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

#### Кафедра МОСТ

##### Направление подготовки: «Программная инженерия»

##### Профиль подготовки: «Разработка программно-информационных систем»

## ОТЧЕТ

### По лабораторной работе №7

##### по дисциплине «Технология программирования мобильных систем»

**Выполнил:** студент группы 381908-4

Кургузиков Кирилл

#### Руководитель:

##### доцент кафедры программной инженерии

##### Борисов Николай Анатольевич

##### Нижний Новгород 2022

**Содержание**

[1. Цель 3](#_Toc97973273)

[2. Постановка задачи 4](#_Toc97973274)

[3. Решение поставленной задачи 6](#_Toc97973275)

[4. Руководство пользователя 8](#_Toc97973276)

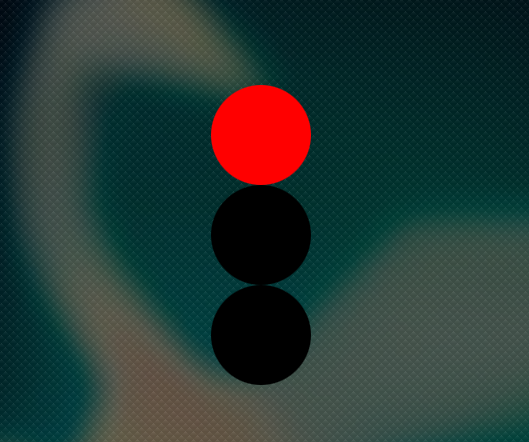
[Приложение 13](#_Toc97973277)

# Цель

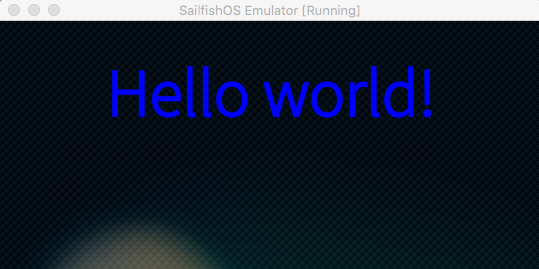
# Научиться создавать пользовательский интерфейс конфигурируемый состояниями, реализовывать анимированные переходы при смене состояний и создавать собственные QML компоненты.

# Постановка задачи

1. 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.



1. Доработать задание 1 так, чтобы во время зеленого сигнала светофора из одного конца экрана в другой плавно двигалась иконка человечка.
2. 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.



1. Выделить сигналы светофора из задания 1 в отдельный компонент и использовать его.
2. 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.
3. 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.
4. Добавить обработчик сигналов PageStack, подсчитывающий количество добавленных и удаленных страниц в PageStack.

# Решение поставленной задачи

1. Светофор

Изменения состояний:

state: {

if (*delayCnt* < 10){

"red"

} else if (*delayCnt* < 20) {

"yellow"

} else {

"green"

}

}

Состояния:

states: [

State {

name: "red"

PropertyChanges { target: *rec\_red*; opacity: 1}

},

State {

name: "yellow"

PropertyChanges { target: *rec\_yellow*; opacity: 1}

},

State {

name: "green"

PropertyChanges { target: *rec\_green*; opacity: 1}

}

]

Таймер:

Timer {

interval: 100

repeat: true

running: true

onTriggered: {

if (*column*.delayCnt == 100) {

*column*.delayCnt = 0;

} else {

*column*.delayCnt++;

}

}

}

2. Светофор с анимацией

В основном реализация такая же, как и в 1 задании, за исключением состояний и анимации.

