Computer Graphics - WebGL, Teil 5

Solid Würfel

Als nächstes möchten wir einen ausgefüllten, farbigen Würfel zeichnen. Der Vertex Buffers und der Element Buffers vom Draht-Modell Würfel könnten wir zwar dazu verwenden, aber dann könnten wir Farben nur an den einzelnen Ecken angeben, was zu einem komischen Farbverlauf führen würde (Abb. 1).

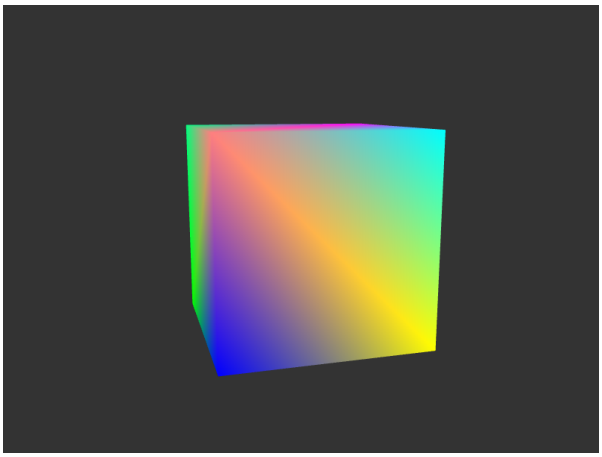

Abbildung 1: Würfel mit gemeinsamen, farbigen Eckpunkten

.

Wenn wir Flächen in einer Farbe zeichnen möchten, müssen wir deshalb jeden Vertex 3x angeben, einmal für jede Fläche in der er vorkommt. So können wir dann pro Vertex und Fläche eine Farbe zuordnen (Abb. 2). Dies werden wir später auch weiter so benutzen um Normalen-Vektoren zu den Seiten anzugeben, was wir dann für die Beleuchtung brauchen. Generell werden separate Vertices dann eingefügt, wenn eine Kante modelliert werden soll. Ist an der Vertices keine Kante vorhanden, zum Beispiel bei einem eigentlich runden Objekt, gibt es nur einen Vertex.

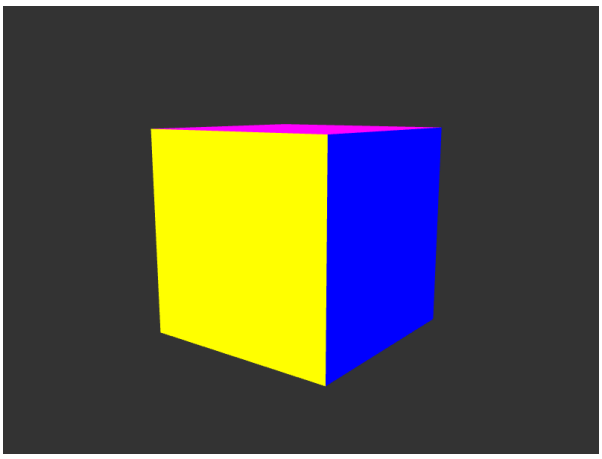

Abbildung 2: Würfel mit farbigen Seiten.

.

Backface Culling und z-Buffer

WebGL stellt 2 Methoden zur Verfügung um zu verhindern, dass nicht sichtbare Flächen gezeichnet werden: Backface Culling und z-Buffer.

Beim Backface Culling werden die hinten liegenden, und damit verdeckten Flächen nicht gezeichnet. Dazu muss WebGL jedoch wissen, welche Flächen nach hinten liegen. Dies geschieht mir der Reihenfolge in der die Eckpunkte (Vertices) der Dreiecke spezifiziert sind. Die Konvention ist, dass die Ecken von aussen gesehen im Gegenuhrzeigersinn angegeben werden. WebGL kann dann aus den 3 Eckpunkten einen Normalenvektor berechnen und anhand von diesem entscheiden, ob die Fläche gezeichnet werden soll oder nicht. Backface Culling muss entsprechend eingeschalten werden mittels `gl.enable`, die weiteren unten aufgeführten Befehle entsprechen der default Konfiguration von WebGL.

```
gl.frontFace(gl.CCW);        // defines how the front face is drawn
gl.cullFace(gl.BACK);        // defines which face should be culled
gl.enable(gl.CULL_FACE);     // enables culling
```

Aufgabe 1: Zeichnen sie einen ausgefüllten Würfel mit verschieden farbigen Seiten. Der Code für den Würfel soll in einem eigenen javascript file analog der Lösung zum Drahtgitter Würfel abgelegt werden.

