Prof. Dr. Elmar Schömer Dr. Kai Werth

JOHANNES GUTENBERG UNIVERSITÄT MAINZ

Übungsblatt 12

1. Aufgabe: Praxis: Shader (1+1+2+2 Punkte)

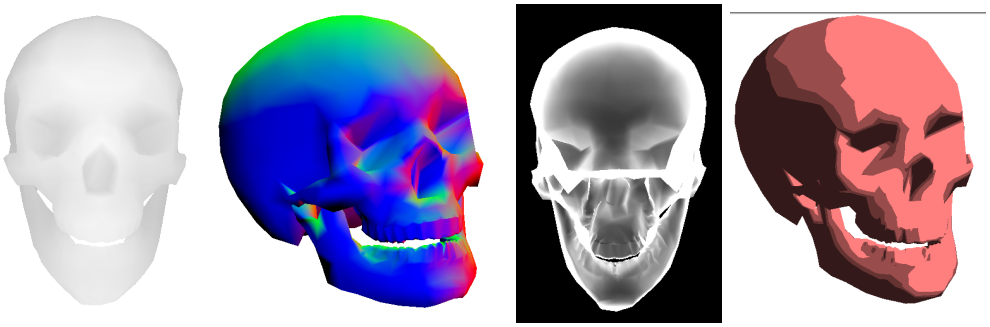
Auf der Homepage finden Sie eine Demo, die ein STL-File einliest und eine Phong-Beleuchtung darauf ausführt. Laden Sie es herunter und schreiben Sie folgende Shader Programme:

(a) Der Shader erzeugt ein gray scale Tiefenbild aus der Szene; je weiter das Pixel von Betrachter entfernt liegt, desto dunkler wird es gemalt.

(b) Der Shader färbt das Pixel seiner Normale entsprechend ein; dabei werden die Normalenwerte zu `RGB`-Werten

(c) Der Shader erzeugt eine Art Röntenbild des `STL`'s. Alle Pixel werden weiss gezeichnet, der Alpha-Kanal (also die Transparenz) ist abhängig von Winkel der Normale des Vertex zur Sichtrichtung; ist der Winkel klein, ist der Alpha-Wert klein, ist der Winkel gross, der Alpha-Wert ebenso.

(d) Der Shader erzeugt ein diskret gefärbtes Bild; erzeugen Sie sich eine diskrete Menge von Farbwerten (im Bild 4 Rot-Töne); mit steigendem Winkel zwischen Lichtquelle und Pixel-Normalen fällt das Pixel in eine der diskreten Farbklassen, und wird in dieser Farbe dann eingefärbt. Hierzu müssen Sie sich geeignete Schwellenwerte überlegen. Schöne Effekte erzielen Sie, wenn die Farben aufsteigend heller erscheinen.



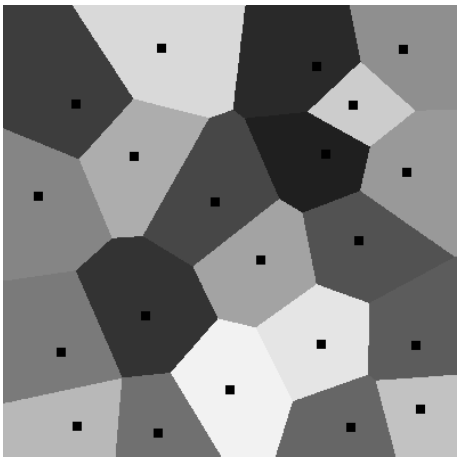
Zum Einfachsten erstellen Sie zu jeder Lösung ein eigenes HTML-File. Sie können auch mehrere Shader verwenden und den aktiven Shader mit Druck der Taste 'S' den aktiven Shader ändern.

2. Aufgabe: Praxis: Voronio-Diagramme und Z-Buffering (Ohne Shader!) (1+3 Punkte)

Gegeben sei eine Menge von Punkten $S = \{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$ in der xy–Ebene. Die Voronoi-Region V_i des Punktes $\mathbf{s}_i$ ist die Menge aller Punkte $\mathbf{p} \in \mathbb{R}^2$, die näher zu $\mathbf{s}_i$ liegen als zu irgendeinem anderen Punkt:

$$V_i = \{\mathbf{p} \in \mathbb{R}^2 \mid \forall j \neq i \ : \ |\mathbf{s}_i - \mathbf{p}| < |\mathbf{s}_j - \mathbf{p}|\}$$

Wir wollen die Voronoi-Regionen $V_1, \ldots, V_n$ in unterschiedlichen Farben malen, indem wir den Mechanismus von OpenGL zur Elimination verdeckter Flächen nutzen. Dazu zeichen wir für jeden Punkt $\mathbf{s}_i$ einen andersfarbigen Kegel mit Öffnungswinkel 90°, dessen Symmetrieachse die (negative) z-Achse ist und dessen Kegelspitze im Punkt $\mathbf{s}_i$ liegt. Dann betrachten wir die Szene unter Orthogonalprojektion in Richtung (negativer) z-Achse.



(a) In `WebGL` existiert keine fertige Methode, einen Kegel zu malen. Doch wir erhalten einen Kegel, wenn wir die Kreiskontur der Leitlinie und die Kegelspitze über einen Triangle-Fan malen.

(b) Schreiben Sie ein `WebGL`-Programm welches obiges Verfahren für eine beliebige Punktmenge implementiert. Es soll möglich sein, die Szene zu rotieren. **Frage**: Was steht im ßWert eines jeden Fragments bei der Draufsicht in negativer z-Achse? Warum ist dieses Verfahren korrekt? Antworten Sie bitte im Quelltext als Kommentar.

3. Aufgabe: Praxis: Voronio-Diagramme und Z-Buffering (Mit Shader!) (2* Punkte)

Obige Kegel können auch im Fragment Shader generiert werden; es wird ein Recheck pro Kegel gerendert welches den Bildschirm ausfüllt. Im Fragment Shader wird dann der Tiefenwert (und Farbe) entsprechend gesetzt. Dazu braucht der Fragment Shader lediglich die Koordinaten der Spitze.