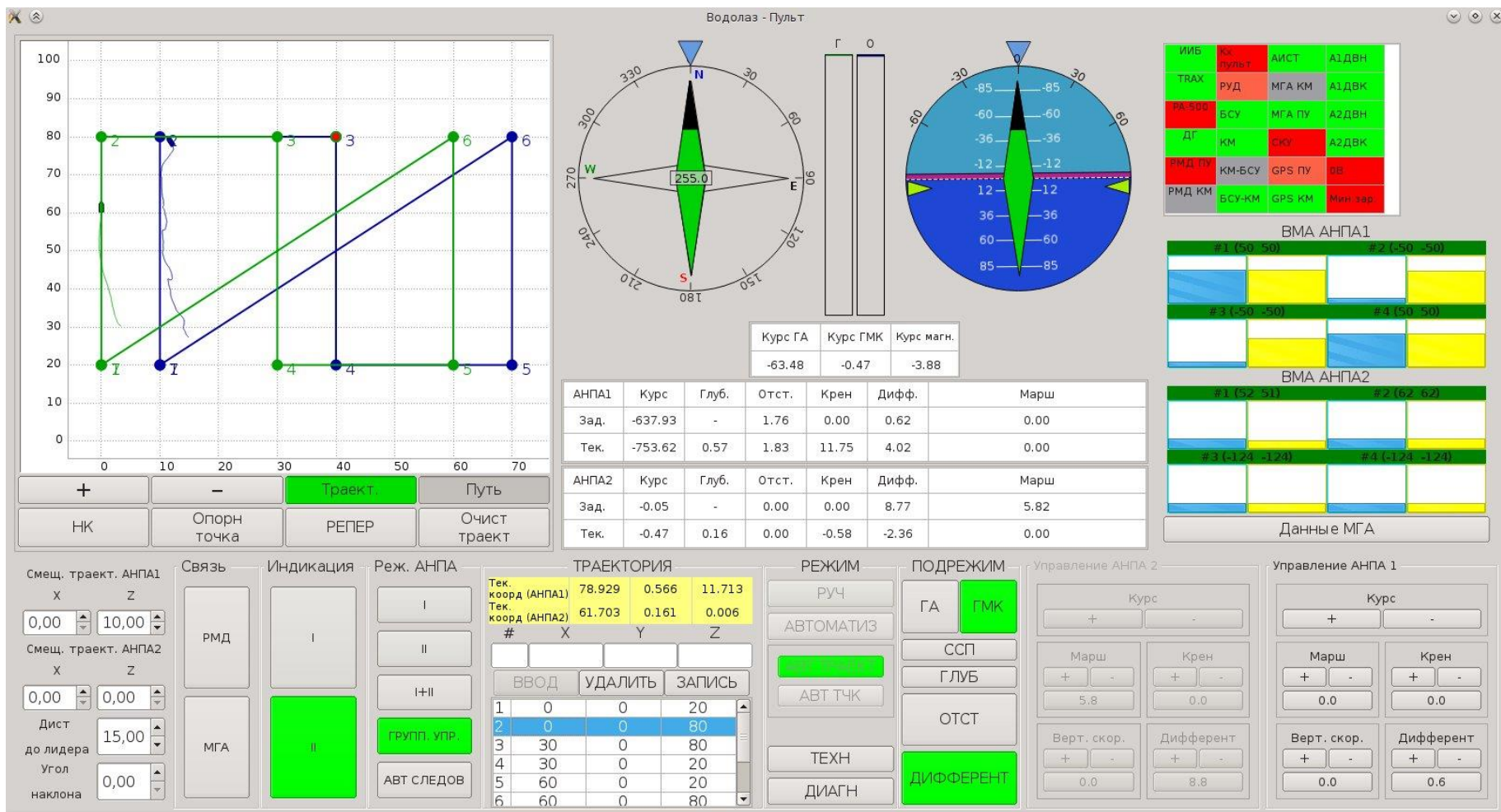


СЕМИНАР 4

Сигналы и слоты

Сигналы и слоты. Введение.



Особенности Qt. Механизм «сигнал-слот».

- Механизм «сигнал-слот» используется для коммуникации между объектами.

