Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования
Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

Отчет по практическому заданию №7

«Инструменты разработки мобильных приложений»

Выполнил:
студент группы 381908-4
Рябцев М. В.

Нижний Новгород
2022

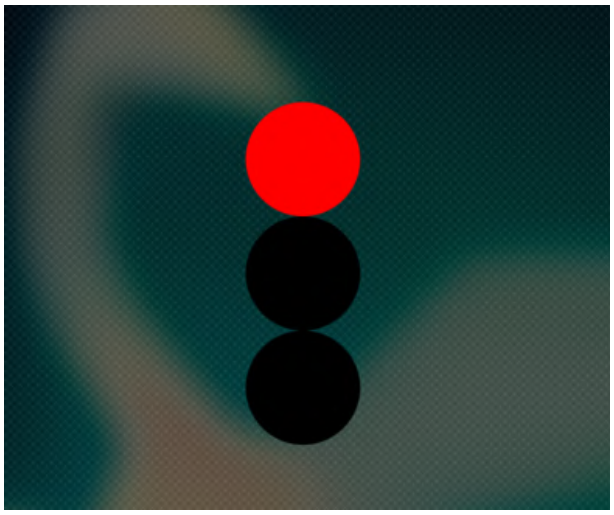
Оглавление

Цель задачи

В данном лабораторной работе требуется научиться создавать пользовательский интерфейс, конфигурируемый состояниями, реализовывать анимированные переходы при смене состояний и создавать собственные QML компоненты.

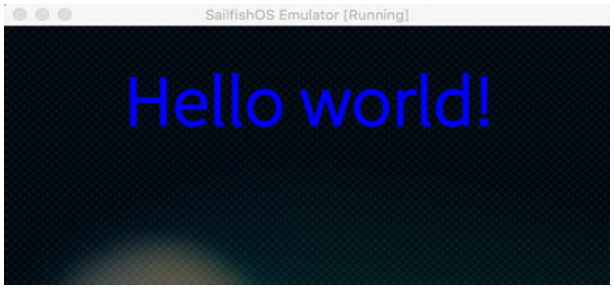
Постановка задачи

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.

Описание программной реализации

1. Светофор:

```
Column {
    anchors.centerIn: parent
    spacing: circleWidth / 10

    property int circleWidth: 100
    property int delayCnt: 0

    Rectangle {
        id: redCircle
        width: parent.circleWidth
        height: width
        color: "red"
        border.color: "grey"
        border.width: 2
        radius: width*0.5
        opacity: 1
    }

    Rectangle {
        id: yellowCircle
        width: parent.circleWidth
        height: width
        color: "yellow"
        border.color: "grey"
        border.width: 2
        radius: width*0.5
        opacity: 0.3
    }

    Rectangle {
        id: greenCircle
        width: parent.circleWidth
        height: width
        color: "green"
        border.color: "grey"
        border.width: 2
        radius: width*0.5
        opacity: 0.3
    }

    Timer {
        interval: 100; running: true; repeat: true
        onTriggered: parent.delayCnt = (parent.delayCnt + 1) % 40
    }

    state: {
```

```
            if (delayCnt < 10){
                "red"
            } else if (delayCnt < 20) {
                "yellow_red"
            } else if (delayCnt < 30) {
                "green"
            } else {
                "yellow_green"
            }
        }

        states: [
            State {
                name: "red"
                PropertyChanges { target: redCircle; opacity: 1}
                PropertyChanges { target: yellowCircle; opacity: 0.3}
                PropertyChanges { target: greenCircle; opacity: 0.3}
            },
            State {
                name: "yellow_red"
                PropertyChanges { target: redCircle; opacity: 1}
                PropertyChanges { target: yellowCircle; opacity: 1}
                PropertyChanges { target: greenCircle; opacity: 0.3}
            },
            State {
                name: "green"
                PropertyChanges { target: redCircle; opacity: 0.3}
                PropertyChanges { target: yellowCircle; opacity: 0.3}
                PropertyChanges { target: greenCircle; opacity: 1}
            },
            State {
                name: "yellow_green"
                PropertyChanges { target: redCircle; opacity: 0.3}
                PropertyChanges { target: yellowCircle; opacity: 1}
                PropertyChanges { target: greenCircle; opacity: 1}
            }
        ]
    }
```

