Projektreflexion

3D Objekt Rendering

Module Number: GPR5300

Module Name: "Graphics and Shader Programming for Games"

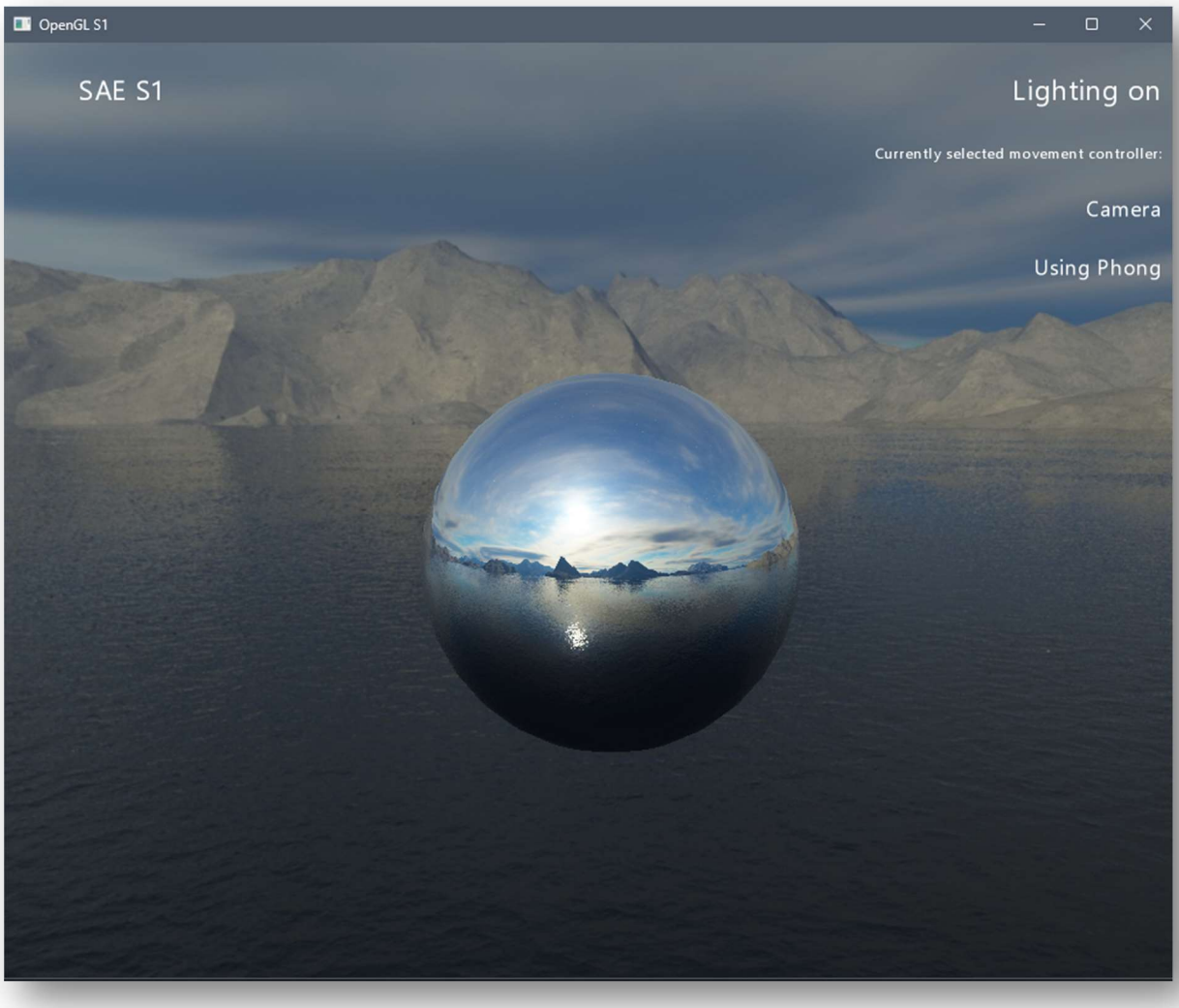
Date Submitted: 16.08.2023

Course: GPD 0921

Name: Dylan Frosini

City: Schaffhausen

Country: Switzerland



Inhaltsverzeichnis

1 Vorwort

Da die Abschlussarbeit eigentlich eine Bündelung von allem ist, was wir in dem Modul gelernt haben, werde ich in dieser Projektreflexion nicht nur auf die letzten 3 Wochen, in der ich effektiv an der Summativen gearbeitet habe, eingehen, sondern ich werde so gut es geht alle Dinge ansprechen, die meiner Meinung nach zur summativen Arbeit beigetragen haben und dabei reflektieren inwiefern, sie einen positiven oder negativen Einfluss auf das Ergebnis hatten.