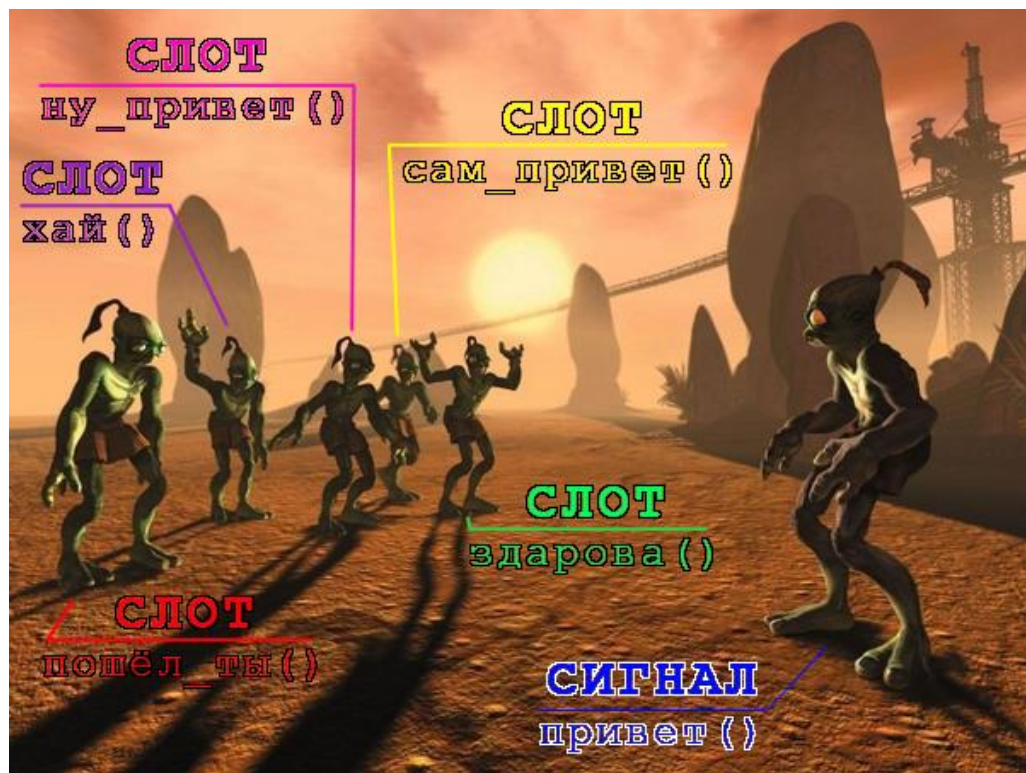


Рис. Механизм «сигнал-слот» в исполнении
мудакенов (с) wiki ИУ5

Сигнал — метод, который в состоянии осуществить пересылку сообщений.

Слот — метод, который присоединяется к сигналам (вызывается в ответ на определенный сигнал).

Мотивация

Преимущества, которые дает программисту механизм «сигнал-слот»:

1. Соединяемые сигналы и слоты абсолютно независимы и реализованы отдельно друг от друга:
 1. Упрощается декомпозиция большого проекта и параллельная разработка;
 2. Предоставлен гибкий механизм для проектирования.
2. Соединение сигналов и слотов можно производить в любой точке приложения.

Мотивация

Преимущества, которые дает программисту механизм «сигнал-слот»:

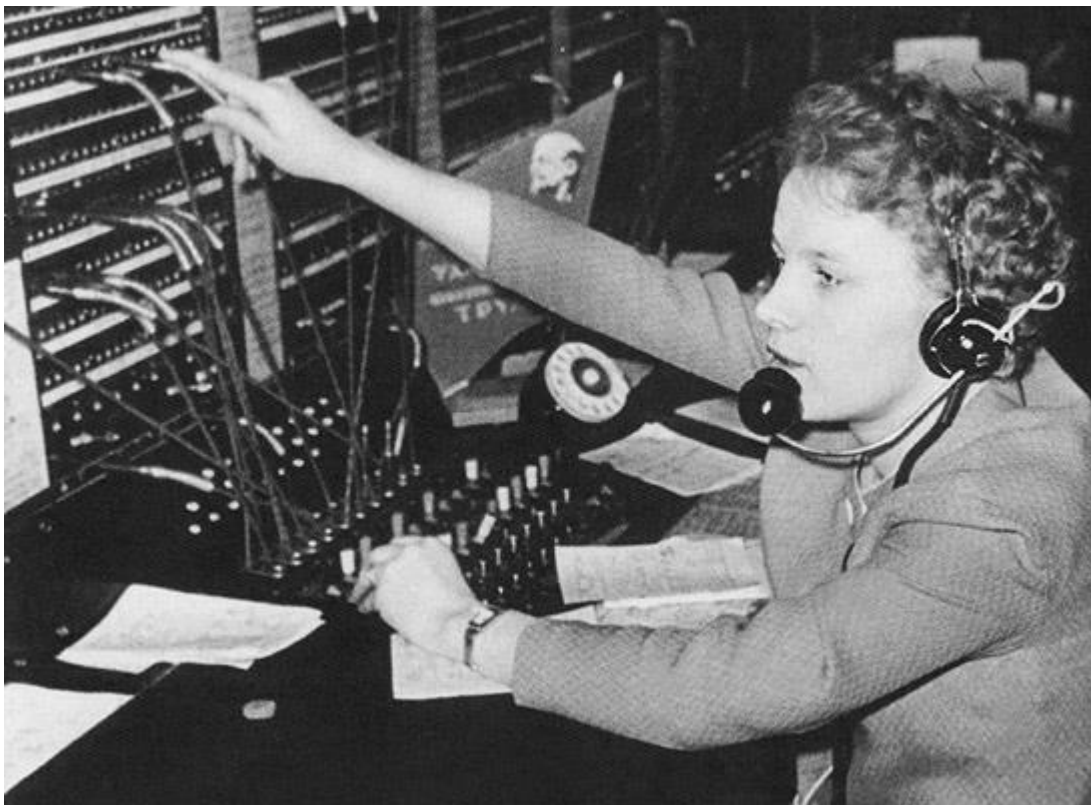
3. Соединение сигнала и слота можно осуществлять даже между объектами, которые находятся в различных потоках.
4. При уничтожении объекта происходит автоматическое разъединение всех сигнально-слотовых связей.
5. Механизм сигналов и слотов типобезопасный

