Institut für Informatik
Henning Wenke, M.Sc.
Nico Marniok, B.Sc.
Sascha Kolodzey, B.Sc.

Universität Osnabrück, 19.06.2012
http://www-lehre.inf.uos.de/~cg
Testat bis 27.06.2012, 15:00 Uhr

Übungen zu Computergrafik

Sommersemester 2012

Blatt 9

Hinweis:

- Die Quellen zur Aufgabenstellung für dieses Blatt werden erst am Mittwoch, den 20.06.2012 um ca. 16:00 Uhr veröffentlicht, da sie die Musterlösung zu Blatt 8 enthalten.
- Auch wenn ihr Programm nicht lauffähig sein sollte oder die gezeigte Szene nicht den Erwartungen entspricht, können Sie trotzdem eine hohe Punktzahl erreichen. Schon triviale Kleinigkeiten, die sie bei der Implementation falsch gemacht haben, können den Start des Programms verhindern. Das Debuggen von OpenGL erfodert sehr viel Einarbeitungszeit, Erfahrung und Praxis. Wenn ihr Tutor den Eindruck hat, dass Sie sich intensiv mit dem Stoff befasst haben, können sie daher auch leicht mit einem fehlerhaften Programm bestehen.
- In Zeile 89 der Datei SolSystem.java wir OpenCL initialisiert. Falls Ihr System OpenCL nicht unterstützt, können Sie die Zeile auskommentieren. Auf Blatt 10 wird das jedoch nicht mehr möglich sein und sie müssen in 31/145 arbeiten.

Aufgabe 9.1 (10 Punkte)

Machen Sie sich mit der erweiterten Funktionalität der Klasse `Geometry` vertraut. Das Vertexlayout kann nun dynamisch festgelegt werden. Dazu dienen die Methoden `addVertexAttribute(...)` und `clearVertexAttributes()`. Die erste Methode muss für jedes Attribut aufgerufen werden. Die Reihenfolge ist dabei entscheidend. Beispielsweise erzeugen folgende Aufrufe das Layout von Aufgabe 8.1.

- `geo.addVertexAttribute(Util.ATTR_POS, 3, 0);`
- `geo.addVertexAttribute(Util.ATTR_NORMAL, 3, 12);`
- `geo.addVertexAttribute(Util.ATTR_COLOR, 4, 24);`
- `geo.addVertexAttribute(Util.ATTR_COLOR2, 4, 40);`

Beachten Sie, dass insbesondere das Attribut `stride` der Methode `glVertexAttribPointer(...)` automatisch berechnet wird.

Erweitern Sie das Programm um das Vertexattribut `ATTR_TEX`, welches an die in Variable `texCoords` des Vertexshaders gebunden wird.

Implementieren Sie die Methode `createSphere(float r, int n, int k)` der Klasse `GeometryFactory`, sodass sie Vertices des folgenden Layouts erzeugt.

$$\{p_x, p_y, p_z, n_x, n_y, n_z, t_s, t_t\}$$

Hierbei sind $\begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}$ die Koordinaten, $\begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix}$ die Normale und $\begin{pmatrix} t_s \\ t_t \end{pmatrix}$ die Texturkoordinaten des Vertex.

`/home/cg/2012/Uebung/Blatt9/Lsg/CG12Blatt9.zip`

Aufgabe 9.2 (20 Punkte)

Erweitern Sie das Programm um die Möglichkeit, Texturen zu laden. Implementieren Sie dazu die Methode `generateTexture(String filename)` in `SolSystem.java`. Benutzen Sie die Methode loadImage(...) der Klasse Util, um die rohen Pixeldaten und Bildinformationen zu gelangen. Generieren Sie dann mithilfe der OpenGL Methoden `glGenTextures()`, `glBindTexture(...)` und `glTexImage2D(...)` die Textur und liefern Sie die generierte ID zurück.