2 Herangehensweise

2.1 Zu Beginn

Mit meinen Vorbereitungen für die summative Abgabe fing ich im Prinzip bereits zu Beginn des Moduls an, als ich mir im Canvas die Vorgaben für die Abschlussarbeit durchgelesen habe. Ich denke das war wichtig, weil es mir geholfen hat, meine Konzentration während der Zeit im Modul auf die für die Abgabe relevanten Dinge zu richten.

Anfangs war ich ein bisschen verunsichert, weil ich die Vorgaben nicht einschätzen konnte und ich vor diesem Modul kaum ausserhalb von Unity programmiert habe. Die Unsicherheit hat sich aber glücklicherweise relativ schnell gelegt, als wir in der ersten Vorlesung einen Probetest gemacht haben und ich gesehen habe, dass mein Mathethematik und Programmier-Knowhow auf dem Niveau sind, die das Modul verlangt.

2.2 Vorlesungen und Praktika

Die weiteren Vorlesungen und Praktika haben uns das nötige Basiswissen vermittelt um später souverän mit OpenGL interagieren zu können.

Ich habe gelernt,

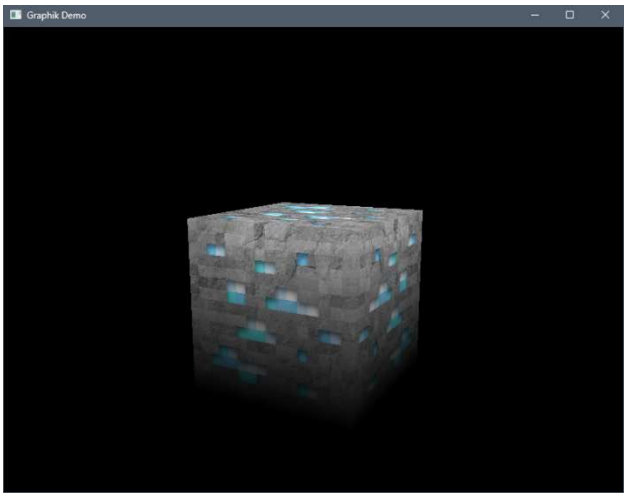
- welche mathematischen Teilgebiete relevant für die Graphikprogrammierung sind und wie ich sie anwende.
- dass OpenGL eine Spezifikation einer Graphik API ist.
- aus welchen Bestandteilen 3D-Modelle bestehen.
- was Vertex Array Objects und Vertex Buffer Objects sind, wie sie aufgebaut sind und wie diese von der CPU auf die GPU übergeben werden.
- was Shader sind und wie man sie mit Programmiersprachen wie z.B. GLSL programmiert.

Die Liste wäre eigentlich noch viel länger, aber ich belasse es beim Wichtigsten. In diesem Modul habe ich sehr viel gelernt und all diese Learnings haben schlussendlich zum Endergebnis – der summativen Abgabe – mit beigetragen.

Was mir sehr geholfen hat, den Stoff zu verinnerlichen, war die Strukturierung des Unterrichts. Mittwochs bekamen wir von Ismael den Stoff vorgetragen und donnerstags waren wir als Klasse gefragt, den Stoff wiederzugeben. Durch die aktive Abfrage am Donnerstag wurde man auf die Probe gestellt. Das sorgte bei mir dafür, dass ich am Mittwoch besser aufgepasst habe, was wesentlich zu meinem Lernerfolg beigetragen hat.

2.3 Formative Abgabe

Die Formative Abgabe bildete den Prototyp für meine Summative. Sie beinhaltete eine kleine Szene mit einem sich rotierenden Würfel und einer beweglichen Kamera. Die bewegliche Kamera zu implementieren war eine Herausforderung und kostete mich viel Zeit. Was daran herausfordernd war und wie ich es implementiert habe, werde ich näher im Kapitel «Herausforderungen und Umsetzung» beschreiben. Abgesehen von der Kamera war die Umsetzung aber problemfrei, da ich alle nötigen Tools von den vorherigen Praktika bereits zur Verfügung gestellt bekommen hatte.