Особенности Qt. Механизм «сигнал-слот».

Одному сигналу может
соответствовать много слотов.



Одному слоту может
соответствовать много сигналов.



Особенности Qt. Механизм «сигнал-слот».

- Можно соединять сигнал и сигнал!



Недостатки

- Сигналы и слоты не являются частью языка C++, поэтому требуется запуск МОС перед компиляцией программы.
- Данный механизм реализован в классе QObject.
 - Для использования сигналов и слотов, класс должен быть унаследован от QObject.
 - Класс – наследник QObject не может быть шаблонным.
 - В случае множественного наследования QObject должен быть первым.
- Для реализации механизма МОС требует записи макроса Q_OBJECT сразу при объявлении класса.
 - Нельзя использовать макрос Q_OBJECT с шаблонными классами.

Недостатки

- Сигналы и слоты медленнее чем механизм callback-ов;
- При некоторых способах формирования сигналов и слотов в процессе компиляции не производится никаких проверок: имеется ли сигнал или слот в соответствующих классах или нет; совместимы ли сигнал и слот друг с другом и могут ли они быть соединены вместе. (хотя в Qt5 реализован новый механизм, который частично устраняет этот недостаток).

Общая информация

Механизм сигналов и слотов реализован благодаря системе мета-объектов qt, поэтому:

- Классы, использующие сигналы и слоты, должны быть наследниками Qobject;
- В классах, где реализованы сигналы и слоты должен быть макрос Q_OBJECT;
- При компиляции должен запускаться moc.

Сигнал

Метод, который:

- вызывается, когда необходимо оповестить о каком-то событии;
- для вызова сигнала используется слово *emit*;
- описан в разделе signals;
- не имеет реализации в .cpp;
- не возвращает значение (тип void);
- может передавать данные, может иметь параметры по-умолчанию;
- может вызываться из любого места программы (ограничен только видимостью объекта, содержащего сигнал);
- «не знает» с какими слотами он соединен.

Сигналы и слоты. Что такое сигнал?

Сигнал объявляется
однажды и на этом всё,
ему не нужна реализация!
(Реализацию сигнала
осуществляет МОС за вас)

То, что пишете вы в своих исходниках (pultwidget.h):

```
pult_sem4/pultwidget.h  marshValueChanged(): void
1  #ifndef PULTWIDGET_H
2  #define PULTWIDGET_H
3
4  #include "ui_pultwidget.h"
5
6  class PultWidget : public QWidget, private Ui::PultWidget
7  {
8      ... Q_OBJECT
9
10     public:
11         ... explicit PultWidget(QWidget *parent = 0);
12         ... //сигнал объявляется после ключевого слова signals:
13     signals:
14         ... //метод сигнала не возвращает никаких значений
15         ... //так что перед названием метода должен быть указан тип void
16         ... void marshValueChanged();
17     };
18
19     #endif // PULTWIDGET_H
20
```

Сигналы и слоты. Что такое сигнал?

То, что дописывает MOC в (moc_pulwidget.cpp) за вас:

```
124 // SIGNAL 0
125 ▼ void PulWidget::marshValueChanged()
126 {
127     ... QObject::activate(this, &staticMetaObject, 0, Q_NULLPTR);
128 }
```


Сигналы и слоты. Как выслать сигнал?

Для вызова сигнала используется ключевое слово `emit`.

```
11 //данная конструкция приводит к вызову метода marshValueChanged()  
12 //обработка сигнала слотом будет происходить в зависимости от способа соединения  
13 void PultWidget::sendMarshSignal() {  
14     ...emit marshValueChanged();  
15 }  
16
```

Сигналы могут высылаться в любом месте программы, однако рекомендуется вызывать их из объектов, которые их содержат.

Сигналы могут высылать информацию, передаваемую в параметре.

Слоты

Методы, которые:

- могут быть соединены с сигналами;
- объявляются в классе после ключевых слов:
 - `private slots:`
 - `protected slots:`
 - **`public slots:`**
- являются обычными методами C++;
- могут быть виртуальными;
- нельзя определять как `static`;
- могут иметь параметры по-умолчанию.

