

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 №3

Выполнила:

студентка группы 382006-2

Кулёва Анна Андреевна

Проверил:

Карчков Денис Александрович

Нижний Новгород

2023

Содержание

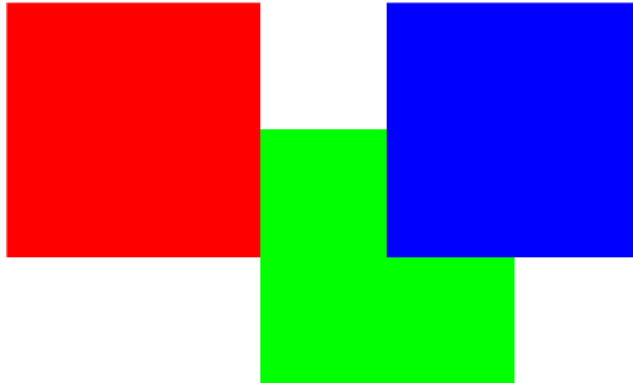
Содержание	2
1. Цель практического занятия.....	3
2. Постановка задачи	4
3. Руководство пользователя	6
4. Руководство программиста.....	9
Заключение.....	11
Приложение.....	12

1. Цель практического занятия

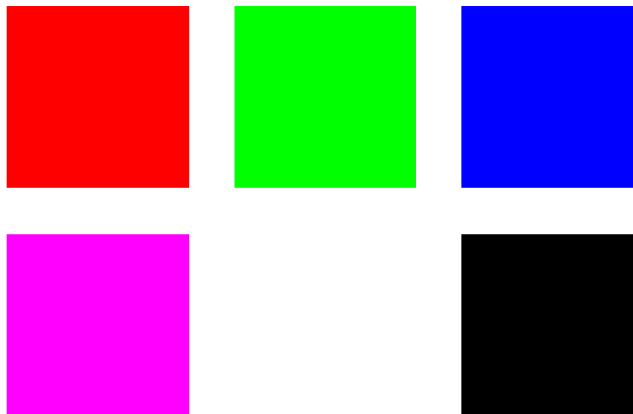
Цель данного практического занятия состоит в том, чтобы освоить базовые навыки построения пользовательских интерфейсов, позиционирования, отрисовки и перемещения элементов; научиться анимировать элементы; научиться создавать диалоги и взаимодействовать с ними.

2. Постановка задачи

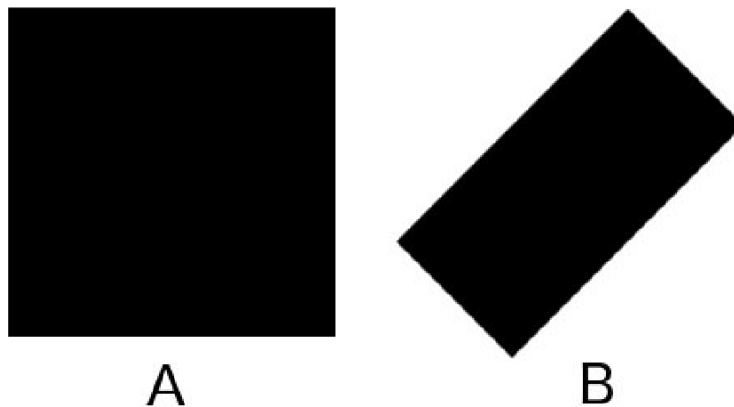
- 1) Создать новый проект со стандартной заготовкой приложения.
- 2) Нарисовать 3 квадрата красного, зелёного и синего цветов следующим образом:



- 3) Поместить текст “Квадрат” белого цвета по центру синего квадрата.
- 4) Нарисовать 5 квадратов с использованием Column и Row следующим образом:

