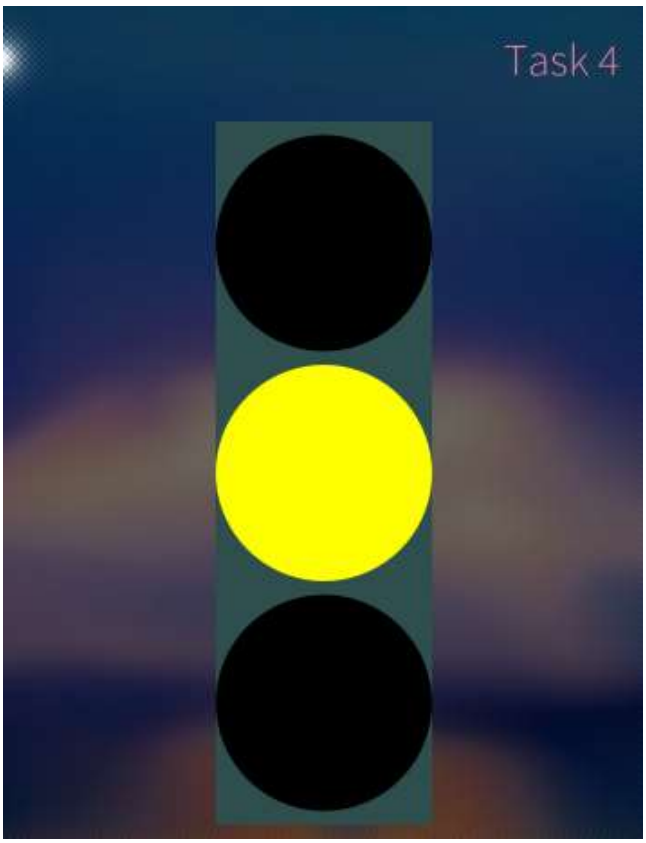

Рисунок 8 Светофор, реализованный в виде отдельного компонента.

5. Аналогично пункту 4 создаем файл *CustomButton.qml*, в котором будет описан наш собственный компонент – пользовательская кнопка. Создаем свойство *default property var some_text* – текст, отображаемый на кнопке, а также вводим псевдоним *property alias color_button*, соответствующий цвету нашей кнопки. Указанные параметры используются при построении встроенного в *Sailfish* объекта *Button*, который оказывается в нашей собственной обёртке.

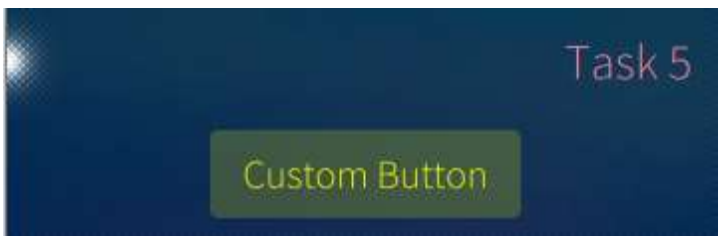

Рисунок 9 Пользовательская кнопка с указанными параметрами отображаемого текста и цвета.

6. Приложение-секундомер реализуем при помощи элемента *Timer*. Задаем интервал изменения таймера в тактах, по событию *onTriggered* обновляем текст на элементе *Text*. Для этого реализуем функцию *dispaly_time*, принимающую текущее значение тактов работы и конвертирующее его в часы, минуты и секунды. Кнопка для запуска/остановки таймера проверяет его статус через вызов встроенного метода *running* и изменяет состояние на противоположное по события *onClicked*.

