## Чтение текстового файла

QFile file("file.txt"); //Создали объект класса для проведения операции с файлом "file.txt";

if (file.open (QIODevice::ReadOnly)) //Если файл открыт (с функцией «Только чтение»)

{

QTextStream stream(&file); //Создали объект поток для работы с файлом

QString str;

while (!stream.atEnd()) //Пока файл не кончился

{

str = stream.readLine(); //Считали одну строку

qDebug() << str;

}

if (stream.status() != QTextStream::Ok)

{

qDebug() << "Ошибка чтения файла";

}

file.close(); //Закрыли файл

}

## Запись в файл

QFile file("file.txt");

QString str = "This is a test";

if (file.open(QIODevice::WriteOnly))

{

QTextStream stream(&file);

stream << str;

file.close();

if (stream.status() != QTextStream::Ok)

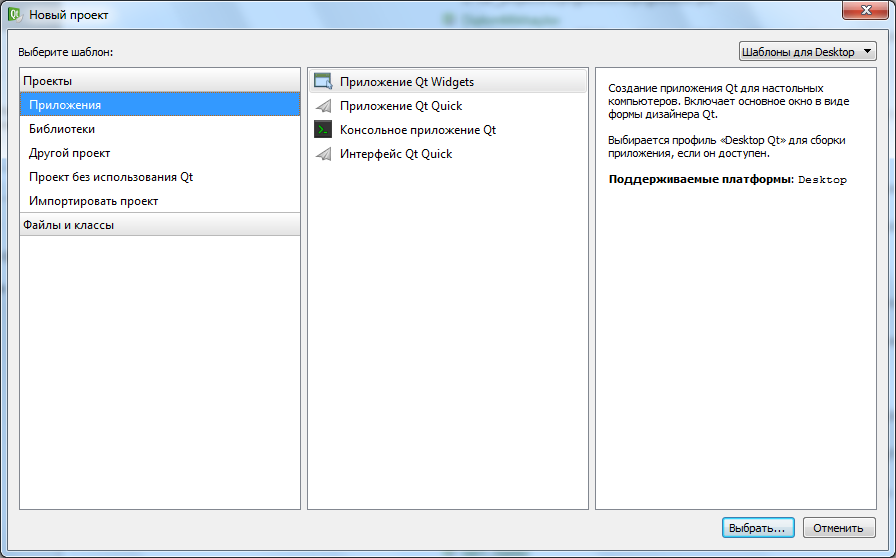
{

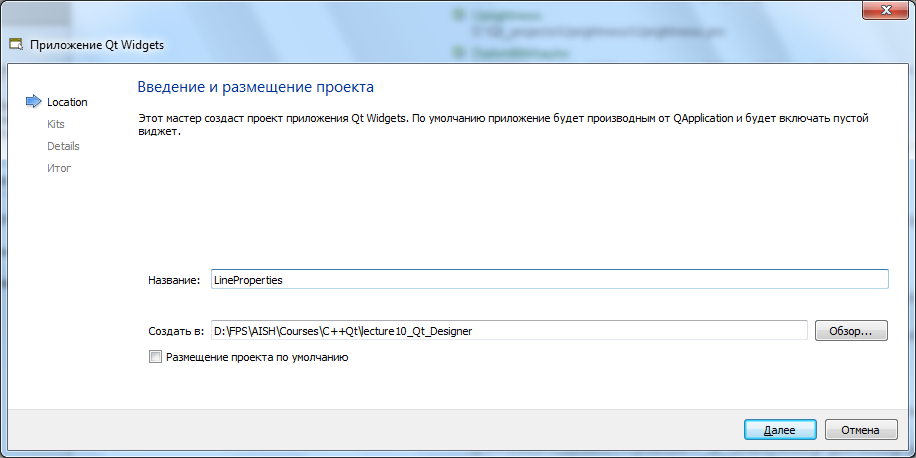
qDebug() << "Ошибка записи файла";