2. Светофор с анимацией:

```
    IconButton {
        id: person
        icon.source: "image://theme/icon-m-media-artists"
        onClicked: console.log("Play clicked!")
        opacity: 0
        x: parent.width * (-1) - 500

        PropertyAnimation {
            id: animation_forward
            target: person;
```

```
            property: "x";
            from: parent.width * (-1) - 50;
            to: parent.width + 50;
            duration: 1000;
        }
    }

    states: [
        State {
            name: "red"
            PropertyChanges { target: redCircle; opacity: 1}
            PropertyChanges { target: yellowCircle; opacity: 0.3}
            PropertyChanges { target: greenCircle; opacity: 0.3}
        },
        State {
            name: "yellow_red"
            PropertyChanges { target: redCircle; opacity: 1}
            PropertyChanges { target: yellowCircle; opacity: 1}
            PropertyChanges { target: greenCircle; opacity: 0.3}
        },
        State {
            name: "green"
            PropertyChanges { target: redCircle; opacity: 0.3}
            PropertyChanges { target: yellowCircle; opacity: 0.3}
            PropertyChanges { target: greenCircle; opacity: 1}
            PropertyChanges { target: person; opacity: 1}
            StateChangeScript {
                script: animation_forward.running = true;
            }
        },
        State {
            name: "yellow_green"
            PropertyChanges { target: redCircle; opacity: 0.3}
            PropertyChanges { target: yellowCircle; opacity: 1}
            PropertyChanges { target: greenCircle; opacity: 1}
        }
    ]
```

3. Анимация текста

```
    Label {
        id: tgt
        anchors.horizontalCenter: parent.horizontalCenter
        y: 100
        text: "Hello"
        color: "blue"
        font.pixelSize: 200
        horizontalAlignment: Text.AlignHCenter
    }

    state: {
```

```
        if (mouseArea.pressedButtons){
            "way"
        } else {
            "back"
        }
    }

    states: [
        State {
            name: "way"
        },
        State {
            name: "back"
        }
    ]

    transitions: [
        Transition {
            from: "back"
            to: "way"
            ParallelAnimation {
                PropertyAnimation { target: tgt; properties: "y"; from: tgt.y; to
                PropertyAnimation { target: tgt; properties: "color"; from: tgt.c
                RotationAnimation { target: tgt; from: 0; to: 180; duration: 1000
            }
        },
        Transition {
            from: "way"
            to: "back"
            PropertyAnimation { target: tgt; properties: "y"; from: tgt.y; to: 10
            PropertyAnimation { target: tgt; properties: "color"; from: tgt.color
            RotationAnimation { target: tgt; from: tgt.rotation; to: 0; duration:

        }
    ]

    MouseArea {
        id: mouseArea
        anchors.fill: parent

        onPressed: {
            console.log("123")
            console.log(mouseArea.pressedButtons)
        }
        onReleased: {
            console.log(mouseArea.pressedButtons)
        }
    }
```

4. Светофор с отдельными компонентами

```
// TrafficLight.qml
Rectangle {
    id: circle
    width: 100
    height: 100
    color: parent.color
    border.color: "grey"
    border.width: 2
    radius: width*0.5
    opacity: 1

    function setOpacity(op) {
        circle.opacity = op
    }
}

TrafficLight {
    id: redCircle
    color: "red"
}
TrafficLight {
    id: yellowCircle
    color: "yellow"
}
TrafficLight {
    id: greenCircle
    color: "green"
}
```

5. Собственный компонент QML Button

```
// MyButton.qml
Rectangle {
    height: 200;
    width: 500;
    radius: 20;
    color: setColor
    property string setColor: "red"
    property string btnText: "text"
    Button {
        height: parent.height;
        width: parent.width;
        anchors.horizontalCenter: parent.horizontalCenter;
        anchors.verticalCenter: parent.verticalCenter;
        text: btnText
    }

    Component.onCompleted: {
        for (var i = 0; i < this.children.length; i++) {
            console.log(this.children[i])
```

```
                if (this.children[i].text) {
                    if (this.children[i].text !== btnText) {
                        btnText = this.children[i].text
                    }
                }
            }
        }
    }

    MyButton {
        id: myButton;
        anchors.centerIn: parent.Center
        Label {
            anchors.top: parent.top
            anchors.horizontalCenter: parent.horizontalCenter
            text: "Тексты совпадают"
        }
        setColor: "red"
    }
```