Объявление слота

Слоты объявляются в
классе после
ключевых слов
`private slots:`
`protected slots:`
`public slots:`

pultwidget.h:

```
5
6 class PultWidget : public QWidget, private Ui::PultWidget
7 {
8     Q_OBJECT
9
10 public:
11     explicit PultWidget(QWidget *parent = 0);
12     //сигнал объявляется после ключевого слова signals:
13 signals:
14     //[[1] метод сигнала не возвращает никаких значений
15     //так что перед названием метода должен быть указан тип void
16     void marshValueChanged();
17 public:
18     //тестовый метод, который генерит сигнал marshValueChanged()
19     void sendMarshSignal();
20
21     //[[2] Слот объявляется после ключевого слова public slots:
22 public slots:
23     void addMarshValue();
24     //[[3] Создадим переменные, для хранения заданных значений и
25     //величин приращения значений
26 private:
27     //переменная, которая хранит заданное значение по маршу
28     double marshValue;
29     //переменная, которая хранит значение шага изменения марша
30     double deltaMarshValue;
31
```

Что такое слот?

Реализация функции слота в pultwidget.cpp:

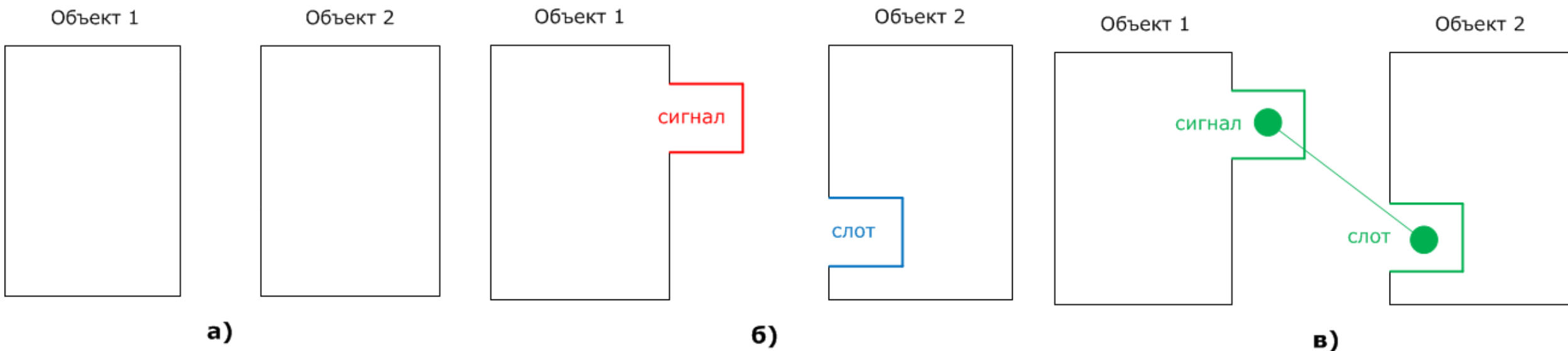
```
1  #include "pultwidget.h"
2
3  PultWidget::PultWidget(QWidget *parent) :
4  ▼  ....QWidget(parent)
5  {
6  ....setupUi(this);
7  ....//проинициализируем значения
8  ....marshValue = 0;
9  ....deltaMarshValue = 0.1;
10 }
11
12 ▼ void PultWidget::addMarshValue()
13 {
14 ....//при нажатии на кнопку "+" увеличиваем значение marshValue на deltaMarshValue
15 ....marshValue += deltaMarshValue;
16 ....//выведем текущее заданное значение на кнопке btnResetMarsh
17 ....btnResetMarsh->setText(QString::number(marshValue));
18 }
```

Как соединить сигнал и слот?

