Objektorientierte Programmierung mit C++ (WS 2010)

Dr. Andreas F. Borchert, Tobias Brosch
Institut für Angewandte Informationsverarbeitung
Universität Ulm

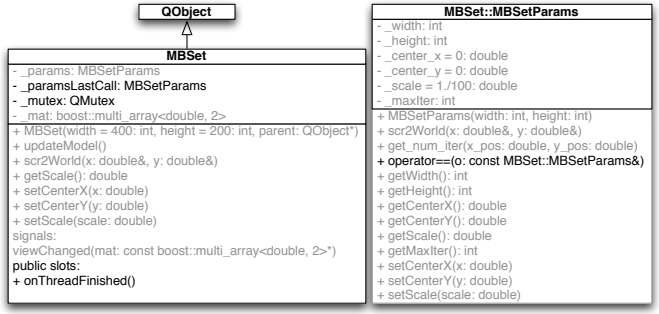
Blatt 13: Abgabetermin 02. Februar 2011 11 Uhr

Mandelbrotmenge Interaktiv (Teil 3)

Wie angekündigt, beschäftigt sich dieses Übungsblatt damit, die Rechenaufgabe in einen eigenen Thread auszulagern. Spätestens hier zahlt es sich aus, das MVC-Pattern verwendet zu haben. Denn wie Wir gleich sehen werden, muss lediglich das Modell angepasst werden.

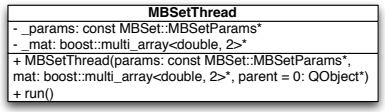
Model

Die Idee ist, in der Methode `MBSet::updateModel()` einen Thread zu starten, welcher in seiner `run()`-Methode unsere interne Matrix aktualisiert. In der Zwischenzeit soll unser Nutzer in der Lage sein, z.B. durch mehrfaches Mausklicken, die Parameter zu ändern. Ist dann irgendwann der Thread mit seiner Arbeit fertig, soll überprüft werden, ob sich die Parameter in der Zwischenzeit verändert haben. Ist dies der Fall, so soll erneut die Methode `updateModel()` ausgeführt werden.



Diese Aufgabe lösen wir, indem wir von `QThread` ableiten, die `run()`-Methode passend implementieren, und das Signal `finished()` von `QThread` mit einem von uns bereitgestellten Slot in `MBSet` verbinden, welcher überprüft, ob sich die jetztigen Parameter zu denen vom Zeitpunkt des Aufrufs des Threads unterscheiden.

Dazu ergänzen wir zunächst `MBSet` wie in den UML-Diagrammen gezeigt, und implementieren unseren Thread in der Klasse `MBSetThread`:



Mittels eines `Mutex` (von Englisch „mutual exclude“) verhindern wir, dass mehrere Threads gleichzeitig gestartet werden:

```
void MBSet::updateModel() {
  if (_mutex.tryLock()) { // check if another thread works already
    // save params, resize matrix

    // create new QThread

    // to delete thread on delete, connect SIGNAL(finished())
    // of thread to SLOT(deleteLater()) of thread

    // connect SIGNAL(finished()) of thread to our
    // SLOT(onThreadFinished()) to check
    // if a further update is necessary

    // start thread

  } // else not needed - then thread finishes, it checks
    // whether a new calculation is necessary
}

void MBSet::onThreadFinished() {
  _mutex.unlock();
  if (!(_paramsLastCall == _params)) {
    updateModel();
  }
  emit viewChanged(&_mat);
}
```

Submission

Einreichen der Lösung mit:

```
submit cpp 13 main.cpp MBSet.h MBSet.cpp \
       MBSetThread.h MBSetThread.cpp
```

Viel Spaß!