6. Приложение-секундомер

```
    //MyCounter.qml
    Rectangle {
        width: 200
        height: 150

        border.color: "grey"
        border.width: 10
        radius: 10

        property string num: "0"

        Label {
            text: num
            anchors.centerIn: parent
        }
    }

    Column {
        anchors.centerIn: parent
        Row {
            anchors.centerIn: parent.Center
            spacing: 5

            id: row
            property int count: 0

            MyCounter {
                num: parseInt(row.count / 60 / 60)
            }
```

```
            MyCounter {
                num: parseInt(row.count / 60)
            }
            MyCounter {
                num: row.count % 60
            }
        }

        Button {
            anchors.horizontalCenter: parent.horizontalCenter
            width: 200
            height: 100
            text: "Старт"
            onClicked: {
                timer.running = !timer.running
                console.log(text)
                text = text === "Старт" ? "Пауза" : "Старт"
            }
        }
    }

    Timer {
        id: timer
        interval: 1000
        repeat: true
        running: false
        onTriggered: {
            row.count++
        }
    }
```

Руководство пользователя

После запуска программы пользователь видит страницу с первым заданием и кнопку “Next”, которая перейдет на следующую страницу. В первом задании мы видим анимированный светофор.

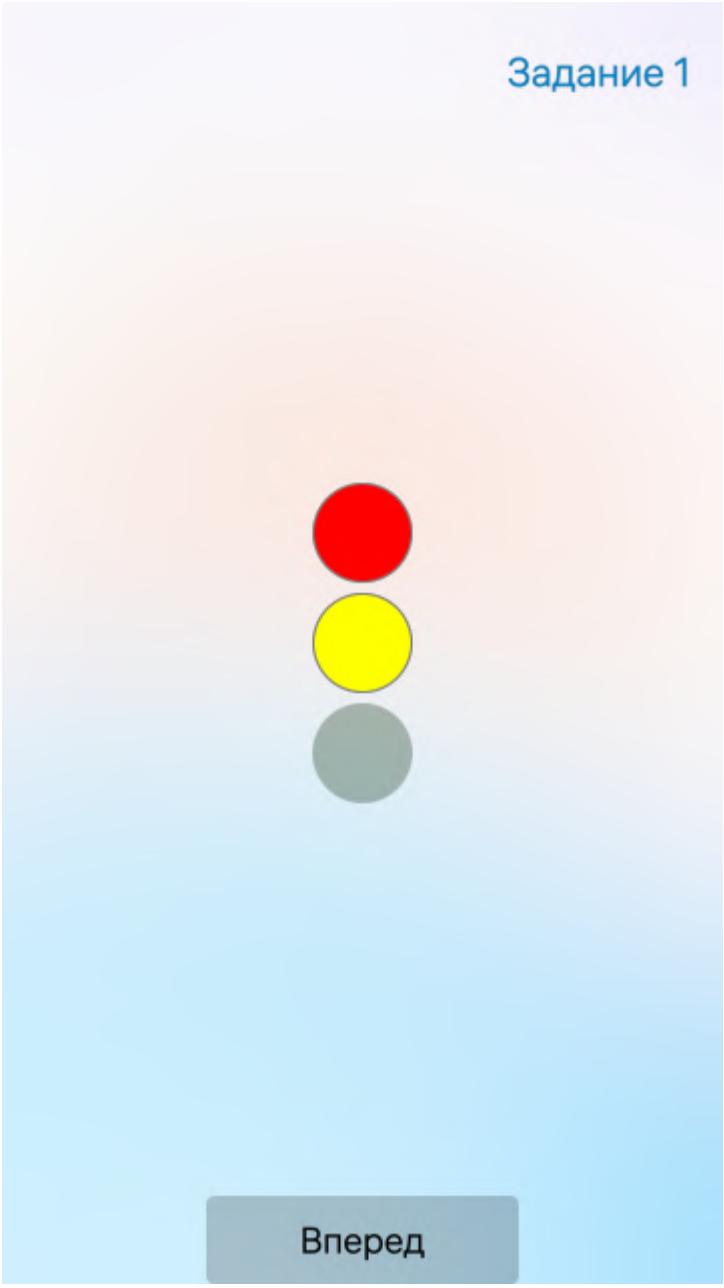


Рис. 1: Страница с первым заданием

Во втором задании на зеленый свет появляется анимация человечка.

