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 проектной работе

Выполнила:
студентка группы 382006-2
Кулёва Анна Андреевна

Проверил:
Карчков Денис Александрович

Нижний Новгород
2023

Содержание

1. Цель проектной работы

Целю данной проектной работы является закрепление всех навыков работы в Qt Creator и эмуляторе, подтверждение полученных навыков работы с пользовательским интерфейсом, анимацией и QML компонентами.

2. Постановка задачи

Создать приложение, аналог мобильной игры “Fruit Ninja”, которое будет соответствовать следующим требованиям:

1. Приложение должно содержать кнопку, запускающую игру.
2. В приложении должны быть таймер, показывающий время, в течение которого продолжается игра, и счётчик очков, заработанных игроком.
3. Требуется создать отдельную компоненту QML, хранящую объект-апельсин.
4. Фруктовые объекты должны генерироваться каждую секунду игры в рандомной части снизу экрана.
5. Требуется реализовать анимацию подкидывания вверх и падения апельсинов.
6. При нажатии на апельсин, игроку должно присваиваться 1 очко, а сам апельсин должен исчезнуть.
7. Игра должна заканчиваться, если игрок позволил упасть 3 апельсинам. При окончании игры должно появляться окно с итоговым счётом игрока.

3. Руководство пользователя

При запуске программы пользователь увидит стартовую страницу с кнопкой «Начать игру», нажав на которую перейдёт к странице с самой игрой (рисунок 1).

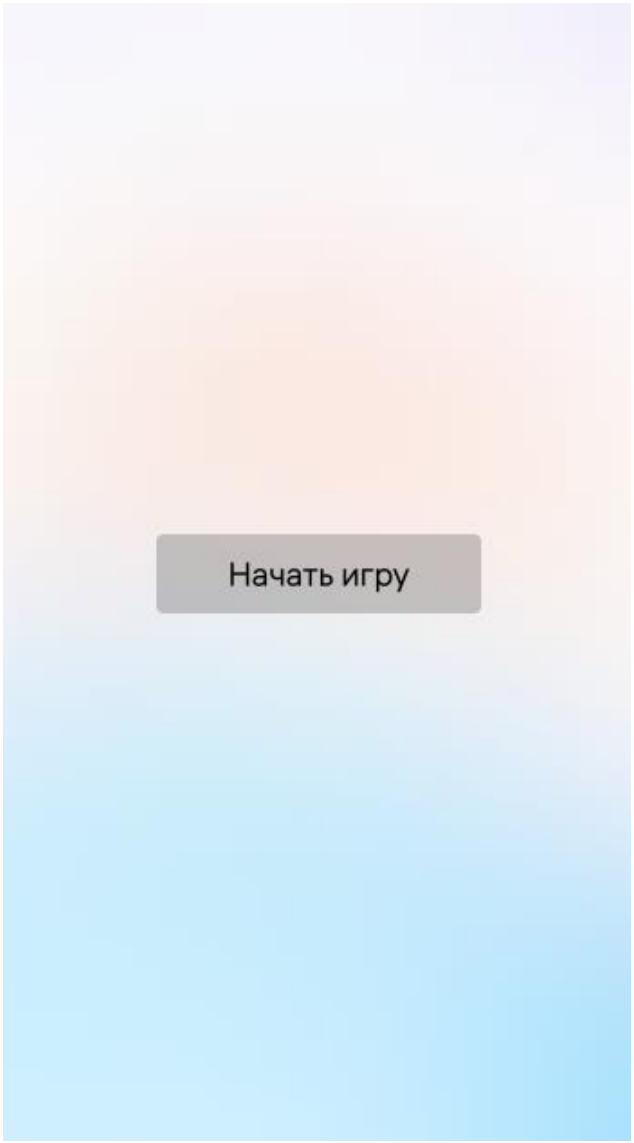

Рисунок 1. Стартовая страница

Сама игра представляет собой поле, из нижней части которого каждую секунду появляются апельсины, по которым игрок должен успеть нажать. В этом случае игрок зарабатывает очко, а апельсин исчезает. Страница с игрой в верхнем правом углу содержит таймер, показывающий, сколько секунд продолжается игра, а в верхнем левом углу – счётчик очков, заработанных игроком (рисунок 2).