Картинка:

Image {

id: *car*

x: 100

y: 550

source: "../pics/car.png"

}

Состояния:

states: [

State {

name: "red"

PropertyChanges { target: *rec\_red*; opacity: 1}

StateChangeScript {

script: *anim\_car\_back*.running = true;

}

},

State {

name: "yellow"

PropertyChanges { target: *rec\_yellow*; opacity: 1}

},

State {

name: "green"

PropertyChanges { target: *rec\_green*; opacity: 1}

StateChangeScript {

script: *anim\_car\_forward*.running = true;

}

}

]

Анимация:

PropertyAnimation {

id: *anim\_car\_forward*

target: *car*;

property: "x";

from: 100;

to: 500;

duration: 1000;

}

PropertyAnimation {

id: *anim\_car\_back*

target: *car*;

property: "x";

from: 500;

to: 100;

duration: 1000;

}

3. Анимация текста

Состояния:

states: [

State {

name: "way"

},

State {

name: "back"

}

]

Действие, при изменении состояний:

transitions: [

Transition {

from: "back"

to: "way"

ParallelAnimation {

PropertyAnimation { target: *lb*; properties: "y"; from: *lb*.y; to: 1000; duration: 2000;}

PropertyAnimation { target: *lb*; properties: "color"; from: *lb*.color; to: "blue"; duration: 100;}

RotationAnimation { target: *lb*; from: 0; to: 180; duration: 1000;}

}

},

Transition {

from: "way"

to: "back"

PropertyAnimation { target: *lb*; properties: "y"; from: *lb*.y; to: 250; duration: 2000}

PropertyAnimation { target: *lb*; properties: "color"; from: *lb*.color; to: "white"; duration: 100;}

RotationAnimation { target: *lb*; from: *lb*.rotation; to: 0; duration: 1000;}

}

]

4. Светофор с отдельными компонентами

Сигналы светофора выделены в отдельный файл:

Item {

property string setColor: "white"

property string setOpacity: "0.2"

anchors.horizontalCenter: *parent*.horizontalCenter;

width: 100

height: 100

Rectangle {

radius: 100;

anchors.fill: *parent*

color: *setColor*

opacity: *setOpacity*

}

}

Объявление сигналов в основном коде:

LightSignal {

id: *ls\_red*

y: 200;

setColor: "red"

}

LightSignal {

id: *ls\_yellow*

anchors.top: *ls\_red*.bottom

setColor: "yellow"

}

LightSignal {

id: *ls\_green*

anchors.top: *ls\_yellow*.bottom

setColor: "green"

}

5. Собственный компонент QML Button

Объявление собственного компонента в отдельном файле:

Rectangle {

height: 100;

width: 300;

radius: 20;

color: *setColor*

Button {

height: *parent*.height;

width: *parent*.width;

anchors.horizontalCenter: *parent*.horizontalCenter;

anchors.verticalCenter: *parent*.verticalCenter;

Label {

anchors.horizontalCenter: *parent*.horizontalCenter;

anchors.verticalCenter: *parent*.verticalCenter;

text: *someText*.text

}

}

}

Использование собственного компонента:

CustomButton {

id: *customButton*;

x: 200;

y: 500;

Label { text: "Кнопка" }

setColor: "red"

}

6. Приложение-секундомер

Создан файл CustomWatch.qml с логикой и визуализацией секундомера.

Использование собственного секундомера в программе:

CustomWatch {

id: *cw*;

x: 100;

y: 300;

}

7. Обработчик сигналов PageStack

Connections {

target: *pageStack*

onDepthChanged: {

*current* = *pageStack*.depth + 1;

*console*.log("depth: " + *current*);

