Стандартные элементы пользовательского интерфейса (UI)

Элементы UI

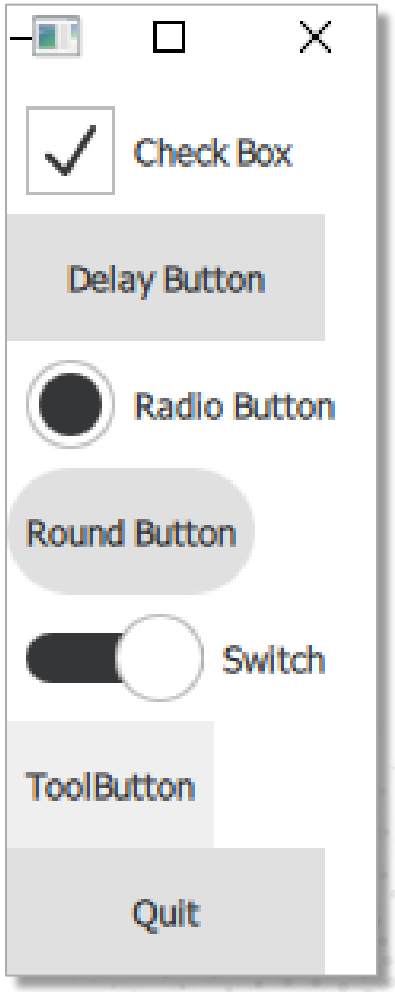
Технология ***Qt Quick*** предоставляет ряд элементов, для создания пользовательского интерфейса. Эти элементы способны полностью заменить виджеты *Qt*. Чтобы использовать готовые элементы, необходимо включить в *QML-файл* модуль ***QtQuick.Controls***.

```
import QtQuick 2.15
import QtQuick.Controls 2.15

ApplicationWindow {
    width: buttons.width
    height: buttons.height
    visible: true
    title: "Buttons"
    Column {
        id: buttons
        CheckBox  { text: "Check Box"}
        DelayButton { text: "Delay Button"}
        RadioButton { text: "Radio Button"}
        RoundButton { text: "Round Button"}
        Switch { text: "Switch"}
        ToolButton { text: "Tool Button"}
        Button { text: "Quit"
                 onClicked: Qt.quit()
                }
    }
}
```

Элементы UI. Кнопки

- ***кнопка флажка*** - она обычно имеет два состояния: включено или выключено;
- ***кнопка задержки***. Процесс задержки показывается при нажатии на саму кнопку, и при достижении конечного значения задержки кнопка считается нажатой (сгенерируется сигнал ***activated()***);
- ***кнопка переключателя***. Группы из этих кнопок представляют собой взаимоисключающие состояния, поэтому в программе их должно быть минимум две в группе;
- ***закругленная кнопку*** - представляет собой обычную кнопку нажатия с закругленными углами;
- ***кнопка выключателя*** - ее принято использовать на мобильных устройствах вместо кнопки флажка;
- ***кнопка инструментов*** - эта кнопка специально предназначена для расположения на панели инструментов;
- ***обычная кнопка нажатия*** - Нажатие на нее осуществляет выход из приложения.



