Übungsblatt 6

(4. Juni 2007)

(3 Punkte)

Dieses Blatt ist der direkte Vorgänger für die abschließende kleine Projektarbeit, in der die in diesem Blatt entstandenen Applikation zu einem kleinen Spiel im zweidimensionalen Raum ausgebaut werden soll.

Aufgabe 1 Lassen Sie jetzt zusätzlich die Klasse `GeometricObject` folgende Schnittstelle implementieren:

MoveableObject.hpp

```
1 #ifndef MOVEABLE_OBJECT_HPP
2 #define MOVEABLE_OBJECT_HPP
3 class MoveableObject{
4 public:
5     virtual void move()=0;
6 };
7 #endif
```

Die Methode `move` soll für eine geometrische Figur die Position der Figur verändern, wenn durch einen äußeren Tick das Objekt aufgefordert wird, sich zu bewegen. Sehen sie hierzu Felder `double dX` und `double dY` in der Klasse `GeometricObject` vor, die angeben, um wieviel sich ein Objekt in `x`- bzw in `y`-Richtung bei einem Aufruf von `move` bewegen soll.

Aufgabe 2 Die folgende neue Version der Klasse `Board` ist jetzt in der Lage, eine Liste von geometrischen Figuren animiert darzustellen:

Board.hpp

```
1 #ifndef BOARD_H_
2 #define BOARD_H_
3
4 #include <QWidget>
5 #include <QTimer>
6 #include <vector>
7 #include "GeometricObject.hpp"
8
```

```
9  class Board:public QWidget {
10 Q_OBJECT
11  public:
12   QTimer timer;
13   Board(std::vector<GeometricObject*>& geos);
14   std::vector<GeometricObject*>& geos;
15   QSize minimumSizeHint();
16   QSize sizeHint() ;
17   void paintEvent(QPaintEvent *event);
18
19   static int showPaintable
20    (std::vector<GeometricObject*>& geos,int argc,char **argv);
21
22 public slots:
23   virtual void tick();
24
25 };
26 #endif /*BOARD_H_*/
```

Board.cpp

```
1  #include "Board.hpp"
2  #include <QtGui>
3
4  Board::Board(std::vector<GeometricObject*>& geos)
5                                          :QWidget(0),geos(geos){
6    setBackgroundRole(QPalette::Base);
7    connect(&timer,SIGNAL(timeout()),this,SLOT(tick()));
8    timer.start(25);
9  }
10
11 QSize Board::minimumSizeHint(){return QSize(100, 100);}
12 QSize Board::sizeHint(){return QSize(400, 300);}
13
14 template <typename IT>
15 void paintIt(IT begin,IT end,QPainter& painter){
16     for (;begin!=end;begin++) (*begin)->paintMe(&painter);
17 }
18 template <typename IT>
19 void moveIt(IT begin,IT end){
20   for (;begin!=end;begin++) (*begin)->move();
21 }
22
```

```
23 void Board::paintEvent(QPaintEvent *){
24   QPen pen;
25   QPainter painter(this);
26   painter.setPen(pen);
27
28   paintIt(geos.begin(),geos.end(),painter) ;
29 }
30
31 int Board::showPaintable
32     (std::vector<GeometricObject*>& geos,int argc,char **argv){
33   QApplication application(argc,argv);
34   Board* b = new Board(geos);
35   b->show();
36   return application.exec();
37 }
38
39 void Board::tick(){
40   moveIt(geos.begin(),geos.end());
41   this->repaint();
42 }
```

a) Testen Sie Sie Klasse `Board` mit ein Instanzen Ihrer geometrischen Figuren.

b) Sorgen Sie jetzt in Ihrer Applikation dafür, dass sich die Objekte am Fensterrand abstoßen.

c) Sorgen Sie jetzt dafür, dass Objekte, die aneinanderstoßen, sich wieder voneinander abstoßen. Benutzen Sie hierzu die Methode `touches`.