Backface Culling allein funktioniert nicht mehr, wenn sich mehrere Objekte überlagern. WegGl verwendet deshalb zusätzlich den z Buffer Algorithmus um herauszufinden, welche Pixel eines Objekts von Pixeln eines anderen Objekts verdeckt werden. Dazu wird die Tiefe des Pixels zusätzliche zu seiner Farbe gespeichert und dann beim Zeichnen einen Tiefenvergleich durchgeführt. Um diese Funktion zu verwenden muss der Vergleich mittels `gl.enable(gl.DEPTH_TEST)`

aktiviert werden und vor dem Zeichnen der Tiefenbuffer mit `gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);` gelöscht werden.

```
// in init
gl.enable(gl.DEPTH_TEST);
...
// in draw
gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT)
```

Texture Mapping für 3D Objekte

Aufgabe 2: Implementieren sie Texture Mapping für die Seiten des Würfels wie in Abb. 3 dargestellt. Dabei müssen, wie im 2D Beispiel zu Texture Mapping, an jedem Vertex die Textur-Koordinaten angegeben werden.

Um Texture Mapping im Fragment Shader ein- und ausschalten zu können, empfiehlt es sich eine uniform Variable einzuführen, die dann im Javascript Programm gesetzt und im Shader abgefragt werden kann.

```
precision mediump float;
varying vec4 vColor;
varying vec2 vTextureCoord;

uniform sampler2D uSampler;
uniform int uEnableTexture;

void main() {
    if (uEnableTexture == 0) {
        gl_FragColor = vColor;
    }
    else {
       gl_FragColor = texture2D(uSampler, vTextureCoord);
    }
}
```

Aufgabe 3: Stellen sie 2 um ihre eigene Achse rotierende Würfel dar: einen farbigen Würfel, sowie einen Würfel mit Texture Mapping.

Aufgabe 4* (optional): Zeigen Sie im gleichen Fenster 4 verschiedene Ansichten des Würfels dar. Benutzen Sie dazu `gl.viewport`.

Aufgabe 5* (optional): Definieren und Zeichnen Sie eine Kugel.

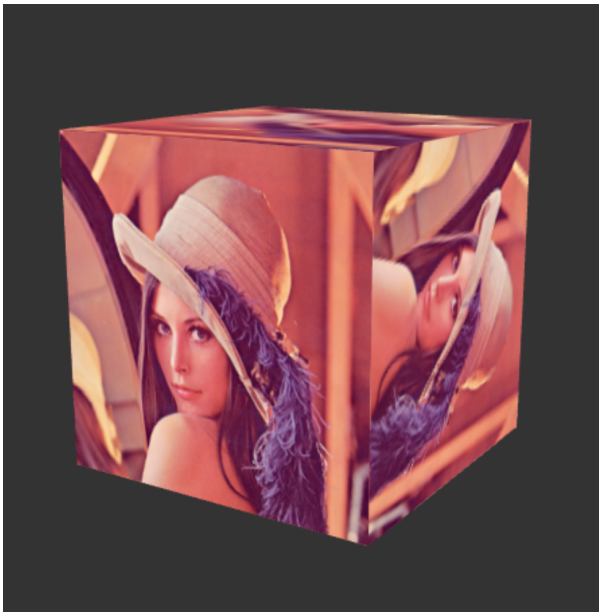

Abbildung 3: Würfel mit Texture Mapping

.