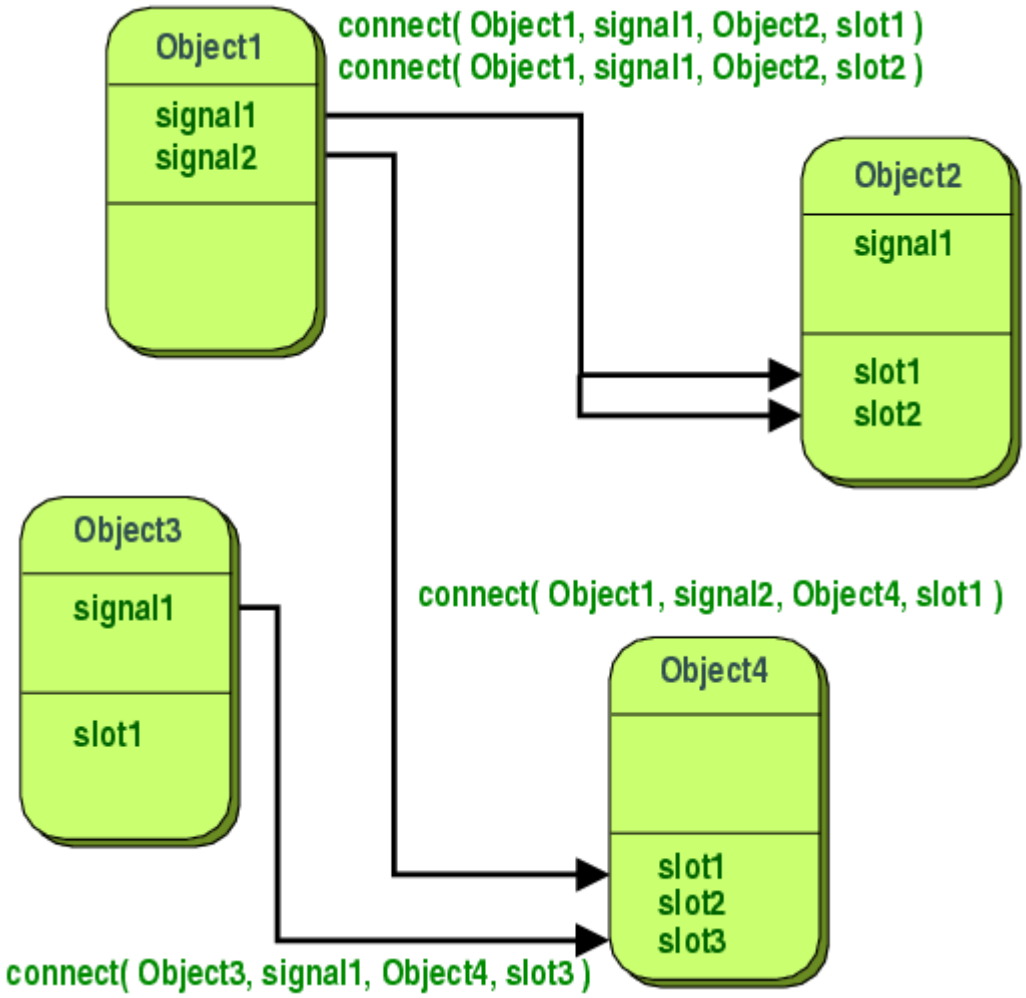
Обработка событий. Слоты. Сигналы. Ивенты.

Механизм **сигналов и слотов** является расширением языка программирования C++ в Qt5 [1], который используется для установления связи между объектами. Если происходит какое-либо определенное событие, то при этом может генерироваться сигнал. Данный сигнал попадает в связанный с ним слот.

В свою очередь, **слот** — это обычный метод в языке C++, который присоединяется к **сигналу**; он вызывается тогда, когда генерируется связанный с ним сигнал. Все графические приложения управляются событиями: всё, что происходит в приложении является результатом обработки тех или иных событий. Они являются важной частью любой графической программы. В большинстве случаев события генерируются пользователем приложения, но они также могут быть сгенерированы и другими средствами, например, подключением к интернету, оконным менеджером или таймером.