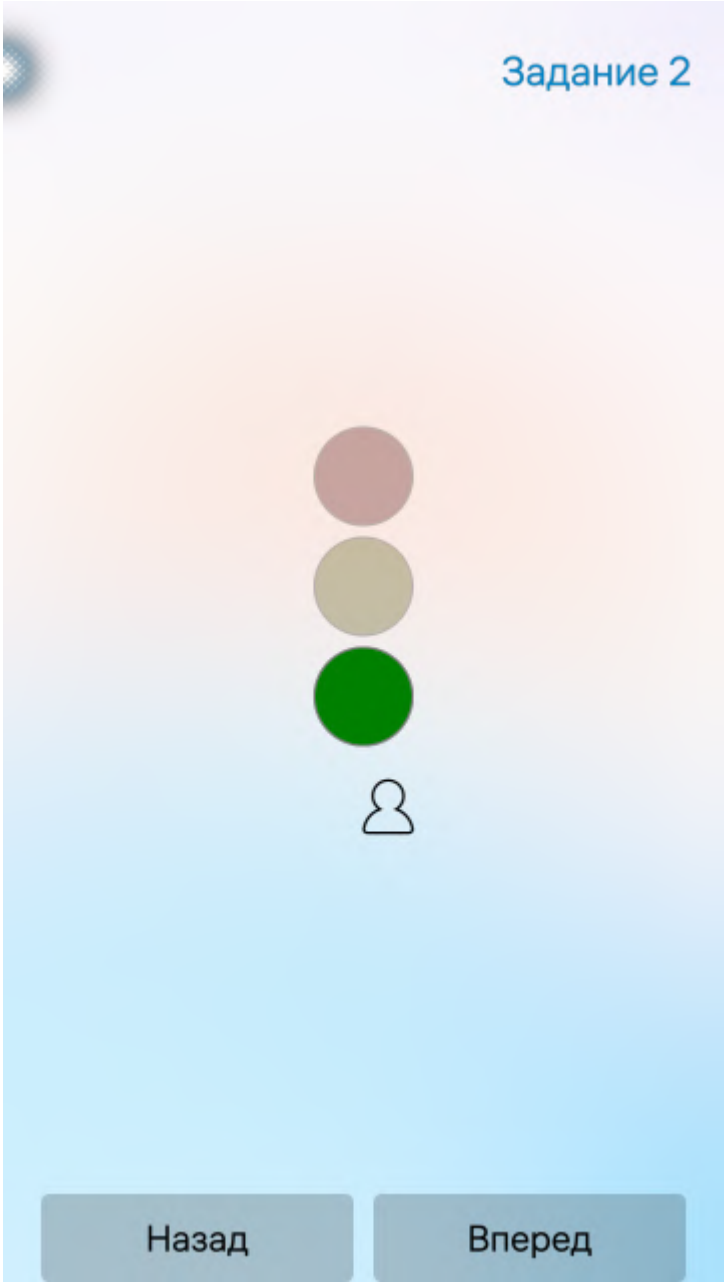


Рис. 2: Страница со вторым заданием

В третьем задании при нажатии на текст он анимируется.

Рис. 3: Анимированный текст

В пятом задании текст кнопки и надписи совпадают.