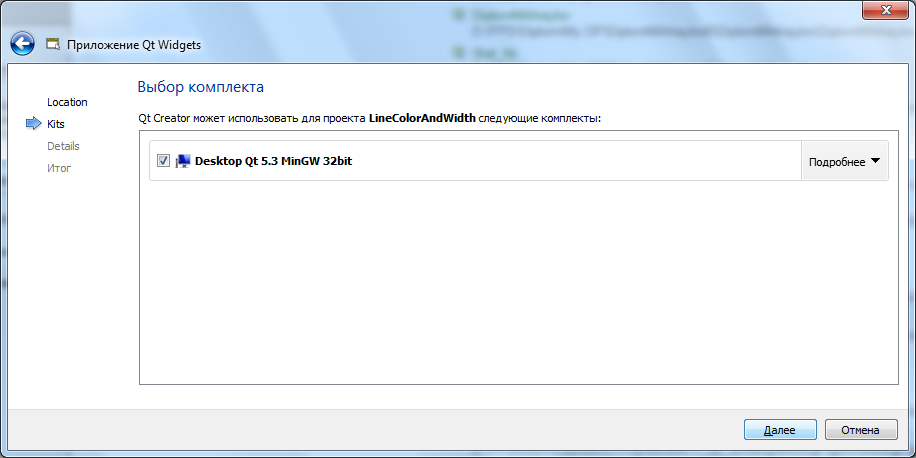
}

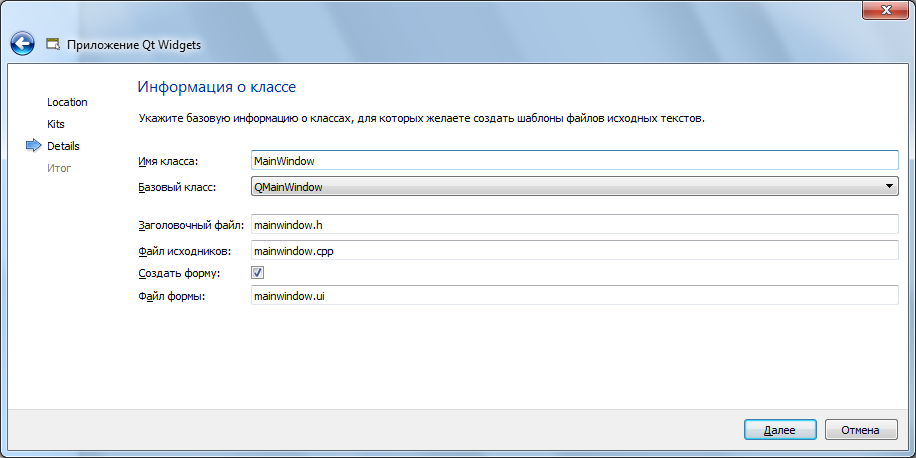
}

## Создаем проект LineProperties.





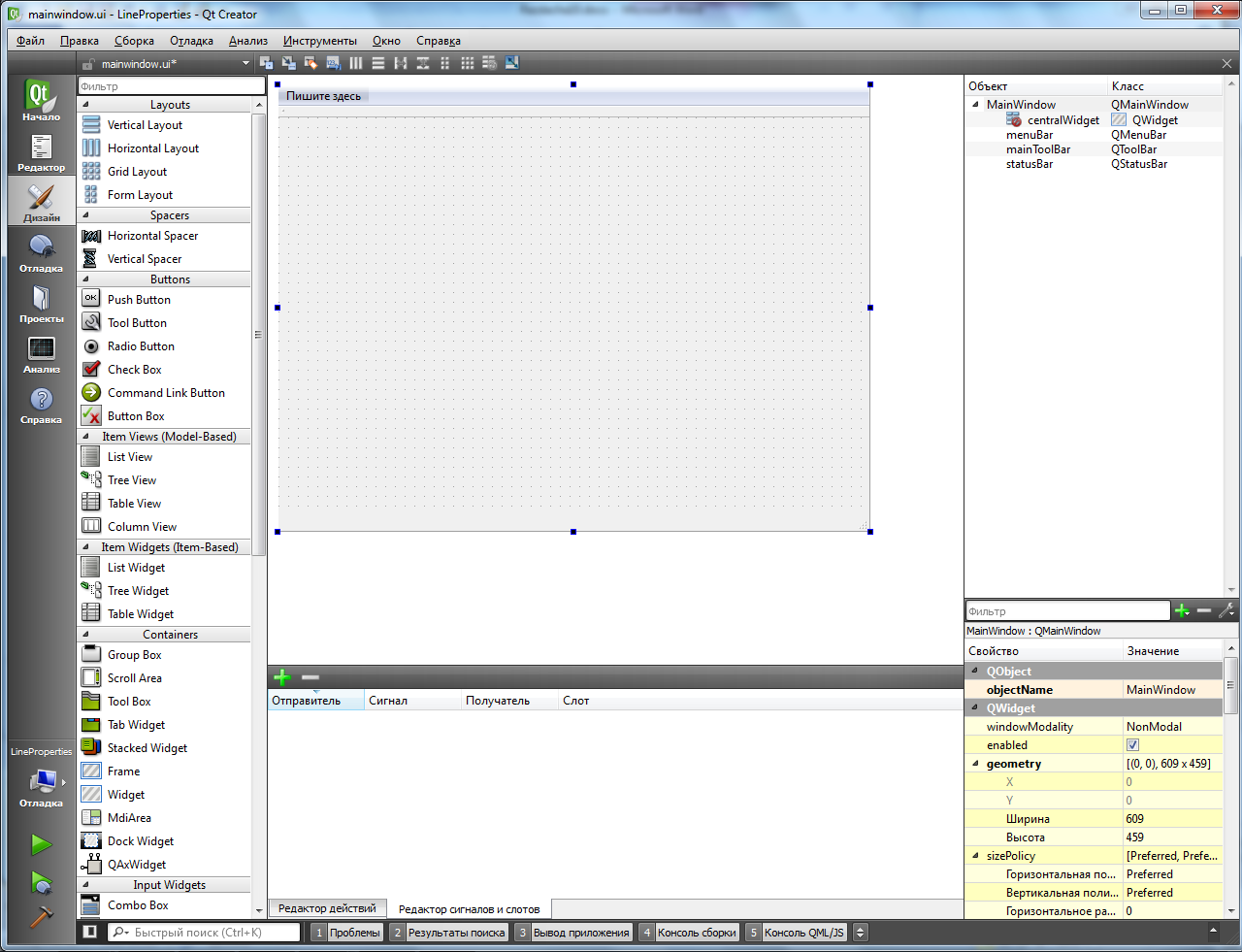




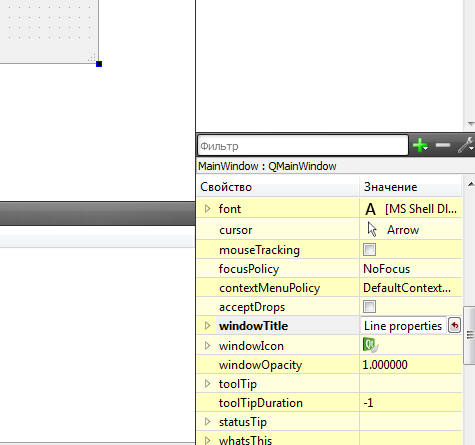
## Работаем в дизайнере.

### Внешний вид проекта

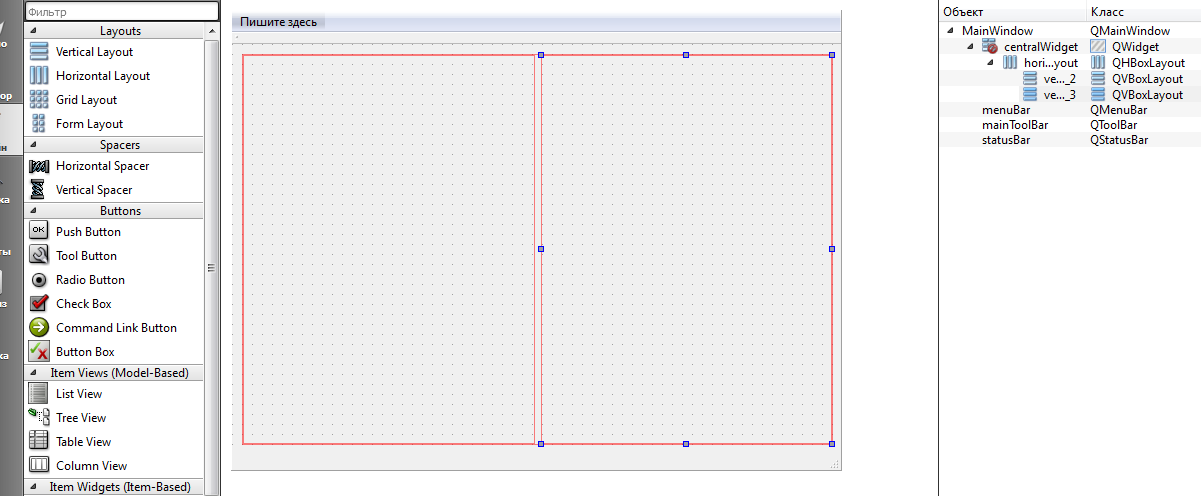
### Как выглядит наше окно в дизайнере сразу после при создании проекта:



### Задаем заголовок окна:

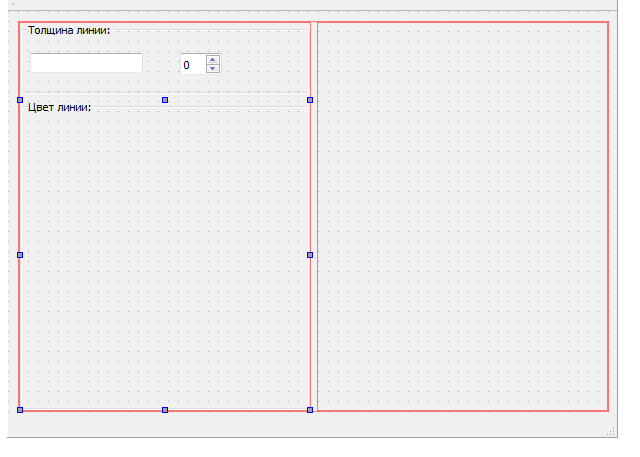


### Заполняем нашу форму компоновками.

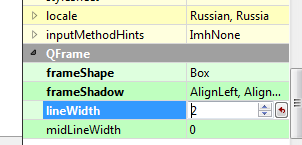


### Наполняем нашу форму элементами управления.

1. Добавляем в левую вертикальную компоновку 2 элемента GroupBox из секции Containers.
2. Изменяем названия элементов GroupBox.
3. Добавляем в верхний GroupBox элементы SpinBox и LineEdit из секции Input Widgets.
4. Выделив нижний GroupBox изменяем его высоту (виджет Свойства -> Параметр minimumSize). В результате выглядеть должно как-то так.