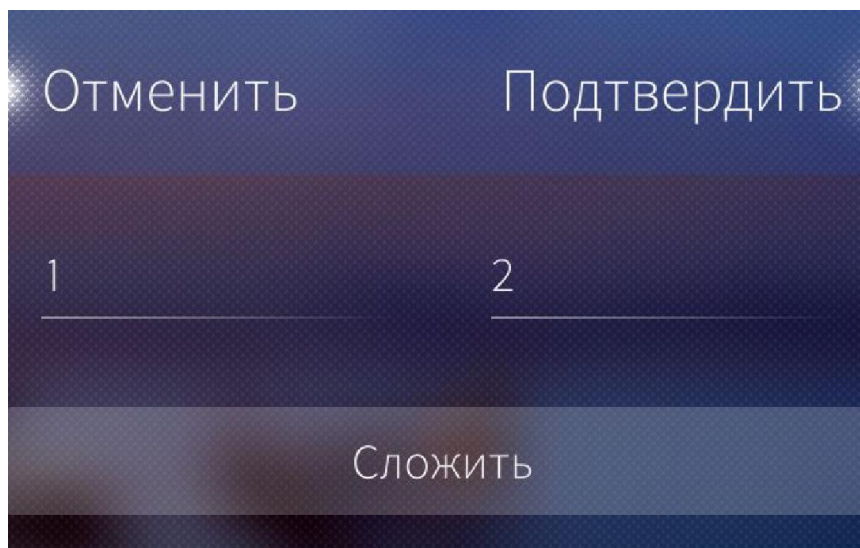


- 5) Нарисовать те же 5 квадратов с использованием Grid.
- 6) Сделать из квадрата “А” прямоугольник “В” с использованием объектов Translate, Scale и Rotation



7) Нарисовать квадрат и анимировать его перемещение вниз с увеличением его размера. Документация по анимации доступна по адресу <http://doc.qt.io/qt-5/qml-qtquick-animation.html>.

8) Реализовать диалог с двумя текстовыми полями, в которые вводятся числа. После нажатия на кнопку “Подтвердить” в консоль выводится сумма чисел. Для преобразования строк к числам использовать функцию `parseInt()`. Валидацией и обработкой ошибок можно пренебречь.



3. Руководство пользователя

Для отображения того или иного задания из списка пользователь должен поставить в определённом объекте (в комментариях объекты пронумерованы номерами заданий) для свойства `visible` значение `true`. Для всех остальных объектов у `visible` должно стоять значение `false`.

При запуске программы пользователь увидит главную страницу, на которой в зависимости от того, какой объект отмечен, как «`visible: true`», будет расположено решение той или иной задачи в виде некоторых отрисованных элементов, анимации или диалога (рисунок 1).

