UNIVERSITÄT OSNABRÜCK

Dr. Thomas Wiemann

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

Wintersemester 2018 / 2019

Laden von Dreiecksnetzen aus PLY-Dateien und Einführung in cmake

Ziel der Übung:

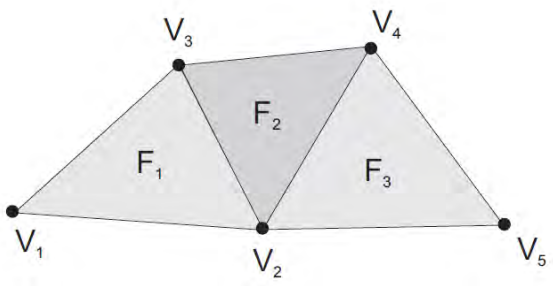
In dieser Aufgabe werden wir die dynamische Allokierung und Verwaltung von Ressourcen üben. Dazu erweitern wir unser Projekt um die Funktionalität, dreidimensionale Objekte aus .ply-Dateien im Binär-Format zu laden. Darüber hinaus machen wir uns mit `cmake`, einem Framework zum Übersetzen komplexer C/C++-Projekte auseinander.

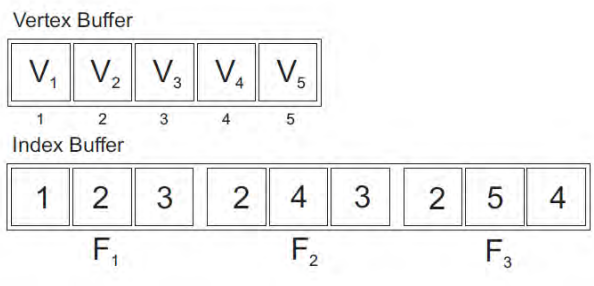
Aufgabe 1: Einarbeitung in cmake (30 Punkte)

Machen Sie sich mit cmake vertraut. Eine Umfangreiche Dokumentation finden Sie unter `http://www.cmake.org`. Ein kleines Tutorial zum Einstieg finden sie unter `http://web.cs.swarthmore.edu/~adanner/tips/cmake.php`. Was sind die Vorteile von `cmake`? Erklären Sie Ihrem Tutor, welche wichtigen Funktionen von `cmake` zur Verfügung gestellt werden. Wie sind `cmake`-Projekte typischerweise aufgebaut? Erklären Sie den Unterschied zwischen In-Source- und Out-of-Source-Builds. Übersetzen Sie das in Ihrem Repository bereitgestellte Programmgerüst mit `cmake`.

Aufgabe 2: Implementierung eines PLY-Importers (50 Punkte)

In Ihrem Repository finden Sie den Rumpf der Programmieraufgabe für dieses Blatt sowie ein 3D-Modell eines Raumschiffes, das im binären Stanford PLY-Format abgelegt wurde. Solche 3D Modelle bestehen aus Vertices, die Punkte im Raum definieren, und Dreiecken, die die Oberfläche des Modells darstellen. Das Modell liegt in Form eines indizierten Vertexbuffers vor. Ein Vertexbuffer ist ein eindimensionales `float`-Array. Jeweils drei Einträge repräsentieren einen Knoten im Dreiecksnetz (x, y, und z-Koordinaten). Die Dreiecksflächen werden in einem zweiten Buffer definiert. Dieser so genannte Indexbuffer ist ein `int`-Array, in dem die Vertices aus dem Vertexbuffer referenziert werden. Jeweils drei Einträge definieren ein Dreieck. Auf diese Art und Weise müssen gemeinsame Knoten der Dreiecke im Netz nicht mehrfach abgespeichert werden:





Eine gute und ausführliche Beschreibung des PLY-Formates finden Sie unter http://paulbourke.net/dataformats/ply/. In der vorliegenden Implementierung bindet `main.c` den Header `plyio.h` ein, um die Funktion

```
void loadPLY(
      char* file,
      Model* model
);
```

nutzen zu können. Der Parameter `file` ist der Name der zu ladenden PLY-Datei, die Struktur `Model` ist wie folgt definiert:

```
typedef struct {
      int numVertices;
      int numFaces;
      int* indexBuffer;
      float* vertexBuffer;
} Model;
```

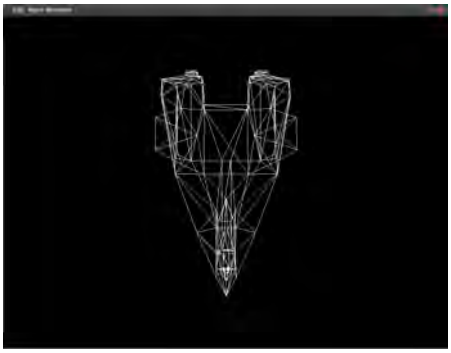
Nach dem erfolgreichen Aufruf der Funktion `loadPLY` soll `model` die entsprechenden Repräsentationen des Index- und Vertexbuffer des geladenen Modells enthalten. Implementieren sie diese Funktion in der Datei `plyio.c`. Die Buffer sollen dynamisch in `loadPLY` allokiert werden. Konnte `file` nicht geöffnet werden oder keine gültige `.ply`-Datei sein, sind alle Parameter auf `0` zu setzen. Sie dürfen für Ihre Implementierung davon ausgehen, dass sie nur binäre PLY-Files öffnen sollen, die Vertices keine weiteren Attribute enthalten und die Faces nur aus Dreiecken bestehen. Parsen Sie den Header des PLY-Files, um die benötigten Längen der Buffer zu erhalten. Das erfolgreiche Laden des Models können sie mit den bereitgestellten Funktionen `printModel` und `printBuffers` testen. Erweitern Sie das Programm um Code, der die vom Modell belegten Resourcen wieder freigibt.

Aufgabe 3: Implementierung des Rendering Codes (20 Punkte)

Implementieren Sie in der Funktion `mainLoop(...)` in der Datei `mainwindow.c` den Code zum Rendern des Models. Das vorgegebene Programmgerüst enthält alle Funktionen zum Erstellen eines Fensters sowie den benötigen Initialisierungscode für OpenGL. Das Model soll im Wireframe-Modus gerendert werden. Dazu müssen Sie für jedes Dreieck einen sogenannten Line-Loop erstellen. Der Code dazu sieht wie folgt aus:

```
glBegin(GL_LINE_LOOP);
glVertex3f(...);
glVertex3f(...);
glVertex3f(...);
glEnd();
```

Als Parameter erhalten die Aufrufe von `glVertex3f(...)` die 3D-Koordinaten eines Vertex des jeweiligen Dreiecks. Nach korrekter Implementierung sollte das Model wie folgt ausgegeben werden:



Abgabe:

Checken Sie Ihre Lösung des Aufgabenblattes bis Montag den 12.11.2018, 8:00 Uhr in den Master-Branch des git-Repositorys Ihrer Gruppe ein