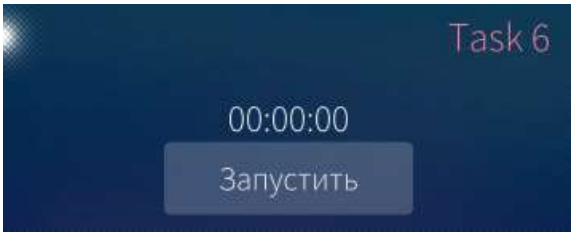

Рисунок 10 Таймер не запущен

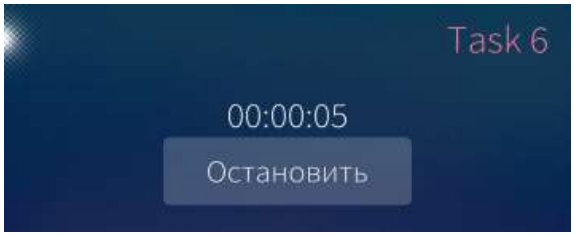

Рисунок 11 Таймер запущен

7. Подсчет числа добавленных в стек и удаленных из него будем вести при помощи введения 3-х переменных *property var added*: 0 – добавлено, *property var removed*: 0 – удалено, *property var removed*: 0 – глубина стека до очередного обновления его состояния. Используем элемент *Connections*, задаем атрибут *target: pageStack* для отслеживания стека страниц. По событию *onDepthChanged* выполняем обновление переменных по определенному алгоритму.

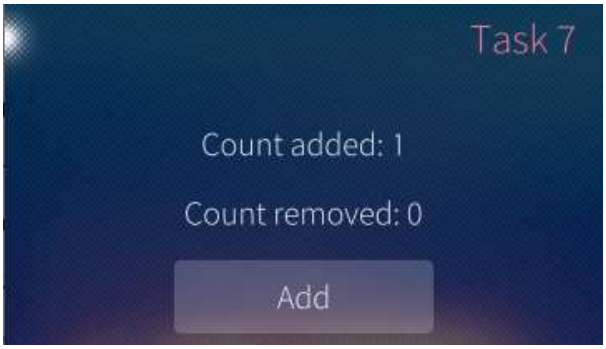

Рисунок 12 Счетчик стека страниц - открыта страница с демонстрацией задания

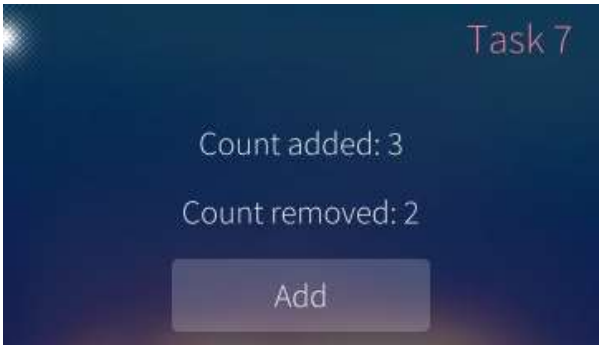

Рисунок 13 Счетчик стека страниц - несколько повторных добавлений и удалений демо-страницы.

4 Приложение

4.1 Файл FirstPage.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    id: page
    allowedOrientations: Orientation.All

    SilicaFlickable {
        anchors.fill: parent
        contentHeight: column.height

        Column {
            id: column
            width: page.width
            spacing: Theme.paddingLarge
            PageHeader {
                title: qsTr("Main Page")
            }
            Button {
                text: qsTr("Task 1")
                onClicked: pageStack.push(Qt.resolvedUrl("task01.qml"))
                anchors.horizontalCenter: parent.horizontalCenter
            }
            Button {
                text: qsTr("Task 2")
                onClicked: pageStack.push(Qt.resolvedUrl("task02.qml"))
                anchors.horizontalCenter: parent.horizontalCenter
            }
            Button {
                text: qsTr("Task 3")
                onClicked: pageStack.push(Qt.resolvedUrl("task03.qml"))
                anchors.horizontalCenter: parent.horizontalCenter
            }
            Button {
                text: qsTr("Task 4")
                onClicked: pageStack.push(Qt.resolvedUrl("task04.qml"))
                anchors.horizontalCenter: parent.horizontalCenter
            }
            Button {
                text: qsTr("Task 5")
                onClicked: pageStack.push(Qt.resolvedUrl("task05.qml"))
                anchors.horizontalCenter: parent.horizontalCenter
            }
            Button {
                text: qsTr("Task 6")
                onClicked: pageStack.push(Qt.resolvedUrl("task06.qml"))
                anchors.horizontalCenter: parent.horizontalCenter
            }
            Button {
                text: qsTr("Task 7")
                onClicked: pageStack.push(Qt.resolvedUrl("task07.qml"))
                anchors.horizontalCenter: parent.horizontalCenter
            }