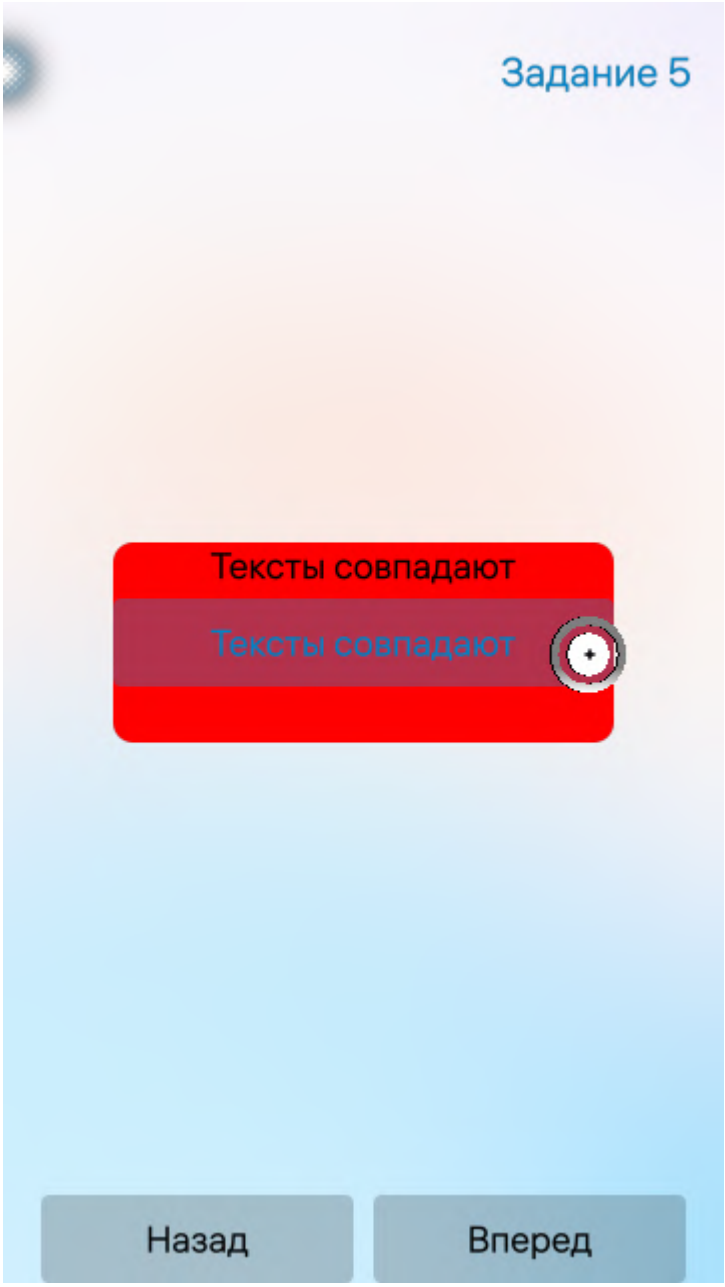


Рис. 4: Пользовательская кнопка

В шестом задании представлен работающий секундомер.

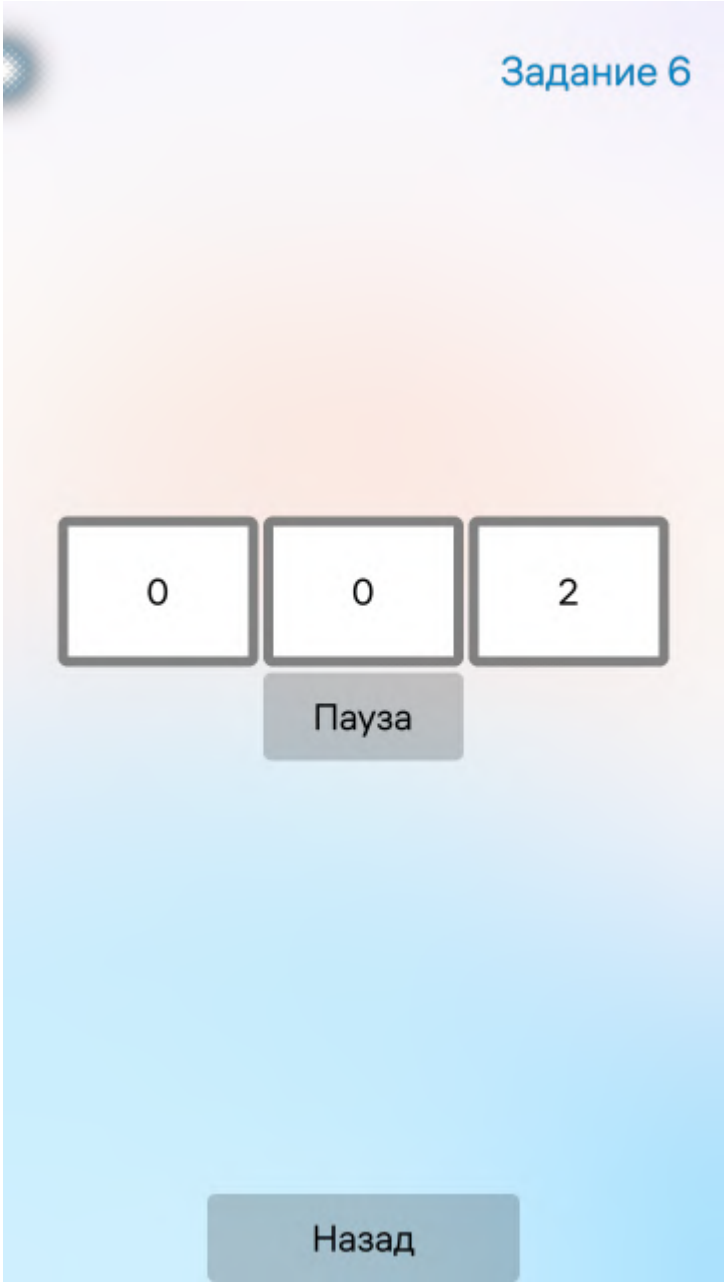


Рис. 5: Секундомер

Заключение

В процессе выполнения данной лабораторной работы мы научились создавать пользовательский интерфейс, конфигурируемый состояниями, реализовывать анимированные переходы при смене состояний и создавать собственные QML компоненты.