1. Добавляем в правую вертикальную компоновку элемент Frame из секции Containers.
2. Добавьте рамку для нашего объекта Frame



1. Заполняем нижний GroupBox объектами Label, Horizontal Slider, и PushButton (“Reset”) как показано на первом рисунке.
2. Добавьте PushButton («Exit») под нижний GroupBox.
3. Добавьте осмысленные имена виджетам, с которыми предполагаете работать (секция QObject).
4. Задайте объектам Horizontal Slider диапазон от 0-255 (секция QAbstractSlider).
5. Задайте объекту SpinBox диапазон от 0-10 (секция QSpinBoxSlider).

## Редактор сигналов/слотов

Создавать соединение посредством дизайнера можно двумя способами:

1 вариант - визуально

2 вариант с помощью редактора сигналов/слотов:

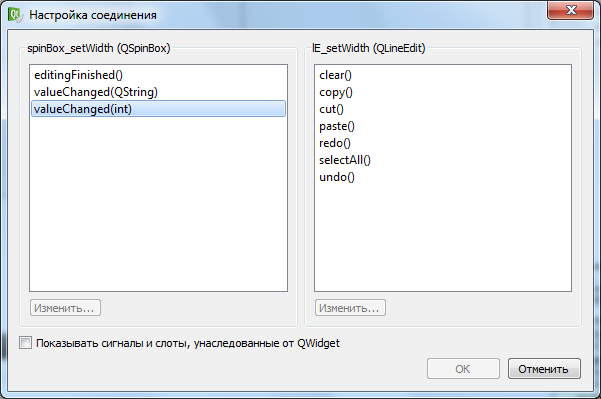
* Открыть закладку «редактор сигналов/слотов»
* Нажать «+»
* Выбрать отправителя и сигнал
* Выбрать получателя и слот

### Визуально соединяем наборный счетчик с lineEdit

1. Создаем слоты для перевода int в QString и наоборот
2. Создаем сигналы для передачи преобразованного параметра в виджет-приемник.
3. Перейти в режим редактирования сигналов/слотов (кнопка на панели инструментов дизайнера или F4 в режиме дизайнера)
4. Выделить мышью виджет, который будет генерировать сигнал (например, наборный счетчик) и соединить мышью с виджетом, который будет обрабатывать посредством своего слота генерируемый сигнал, например, слайдер



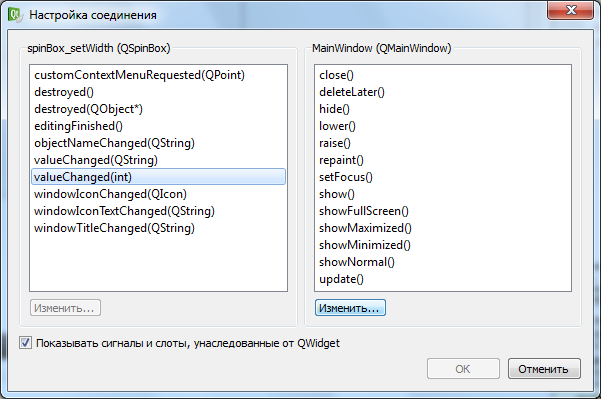
1. При этом будет выведено диалоговое окно, в котором нужно выбрать требуемые сигнал и слот:



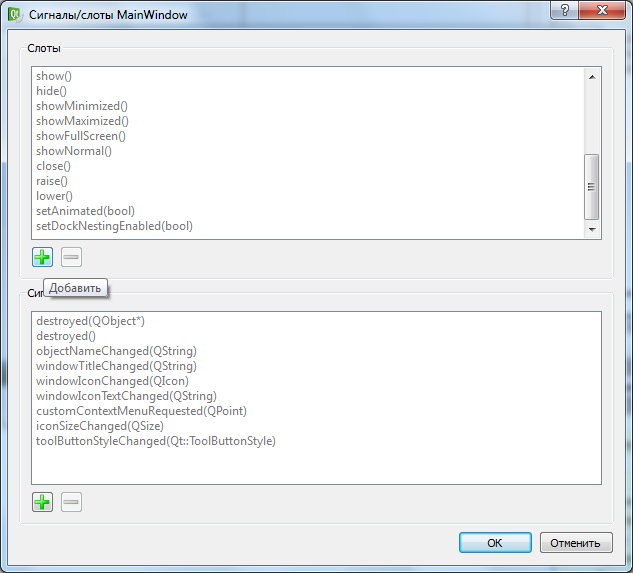
1. Если требуемый слот (сигнал) отсутствует, то необходимо соединить наши виджеты через виджет-посредник. В нашем случае – это класс MainWindow, , где мы уже определили необходимые сигналы и слоты.