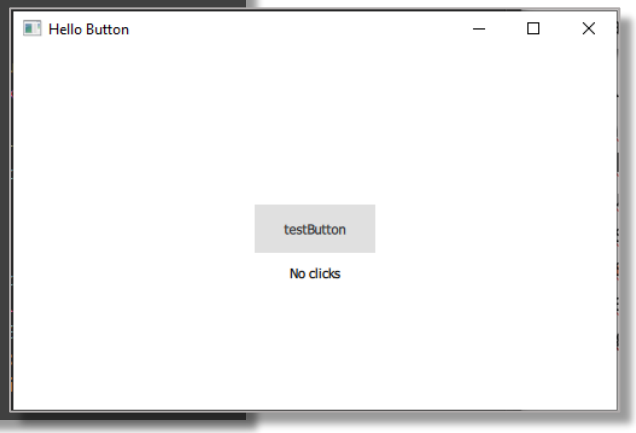
Свойства элемента UI

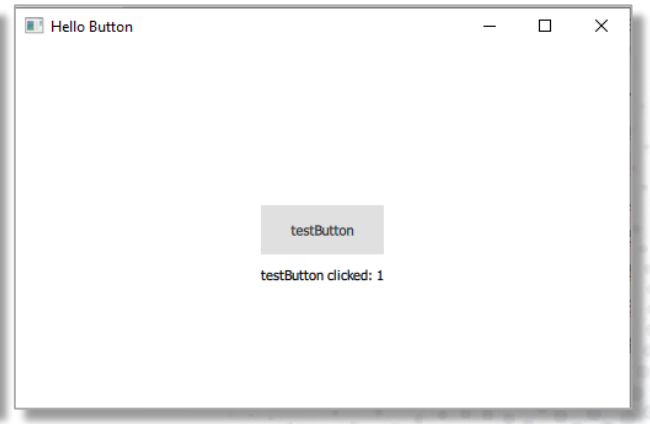
Button наследует от ***AbstractButton***, поэтому именно здесь определена и описана большая часть свойств и методов.

```
Button {
    id: testButton
    anchors.centerIn: parent
    text: qsTr("testButton")
    property var clickCounter: 0
    onClicked: {
        clickCounter++;
        testButtonClicks.text = qsTr("testButton clicked: ") + clickCounter;
    }
}
Text {
    id: testButtonClicks
    anchors.horizontalCenter: parent.horizontalCenter
    anchors.top: testButton.bottom
    anchors.topMargin: 10
    text: qsTr("No clicks")
}
```

Обработчики сигналов:

- ***onCanceled*** – «отпущено»+ мышь вне кнопки
- ***onClicked***- клик на кнопку
- ***onDoubleClicked*** – двойной клик
- ***onPressAndHold***- нажатие и удержание
- **onPressed** – нажатие кнопки
- ***onReleased*** – отпускание кнопки
- ***onToggled*** - переключение





Кастомизация элемента UI

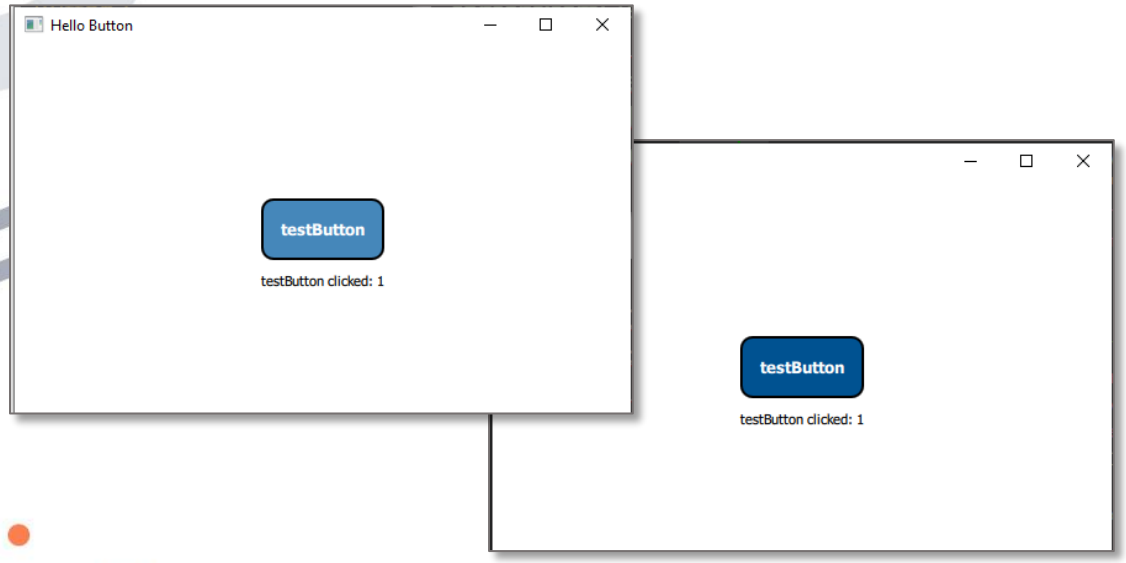
У кнопки есть два объекта для настройки стиля - **прямоугольная область** и **текст**, заключенный внутри нее. Поэтому QML предоставляет нам два соответствующих свойства:

