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

Институт информационных технологий, математики и механики

Отчет по практическому заданию для лекции №7

Выполнила:
студентка группы 382006-2
Кулёва Анна Андреевна

Проверил:
Карчков Денис Александрович

Нижний Новгород
2023

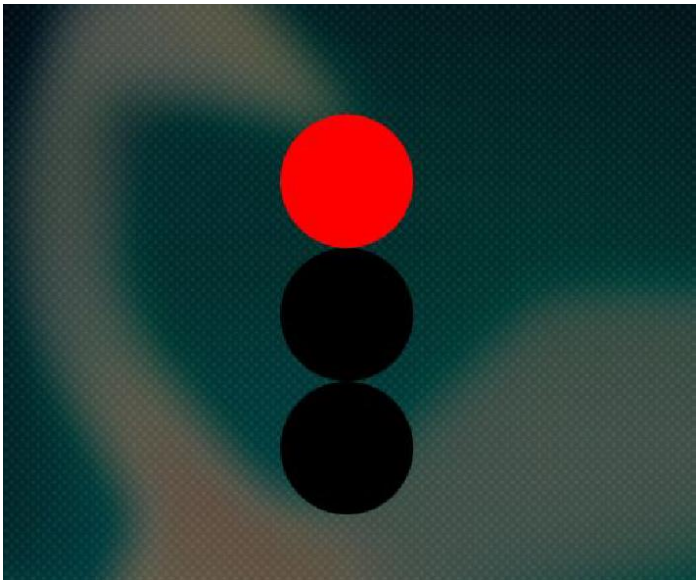
Содержание

1. Цель практического занятия

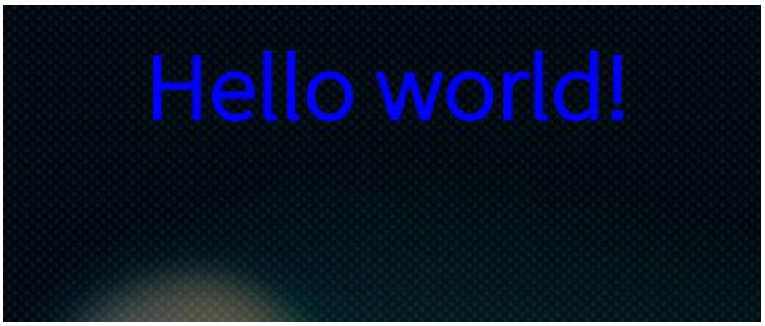
Цель данного практического занятия состоит в том, чтобы научиться создавать пользовательский интерфейс конфигурируемый состояниями, реализовывать анимированные переходы при смене состояний и создавать собственные QML компоненты.

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. Руководство пользователя

При запуске программы пользователь увидит главную страницу с первым заданием: приложением, отображающим светофор (рисунок 1). Четвёртое задание выглядит аналогично. Переключаться между заданиями можно с помощью кнопок «Вперёд» и «Назад», нажав на которые пользователь перейдёт на следующую страницу или вернётся к предыдущей.