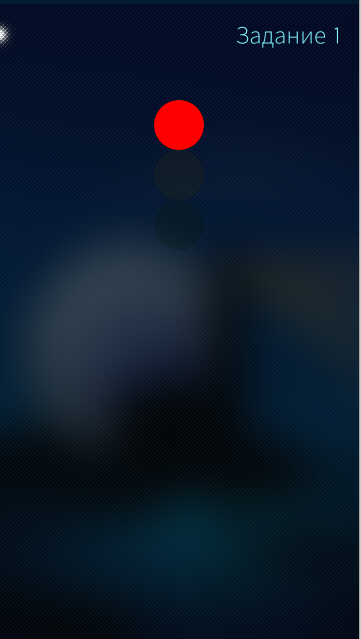
}

}

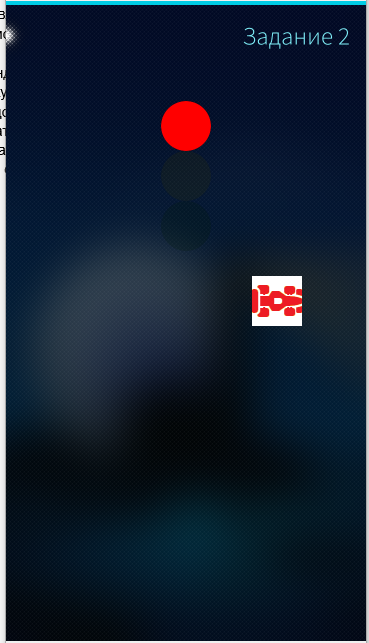
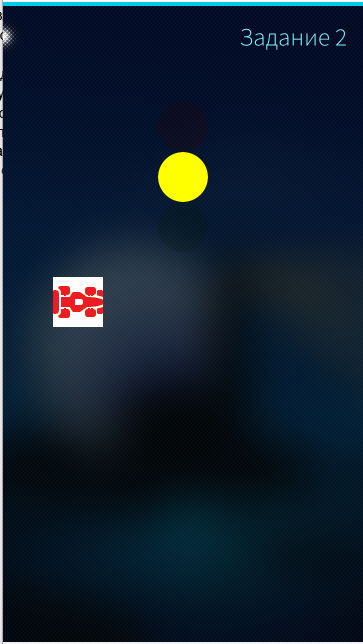
# Руководство пользователя

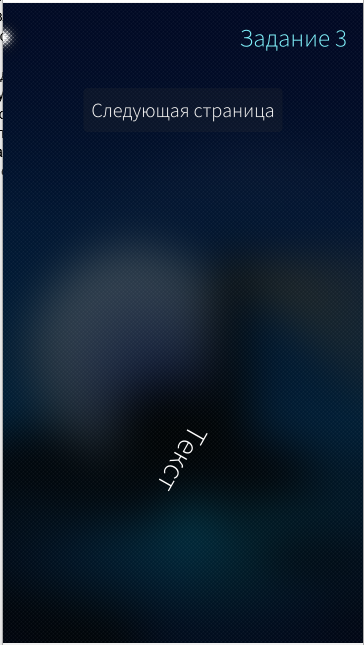
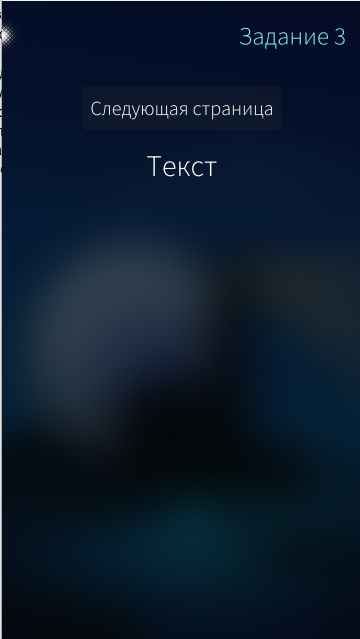
# 1. Светофор