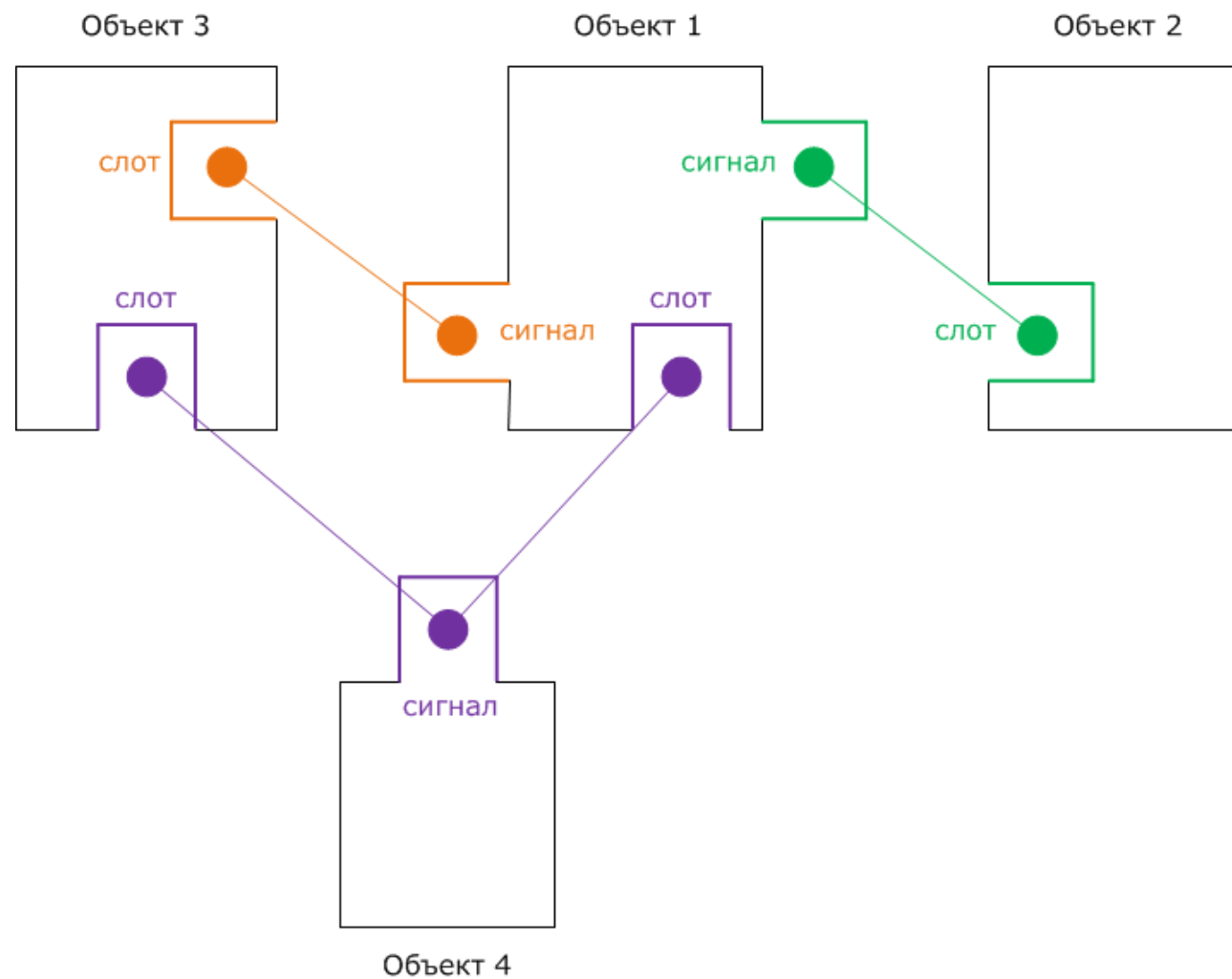
Связь между объектами устанавливается следующим образом: у одного объекта должен быть сигнал, а у второго – слот.

Чтобы соединить два объекта, нужно:

- создать у одного сигнал, а у второго слот;
- соединить сигнал первого и слот второго.



Сигналы и слоты. Как соединить сигнал и слот?



Сигналы и слоты. Как соединить сигнал и слот?

1. Метод `connect()`:
 1. С использованием макросов `SIGNAL()` и `SLOT()`;
 2. С использованием указателей на функции;
 3. Соединение сигналов с лямбдами.
2. Автоматическое соединение сигналов и слотов;
3. Использование `signal mapper`.

1. Соединение с использованием макросов

```
QObject::connect (const QObject* sender,  
                  const char*      signal,  
                  const QObject*  receiver,  
                  const char*      slot,  
                  Qt::ConnectionType type = Qt::AutoConnection  
                  );
```

`sender` – указатель на объект, отправляющий сигнал;

`signal` – сигнал, с которым осуществляется соединение, причем имя сигнала заключается в специальный макрос `SIGNAL(method())`;

`receiver` – указатель на объект, который имеет слот для обработки сигнала;

`slot` – слот, который вызывается при получении сигнала. Прототип слота заключается в специальный макрос `SLOT(method())`;

`type` – управляет режимом обработки*.

Qt::ConnectionType

- Имеется три возможных режима обработки:
- 1- Qt::DirectConnection – сигнал обрабатывается сразу вызовом соответствующего метода слота,
- 2- Qt::QueuedConnection – сигнал преобразуется в событие и ставится в общую очередь для обработки,
- 3- Qt::AutoConnection – автоматический режим, который действует следующим образом: если отсылающий сигнал объект находится в одном потоке с принимающим его объектом, то устанавливается режим Qt::DirectConnection, в противном случае – режим Qt::QueuedConnection. Этот режим определен в методе connection() по умолчанию.