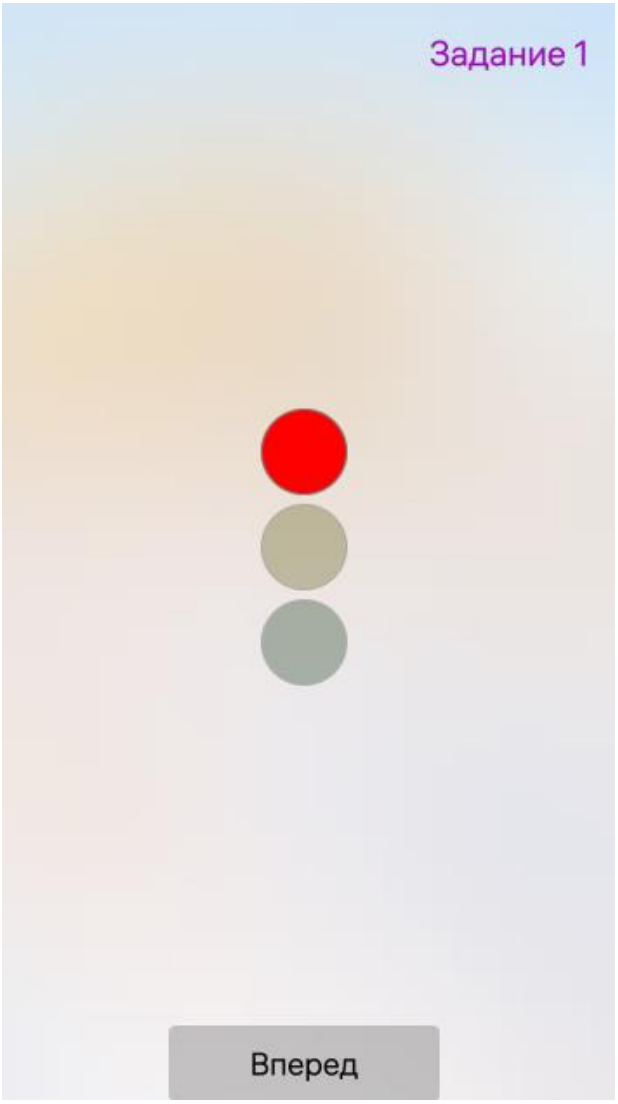


Рисунок 1. Светофор

При переходе к следующему заданию, пользователь увидит доработанное первое задание. Когда загорается зелёный свет, от левого края до правого плавно двигается фигурка человека (рисунок 2).

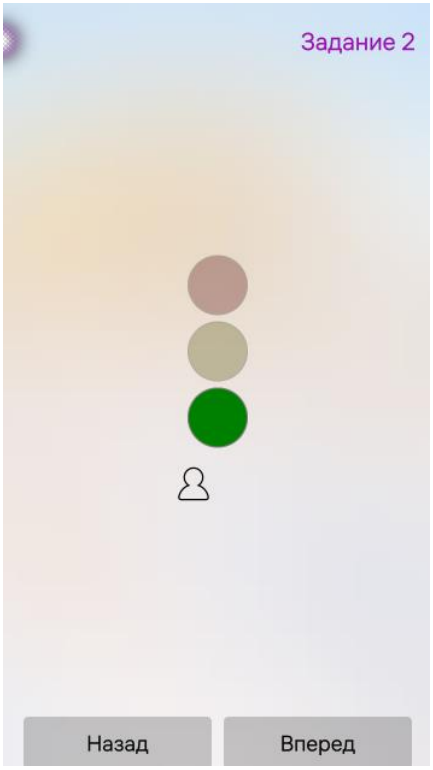


Рисунок 2. Доработанное первое задание

На следующей странице отображается текст, который анимируется при нажатии.

Рисунок 3. Анимированный текст

На пятой странице пользователь увидит поле и кнопку, текст в которых совпадает. Пользователь также может изменить цвет элементов с помощью кнопки «Изменить цвет на красный» (рисунок 4).

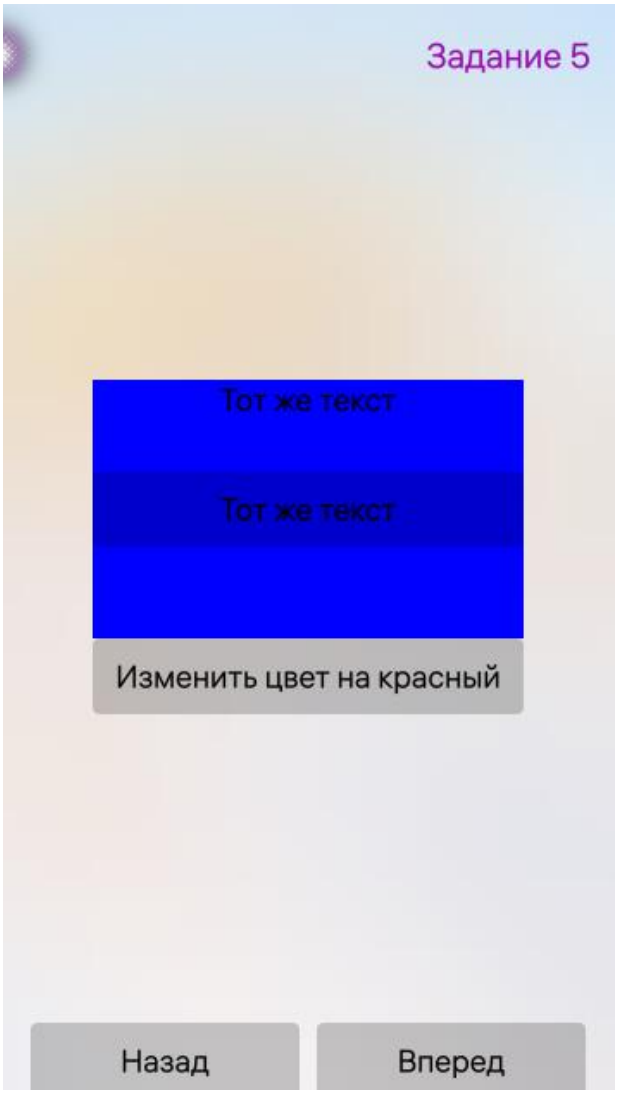


Рисунок 4. Текстовое поле и кнопка

На следующей странице отображается секундомер, который можно запустить по кнопке «Старт» и остановить по кнопке «Стоп» (рисунок 5-6). В трёх окнах отображаются соответственно часы, минуты и секунды. Время старта и останова отображаются ниже.

