МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ  
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»

**ОТЧЁТ**

по лабораторной работе №7

**СОБСТВЕННЫЕ КОМПОНЕНТЫ**

**Выполнил:** студент группы

382008-1

\_\_\_А.Д. Кандрин \_\_\_\_\_\_\_

Подпись

**Проверил:** младший научный сотрудник

\_\_\_\_Д.А. Карчков\_\_\_\_\_\_\_\_

Подпись

Нижний Новгород  
2023 г.

## Содержание

[1. Постановка задачи 3](#_Toc27873363)

[2. ХОД РАБОТЫ](#_Toc27873364) 4

[3. ЗАКЛЮЧЕНИЕ](#_Toc27873365) 15

[4. ЛИТЕРАТУРА](#_Toc27873374) 16

## Постановка задачи.

Цель: научиться создавать пользовательский интерфейс конфигурируемый состояниями, реализовывать анимированные переходы при смене состояний и создавать собственные QML компоненты.

## Ход работы.

1. Создадим приложение, отображающее светофор. На экране должно присутствовать 3 разноцветных сигнала, которые загораются и гаснут в том же порядке, что и сигналы светофора. Сделаем автоматическую смену состояний.

Column {

id: column

width: page.width

spacing: Theme.paddingLarge

states: [

State

{

name: "Stop"

},

State

{

name: "WaitForStart"

},

State

{

name: "Wait"

},

State

{

name: "Start"

},

State

{

name: "WaitForStop"

}

]

state: "Stop"

function switchState()

{

if (column.state == "Stop")

column.state = "WaitForStart"

else if (column.state == "WaitForStart")

column.state = "Start"

else if (column.state == "Start")

column.state = "WaitForStop"

else if (column.state == "WaitForStop")

column.state = "Wait"

else if (column.state == "Wait")

column.state = "Stop"

}

Timer

{

interval: 3500

running: true

repeat: true

onTriggered:

{

column.switchState()

}

}

Rectangle

{

id: redLight

anchors.horizontalCenter: parent.horizontalCenter

width: 300

height: 300

radius: width / 2

color: "maroon"

states: [

State

{

name: "On"

when: column.state == "Stop" || column.state == "WaitForStart"

PropertyChanges {

target: redLight

color: "red"

}

},

State

{

name: "Off"

when: !(column.state == "Stop" || column.state == "WaitForStart")

PropertyChanges {

target: redLight

color: "maroon"

}

}

]

}

Rectangle

{

id: yellowLight

anchors.horizontalCenter: parent.horizontalCenter

width: 300

height: 300

radius: width / 2

color: "darkgoldenrod"

states: [

State

{

name: "On"

when: column.state == "Wait" || column.state == "WaitForStart"

PropertyChanges {

target: yellowLight

color: "yellow"

}

},

State

{

name: "Off"

when: !(column.state == "Wait" || column.state == "WaitForStart")

PropertyChanges {

target: yellowLight

color: "darkgoldenrod"

}

}

]

}

Rectangle

{

id: greenLight

anchors.horizontalCenter: parent.horizontalCenter

width: 300

height: 300

radius: width / 2

color: "darkgreen"

states: [

State

{

name: "On"

when: column.state == "Start"

PropertyChanges {

target: greenLight

color: "lime"

}

},

State

{

name: "Switch"

when: column.state == "WaitForStop"

},

State

{

name: "Off"

when: column.state == "Stop" || column.state == "Wait" || column.state == "WaitForStart"

PropertyChanges {

target: greenLight

color: "darkgreen"

}

}

]

transitions: [

Transition {

to: "Switch"

SequentialAnimation

{

loops: Animation.Infinite

ScriptAction

{

script: greenLight.color = "lime"

}

PauseAnimation {

duration: 500

}

ScriptAction

{

script: greenLight.color = "darkgreen"

}

PauseAnimation {

duration: 500

}

}

}

]

}

}

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

Page {

id: page

Column {

id: column

width: page.width

spacing: Theme.paddingLarge

states: [

State

{

name: "Stop"

