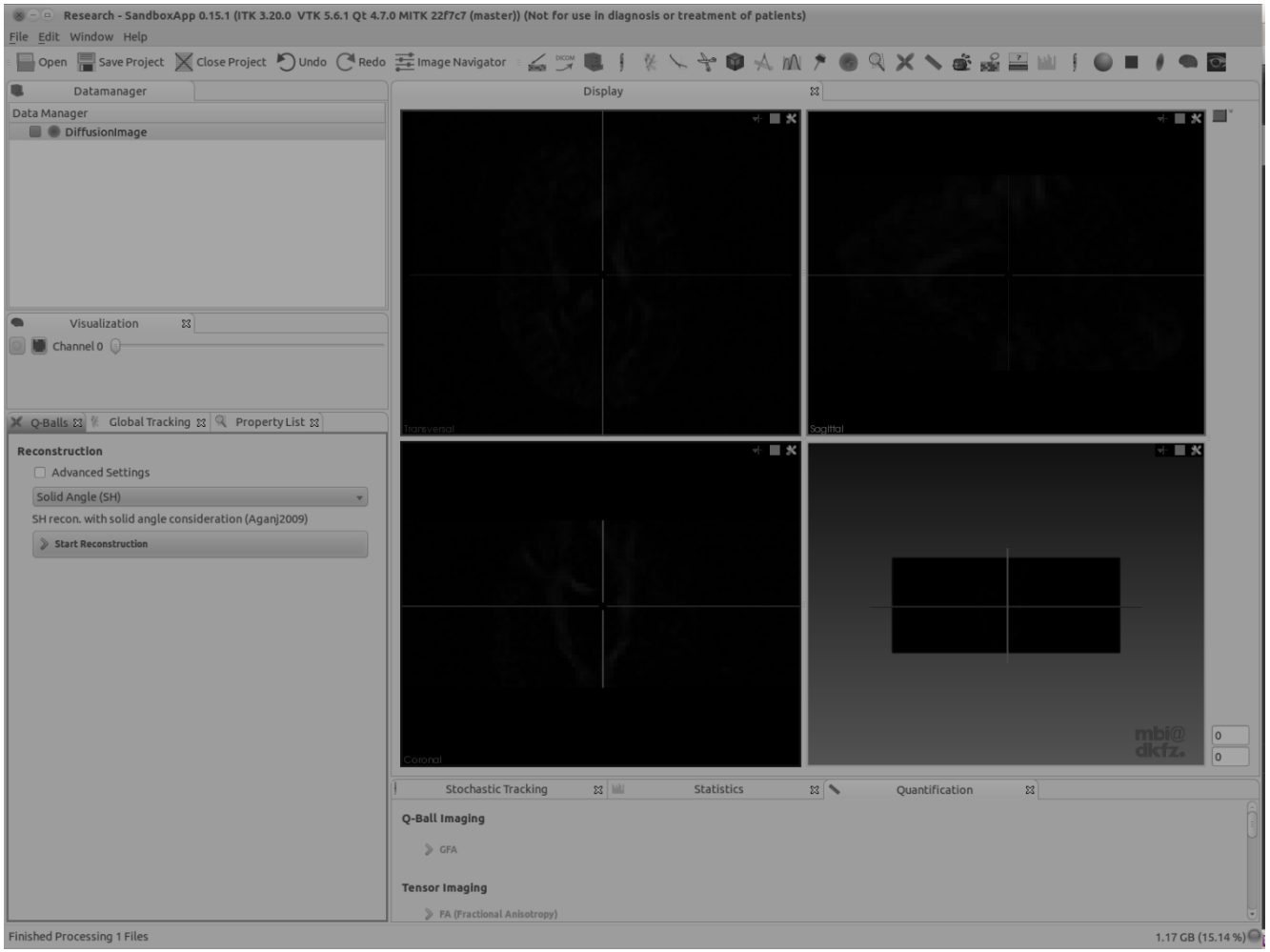