# 

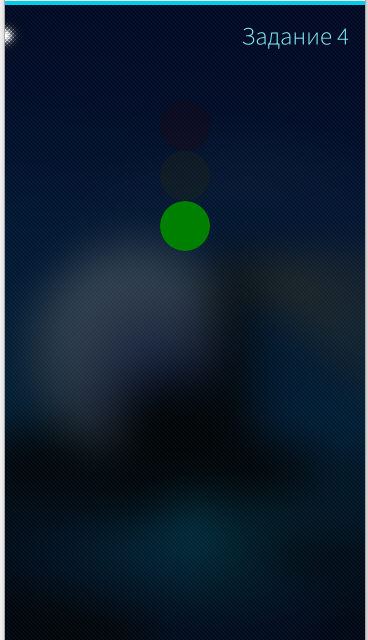


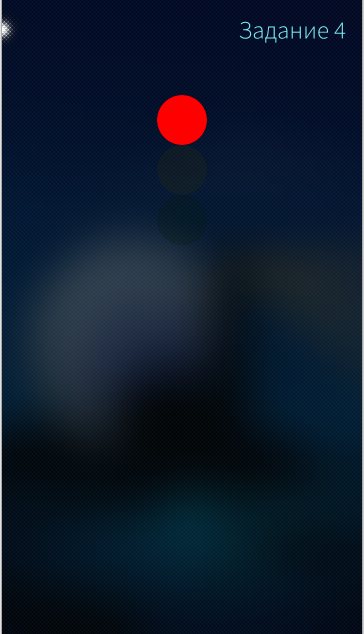
2. Светофор с анимацией

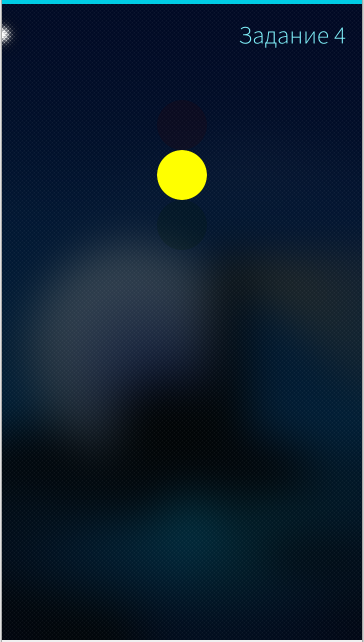
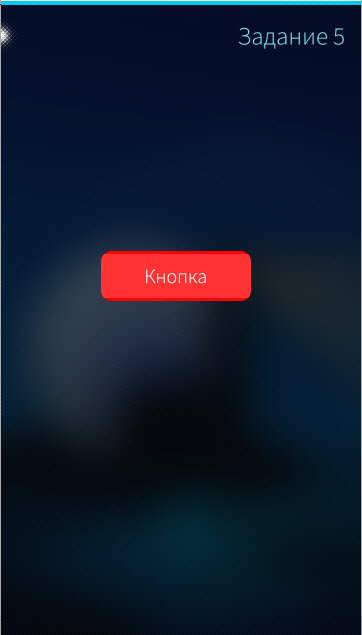


3. Анимация текста

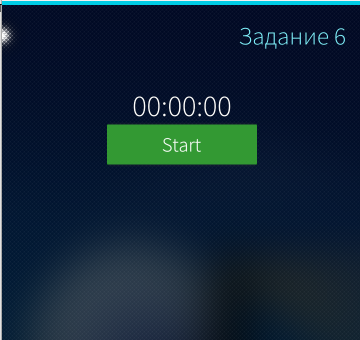
4. Светофор с сигналами в отдельном компоненте

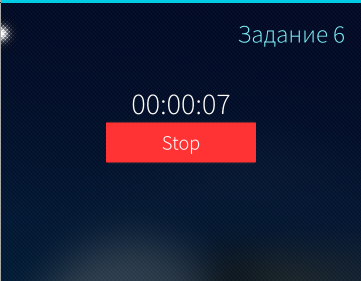




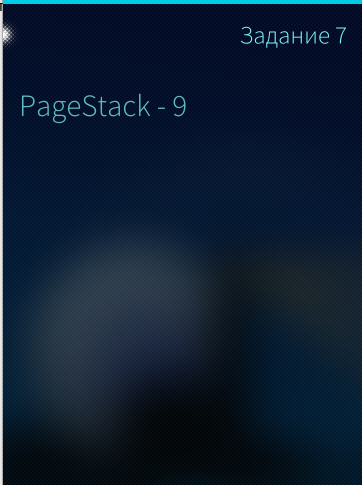
5. Собственная кнопка  


6. Секундомер на собственных компонентах





7. обработчик сигналов PageStack



# Приложение

CustomButton.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Item {

property string setColor: "blue"

default property var someText

Rectangle {

height: 100;

width: 300;

radius: 20;

color: *setColor*

Button {

height: *parent*.height;

width: *parent*.width;

anchors.horizontalCenter: *parent*.horizontalCenter;

anchors.verticalCenter: *parent*.verticalCenter;