Ein Screenshot von meiner formativen Abgabe.

2.4 Planung der Summativen

Ich wartete mit der eigentlichen Planung der Summativen bis ca. 3 Wochen vor der Abgabe, weil ich noch möglichst viele Unterrichte und Praktika gemacht haben wollte, um zu vermeiden nicht unnötig Themen selbstständig aufzuarbeiten, die wir später noch gemeinsam angeschaut hätten.

Meine Planung bestand unter anderem daraus, die Vorgaben durchzulesen und davon eine To-Do-Liste mit konkreten Handlungsschritten herauszufiltern, die mein Projekt in einen abgabewürdigen Zustand bringen würden. Ich habe die Liste dann nach Wichtigkeit geordnet:

1. Implementierung eines Shaders mit einem simplen «Metallic» Workflow
2. Implementierung eines Shaders mit einem simplen «Specular» Workflow
3. Implementierung einer Skybox
4. Projekt aufräumen und umstrukturieren.
5. Implementierung eines Wavefront OBJ – Loaders

Während der Umsetzung, habe ich mich leider nicht an meine Prioritäten gehalten, was dazu führte, dass ich den Shader mit dem simplen «Specular» Workflow fast nicht mehr implementieren konnte.

Ich hatte Glück, dass ich mir genügend Zeit eingeplant habe und im Grossen und Ganzen alles von dem was ich mir vorgenommen habe, umsetzen konnte.

3 Herausforderungen und Umsetzung

Viele Dinge, die wir in diesem Modul gemacht haben, waren für mich neu und somit gab es auch einige Herausforderungen, die es zu meistern galt.

3.1 Visual Studio Projekte

Die Herausforderungen begannen bereits beim Aufsetzen des Visual Studio Projektes für C++. Vor dem zweiten Praktikum war mir nicht bewusst, wie wenig ich mich mit dem Aufbau eines Visual Studio Projektes auskenne, vor allem nicht mit einem C++ - Projekt.

Meine Ambition bei den Praktika war es, die Dinge so weit zu verstehen, dass ich sie ohne fremde Hilfe zu einem späteren Zeitpunkt replizieren konnte. Das hiess für mich auch, dass ich mich nicht einfach strikt an die Vorgaben der Praktika hielt, um möglichst wenig Widerstand bei der Bearbeitung zu haben, sondern dass ich ein wenig meine Grenzen austeste, um dabei möglichst viel zu lernen.

Im Falle von Visual Studio Projekten hiess das konkret,

- die Projekte von VS 2017 auf VS 2022 upzugraden und alle damit verbundenen Probleme selbst zu lösen.
- mich mit MSBuild auseinanderzusetzen und zu verstehen wie die XML-Datei eines VS-Projektes aufgebaut ist.
- das Zielframework der C# Projekte auf .NET 6.0 zu wechseln.
- die Ziel Prozessorarchitektur auf «AnyCPU» zu wechseln.

All diese Dinge brachten Probleme mit sich, die ich aber mit genügend Zeit und Geduld lösen konnte. Ich hätte mich natürlich einfach strikt an die Vorgaben der Praktika halten können und mir einiges an Zeit gespart. Jedoch war es unter anderem diese Vorarbeit, die mir die eigentliche Arbeit an der summativen erleichtert hat.

Weil ich mich dadurch besser mit dem Aufbau von Visual Studio Projekten auskannte, konnte ich Fabian und Marvin helfen, die auch Probleme beim Aufsetzen der Projekte hatten.

