Graphische Datenverarbeitung I

Praxisübung 2

Prof. Dr. Dieter Fellner
Daniel Ströter, M.Sc.
David Spitzenberg, M.Sc.

TECHNISCHE UNIVERSITÄT DARMSTADT

Wintersemester 2021/2022
Ausgabe: 23. November 2021, Abgabe: 7. Dezember 2021, 09:30 Uhr

Abgabemodalitäten

Das *lauffähige* Programm, mit allen Unteraufgaben, wird im Rahmen eines Zoom-Meetings testiert. Zusätzlich muss der Code vor der Deadline auf Moodle abgegeben werden. Es wird eine Abgabe pro Gruppe eingereicht. Die hochgeladene Datei muss folgende Anforderungen erfüllen:

- Es muss eine **zip**-Datei sein.
- Der Name der Datei muss wie folgt lauten: Gruppe*Gruppennummer*_Praxis*Praxisübungsnummer*.zip. Zum Beispiel heißt die Abgabe dieser Hausaufgabe der Gruppe 3 wie folgt: Gruppe3_Praxis2.zip

Während des Testats sind Verständnisfragen und ein Blick über den Code möglich. Alle Fragen sollten von allen Gruppenmitgliedern beantwortet werden können. Wenn eine Aufgabe nicht vollständig funktioniert gibt es Teilpunkte, falls der Versuch in die richtige Richtung geht.

Sie können das Programm gerne selbständig weiter ausbauen. Schöne Ergebnisse werden mit Bonuspunkten belohnt.

Hinweis: Bitte das Programm auf release (optimiert) erstellen. Das kann bei OpenGL einen riesigen Performance-Sprung ausmachen.

In dieser Übung sollen 3D-Modelle durch array pointer und VBOs schneller gezeichnet werden. Als zweites soll ein eigener Shader programmiert werden. Abschließend soll eine etwas umfangreichere und dynamische Szene erstellt werden.

Es liegt wieder ein Framework bereit. Dieses liest bereits eine OFF Datei, sowie einen einfachen Shader ein. Sie können auch ihr eigenes Framework weiter verwenden und die neuen Funktionen einpflegen.

Aufgabe 1: Nutzung von Vertex Arrays (2 Punkte)

Anstelle des langsamen immediate modes werden die Modelle nun mittels *vertex arrays* (siehe beispielsweise [1]) visualisiert. Die Funktionen

```
glEnableClientState(GL_VERTEX_ARRAY);
glEnableClientState(GL_NORMAL_ARRAY);
glDisableClientState(...);
```

aktivieren und deaktivieren *vertex arrays* und *normal arrays*.

Um ein Modell zu rendern müssen nach der Aktivierung die Pointer für die Eckpunkte und Normalen übergeben werden. Nutzen Sie dazu die Funktionen **glVertexPointer** und **glNormalPointer**.

Anschließend zeichen wir das Modell mit **glDrawElements**(GL_TRIANGLES, ...);

Aufgabe 2: Nutzung von Vertex Buffer Objects (2 Punkte)

Bei jedem Aufruf von **glDrawElements** werden die Positionen, Normalen und Indizes erneut aus dem Hauptspeicher auf die Grafikkarte geladen. Bei großen (oder vielen) Dreiecksnetzen kann es dadurch zu einer deutlich langsameren Darstellung kommen. Deshalb sollen Sie zur Beschleunigung *vertex buffer objects* (VBOs) verwenden, die es Ihnen erlauben, die Dreiecksnetzdaten direkt im Grafikkartenspeicher abzulegen.

Eine Erläuterung zu VBOs finden Sie beispielsweise unter [2].

Aufgabe 3: Performanceuntersuchungen (1 Punkt)

Untersuchen Sie die Performanzunterschiede zwischen dem immediate mode, Vertex Arrays und VBOs. Ein Timer zur Messung der Zeit kann hierfür nützlich sein. Die Szene sollte zudem genügend komplex aufgebaut sein, um einen Unterschied feststellen zu können.

Aufgabe 4: Shader (3 Punkte)

Im nächsten Schritt programmieren wir die Grafikpipeline mittels Vertex- und Fragment-Shadern. Im Framework ist das Laden der Shader (`FlatGreyShader.vert`, `FlatGreyShader.frag`) bereits implementiert. Sie können den Shader mit der Taste 3 aktivieren. Falls der Shader-Code Fehlerhaft ist, wird der Fehlerbericht auf der Konsole ausgegeben.

Im Lighthouse 3D Tutorial [3] finden Sie einige Beispiele für Shader Programme. Implementieren Sie zumindest einen der Toon Shader unter Shader Examples. Bei Erfolg können Sie gerne weitere der Beispiele implementieren.

Aufgabe 5: Kreative Aufgabe (2 Punkte)

Erstellen Sie abschließend eine komplexe und dynamische Szene. Sie können hier ihrer Kreativität freien Lauf lassen. Es sollten mehrere (verschachtelte) Transformationen, verschiedene Objekte und Farben beteiligt sein. Beispiele für solch eine Szene sind:

- Unser Sonnensystem. Hier dürfen und sollten die Objekte umskaliert werden, da sonst außer einer riesigen Sonne nichts zu sehen ist.
- Ein sich drehendes Kinderkarusell.
- Ein tanzender oder laufender Roboter.
- Der Star Wars Kampf um den Todesstern, oder was auch immer abgefahrenes Ihnen einfällt.

Beachten Sie, dass Sie zwischen dieser Szene und der Performanztest-Szene für das Testat wechseln können.

Literatur

[1] Vertex Array Tutorial `http://www.songho.ca/opengl/gl_vertexarray.html`

[2] Vertex Buffer Object Tutorial `http://www.songho.ca/opengl/gl_vbo.html`

[3] Lighthouse 3D's GLSL Tutorial. `http://www.lighthouse3d.com/tutorials/glsl-12-tutorial/`