Рисунок 5-6. Секундомер

При переходе со станицы на страницу в консоли отображаются данные о количестве добавленных и удалённых страниц (рисунок 7).

```
[D] expression for onDepthChanged:56 - добавленных: 4 удалённых: 0
[D] onCompleted:31 - Button_QMLTYPE_118(0x1be5980)
[D] onCompleted:31 - QQuickMouseArea(0x1be84c0)
[D] onCompleted:31 - Label_QMLTYPE_86(0x1be8b10)
[D] expression for onDepthChanged:56 - добавленных: 5 удалённых: 0
[D] onClicked:21 - Clicked!
[D] onClicked:21 - Clicked!
[D] expression for onDepthChanged:56 - добавленных: 6 удалённых: 0
[D] onClicked:41 - Старт 0:0:0
[D] onClicked:41 - Стоп 0:1:94
[D] expression for onDepthChanged:56 - добавленных: 6 удалённых: 1
[D] expression for onDepthChanged:56 - добавленных: 7 удалённых: 1
[D] onClicked:41 - Старт 0:0:0
[D] onClicked:41 - Стоп 0:1:84
```

Рисунок 7. Консоль

4. Руководство программиста

Программа реализована на языке программирования QML.

1. Выполнение шага 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"
            }
        }

        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"
                PropertyChanges { target: redCircle; opacity: 0.3}
                PropertyChanges { target: yellowCircle; opacity: 1}
                PropertyChanges { target: greenCircle; opacity: 0.3}
            }
        ]
    }
```

2. Выполнение шага 2. Светофор с анимированной иконкой человека:

```
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"
            }
        }

        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"
                PropertyChanges { target: redCircle; opacity: 0.3}
                PropertyChanges { target: yellowCircle; opacity: 1}
                PropertyChanges { target: greenCircle; opacity: 0.3}
            }
        ]
    }
```

3. Выполнение шага 3. Анимированный текст:

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

        }
    ]

    MouseArea {
        id: mouseArea
        anchors.fill: parent

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

4. Выполнение шага 4. Светофор с отдельными компонентами:

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
    }
}
```

Task4.qml

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

5. Выполнение шага 5. QML компонент MyButton:

```
Rectangle {
    id: rectangle
    height: 300;
    width: 500;
    color: setColor
    property string setColor: "blue"
    property string btnText: "text"
    Button {
        height: parent.height;
        width: parent.width;
        anchors.horizontalCenter: parent.horizontalCenter;
        anchors.verticalCenter: parent.verticalCenter;
        text: btnText
    }

    MouseArea {
            anchors.fill: parent
            onClicked: console.log("Clicked!")
        }

        function changeColor(color) {
             rectangle.setColor = color;
        }
```

```
    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
                }
            }
        }
    }
}
```

6. Выполнение шага 6. Секундомер (с использованием QML компоненты MyCounter):

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
        color: "black"
    }
}
```

Task6.qml

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

            id: row
            property int count: 0

            MyCounter {
                num: parseInt(row.count / 100 / 60)
            }
            MyCounter {
                num: parseInt(row.count / 100 % 60)
            }
            MyCounter {
                num: parseInt(row.count % 100)
            }
```

```
        }

        Button {
            anchors.horizontalCenter: parent.horizontalCenter
            width: 200
            height: 100
            text: "Старт"
            onClicked: {
                var time = parseInt(row.count / 100 / 60) + ":" +
parseInt(row.count / 100 % 60) + ":" + parseInt(row.count % 100)
                timer.running = !timer.running
                console.log(text === "Старт" ? "Старт " + time : "Стоп
" + time)
                timeModel.append({ time: text === "Старт" ? "Старт " +
time : "Стоп " + time })
                text = text === "Старт" ? "Стоп" : "Старт"
            }
        }

        SilicaListView {
            width: parent.width
            height: 400
            model: timeModel

            delegate: Text {
                text: time
            }

            spacing: 5
        }

    }

    ListModel {
        id: timeModel
    }

    Timer {
        id: timer
        interval: 3
        repeat: true
        running: false
        onTriggered: {
            row.count++
            // console.log(row.count)
        }
    }
```