        } // Column
    } // SilicaFlickable
} // Page
```

4.2 Файл task01.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    id: page

    Component.onCompleted: change_state()
    property int size: 240

    SilicaFlickable {
        anchors.fill: parent
        contentHeight: column.height

        Column {
            id: column
            width: page.width

            PageHeader {
                title: "Task 1"
            }

            Rectangle {
                id: traffic_light
                color: "darkslategray"
                width: size
                height: 3 * size + 60
                anchors.horizontalCenter: parent.horizontalCenter
                state: "red_lights"

                Column {
                    anchors.centerIn: parent
                    spacing: 15

                    Rectangle {
                        id: red_light
                        width: size
                        height: size
                        radius: size
                        color: "black"
                        anchors.horizontalCenter: parent.horizontalCenter
                    }

                    Rectangle {
                        id: yellow_light
                        width: size
                        height: size
                        radius: size
                        color: "black"
                        anchors.horizontalCenter: parent.horizontalCenter
                    }

                    Rectangle {
                        id: green_light
                        width: size
                        height: size
                        radius: size
                        color: "black"
                        anchors.horizontalCenter: parent.horizontalCenter
                    }
                } // Column - traffic light signals
```

```
                states: [
                    State {
                        name: "none"
                    },

                    State {
                        name: "red_lights"
                        PropertyChanges {
                            target: red_light
                            color: "red"
                        }
                    },

                    State {
                        name: "red_and_yellow_lights"
                        PropertyChanges {
                            target: red_light
                            color: "red"
                        }
                        PropertyChanges {
                            target: yellow_light
                            color: "yellow"
                        }
                    },

                    State {
                        name: "yellow_lights"
                        PropertyChanges {
                            target: yellow_light
                            color: "yellow"
                        }
                    },

                    State {
                        name: "green_lights"
                        PropertyChanges {
                            target: green_light
                            color: "green"
                        }
                    }
                ]

                transitions: [
                    Transition {
                        from: "red_lights"
                        to: "red_and_yellow_lights"
                        SequentialAnimation {
                            PauseAnimation {
                                duration: 3000
                            }
                            ColorAnimation {
                                targets: [red_light, yellow_light,
green_light]
                                property: "color"
                                duration: 0
                            }
                            ScriptAction {
                                script: change_state()
                            }
                        }
                    },

                    Transition {
```

```
                    from: "red_and_yellow_lights"
                    to: "green_lights"
                    SequentialAnimation {
                        PauseAnimation {
                            duration: 1000
                        }
                        ColorAnimation {
                            targets: [red_light, yellow_light,
green_light]
                            property: "color"
                            duration: 0
                        }
                        ScriptAction {
                            script: change_state()
                        }
                    }
                },

                Transition {
                    from: "green_lights"
                    to: "yellow_lights"

                    SequentialAnimation {
                        PauseAnimation {
                            duration: 3000
                        }
                        // Begin green light flashing animation
                        ColorAnimation {
                            target: green_light
                            property: "color"
                            to: "black"
                            duration: 0
                        }
                        PauseAnimation {
                            duration: 500
                        }
                        ColorAnimation {
                            target: green_light
                            property: "color"
                            to: "green"
                            duration: 0
                        }
                        PauseAnimation {
                            duration: 500
                        }
                        ColorAnimation {
                            target: green_light
                            property: "color"
                            to: "black"
                            duration: 0
                        }
                        PauseAnimation {
                            duration: 500
                        }
                        ColorAnimation {
                            target: green_light
                            property: "color"
                            to: "green"
                            duration: 0
                        }
                        PauseAnimation {
                            duration: 500
                        }
```

```
                            ColorAnimation {
                                target: green_light
                                property: "color"
                                to: "black"
                                duration: 0
                            }
                            PauseAnimation {
                                duration: 500
                            }
                            ColorAnimation {
                                target: green_light
                                property: "color"
                                to: "green"
                                duration: 0
                            }
                            PauseAnimation {
                                duration: 500
                            }
                            ColorAnimation {
                                target: green_light
                                property: "color"
                                to: "black"
                                duration: 0
                            }
                            // End green light flashing animation

                            ColorAnimation {
                                targets: [red_light, yellow_light,
green_light]
                                property: "color"
                                duration: 0
                            }
                            ScriptAction {
                                script: change_state()
                            }
                        }
                    },

                    Transition {
                        from: "yellow_lights"
                        to: "red_lights"

                        SequentialAnimation {
                            PauseAnimation {
                                duration: 1000
                            }
                            ColorAnimation {
                                targets: [red_light, yellow_light,
green_light]
                                property: "color"
                                duration: 0
                            }
                            ScriptAction {
                                script: change_state()
                            }
                        }
                    }
                ]
            } // Rectangle - id: traffic_light
        } // Column
    } // SilicaFlickable

    function change_state() {
```

```
        switch(traffic_light.state) {
        case "red_lights":
            traffic_light.state = "red_and_yellow_lights";
            break;
        case "red_and_yellow_lights":
            traffic_light.state = "green_lights";
            break;
        case "green_lights":
            traffic_light.state = "yellow_lights";
            break;
        case "yellow_lights":
            traffic_light.state = "red_lights";
            break;
        }
    }
}
```