Пример

Пример соединения сигнала и слота:

```
2
3 PultWidget::PultWidget(QWidget *parent) :
4     QWidget(parent)
5 {
6     setupUi(this);
7     //проинициализируем значения
8     marshValue = 0;
9     deltaMarshValue = 0.1;
10
11     // [4] при нажатии на кнопку QPushButton, она генерит сигнал clicked()
12     // соединим кнопку "+" со слотом addMarshValue():
13     // указатель на объект, содержащий слот addMarshValue – это this
14     connect(btnPlusMarsh, SIGNAL(clicked()), this, SLOT(addMarshValue()));
15     // в принципе синтаксис метода connect, когда происходит соединение со слотом,
16     // принадлежащим классу, в котором происходит connect можно упростить и записать так:
17     // connect(btnPlusMarsh, SIGNAL(clicked()), SLOT(addMarshValue()));
18 }
```


Соединение с использованием указателей

```
QObject::connect (const QObject*           sender,  
                  const MetaMethod&        signal,  
                  const QObject*           receiver,  
                  const MetaMethod&        slot,  
                  Qt::ConnectionType type = Qt::AutoConnection  
                  );
```

Отличие от предыдущего метода в том, что сигнал и метод передаются через указатели на методы сигналов и слотов классов напрямую и без использования макросов `SIGNAL(method())` и `SLOT(method())`.

Для примера создадим слот под «-» значений

```
24  ....
25  .... // [5] Создадим слот для уменьшения заданных значений:
26  .... void decMarshValue();
27  ....
28  ....
```

```
28
29  ▼ void PultWidget::decMarshValue()
30  {
31  .... marshValue-=deltaMarshValue;
32  .... btnResetMarsh->setText(QString::number(marshValue));
33  }
34
```

Пример

Пример соединения сигнала и слота альтернативным способом:

```
18 .....  
19 ..... // [6] . соединим сигнал и слот по второй схеме  
20 ..... connect(btnMinusMarsh, &QPushButton::clicked, this, &PultWidget::decMarshValue);
```

Особенность метода в том, что если вы ошибетесь с названием слотов и сигналов, ваша ошибка будет выявлена сразу в процессе компиляции программы.

Недостаток – то, что необходимо явно указывать имена классов для сигнала и слота и следить за совпадением их параметров.

Сигналы и слоты. Соединение с лямбдами.

1. `[capture] (params) mutable exception attribute -> ret { body }`
2. `[capture] (params) -> ret { body }`
3. `[capture] (params) { body }`
4. `[capture] { body }`

capture - определяет, какие символы, видимые в области объявления функции, будут видны внутри тела функции.

params - Список параметров, как в объявлении функции

ret - Возвращаемый тип. Если нет, то он выводится из возвращаемого значения (или `void`, если функция не возвращает никакого значения)

Сигналы и слоты. Соединение с лямбдами.

```
[ capture ] ( params ) { body }
```

capture - определяет, какие символы, видимые в области объявления функции, будут видны внутри тела функции.

Список символов может быть передан следующим образом:

[a,&b] где *a* захвачена по значению, а *b* захвачена по ссылке.

[this] захватывает указатель **this** по значению.

[&] захват всех символов по ссылке

[=] захват всех символов по значению

[] ничего не захватывает

params - Список параметров, как в объявлении функции

ret - Возвращаемый тип. Если нет, то он выводится из возвращаемого значения (или `void`, если функция не возвращает никакого значения)

Пример

```
1  
2  
3  ▾ ... // [7] - соединим кнопку Reset с использованием лямбда - функции  
4 ... connect(btnResetMarsh, &QPushButton::clicked, [this]() {  
5 ...     marshValue = 0;  
6 ...     btnResetMarsh->setText("0");  
7 ... });
```

Сигналы и слоты. Автоматическое соединение.

```
void on<object name>_<signal_name>(<signal parameters>)
```

```
void Widget::on_btnPlus_clicked() {
```

Продвинутое использование сигналов и слотов

QSignalMapper – класс, который позволяет соединять один слот со многими, но различными сигналами.

Класс позволяет:

- собрать сигналы, не передающие данных;
- переслать их слоту, при этом передав идентификатор источника сигнала:
 - int id;
 - QString текст;
 - Указатель на объект QWidget, QObject.