Label {

anchors.horizontalCenter: *parent*.horizontalCenter;

anchors.verticalCenter: *parent*.verticalCenter;

text: *someText*.text

}

}

}

}

CustomWatch.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Item {

property string setHour: "00"

property string setMin: "00"

property string setSec: "00"

property int hour: 0;

property int min: 0;

property int sec: 0;

height: *lb*.height;

width: *lb*.width;

anchors.horizontalCenter: *parent*.horizontalCenter;

Label {

id: *lb*

text: *setHour* + ":" + *setMin* + ":" + *setSec*

font.pixelSize: 60

}

Rectangle {

id: *rt*

width: 300;

height: 80;

anchors.top: *lb*.bottom

anchors.horizontalCenter: *parent*.horizontalCenter;

color: "Green"

Button {

id: *bt*;

text: "Start"

width: *parent*.width;

height: *parent*. height;

MouseArea {

id: *mouseArea*

anchors.fill: *parent*

}

}

}

Component.onCompleted: *mouseArea*.clicked.connect(*mouseAreaClickedHandler*)

function *mouseAreaClickedHandler*(mouse) {

*console*.log("ClickedHandler")

if (*bt*.text === "Start")

{

*bt*.text = "Stop";

*rt*.color = "red"

*tm*.start();

}

else

{

*bt*.text = "Start";

*rt*.color = "Green"

*setHour* = "00";

*setMin* = "00";

*setSec* = "00";

*tm*.stop();

}

}

Timer {

id: *tm*;

interval: 1000;

repeat: true

running: false

onTriggered: {

if (*sec* != "59")

{

*sec*++;

if (*sec* < 10) {

*setSec* = "0" + *sec*;

}

else {

*setSec* = *sec*;

}

}

else {

*sec* = 0;

*setSec* = "00";

*min*++;

if (*min* != "59")

{

if (*min* < 10) {

*setMin* = "0" + *min*;

}

else {

*setMin* = *min*;

}

}

else{

*min* = 0;

*setMin* = *min*;

*hour*++;

if (*hour* != "23")

{

if (hout < 10) {

*setHour* = "0" + *hour*;

}

else {

*setHour* = *hour*;

}

}else {

*hour* = 0 ;

*setHour* = "00";

*min* = 0;

*setMin* = "00";

*sec* = 0;

setSet = "00";

*tm*.stop();

}

}

}

}

}

}

LightSignal.qml

import QtQuick 2.0

Item {

property string setColor: "white"

property string setOpacity: "0.2"

anchors.horizontalCenter: *parent*.horizontalCenter;

width: 100

height: 100

Rectangle {

radius: 100;

anchors.fill: *parent*

color: *setColor*

opacity: *setOpacity*

}

}

Page\_1.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page*

allowedOrientations: Orientation.All

SilicaFlickable {

anchors.fill: *parent*

PullDownMenu {

MenuItem {

text: *qsTr*("Show Page 2")

onClicked: pageStack.animatorPush(*Qt*.resolvedUrl("Page\_2.qml"))

}

}

contentHeight: *column*.height

Column {

property int delayCnt: 0

id: *column*

width: *page*.width

spacing: Theme.paddingLarge

PageHeader {

title: *qsTr*("Задание 1")

}

state: {

if (*delayCnt* < 10){

"red"

} else if (*delayCnt* < 20) {

"yellow"

} else {

"green"

}

}

Rectangle {

id: *rec\_red*

anchors.horizontalCenter: *parent*.horizontalCenter;

y: 200;

width: 100; height: 100

color: "red"

radius: 100;

opacity: 0.2

}

Rectangle {

id: *rec\_yellow*

anchors.top: *rec\_red*.bottom

anchors.horizontalCenter: *parent*.horizontalCenter

width: 100; height: 100

color: "yellow"

radius: 100;

opacity: 0.2

}

Rectangle {

id: *rec\_green*

anchors.top: *rec\_yellow*.bottom

anchors.horizontalCenter: *parent*.horizontalCenter

width: 100; height: 100

color: "green"

radius: 100;

opacity: 0.2

}

states: [

State {

name: "red"

PropertyChanges { target: *rec\_red*; opacity: 1}

},

State {

name: "yellow"