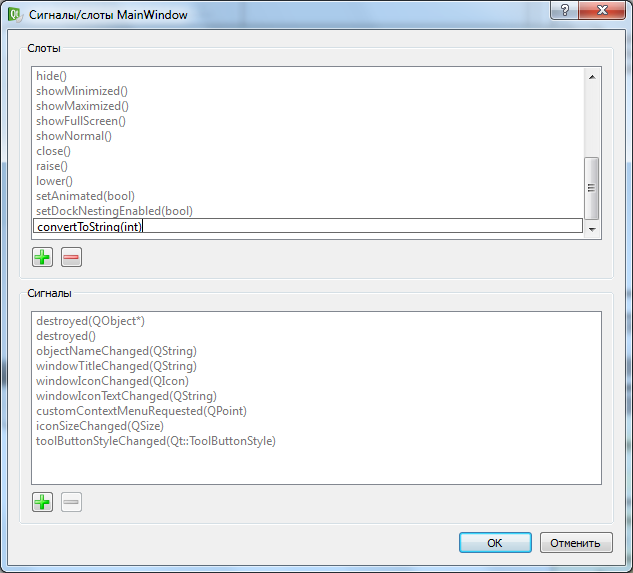
После визуального соединения



Выбираем «Изменить»

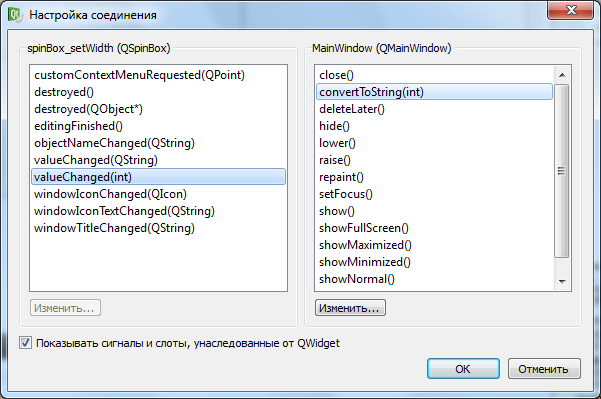


Выбираем «+»



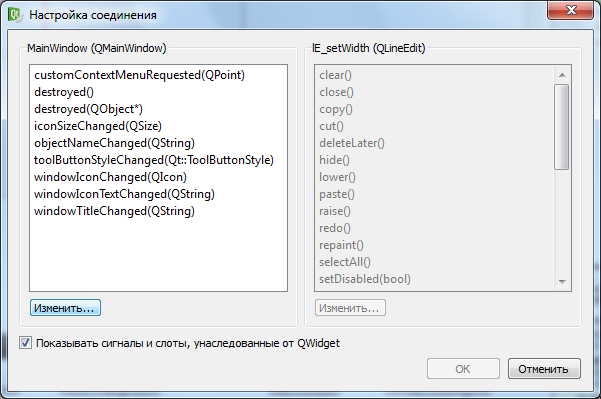
Вписываем наш слот.

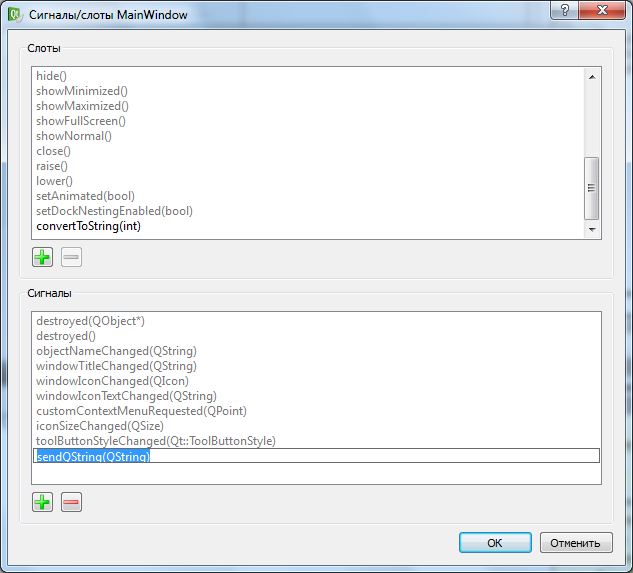
ОК

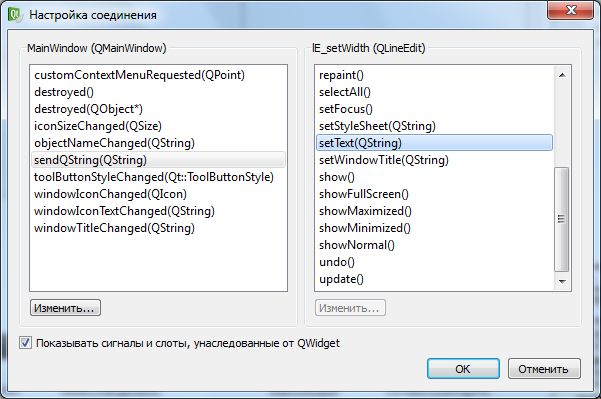


ОК

1. Для добавления сигнала – соединяем таким же способом MainWindow c lineEdit.