Приложение

Task1.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    objectName: "mainPage"
    allowedOrientations: Orientation.All

    PageHeader {
        objectName: "pageHeader"
        title: qsTr("Задание 1")
    }

    Column {
        anchors.centerIn: parent
        spacing: circleWidth / 10

        property int circleWidth: 100
        property int delayCnt: 0

        Rectangle {
            id: redCircle
            width: parent.circleWidth
            height: width
            color: "red"
            border.color: "grey"
            border.width: 2
            radius: width*0.5
            opacity: 1
        }

        Rectangle {
            id: yellowCircle
            width: parent.circleWidth
            height: width
            color: "yellow"
            border.color: "grey"
            border.width: 2
            radius: width*0.5
            opacity: 0.3
        }

        Rectangle {
            id: greenCircle
            width: parent.circleWidth
```

```
    height: width
    color: "green"
    border.color: "grey"
    border.width: 2
    radius: width*0.5
    opacity: 0.3
}

Timer {
    interval: 100; running: true; repeat: true
    onTriggered: parent.delayCnt = (parent.delayCnt + 1) % 40
}

state: {
    if (delayCnt < 10){
        "red"
    } else if (delayCnt < 20) {
        "yellow_red"
    } else if (delayCnt < 30) {
        "green"
    } else {
        "yellow_green"
    }
}

states: [
    State {
        name: "red"
        PropertyChanges { target: redCircle; opacity: 1}
        PropertyChanges { target: yellowCircle; opacity: 0.3}
        PropertyChanges { target: greenCircle; opacity: 0.3}
    },
    State {
        name: "yellow_red"
        PropertyChanges { target: redCircle; opacity: 1}
        PropertyChanges { target: yellowCircle; opacity: 1}
        PropertyChanges { target: greenCircle; opacity: 0.3}
    },
    State {
        name: "green"
        PropertyChanges { target: redCircle; opacity: 0.3}
        PropertyChanges { target: yellowCircle; opacity: 0.3}
        PropertyChanges { target: greenCircle; opacity: 1}
    },
    State {
        name: "yellow_green"
        PropertyChanges { target: redCircle; opacity: 0.3}
        PropertyChanges { target: yellowCircle; opacity: 1}
        PropertyChanges { target: greenCircle; opacity: 1}
    }
```

```
        ]
    }

    Row {
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.bottom: parent.bottom
        spacing: 20