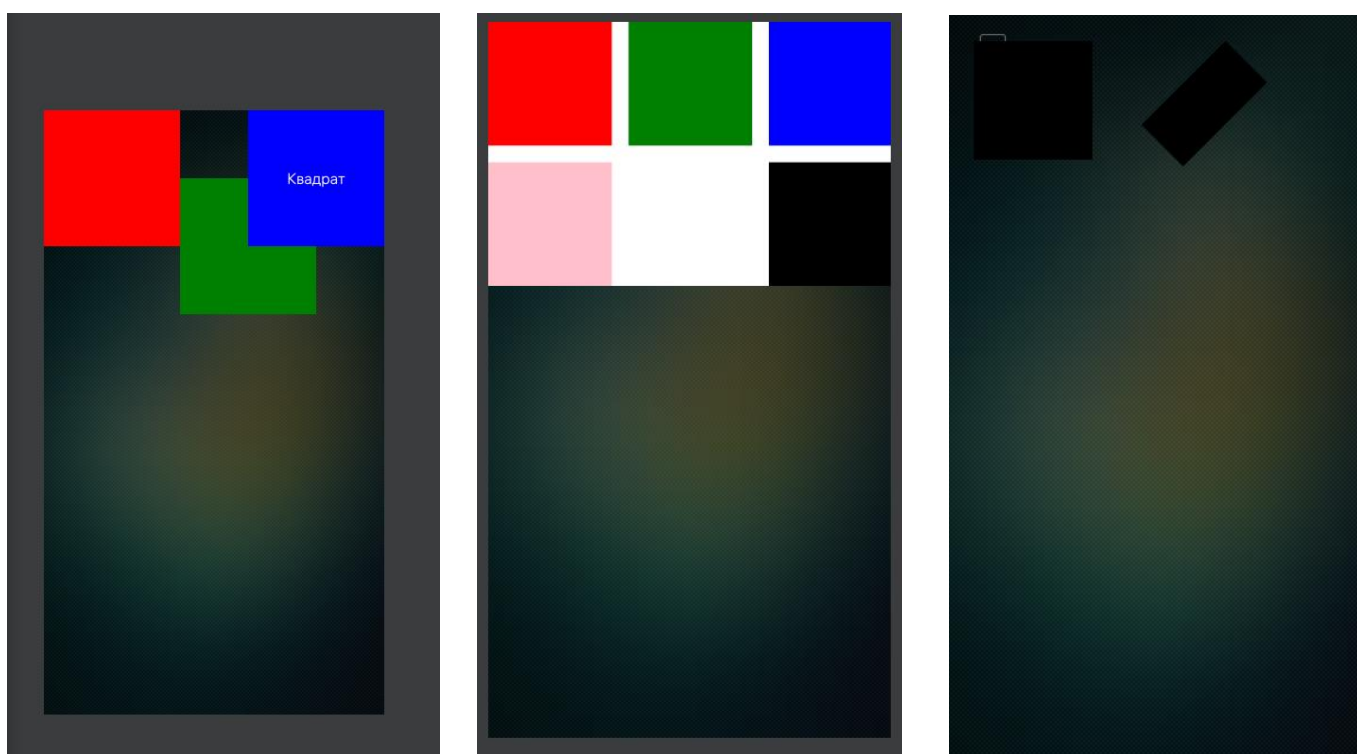


Рисунок 1. Главная страница, первое, второе и третье, четвёртое задания

После запуска шестого задания пользователь сможет взаимодействовать с кнопкой «Addition» (рисунок 2), при нажатии на которую переместиться в диалоговое окно.

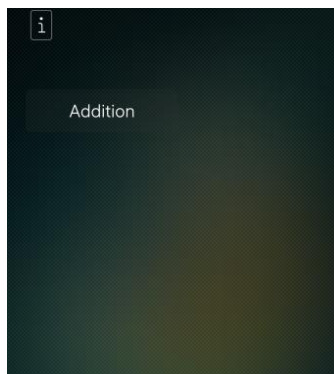


Рисунок 2. Кнопка «Addition», задание 6

В диалоговом окне (рисунок 3) пользователь может взаимодействовать с двумя текстовыми полями, сумму которых впоследствии можно будет рассчитать, нажав на кнопку «Асепт». При нажатии на кнопку «Cancel» сбросятся значения полей, и пользователь вернётся на предыдущую страницу.

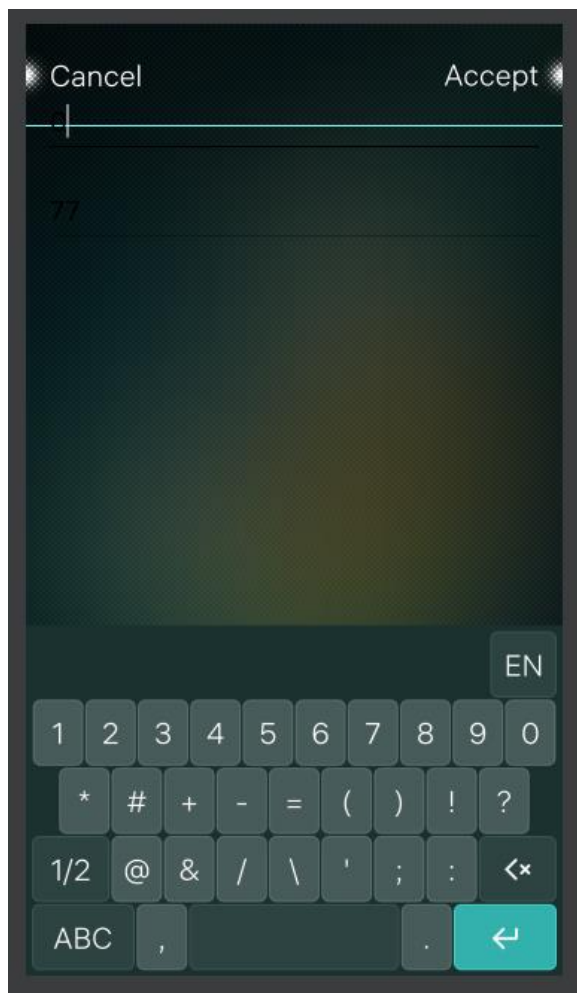
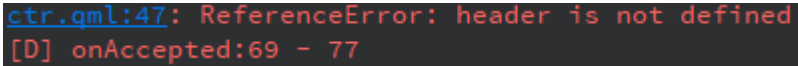


Рисунок 3. Диалоговое окно, задание 6

При этом результат вычислений выносится в консоль (рисунок 4).