Пример (signalsSlots)

Рассмотрим простой пример взаимодействия объектов посредством сигналов:

counter.h - создадим обычный класс "счетчика"

```
#ifndef COUNTER_H
#define COUNTER_H
#include <QObject>

class Counter : public QObject
{
   Q_OBJECT

private:

   int m_value;

public:

   Counter() {
       m_value = 0;
```

```
    }
  // возвращение текущего значения переменной "m_value"
  // для безопасности делаем это методом, который возвращает значение
  // чтобы из вне нельзя было изменить данное значение
   int value() const {
       return m_value;
   }

public slots:

   void setValue(int value); // помещаем в слот метод сеттера значения
                             //(присвоение значения)

signals:
  // в виде сигнала отслеживаем событие изменения значения
   void valueChanged(int newValue);


};

#endif // COUNTER_H
```

counter.cpp

```
#include "counter.h"
#include <QDebug>

void Counter::setValue(int value)
{
   // Чтобы избежать бесконечного вызова метода необходима проверка
   if (value != m_value) {
     m_value = value;
     qDebug() << "Value is changed: " << m_value;
     // emit вырабатывает сигнал valueChanged() объекта
     //с новым значением в качестве аргумента
     emit valueChanged(value);
   }

}
```

main.cpp

```
#include <iostream>
#include <counter.h>
#include <QDebug>


using namespace std;

int main()
{
```

```
    Counter counter_one, counter_two;
    // подключение сигнала от отбъекта "counter_one" к слоту
    // "counter_two"
    // т.е. при каждом изменении значения объекта "counter_one" будет
    // менять и значение "counter_two"
    QObject::connect(&counter_one, SIGNAL(valueChanged(int)),
                     &counter_two, SLOT(setValue(int)));

    qDebug() << "counter_one and counter_two:";
    qDebug() << counter_one.value() << counter_two.value();

    counter_one.setValue(12);

    qDebug() << "counter_one and counter_two:";
    qDebug() << counter_one.value() << counter_two.value();

    counter_two.setValue(45);

    qDebug() << "counter_one and counter_two:";
    qDebug() << counter_one.value() << counter_two.value();

    return 0;
}
```

Более реальный пример (QWidget & Button):

Добавляем в проект класс **QuitWidget**, который реализует использование сигналов для закрытия окна приложения.

quitwidget.h

```
#ifndef QUITWIDGET_H
#define QUITWIDGET_H
#include <QWidget>


class QuitWidget:public QWidget
{
public:
    QuitWidget(QWidget *parent = nullptr);

    ~QuitWidget();
};

#endif // QUITWIDGET_H
```

quitwidget.cpp

```
#include "quitwidget.h"
#include <QPushButton>
#include <QApplication>

QuitWidget::QuitWidget(QWidget *parent)
    : QWidget(parent)
{

    QPushButton *quitBtn = new QPushButton("Quit", this);
    connect(quitBtn, &QPushButton::clicked, qApp, &QApplication::quit);
}

QuitWidget::~QuitWidget()
{
}
```

Метод **connect()** соединяет сигнал со слотом. Когда мы нажимаем на кнопку Quit, генерируется сигнал щелчка кнопки мыши. **qApp** — это глобальный указатель на объект нашего приложения. Он определяется в заголовочном файле **QApplication**. Метод **quit()** вызывается при появлении сигнала щелчка мышкой.

Немного изменяем файл **main.cpp:**

```
#include <iostream>
#include <counter.h>
#include <QDebug>
#include <quitwidget.h>
#include <QApplication>

//using namespace std;

int main(int argc, char *argv[])
{
    // First part
//    Counter counter_one, counter_two;

//    QObject::connect(&counter_one, SIGNAL(valueChanged(int)),
//                &counter_two, SLOT(setValue(int)));

//    qDebug() << "counter_one and counter_two:";
//    qDebug() << counter_one.value() << counter_two.value();

//    counter_one.setValue(12);

//    qDebug() << "counter_one and counter_two:";
//    qDebug() << counter_one.value() << counter_two.value();

//    counter_two.setValue(45);

//    qDebug() << "counter_one and counter_two:";
//    qDebug() << counter_one.value() << counter_two.value();
```

```
    // Quit button
    QApplication a(argc, argv);
    QuitWidget quit;
    quit.resize(300, 300);
    quit.setWindowTitle("Quit");
    quit.show();

    return a.exec();
}
```

Обработка нажатий клавиатуры

В следующем примере мы рассмотрим способ реагирования на нажатие клавиатуры, в частности, будем менять текст внутри нашего окна (Виджета).