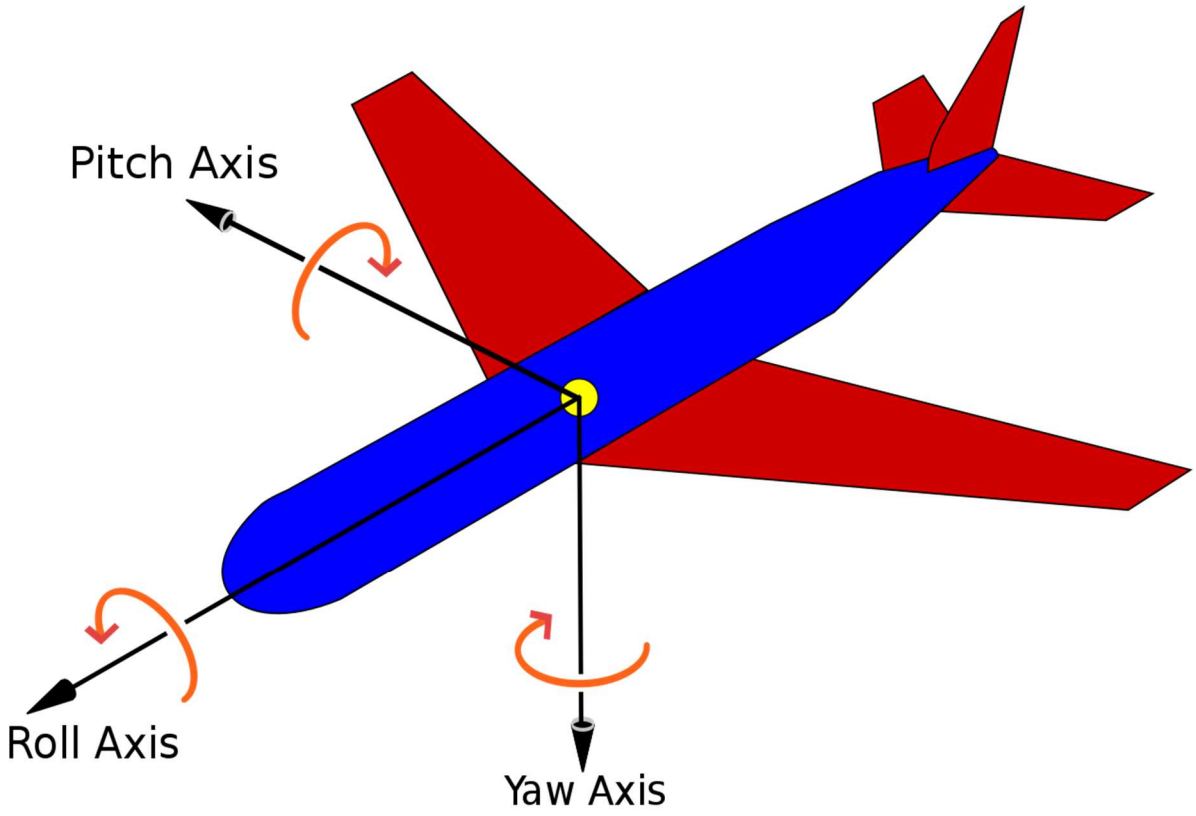
Im Nachhinein betrachtet bin ich sehr froh, diese extra Schritte gemacht zu haben, weil ich denke, dass es mich langfristig zu einem besseren Programmierer machen wird.

3.2 Bewegliche Kamera

Die nächste grössere Hürde, die überwunden werden musste, war es eine bewegliche Kamera zu implementieren.

Mein grösstes Problem bei dieser Aufgabe war die Recherche. Ich wusste anfangs nicht, was es ist nach dem ich genau suche. Nachdem ich mich durch mehrere Foren gelesen habe, stiess ich schliesslich auf das Konzept der Roll-Nick-Gier Winkel (Roll, Pitch and Yaw auf Englisch).

- Roll repräsentiert die Rotation um die lokale Z-Achse.
- Pitch repräsentiert die Rotation um die lokale X-Achse.
- Yaw repräsentiert die Rotation um die lokale Y-Achse.



(Wikipedia, *2010)*

Mit diesen Winkeln ist es möglich die lokalen vorwärts, aufwärts und rechts – Vektoren zu bestimmen, was genau das war, dass ich gebraucht habe.

Die Formeln dafür fand ich im Internet und schrieb sie als Methoden in mein Projekt.

3.3 OBJ-Loader

Für das Parsen von Wavefront OBJ – Dateien, habe ich mich dafür entschieden ein Third Party – Library zu benutzen. Das integrieren des Projekts in meine bestehende Projektmappe war relativ simpel, jetzt wo ich mich besser mit Visual Studio auskannte.

