

СЕМИНАР 6

Виджеты компоновки

Виджеты вывода информации

QLabel

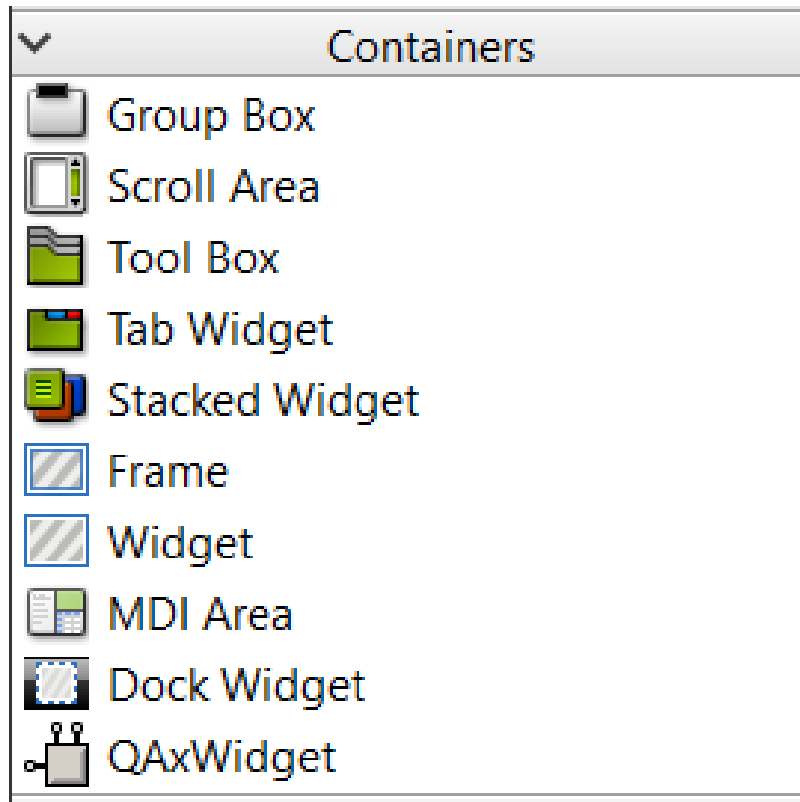
QProgressBar

QTableWidget

Стандартные виджеты Qt

- Контейнерные виджеты;
- Виджеты управления;
- Виджеты вывода информации;

Контейнерные виджеты

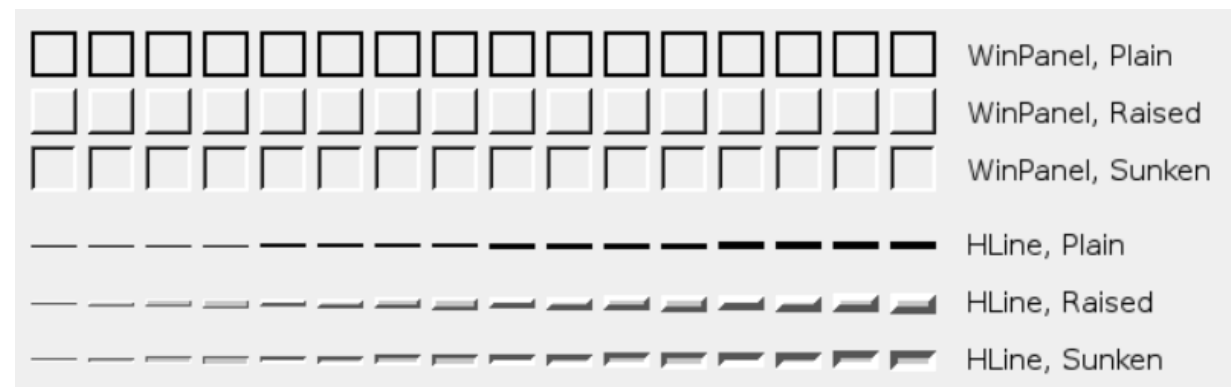
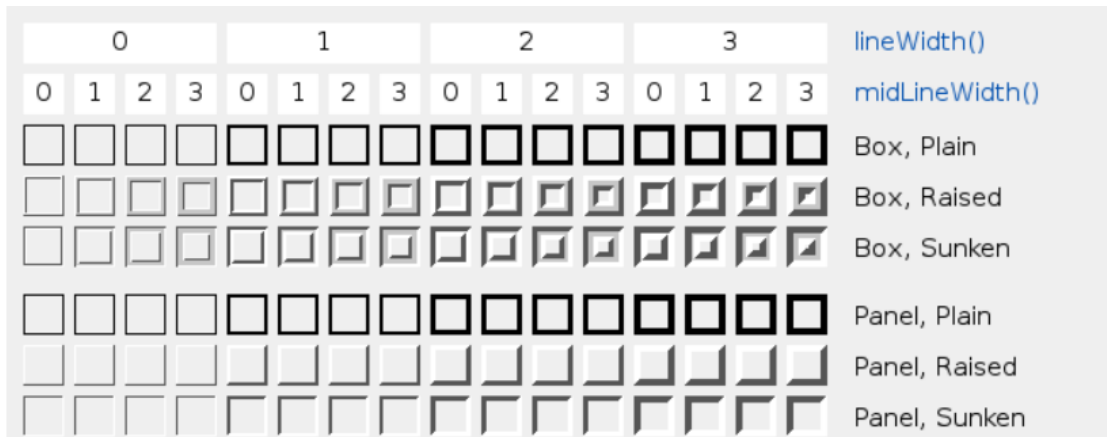


Виджеты, которые управляют размещением/отображением элементов на основной форме

QWidget и QFrame



- Позволяют добавлять на форму пустой виджет или frame;
- Для чего это нужно?
 - такой виджет можно преобразовать в custom-виджет, который вы сами запрограммировали, т.е. QWidget и QFrame - placeholders;
- Чем отличается QWidget от QFrame?
 - QFrame – основан на QWidget , но имеет возможность отображать рамку виджета (стиль рамки можно настраивать).



QGroupBox – контейнер группировки кнопок

Предназначение:

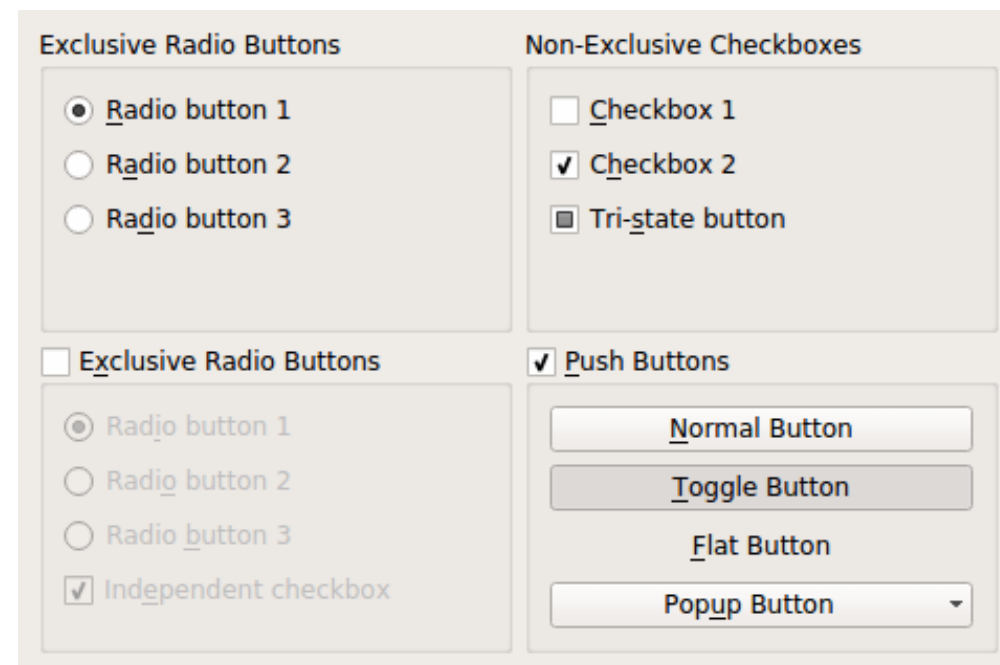
- Повысить удобство использования интерфейса;
- Упростить понимание программы.

Что собой представляет?