A screenshot of a console window with a dark background. It displays a red error message: 'ctr.qml:47: ReferenceError: header is not defined' on the first line, and '[D] onAccepted:69 - 77' on the second line.

```
ctr.qml:47: ReferenceError: header is not defined
[D] onAccepted:69 - 77
```

Рисунок 4. Консоль

4. Руководство программиста

Программа реализована на языке программирования QML. Изменения были произведены в файле MainPage.qml. Также был создан новый файл ctr.qml.

Были добавлены свойства size и size1, равные 1 / 2.5 и 1 / 3.25 ширине экрана:

property int size: Screen.width / 2.5

property int size1: Screen.width / 3.25

1) Для выполнения первого задания были реализованы следующие элементы:

Элемент Rectangle (3 штуки)

- id – идентификатор объекта
- width – ширина объекта в пикселях. Ему присвоена величина size
- height – высота объекта в пикселях. Ему присвоена величина size
- color – цвет объекта

Для того, чтобы расположить квадраты, как показано на рисунке, использовались следующие свойства: anchors.left, anchors.right, anchors.top.

anchors.left: rect1.right

anchors.top: rect1.verticalCenter

anchors.right: rect2.verticalCenter

anchors.left: rect2.horizontalCenter

Элемент Text (1 штука)

- text – текст надписи
- color – цвет надписи

Для того, чтобы расположить текст в центре квадрата использовались свойства:

anchors.verticalCenter: rect3.verticalCenter

anchors.horizontalCenter: rect3.horizontalCenter

2) Для выполнения второго и третьего заданий были использованы элементы Column, Row и Grid, для размещения шести квадратов в 2 строки и 3 столбца. Всем квадратам было присвоено значение size1 для width и height.

- Column – компонент для размещения элементов в столбец
- Row – для размещения элементов в ряд
- Grid – для размещения элементов сеткой

3) Для выполнения третьего задания по трансформации квадрата использовалось свойство `transform` с параметрами `Translate`, `Scale` и `Rotation`.

- `Translate` – перемещение элемента на указанное расстояние
- `Scale` – изменение масштаба элемента по осям в указанное количество раз
- `Rotation` – вращение элемента на указанный угол

4) Для выполнения пятого задания по созданию анимации квадрата использовался `NumberAnimation`, который анимирует изменения числовых значений. `NumberAnimation` имеет следующие параметры:

- `target` – целевой элемент анимации
- `property` – свойство, которое анимация меняет
- `from` – начальное значение анимации
- `to` – конечное значение анимации
- `duration` – продолжительность анимации
- `loops` – зацикливание анимации

5) Для выполнения шестого задания: обработки данных пользователя, используется элемент `Dialog`, в котором есть два свойства, определяющие, что происходит, когда пользователь нажимает `Accept` или `Cancel`: `onAccepted` и `onCanceled`. Для ввода данных используются элементы `TextField`. Для вывода суммы двух чисел в консоль свойство `onAccepted` имеет вид:

```
onAccepted:console.log(parseInt(one.text)+parseInt(two.text))
```

Заключение

В данной лабораторной работе я освоила базовые навыки построения пользовательских интерфейсов, позиционирования, отрисовки и перемещения элементов; научилась анимировать элементы; научилась создавать диалоги и взаимодействовать с ними. Также были выполнены все шаги практического задания.