Продвинутое использование сигналов и слотов

- Установка соответствия сигнал – идентификатор (mapping) осуществляется методом `setMapping()`;
- При появлении сигнала одного из объектов, объект `QSignalMapping` генерирует сигнал `mapped`(выбранный тип идентификатора)

Пример 1

```
23 public slots:
24     ***void addMarshValue();
25
26     ***//[5] Создадим слот для уменьшения заданных значений:
27     ***void decMarshValue();
28
29     ***//[9] создадим слот под обработку кнопок изменения дифферента
30     ***void changePitchValue(QString source);
31
32
33 private:
34     ***//[7] Добавим новые переменные
35     ***//переменная, которая хранит заданное значение по маршу
36     ***double marshValue, pitchValue;
37     ***//переменная, которая хранит значение шага изменения марша
38     ***double deltaMarshValue, deltaPitchValue;
39
40     ***//[8] создадим указатель на объект QSignalMapper и не забудем
41     ***//сделать #include <QSignalMapper> в самом начале
42     ***QSignalMapper *signalMapper = nullptr;
43
44     };
```

Пример 1

```
5 {
6     ***setupUi(this);
7     ***//проинициализируем значения
8     ***//[10]
9     ***marshValue=pitchValue=0;
10    ***deltaMarshValue=deltaPitchValue=0.1;
11    ***//создадим объект signalMapper (выделим под него память)
12    ***signalMapper=new QSignalMapper(this);
13    ***//[4] при нажатии на кнопку QPushButton, она генерит сигнал clicked()
14    ***//соединим кнопку "+" со слотом addMarshValue():
15    ***//указатель на объект, содержащий слот addMarshValue -- это this
16    ***connect(btnPlusMarsh, SIGNAL(clicked()), this, SLOT(addMarshValue()));
17    ***//в принципе синтаксис метода connect, когда происходит соединение со слотом,
18    ***//принадлежащим классу, в котором происходит connect можно упростить и записать так:
19    ***//connect(btnPlusMarsh, SIGNAL(clicked()), SLOT(addMarshValue()));
20
21    ***//[6] соединим сигнал и слот по второй схеме
22    ***connect(btnMinusMarsh, &QPushButton::clicked, this, &PultWidget::decMarshValue);
23
24    ***//[7] соединим кнопку Reset с использованием лямбда-функции
25    > ***connect(btnResetMarsh, &QPushButton::clicked, [this]() { ...
26    ***});
27    ***//[11]
28    ***//устанавливаем соответствие сигналов и идентификаторов
29    ***signalMapper->setMapping(btnPlusPitch, "+");
30    ***signalMapper->setMapping(btnMinusPitch, "-");
31    ***signalMapper->setMapping(btnResetPitch, "Reset");
32
33    ***connect(signalMapper, SIGNAL(mapped(QString)), this, SLOT(changePitchValue(QString)));
34
35    ***connect(btnPlusPitch, SIGNAL(clicked()), signalMapper, SLOT(map()));
36    ***connect(btnMinusPitch, SIGNAL(clicked()), signalMapper, SLOT(map()));
37    ***connect(btnResetPitch, SIGNAL(clicked()), signalMapper, SLOT(map()));
38
39 }
```

```
57
58 void PultWidget::changePitchValue(QString source)
59 {
60     ***if (source == "+") pitchValue+=deltaPitchValue;
61     ***else if (source == "-") pitchValue-=deltaPitchValue;
62     ***else if (source == "Reset") pitchValue=0;
63     ***btnResetPitch->setText(QString::number(pitchValue));
64 }
65
```

Пример 2

```
35  ....// [13] изменяем тип connect : вместо:
36  ....// connect(signalMapper, SIGNAL(mapped(QString)), this, SLOT(changePitchValue(QString)));
37  ....// используем соединение через указатели:
38  ....connect(signalMapper, (static_cast<void(QSignalMapper::*)(const QString &)>(&QSignalMapper::mapped)),
39  ....this, &PultWidget::changePitchValue);
40  ....
```

Практическая часть 1.

Соединить сигналы и слоты таким образом, чтобы:

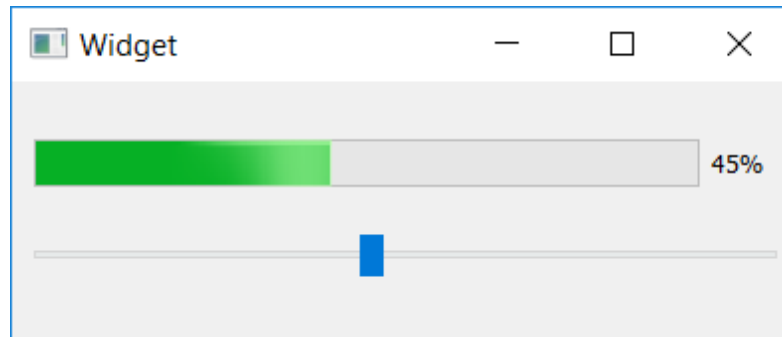
1. При нажатии на кнопку «+» заданная скорость, отображаемая на кнопке сброса увеличивалась;
 2. При нажатии на кнопку «-» заданная скорость – уменьшалась;
 3. При нажатии на кнопку сброса заданная скорость – обнулялась.
- При этом использовать все методы соединения сигналов и слотов.

Практическая часть 2.

