3D-Computergrafik I

Sommersemester 2015 – Blatt 2

Hasso-Plattner-Institut
Fachgebiet Computergrafische Systeme

Aufgabe 5: 2D-Metaballs (9 Punkte)

Schreiben Sie einen Plotter für implizite Funktionen zur Darstellung von 2D-Metaballs. Gegeben seien n Kreise. Eine 2D-Metaball-Funktion ist definiert als eine Menge von Dichtefunktionen

$$f_i(x, y) = \frac{Masse}{(x - x_i)^2 + (y - y_i)^2},$$

wobei (x_i, y_i) das Zentrum des i-ten Kreises angibt und (x, y) den zu untersuchenden Punkt. $f_i(x, y)$ gibt also die Dichte des Kreises am Punkt (x, y) zurück (Abstand zum Kreismittelpunkt). Wenn die Summe der Dichten aller Metaballs am Punkt (x, y) größer als ein Schwellenwert s ist, also

$$\sum_{i=0}^{n} f_i(x, y) > s,$$

wird die Fläche an dieser Stelle gefüllt. Weitere Erläuterungen zu Metaballs können Sie in (Blinn, 1982) nachlesen; das Paper finden Sie im Moodle.

5. a) Metaballs sollen interaktiv durch Mausklick auf dem Canvas erzeugt werden. Erweitern sie dafür die Methode `mouseReleaseEvent(QMouseEvent *event)` um entsprechende Zugriffe auf die Instanzvariable `m_metaballs`. (2 Punkte)

5. b) Implementieren Sie das Metaball-Rendering in der Methode `renderMetaballs()`. Berücksichtigen Sie dabei den Parameter `s`, welcher den Schwellwert für die Oberflächendefinition darstellt (Gewichtungen größer als `s` gehören zur gezeichneten Oberfläche). Für die Oberfläche soll die Farbe `innerColor` und für die Umgebung die Farbe `outerColor` benutzt werden. (4 Punkte)

5. c) Finden sie einen geeigneten Wert für den Schwellwertparameter `s`. Fügen sie zusätzlich einen weiteren Parameter `e` hinzu, welcher eine ϵ-Umgebung um `s` definieren soll, in der zwischen der Oberfläche und der Umgebung interpoliert wird. Definieren sie sich hierfür eine zusätzliche Farbe und interpolieren sie zwischen den drei Farben. (3 Punkte)

Abbildung 1: Ein Beispiel für 2D-Metaballs.

Aufgabe 6: Terrain Rendering (7 Punkte)

Diese Aufgabe befasst sich dem 3D-Rendering eines Dreiecksnetzes. Es stellt ein Modell einer Geländeoberfläche dar. Es soll verfeinert werden, um visuell mehr Details zu erhalten. Auch soll die Farbgebung des Dreiecksnetzes automatisch von der Geländehöhe abgeleitet werden. In dem Programmrahmen kann das Terrain mittels linker und rechter Maustaste verschoben und gedreht werden. Zoomen ist mit dem Mausrad möglich. Folgende Teilaufgaben sind zu bearbeiten:

6. a) Implementieren Sie die Methode `void tessellate(const Triangle &, std::vector< Triangle> &)` in `exercise6.cpp`, die für ein gegebenes Dreieck eine Verfeinerung berechnet, indem das Dreieck in vier kleinere Dreiecke zerlegt wird. Hierbei soll ausgehend vom ursprünglichen Dreieck eine Liste von Dreiecken generiert werden, die die Gesamtfläche des übergebenen Dreiecks vollständig ausfüllen, sich aber nicht überschneiden. Mit der Taste `<w>` kann zwischen Wireframe und Dreiecksrendering umgeschaltet werden, was die Überprüfung der Tessellation vereinfacht.