},

State

{

name: "WaitForStart"

},

State

{

name: "Wait"

},

State

{

name: "Start"

},

State

{

name: "WaitForStop"

}

]

state: "Stop"

function switchState()

{

if (column.state == "Stop")

column.state = "WaitForStart"

else if (column.state == "WaitForStart")

column.state = "Start"

else if (column.state == "Start")

column.state = "WaitForStop"

else if (column.state == "WaitForStop")

column.state = "Wait"

else if (column.state == "Wait")

column.state = "Stop"

}

Timer

{

interval: 3500

running: true

repeat: true

onTriggered:

{

column.switchState()

}

}

Rectangle

{

id: redLight

anchors.horizontalCenter: parent.horizontalCenter

width: 300

height: 300

radius: width / 2

color: "maroon"

states: [

State

{

name: "On"

when: column.state == "Stop" || column.state == "WaitForStart"

PropertyChanges {

target: redLight

color: "red"

}

},

State

{

name: "Off"

when: !(column.state == "Stop" || column.state == "WaitForStart")

PropertyChanges {

target: redLight

color: "maroon"

}

}

]

}

Rectangle

{

id: yellowLight

anchors.horizontalCenter: parent.horizontalCenter

width: 300

height: 300

radius: width / 2

color: "darkgoldenrod"

states: [

State

{

name: "On"

when: column.state == "Wait" || column.state == "WaitForStart"

PropertyChanges {

target: yellowLight

color: "yellow"

}

},

State

{

name: "Off"

when: !(column.state == "Wait" || column.state == "WaitForStart")

PropertyChanges {

target: yellowLight

color: "darkgoldenrod"

}

}

]

}

Rectangle

{

id: greenLight

anchors.horizontalCenter: parent.horizontalCenter

width: 300

height: 300

radius: width / 2

color: "darkgreen"

states: [

State

{

name: "On"

when: column.state == "Start"

PropertyChanges {

target: greenLight

color: "lime"

}

},

State

{

name: "Switch"

when: column.state == "WaitForStop"

},

State

{

name: "Off"

when: column.state == "Stop" || column.state == "Wait" || column.state == "WaitForStart"

PropertyChanges {

target: greenLight

color: "darkgreen"

}

}

]

transitions: [

Transition {

to: "Switch"

SequentialAnimation

{

loops: Animation.Infinite

ScriptAction

{

script: greenLight.color = "lime"

}

PauseAnimation {

duration: 500

}

ScriptAction

{

script: greenLight.color = "darkgreen"

}

PauseAnimation {

duration: 500

}

}

}

]

}

}

// 2. Доработать задание 1 так, чтобы во время зеленого сигнала светофора

// из одного конца экрана в другой плавно двигалась иконка человечка.

Image

{

id: human

width: 300

height: 300

source: "qrc:///human-icon.png"

y: 850

states: [

State

{

name: "Stop"

when: !(column.state == "Start" || column.state == "WaitForStop")

PropertyChanges {

target: human

x: -200

}

},

State

{

name: "Start"

when: column.state == "Start" || column.state == "WaitForStop"

PropertyChanges

{

target: human

x: page.width

}

}

]

transitions: [

Transition

{

to: "Start"

NumberAnimation {

property: "x"

duration: 6500

}

}

]

}

}

1. Создадим приложение, отображающее строку текста вверху экрана. При нажатии на текст он будет плавно перемещаться вниз экрана, поворачиваться на 180 градусов и менять цвет. Когда нажатие прекращается, он будет так же плавно возвращаться в исходное положение.

Page {

id: page

// 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.