Рисунок 2. Игра в апельсины

Если в ходе игры игрок не успеет кликнуть на 3 апельсина, и они упадут вниз, то игра закончится, и на экране появится финальный счёт игрока (рисунок 3).

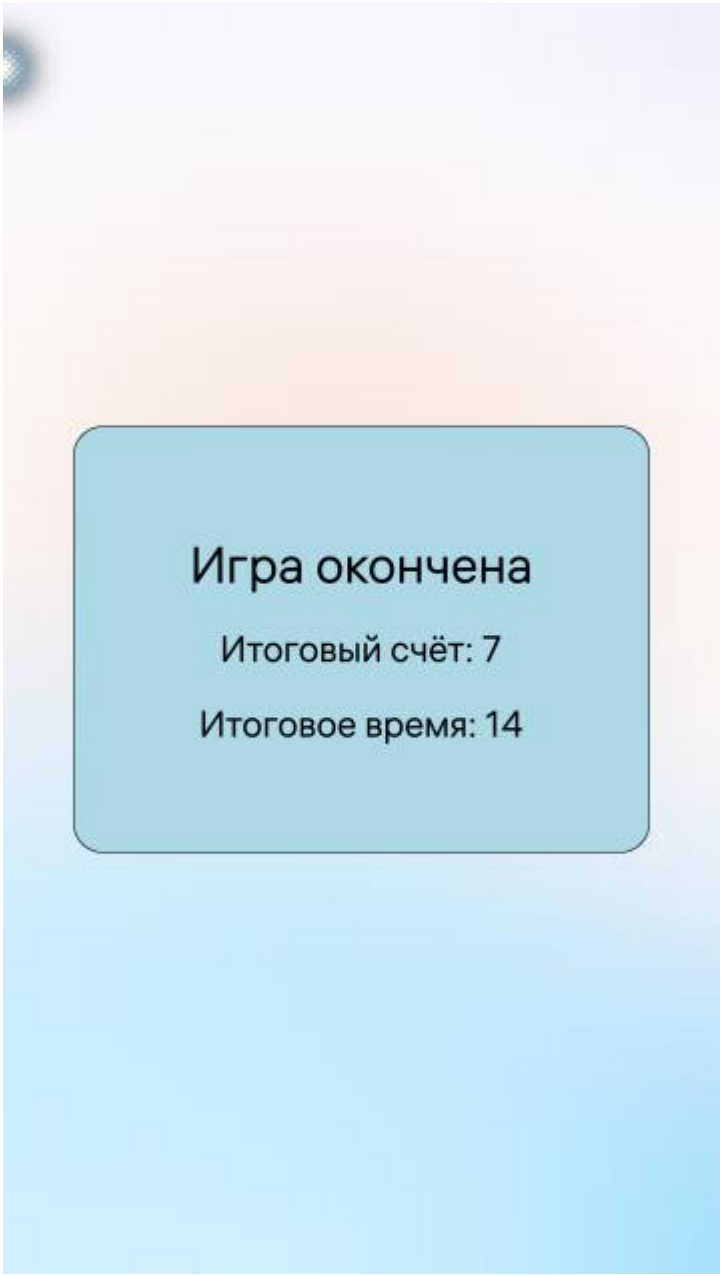

Рисунок 3. Финальный счёт

4. Руководство программиста

Программа реализована на языке программирования QML.

1. В начале работы проект загружает стартовую страницу :

initialPage: *Qt*.resolvedUrl("pages/StartPage.qml")

На ней расположена кнопка “Начать игру”, которая по клику отправляет игрока на страницу с самой игрой:

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

2. Проект содержит отдельный QML компонент Orange.qml, описывающий ключевой объект игры – апельсин, который игрок должен поймать мышкой.