Die eigentliche Herausforderung hier war es, die Daten (Vertices, UVs, Normals) vom OBJ-Format in ein Format zu bringen, mit dem OpenGL arbeiten kann.

Auf Stackoverflow konnte ich einen Algorithmus finden, der mir genau das macht. Er war jedoch sehr unoptimiert und langsam.

Der Algorithmus arbeitete mit der IndexOf()-Methode von der List Klasse, die eine Komplexität von O(n) hat. Er überprüfte damit alle 3 Listen von Vertices, UVs und Normals in jeder Iteration, um duplizierte Daten zu verhindern. Mein erster Gedanke war es, die IndexOf() – Methode einfach wegzulassen und duplizierte Daten in Kauf zu nehmen.

Ich probierte es, der Algorithmus war sehr viel schneller, aber duplizierte unnötigerweise 4000 Vertices, die das Rendering verlangsamen würden und kostbaren VRAM-Platz einnehmen würden. Aus diesem Grund entschied ich mich weiter nach einem besseren Weg zu suchen.

Mein nächster Gedanke war es Dictionarys oder Hashmaps zu verwenden, da deren Look-Up Zeit ~O(1) ist. Nach einigen misslungenen Versuchen ging ich mit dieser Idee zu ChatGPT und fragte um Rat.

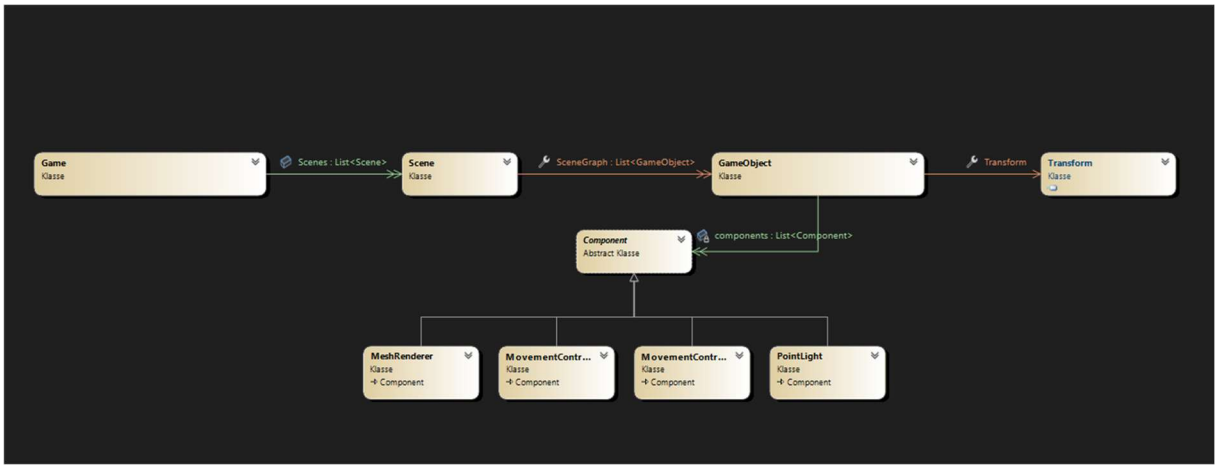
Das fehlende Puzzleteil, war die Tuple Struktur. Mit Ihr konnte ich einzigartige 3er Pärchen von Vertices, UVs und Normals in einem Dictionary speichern und mit einem effizienten einmaligen Look-Up überprüfen, ob ein solches Pärchen bereits existiert.

Mit der Hilfe von ChatGPT konnte ich so also den Algorithmus sehr stark optimieren und auch hier denke ich, dass sich dieser extra Aufwand gelohnt hat, weil ich sehr viel dabei gelernt habe.

3.4 Projekt-Architektur

Bei der Architektur meines Programms habe ich es mir einfach gemacht und mich an der Architektur von Unity orientiert.

Hier eine übersichtliche Form meines Klassendiagrams:



Um meinen Code in dieses Format zu bringen, musste ich eine Menge umstrukturieren. Die Schwierigkeit dabei war, dass ich nicht in kleinen Schritten vorgehen konnte, sondern direkt über hundert Zeilen Code verschieben musste und dabei trotzdem sicherstellen musste, dass die ursprüngliche Funktionalität erhalten bleibt.

