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**

Направление подготовки: «Программная инженерия»

**Лабораторная работа №2**

По курсу

**«Программирование для мобильных систем»**

**Выполнил(а):** студент группы 381808-2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Алехин Денис

Подпись Андреевич

**Преподаватель:** доцент кафедры ПИ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Борисов Николай

Подпись Анатольевич

Нижний Новгород  
2021

Содержание

[Введение 3](#_Toc66202768)

[Постановка задачи 4](#_Toc66202769)

[Программная реализация 5](#_Toc66202770)

[Заключение 6](#_Toc66202771)

[Приложения 7](#_Toc66202772)

# Введение

Для позиционирования элементов на экране редко используются абсолютные координаты, так как данный подход не является универсальным для разных типов устройств и разрешений экранов. Для данной задачи зачастую используются другие подходы, которые будут рассмотрены и изучены в данной лабораторной работе.

# Постановка задачи

Освоить базовые навыки построения пользовательских интерфейсов, позиционирования, отрисовки и перемещения элементов. Научиться анимировать элементы. Научиться создавать диалоги и взаимодействовать с ними.

# Программная реализация

Первым этапом написания программы, было изучение способов взаимного расположения элементов. Для этого были использованы такие способы позиционирования и перемещения элементов как anchors, column, row и grid. Первый удобен при нестандартном и сложном расположении элементов относительно друг друга, в то время как column, row и grid удобно использовать при наличии паттерна столбца, строчки или таблицы соответственно.

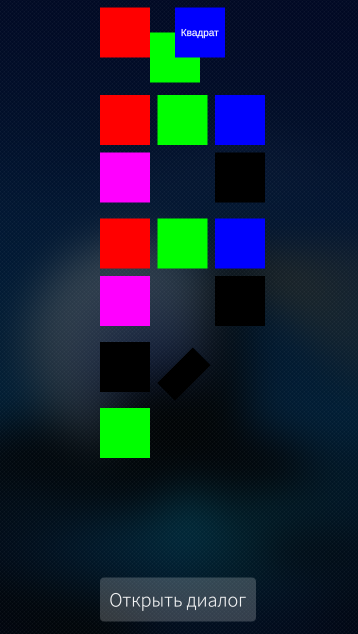
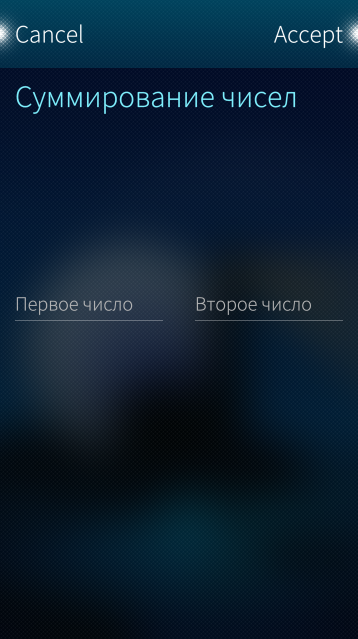
Для изменения размеров, угла поворота и анимации элементов были использованы свойства объектов Scale, Rotation и Translate.

Реализацию диалога, было решено сделать с применением кнопки, при нажатии на которую появляется диалог с возможностью ввести два числа и получить их сумму в консольный вывод.

# Заключение

В результате выполнения лабораторной работы были освоены базовые навыки построения пользовательских интерфейсов, позиционирования, отрисовки и перемещения элементов. Был получен навык анимирования элементов и создания диалоги и взаимодействия с ними.

# Приложения

**** ****

Содержимое файла FirstPage.qml:

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page*

// The effective value will be restricted by ApplicationWindow.allowedOrientations

allowedOrientations: Orientation.All

// To enable PullDownMenu, place our content in a SilicaFlickable

SilicaFlickable {

anchors.fill: *parent*

// PullDownMenu and PushUpMenu must be declared in SilicaFlickable, SilicaListView or SilicaGridView

PullDownMenu {

MenuItem {

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

onClicked: *pageStack*.push(*Qt*.resolvedUrl("SecondPage.qml"))

}

}

// Tell SilicaFlickable the height of its content.

contentHeight: *column*.height

// Place our content in a Column. The PageHeader is always placed at the top

// of the page, followed by our content.

Item {

x: 200; y: 25

Rectangle {

id: *redRect*

width: 100

height: 100

color: "#FF0000"

}

Rectangle {

id: *greenRect*

width: 100

height: 100

color: "#00FF00"

anchors.left: *redRect*.right

anchors.top: *redRect*.verticalCenter

}

Rectangle {

id: *blueRect*

width: 100

height: 100

color: "#0000FF"

anchors.left: *greenRect*.horizontalCenter

anchors.bottom: *greenRect*.verticalCenter

}

Text {

text: "Квадрат"

font.family:"Helvetica"

font.pointSize: 15

color:"white"

anchors.centerIn: *blueRect*

}

}

Column {

id: *column*

x: 200; y: 200

width: *page*.width

spacing: Theme.paddingLarge

Column {

spacing: 15

Row {

spacing: 15

Rectangle {width: 100; height: 100; color: "#FF0000"}

Rectangle {width: 100; height: 100; color: "#00FF00"}

Rectangle {width: 100; height: 100; color: "#0000FF"}

}

Row {

spacing: 130

Rectangle {width: 100; height: 100; color: "#FF00FF"}

Rectangle {width: 100; height: 100; color: "#000000"}

}

}

Grid{

id: *grid*

columns: 3

rows: 2

spacing: 15

Rectangle {width: 100; height: 100; color: "#FF0000"}

Rectangle {width: 100; height: 100; color: "#00FF00"}

Rectangle {width: 100; height: 100; color: "#0000FF"}

Rectangle {width: 100; height: 100; color: "#FF00FF"}

Rectangle {opacity: 0; width: 100; height: 100; color: "#000000"}

Rectangle {width: 100; height: 100; color: "#000000"}

}

Rectangle {

id: *square*

width: 100;

height: 100;

color: "#000000"

}

Rectangle {

width: 100;

height: 100;

color: "#000000"

transform: [

Scale {yScale: 0.5},

Rotation {angle: 315},

Translate {x:115; y: -50 }

]

}

Rectangle {

id:*animRect*

width: 100;

height: 100;

color: "#00FF00"

ParallelAnimation {

id:*anim*

loops: Animation.Infinite

running: true

PropertyAnimation {

target:*animRect*

properties: "scale"

from: 1

to: 1.5

duration: 2000

running: true

}

PropertyAnimation{

target:*animRect*

property: "y"

from: 625

to: 750

duration: 2000

running: true

}

}

}

Button {

text: "Открыть диалог"

onClicked: *pageStack*.push(*dialog*)

transform: Translate {id:*translate*; y: 75}

}

}

}

Dialog {

id: *dialog*

DialogHeader {

title: "Суммирование чисел"

}

TextField {

placeholderText: "Первое число"

id: *firstTextField*

width: *parent*.width / 2

anchors.verticalCenter: *parent*.verticalCenter

anchors.left: *parent*.left

}

TextField {

placeholderText: "Второе число"

id: *secondTextField*

width: *parent*.width / 2

anchors.verticalCenter: *parent*.verticalCenter

anchors.left: *firstTextField*.right

}

onAccepted: *console*.log(parseInt(*firstTextField*.text) + parseInt(*secondTextField*.text))

}

}