PropertyChanges { target: *rec\_yellow*; opacity: 1}

},

State {

name: "green"

PropertyChanges { target: *rec\_green*; opacity: 1}

}

]

}

Timer {

interval: 100

repeat: true

running: true

onTriggered: {

if (*column*.delayCnt == 100) {

*column*.delayCnt = 0;

} else {

*column*.delayCnt++;

}

}

}

}

}

Page\_2.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page*

allowedOrientations: Orientation.All

SilicaFlickable {

anchors.fill: *parent*

PullDownMenu {

MenuItem {

text: *qsTr*("Show Page 3")

onClicked: pageStack.animatorPush(*Qt*.resolvedUrl("Page\_3.qml"))

}

}

contentHeight: *column*.height

Column {

property int delayCnt: 0

id: *column*

width: *page*.width

spacing: Theme.paddingLarge

PageHeader {

title: *qsTr*("Задание 2")

}

state: {

if (*delayCnt* < 20){

"red"

} else if (*delayCnt* < 50) {

"yellow"

} else {

"green"

}

}

Image {

id: *car*

x: 100

y: 550

source: "../pics/car.png"

}

Rectangle {

id: *rec\_red*

anchors.horizontalCenter: *parent*.horizontalCenter;

y: 200;

width: 100; height: 100

color: "red"

radius: 100;

opacity: 0.2

}

Rectangle {

id: *rec\_yellow*

anchors.top: *rec\_red*.bottom

anchors.horizontalCenter: *parent*.horizontalCenter

width: 100; height: 100

color: "yellow"

radius: 100;

opacity: 0.2

}

Rectangle {

id: *rec\_green*

anchors.top: *rec\_yellow*.bottom

anchors.horizontalCenter: *parent*.horizontalCenter

width: 100; height: 100

color: "green"

radius: 100;

opacity: 0.2

}

states: [

State {

name: "red"

PropertyChanges { target: *rec\_red*; opacity: 1}

StateChangeScript {

script: *anim\_car\_back*.running = true;

}

},

State {

name: "yellow"

PropertyChanges { target: *rec\_yellow*; opacity: 1}

},

State {

name: "green"

PropertyChanges { target: *rec\_green*; opacity: 1}

StateChangeScript {

script: *anim\_car\_forward*.running = true;

}

}

]

}

PropertyAnimation {

id: *anim\_car\_forward*

target: *car*;

property: "x";

from: 100;

to: 500;

duration: 1000;

}

PropertyAnimation {

id: *anim\_car\_back*

target: *car*;

property: "x";

from: 500;

to: 100;

duration: 1000;

}

Timer {

interval: 50

repeat: true

running: true

onTriggered: {

if (*column*.delayCnt == 100) {

*column*.delayCnt = 0;

} else {

*column*.delayCnt++;

}

}

}

}

}

Page\_3.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*("Задание 3")

}

Button {

text: "Следующая страница"

anchors.horizontalCenter: *parent*.horizontalCenter

onClicked: pageStack.animatorPush(*Qt*.resolvedUrl("Page\_4.qml"))

}

state: {

if (*ma*.pressedButtons == 1) {

"way"

} else {

"back"

}

}

Label {

id: *lb*

text: "Текст"

anchors.horizontalCenter: *parent*.horizontalCenter

y: 250;

color: "white"

font.pixelSize: 60

MouseArea {

id: *ma*

anchors.fill: *parent*

}

}

states: [

State {

name: "way"

},

State {

name: "back"

}

]

transitions: [

Transition {

from: "back"

to: "way"

ParallelAnimation {

PropertyAnimation { target: *lb*; properties: "y"; from: *lb*.y; to: 1000; duration: 2000;}

PropertyAnimation { target: *lb*; properties: "color"; from: *lb*.color; to: "blue"; duration: 100;}

RotationAnimation { target: *lb*; from: 0; to: 180; duration: 1000;}

}

},

Transition {

from: "way"

to: "back"

PropertyAnimation { target: *lb*; properties: "y"; from: *lb*.y; to: 250; duration: 2000}

