8/15/2011

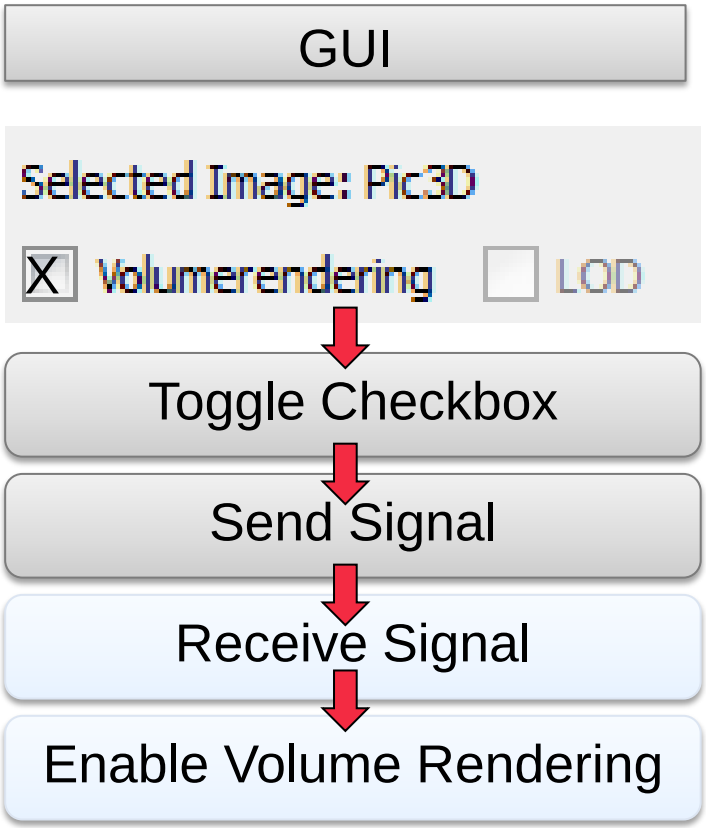
Signal & Slots in Qt

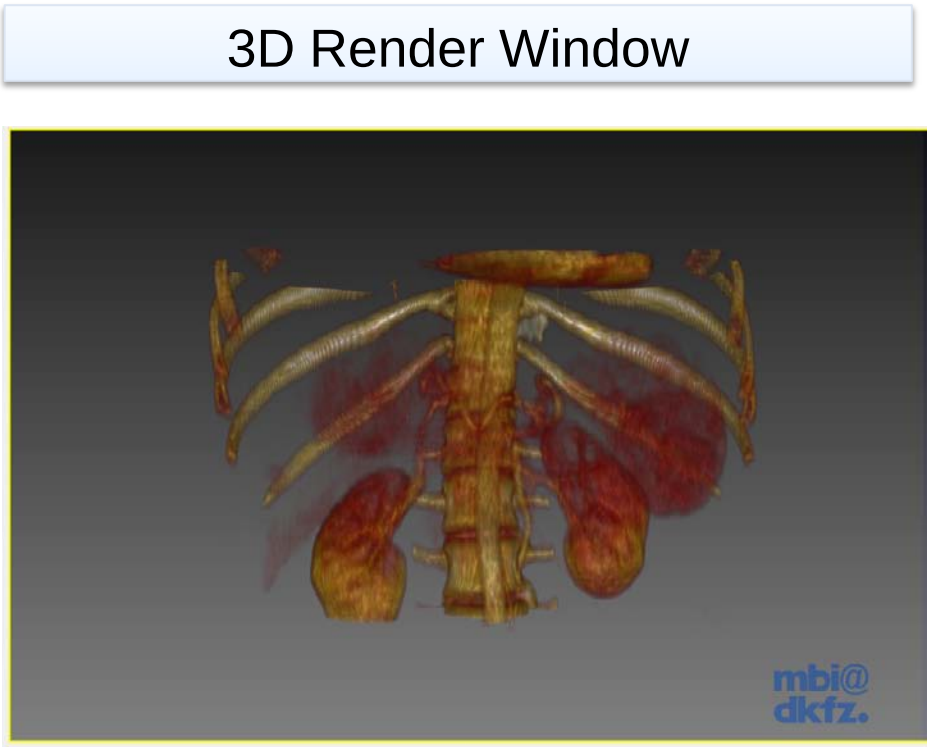
Diana Wald

dkfz.
GERMAN
CANCER RESEARCH CENTER
IN THE HELMHOLTZ ASSOCIATION

Signal & Slots

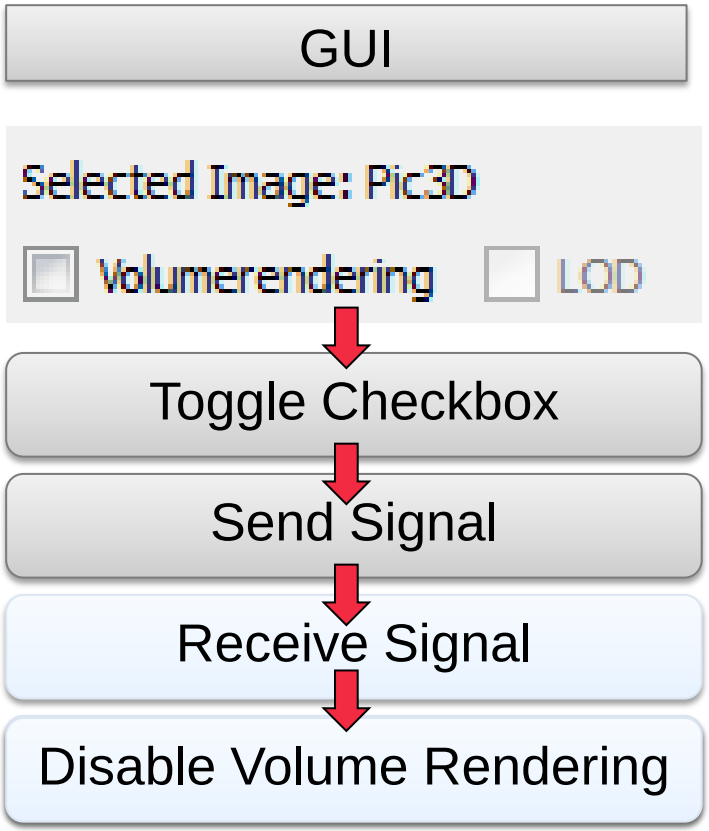
- Central feature in Qt
- Used for communication between objects

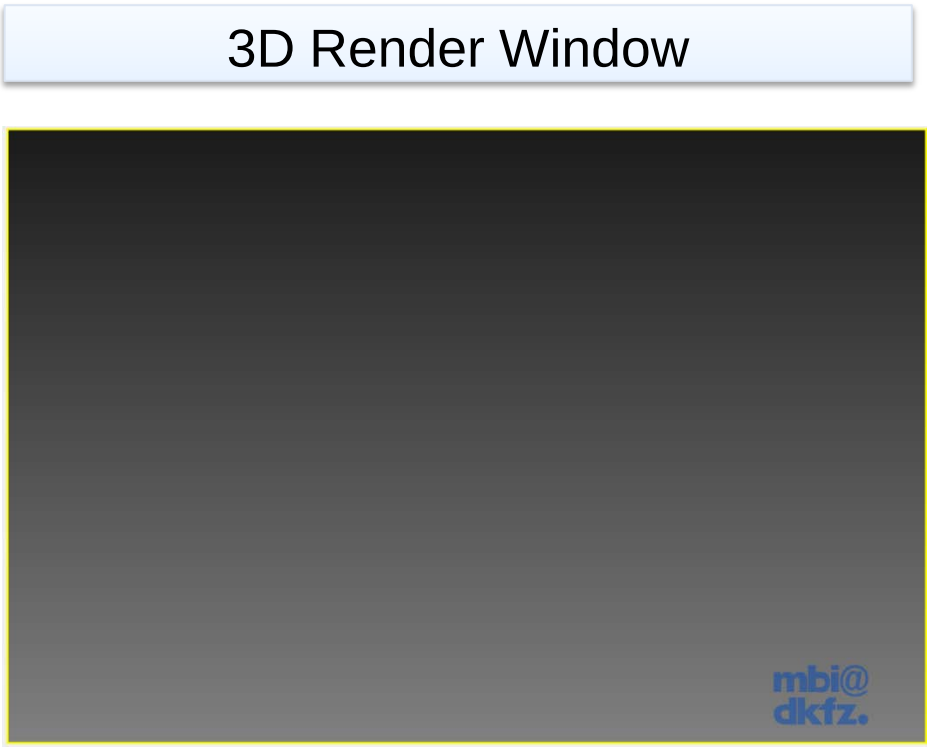




Signal & Slots

- Central feature in Qt
- Used for communication between objects





Signals

- Signals are emitted by an object when its internal state has changed in some way that might be interesting to the object's client or owner

- Qt's widgets have many predefined signals
  - QCheckBox:

Signals

void **stateChanged** (int *state*)

- 4 signals inherited from QAbstractButton
- 1 signal inherited from QWidget
- 1 signal inherited from QObject

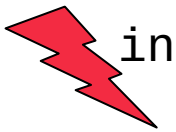
  - QAbstractButton:

void **clicked** (bool *checked* = false)
void **pressed** ()
void **released** ()
void **toggled** (bool *checked*)

Signals

- Own signals can be defined in subclasses of Qt widgets
- Only the class that defines a signal and its subclasses can emit the signal
- If several slots are connected to one signal, the slots will be executed one after the other in the order they have been connected
- Signals can never have return types

```
signals:
  void SignalSegmentationFinished()
  int SignalSegmentationFinished()
```



