

Dokumentation zur Projektaufgabe für das Modul Tool- und Pluginprogrammierung im Sommersemester 2022

Thema und Motivation

Als Thema für meine Modulaufgabe habe ich mich für **Entwicklung einer Mesh Decimation-Anwendung** entschieden, da ich in meinem nächsten privaten Projekt mit 3D-Meshes arbeiten will und ich dann potenziell Verwendung für einen Mesh Decimation Algorithmus finde.

Während meiner Recherche bin ich auf ein Paper mit dem Titel '*Surface Simplification Using Quadric Error Metrics*' von Michael Garland und Paul S. Heckbert gestoßen und habe mich dafür entschieden den in diesem Paper vorgestellten Algorithmus (zumindestens teilweise) zu implementieren.

Da ich am Anfang bisschen überfordert war und nicht wirklich wusste wo ich anfangen soll, habe ich nach anderen Implementationen gesucht, durch die ich den Algorithmus besser verstehen kann. Dabei habe ich ein GitHub Repository (<https://github.com/Meirshomron/MeshSimplification>) gefunden, das mir vor allem bei der Umsetzung der Fehlerberechnung geholfen hat.

Bibliotheken

In diesem Projekt verwende ich vier bzw. fünf Bibliotheken, die es mir ermöglichen eine grafischen Anwendung zu entwickeln, in der der Algorithmus an einem Modell ausgeführt werden kann.

Für das Erstellen des OpenGL-Kontextes und den Zugriff auf Maus und Tastatur verwende ich **GLFW** (<https://www.glfw.org/>). Ich habe mich dafür entschieden, da ich damit die meiste Erfahrung habe und gut damit klar komme.

Um mir den Umgang mit OpenGL zu vereinfachen, verwende ich das von mir selbst entwickelte Framework **Ignis** (<https://github.com/oliverjakobs/Ignis>). Grundlegender Bestandteil dieses Frameworks ist die OpenGL-Loading-Library **GLAD** (<https://glad.dav1d.de/>).

Da mein Programm interaktiv sein soll und ich das über eine grafische Benutzeroberfläche umsetzen will, verwende ich die Bibliothek **Dear ImGui** (<https://github.com/ocornut/imgui>).

Damit ich mir keine Gedanken machen muss, ob alle mathematischen Operationen auch wirklich richtig funktionieren (und mit OpenGL kompatibel sind), benutze ich die Mathe-Bibliothek **glm** (<https://github.com/g-truc/glm>).

Designentscheidungen

Meine Designentscheidungen unterteile ich in zwei Abschnitte. Im ersten Abschnitt geht es um allgemeine Entscheidungen, bei denen es um den Umfang des Projektes oder die Benutzerschnittstelle geht. Im zweiten Abschnitt geht es dann um Entscheidungen, die die eigentliche Implementation des Algorithmuses betreffen.

Allgemeine Entscheidungen

In dem im ersten Abschnitt erwähnten Paper werden die beiden Vertices für den Edge-Collapse

Zu Beginn habe ich mir Gedanken

Zuerst habe ich mir überlegt, was das Programm machen soll und was nicht.

Das Programm soll ein 3D-Model mit dem Format `.obj` laden und anzeigen können. Dann soll über ein GUI ausgewählt werden können auf wie viele Faces das Mesh reduziert werden soll. Durch einen Button soll dann der Algorithmus mit den vorher festgelegtem Ziel ausgeführt werden. Außerdem soll es einen Button geben, der das Mesh auf seinen ursprünglichen Zustand zurücksetzt.

Als kleines Extra habe ich noch zwei Render-Optionen hinzugefügt, die mit Check-boxen ein- und ausgeschaltet werden können.

Des Weiteren habe ich mich gegen den Export in ein Dateiformat entschieden, da es von der Komplexität eher trivial ist und ich im Normalfall in ein Engine-Spezifisches Format exportieren würde.

Flat-Shading - ermöglicht Vertices mit nur Positionen, außerdem hebt das die Faces hervor. Außerdem können beim Flat-Shading die Face-Normals im Fragment-Shader berechnet werden. Dadurch ist es möglich, dass die Vertices nur aus Positionen bestehen. Ein Vertex ist also einfach nur ein *vec3*. Das macht die Implementation des Algorithmuses übersichtlicher.

Entscheidungen zur Algorithmus Implementation

Ein Vertex hat eine Position und eine Fehler-Matrix und da in meiner Implementation ein .

Pairs - Stellt alle Paare an vertices da, die zu einem neuen Vertex zusammengefasst werden können.

Die Pairs werden in einem Heap gespeichert.

Vertices werden nicht entfernt - Es werden nur die Indices verändert bzw. entfernt. Das Entfernen der überflüssigen Vertices würde ich erst beim exportieren machen

Bedienung

Build