4.3 Файл task02.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    id: page

    Component.onCompleted: change_state()
    property int size: 240

    SilicaFlickable {
        anchors.fill: parent
        contentHeight: column.height

        Column {
            id: column
            width: page.width

            PageHeader {
                title: "Task 2"
            }

            Rectangle {
                id: traffic_light
                color: "darkslategray"
                width: size
                height: 3 * size + 60
                anchors.horizontalCenter: parent.horizontalCenter
                state: "red_lights"

                Column {
                    anchors.centerIn: parent
                    spacing: 15

                    Rectangle {
                        id: red_light
                        width: size
                        height: size
                        radius: size
                        color: "black"
                        anchors.horizontalCenter: parent.horizontalCenter
                    }

                    Rectangle {
                        id: yellow_light
```

```
        width: size
        height: size
        radius: size
        color: "black"
        anchors.horizontalCenter: parent.horizontalCenter
    }

    Rectangle {
        id: green_light
        width: size
        height: size
        radius: size
        color: "black"
        anchors.horizontalCenter: parent.horizontalCenter
    }
} // Column - traffic light signals

states: [
    State {
        name: "none"
    },

    State {
        name: "red_lights"
        PropertyChanges {
            target: red_light
            color: "red"
        }
    },

    State {
        name: "red_and_yellow_lights"
        PropertyChanges {
            target: red_light
            color: "red"
        }
        PropertyChanges {
            target: yellow_light
            color: "yellow"
        }
    },

    State {
        name: "yellow_lights"
        PropertyChanges {
            target: yellow_light
            color: "yellow"
        }
    },

    State {
        name: "green_lights"
        PropertyChanges {
            target: green_light
            color: "green"
        }
    }
]

transitions: [
    Transition {
        from: "red_lights"
        to: "red_and_yellow_lights"
```

```
                SequentialAnimation {
                    PauseAnimation {
                        duration: 3000
                    }
                    ColorAnimation {
                        targets: [red_light, yellow_light,
green_light]
                        property: "color"
                        duration: 0
                    }
                    ScriptAction {
                        script: change_state()
                    }
                }
            },

            Transition {
                from: "red_and_yellow_lights"
                to: "green_lights"
                SequentialAnimation {
                    PauseAnimation {
                        duration: 1000
                    }
                    ColorAnimation {
                        targets: [red_light, yellow_light,
green_light]
                        property: "color"
                        duration: 0
                    }
                    ScriptAction {
                        script: change_state()
                    }
                }
            },

            Transition {
                from: "green_lights"
                to: "yellow_lights"