Implementieren Sie dann die Hilfsmethode `texture2Uniform(int texture, int target, int slot, int location)` der Klasse `SolSystem.java`, die eine Textur an eine bestimmte Uniform Variable bindet. Benutzen Sie dazu die OpenGL Methoden `glActiveTexture(...)`, `glBindTexture(...)` und `glUniform1i(...)`.

`/home/cg/2012/Uebung/Blatt9/Lsg/CG12Blatt9.zip`

Aufgabe 9.3 (30 Punkte)

Erzeugen Sie den Vertexshader `FragmentLighting_VS.glsl` mit folgenden Spezifikationen.

- Eingang: `positionMC` vom Typ `vec3`, `normalMC` vom Typ `vec3`, `texCoords` vom Typ `vec2`.
- Ausgang: `positionWC` vom Typ `vec3`, `normalWC` vom Typ `vec3`, `fragmentTexCoords` vom Typ `vec2`.
- Uniforms: `viewProj` vom Typ `mat4`, `model` vom Typ `mat4`, `modelIT` vom Typ `mat4`.
- Die Position (positionWC) und die Normale (normalWC) in Weltkoordinaten sollen berechnet und die Texturkoordinaten durchgereicht werden.

Laden Sie den Fragmentshader `FragmentLighting_FS.glsl` von der CG-Seite und machen Sie sich mit seinem Aufbau vertraut. Implementieren Sie die folgenden Methoden.

- In `plIntensity(vec3 p, vec3 p_p, vec3 I_p_max)` soll die Intensität einer Punktlichtquelle auf einen übergebenen Punkt berechnet werden.

- In `calcLighting(vec3 pos, vec3 normal, vec3 c_d, vec3 c_s)` soll ein Punkt komplett beleuchtet werden. Dabei sollen alle Punktlichtquellen, die in dem Array `plPosition` und `plMaxIntensity` stehen, mit in die Beleuchtung eingehen. Verwenden Sie dazu die Uniform Variablen des Shaders und die im Skript vorgestellten Formeln.
- In `main(void)` soll zunächst die diffuse, sowie die spekulare Materialfarbe bestimmt und anschließend mittels `calcLighting(...)` das aktuelle Fragment beleuchtet werden.

`/home/cg/2012/Uebung/Blatt9/Lsg/CG12Blatt9.zip`

Aufgabe 9.4 (30 Punkte)

- Haben Sie die vorangegangen Aufgaben gelst, sollte das Programm nun ohne Probleme starten. Machen Sie Ihren Tutor anhand der gezeigten Szene und einer eigenen Skizze auf die Vor- und Nachteile der gewhlten Beleuchtungsmethode aufmerksam.
- Implementieren Sie die Methode `createSphereWithDisplacement(float r, int n, int k, float displaceHeight, String displaceFile)`. Diese soll die Vertices der Erde entlang ihrer Normalen um den Rotwert der im Parameter `displaceFile` angegebenen Textur verschieben. Steht zu einem Vertex $v_{\varphi,\theta}$ in der Textur ein Rotwert von $r_{\varphi,\theta}$, so soll die Gesamtverschiebungsdistanz $d_{\varphi,\theta} = \text{displaceHeight} \cdot r_{\varphi,\theta}$ betragen.
- Da sie die Vertices nun verchoben haben, stimmen die Normalen natrlich nicht mehr mit der zugrunde liegenden Geometrie berein. berlegen Sie sich ein Verfahren, die Normalen an die neue Geometrie anzupassen und implementieren Sie es. Erklren SIe Ihrem Tutor die dabei die Unterschiede im Vergleich zu nicht vernderten Normalen.
- ndern Sie die Klasse `SolSystem` derart ab, dass die Erde mithilfe der neuen Methode erstellt wird. Als Displacementtextur soll dabei `earth_height.jpg` verwendet werden.

`/home/cg/2012/Uebung/Blatt9/Lsg/CG12Blatt9.zip`

Aufgabe 9.5 (10 Punkte)

Beantworten Sie Ihrem Tutor Fragen zur Vorlesung.