        Button {
            text: "Вперед"
            onClicked: pageStack.push(Qt.resolvedUrl(qsTr("Task2.qml")))
        }
    }
}
```

Task2.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    objectName: "mainPage"
    allowedOrientations: Orientation.All

    PageHeader {
        objectName: "pageHeader"
        title: qsTr("Задание 2")
    }

    Column {
        anchors.centerIn: parent
        spacing: circleWidth / 10

        property int circleWidth: 100
        property int delayCnt: 0

        Rectangle {
            id: redCircle
            width: parent.circleWidth
            height: width
            color: "red"
            border.color: "grey"
            border.width: 2
            radius: width*0.5
            opacity: 1
        }

        Rectangle {
```

```
    id: yellowCircle
    width: parent.circleWidth
    height: width
    color: "yellow"
    border.color: "grey"
    border.width: 2
    radius: width*0.5
    opacity: 0.3
}

Rectangle {
    id: greenCircle
    width: parent.circleWidth
    height: width
    color: "green"
    border.color: "grey"
    border.width: 2
    radius: width*0.5
    opacity: 0.3
}

IconButton {
    id: person
    icon.source: "image://theme/icon-m-media-artists"
    onClicked: console.log("Play clicked!")
    opacity: 0
    x: parent.width * (-1) - 500

    PropertyAnimation {
        id: animation_forward
        target: person;
        property: "x";
        from: parent.width * (-1) - 50;
        to: parent.width + 50;
        duration: 1000;
    }
}

Timer {
    interval: 100; running: true; repeat: true
    onTriggered: parent.delayCnt = (parent.delayCnt + 1) % 40
}

state: {
    if (delayCnt < 10){
        "red"
    } else if (delayCnt < 20) {
        "yellow_red"
    } else if (delayCnt < 30) {
        "green"
```

```
        } else {
            "yellow_green"
        }
    }

    states: [
        State {
            name: "red"
            PropertyChanges { target: redCircle; opacity: 1}
            PropertyChanges { target: yellowCircle; opacity: 0.3}
            PropertyChanges { target: greenCircle; opacity: 0.3}
        },
        State {
            name: "yellow_red"
            PropertyChanges { target: redCircle; opacity: 1}
            PropertyChanges { target: yellowCircle; opacity: 1}
            PropertyChanges { target: greenCircle; opacity: 0.3}
        },
        State {
            name: "green"
            PropertyChanges { target: redCircle; opacity: 0.3}
            PropertyChanges { target: yellowCircle; opacity: 0.3}
            PropertyChanges { target: greenCircle; opacity: 1}
            PropertyChanges { target: person; opacity: 1}
            StateChangeScript {
                script: animation_forward.running = true;
            }
        },
        State {
            name: "yellow_green"
            PropertyChanges { target: redCircle; opacity: 0.3}
            PropertyChanges { target: yellowCircle; opacity: 1}
            PropertyChanges { target: greenCircle; opacity: 1}
        }
    ]
}

Row {
    anchors.horizontalCenter: parent.horizontalCenter
    anchors.bottom: parent.bottom
    spacing: 20

    Button {
        text: "Назад"
        onClicked: pageStack.pop()
    }
    Button {
        text: "Вперед"
        onClicked: pageStack.push(Qt.resolvedUrl(qsTr("Task3.qml")))
    }
```

```
    }
}
```

Task3.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    objectName: "mainPage"
    allowedOrientations: Orientation.All

    PageHeader {
        objectName: "pageHeader"
        title: qsTr("Задание 3")
    }

    Label {
        id: tgt
        anchors.horizontalCenter: parent.horizontalCenter
        y: 100
        text: "Hello"
        color: "blue"
        font.pixelSize: 200
        horizontalAlignment: Text.AlignHCenter
    }

    state: {
        if (mouseArea.pressedButtons){
            "way"
        } else {
            "back"
        }
    }

    states: [
        State {
            name: "way"
        },
        State {
            name: "back"
        }
    ]

    transitions: [
        Transition {
            from: "back"
            to: "way"
```

```
            ParallelAnimation {
                PropertyAnimation { target: tgt; properties: "y"; from: tgt.y; to: 800
                PropertyAnimation { target: tgt; properties: "color"; from: tgt.color
                RotationAnimation { target: tgt; from: 0; to: 180; duration: 1000;}
            }
        },
        Transition {
            from: "way"
            to: "back"
            PropertyAnimation { target: tgt; properties: "y"; from: tgt.y; to: 100; du
            PropertyAnimation { target: tgt; properties: "color"; from: tgt.color; to
            RotationAnimation { target: tgt; from: tgt.rotation; to: 0; duration: 100

        }
    ]

    MouseArea {
        id: mouseArea
        anchors.fill: parent

        onPressed: {
            console.log("123")
            console.log(mouseArea.pressedButtons)
        }
        onReleased: {
            console.log(mouseArea.pressedButtons)
        }
    }


    Row {
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.bottom: parent.bottom
        spacing: 20

