Copy-on-write und Implicit Sharing in Qt

Roland Hieber

Stratum 0 e. V.

14. Dezember 2012

Motivation

```
class Node {
        double lat_;
        double lon_;
public:
        Node() : lat_(0), lon_(0) {}
        Node(double lat, double lon) : lat_(lat), lon_(lon) {}
        void setLat(double lat) { lat_ = lat; }
        void setLon(double lon) { lon_ = lon; }
        double lat() { return lat_; }
        double lon() { return lon_; }
};

Node doSomething(Node n) { /* ... */ return n; }

void main(int argc, char ** argv) {
        Node n = doSomething(Node(52.2785658, 10.211247));
}
```

Motivation

```
class Node {
        double lat_;
        double lon_;
public:
        Node() : lat_(0), lon_(0) {}
        Node(double lat, double lon) : lat_(lat), lon_(lon) {}
        void setLat(double lat) { lat_ = lat; }
        void setLon(double lon) { lon_ = lon; }
        double lat() { return lat_; }
        double lon() { return lon_; }
};

Node doSomething(Node n) { /* ... */ return n; }

void main(int argc, char ** argv) {
        Node n = doSomething(Node(52.2785658, 10.211247));
}
```

- drei Kopien eines `Node`-Objekt
- braucht jedesmal Speicherplatz
  - obwohl der Inhalt identisch ist
- Lösungsmöglichkeiten: (Shared) Pointer, Copy-on-write

Copy-on-write

Methodik:

- Kopien teilen sich vorerst einen Speicherbereich *(shallow copy)*
- Kopien werden erst wirklich kopiert, wenn sie geändert werden *(deep copy)*

Copy-on-write

Objekt kopieren:

- vorerst auf Speicherbereich des alten Objekts verweisen
- gemeinsamen Referenzzähler inkrementieren

Copy-on-write

Objekt ändern:

- Falls Referenzzähler gleich 1:
  - *nur ein Objekt referenziert gemeinsamen Speicher*
  - Änderungen wie gewohnt vornehmen
- Falls Referenzzähler größer als 1:
  - *mehr als ein Objekt referenziert gemeinsamen Speicher*
  - alten Referenzzähler dekrementieren
  - neuen Speicher reservieren
  - alten Speicherbereich kopieren
  - neuen Referenzzähler auf 1 setzen
  - Änderungen wie gewohnt in neuem Speicherbereich vornehmen

Copy-on-write

Objekt löschen:

- gemeinsamen Referenzzähler dekrementieren
- Falls Referenzzähler 0:
  - *kein Objekt referenziert mehr den gemeinsamen Speicher*
  - Speicherbereich freigeben

Umsetzung in Qt

Implementierung als Flyweight Pattern:

- `NodeData` speichert nur Daten
  - abgeleitet von `QSharedData`
  - threadsichere Referenzzählung

Umsetzung in Qt

Implementierung als Flyweight Pattern:

- `NodeData` speichert nur Daten
  - abgeleitet von `QSharedData`
  - threadsichere Referenzzählung
- `Node` als Wrapper für Zugriff von außen
  - enthält Member vom Typ `QSharedDataPointer<NodeData>`
    - Shared Pointer auf QSharedData-Objekte
    - `detach()` erstellt deep copies
    - Überladung von `operator->()` ruft bei Schreibzugriff `detach()` auf
  - Copy-Konstruktor kopiert nur Referenz auf `QSharedDataPointer<NodeData>`!

Beispiel

nodedata.h

```
#include <QSharedData>
class NodeData : QSharedData {
public:
        double lat_;
        double lon_;
        NodeData() : lat_(0), lon_(0) {}
        NodeData(const NodeData& o) : QSharedData(o),
                lat_(o.lat_), lon_(o.lon_) {}
        virtual ~NodeData() {}
};
```

Beispiel

node.cpp

```
#include "nodedata.h"
#include <QSharedDataPointer>
class Node {
        QSharedDataPointer<NodeData> d;
public:
        Node() {}
        Node(double lat, double lon) {
                d = new NodeData;
                d->lat_ = lat;
                d->lon_ = lon;
        }
        Node(const Node& o) : d(o.d) {}

        void setLat(double lat) { d->lat_ = lat; }
        void setLon(double lon) { d->lon_ = lon; }
        double lat() { return d->lat_; }
        double lon() { return d->lon_; }
};
```

Vielen Dank für die Aufmerksamkeit!


Qt Documentation: Implicit Sharing.
https://qt-project.org/doc/qt-4.8/implicit-sharing.html


Qt Documentation: QSharedData.
https://qt-project.org/doc/qt-4.8/qshareddata.html


Qt Documentation: QSharedDataPointer. https://qt-project.org/doc/qt-4.8/qshareddatapointer.html


Wikipedia: *Flyweight Pattern*.
https://en.wikipedia.org/w/index.php?oldid=526237500