- QFrame с названием и местом для размещения виджетов;
- Поддерживает shortcut
- Может быть «выбираемым» (checkable) – и это состояние передается дочерним виджетам (они переходят в состояние disabled или становятся недоступны)

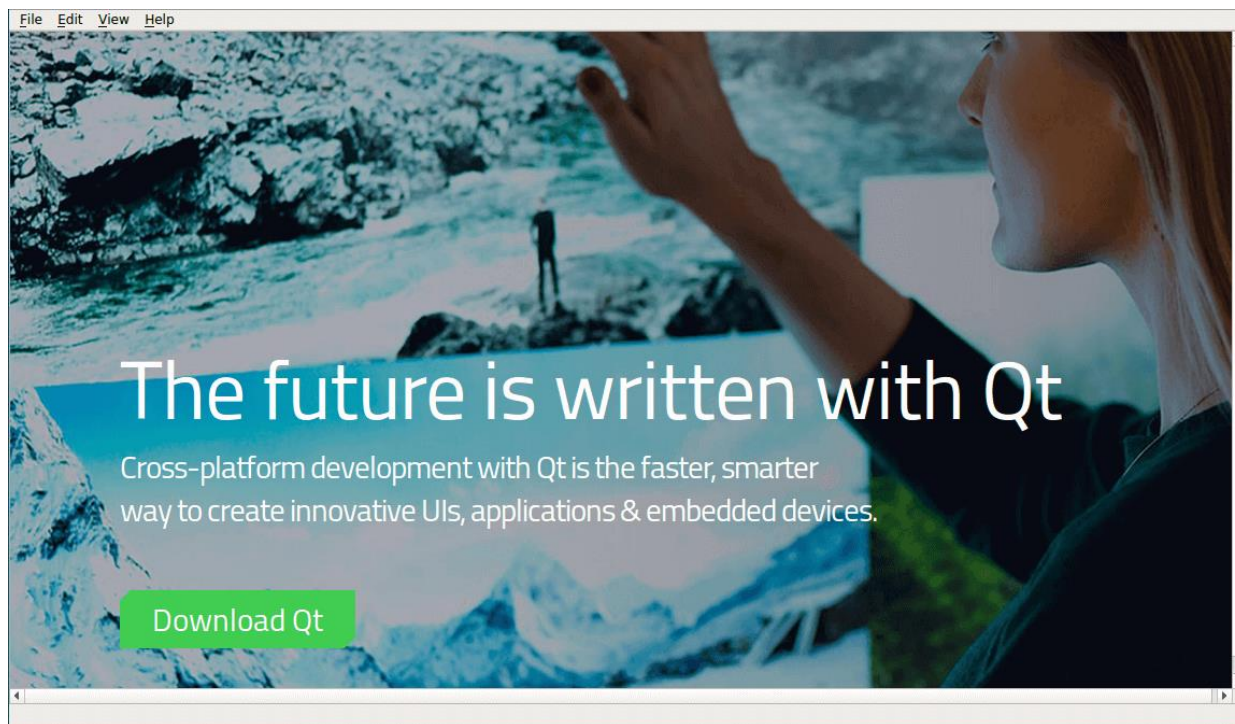
Как использовать?

- Добавить на форму и разместить в нём объекты;
- Сделать свой класс-наследник.

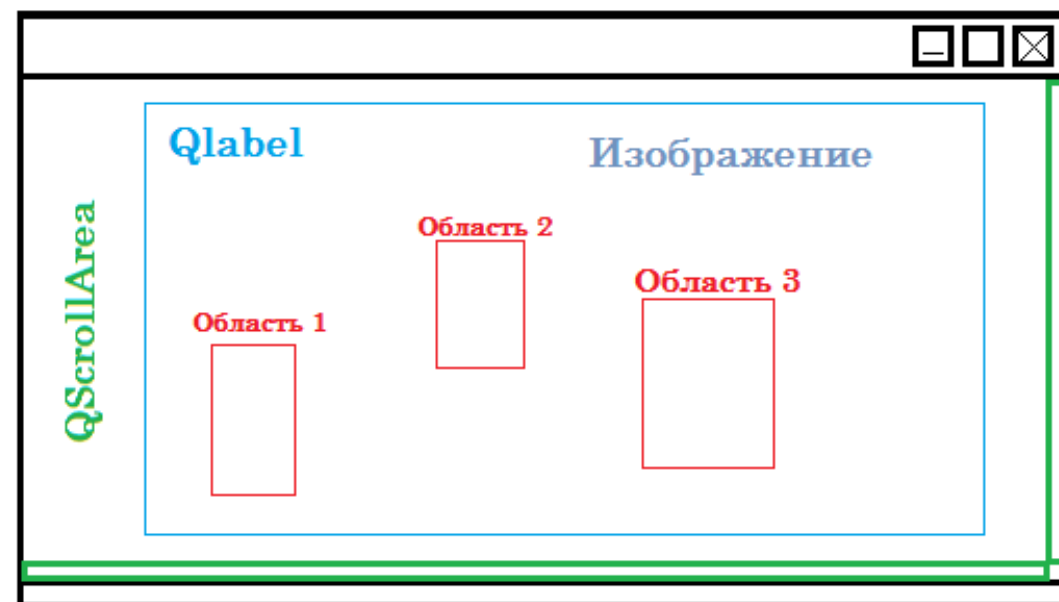


QScrollArea – виджет видовой прокрутки

- Окно просмотра части информации;
- Размещает внутри виджет и если его размеры превышают размеры окна, то появляются полосы прокрутки;



QMainWindow

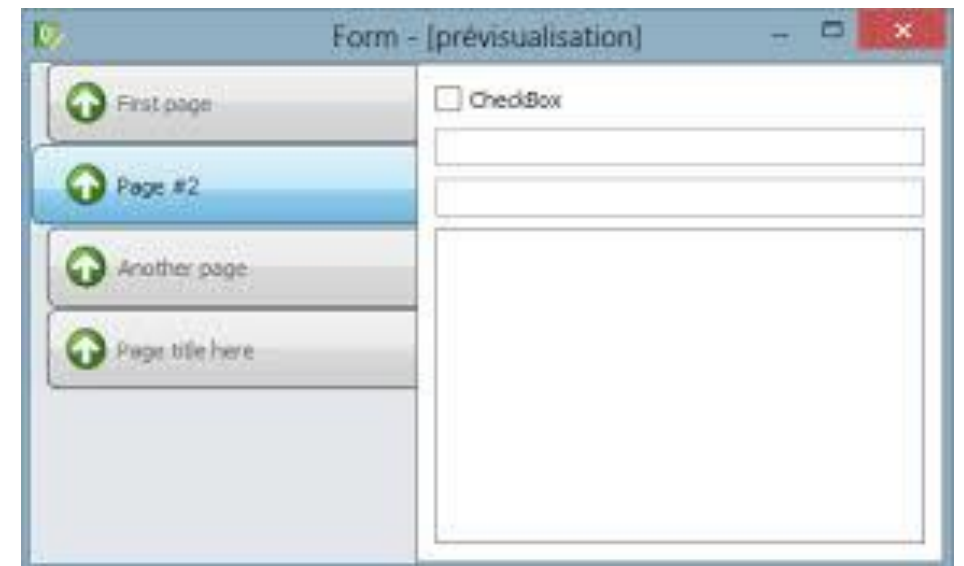
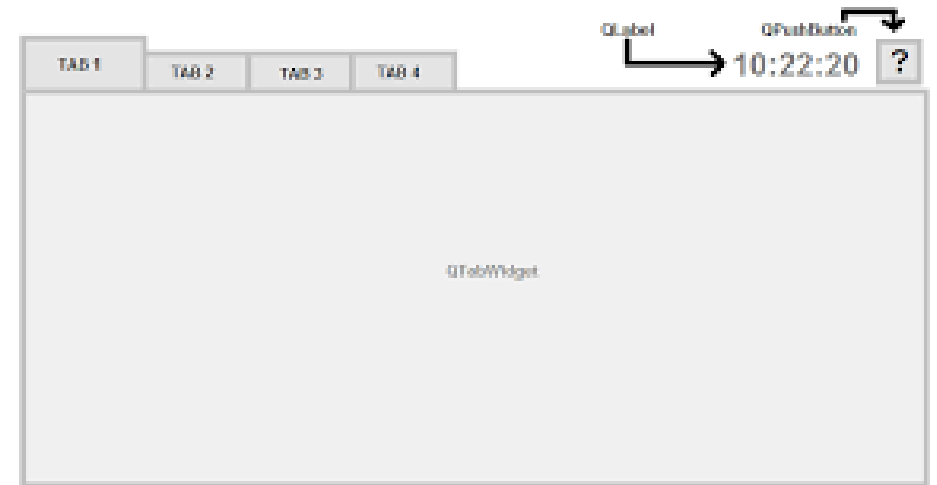


QTabWidget – вкладки

- Назначение – разгрузить сложное для восприятия окно приложения (когда имеется множество опций, параметров и т.п.);
- Разделяет основное окно на серию логически скомпонованных подокон;
- В один момент отображается одна вкладка.