- ***background***
- ***contentItem***

```
Button {
        id: testButton
        anchors.centerIn: parent
        text: qsTr("testButton")
    . . .
}
```

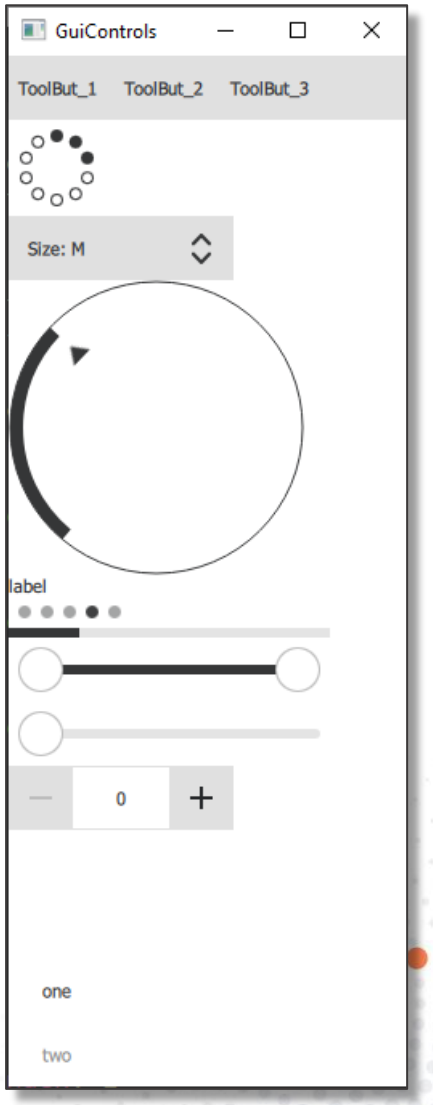
```
contentItem: Text {
        text: testButton.text;  color: "#ffffff"
        horizontalAlignment: Text.AlignHCenter
        verticalAlignment: Text.AlignVCenter
        font.pointSize: 10;  font.bold: true
    }
```

```
background: Rectangle {
        property var normalColor: "#005291"
        property var hoveredColor: "#4587ba"
        property var pressedColor: "#002948"
        implicitWidth: 100; implicitHeight: 50
        color: testButton.pressed ? pressedColor :
testButton.hovered ? hoveredColor : normalColor
        radius: 10; border.width: 2.0;  border.color: "#000000"
}
```



Элементы UI. Другие элементы управления

#	Наименование	Описание
1	**ToolBar**	Элемент панели инструментов. Он предназначен для размещения элементов в нижней либо в верхней зоне главного окна приложения
2	**BusyIndicator**	Элемент для отображения состояния занятости приложения. Используется когда невозможно определить окончание операции
3	**ComboBox**	Элемент выпадающего списка.
4	**Dial**	Элемент установщик. Для установки значений его требуется вращать. Он не задуман для установки точных значений
5	**Label**	Элемент надписи. Представляет собой текстовое поле, которое служит для отображения поясняющего текст
6	**PageIndicator**	Элемент индикатора страниц. Информирует об индексе отображаемой в текущий момент страницы
7	**ProgressBar**	Элемент индикации выполнения. Демонстрирует процесс выполнения операции тем, что заполняет область слева направо. Полное заполнение области сигнализирует о завершении операции
8	**RangeSlider**	Элемент для установки диапазона значений. Элемент содержит два элемента ползунка
9	**Slider**	Элемент ползунок. значений, Используется которые не требуют для установки высокой в заданном точности
10	**SpinBox**	Элемент счетчика. Предоставляет пользователю доступ к ограниченному диапазону чисел. Используется для установки значений с высокой точностью
11	**Tumbler**	Элемент предоставления выбора в виде столбцов. Может состоять из одного или нескольких столбцов, при испольвании которых нужные значения помещаются в средний ряд