Der Grund warum ich so vorgehen musste, war das ich anfangs viele Dinge wie z.B. die Kamera und das Point Light als statische einzelne Instanzen in der «Program» Klasse gehalten habe um schnell Ergebnisse zu sehen.

Besser wäre es gewesen sich diese Gedanken am Anfang des Projekts zu machen, um eine solche Riesenaktion zu verhindern.

4 Was gibt es zu verbessern?

4.1 Bessere Planung und Durchführung der Planung

Ich habe es bereits im Kapitel «Herausforderungen und Umsetzung» erwähnt, aber ich hätte einiges besser machen können, wenn es darum geht nach Prioritäten zu arbeiten. Hätte ich nicht alle Dinge geschafft, die ich mir vorgenommen hätte und zu stark unter Zeitdruck geraten wäre, hätte ich einige der Vorgaben für die Summative nicht erfüllt.

Für meine nächsten Arbeiten sollte ich mich besser an Prioritätenliste halten und weniger von den Essenziellen Aufgaben abweichen, bevor ich mit anderen Dingen anfange.

Wenn ich die Zeit zurückdrehen könnte, würde ich mir ausserdem von Anfang an mehr Zeit für die Planung der Architektur meiner Software nehmen. Natürlich ist das im Nachhinein einfach gesagt, wenn man alles bereits vor sich sieht. Jedoch hätte ich mir, wenn ich mir ein wenig mehr Gedanken gemacht hätte, durchaus einiges einfacher gemacht.

4.2 Offene Baustellen

Das Projekt hat noch einige offene Baustellen, zu denen ich leider nicht mehr gekommen bin:

- Szenen unterstützen momentan nur ein aktives Punktlicht und gerichtetes Licht.
- Viele Parameter wie die Ambient Color sind momentan noch globale Eigenschaften einer Szene.
- Viele Parameter hätten in eine Matrix gepackt werden können, um eine effizientere Übergabe von CPU auf GPU zu gewährleisten.
- Die Transform Klasse erbt nicht von der Component Klasse.
- Ich muss als Model Matrix die SRT Matrix des Transforms nehmen, um lokal Skalieren und Rotieren zu können (und eigentlich sollte es die TRS Matrix sein)

5 Schlusswort

Dieses Modul war für mich mit Abstand, das Modul an der SAE, in dem ich die grössten Fortschritte als Spielentwickler gemacht habe. Dadurch das wir viel «hinter die Kulissen» geschaut haben und uns angeschaut haben wie z.B. 3D-Modelle oder eine Game-Engine in Ihren Grundsätzen aufgebaut sind, haben sich für mich sehr viele Fragen und Unsicherheiten geklärt, die ich vorher gehabt habe.

Auch wenn wir viel gefordert wurden mit den Praktika und der aktiven Abfrage im Unterricht hat mir dieses Modul sehr viel Spass gemacht und ich bin sehr stolz auf die Fortschritte die ich hier gemacht habe.

6 Quellenangaben

Ambrye et al (2022) *opengl4csharp* [Online]. Available at https://github.com/giawa (Accessed: 15.08.2023)

Ambrye et al (2022) gui4*opengl4csharp* [Online]. Available at https://github.com/giawa (Accessed: 15.08.2023)

De Vries, J. (no date) *Cubemaps* [Online]. Available at https://learnopengl.com/Advanced-OpenGL/Cubemaps (Accessed: 15.08.2023)

GameReady Cottage 3D-Modell (2019) [Online]. Available at https://free3d.com/3d-model/gameready-cottage-free-163528.html (Accessed: 15.08.2023)

Jansson et al (2018) *ObjLoader* [Online]. Available at https://github.com/chrisjansson/ObjLoader (Accessed: 15.08.2023)

Jönsson et al (2001) *Bitmap Font Generator* [Online]. Available at https://www.angelcode.com/ (Accessed: 15.08.2023)

Wikipedia (2010) *Aircraft principal axes* [Online]. Available at https://en.wikipedia.org/wiki/Aircraft_principal_axes (Accessed: 15.08.2023)

3D Gear 1 (2023) [Online]. Available at https://www.turbosquid.com/3d-models/3d-gear-1-2018640 (Accessed: 15.08.2023)