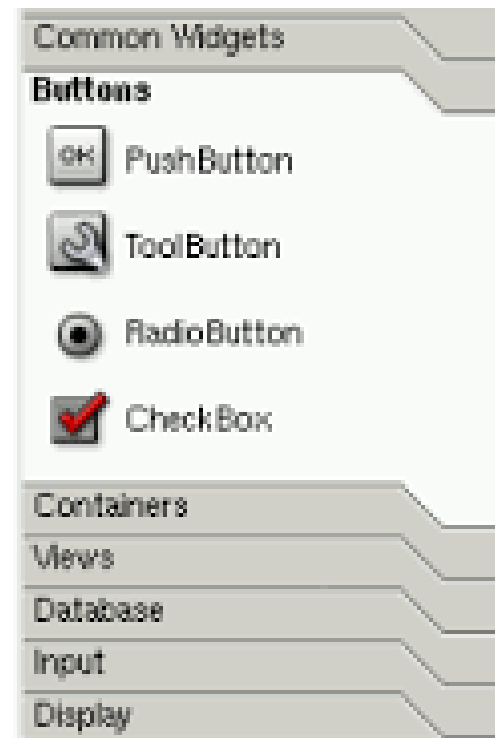
Порядок работы:

- Создается *TabWidget*;
- Создается виджет для каждой вкладки
- Виджеты вкладок добавляются в *TabWidget*



QToolBox – виджет панели инструментов

- Представляет собой вкладки, расположенные вертикально;
- Связанные виджеты отображаются под названием вкладок;
- В один момент времени отображается развернутым виджет **только одной** вкладки.

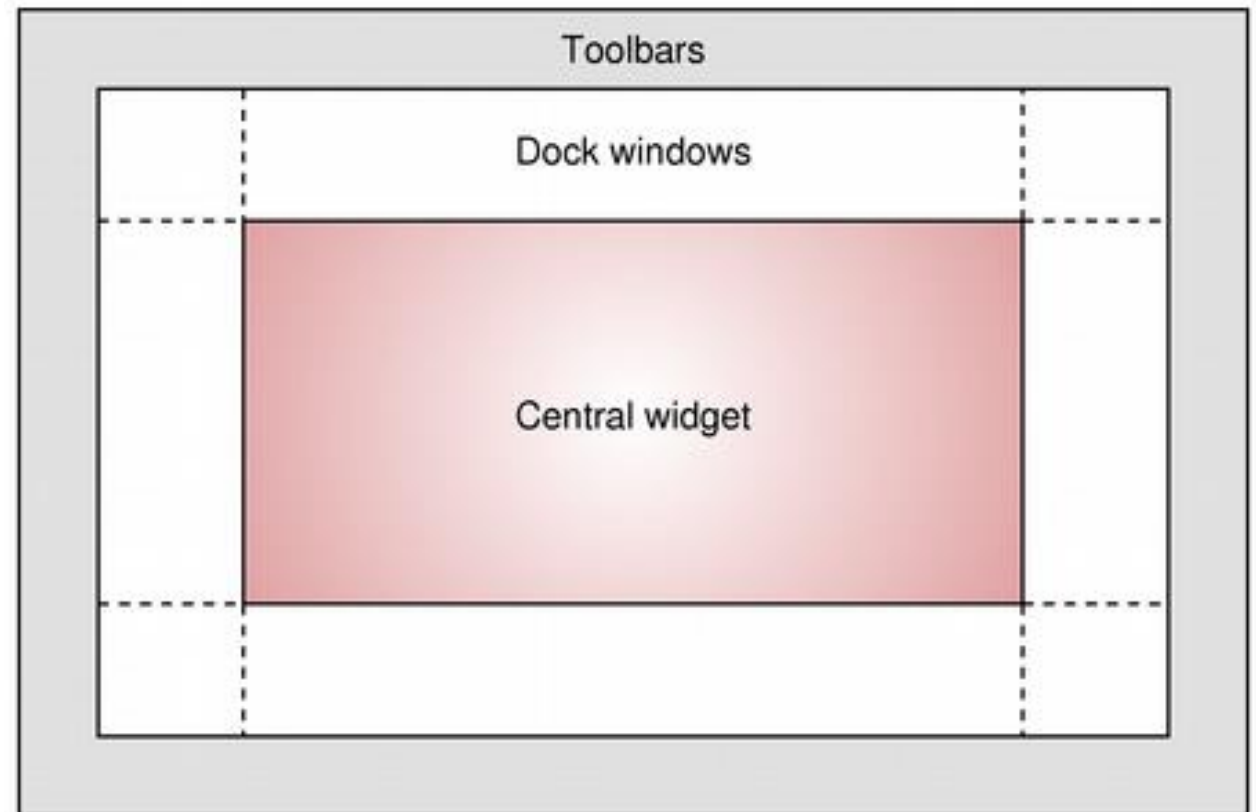


QStackedWidget

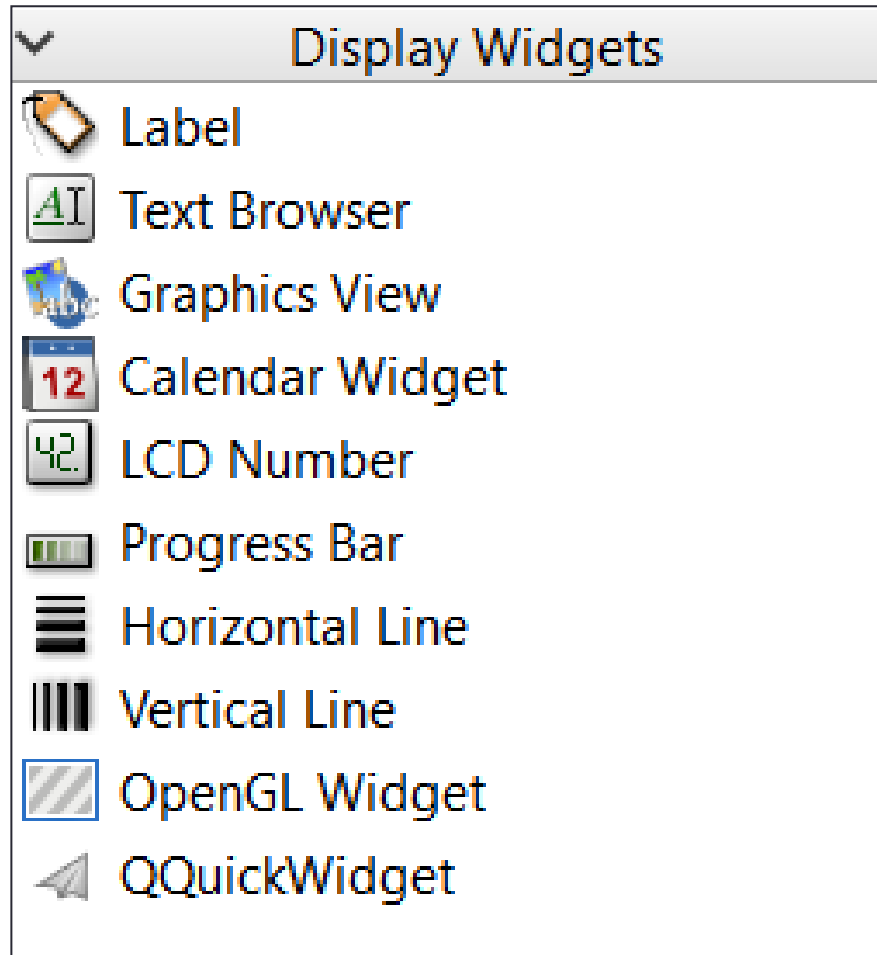
- Виджет, который в отдельно взятый промежуток времени показывает только один из виджетов, хранящихся в стеке;
- Для каждого дочернего виджета установлен свой идентификатор;

QDocWidget - Доки

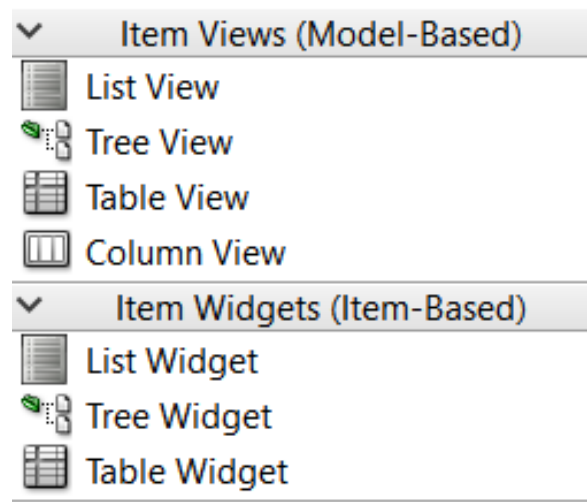
- Составной виджет, который можно перемещать и располагать у любой из 4 сторон окна приложения;
- Используется с QMainWindow;



Проектирование GUI. Вывод информации.



- Надпись - QLabel;
- Индикатор процесса - QProgressBar
- Электронный индикатор - QLCDNumber
- Графическая сцена – QGraphicsView
- Таблицы – QTableWidgetItem
- И т.д...