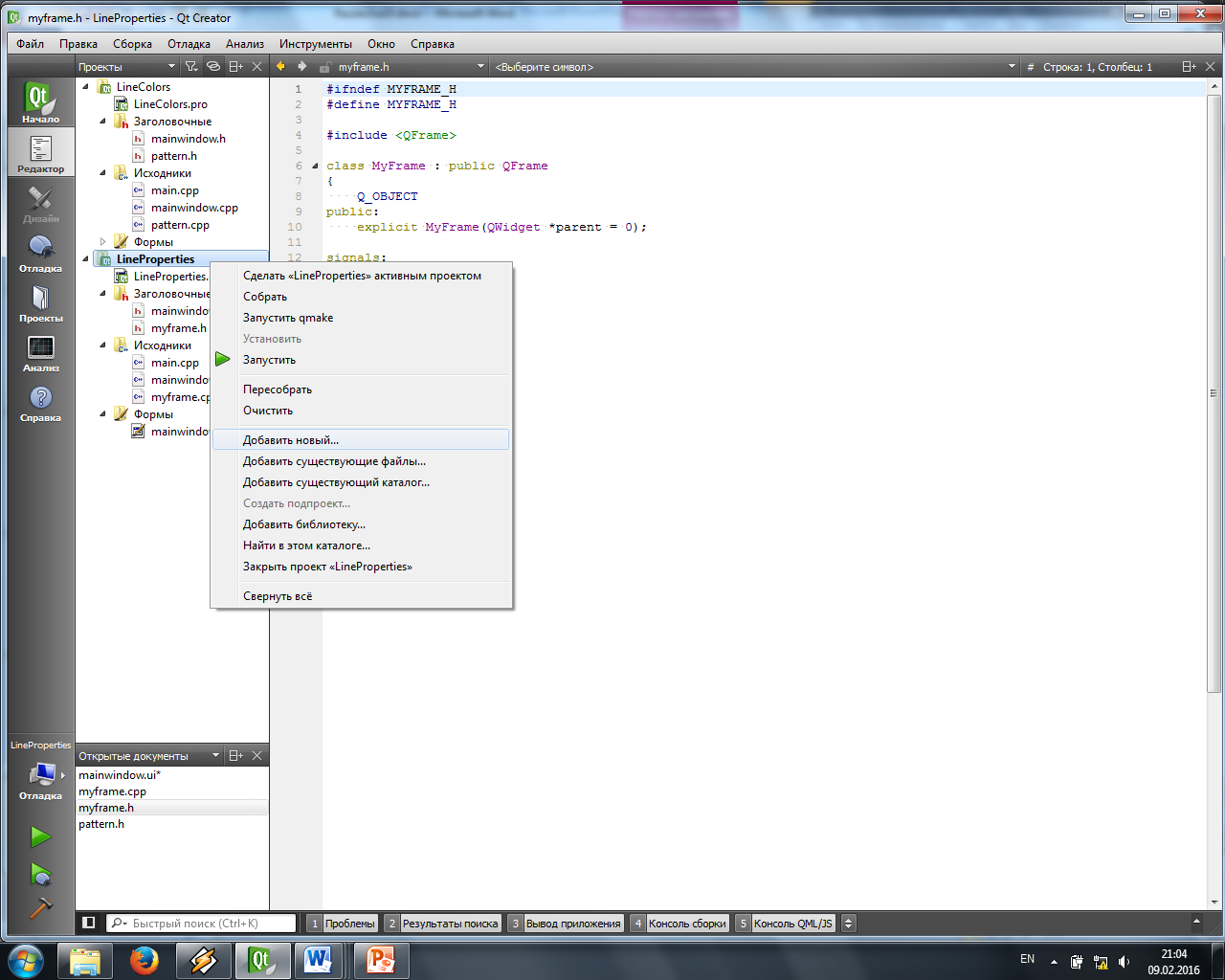




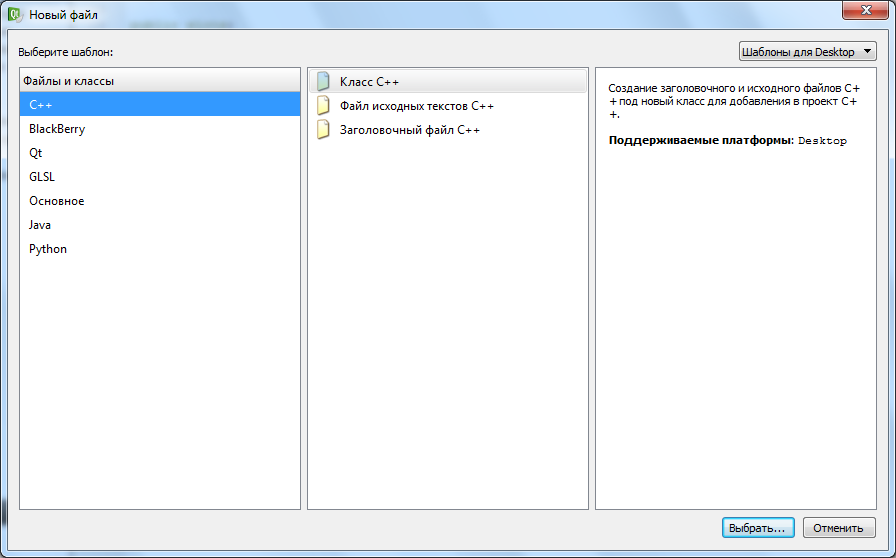


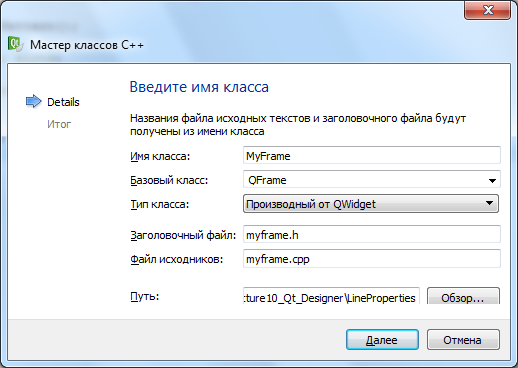