Апельсин реализован с помощью Rectangle заданного размера: Screen.height/10 – одна десятая высоты экрана. Его изначальное положение определяется с помощью функции *Math*.random(:

```
x: Math.random() * (sizex - 2 * tgt.width)
y: sizey
```

Внутри Rectangle содержится изображение Image, отвечающее за внешний вид объекта.

```
Image {
    height: sizey/10
    width: sizey/10
    source: "orange.png"
    anchors.centerIn: parent
}
```

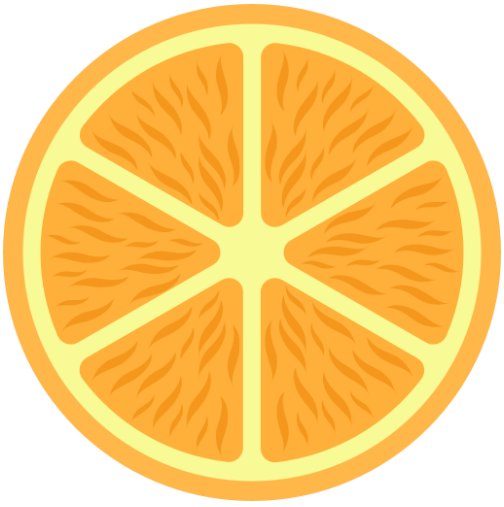

Рисунок 4. orange.png

Также Rectangle содержит кнопку, нажатие на которую увеличивает счёт на единицу и уничтожает объект:

```
onClicked: {
        score++
        tgt.destroy()
    }
```

Внутри объекта описаны две анимации SequentialAnimation и RotationAnimation.

- SequentialAnimation описывает последовательное движение апельсина до некоторой рандомной точки *property int* targetY: *Math*.random() * (sizey / 4) + sizey / 4 и назад до нижней части экрана. Движение происходит по координате y
- RotationAnimation описывает вращение апельсина от 0 до 180 градусов. Вращение происходит параллельно с перемещением.

При перемещении отслеживается координата объекта по y, и, если апельсин пересекает нижнюю границу, то начисляются очки падения `lose`:

```
onYChanged: {
        if (tgt.y == sizey + 2) {
            tgt.destroy()
            lose++
        }
    }
```

3. Файл, описывающий главную страницу игры MainPage.qml содержин несколько переменных типа int:

- *property int time* – переменная, описывающая время, в течение которого длится игра
- *property int score* – переменная, описывающая счёт игрока
- *property int lose* – переменная, описывающая количество падений фруктов
- *property int sizex*: Screen.width – переменная, описывающая ширину экрана
- *property int sizey*: Screen.height – переменная, описывающая высоту экрана

В файле создан объект `Timer` с интервалом 1 секунда. При изменении времени он прибавляет к значению переменной time единицу (каждую секунду), а также отслеживает значение переменной lose. Если количество упавших апельсинов будет превышать 3, тогда таймер останавливается, очки игрока выводятся на экран (*end_rect*.visible = true). Также, если количество очков падения не превышает 3, то каждую секунду создаётся новый апельсин с помощью функции *createOrange*():

```
Timer {
        id: timer
        interval: 1000
        running: true; repeat: true
        onTriggered: {
            parent.time = parent.time + 1
            if (lose >= 3)
            {
```

```
                end_rect.visible = true
                scoreLabel.visible = false
                timeLabel.visible = false
                timer.running = false
            }
            createOrange()
        }
        }
```

Функция, отвечающая за создание объекта Orange.qml приведена ниже:

```
    function createOrange() {
                var orangeComponent = Qt.createComponent("Orange.qml")
                if (orangeComponent.status === Component.Ready) {
                    var orange =
orangeComponent.createObject(orangeContainer)
                    orange.visible = true
                }
            }
            Item {
                id: orangeContainer
        }
```

В верхней части экрана располагаются 2 объекта Label, выводящие на экран счёт игрока и время, прошедшее от начала игры.