Надписи. Класс QLabel.

Виджет надписи служит для отображения текстовой или графической информации пользователю (при этом пользователь вносить изменения не может, изменять отображаемую информацию может только приложение).

QLabel может выводить не только текстовую, но и графическую и анимационную информацию.

Метод	Описание
setText()	Вывод текста
setPixmap()	Отображение картинки
setMovie	Отображение видео

Может выводить информацию в формате HTML.
Пример использования этих функций и не только представлен в проекте Labels.



Надписи. Класс QLabel.

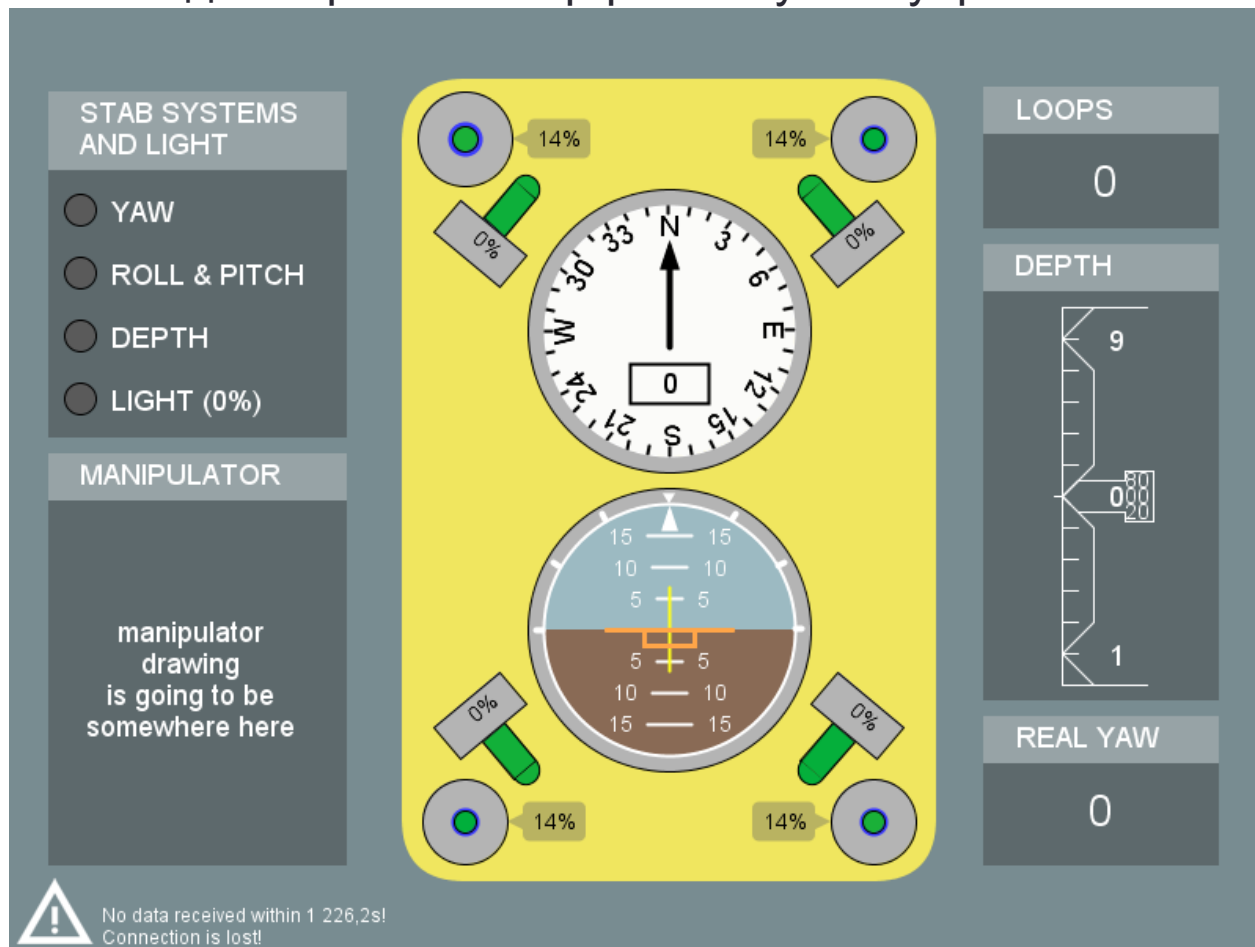


Проект с виджетом, который представлен на слайде называется Labels и находится для вас в свободном доступе.

В данном проекте приведены примеры использования многих полезных функций по работе с надписями и многими другими виджетами Qt (установка картинки, загрузка видео, масштабирование картинки, перенос текста в метке, установка шрифта и т.п.).