        Button {
            text: "Назад"
            onClicked: pageStack.pop()
        }
        Button {
            text: "Вперед"
            onClicked: pageStack.push(Qt.resolvedUrl(qsTr("Task4.qml")))
        }
    }
}
```

Task4.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    objectName: "mainPage"
    allowedOrientations: Orientation.All

    PageHeader {
        objectName: "pageHeader"
        title: qsTr("Задание 4")
    }

    Column {
        anchors.centerIn: parent
        spacing: circleWidth / 10

        property int circleWidth: 100
        property int delayCnt: 0

        TrafficLight {
            id: redCircle
            color: "red"
        }
        TrafficLight {
            id: yellowCircle
            color: "yellow"
        }
        TrafficLight {
            id: greenCircle
            color: "green"
        }

        Timer {
            interval: 100; running: true; repeat: true
            onTriggered: parent.delayCnt = (parent.delayCnt + 1) % 40
        }

        state: {
            if (delayCnt < 10){
                "red"
            } else if (delayCnt < 20) {
                "yellow_red"
            } else if (delayCnt < 30) {
                "green"
            } else {
                "yellow_green"
            }
        }
```

```
    states: [
        State {
            name: "red"
            StateChangeScript {
                script: {
                    redCircle.opacity = 1
                    yellowCircle.opacity = 0.3
                    greenCircle.opacity = 0.3
                }
            }
        },
        State {
            name: "yellow_red"
            StateChangeScript {
                script: {
                    redCircle.opacity = 1
                    yellowCircle.opacity = 1
                    greenCircle.opacity = 0.3
                }
            }
        },
        State {
            name: "green"
            StateChangeScript {
                script: {
                    redCircle.opacity = 0.3
                    yellowCircle.opacity = 0.3
                    greenCircle.opacity = 1
                }
            }
        },
        State {
            name: "yellow_green"
            StateChangeScript {
                script: {
                    redCircle.opacity = 0.3
                    yellowCircle.opacity = 1
                    greenCircle.opacity = 1
                }
            }
        }
    ]
}

Row {
    anchors.horizontalCenter: parent.horizontalCenter
    anchors.bottom: parent.bottom
    spacing: 20
```

```
        Button {
            text: "Назад"
            onClicked: pageStack.pop()
        }
        Button {
            text: "Вперед"
            onClicked: pageStack.push(Qt.resolvedUrl(qsTr("Task5.qml")))
        }
    }
}
```

TrafficLight.qml

```
import QtQuick 2.0

Rectangle {
    id: circle
    width: 100
    height: 100
    color: parent.color
    border.color: "grey"
    border.width: 2
    radius: width*0.5
    opacity: 1

    function setOpacity(op) {
        circle.opacity = op
    }
}
```

Task5.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    objectName: "mainPage"
    allowedOrientations: Orientation.All

    PageHeader {
        objectName: "pageHeader"
        title: qsTr("Задание 5")
    }

    Column {
        anchors.centerIn: parent
```

```
        MyButton {
            id: myButton;
            anchors.centerIn: parent.Center
            Label {
                anchors.top: parent.top
                anchors.horizontalCenter: parent.horizontalCenter
                text: "Тексты совпадают"
            }
            setColor: "red"
        }

    }

    Row {
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.bottom: parent.bottom
        spacing: 20

        Button {
            text: "Назад"
            onClicked: pageStack.pop()
        }
        Button {
            text: "Вперед"
            onClicked: pageStack.push(Qt.resolvedUrl(qsTr("Task6.qml")))
        }
    }
}
```

MyButton.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Rectangle {
    height: 200;
    width: 500;
    radius: 20;
    color: setColor
    property string setColor: "red"
    property string btnText: "text"
    Button {
        height: parent.height;
        width: parent.width;
        anchors.horizontalCenter: parent.horizontalCenter;
        anchors.verticalCenter: parent.verticalCenter;
        text: btnText
```

```
    }

    Component.onCompleted: {
        for (var i = 0; i < this.children.length; i++) {
            console.log(this.children[i])

            if (this.children[i].text) {
                if (this.children[i].text !== btnText) {
                    btnText = this.children[i].text
                }
            }
        }
    }
}
```

Task6.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    objectName: "mainPage"
    allowedOrientations: Orientation.All

    PageHeader {
        objectName: "pageHeader"
        title: qsTr("Задание 6")
    }

    Column {
        anchors.centerIn: parent
        Row {
            anchors.centerIn: parent.Center
            spacing: 5

            id: row
            property int count: 0

            MyCounter {
                num: parseInt(row.count / 60 / 60)
            }
            MyCounter {
                num: parseInt(row.count / 60)
            }
            MyCounter {
                num: row.count % 60
            }
```

```
        }

        Button {
            anchors.horizontalCenter: parent.horizontalCenter
            width: 200
            height: 100
            text: "Старт"
            onClicked: {
                timer.running = !timer.running
                console.log(text)
                text = text === "Старт" ? "Пауза" : "Старт"
            }
        }
    }

    Timer {
        id: timer
        interval: 1000
        repeat: true
        running: false
        onTriggered: {
            row.count++
        }
    }


    Row {
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.bottom: parent.bottom
        spacing: 20

        Button {
            text: "Назад"
            onClicked: pageStack.pop()
        }
    }
}
```

MyCounter.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Rectangle {
    width: 200
    height: 150

    border.color: "grey"
```

```
    border.width: 10
    radius: 10

    property string num: "0"

    Label {
        text: num
        anchors.centerIn: parent
    }
}
```