PropertyAnimation { target: *lb*; properties: "color"; from: *lb*.color; to: "white"; duration: 100;}

RotationAnimation { target: *lb*; from: *lb*.rotation; to: 0; duration: 1000;}

}

]

}

}

}

Page\_4.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page*

allowedOrientations: Orientation.All

SilicaFlickable {

anchors.fill: *parent*

PullDownMenu {

MenuItem {

text: *qsTr*("Show Page 5")

onClicked: pageStack.animatorPush(*Qt*.resolvedUrl("Page\_5.qml"))

}

}

contentHeight: *column*.height

Column {

property int delayCnt: 0

id: *column*

width: *page*.width

spacing: Theme.paddingLarge

PageHeader {

title: *qsTr*("Задание 4")

}

state: {

if (*delayCnt* < 10){

"red"

} else if (*delayCnt* < 20) {

"yellow"

} else {

"green"

}

}

LightSignal {

id: *ls\_red*

y: 200;

setColor: "red"

}

LightSignal {

id: *ls\_yellow*

anchors.top: *ls\_red*.bottom

setColor: "yellow"

}

LightSignal {

id: *ls\_green*

anchors.top: *ls\_yellow*.bottom

setColor: "green"

}

states: [

State {

name: "red"

PropertyChanges { target: *ls\_red*; setOpacity: "1"}

},

State {

name: "yellow"

PropertyChanges { target: *ls\_yellow*; setOpacity: "1"}

},

State {

name: "green"

PropertyChanges { target: *ls\_green*; setOpacity: "1"}

}

]

}

Timer {

interval: 100

repeat: true

running: true

onTriggered: {

if (*column*.delayCnt == 100) {

*column*.delayCnt = 0;

} else {

*column*.delayCnt++;

}

}

}

}

}

Page\_5.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page*

allowedOrientations: Orientation.All

SilicaFlickable {

anchors.fill: *parent*

PullDownMenu {

MenuItem {

text: *qsTr*("Show Page 6")

onClicked: pageStack.animatorPush(*Qt*.resolvedUrl("Page\_6.qml"))

}

}

contentHeight: *column*.height

Column {

id: *column*

width: *page*.width

spacing: Theme.paddingLarge

PageHeader {

title: *qsTr*("Задание 5")

}

CustomButton {

id: *customButton*;

x: 200;

y: 500;

Label { text: "Кнопка" }

setColor: "red"

}

}

}

}

Page\_6.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page*

allowedOrientations: Orientation.All

SilicaFlickable {

anchors.fill: *parent*

PullDownMenu {

MenuItem {

text: *qsTr*("Show Page 7")

onClicked: pageStack.animatorPush(*Qt*.resolvedUrl("Page\_7.qml"))

}

}

contentHeight: *column*.height

Column {

id: *column*

width: *page*.width

spacing: Theme.paddingLarge

PageHeader {

title: *qsTr*("Задание 6")

}

CustomWatch {

id: *cw*;

x: 100;

y: 300;

}

}

}

}

Page\_7.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page*

property int count: 0;

allowedOrientations: Orientation.All

SilicaFlickable {

anchors.fill: *parent*

PullDownMenu {

MenuItem {

text: *qsTr*("Show Page 1")

onClicked: *pageStack*.animatorPush(*Qt*.resolvedUrl("Page\_1.qml"))

}

}

contentHeight: *column*.height

Column {

id: *column*

width: *page*.width

spacing: Theme.paddingLarge

PageHeader {

title: *qsTr*("Задание 7")

}

Label {

x: Theme.horizontalPageMargin

text: "PageStack - " + *current*;

color: Theme.secondaryHighlightColor

font.pixelSize: Theme.fontSizeExtraLarge

}

}

}

}

lab\_7\_1.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

import "pages"

ApplicationWindow {

property int current: 0;

Connections {

target: *pageStack*

onDepthChanged: {

*current* = *pageStack*.depth + 1;

*console*.log("depth: " + *current*);

}

}

initialPage: Component { Page\_7 { } }

cover: *Qt*.resolvedUrl("cover/CoverPage.qml")

allowedOrientations: defaultAllowedOrientations

}