Практическое задание 1

Создать проект интерфейса пульта управления



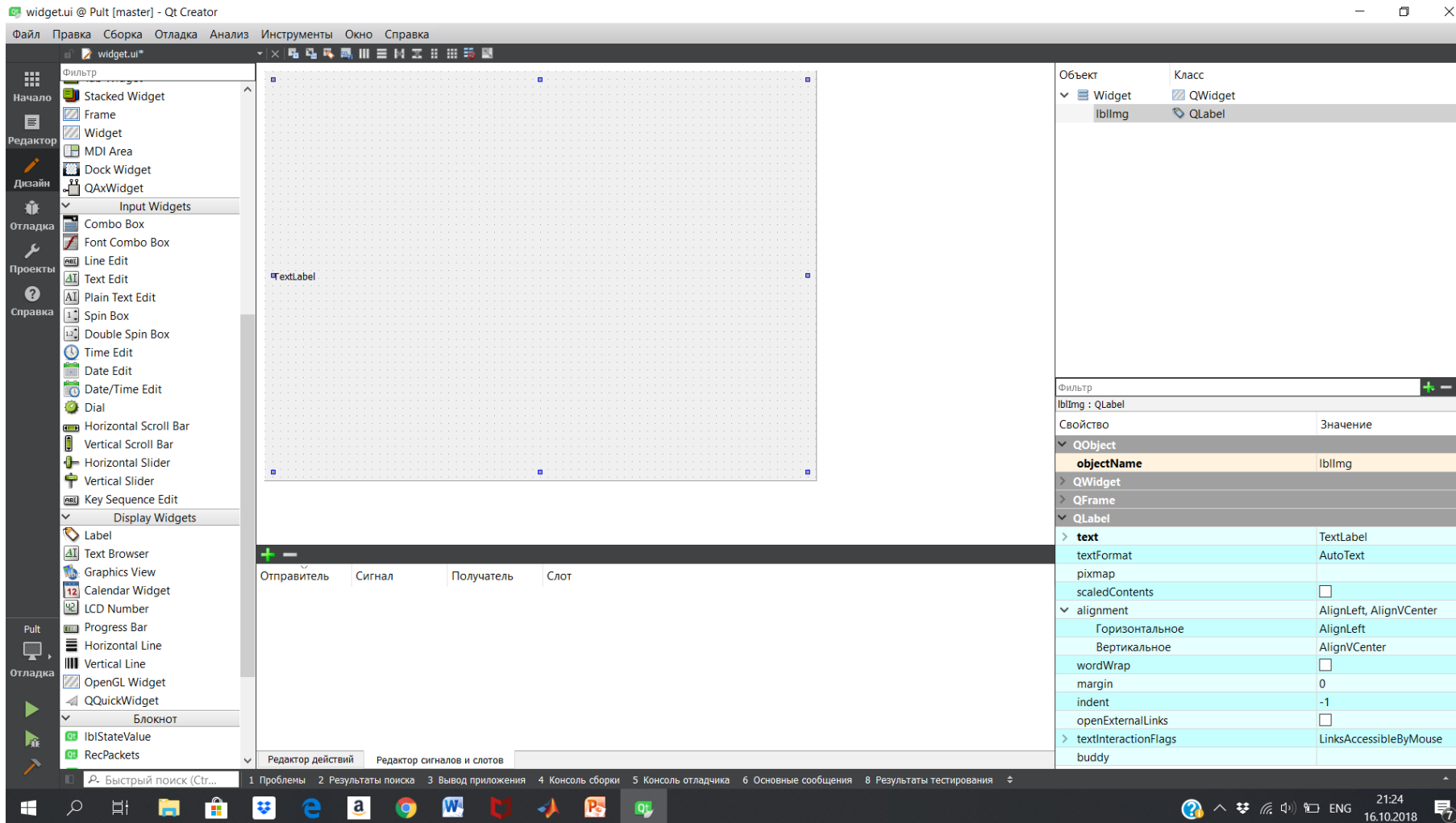
Создать и разместить в проекте метку

Создать файл ресурсов и поместить туда изображение пульта

Создать объект растрового изображения QPixmap с изображением из файла ресурсов

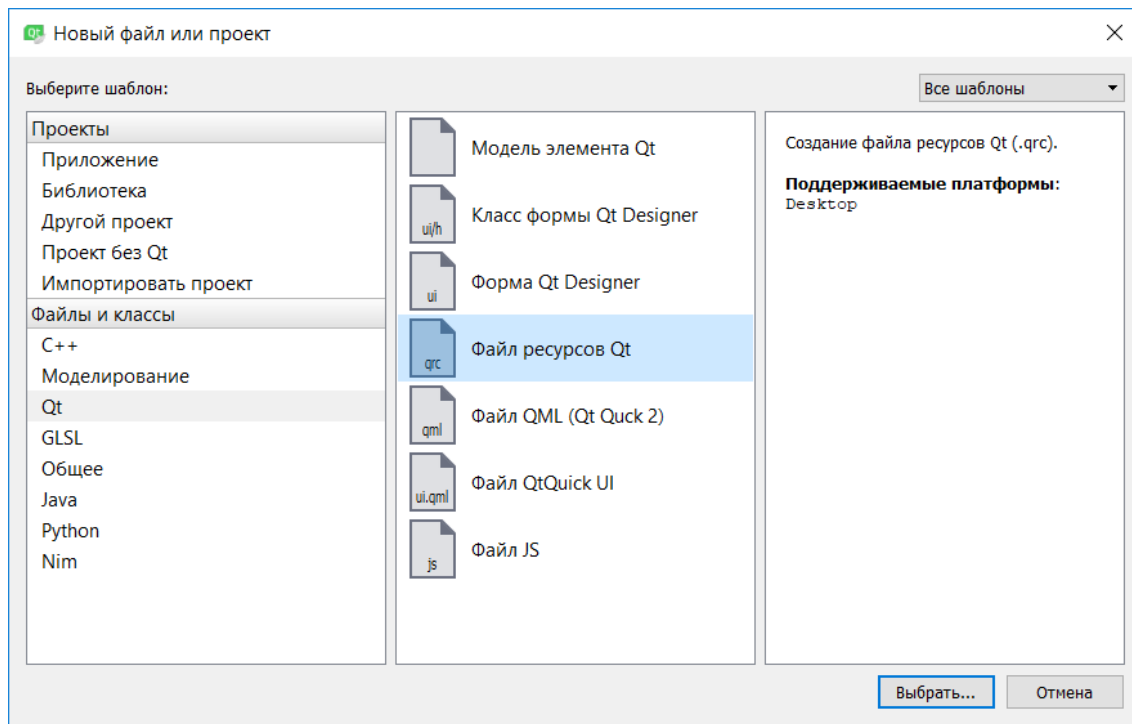
Установить в виджете метки изображение пульта (QPixmap)

1. Создание и размещение метки



1. Проект создать с формой
2. На форме разместить метку QLabel (название объекта в коде lblImg)
3. Скомпоновать виджет по вертикали

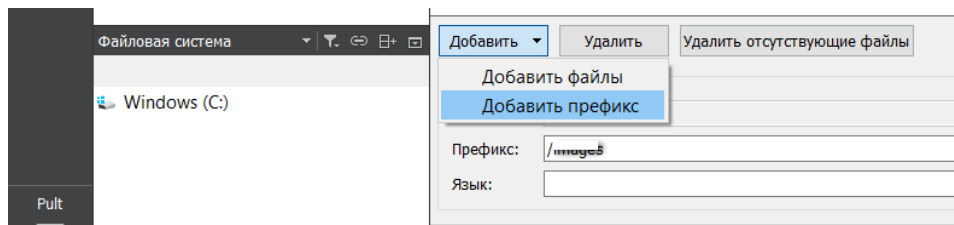
2. Файл ресурсов



1. Добавить к проекту файл ресурсов (Файл->Создать файл или проект->Файл и класс Qt->Файл ресурсов Qt)

2. В файле ресурсов добавить префикс (например "/")

3. Добавить файл рисунка пульта images/pult.png

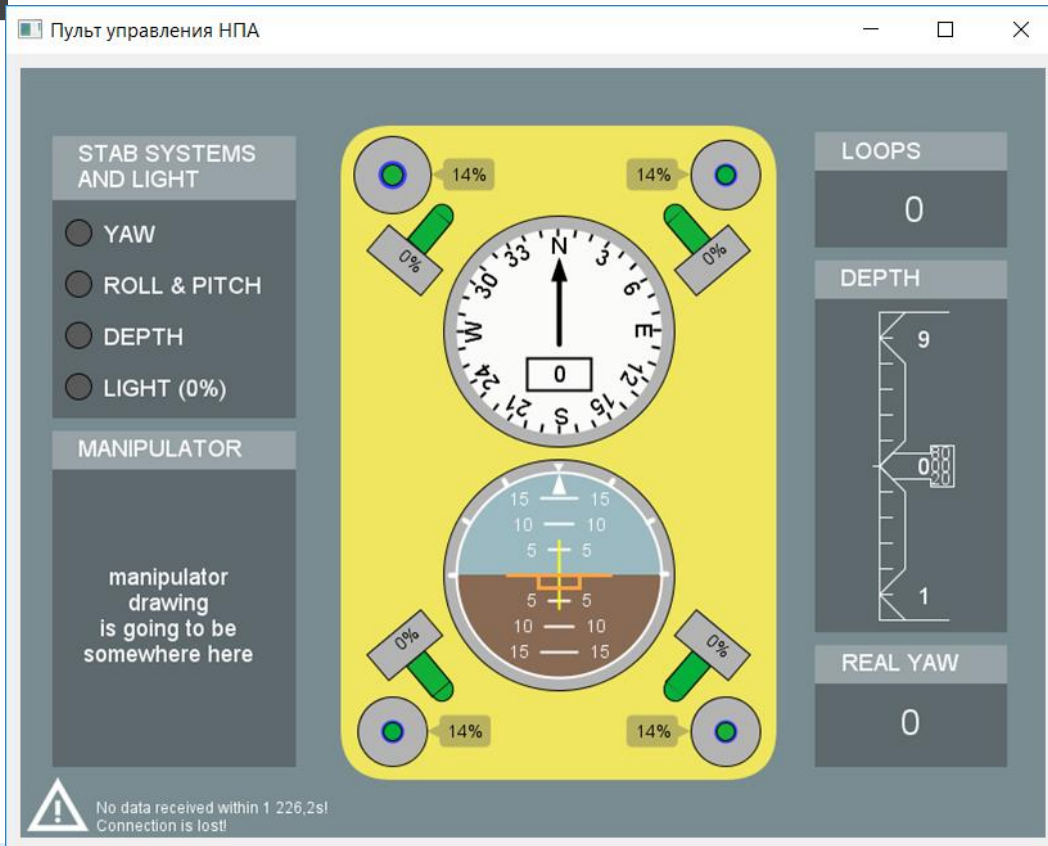


3-4. Создание и установка объекта растрового изображения

3. В классе виджета дописать код, приведенный на рисунке

```
Creator
Анализ Инструменты Окно Справка
Pult/widget.cpp Widget::Widget(QWidget *)
1 #include "widget.h"
2
3 Widget::Widget(QWidget *parent) :
4     QWidget(parent)
5 {
6     setupUi(this);
7     //0. установим название нашего пульта
8     setWindowTitle("Пульт управления НПА");
9     //1. сначала создадим файл ресурсов, спрячем туда нашу
10    //картинку для метки
11    //2. создадим объект QPixmap
12    //в конструктор объекта QPixmap передаем путь к картинке в qrc файле
13    QPixmap pix(":/images/pult.png");
14    //установим изображение в метке
15    //При этом если автоматическое изменение размера позволено,
16    //то метка сама поменяет размер и вы просто загружаете картинку
17    lblImg->setPixmap(pix);
18    //если нужно поменять размер картинки в ПУ
19    //pix=pix.scaled(QSize(100,100),Qt::KeepAspectRatio);
20    //lblImg->setPixmap(pix);
21    //выравниваем картинку по центру
22    //lblImg->setAlignment(Qt::AlignCenter);
23
24 }
```

4. Наслаждаться результатом



Выводы

1. Класс QLabel позволяет:

- Размещать текстовую информацию:
 - Простой текст;
 - Текст в формате HTML (включая простой текст, таблицы, картинки и т.п.);
 - Гипертекстовые ссылки (при нажатии на ссылку вызывается сигнал `linkActivated()`);
- Размещать графические изображения (метод `setPixmap`);
- Размещать видео (метод `setMovie()`, класс в котором можно размещать видео для загрузки - `QMovie`);

2. При этом можно управлять размещением информации

3. Метку можно ассоциировать с любым другим виджетом (текст должен содержать знак `&`, метод связывающий метку с другим виджетом - `setBuddy(указатель_на_виджет)`)

Стили в Qt

- Классы стилей
- Каскадные стили документа
 - За основу был взят язык CSS (Cascading Style Sheets), используемый в HTML
 - Каскадный стиль - текстовое описание стиля
 - Стиль может быть описан в отдельном документе .qss. Устанавливается, используя метод `QApplication::setStyleSheet()`
 - Стиль может быть описан в отдельном виджете `QWidget::setStyleSheet()`
 - Каскадный стиль можно подключить в командной строке, указав после ключа — `stylesheet имя_файла_стиля.qss`

Синтаксис каскадных стилей

Селектор {свойство: значение}

Селектор – целевой элемент. Указывает для какого из виджетов будет использоваться стиль.