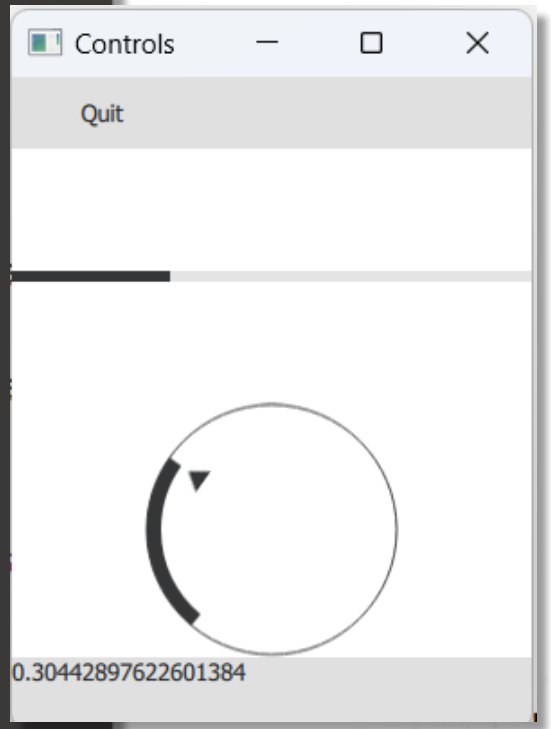


Пример использования элементов UI

```
import QtQuick 2.15
import QtQuick.Controls 2.15

ApplicationWindow {
    visible: true; width: 200;  height: 200
    title: "Controls"
    header: ToolBar {
            Button {
                text: "Quit"
                onClicked: Qt.quit();
            }
    }
 . . .
```

```
. . .
footer: ToolBar {
    id: statusbar
    Label {
        id: statuslbl
    }
        ProgressBar {
            x:0;  y: 0
            width: parent.width
            height: parent.height / 2
            value: slider.value
        }
        Dial {
            id: slider
            x: 0; y: parent.height / 2
            width: parent.width
            height: parent.height / 2
            value: 0.75;  stepSize: 0.1
            onValueChanged: statuslbl.text = slider.value
        }
    }
}
```

