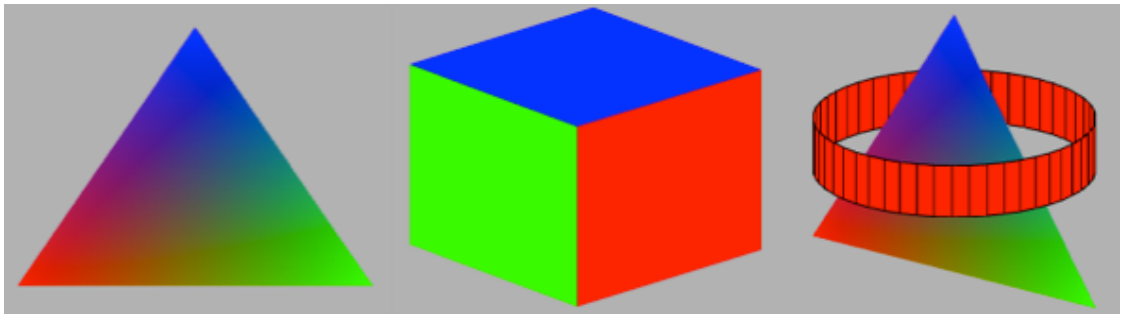
Computergrafik 2: Aufgabe 2, Teil 1/2.

3D-Geometrie und Vertex Buffer Objects



Lernziele / Motivation

In dieser Übung machen Sie sich mit der Darstellung von Geometrie durch Punkte, Linien und Dreiecke in WebGL vertraut und lernen den Umgang mit *Vertex Buffer Objects, Attribute Buffers und Index Buffers*.

Übungsframework

Für diese Übung laden Sie sich bitte das Mini-Framework zur Aufgabe 2 aus Moodle herunter und entpacken Sie es in einem entsprechenden Verzeichnis neben dem zur Aufgabe 1. Bitte beachten Sie, dass das Verzeichnis `lib/` aus Aufgabe 1 auf dem gleichen Verzeichnis-Niveau zu Verfügung steht! Für eine Übersicht des Frameworks studieren Sie bitte die Folien zur SU-Einheit "WebGL-Einführung".

Aufgabe 2.1: `Triangle` und Per-Vertex-Attribute

Studieren Sie das Modul `models/triangle.js`. Es modelliert ein Dreieck, stellt dieses jedoch nur als drei Punkte dar. Ändern Sie dieses Modul so, dass das Dreieck als eine Fläche (bestehend aus einem Dreieck) gezeichnet wird. Fügen Sie einen weiteren Attribut-Buffer hinzu, der eine Farbe pro Vertex definiert, binden Sie diesen Buffer an das Vertex-Attribut (die Shader-Variable) "`vertexColor`", und verwenden Sie für das Dreieck in `MyScene.draw()` das Programm `prog_vertexColor` anstelle des Programms `prog_red` (Vorsicht: nur für `Triangle`, nicht für die anderen Objekte!).

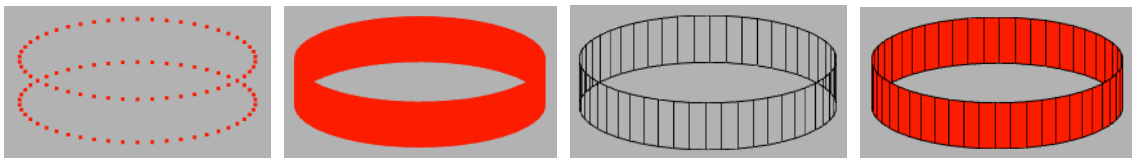
Aufgabe 2.2: `Cube`, Index Buffers und `drawElements()`

Studieren Sie das Modul `models/cube.js`. Es modelliert die sechs Seiten eines Würfels jeweils mittels separater Vertices (d.h. ein Vertex wird für jede Seite noch einmal wiederholt und existiert somit in jeweils drei Instanzen). Die Darstellung erfolgt jedoch lediglich als Punkte...

Erweitern Sie das Modul `cube.js` so, dass die Oberfläche des Würfels mit Hilfe von Dreiecken gezeichnet wird. Hierzu benötigen Sie einen zusätzlichen Index-Buffer (`vbo.Indices`), in dem jeweils die richtigen Vertices zu einem Dreieck verbunden werden (z.B. besteht die Vorderseite aus den beiden Dreiecken ABC und ACD). In der `draw()`-Methode des Cube muss der `drawArrays()`-Aufruf durch einen `drawElements()`-Befehl ersetzt werden.