“свойство: значение” – определение селектора.

Примеры:

```
QLabel {  
    color*: red;  
    background-color: white;  
}
```

//Все метки будут иметь красный текст на белом фоне.

*Цвет текста можно задавать названием, в формате RGB, в HTML-стиле.

Синтаксис каскадных стилей

1. Если в разных селекторах используются одинаковые определения:

```
QLabel, QPushButton, QLineEdit { background: white }
```

2. Для применения определения сразу ко всем виджетам:

```
*{ background: white }
```

3. Применение стиля только к дочерним виджетам какого-либо виджета-родителя:

```
QDialog QPushButton { color: rgb(255, 0, 0)}
```

4. Стил можно менять для подэлементов виджета

5. Стил можно применять к элементам, находящимся в определенном состоянии.

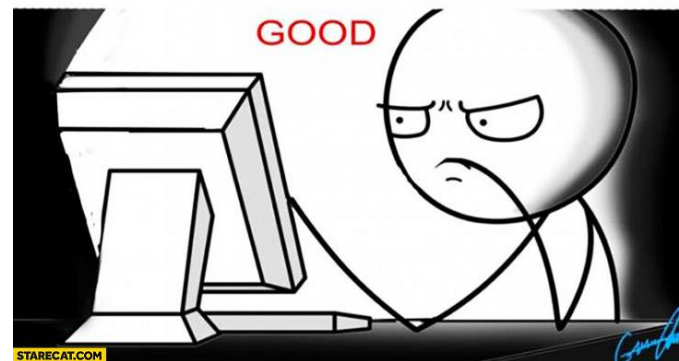
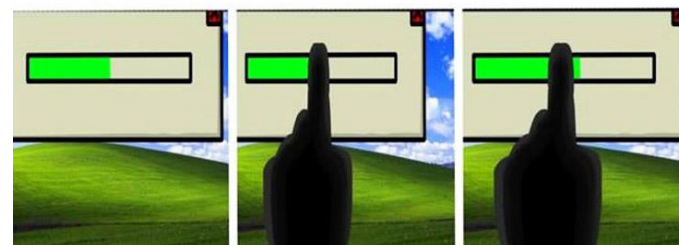
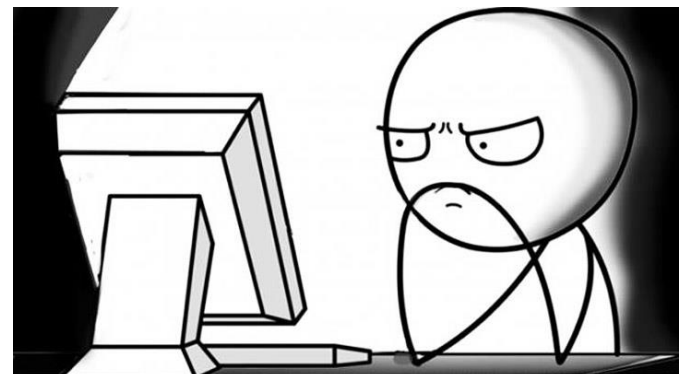
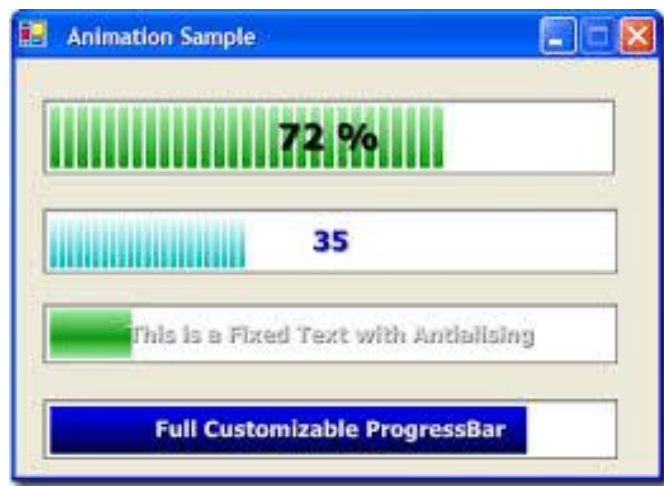
****Каскадный стиль не чувствителен к регистру, так что Red= RED = red**

*****комментарии помещаются внутрь символов /* */, /*комментарий*/**

Индикатор процесса. QProgressBar.

Индикатор процесса (Progress Bar) – виджет, демонстрирующий процесс выполнения операции. Полное заполнение информирует пользователя о завершении операции.

Необходим в том случае, когда программа выполняет продолжительные действия, - виджет дает пользователю понять, что программа не зависла, а находится в работе.



QProgressBar. Полезная информация

- Виджет может быть расположен как горизонтально, так и вертикально. Ориентацию можно задать методом `setOrientation(Qt::Vertical/Qt::Horizontal)`
- Можно задавать количество шагов заполнения виджета (метод `setRange(количество_шагов)`)
- Сбросить индикатор в 0 можно методом `(reset())`
- Выставлять процент заполнения виджет можно методом `setValue()`

Использование QProgressBar в GUI подводных аппаратов

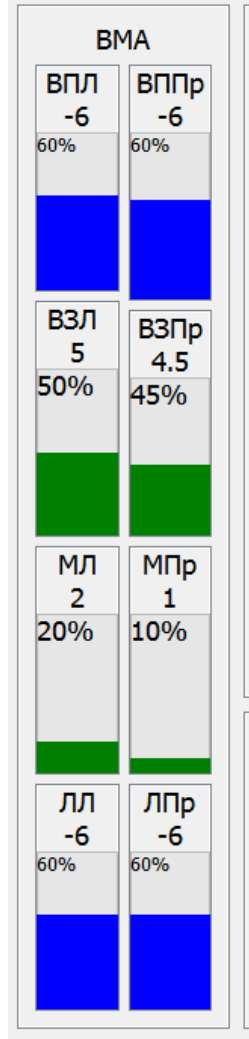
PultWidget

Можно использовать для вывода информации о «загрузке» по тяге двигателей подводного аппарата или иных устройств. (цвет виджета изменяется в зависимости от тяги (напряжения) ВМА)

1). Для этого необходимо разместить progress bar вертикально:

- В коде это можно сделать методом `setOrientation (Qt::Vertical)`;
- В QtDesigner выставив значение “Vertical” параметру “orientation:” в свойства QProgressBar вашего объекта.

▼ QObject	
objectName	barVPL
> QWidget	
▼ QProgressBar	
minimum	0
maximum	100
value	24
> alignment	AlignLeft, AlignVCenter
textVisible	<input checked="" type="checkbox"/>
orientation	Vertical
invertedAppearance	<input type="checkbox"/>
textDirection	TopToBottom
> format	%p%



Индикатор процесса. QProgressBar.

Затем в коде выставить необходимые значения положения шкалы progress bar'a методом setValue().

Данный метод принимает значения типа int (т.е. целочисленные).

Приведем значения тяги к процентам и целочисленным значениям:

```
barVPr->setValue(int((fabs(VPr_Value)/maxVMA_Value)*100));
```

//но на метке над индикатором выводим реальное значение с учетом знака

```
lblVPr_Value->setText(QString::number(VPr_Value));
```

Значения progress bar выставляем вне зависимости от знака напряжения на ВМА. В зависимости от знака можем изменять цвет нашего индикатора (для этого управляем стилем нашего progress bar'a):

```
if (VPr_Value > 0) barVPr->setStyleSheet("QProgressBar::chunk  
{background-color: green;}");
```

```
else barVPr->setStyleSheet("QProgressBar::chunk {background-color:  
blue;}");
```

*Полезное руководство по управлению стилями Qt:

<http://doc.crossplatform.ru/qt/latest/stylesheets-reference.html#chunk-sub>

Практическая часть::Индикатор процесса. QProgressBar.

Запрограммировать форму таким образом, чтобы:

- 1). Индикаторы процесса отображали значение заданное слайдером.
- 2). Цвет индикаторов должен меняться в зависимости от знака заданного значения.
- 3). Метка над индикатором должна отображать заданное значение с учетом знака.

