3D-Computergrafik I

Sommersemester 2015 – Aufgabenblatt 7

Hasso-Plattner-Institut
Fachgebiet Computergrafische Systeme

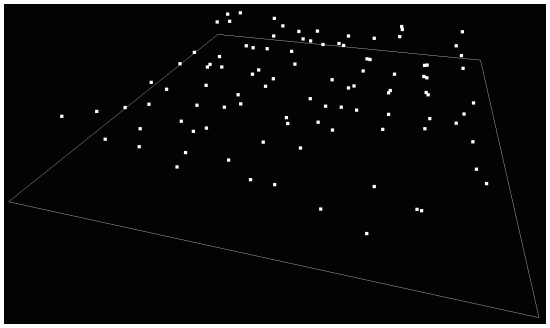
Aufgabe 21: Höhenfeld-Triangulierung und Bézierflächen (7 Punkte)

Eine häufig verwendete Repräsentationsform von Geländemodellen sind sogenannte Höhenfelder. Hierbei handelt es sich um zweidimensionale Skalarfelder, die ein Höhenrelief beschreiben. In dieser Aufgabe soll ein solches Höhenfeld auf verschiedene Arten visualisiert werden.

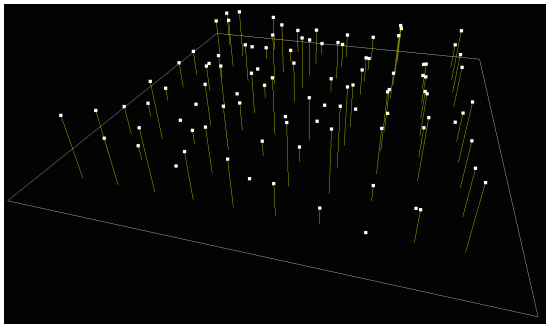
Das Höhenfeld ist als regelmäßiges quadratisches Raster (genauer: ein multidimensionales Array) `m_heightField` mit einer Seitenlänge von `SIZE` gegeben. Zur Visualisierung des Höhenfelds implementieren sie in der Klasse `Exercise21` die folgenden Methoden:

- `drawHeightFieldPoints()`. In dieser Methode sollen die einzelnen Höhenwerte in Form von GL_POINTS gerendert werden. Die Punktgröße soll auf einen von Ihnen gewählten sinnvollen Wert gesetzt werden (siehe Abbildung 1a).
- `drawHeightFieldLines()`. Visualisieren Sie das Höhenfeld mit Hilfe von GL_LINES, die senkrecht auf der xz-Ebene stehen und deren Länge dem jeweiligen Höhenwert entspricht (siehe Abbildung 1b).
- `drawTriangulatedHeightField()`. In dieser Methode soll das Höhenfeld trianguliert werden. Verwenden Sie $GL_TRIANGLE_STRIP$s als Renderingprimitive (siehe Abbildung 1c).

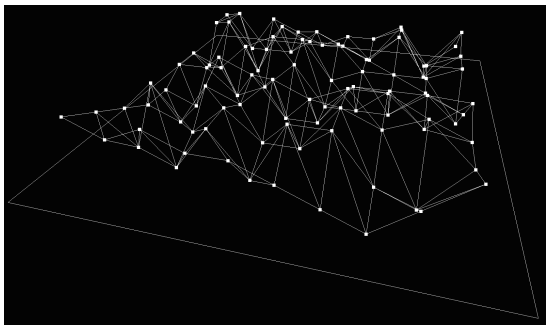
Abschließend sollen die Höhenwerte als Kontrollpunkte einer Bézierfläche interpretiert werden (siehe Abbildung 1d). Nutzen Sie zum Rendering der Bézierfläche einen zweidimensionalen OpenGL-Evaluator[1]. Initialisieren Sie diesen in der Methode `initialize()` unter Verwendung der Funktionen `glMap2f()` und `glMapGrid2f()`. Implementieren Sie zudem die Methode `drawHeightFieldBezierPatch()` um die Bézierfläche durch Verwendung der Funktion `glEvalMesh2()` zu rendern.



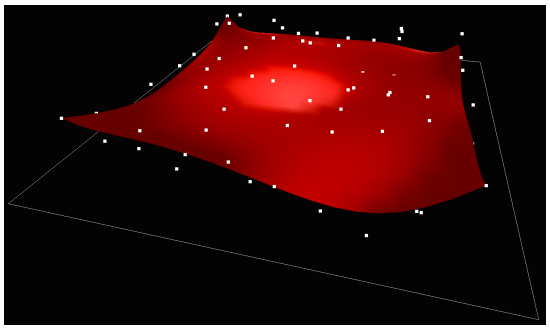
(a) Höhenwerte als Punkte



(b) Höhenwerte als Linien



(c) Trianguliertes Höhenfeld



(d) Bézierfläche

Abbildung 1: Verschiedene Visualisierungen eines Höhenfelds.