                SequentialAnimation {
                    PauseAnimation {
                        duration: 3000
                    }
                    // Begin green light flashing animation
                    ColorAnimation {
                        target: green_light
                        property: "color"
                        to: "black"
                        duration: 0
                    }
                    PauseAnimation {
                        duration: 500
                    }
                    ColorAnimation {
                        target: green_light
                        property: "color"
                        to: "green"
                        duration: 0
                    }
                    PauseAnimation {
                        duration: 500
                    }
                    ColorAnimation {
```

```
                        target: green_light
                        property: "color"
                        to: "black"
                        duration: 0
                    }
                    PauseAnimation {
                        duration: 500
                    }
                    ColorAnimation {
                        target: green_light
                        property: "color"
                        to: "green"
                        duration: 0
                    }
                    PauseAnimation {
                        duration: 500
                    }
                    ColorAnimation {
                        target: green_light
                        property: "color"
                        to: "black"
                        duration: 0
                    }
                    PauseAnimation {
                        duration: 500
                    }
                    ColorAnimation {
                        target: green_light
                        property: "color"
                        to: "green"
                        duration: 0
                    }
                    PauseAnimation {
                        duration: 500
                    }
                    ColorAnimation {
                        target: green_light
                        property: "color"
                        to: "black"
                        duration: 0
                    }
                    // End green light flashing animation

                    ColorAnimation {
                        targets: [red_light, yellow_light,
green_light]
                        property: "color"
                        duration: 0
                    }
                    ScriptAction {
                        script: change_state()
                    }
                }
            },

            Transition {
                from: "yellow_lights"
                to: "red_lights"

                SequentialAnimation {
                    PauseAnimation {
                        duration: 1000
                    }
```

```
                            ColorAnimation {
                                targets: [red_light, yellow_light,
green_light]
                                property: "color"
                                duration: 0
                            }
                            ScriptAction {
                                script: change_state()
                            }
                        }
                    }
                ]
            } // Rectangle - id: traffic_light

            Image {
                id: image_person
                source: "image://theme/icon-m-person"
                state: "left_side"

                states: [
                    State {
                        name: "left_side"
                        PropertyChanges {
                            target: image_person
                            x: 10
                        }
                    },

                    State {
                        name: "right_side"
                        PropertyChanges {
                            target: image_person
                            x: parent.width - 100
                        }
                    }
                ]

                transitions: [
                    Transition {
                        NumberAnimation {
                            property: "x"
                            duration: 3000
                        }
                    }
                ]
            } // Image - image_person

        } // Column
    } // SilicaFlickable

    function change_state() {
        switch(traffic_light.state) {
        case "red_lights":
            traffic_light.state = "red_and_yellow_lights";
            break;
        case "red_and_yellow_lights":
            traffic_light.state = "green_lights";
            break;
        case "green_lights":
            image_person.state = (image_person.state == "left_side") ?
"right_side" : "left_side";
            traffic_light.state = "yellow_lights";
            break;
```

```
        case "yellow_lights":
            traffic_light.state = "red_lights";
            break;
        }
    }
}
```

4.4 Файл task03.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    id: page

    SilicaFlickable {
        anchors.fill: parent
        contentHeight: column.height

        Column {
            id: column
            width: page.width

            PageHeader {
                title: "Task 3"
            }

            Label {
                id: label
                text: "Tap and hold!"
                color: "black"
                font.pixelSize: 64
                anchors.horizontalCenter: parent.horizontalCenter
                state: "normal"

                states: [
                    State {
                        name: "normal"

                        PropertyChanges {
                            target: label
                            y: 100
                        }
                        PropertyChanges {
                            target: label
                            color: "white"
                        }
                        PropertyChanges {
                            target: label
                            rotation: 0
                        }
                    },

                    State {
                        name: "inverted"

                        PropertyChanges {
                            target: label
                            y: 500
                        }
                        PropertyChanges {
                            target: label
                            color: "purple"
```

```
                        }
                        PropertyChanges {
                            target: label
                            rotation: 180
                        }
                    }
                ]

                transitions: [
                    Transition {
                        NumberAnimation {
                            property: "y"
                            duration: 1500
                        }
                        RotationAnimation {
                            direction: RotationAnimation.Counterclockwise
                            duration: 1500
                        }
                        ColorAnimation {
                            duration: 1500
                        }
                    }
                ]

                MouseArea {
                    anchors.fill: parent
                    onPressed: label.state = (label.state == "normal") ?
"inverted" : "normal"
                    onReleased: label.state = (label.state == "normal") ?
"inverted" : "normal"
                }

            }
        }
    }
}
```