Slots

- A slot is a function that is called in response to a particular signal
- Slots are normal C++ functions and can be called normally; their only special feature is that signals can be connected to them
- Qt's widget have many pre-defined slots
  - QCheckBox:

    - 5 public slots inherited from QAbstractButton
    - 19 public slots inherited from QWidget
    - 1 public slot inherited from QObject

  - QAbstractButton:

    ```
    void animateClick ( int msec = 100 )
    void click ()
    void setChecked ( bool )
    void setIconSize ( const QSize & size )
    void toggle ()
    ```

- But it is common to define own slots in subclasses of Qt widgets
  e.g. Volume Visualization: `OnEnableRendering(bool)`

Signal/Slot mechanism

Flexibility

- Each class can define any number of new signals and slots
- Sent signals can have any number of arguments of any type
- A signal can be connected to several slots
- A slot can receive messages from multiple signals from different objects
- When you delete a QObjects in the destructor all connections between signals and slots will be deleted

Disadvantage

- Additional compilation with moc
- Signal/Slots are a little bit slower than usual callback function calls

Connect

- To connect the signal to the slot, the Qobject::connect() and the SIGNAL() and SLOT() macro is used

```
connect( const Object *sender, SIGNAL(const char* signal),
         const Object *receiver, SLOT(const char* slot));
```

Example: Volume Visualization

Predefined Signal of QAbstractButton

```
connect( m_Controls->m_EnableRenderingCB, SIGNAL( toggled(bool) ),
         this, SLOT( OnEnableRendering(bool) ));
```

QmitkVolumeVisualizationView

Self-defined slot

```
void QmitkVolumeVisualizationView::OnEnableRendering(bool state)
{
  if(m_SelectedNode.IsNull())
    return;

  m_SelectedNode->SetProperty("volumerendering",mitk::BoolProperty::New(state));
  UpdateInterface();
  mitk::RenderingManager::GetInstance()->RequestUpdateAll();
}
```

Connect

- The signature of a signal must match the signature of the receiving slot
- Slots can have a shorter signature than the signal because it can ignore arguments

```
connect(sender, SIGNAL(destroyed(QObject*)), this, SLOT(OnDestroyed(Qbject*)));

connect(sender, SIGNAL(destroyed(QObject*)), this, SLOT(OnDestroyed()));

connect(sender, SIGNAL(destroyed()), this, SLOT(OnDestroyed()));

connect(sender, SIGNAL(destroyed()), this, SLOT(OnDestroyed(QObject*)));
```

The last slot expects a Qobject that will not be send by the signal → Runtime error

- When a signal is emitted, the slots connected to it are usually executed immediately, just like a normal function call

Disconnect / block Signal

Disconnect

- A connection can be deleted by QObject::disconnect

```
disconnect( const Object *sender, SIGNAL(const char* signal),
            const Object *receiver, SLOT(const char* slot));
```

- OR through destroying one of the two objects

Block Signals

- Transmission of signals can be prevented by calling:

```
blockSignal(TRUE)
```

- The block can be canceled by:

```
blockSignal(FALSE)
```

Further properties of signal/slots

- Qt provides the QObject::sender() function, which returns a pointer to the object that sent the signal

  Note: if the slot was not activated by a signal, the return is undefined

- Signals and slots are loosely coupled: A class which emits a signal neither knows nor cares which slots receive the signal

- It is possible to connect a signal directly to another signal

Style Guide for Signal/Slots

Slots are named according to the following general rule

```
On[variable name who send the signal][signal]();

E.g. connect( loadImagePushButton, SIGNAL( clicked(bool ) ),
              this, SLOT( OnLoadImagePushButtonClicked( bool ) ) );
```

Signals are named according to the following general rule

```
Signal[MethodName]();

emit SignalMethodName();

E.g. SignalLayoutDesignChanged();
```

Example

.h

```
#include <QObject>

class Counter : public QObject {

Q_OBJECT

public:

  Counter() { m_value = 0; }

  int value() const
  {
    return m_value;
  }


public slots:

  void OnValueChanged(int v);


signals:

  void SignalValueChanged(int v);


private:

  int m_value; };
```

.cpp

```
void Counter::OnValueChanged(int v)
{
  if( v!= m_value )
  {
    m_value = v;
    emit SignalValueChanged(v);
  }
}
```

.cpp

```
Counter a, b;

QObject::connect(&a,SIGNAL(SignalValueChanged(
          int)),&b, SLOT(OnValueChanged(int)));

a.OnValueChanged(12);
//ouput:
//a.value()==12
//b.value()==12

b.OnValueChanged(48);
//output:
//a.value()==12
//b.value()==48
```

Building signal/slots

- C++ preprocessor changes or removes the signals, slots, and emit keywords so that the compiler is presented with standard C++

- By running the moc on class definitions that contain signals or slots, a C++ source file is produced which should be compiled and linked with the other object files for the application

References

- Qt homepage: http://doc.qt.nokia.com/4.7/signalsandslots.html