Computer Graphics – WebGL, Teil 4

1 Zeichnen mit Element Array

Als erstes einfaches Object in 3D möchten wir einen Würfel als Drahtgittermodell (Wire Frame) zeichnen. Ein Würfel hat acht Ecken (Vertices), und wir wollen die Kanten zwischen den Ecken zeichnen. Dazu müssten wir jedoch jeden Vertex mehrmals spezifizieren, da er jeweils für drei Kanten gebraucht wird. Es gibt jedoch in WebGL die Möglichkeit, einen definierten Vertex mehrmals zu verwenden, indem ein *Element Array* definiert wird.

Bei einem *Element Array* wird ein zusätzlicher Buffer definiert, der die Indices der Vertices enthält, die wir zeichnen möchten. An Stelle von `gl.drawArrays()` wird dann `gl.drawElements()` verwendet.

Das folgenden Beispiel zeichnet zwei Linien mit einem gemeinsamen Vertex. Da wir ab jetzt in 3D zeichnen, werden die Vertices jeweils durch drei Werte für x, y und z spezifiziert. Entsprechend muss im Vertex Shader nun ein Attribut vom Typ `vec3` verwendet werden. Die sieht etwa so aus:

```
...

// define

var vertices =
[
    0, 0, 0,   // v0
    1, 0, 0,   // v1
    1, 1, 0,   // v2
];

vertexBuffer  = gl.createBuffer();
gl.bindBuffer (gl.ARRAY_BUFFER, vertexBuffer);
gl.bufferData (gl.ARRAY_BUFFER, new Float32Array (vertices),
    gl.STATIC_DRAW);

var vertexIndices =
[
    0, 1,
```

```
    1, 2,
    0, 2
];

edgeBuffer = gl.createBuffer();
gl.bindBuffer (gl.ELEMENT_ARRAY_BUFFER, edgeBuffer);
gl.bufferData (gl.ELEMENT_ARRAY_BUFFER, new Uint16Array(vertexIndices),
    gl.STATIC_DRAW);

...

// draw

gl.bindBuffer (gl.ARRAY_BUFFER, vertexBuffer);
gl.vertexAttribPointer (aVertexPositionId, 3, gl.FLOAT, false, 0, 0);
gl.enableVertexAttribArray (aVertexPositionId);
gl.bindBuffer (gl.ELEMENT_ARRAY_BUFFER, edgeBuffer);
gl.drawElements (gl.LINES, 6 /* Anzahl Indices */ , gl.UNSIGNED_SHORT, 0);
```

Um zukünftig mehrere Objekte zeichnen zu können, empfiehlt es sich, diese in eigenen JavaScript-Files zur Verfügung zu stellen. Hier ist ein Vorschlag, wie ein Würfelobjekt erzeugt und gezeichnet werden kann. Verwendet wird in JavaScript eine Konstruktorfunktion, die mittels `new` neue Objekte erzeugt. Danach können dann Methoden des Objekts aufgerufen werden.

```
/**
 * Define a wire frame cube with methods for drawing it.
 *
 * @param gl the webgl context
 * @param color the color of the cube
 * @returns object with draw method
 * @constructor
 */

function WireFrameCube (gl, color)
{
    function defineVertices (gl)
    {
        // define the vertices of the cube

        var vertices =
        [
            ...
        ];

        var buffer  = gl.createBuffer();
        gl.bindBuffer (gl.ARRAY_BUFFER, buffer);
        gl.bufferData (gl.ARRAY_BUFFER, new Float32Array(vertices),
            gl.STATIC_DRAW);
        return buffer;
    }

    function defineEdges (gl)
    {
```

```
        // define the edges for the cube, there are 12 edges in a cube

        var vertexIndices =
        [
            ...
        ];

        var buffer = gl.createBuffer();
        gl.bindBuffer (gl.ELEMENT_ARRAY_BUFFER, buffer);
        gl.bufferData (gl.ELEMENT_ARRAY_BUFFER,
            new Uint16Array (vertexIndices), gl.STATIC_DRAW);
        return buffer;
    }

    return {
        bufferVertices: defineVertices (gl),
        bufferEdges: defineEdges (gl),
        color: color,

        draw: function (gl, aVertexPositionId, aVertexColorId)
        {
            ...
        }
    }
}
```

Verwendet wird das Objekt im Main-JavaScript-Programm dann mit

```
// definition

wiredCube = new WireFrameCube (gl, [1.0, 1.0, 1.0, 0.5]);
...

// in draw

wiredCube.draw (gl, ctx.aVertexPositionId, ctx.aVertexColorId);
```

2 3D Darstellung

2.1 Kameraposition

Zur Definition der Kameraposition wird die ModelView-Matrix benutzt. Diese hatten wir in der letzten Übung für Modelltransformationen verwendet, um ein Objekt zu transformieren, und nun benutzen wir sie auch für die Kameraposition und -richtung (View), deshalb auch der Name *ModelView*-Matrix. Die Kameraparameter werden am einfachsten mit dem Befehl `mat4.lookAt()` gesetzt.

`mat4.lookAt (out, eye, center, up)`
: definiert eine Viewing-Transformation-Matrix mit der Kamera an der Position `eye`, Blickrichtung auf `center` und dem gegebenen `up`-Vektor. Der `up`-Vektor bestimmt die Drehung des Bildes um die Blickrichtung, die Projektion dieses Vektors auf die Bildebene definiert die y-Achse des Bildes.