4.5 Файл task04.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    id: page

    Component.onCompleted: traffic_light.change_state()


    Column {
        id: column
        width: page.width

        PageHeader {
            title: "Task 4"
        }

        TrafficLight {
            id: traffic_light
        }
    }
}
```

4.5.1 Файл TrafficLight.qml

```
import QtQuick 2.0

Rectangle {
    id: traffic_light

    color: "darkslategray"
    width: size
    height: 3 * size + 60
    anchors.horizontalCenter: parent.horizontalCenter
    state: "red_lights"
    property var size: 240

    function change_state() {
        switch(traffic_light.state) {
        case "red_lights":
            traffic_light.state = "red_and_yellow_lights";
            break;
        case "red_and_yellow_lights":
            traffic_light.state = "green_lights";
            break;
        case "green_lights":
            traffic_light.state = "yellow_lights";
            break;
        case "yellow_lights":
            traffic_light.state = "red_lights";
            break;
        }
    }

    Column {
        anchors.centerIn: parent
        spacing: 15

        Rectangle {
            id: red_light
            width: size
            height: size
            radius: size
            color: "black"
            anchors.horizontalCenter: parent.horizontalCenter
        }

        Rectangle {
            id: yellow_light
            width: size
            height: size
            radius: size
            color: "black"
            anchors.horizontalCenter: parent.horizontalCenter
        }

        Rectangle {
            id: green_light
            width: size
            height: size
            radius: size
            color: "black"
            anchors.horizontalCenter: parent.horizontalCenter
        }
    } // Column - traffic light signals
```

```
states: [
    State {
        name: "none"
    },

    State {
        name: "red_lights"
        PropertyChanges {
            target: red_light
            color: "red"
        }
    },

    State {
        name: "red_and_yellow_lights"
        PropertyChanges {
            target: red_light
            color: "red"
        }
        PropertyChanges {
            target: yellow_light
            color: "yellow"
        }
    },

    State {
        name: "yellow_lights"
        PropertyChanges {
            target: yellow_light
            color: "yellow"
        }
    },

    State {
        name: "green_lights"
        PropertyChanges {
            target: green_light
            color: "green"
        }
    }
]

transitions: [
    Transition {
        from: "red_lights"
        to: "red_and_yellow_lights"
        SequentialAnimation {
            PauseAnimation {
                duration: 3000
            }
            ColorAnimation {
                targets: [red_light, yellow_light, green_light]
                property: "color"
                duration: 0
            }
            ScriptAction {
                script: change_state()
            }
        }
    },

    Transition {
        from: "red_and_yellow_lights"
```

```
        to: "green_lights"
        SequentialAnimation {
            PauseAnimation {
                duration: 1000
            }
            ColorAnimation {
                targets: [red_light, yellow_light, green_light]
                property: "color"
                duration: 0
            }
            ScriptAction {
                script: change_state()
            }
        }
    },

    Transition {
        from: "green_lights"
        to: "yellow_lights"

        SequentialAnimation {
            PauseAnimation {
                duration: 3000
            }
            // Begin green light flashing animation
            ColorAnimation {
                target: green_light
                property: "color"
                to: "black"
                duration: 0
            }
            PauseAnimation {
                duration: 500
            }
            ColorAnimation {
                target: green_light
                property: "color"
                to: "green"
                duration: 0
            }
            PauseAnimation {
                duration: 500
            }
            ColorAnimation {
                target: green_light
                property: "color"
                to: "black"
                duration: 0
            }
            PauseAnimation {
                duration: 500
            }
            ColorAnimation {
                target: green_light
                property: "color"
                to: "green"
                duration: 0
            }
            PauseAnimation {
                duration: 500
            }
            ColorAnimation {
                target: green_light
```