В **quitwidget.h** добавляем пару строк, связанных с текстовым окном и методом реагирования на нажатие клавиатуры.

```
#ifndef QUITWIDGET_H
#define QUITWIDGET_H
#include <QWidget>
#include <QLabel>
#include <QKeyEvent>
#include <QString>


class QuitWidget:public QWidget
{
private:
    QLabel *label;
    QString text = "Hello";
public:
    QuitWidget(QWidget *parent = nullptr);

    ~QuitWidget();

    // метод реагирования на нажатие клавиатуры, используя бибилотеку QKeyEvent
    void keyPressEvent(QKeyEvent *e);
};

#endif // QUITWIDGET_H
```

Виджет QLabel [3] используется для отображения текста или изображения. Текст вводится с использованием библиотеки **QString** [4].

quitwidget.cpp

```
#include "quitwidget.h"
#include <QPushButton>
#include <QApplication>

QuitWidget::QuitWidget(QWidget *parent)
    : QWidget(parent)
{

    QPushButton *quitBtn = new QPushButton("Quit", this);
    connect(quitBtn, &QPushButton::clicked, qApp, &QApplication::quit);


    label = new QLabel(text, this);
    label->move(10, 50);

}

void QuitWidget::keyPressEvent(QKeyEvent *event){

    label->setText(event->text());
}

QuitWidget::~QuitWidget()
{
}
```

Обработка событий (QMoveEvent)

Класс **QMoveEvent** содержит параметры событий, возникающих при перемещении виджета. В следующем примере мы реагируем на событие перемещения, затем определяем текущие координаты ***x*** и ***y*** верхнего левого угла клиентской области окна и устанавливаем эти значения в заголовок окна.

quitwidget.h проебражается в:

```
#ifndef QUITWIDGET_H
#define QUITWIDGET_H
#include <QWidget>
#include <QLabel>
#include <QKeyEvent>
#include <QString>
#include <QMoveEvent>


class QuitWidget:public QWidget
{
private:
    QLabel *label;
```

```
    QString text = "Hello";

    QLabel *moveLabel;
public:
    QuitWidget(QWidget *parent = nullptr);

    ~QuitWidget();

    void keyPressEvent(QKeyEvent *e);
    void moveEvent(QMoveEvent *e);
};

#endif // QUITWIDGET_H
```

quitwidget.cpp

```
#include "quitwidget.h"
#include <QPushButton>
#include <QApplication>
#include <QString>

QuitWidget::QuitWidget(QWidget *parent)
    : QWidget(parent)
{

    QPushButton *quitBtn = new QPushButton("Quit", this);
    connect(quitBtn, &QPushButton::clicked, qApp, &QApplication::quit);


    label = new QLabel(text, this);
    label->move(10, 50);

    moveLabel = new QLabel("Move info", this);
    moveLabel->move(10, 30);

}

void QuitWidget::keyPressEvent(QKeyEvent *event){

    label->setText(event->text());
}

void QuitWidget::moveEvent(QMoveEvent *event){
    int x = event->pos().x();
    int y = event->pos().y();
    QString position = QString::number(x) + ", " + QString::number(y);
    moveLabel->setText(position);
}

QuitWidget::~QuitWidget()
{
}
```

Задание на лабораторную работу

1) Изменить код программы для **обработки нажатий клавиатуры** так, чтобы при нажатии на любую клавишу текст добавлялся в **label** и обновлялся на экране.

2) Доработайте программу, чтобы она выводила текущее время (с помощью QTimer) в области названия приложенияю

Исходный код программы находится по ссылке [2].

3) Создайте собственную ветку программы (git). Решением будет ссылка на репозиторий с модифицированным приложением.

Литературы

[1] https://doc.qt.io/qt-5/signalsandslots.html

[2] https://github.com/fzybot/QtLearning/tree/main/signalsSlots

[3] https://doc.qt.io/qt-5/qlabel.html

[4] https://doc.qt.io/qt-5/qstring.html