Computergrafik
Übungsblatt 5

Aufgabe 1 Kreiszeichen in WebGL

Öffnen Sie die Datei Circle.html in einem Browser und editieren Sie die Datei Circle.js um folgende Aufgaben zu lösen.

(a) Ergänzen Sie die Methode `initScene` wie folgt: Erzeugen Sie einen Vertex-Buffer, der zwei 2D Dreiecke beinhaltet und das komplette WebGL-Canvas abdeckt (z.B. Dreieck A: $[-1,-1],[-1,1],[1,1]$ Dreieck B: $[1,1],[1,-1],[-1,-1]$).

Packen Sie den Vertex-Buffer in ein Vertex-Array. Ergänzen Sie die Methode `drawScene` soweit, dass die zwei 2D Dreiecke gezeichnet werden.

Hinweise: gl.bindBuffer, gl.bufferData, gl.createVertexArray, gl.bindVertexArray, gl.enableVertexAttribArray, gl.vertexAttribPointer, gl.drawArrays.

(b) Der Vertex-Shader soll nun eine `vec2 v_localCoordinates` Variable an den Fragment-Shader weiterleiten. Diese Variable soll die 2D Eckpunktkoordinaten der Dreiecke über dessen Pixel interpolieren. Der Fragment-Shader soll anschließend die interpolierte Eckpunktkoordinate in den roten und grünen Farbkanal ausgeben. Erweitern Sie dazu den Fragment- und den Vertex-Shader!

(c) Färben Sie nun den roten Farbkanal mit der Länge des Vektors `v_localCoordinates` ein.

Hinweis: Nutzen Sie die GLSL-Funktion length.

(d) Nutzen Sie nun die eben berechnete Länge des Vektors um eine Ellipse zu zeichnen! Pixel innerhalb der Ellipse sollen dabei rot und Pixel außerhalb der Ellipse sollen dabei weiß eingefärbt werden.

(e) Multiplizieren Sie die Koordinaten `v_localCoordinates.xy` im Vertex- oder im Fragment-Shader mit geeigneten Werten durch, so dass Sie statt einer Ellipse einen Kreis zeichnen. Übergeben Sie diese Variablen als `uniform` an die Shader.

Hinweis: Genannte „geeignete Werte“ hängen vom Seitenverhältnis des Canvas ab und ob das Bild höher als breiter oder breiter als höher ist.

(f) Geben Sie dem Kreis einen schwarzen Rand. Die Randbreite soll dabei dem Slider Wert entsprechen.
Hinweise:

- Den Wert des Sliders bekommen Sie über
  `document.getElementById("borderSizeSlider").value;`

- Überlegen Sie sich, was die Größe eines Pixels übertragen auf das Intervall $[-1;\ 1] \times [-1; 1]$ entspricht.

Aufgabe 2 Plasma in WebGL

Öffnen Sie die Datei Plasma.html in einem Browser und editieren Sie die Datei Plasma.js um folgende Aufgabe zu lösen.

Schreiben Sie ein Programm, das mittels WebGL, Vertex-Shader und Fragment-Shader die RGB-Kanäle aller Pixel des WebGL Canvas mit nachfolgender Funktion einfärbt:

$$r(x, y, t) = |\sin(1{,}3 \cdot x \cdot \pi + 0{,}3 \cdot t) \cdot \cos(2{,}3 \cdot y \cdot \pi + 0{,}3 \cdot t)|$$

$$g(x, y, t) = |\sin(1{,}9 \cdot x \cdot \pi - 0{,}7 \cdot t) \cdot \cos(1{,}4 \cdot y \cdot \pi - 0{,}6 \cdot t)|$$

$$b(x, y, t) = |\sin(1{,}5 \cdot x \cdot \pi + 1{,}5 \cdot t) \cdot \cos(2{,}7 \cdot y \cdot \pi + 1{,}3 \cdot t)|$$

Dabei ist $x \in [0{,}1]$,$y \in [0{,}1]$ und es gilt:

- Linker Rand des WebGL Canvas $x = 0$;
- Rechter Rand des WebGL Canvas $x = 1$;
- Unterer Rand des WebGL Canvas $y = 0$;
- Oberer Rand des WebGL Canvas $y = 1$;

Die Variable t ist der Wert des Argumentes `timer` der Methode `drawScene`.

Aufgabe 3 Kreisapproximation durch Dreiecke

Öffnen Sie die Datei CircleApproximation.html in einem Browser und editieren Sie die Datei CircleApproximation.js um folgende Aufgabe zu lösen.

Approximieren Sie einen Kreis mit n Kreissegmenten. Der Wert n wird dabei von einem Slider gesteuert. Vervollständigen Sie die Methode `fillBuffer`, welche den Vertex-Buffer befüllt, sobald sich der Wert des Sliders ändert. Binden sie die Shader und das Vertex-Array-Object in der Methode `drawScene` um die Kreisapproximation zu zeichnen.