Приложение

MainPage.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

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

    property int size: Screen.width / 2.5
    property int size1: Screen.width / 3.25

    PageHeader {
        objectName: "pageHeader"
        extraContent.children: [
            IconButton {
                objectName: "aboutButton"
                icon.source: "image://theme/icon-m-about"
                anchors.verticalCenter: parent.verticalCenter

                onClicked: pageStack.push(Qt.resolvedUrl("AboutPage.qml"))
            }
        ]
    }

    //Task 1
    Item {
        visible: false
        id: it1

        Rectangle {
            id: rect1
            color: "red"
            width: size
            height: size
        }

        Rectangle {
            id: rect2
            color: "green"
            width: size
            height: size
            anchors.left: rect1.right
            anchors.top: rect1.verticalCenter
        }

        Rectangle {
            id: rect3
            color: "blue"
            width: size
            height: size
            anchors.right: rect2.verticalCenter
            anchors.left: rect2.horizontalCenter
        }

        Text {
            text: "Квадрат"
            color: "white"
            anchors.verticalCenter: rect3.verticalCenter
            anchors.horizontalCenter: rect3.horizontalCenter
        }
    }
}
```

```

    }

}

}

//Task 2
Item {
    visible: true
    id: it2
    Rectangle {
        width: Screen.width
        height: 2*size1 + 30
        color: "white"
        Column {
            spacing: 30
            Row {
                spacing: 30
                Rectangle {
                    width: size1
                    height: size1
                    color: "red"
                }
                Rectangle {
                    width: size1
                    height: size1
                    color: "green"
                }
                Rectangle {
                    width: size1
                    height: size1
                    color: "blue"
                }
            }
            Row {
                spacing: 30
                Rectangle {
                    width: size1
                    height: size1
                    color: "pink"
                }
                Rectangle {
                    width: size1
                    height: size1
                    color: "green"
                    opacity: 0
                }
                Rectangle {
                    width: size1
                    height: size1
                    color: "black"
                }
            }
        }
    }
}

}

```

```

//Task 3
Item {
    visible: false
    Rectangle {
        width: Screen.width
        height: 2*size1 + 30
        color: "white"
        Grid {

```

```

        spacing: 30
        columns: 3
        rows: 2
        Rectangle {
            width: size1
            height: size1
            color: "red"
        }
        Rectangle {
            width: size1
            height: size1
            color: "green"
        }
        Rectangle {
            width: size1
            height: size1
            color: "blue"
        }
        Rectangle {
            width: size1
            height: size1
            color: "pink"
        }
        Rectangle {
            width: size1
            height: size1
            color: "green"
            opacity: 0
        }
        Rectangle {
            width: size1
            height: size1
            color: "black"
        }
    }
}

//Task 4
Item {
    visible: false

    Rectangle {
        width: 200
        height: 200
        color: 'black'
        transform: [Translate{x:50; y:50}]
    }

    Rectangle {
        width: 100
        height: 100
        color: 'black'
        transform: [
            Translate{x:370; y:-150},
            Scale { yScale: 2 },
            Rotation { angle: 45} ]
    }
}

// Task 5
Item {
    visible: false
    Rectangle{

```

```

        id: animated
        width: 200; height: 200
        color: 'black'
        transform: [Translate{x:200; y:300}]
        NumberAnimation on y {
            from: 0; to: 100
            loops: Animation.Infinite
            duration: 500
        }
        NumberAnimation on width {
            from: 200; to: 400
            loops: Animation.Infinite
            duration: 500
        }
        NumberAnimation on height {
            from: 200; to: 400
            loops: Animation.Infinite
            duration: 500
        }
    }
}

// Task 6: Calculator
Item{
    visible: false
    x:50; y:200
    Button{
        text:"Addition"
        onClicked: pageStack.push(Qt.resolvedUrl("ctr.qml"))
    }
}
}

```

ctr.qml

```

import QtQuick 2.0
import Sailfish.Silica 1.0

Dialog {
    DialogHeader{id:head}
    Column{
        y:100
        height:120
        anchors.top:header.bottom
        width:parent.width

        TextField{
            id:one
            width:parent.width
            height:120
            color:"black"
            anchors.top:header.bottom
        }

        TextField{
            id:two
            width:parent.width
            height:120
        }
    }
}

```

```
        color:"black"  
        anchors.top:header.bottom  
    }  
}  
  
onAccepted:console.log(parseInt(one.text)+parseInt(two.text))  
}
```