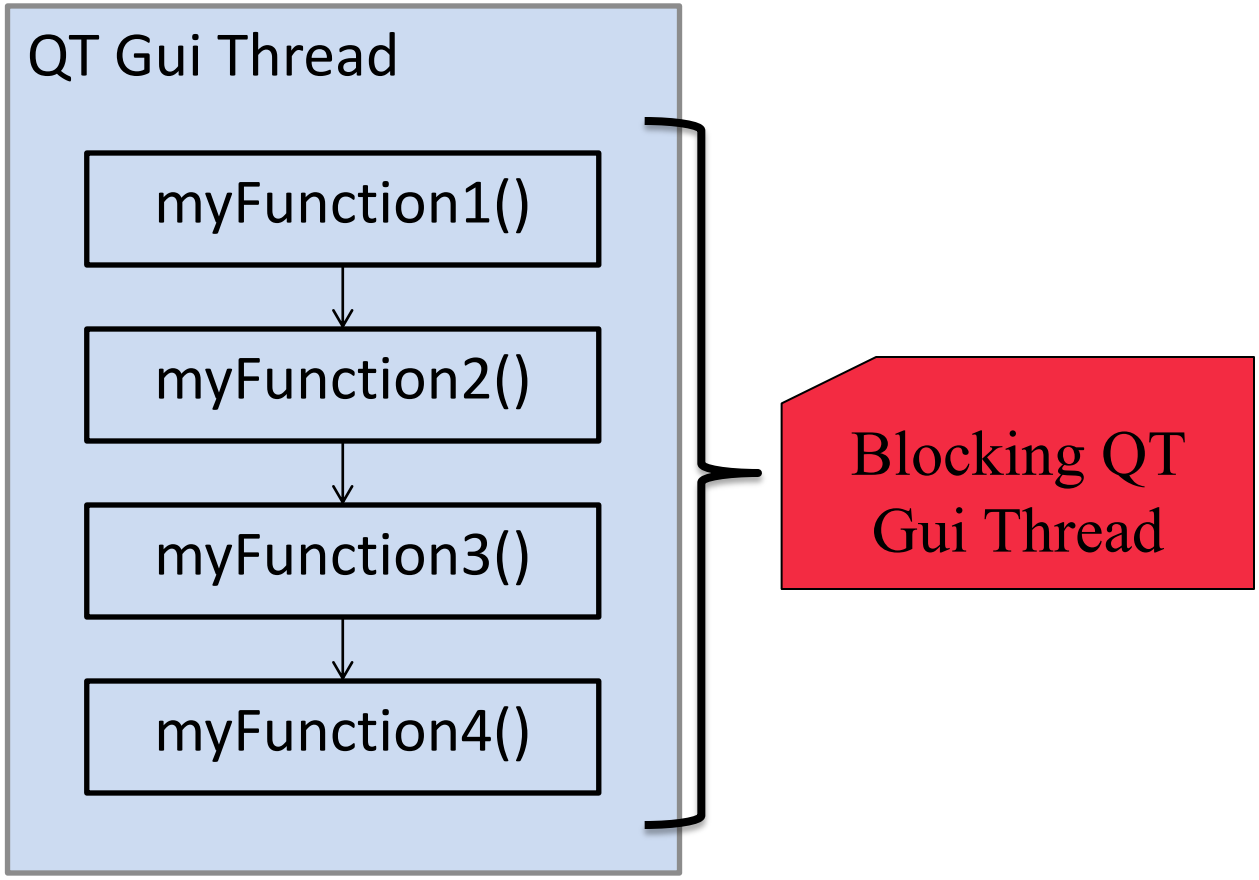
Using QThread in MITK