Fügen Sie dem `Cube` Per-Vertex-Farben hinzu, so dass jede *Seite* des Würfels mit einer eigenen konstanten Farbe dargestellt wird (z.B. oben und unten rot, links und rechts grün, vorne und hinten blau). Verfahren Sie hierbei analog zu `Triangle`.

Aufgabe 2.3: `Band` und Oberfläche vs. Wireframe



Erweitern Sie bitte das Modul `models/band.js`, welches ein "Band" oder einen "Ring" darstellen soll. Das Band wird in einer `for`-Schleife durch Vertices konstruiert, die auf zwei parallelen Kreisen liegen. Implementieren Sie die analog zum Würfel die Darstellung der Oberfläche mittels Dreiecken. Die ersten beiden Dreiecke des Bandes haben z.B. die Indizes [0, 1, 2, 2, 1, 3].

Implementieren Sie zusätzlich eine Wireframe-Darstellung, bei welcher der Ring so dargestellt wird, als sei er aus Linien zu Rechtecken zusammengesetzt. Übergeben Sie im `config`-Objekt des Konstruktors von `Band` einen weiteren Parameter `asWireframe`, der angibt, ob das Band als Wireframe oder als Oberfläche dargestellt werden soll. Erlauben Sie mittels der `drawOptions` in `main.js` dem Benutzer, zwischen Wireframe- und der Oberflächendarstellung zu wählen. Wählen Sie für die Wireframe-Darstellung des Bands eine andere Farbe als für die Oberflächen-Darstellung, indem Sie das eine Objekt mittels `prog_red` zeichnen, und das andere mittels `prog_black`.

Aufgabe 2.4: Z Fighting, Depth Test, Backface Culling (nur für sehr gute Note)

Diese Aufgabe übt einige weitere grundsätzliche Optionen der OpenGL-Darstellung und muss nur bearbeitet werden, wenn Sie eine sehr gute Note anstreben.

1. Wenn Sie nun Wireframe und Oberfläche gleichzeitig darstellen, werden Sie den sogenannten "Z Fighting" Effekt erleben - dort wo Linien und Flächen gleichzeitig sind, weiß WebGL nicht, welches der beiden es zeichnen soll. Recherchieren und verwenden Sie den WebGL-Befehl `polygonOffset`, um dieses Problem zu lösen.

2. Bauen Sie eine `drawOption` ein, mit der Sie den Tiefentest ein- und ausschalten können (siehe `gl.enable(GL_DEPTH)` in der Szene), und studieren Sie den Effekt.
3. Experimentieren Sie mit dem sogenannten *Backface Culling* von OpenGL. Studieren Sie den Befehl `gl.cullFace()` sowie das zugehörige `glEnable()`, und bauen Sie entsprechende Umschalter in das UI ein, die es ermöglichen, die Vorderseite und/oder Rückseiten der Polygone zu zeichnen. Verstehen Sie, welchen Effekt diese Befehle auf den Würfe und das Band haben, und korrigieren Sie ggf. die Reihenfolge der Indizes in Ihren Buffern (Stichwort: *Winding Order*), so dass Vorder- und Rückseite konsistent dargestellt werden.

Abgabe

Diese Aufgaben sind der erste von zwei Teilen der Aufgabe 2 und sollte innerhalb von zwei von vier Wochen bearbeitet werden. Die Abgabe der gesamten Aufgabe 2 soll via Moodle bis zu dem dort angegebenen Termin erfolgen. Verspätete Abgaben werden wie in den Handouts beschrieben mit einem Abschlag von 2/3-Note je angefangener Woche Verspätung belegt. Geben Sie bitte pro Gruppe jeweils nur eine einzige `.zip`-Datei mit den Quellen Ihrer Lösung sowie mit den ggf. geforderten Screenshots ab.

Demonstrieren und erläutern Sie dem Übungsleiter Ihre Lösung *in der nächsten Übung nach dem Abgabetag*. Die Qualität Ihrer Demonstration ist, neben dem abgegebenen Code, ausschlaggebend für die Bewertung! Es wird erwartet, dass alle Mitglieder einer Gruppe anwesend sind und Fragen beantworten können.