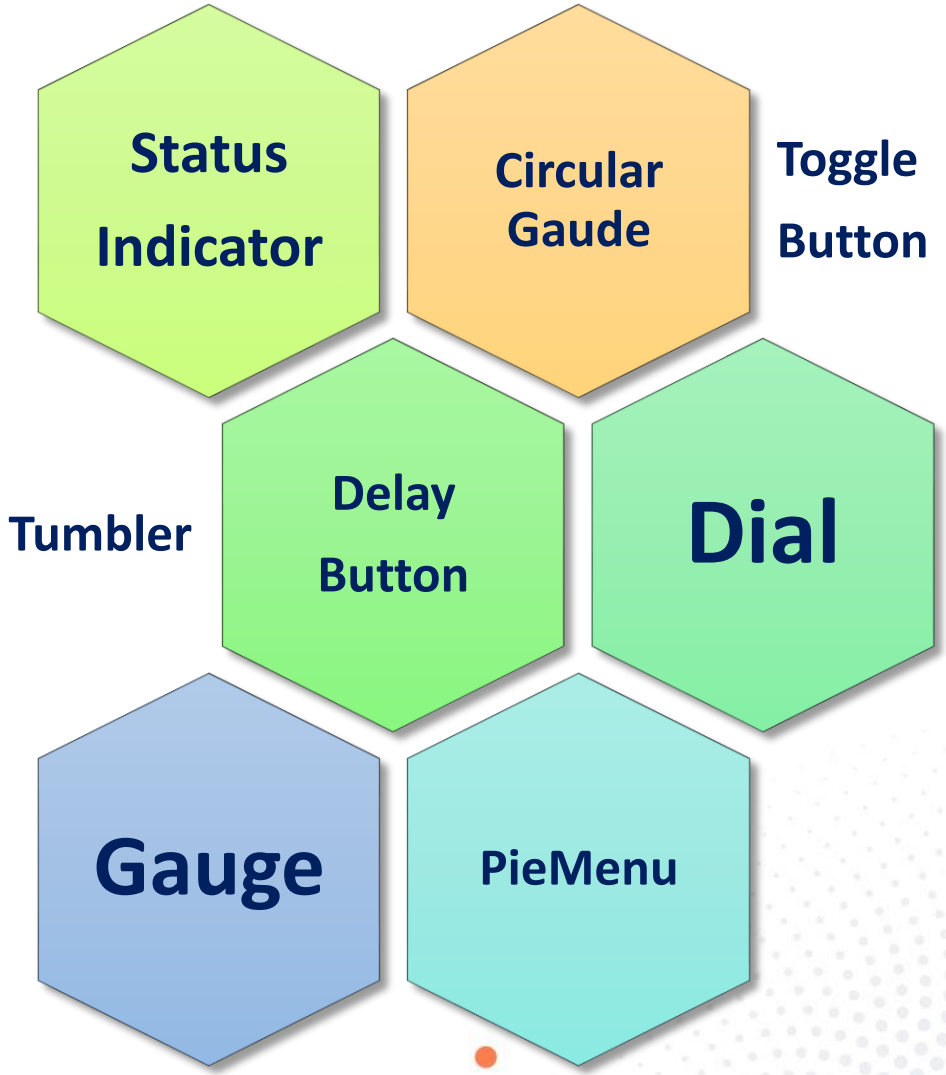


Дополнительные элементы UI

Имеется еще один дополнительный модуль с элементами управления. Этот модуль называется ***QtQuick.Extras***. Их можно использовать совместно с основными элементами управления. (если он установлен в вашей IDE)

*Чтобы избежать конфликтов с элементами, находящимися в обоих модулях с одинаковыми именами, - например: ***Dial*** и т.д., рекомендуется снабдить модуль отдельным идентификатором и затем обращаться к его элементам с помощью идентификатора модуля. Например:

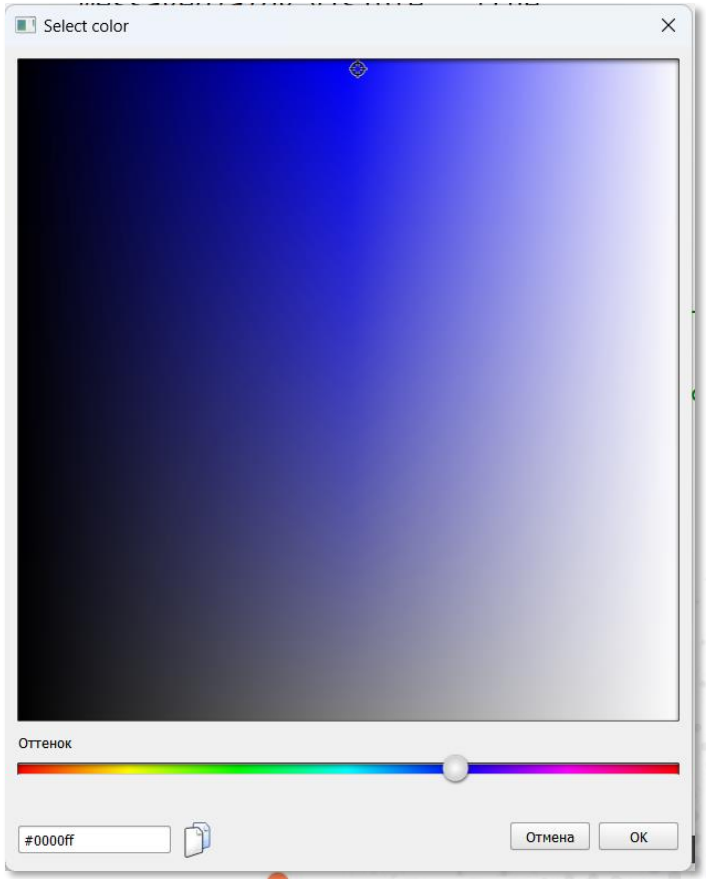
```
import QtQuick.Extras 1.4 as QQE
. . .
QQE.DelayButton { text: "Delay Button "}
```



Диалоговые окна. Выбор цвета

Технология Qt Quick предоставляет стандартные диалоговые окна для задания цвета: ColorDialog. Для использования диалоговых окон необходимо включить модуль QtQuick. Dialogs.