2.2 Projektion

Die 3D Darstellung in WebGL erfolgt über eine Projektionsmatrix, welche die 3D-Vertex-Koordinaten in 2D-Koordinaten transformieren. Die Projektionsmatrix muss als weitere `uniform`-Variable an den Vertex Shader übergeben und dort zur Transformation der Vertices verwendet werden.

Frage: Muss die Projektions Matrix vor oder nach der ModelView Matrix angewendet werden?

Die Projektionsmatrix kann mit einem der folgenden Befehle gesetzt werden:

`mat4.ortho (out, left, right, bottom, top, near, far)`
: definiert eine orthogonale Projektion mit den entsprechenden Grenzen.

`mat4.frustum (out, left, right, bottom, top, near, far)`
: definiert eine perspektivische Projektion, in der das Viewing Frustum ähnlich wie bei `gl.ortho()` direkt gesetzt wird.

`mat4.perspective (out, fovy, aspect, near, far)`
: definiert eine perspektivische Projektion mit einem bestimmten Öffnungswinkel und einer bestimmten Aspect Ratio.

Um Kameraposition, Perspektive und Würfel am richtigen Ort zu plazieren, empfiehlt es sich, eine Skizze zu zeichnen!

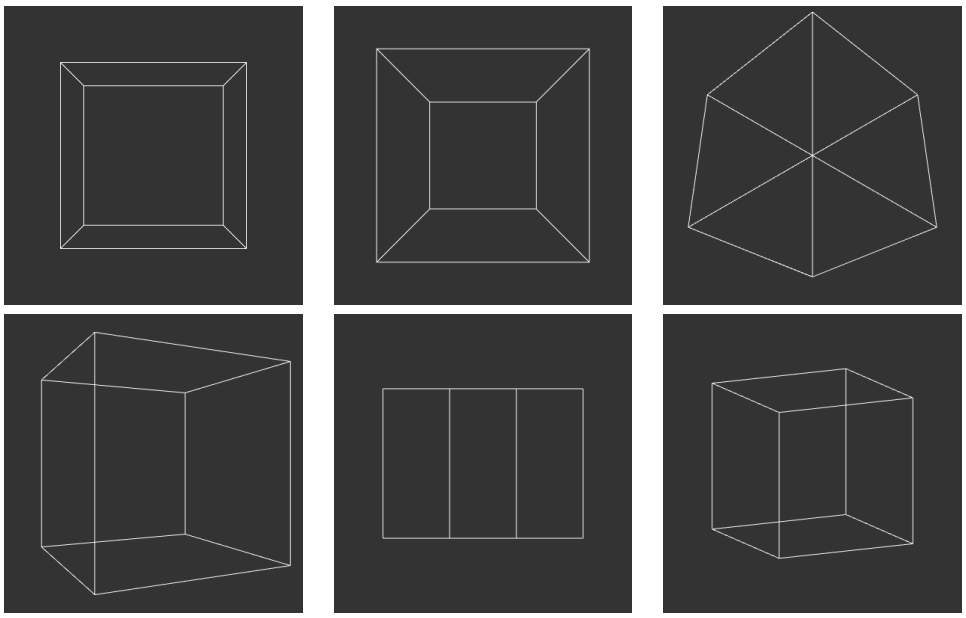
Aufgabe 1

Definieren sie den Würfel, sowie die Projektionsmatrix und die Kameraposition, um den Würfel richtig darzustellen.

Probieren Sie dazu verschiedene Projektionen und Kamerapositionen aus.

Aufgabe 2

Wie erreichen Sie die folgenden Darstellungen?



Aufgabe 3

Rotieren Sie den Würfel in einer Animation um die eigene Achse.

Aufgabe 4*

In Aufgabe 3 war zu sehen, dass Bewegung die Wahrnehmung der räumlichen Tiefe in der computergraphischen Darstellung verbessert. Eine weitere Methode ist das Depth Cueing: Dabei werden weiter vorn liegende Teile der Geometrie heller dargestellt als die weiter hinten liegenden. Man hat den Eindruck, als verschwinde der hintere Teil der Geometrie im Nebel. Modifizieren Sie den Fragment Shader, so dass Sie diesen Effekt erzielen.

Aufgabe 5*

Eine weitere Methode zur Verbesserung der Tiefenwahrnehmung ist die Stereosicht. Für jedes Auge wird ein eigenes Bild gezeichnet. Durch den Abstand der beiden Augen sind die beiden Bilder leicht unterschiedlich, und das menschliche Gehirn kann aus diesen Unterschieden die räumliche Tiefe rekonstruieren. Wie werden nun die beiden Bilder dargestellt? Eine Möglichkeit ist, das linke Bild in roter Farbe und das rechte Bild in türkiser Farbe zu zeichnen. Durch Betrachtung mit einer Rot-Cyan-Anaglyphenbrille gelangen beide Bilder in das jeweils richtige Auge. Mit dem Befehl `gl.colorMask (r, g, b, a)` können Sie durch Angabe von `true` oder `false` das Zeichnen in die entsprechenden Farbkanäle erlauben oder verbieten.