Einführung in C++ – Übung 10
Testatgruppe A (Isaak)

Rasmus Diederichsen

17. Dezember 2014

Aufgabe 10.1 Generische Liste

src/util/List.hpp

```
#ifndef LIST_H

#define LIST_H

/**
 * @brief   A simple generic list class
 */
template<typename T> class List
{
   public:
      /**
       * @brief Struct to represent a node in the list.
       */
      struct Node {
         Node* next;
         T data;
      };
      /**
       * @brief Constructs an empty list.
       */
      List<T>();
      /**
       * @brief  Destructor. Frees the generated nodes.
       */
      ~List<T>();
      /**
       * @brief Inserts an item into the list, i.e. a new node
       *        containing @ref item is created.
       * @param item To be inserted
       *
       */
      void insert(T item);
      /**
       * @brief Iterates over all items in the list and calls
       * the given function @ref do_something(...) for
```

```
36          * every item stored in the list.
37          *
38          * @param do_something Function pointer to apply to all
                elements.
39          */
40         void for_each(void (*do_something)(T item));
41     private:
42         // Root of the list
43         Node* m_list;
44 };
45
46
47 #include "List.tcc"
48 #endif
49 /* end of include guard: LIST_H */
```

src/util/List.tcc

```
1  #include <iostream>
2  template<typename T> List<T>::List()
3  {
4      m_list = NULL;
5  }
6  template<typename T> List<T>::~List()
7  {
8      if (m_list->next != NULL) delete m_list->next;
9      delete m_list;
10 }
11 template<typename T> void List<T>::insert(T item)
12 {
13     if (m_list == NULL)
14     {
15         m_list = new Node;
16         m_list->data = item;
17         m_list->next = NULL;
18     } else
19     {
20         Node* m = new Node;
21         m->data = item;
22         m->next = m_list;
23         m_list = m;
24     }
25 }
26 template<typename T> void List<T>::for_each(void (*do_something)(T
       item))
27 {
28     Node* tmp = m_list;
29     while (tmp != NULL)
30     {
31         do_something(tmp->data);
32         tmp = tmp->next;
33     }
34 }
```

Aufgabe 10.2 Implementierung eines Asteroidenfelds

src/rendering/AsteroidField.hpp

```
/**
 *  @file Starfield.hpp
 *
 *  @date 27.11.2011
 *  @author Tim Kühnen
 *  @author Dominik Feldschnieders
 *  @author Henning Strüber
 *  @author Thomas Wiemann
 */

#ifndef STARFIELD_HPP_
#define STARFIELD_HPP_

#include <cmath>
#include <algorithm>
#include <vector>

#include "Asteroid.hpp"
#include "FixedObject.hpp"
#include "math/Vertex.hpp"
#include "util/List.hpp"

using std::for_each;
using std::vector;
using std::generate;

namespace asteroids
{


/**
 * @brief   Representatio of an asteroid field
 */
class AsteroidField : public FixedObject
{
public:


        /**
         * @brief   Creates an asteroid field with n asteroids in
             it
         */
        AsteroidField(int n, string basePath = "");

        /**
         * @brief   Dtor.
         */
        virtual ~AsteroidField();

        /**
         * @brief   Renders all asteroids
         */
        void render();

private:

```

```
56      /// The asteroids in the field
57      List<Asteroid*> asteroids;
58
59      /// Relative path to look for the asteroid model
60      string m_basePath;
61  };
62
63  }
64
65  #endif
```

src/rendering/AsteroidField.cpp

```
1   #include "AsteroidField.hpp"
2   #include "math/Randomizer.hpp"
3   #include "rendering/Asteroid.hpp"
4
5   namespace asteroids
6   {
7
8
9   AsteroidField::AsteroidField(int quantity, string basePath) :
10      m_basePath(basePath)
11  {
12          // Generate asteroids
13          for(int i = 0; i < quantity; i++)
14          {
15              TriangleMesh* mesh = TriangleMeshFactory::instance().
                    getMesh(m_basePath + "asteroid.3ds");
16         asteroids.insert(new Asteroid(mesh,Randomizer::instance()->
               getRandomVertex(2500.),Randomizer::instance()->
               getRandomNumber(1.,5.)));
17          }
18
19  }
20
21  void deleteAsteroid(Asteroid* a)
22  {
23      delete a;
24  }
25
26  AsteroidField::~AsteroidField()
27  {
28      asteroids.for_each(deleteAsteroid);
29  }
30
31
32  void renderAstroid(Asteroid* a)
33  {
34      a->render();
35  }
36  void AsteroidField::render()
37  {
38      asteroids.for_each(renderAstroid);
39  }
40
41
42  }
```