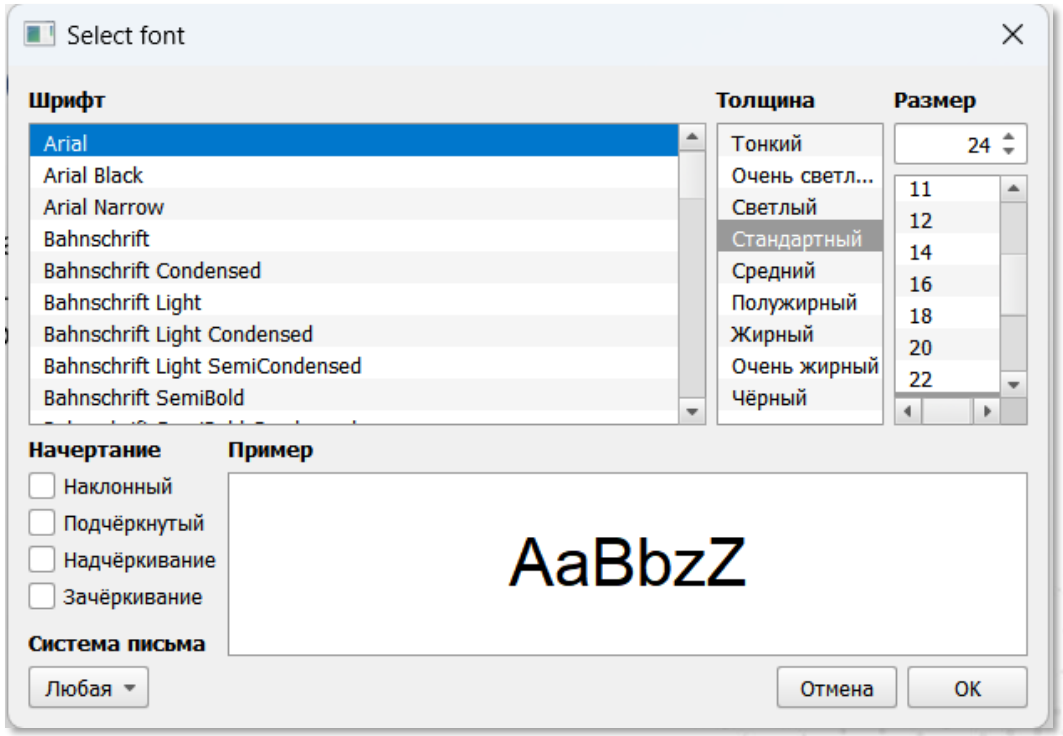
```
ColorDialog {
    id: colorDialog
    visible: false
    modality: Qt.WindowModal
    title: "Select color"
    color: "blue"
    onAccepted: {
      messageDialog.informativeText = "Selected color: " + color
      messageDialog.visible = true
    }
  }
```



Диалоговые окна. Выбор шрифта

Технология ***Qt Quick*** предоставляет стандартные диалоговые окна установки шрифта: ***FontDialog*** для использования диалоговых окон необходимо включить модуль ***QtQuick.Dialogs XX***.

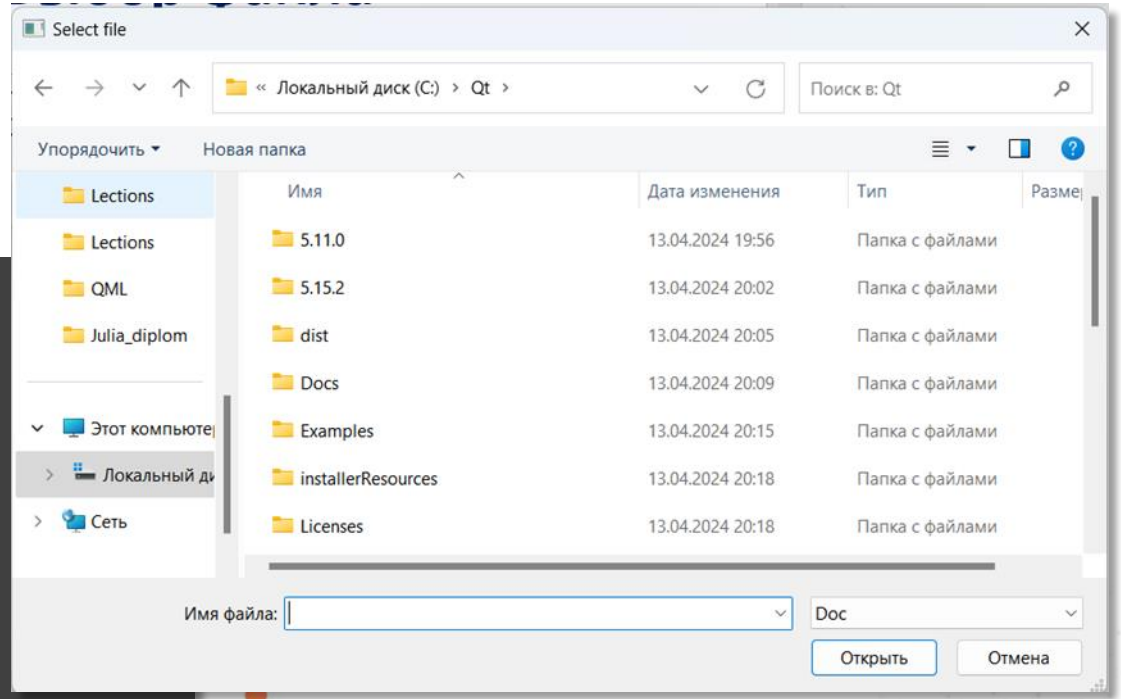
```
FontDialog {
    id: fontDialog
    visible: false
    modality: Qt.WindowModal
    title: "Select font"
    onAccepted: {
        messageDialog.informativeText = "Selected font: " + font
        messageDialog.visible = true
    }
}
```