Подсказка:

При изменении положения слайдера посылается сигнал `sliderMoved(int)`, который передает актуальное значение положения слайдера;

Таблицы. QTableWidgetItem.

- Класс QTableWidgetItem представляет собой таблицу.
- Объект ячейки таблицы реализован в классе QTableWidgetItem.
- Вставить ячейку в таблицу – метод setItem(N_строки, N_столбца)
- В созданных ячейках можно размещать текст (setText()), изображения (setIcon()) и виджеты (setCellWidget()).

Задание траектории
движения АНПА

ТРАЕКТОРИЯ			
Тек. коорд (АНПА1)	78.929	0.566	11.713
Тек. коорд (АНПА2)	61.703	0.161	0.006
#	X	Y	Z
ВВОД		УДАЛИТЬ	ЗАПИСЬ
1	0	0	20
2	0	0	80
3	30	0	80
4	30	0	20
5	60	0	20
6	60	0	80

Вывод информации о параметрах движения
АНПА, данных с датчиков

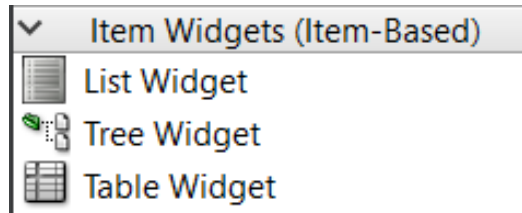
		Курс ГА	Курс ГМК	Курс магн.		
		-63.48	-0.47	-3.88		
АНПА1	Курс	Глуб.	Отст.	Крен	Дифф.	Марш
Зад.	-637.93	-	1.76	0.00	0.62	0.00
Тек.	-753.62	0.57	1.83	11.75	4.02	0.00
АНПА2	Курс	Глуб.	Отст.	Крен	Дифф.	Марш
Зад.	-0.05	-	0.00	0.00	8.77	5.82
Тек.	-0.47	0.16	0.00	-0.58	-2.36	0.00

Вывод информации об
ошибках, качестве
связи и т.п.

ИИБ	Кх пульс	АИСТ	А1ДВН
TRAX	РУД	МГА КМ	А1ДВК
РА-500	БСУ	МГА ПУ	А2ДВН
ДГ	КМ	СКУ	А2ДВК
РМД ПУ	КМ-БСУ	GPS ПУ	ОВ
РМД КМ	БСУ-КМ	GPS КМ	Мин зар

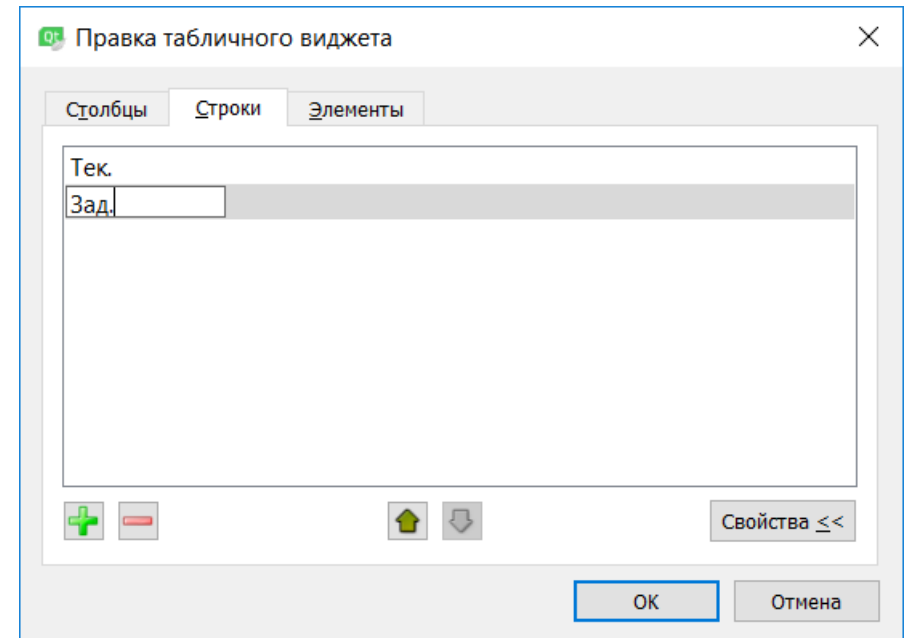
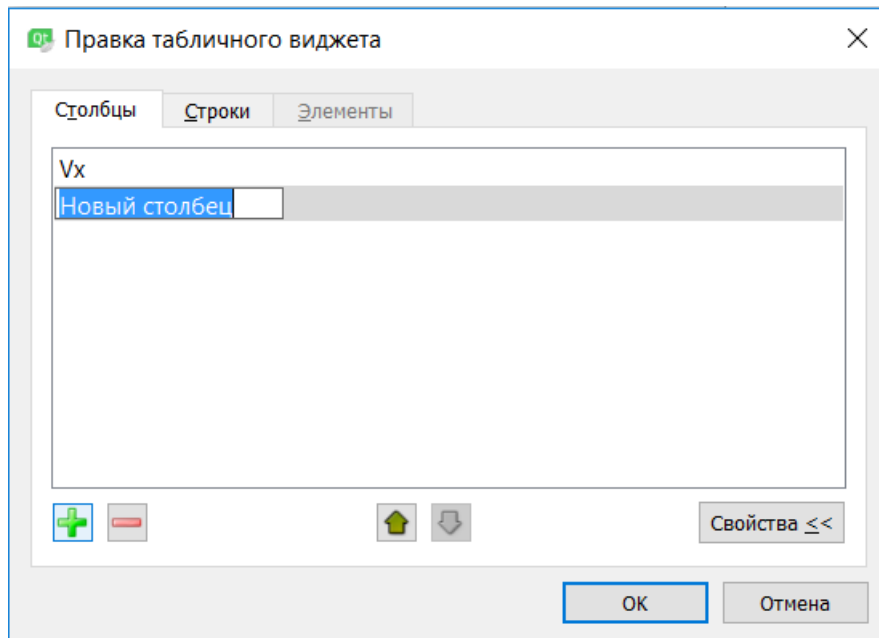
Создание таблицы. QTableWidgetItem.

1. В QtDesigner в форме выбираем Table Widget:



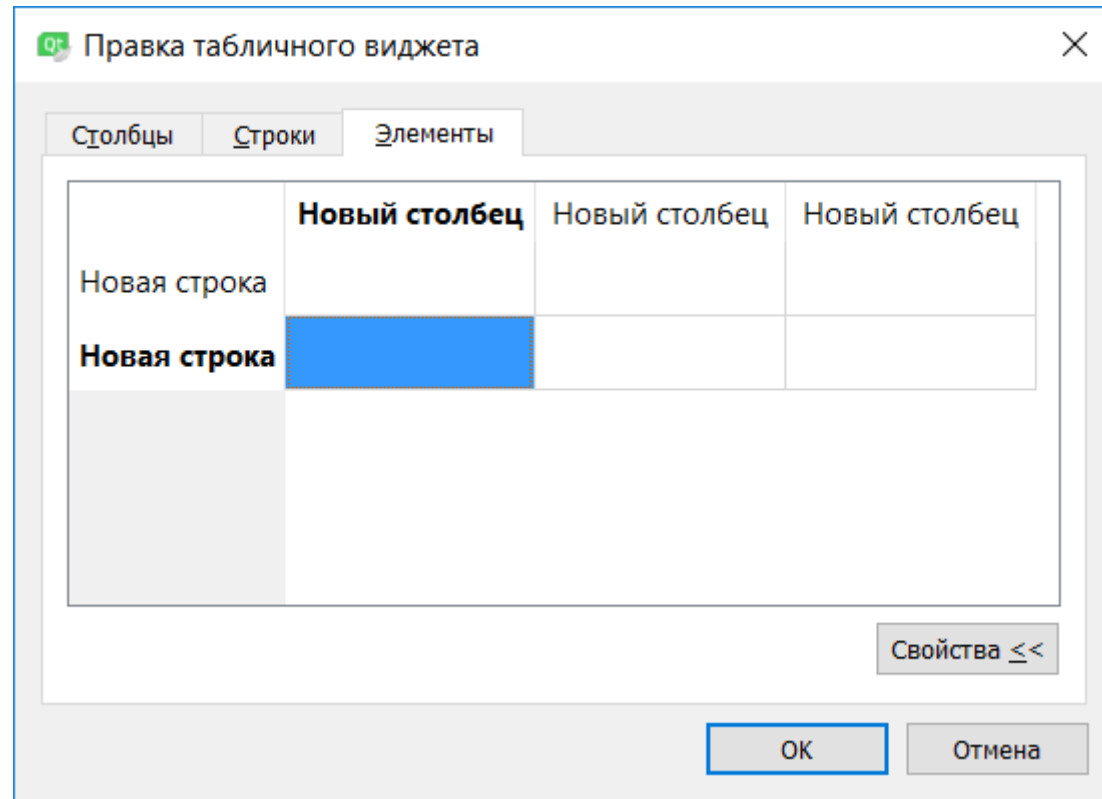
2. Размещаем на форме.

