Prof. G. Zachmann
P. Meißenhelter
R. Fischer
A. Mühlenbrock

Universität Bremen
Fachbereich 3
Arbeitsgruppe CGVR
16. November 2021

Wintersemester 2021/22

Übungen zu Computergraphik - Blatt 4

Abgabe am 22.11.2021, 23:59 Uhr

In diesem Aufgabenblatt beschäftigen wir uns noch einmal mit der Programmierung eines Vertex- und Fragment-Shaders. Grundlage hierfür ist das HeightMapFramework, welches auf der CGVR-Seite zum Download bereitsteht. Der Programmcode für C++ ist bereits vollständig vorgegeben, d.h. es müssen ausschließlich die Shader angepasst werden.

Aufgabe 1 (Heightmap, *3 + 2 + 1 + 1 + 2 + 1 Punkte*)

Der vorgegebene Vertex-Shader führt bisher lediglich Transformationen durch, damit das Gitter aus der gewünschten Perspektive dargestellt wird, die mittels Mausbewegung verändert werden kann. Doch bevor diese Transformationen stattfinden, können wir weitere Anpassungen an der 3D-Koordinate vornehmen. Das kann zum Beispiel sinnvoll sein, wenn man eine Heightmap als dreidimensionales Mesh darstellen möchte.

Heightmaps werden häufig in normale Bilddateien (z.B. im PNG-Format) gespeichert und enthalten in ihren Farbwerten Höhenwerte. Um mithilfe einer Heightmap ein dreidimensionales Mesh zu erzeugen, können die Vertices eines 2D-Gitters in ihrer Höhe um den entsprechenden Pixelwert verschoben werden, wie in der nachfolgenden Abbildung sichtbar:

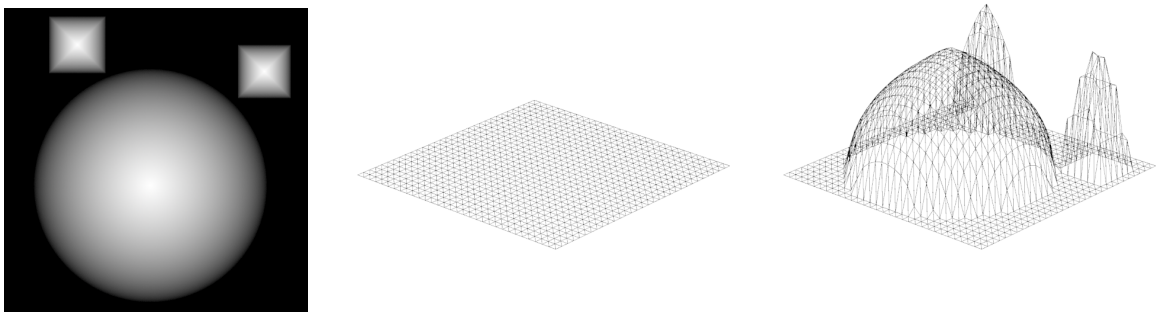

Abbildung 1: Links: Heightmap. Mitte: 2D-Gitter (Wireframe-Ansicht). Rechts: 2D-Gitter, dessen Vertices mithilfe der Heightmap entlang der Höhe verschoben wurden.

Im C++ Programmcode wird bereits das Mesh eines 2D-Gitter, welches aus 512∗512 Vertices besteht, generiert und in einem VertexBuffer der Grafikkarte geladen. Zudem wird eine PNG mit dem Namen `heightmap.png` als Textur geladen. Um auf die Werte eines Pixels der Textur zuzugreifen, kann die GLSL-Funktion `texture(...)` verwendet werden. Diese ist sowohl im Vertex- als auch im Fragment-Shader verfügbar. Sie erwartet als ersten Parameter die Textur, welche über die Uniform-Variable `heightMapTexture` bereits vorgegeben ist. Als zweiten Parameter erwartet sie die normalisierten Koordinaten eines Pixels, die auch UV-Koordinaten genannt werden. Diese müssen im Typ `vec2`

vorliegen. Mithilfe der folgenden Zeile erhält man so zum Beispiel die RGBA-Farbwerte des Pixels in der linken unteren Ecke:

```
vec4 color = texture(heightMapTexture, vec2(0.0, 0.0));
```

Für die obere rechte Ecke kann man folgende Zeile verwenden:

```
vec4 color = texture(heightMapTexture, vec2(1.0, 1.0));
```

Die Aufgabe lautet wie folgt:

a) Erzeuge sinnvolle UV-Koordinaten im Vertex-Shader und übergebe diese an den Fragment-Shader (hierzu kannst Du annehmen, dass das Gitter entlang der x- und z-Achse zwischen 0 und 1 aufgespannt ist. Die Position des jeweiligen Vertex ist in der Variable `position` gespeichert). Setze die Ausgabe-Variable `color` dann auf den Farbwert der Heightmap-Textur an der Stelle der UV-Koordinate, sodass die Heightmap in Form eines Bildes auf das bereits vorhandene Gitter sichtbar wird. Beachte, dass eine Heightmap i.d.R. nur einen Farbkanal benötigt. Trotz allem soll die gezeigte Textur in grau (und nicht in rot) erscheinen.

b) Verschiebe nun im Vertex-Shader jedes Vertex in Abhängigkeit von der Höhe, die in der Heightmap an der Stelle der UV-Koordinate gespeichert ist. Verwende lediglich den roten Farbkanal.

c) Skaliere die Höhe mit der Uniform-Variable `height`. Diese Uniform-Variable kann in der GUI über den Schieberegler verändert werden.

d) Verschiebe die Vertices, sodass das 3D-Objekt entlang der x- und z-Achse zentriert gezeigt wird.

e) Modifiziere den Vertex-Shader, sodass über die Zeit abwechselnd zwischen einer flachen Ebene und der 3D-Geometrie hin und her animiert wird (zwischen einer Höhe von 0 und der maximal eingestellten Höhe aus (c)). Verwende dazu die Sinus-Funktion. Das ganze Sinusintervall (2π) soll genau 3 Sekunden dauern. Der Multiplikator der Höhe sollte sich zwischen 0 und 1 hin und zurück bewegen. *Hinweis: Im Vertex-Shader kann Du bereits auf die Uniform-Variable time zugreifen, die die Laufzeit des Programms in Sekunden angibt.*

f) Mache jeweils ein Screenshot der Anwendung für drei unterschiedliche Heightmaps und füge diese in die Abgabe-PDF mit ein. Erstelle diese Heightmaps per Hand (z.B. in Paint oder GIMP). Bitte auch die originale Heightmap mit in die Abgabe-PDF einfügen. *Hinweis: Es können lediglich PNGs verwendet werden, die 8 Bit per Farbkanal besitzen und entweder 1,3 oder 4 Farbkanäle enthalten - andere PNGs werden möglicherweise falsch geladen. Um eine andere Heightmap in die Anwendung zu laden, ersetze jeweils die* `heightmap.png` *und starte es erneut.*