Column {

id: column

width: page.width

spacing: Theme.paddingLarge

Label

{

id: label

anchors.horizontalCenter: parent.horizontalCenter

text: "Hello world!"

font.pixelSize: Theme.fontSizeExtraLarge

state: "StartState"

property var startTime

property var currentY

property var currentRotation

property var currentColor

states: [

State

{

name: "Pressed"

PropertyChanges

{

target: label

y: 900

rotation: 180

color: "blue"

}

},

State

{

name: "Released"

PropertyChanges {

target: label

y: label.currentY

rotation: label.currentRotation

color: label.currentColor

}

},

State

{

name: "StartState"

PropertyChanges {

target: label

y: 0

rotation: 0

color: "red"

}

}

]

Timer

{

id: timer

interval: 0

repeat: false

onTriggered: {

console.log("Here");

if(label.startTime !== undefined)

returnAnimation.returnTime = new Date().getTime() - label.startTime;

console.log("Return time = " + returnAnimation.returnTime);

console.log("Change state to start")

label.state = "StartState"

}

}

transitions: [

Transition

{

to: "Pressed"

SequentialAnimation

{

ScriptAction

{

script: label.startTime = new Date().getTime()

}

NumberAnimation

{

properties: "y,rotation"

duration: 5000

}

}

ColorAnimation {

duration: 5000

}

onRunningChanged: {

if (!running)

{

label.currentY = label.y

label.currentRotation = label.rotation

label.currentColor = label.color

}

}

},

Transition {

to: "Released"

SequentialAnimation

{

ScriptAction

{

script: timer.start()

}

}

},

Transition

{

id: returnAnimation

property int returnTime

to: "StartState"

NumberAnimation

{

properties: "y,rotation"

duration: returnAnimation.returnTime

}

ColorAnimation {

duration: returnAnimation.returnTime

}

}

]

MouseArea

{

anchors.fill: label

onPressed: label.state = "Pressed"

onReleased: label.state = "Released"

}

}

}

}

1. Выделим сигналы светофора из задания 1 в отдельный компонент и использовать его .

Для этого достаточно вынести Column в задании светофора вынести в отдельный файл – внутрь Item. И добавить этому Item’у ширину, например 300. Дублировать код здесь не будем.

Впоследствии использовать этот компонент можно будет так:

Page {

TrafficLight {

anchors.horizontalCenter: parent.horizontalCenter

}

}

1. Создадим QML компонент со свойством по умолчанию, который берет значение свойства text любого объявленного внутри него объекта и создает Button с тем же текстом. Добавим возможность задавать цвет кнопки при объявлении компонента.

Button

{

default property var someText

text: someText.text

}

1. Создадим приложение-секундомер. На экране будут отображаться значения часов, минут и секунд. Секундомер запускается по сигналу кнопки, при повторном нажатии секундомер останавливается. Для отображения часов, минут и секунд используем собственные QML компоненты.

Item {

id: clock

width: label.width

height: label.height

property int seconds: 0

Timer

{

id: timer

interval: 1000

repeat: true

onTriggered: seconds++

}

function start()

{

clock.seconds = 0;

timer.start();

}

function stop()

{

timer.stop();

}

function prependWithZeros(num)

{

return num > 9 ? num : "0" + num

}

Label

{

id: label

text: prependWithZeros(Math.floor(seconds / 3600)) + ":" + prependWithZeros(Math.floor(seconds / 60) % 60) + ":" + prependWithZeros(seconds % 60)

anchors.horizontalCenter: parent.horizontalCenter

}

}

1. Добавим обработчик сигналов PageStack, подсчитывающий количество добавленных и удаленных страниц в PageStack.

property int prevDepth

property int pushPages : 0

property int popPages : 0

function onDepthChangedImpl()

{

if (prevDepth === undefined)

pushPages++;

else

{

if(prevDepth > pageStack.depth)

{

popPages++;

}

else

{

pushPages++;

}

}

prevDepth = pageStack.depth

console.log("Added pages = " + pushPages);

console.log("Removed pages = " + popPages);

}

Component.onCompleted:

{

pageStack.onDepthChanged.connect(onDepthChangedImpl)

}

1. **Заключение.**

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

**4. Литература.**

1. Документация Qt: сайт. — URL: https://doc.qt.io (дата обращения: 01.03.2022). — Текст: электронный.