```
                    property: "color"
                    to: "black"
                    duration: 0
                }
                PauseAnimation {
                    duration: 500
                }
                ColorAnimation {
                    target: green_light
                    property: "color"
                    to: "green"
                    duration: 0
                }
                PauseAnimation {
                    duration: 500
                }
                ColorAnimation {
                    target: green_light
                    property: "color"
                    to: "black"
                    duration: 0
                }
                // End green light flashing animation

                ColorAnimation {
                    targets: [red_light, yellow_light, green_light]
                    property: "color"
                    duration: 0
                }
                ScriptAction {
                    script: change_state()
                }
            }
        },

        Transition {
            from: "yellow_lights"
            to: "red_lights"

            SequentialAnimation {
                PauseAnimation {
                    duration: 1000
                }
                ColorAnimation {
                    targets: [red_light, yellow_light, green_light]
                    property: "color"
                    duration: 0
                }
                ScriptAction {
                    script: change_state()
                }
            }
        }
    ]
} // Rectangle - id: traffic_light
```

4.6 Файл task05.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0
```

```
Page {
    id: page

    Column {
        id: column
        width: page.width

        PageHeader {
            title: "Task 5"
        }

        CustomButton {
            anchors.horizontalCenter: parent.horizontalCenter
            color_button: "yellow"

            Text { text: "Custom Button" }
        }

    }
}
```

4.7 Файл task06.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    id: page

    SilicaFlickable {
        anchors.fill: parent

        Column {
            id: column
            width: page.width

            PageHeader {
                title: "Task 6"
            }

            Timer {
                id: timer
                property var ticks: 0
                interval: 1000
                repeat: true

                onTriggered: time.text = dispaly_time(++ticks);
            }

            Text {
                id: time
                color: "white"
                text: dispaly_time(timer.ticks)
                font.pixelSize: 46
                anchors.horizontalCenter: parent.horizontalCenter
            }

            Button {
                id: button
                anchors.horizontalCenter: parent.horizontalCenter
                text: timer.running ? "Остановить" : "Запустить"

                onClicked: {
```

```
                    timer.running ? timer.stop() : timer.start();
                }
            }
        }
    }

    function dispaly_time(ticks) {
        var hh = Math.floor(ticks / 3600);
        var mm = Math.floor((ticks % 3600) / 60);
        var ss = ticks % 60

        return pad(hh, 2) + ":" + pad(mm, 2) + ":" + pad(ss, 2);
    }

    function pad(n, width) {
        n = n + "";
        return n.length >= width ? n : new Array(width - n.length +
1).join("0") + n;
    }
}
```

4.8 Файл task07.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    id: page

    property var added: 0
    property var removed: 0
    property var prev: 0

    Connections {
        target: pageStack
        onDepthChanged: {
            if (page.prev > pageStack.depth) {
                page.removed++
            }
            else {
                page.added++
            }
            page.prev = pageStack.depth
        }
    }

    Column {
        id: column

        width: page.width
        spacing: Theme.paddingLarge
        PageHeader {
            title: "Task 7"
        }
        Label {
            text: "Count added: " + added
            anchors.horizontalCenter: parent.horizontalCenter
        }
        Label {
            text: "Count removed: " + removed
            anchors.horizontalCenter: parent.horizontalCenter
        }
```

```
        Button {
            text: "Add"
            anchors.horizontalCenter: parent.horizontalCenter
            onClicked: pageStack.push(Qt.resolvedUrl("task07-demo.qml"))
        }
    }
}
```

4.8.1 Файл task07-demo.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    id: page

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

        PageHeader {
            title: "Task 7 \nAddon"
        }
        Label {
            anchors.horizontalCenter: parent.horizontalCenter
            text: "Demo-page in pageStack "
        }
        Button {
            text: "Remove this page from pageStack"
            anchors.horizontalCenter: parent.horizontalCenter
            onClicked: pageStack.pop()
        }
    }
}
```

5 Используемая литература

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