**2 ЛАБОРАТОРНАЯ РАБОТА №2**

**«Разработка мобильных приложений для ОС Aurora»**

**2.1 Цель работы**

Ознакомиться с реализацией перемещения объектов в операционной системе Aurora.

* 1. **Вариант задания**

1. Создать прямоугольник, чтобы при нажатии на него он передвигался в нижнюю часть экрана.
2. Создать три прямоугольника и применить к ним анимацию вращения без остановок, а также задать углы вращения от 0 до 90, 180 и 360 соответственно.
3. Получить и отобразить Name, Nominal и ParentCode в виде списка из ресурса ЦБ РФ по адресу.

**2.3 Ход выполнения работы**

2.3.1 В начале выполнения лабораторной работы была написана программа, которая перемещает созданный прямоугольник по нажатию на него. Код программы представлен в листинге 2.1.

Листинг 2.1 – Текст первой программы

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

objectName: "mainPage"

allowedOrientations: Orientation.All

Rectangle{

id: rect

width:400

height: 100

color:"lightpink"

MouseArea{

id:mouseArea

anchors.fill: parent

onClicked: rect.state==='clicked'?rect.state="":rect.state='clicked';

}

states:[

State {

name: "clicked"

PropertyChanges {target: rect; x:300; y:1000}

}

]

}

}

Кнопка до перемещения показана на рисунке 2.1, а после нажатия на рисунке 2.2.



Рисунок 2.1 – Внешний вид приложения до нажатия



Рисунок 2.2 – Внешний вид приложения после нажатия

2.3.2 Далее была написана программа, в которой вращаются 3 прямоугольника по заданным для каждого параметрам, код программы отражен в листинге 2.2.

Листинг 2.2 – Текст второй программы

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

objectName: "mainPage"

allowedOrientations: Orientation.All

Column{

anchors.centerIn: parent

spacing: 300

Rectangle{

id:rect1

width:200

height:100

color:"lightpink"

RotationAnimation on rotation {

loops: Animation.Infinite

from: 0

to: 90

}

}

Rectangle{

id:rect2

width:200

height:100

color:"lightyellow"

RotationAnimation on rotation {

loops: Animation.Infinite

from: 0

to: 180

}

}

Rectangle{

id:rect3

width:200

height:100

color:"lightgreen"

RotationAnimation on rotation {

loops: Animation.Infinite

from: 0

to: 30

}

}

}

}

Внешний вид полученного в этом пункте приложения показан на рисунке 2.3.

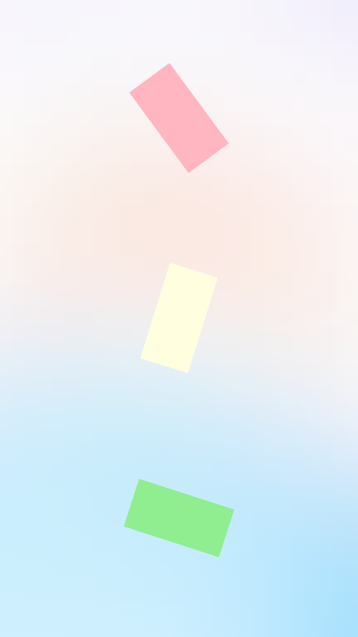


Рисунок 2.3 – Внешний вид второго приложения

2.3.3 Затем была написана программа, которая загружает в приложения данные из xml-файла, код программы отражен в листинге 2.3.

Листинг 2.3 – Текст третьей программы

import QtQuick 2.0

import Sailfish.Silica 1.0

import QtQuick.XmlListModel 2.0

Page {

objectName: "mainPage"

allowedOrientations: Orientation.All

XmlListModel{

id: model

source:"https://cbr.ru/scripts/XML\_val.asp?d=0"

query: "/Valuta/Item"

XmlRole{name: "name"; query: "Name/string()"}

XmlRole{name: "nominal"; query: "Nominal/string()"}

XmlRole{name: "parentCode"; query: "ParentCode/string()"}

onStatusChanged: {

if(status === XmlListModel.Ready){

console.log("Got", count, "MakeLib")

}else if (status === XmlListModel.Error){

console.error("Error loading Market Libs:", errorString());

}

}

}

ListView{

anchors.fill: parent

model:model

spacing: 100

delegate: Item {

width: parent.width

height: 60

Column{

Text { text: "Name:"+name }

Text { text: "Nominal:"+nominal }

Text { text: "ParentCode:"+parentCode }

}

}

}

}

Внешний вид полученного в этом пункте приложения показан на рисунке 2.4.

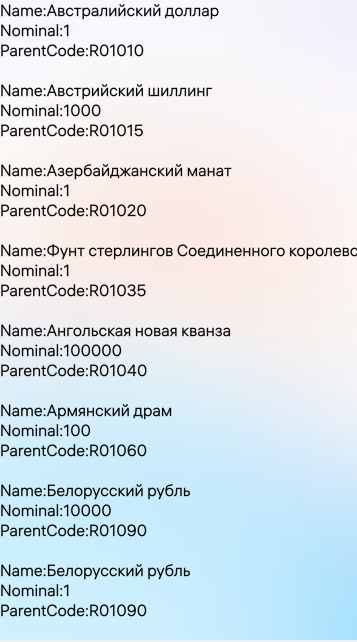


Рисунок 1.4 – Внешний вид третьего приложения

**Выводы**

В ходе лабораторной работы было проведено ознакомление с инструментами для создания мобильных приложений под ОС Aurora. Было написано 3 программы, выполняющие базовые действия, такие как нажатия на кнопки, ввод текста, вставка картинок. В конце выполнения лабораторной работы был написан отчет.