[1] http://www.glprogramming.com/red/chapter12.html

Aufgabe 22: Toon-Shader und Phong-Beleuchtungsmodell (8 Punkte)

In dieser Aufgabe sollen Sie unter Nutzung von Shaderprogrammen die folgenden zwei Renderingeffekte realisieren: (a) einen Cartoon-Effekt und (b) die Beleuchtung und Schattierung nach Phong (siehe Abbildung 2).

Cartoon-Effekt Vervollständigen Sie den Fragment-Shader `toon.frag` um die Implementierung eines Cartoon-Effekts ähnlich dem in der Vorlesung besprochenen *Cel-Shading*. Hierbei wird anhand der *Intensität* I eine Farbe aus einem diskreten Farbraum mit geringer Anzahl an Farben ausgewählt. Gehen Sie dabei wie folgt vor:

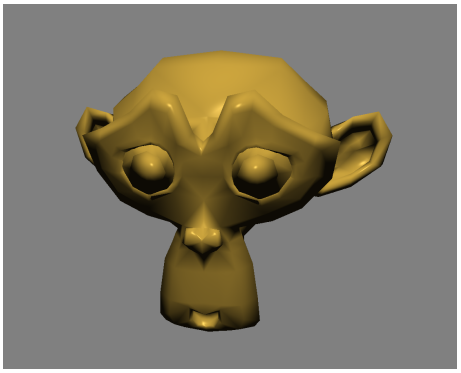
- Berechnen Sie die Intensität als Winkel zwischen der Normalen der Oberfläche und der Richtung der Lichtquelle.
- Diskretisieren Sie die im Shader gegebene Grundfarbe *baseColor* auf drei Intensitätsbereiche (zwischen $I >= 0,01$ und $I <= 0,99$).
- Integrieren Sie zusätzlich eine Farbgebung für die Intensitätsbereiche für dunkel schattierte Bereiche ($I < 0,01$), sowie einen Bereich für Highlights ($I > 0,99$) mit einer helleren Farbe.

Phong-Beleuchtungsmodell Bei dem Phong-Beleuchtungsmodell werden sowohl die Lichtparameter als auch die Materialeigenschaften anhand einer Gewichtungsfunktion (1) in die Berechnung der Farbwerte mit einbezogen. Die Parameter für das Licht werden dabei pro Lichtquelle durch einen *ambienten* (I_a), einen *diffusen* (I_d), sowie einen *spekularen* Anteil (I_s) simuliert. Für das Material werden analog die *ambienten* (k_a), *spekularen* (k_s) und *diffusen* (k_d) Parameter vergeben um verschiedene Materialeigenschaften simulieren zu können. Zusätzlich wird beim Material ein Koeffizient *shininess* (k_{shi}) für die Stärke der Glanzlichter festgelegt, durch den Oberflächen matter oder spiegelnder gestaltet werden können (Wertebereich 0 bis 128). Die Formel zur Berechnung der gesamten Illumination I ist gegeben durch

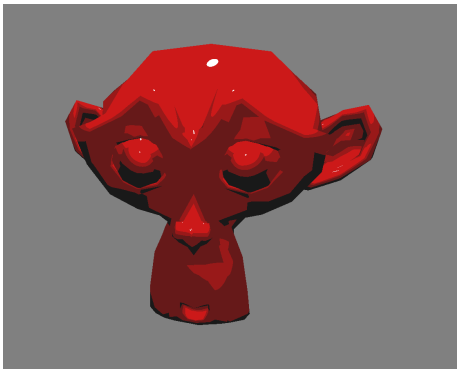
$$I = I_a k_a + I_d k_d \langle N, L \rangle + I_s k_s \langle R, V \rangle^{k_{shi}},$$

wobei N die Oberflächennormale, L die Lichtrichtung, R die Reflektionsrichtung und V die Kameraausrichtung sind.

Vervollständigen Sie den Fragment Shader `phong.frag` unter Verwendung der gegebenen *uniform*-Variablen so, dass eine korrekte Beleuchtungsberechnung nach Phong erfolgt. Vernachlässigen Sie hierbei die Abschwächung des Lichts (*attenuation*) sowie den Emissionskoeffizienten ($k_e m$).



(a) Phong-Beleuchtungsmodell



(b) Toon-Shader

Abbildung 2: Unterschiedliche Renderingeffekte am Beispiel eines 3D-Modells.

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 **21.07.2015**, um **13.15** Uhr ein.
- Tragen Sie Ihre Matrikelnummern in die Quellcode-Dateien ein. Beachten Sie, dass nur die vollständigen Quelltexte und Projektdateien geschickt werden sollen. Senden Sie uns keinesfalls ausführbare Dateien oder bereits kompilierte Binär- oder Temporärdateien (*.obj, *.pdb, *.ilk, *.ncb etc.) zu! Testen Sie vor dem Verschicken, ob die Projekte aus den Zip-Dateien fehlerfrei kompiliert und ausgeführt werden können.
- Zippen Sie Ihre Lösungen in **eine** Zip-Datei. Geben Sie der Zip-Datei einen Namen nach folgendem Schema:

  cg1_blatt7_matrikelnummer1_matrikelnummer2.zip.
- Die Zip-Datei laden Sie dann bei moodle hoch.