3. Создаем столбцы и строки.



Таблицы. QTableWidgetItem.

4. На вкладке «Элементы» можно задать свойства для каждого элемента таблицы



Header	
horizontalHeaderVisible	<input checked="" type="checkbox"/>
horizontalHeaderCascadingSectionResizes	<input type="checkbox"/>
horizontalHeaderDefaultSectionSize	125
horizontalHeaderHighlightSections	<input checked="" type="checkbox"/>
horizontalHeaderMinimumSectionSize	46
horizontalHeaderShowSortIndicator	<input type="checkbox"/>
horizontalHeaderStretchLastSection	<input type="checkbox"/>
verticalHeaderVisible	<input checked="" type="checkbox"/>
verticalHeaderCascadingSectionResizes	<input type="checkbox"/>
verticalHeaderDefaultSectionSize	37
verticalHeaderHighlightSections	<input checked="" type="checkbox"/>
verticalHeaderMinimumSectionSize	30
verticalHeaderShowSortIndicator	<input type="checkbox"/>
verticalHeaderStretchLastSection	<input type="checkbox"/>

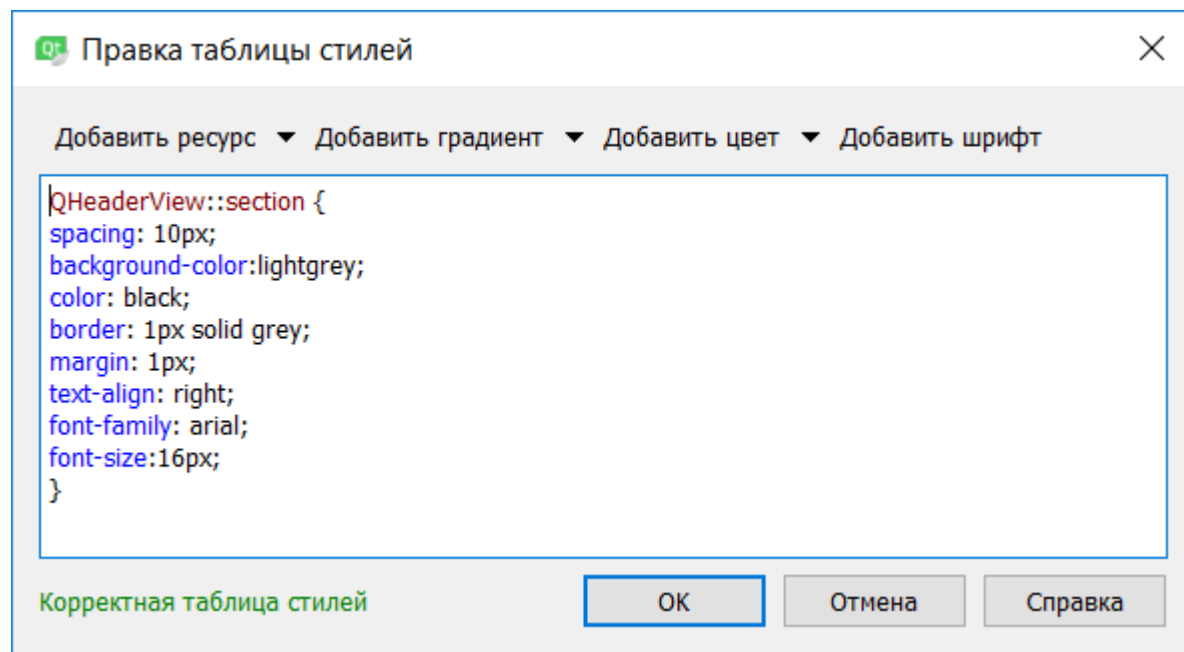
Таблицы. QTableWidgetItem.

5. В свойствах таблицы можете задать отображение заголовков столбцов и строк таблицы (`horizontalHeaderVisible = true`, `verticalHeaderVisible=true` и т.д.)

▼ Header	
<code>horizontalHeaderVisible</code>	<input checked="" type="checkbox"/>
<code>horizontalHeaderCascadingSectionResizes</code>	<input type="checkbox"/>
<code>horizontalHeaderDefaultSectionSize</code>	125
<code>horizontalHeaderHighlightSections</code>	<input checked="" type="checkbox"/>
<code>horizontalHeaderMinimumSectionSize</code>	46
<code>horizontalHeaderShowSortIndicator</code>	<input type="checkbox"/>
<code>horizontalHeaderStretchLastSection</code>	<input type="checkbox"/>
<code>verticalHeaderVisible</code>	<input checked="" type="checkbox"/>
<code>verticalHeaderCascadingSectionResizes</code>	<input type="checkbox"/>
<code>verticalHeaderDefaultSectionSize</code>	37
<code>verticalHeaderHighlightSections</code>	<input checked="" type="checkbox"/>
<code>verticalHeaderMinimumSectionSize</code>	30
<code>verticalHeaderShowSortIndicator</code>	<input type="checkbox"/>
<code>verticalHeaderStretchLastSection</code>	<input type="checkbox"/>

Таблицы. QTableWidgetItem.

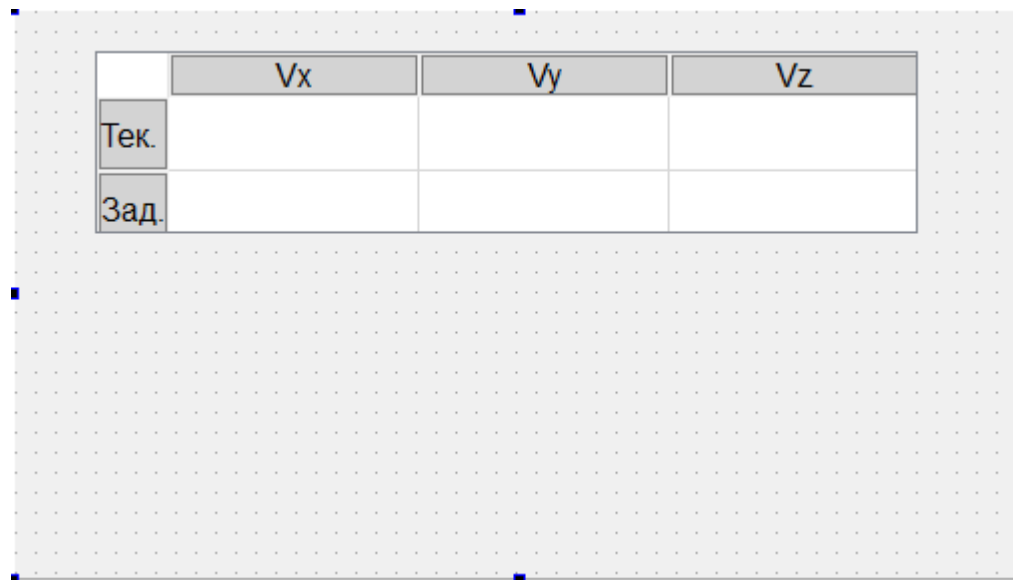
6. В свойствах таблицы в графе StyleSheet (вкладка QWidget) можно задать стиль таблицы. Например:



```
QHeaderView::section {
    spacing: 10px;
    background-   color:lightgrey;
    color: black;
    border: 1px solid grey;
    margin: 1px;
    text-align: right;
    font-family: arial;
    font-size:16px;
}
```

Таблицы. QTableWidgetItem.

7. В результате:



	Vx	Vy	Vz
Тек.			
Зад.			

Таблицы. QTableWidgetItem.

Чтобы как-то задавать/изменять значения в таблице:

2). В коде создаем QTableWidgetItem под каждый редактируемый элемент таблицы:

```
for (int i = 0; i < tblPaPosition->rowCount(); ++i)
    for (int j = 0; j < tblPaPosition->columnCount(); ++j) {
        if (tblPaPosition->item(i, j) == 0)
            tblPaPosition->setItem(i, j, new QTableWidgetItem());
        tblPaPosition->item(i, j)->setTextAlignment(Qt::AlignHCenter);
    }
```

3). Чтобы заполнить элемент таблицы нужно вызвать метод setText() для каждого элемента таблицы:

```
tblPaPosition->item(1,1)->setText(QString::number(marshValueDes));
```

*вместо (1,1) – необходимые номера строки и столбца

Таблицы. QTableWidgetItem.

Задание:

Выводить заданное значение управляющего сигнала по маршу (для этого на форму добавить слайдер) в соответствующей ячейке таблицы.

Те, у кого не получилось создать таблицу по ходу занятия могут воспользоваться формой TablesChallenge.ui.