При завершении игры появляется табличка с итоговым счётом игрока. Она реализована с помощью объекта Rectangle, у которого поставлено visible: false. Свойство visible меняется в таймере. Rectangle содержит Column с тремя объектами Label: первая надпись сообщает об окончании игры (text: "Игра окончена"), вторая надпись выводит счёт (text: "Итоговый счёт: " + *score*), третья надпись выводит время (text: "Итоговое время: " + *time*).

Заключение

В данной лабораторной работе я закрепила мои навыки работы в Qt Creator и эмуляторе, а также навыки работы с пользовательским интерфейсом, анимацией и QML компонентами.

В результате проектной работы было создано приложения, соответствующее всем поставленным требованиям.

Приложение

StartPage.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    property int sizex: Screen.width
    property int sizey: Screen.height
    Button {
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.centerIn: parent
        width: sizex / 2
        text: "Начать игру"
        onClicked: pageStack.push(Qt.resolvedUrl("MainPage.qml"))
    }

}
```

Orange.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Item {
    id: orange
    Rectangle {
        id: tgt
        width: sizey / 10
        height: sizey / 10
        x: Math.random() * (sizex - 2 * tgt.width)
        y: sizey
        radius: width / 2

        property int startY: sizey
        property int targetY: Math.random() * (sizey / 4) + sizey / 4

        Image {
            height: sizey/10
            width: sizey/10
            source: "orange.png"
            anchors.centerIn: parent
        }

        Button {
            anchors.centerIn: parent
            width: tgt.width
            height: tgt.height
            opacity: 0
            onClicked: {
                score++
                tgt.destroy()
            }
        }

        SequentialAnimation {
            running: true
```

```
            NumberAnimation { target: tgt; property: "y"; to: tgt.targetY;
duration: 1500 }
            NumberAnimation { target: tgt; property: "y"; to: tgt.startY + 2;
duration: 1500 }
        }

        RotationAnimation on rotation {
             loops: Animation.Infinite
             duration: 1000
             from: 0
             to: 180
         }

        onYChanged: {
                if (tgt.y == sizey + 2) {
                    tgt.destroy()
                    lose++
                    //console.log(lose)
                }
            }

    }
}
```

MainPage.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    property int time: 0
    property int score: 0
    property int lose: 0
    property int sizex: Screen.width
    property int sizey: Screen.height

    Timer {
        id: timer
        interval: 1000
        running: true; repeat: true
        onTriggered: {
            parent.time = parent.time + 1
            if (lose >= 3)
            {
                end_rect.visible = true
                scoreLabel.visible = false
                timeLabel.visible = false
                timer.running = false
            }
            createOrange()
        }
    }

                function createOrange() {
                    var orangeComponent = Qt.createComponent("Orange.qml")
                    if (orangeComponent.status === Component.Ready) {
                        var orange =
orangeComponent.createObject(orangeContainer)
                        orange.visible = true
                    }
                }
                Item {
                    id: orangeContainer
```

```
            }

        Label {
            id: scoreLabel
            visible: true
            x: 50
            anchors.top: parent.top
            text: "Счёт: " + score.toString()
        }
        Label {
            id: timeLabel
            visible: true
            x: sizex - 200
            anchors.top: parent.top
            text: "Время: " + time.toString()
        }


        Rectangle {
            id: end_rect
            visible: false
            width: 4*sizex/5
            height: 1*sizey/3
            color: "lightblue"
            border.color: "black"
            radius: 30
            anchors.centerIn: parent

        Column {
                id: column
                visible: true
                width: parent.width
                anchors.centerIn: parent
                spacing: 30
                Label {
                    anchors.horizontalCenter: parent.horizontalCenter
                    id: label_end;
                    text: "Игра окончена"
                    font.pixelSize: Theme.fontSizeExtraLarge

                }
                    Label {
                        anchors.horizontalCenter: parent.horizontalCenter
                        id: label_score;
                        text: "Итоговый счёт: " + score
                    }
                    Label {
                        anchors.horizontalCenter: parent.horizontalCenter
                        id: label_time;
                        text: "Итоговое время: " + time
                    }
                }
        }

}
```