Диалоговые окна. Выбор файла

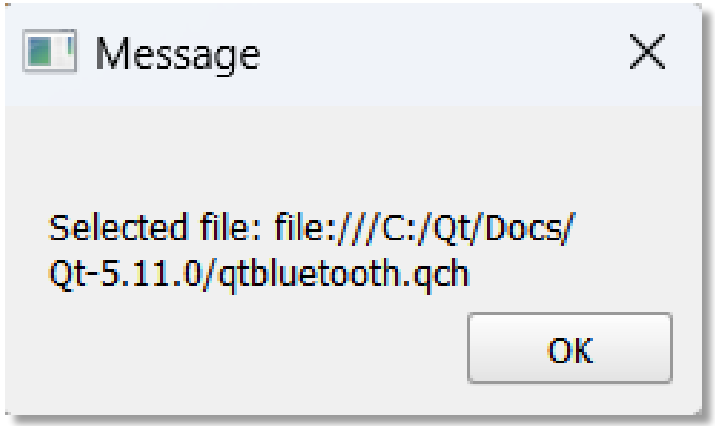
Технология ***Qt Quick*** предоставляет стандартные диалоговые окна для открытия файлов: ***FileDialog,*** Для использования диалоговых окон необходимо включить модуль ***QtQuick.Dialogs XX***.

```
FileDialog {
        id: fileDialog
        visible: false
        modality: Qt.WindowModal
        title: "Select file"
        folder: "file:///C:/Qt/"
        nameFilters: ["Doc (* .txt * .html) ", "All files (*) "]
        onAccepted: {
            messageDialog.informativeText = "Selected file: " + fileUrls
            messageDialog.visible = true
        }
    }
```



Диалоговые окна. Вывод сообщения

```
MessageDialog {
    id: messageDialog
    visible: false
    modality: Qt.NonModal
    title: "Message"
}
```



Практическое задание

```
import QtQuick 2.15
import QtQuick.Controls 2.15
import QtQuick.Dialogs 1.3
import QtQuick.Controls.Styles 1.4
```

Создайте приложение открывающее диалоговые окна.

```
ApplicationWindow {
   width: buttons.width;   height: buttons.height;  visible: true
   title: "Dialogs"
   Row {   // горизонтальное позицианирование
     id: buttons
     Button { text: "Quit"
                  onClicked: Qt.quit() }
     Button {  background: Rectangle {color: "lightgreen"}
       text: "File Dialog"
       onClicked: {fileDialog.visible = true}
      }
     Button {  background:Rectangle{color: "lightyellow"}
       text: "Color Dialog"
       onClicked: {colorDialog.visible = true}
      }
     Button {
       background:Rectangle{color: "lightblue"}
       text: "Font Dialog"
       onClicked: {fontDialog.visible = true}
      }
   }
   ColorDialog { . . .}
. . .
}
```

Домашка #7

Создать небольшое приложение и выполнить кастомизацию элементов.
(пример:
https://microtechnics.ru/qt-qml-dialog-svojstva-sobytiya-i-signaly-kastomizacziya-stilya/
)