Для тех, кто подготовил панель управления ПА или первые прототипы своих пультов:

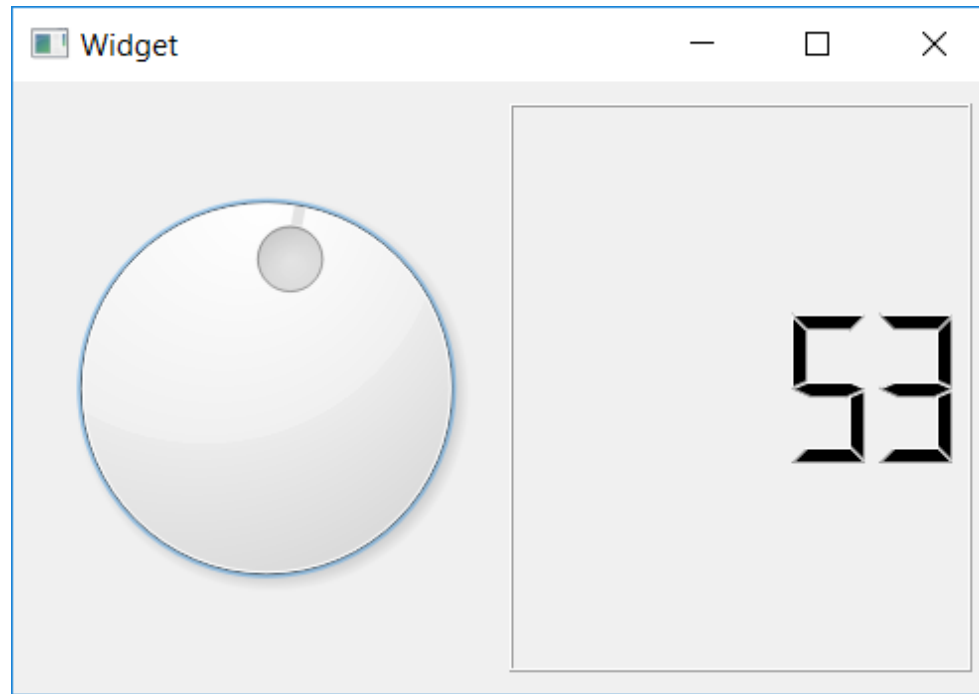
- соединить задатчики движения ПА с виджетами, выводящими заданные значения каждым из рассмотренных способов соединения сигналов и слотов.
- Для тех, кто этого не сделал – соединить сигналы и слоты в тестовых проектах.

Задание 1



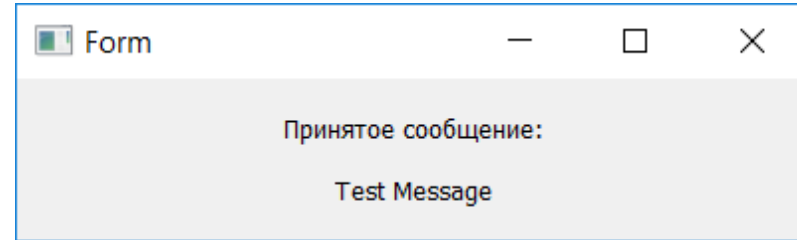
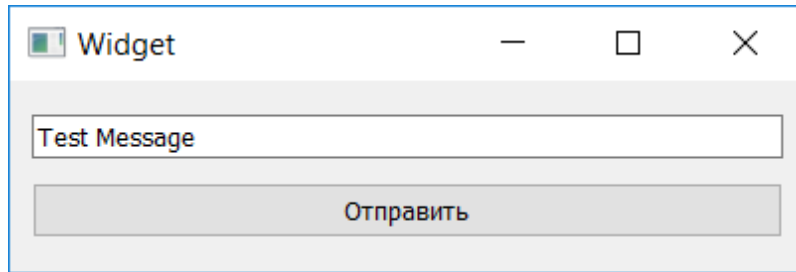
1. При изменении положения слайдера, должно соответственно измениться заполнение индикатора процесса
2. Подсказки:
 - Сигнал изменения положения задатчика dial `QSlider::valueChanged(int)`
 - Слот, который может изменить заполнение индикатора процесса – `QProgressBar::setValue(int)`

Задание 2



1. При изменении положения элемента dial, значение положения задающего элемента, должно отобразиться на цифровом индикаторе lcdNumber
2. Подсказки:
 - Сигнал изменения положения задатчика dial `QDial::valueChanged(int)`
 - Слот, который может вывести значение на цифровом индикаторе - `QLCDNumber::display(int)`- Вывести текст на метке можно с помощью

Задание 3



1. При нажатии кнопки «Отправить» сообщение, содержащееся в строке ввода окна «Widget» должно быть отображено в окне Form.
2. Подсказки:
 - Сигнал нажатия кнопки `QPushButton::clicked()`
 - Получить текст из строки ввода можно с помощью метода `QLineEdit::text()`
 - Вывести текст на метке можно с помощью метода `QLabel::setText(QString)`