# Пользовательский виджет на примере Frame

1. Создаем класс – наследник QFrame.

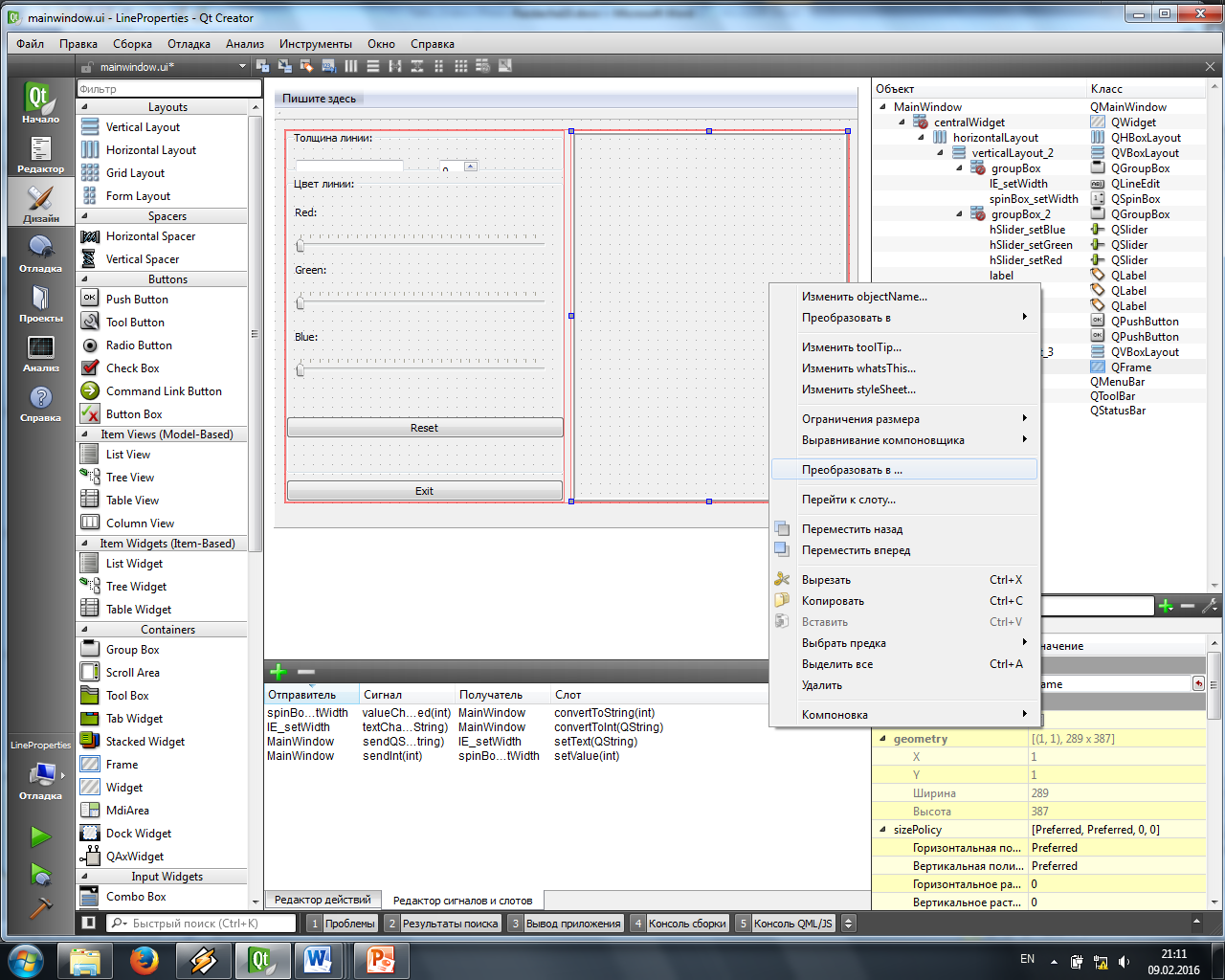


Класс С++ - Выбрать.

