UNIVERSITÄT OSNABRÜCK

Dr. Thomas Wiemann

8. Übungsblatt zur Vorlesung „Einführung in die Programmiersprache C++“

Wintersemester 2014 / 2015

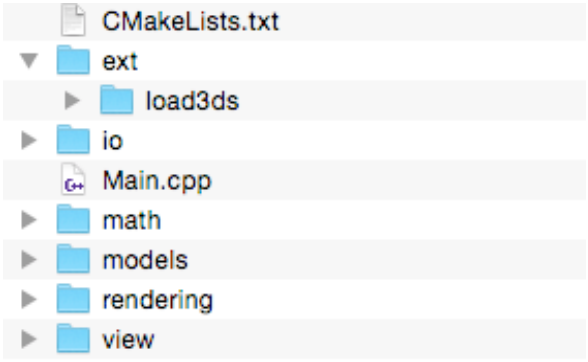
Aufgabe: Integration von Bitmap-Texturen und Refaktoring

Ziel der Übung:

In dieser Übung werden wir die Struktur des Viewers überarbeiten und an geeigneten Stellen Funktionalitäten in Oberklassen auslagern. Auf diese Art und Weise werden wir uns mit dem Klassenkonzept und der C++-Syntax zur Vererbung vertraut machen. Weiter werden Sie sich mit dem Factory-Designpattern auseinander setzen, indem Sie Fabrikklassen zum Laden von 3D-Modellen und Bildern in unterschiedlichen Dateiformaten implementieren. In StudIP finden Sie eine neue Version der Software mit einer erweiterten Verzeichnisstruktur. Ihre Aufgabe besteht nun zunächst darin, die Software so umzubauen, dass Sie leichter wartbar und erweiterbar wird.

Neue Struktur der Software

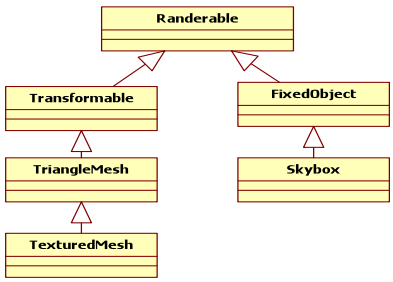
In dieser Aufgabe werden wir die Software um diverse Features erweitern. Zur besseren Übersichtlichkeit haben wir die vorhandenen Quelldateien in verschiedene Unterordner aufgeteilt:



Im Ordner "ext" liegt der Quellcode einer externen Bibliothek zum laden von .3ds-Dateien. Da diese Bibliothek standardmäßig nicht mit den gängigen Linux-Distributionen verteilt wird, haben wir die Quellen hier in unsere Software übernommen und in das `cmake`-Buildsystem integriert. Im "io"-Ordner liegen alle Dateien, die zum Laden und Speichern von 3D-Modellen benötigt werden. Der "math"-Order beinhaltet die Funktionalitäten, die mit mathematischen Operationen zu tun haben, insbesondere die Vertex- / Matrix- / Quaternion-Implementierungen. Unter "models" liegen alle Dateien, die zur Modellierung der Spielwelt benötigt werdden. Dazu gehören verschiedene Flugzeugmodelle mit den entsprechenden Ressourcen sowie ein Satz Bitmaps zur Darstellung der Weltraumumgebung. Der Ordner "rendering" enthält die Quelltexte zu allen Objekten, die in unserer Software gerendert werden können. In "view" liegen die Klassen, die für die Darstellung verantwortlich sind.

Aufgabe 1: Auslagern von Funktionalität in Oberklassen:

Schauen Sie sich die Klassen im Unterverzeichnis "rendering" der Software an. Dort finden Sie neben der bekannten `TraingleMesh`-Klasse nun die Klassen `Texture` und `Skybox` sowie eine Klasse `TexturedMesh` zur Repräsentation von texturierten 3D-Modellen. Die `Skybox`-Klasse nutzt Texturen, um einen Hintergrund um die 3D-Szene zu malen. Überarbeiten Sie die Klassen so, dass sich folgende Klassenstruktur ergibt (10 Punkte):



Lagern Sie Funktionalitäten der Unterklassen geeignet in die Oberklassen aus. Gibt es Kandidaten für pure-virtual-Methoden und bei welchen Klassen macht es Sinn, sie in abstrakte Klassen zu verwandeln (10 Punkte)?

Aufgabe 2: Erweiterungen für neue Dateiformate

Im Ordner "`io`" finden Sie die Klassen `ReadPLY` und `ReadOBJ` zum Laden von Dreiecksnetzen. Um die Stellen im Programm, an denen neue Objekte erzeugt werden, zu vereinfachen implementieren Sie eine Klasse `TriangleMeshFactory`. Diese Klasse soll eine Methode

```
TriangleMesh* TriangleMeshFactory::getMesh (const string &filename)
```

enthalten, die eine neue Instanz eines Meshes liefert. Parsen Sie die Dateiendung der übergebenen Datei und benutzen Sie die zur Verfügung gestellten Reader, um ein neues Mesh zu erzeugen. Nutzen Sie das `MeshIO`-Interface, um den Code so kompakt wie möglich zu halten (5 Punkte). Implementieren Sie zudem analog eine Fabrikklasse für Texturen (`TextureFactory`) (10 Punkte):

```
Texture* TextureFactory::getTexture (const string &filename)
```

Texturen sind 2D-Bilder, die wie eine Tapete über die 3D-Modelle gelegt werden, um eine realistische Oberfläche zu simulieren. Zum Laden von Bilddaten in unterschiedlichen Formaten werden Ihnen u.a. die Klassen `ReadTGA` und `ReadTIFF` zur Verfügung gestellt. Implementieren Sie zusätzlich die Klassen `ReadPPM`, die Bilder im .ppm-Format (binär und ASCII) lesen kann (25 Punkte). Weiterhin soll mittels der `libjpeg`-Blibliothek (20 Punkte) ein jpg-Reader (`ReadJPG`) implementiert werden. Die Fabrik-Klasse soll als Singleton realisiert sein. Die 3ds-Modelle im "rendering"-Ordner haben die Referenzen zu Ihren Texturen ohne absoluten Pfad. Das ist problematisch, wenn das viewer-Programm aus einem anderen Ordner als dem "models"-Ordner aufgerufen wird. Daher muss bei `getTexture(...)` der komplette Pfad zum Modell, das geladen werden soll, an den Parameter `filename` angehängt werden. Fügen Sie der `TextureFaktory`-Klasse eine Methode `setBasePath(...)` hinzu, die in der `TriangleMeshFactory` gesetzt wird, und den Verzeichnispfad zum zu ladenden Modell korrekt setzt. D.h., bei allen `getTexture(...)`-Aufrufen wird dieser Pfad an den Paramter `filename` angehängt (10 Punkte). Die Reader-Klassen für die verschiedenen Dateiformate sollen Konkretisierungen der abstrakten Klasse `BitmapReader` sein, deren Signatur Sie im "io"-Ordner finden. Erklären Sie Ihren Tutor die Elemente dieser Klasse (10 Punkte).

Abgabe

Ihre neu erstellten und die modifizierten Klassen sind bis Montag den 8. Dezember 08:00 Uhr mit der Angabe Ihrer Gruppe an die Adresse `cpp@informatik.uni-osnabrueck` zu mailen. Bringen Sie zusätzlich einen Ausdruck der `ReadPPM`, `ReadJPG` und Fabrik-Implementierungen zum Testat mit.