Peter Neher

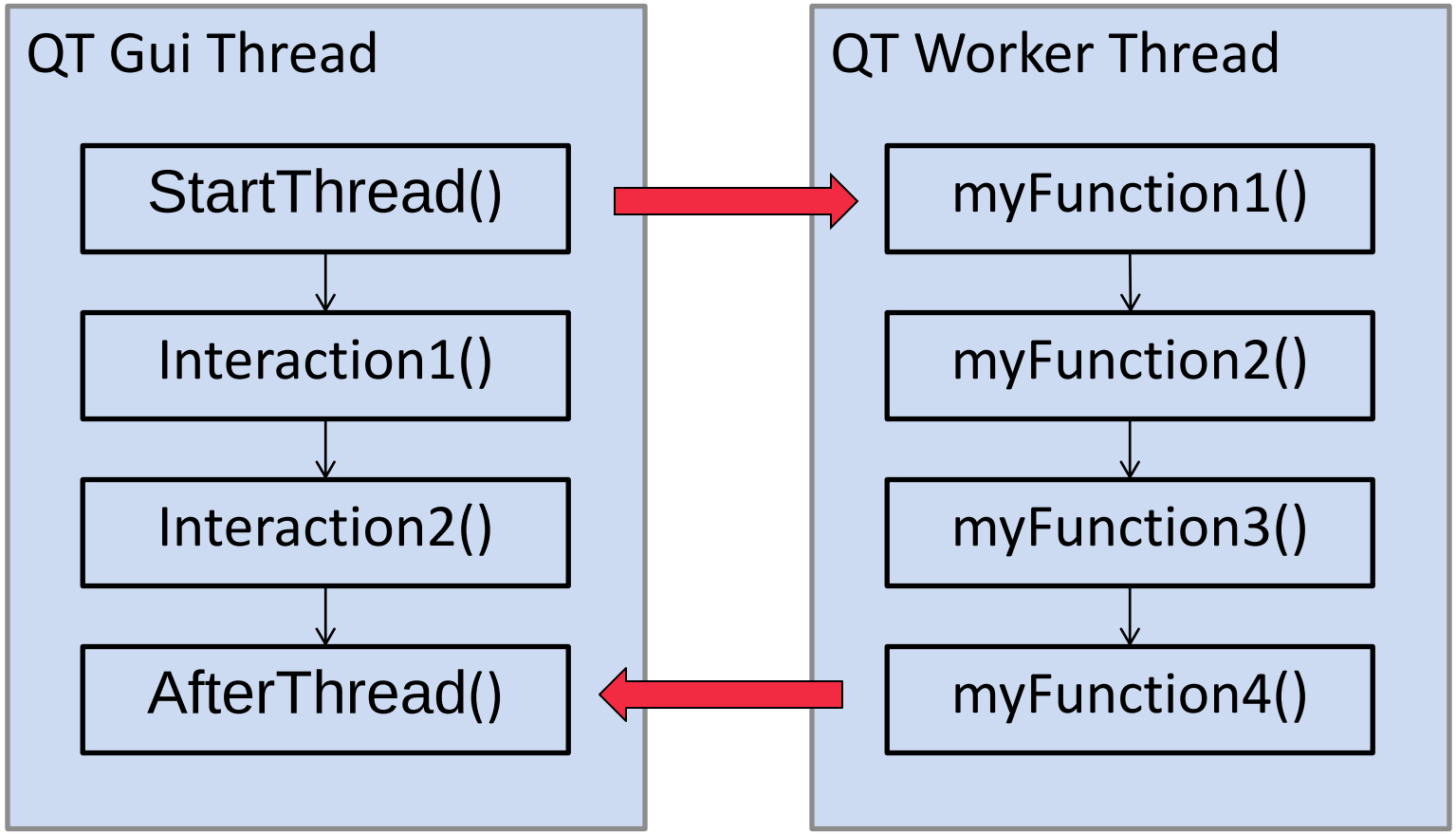
Motivation


Research - SandboxApp 0.15.1 (ITK 3.20.0 VTK 5.6.1 Qt 4.7.0 MITK 22f7c7 (master)) (Not for use in diagnosis or treatment of patients)
File Edit Window Help
Open Save Project Close Project Undo Redo Image Navigator
Datamanager
Data Manager
DiffusionImage
Visualization
Channel 0
Q-Balls Global Tracking Property List
Reconstruction
Advanced Settings
Solid Angle (SH)
SH recon. with solid angle consideration (Aganj2009)
Start Reconstruction
Display
Transversal
Sagittal
Coronal
mbi@dkfz.
Stochastic Tracking Statistics Quantification
Q-Ball Imaging
GFA
Tensor Imaging
FA (Fractional Anisotropy)
Finished Processing 1 Files
1.17 GB (15.14 %)

Motivation


QT Gui Thread
myFunction1()
myFunction2()
myFunction3()
myFunction4()
Blocking QT Gui Thread

Using Worker Threads



Example

```
#include <QThread>

class MyFilterThread : public QThread
{
  Q_OBJECT

  StartThread();

protected:
    void run();
};
```

Example

```
// process filter output
void MyFilterThread::AfterThread(){}

// setup and update filter
void  MyFilterThread::run(){}

void MyFilterThread::StartThread()
{
  QThread::start();
}
```

- **Filter object has to be created in the thread it's supposed to run in**
- **For another simple Method see: QtConcurrent**

Cross-Thread Events

- QT signal/slot principle supports multi threading
- ITK events are not thread safe!
- ITK event handling is always performed in the caller thread
- → Invoke QT events via ITK events

Cross-Thread Events

```
// ITK event handling (signal emitted and handled in the worker thread)
command = itk::MemberCommand< MyFilterThread >::New();
command->SetCallbackFunction(this, &MyFilterThread::MyCallbackFunction);
m_FilterObserverTag = m_MyFilter->AddObserver(itk::AnyEvent(), command);

// QT event handling (signal from worker thread to gui thread)
QObject::connect(this,SIGNAL(MySignal()),this,SLOT(MySlot()),Qt::QueuedConnection);
```

The END