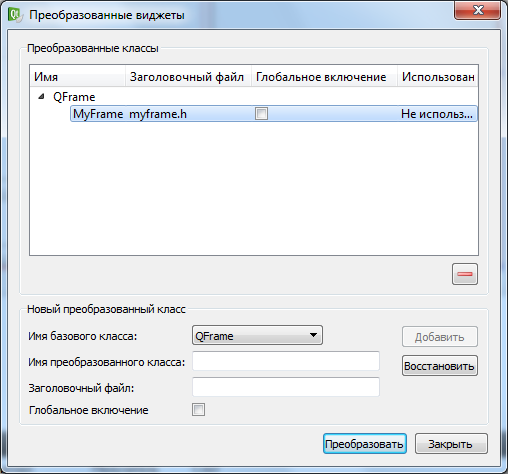




1. Переходим в наш дизайнер
2. Выбираем наш Frame.

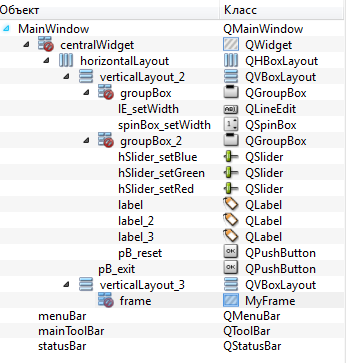


1. Преобразовать в…

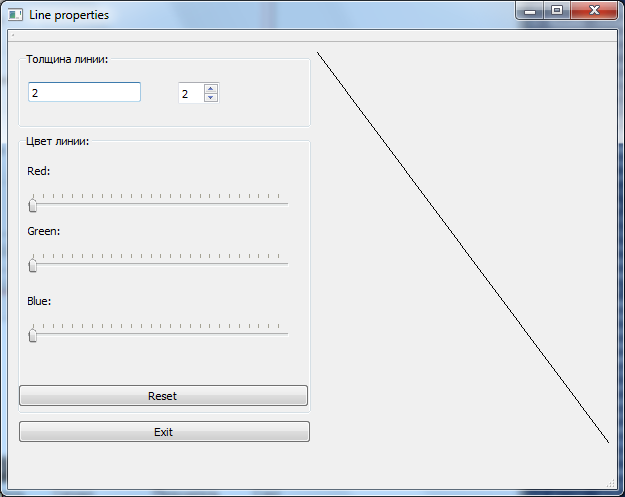


Добавляем наш новый класс-наследник QFrame, -> выбираем его в списке -> Преобразовать

Результат должен быть :

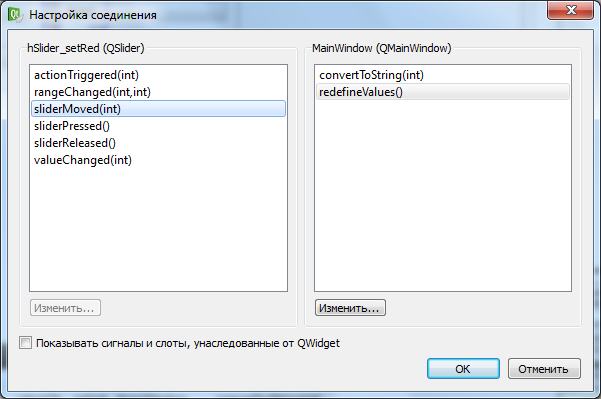


1. Переопределяем в классе MyFrame методы класса QFrame, необходимые для рисования
2. Так как теперь мы рисуем в классе MyFrame, то, соответственно, в нем переопределяем метод paintEvent()
3. Можно проверить результат, нарисовав прямую.



# Соединяем QSlider и наш QFrame

1. Для того, чтобы рисовать разными стилями в нашем frame необходимо задать свойства пера QPen. Разумно объявить объект QPen полем класса. Не забудьте #include <QPen>
2. В методе paintEvent устанавливаем наш объект QPen для рисования с его помощью;
3. В классе Mainwndow создаем слот redefineValues(), в котором присваиваем целым значениям текущие значения трех слайдеров с помощью метода int QSlider::value() const;
4. Создаем объект [**QColor**](qthelp://org.qt-project.qtgui.532/qtgui/qcolor.html#QColor-2)(int*r*, int*g*, int*b*, int*a* = 255);
5. Создаем объект [**QPen**](qthelp://org.qt-project.qtgui.532/qtgui/qpen.html#QPen-3)(const QColor &*color*)
6. Создаем в классе MyFrame слот сhangePen, принимающий параметр QPen и присвающий его нашему полю QPen/
7. Создаем в классе MainWindow сигнал sendPen, передающий созданный объект QPen слоту changePen;
8. Соединяем sendPen и сhangePen;
9. Соединяем сигналы sliderMoved QSlider от каждого QSlider и слот redefineValues;



1. Проверяем

# QPushButtom “Exit”

1. При нажатии кнопки вызывается сигнал clicked().
2. Свяжите сигнал clicked() с слотом MainWindow close() (закрытие окна):