7. Выполнение шага 7. Обработчик сигналов PageStack:

```
    Connections {
        property int pushed: 0
        property int popped: 0
        property int current: 0
        target: pageStack
```

```
        onDepthChanged: {
            if (current < pageStack.depth) pushed++
            if (current > pageStack.depth) popped++
            current = pageStack.depth;
            console.log("добавленных: " + pushed, "удалённых: " +
popped);
        }
    }
```

Заключение

В данной лабораторной работе я научилась создавать пользовательский интерфейс конфигурируемый состояниями, реализовывать анимированные переходы при смене состояний и создавать собственные QML компоненты. Также были выполнены все шаги практического задания.

Приложение

MyButton.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Rectangle {
    id: rectangle
    height: 300;
    width: 500;
    color: setColor
    property string setColor: "blue"
    property string btnText: "text"
    Button {
        height: parent.height;
        width: parent.width;
        anchors.horizontalCenter: parent.horizontalCenter;
        anchors.verticalCenter: parent.verticalCenter;
        text: btnText
    }

    MouseArea {
            anchors.fill: parent
            onClicked: console.log("Clicked!")
        }

        function changeColor(color) {
             rectangle.setColor = color;
        }


    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
                }
            }
        }
    }
}
```

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
        color: "black"
    }
}
```

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
    }
}
```

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"
        }
    }

    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"
            PropertyChanges { target: redCircle; opacity: 0.3}
            PropertyChanges { target: yellowCircle; opacity: 1}
            PropertyChanges { target: greenCircle; opacity: 0.3}
        }
    ]
}
```

```
    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"
    }
}

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"
```

```
                PropertyChanges { target: redCircle; opacity: 0.3}
                PropertyChanges { target: yellowCircle; opacity: 1}
                PropertyChanges { target: greenCircle; opacity: 0.3}
            }
        ]
    }

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

        }
    ]

    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"
        }
    }

    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"
                StateChangeScript {
                    script: {
                        redCircle.opacity = 0.3
                        yellowCircle.opacity = 1
                        greenCircle.opacity = 0.3
                    }
                }
            }
        ]
    }

    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")))
        }
    }
}
```

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: "Тот же текст"
            }
            setColor: "blue"
        }
        Button {
            id: button
            width: myButton.width
            anchors.centerIn: parent.Center
            onClicked: myButton.changeColor("red")
            text: "Изменить цвет на красный"
        }


    }

    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("Task7.qml")))
        }
    }
}
```

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 / 100 / 60)
            }
            MyCounter {
                num: parseInt(row.count / 100 % 60)
            }
            MyCounter {
```

```
                num: parseInt(row.count % 100)
            }
        }

        Button {
            anchors.horizontalCenter: parent.horizontalCenter
            width: 200
            height: 100
            text: "Старт"
            onClicked: {
                var time = parseInt(row.count / 100 / 60) + ":" +
parseInt(row.count / 100 % 60) + ":" + parseInt(row.count % 100)
                timer.running = !timer.running
                console.log(text === "Старт" ? "Старт " + time : "Стоп " +
time)
                timeModel.append({ time: text === "Старт" ? "Старт " + time :
"Стоп " + time })
                text = text === "Старт" ? "Стоп" : "Старт"
            }
        }

        SilicaListView {
            width: parent.width
            height: 400
            model: timeModel

            delegate: Text {
                text: time
            }

            spacing: 5
        }

    }

    ListModel {
        id: timeModel
    }

    Timer {
        id: timer
        interval: 3
        repeat: true
        running: false
        onTriggered: {
            row.count++
            // console.log(row.count)
        }
    }


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

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

Lab7.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

ApplicationWindow {
    objectName: "applicationWindow"
    initialPage: Qt.resolvedUrl("pages/Task1.qml")
    cover: Qt.resolvedUrl("cover/DefaultCoverPage.qml")
    allowedOrientations: defaultAllowedOrientations

    Connections {
        property int pushed: 0
        property int popped: 0
        property int current: 0
        target: pageStack
        onDepthChanged: {
            if (current < pageStack.depth) pushed++
            if (current > pageStack.depth) popped++
            current = pageStack.depth;
            console.log("добавленных: " + pushed, "удалённых: " + popped);
        }
    }
}
```