Nutzen Sie in der Methode `void render()` die Länge der Liste `m_currentBuffer` und ergänzen damit die Implementierung von `void render()` in `exercise6.cpp` um einen oder mehrere Aufrufe von `void drawTriangles(int start, int count)`. Der erste Parameter gibt dabei an, bei welchem Dreieck in der Liste angefangen werden soll zu rendern, und der zweite Parameter gibt an, wie viele Dreiecke zu rendern sind. (2 Punkte)

6. b) Benutzen Sie die Variablen `position` und `height` des Shaders `exercise6.vert`, um daraus die Weltkoordinaten `worldCoord` zu berechnen (rechtshändiges Koordinatensystem mit positiver y-Achse für die Höhe). (1 Punkt)

6. c) Passen Sie die Implementierung des Shaders `exercise6.vert` für die Zuweisung von `colorValue` so an, dass der Höhenwert benutzt wird. Passen Sie auch den Fragment-Shader `exercise6.frag` so an, dass die Variable `colorValue` als Interpolationswert zwischen den vier Farben `#FF348700`, `#F382C0A`, `#FF808080` und `#FFE9E9E9` (Die Keypoints sind 0.1, 0.333, 0.667 und 0.9) benutzt wird und das Ergebnis in `color` geschrieben wird. (3 Punkte)

6. d) Passen Sie die Shader `exercise6.vert` und `exercise6.frag` so an, dass für die Variablen `colorValue1`, `colorValue2` und `colorValue3` respektive keine Interpolation, perspektivische Interpolation, oder nichtperspektivische Interpolation (Interpolation in Fensterkoordinaten) benutzt werden. Mit den Tasten `<1>`, `<2>` und `<3>` kann zwischen den drei Interpolationsmodi gewechselt werden. (1 Punkt)

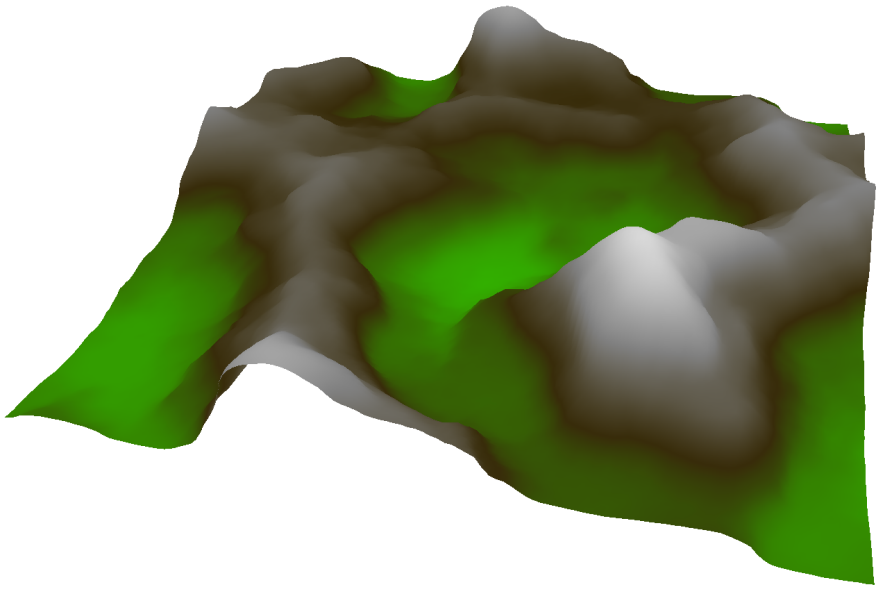

Abbildung 2: Beispiel einer Geländedarstellung.

Referenzen

Blinn, J. F. (1982): A Generalization of Algebraic Surface Drawing. ACM Transactions on Graphics (TOG), Volume 1 Issue 3.

Allgemeine Hinweise:

- Die Aufgaben sollen maximal zu zweit bearbeitet werden; Ausnahmen müssen mit den Übungsleitern abgesprochen werden.
- Bitte reichen Sie Ihre Lösungen bis Dienstag, den **12.05.2015**, um **13.15** Uhr ein.
- Tragen Sie ihre Matrikelnummern in die Quellcode-Dateien ein. Beachten Sie, dass nur die vollständigen Quelltexte, Projektdateien und Mediadateien geschickt werden sollen. Senden Sie uns keinesfalls ausführbare Dateien oder bereits kompilierte Binär- oder Temporärdateien (*.obj, *.pdb, *.ilk, *.ncb etc.)! Testen Sie vor dem Verschicken, ob die Projekte aus den Zipdateien fehlerfrei kompiliert und ausgeführt werden können.
- Zippen Sie ihre Lösungen für die Aufgaben 5 und 6 in **eine** Zip-Datei. Geben Sie der Zip-Datei einen Namen nach folgendem Schema:

  cg1_blatt2_matrikelnummer1_matrikelnummer2.zip.
- Die Zip-Datei laden Sie dann bei moodle hoch.