Institut für Visualisierung
und Interaktive Systeme

Dr. Guido Reina Dr. Michael Krone

12. Übungsblatt zur Vorlesung Computergraphik im WS 2017/18

Abgabe am Mittwoch, 31.01.2018 (12:00 Uhr)
Besprechung am Montag, 05.02.2018 (17:30 - 19:00 Uhr)

Hinweis: Unterstützt werden Visual Studio 2015 und 2017. Bei der Installation müssen Sie unter „Einzelne Komponenten“ den „NuGet-Paket-Manager“ aktivieren. Sollten Sie Visual Studio 2017 verwenden, brauchen Sie zusätzlich das „Toolset für VC++ 2015.3 v140“. Diese Features lassen sich auch im Nachhinein mit dem Visual Studio Installer hinzufügen. Achten Sie bitte auch darauf, beim ersten Öffnen des Projekts, die verwendete Toolset-Version *nicht* auf v141 umzustellen (kein Upgrade).

Das Programmskelett greift für die Verwendung von OpenGL auf externe Open-Source-Bibliotheken zu. In der mitgelieferten Visual-Studio-Solution sind die Bibliotheken bereits eingebunden und werden (sofern der NuGet-Paket-Manager installiert ist) automatisch nachgeladen. Falls Sie einen anderen Compiler verwenden wollen, müssen Sie sich selbst um die Einbindung folgender Bibliotheken kümmern:

- glfw: http://www.glfw.org/
- glad: https://github.com/Dav1dde/glad
- glm: https://glm.g-truc.net

Über ILIAS erhalten Sie das Visual-Studio-Projekt MipMapping. Wenn Sie dieses öffnen und ausführen sehen Sie das Ergebnis des Renderers aus dem letzten Übungsblatt. Um die Ladezeit zu verkürzen, sollten Sie zudem das Projekt im *Release*-Modus kompilieren.

Mit den Tasten W, A, S, D, *Shift* und *Space* kann man sich im Raum bewegen. Die Pfeiltasten erlauben eine Änderung der Blickrichtung und mit den Tasten 1 und 2 lässt sich die Richtung des Sonnenlichts ändern.

Aufgabe 1 *Mip-Mapping [6 Punkte]*

In dieser Aufgabe soll das Terrain texturiert und die Verwendung von Mip Mapping demonstriert werden. Bei Mip-Mapping steht eine Texturpyramide zur Verfügung, die die gleiche Textur in unterschiedlichen Auflösungen enthält (siehe Vorlesung Kapitel 6, Folie 50). Beim Zeichnen eines Polygons wird dann abhängig von der Entfernung zum Betrachter automatisch eine passende Auflösung der Textur gewählt.

1. Welche Probleme würden sich ergeben, wenn das ganze Terrain mit einer Textur der gleichen Auflösung gezeichnet wird?

2. Vervollständigen Sie die Funktion `UpdateGroundTexture` in `main.cpp`, in der die Textur für das Terrain initialisiert wird. Das richtige OpenGL-Textur-Handle ist bereits gebunden.

   - In `g_ground_textures` befinden sich die Texturen, die für die ersten Mip-Map Level verwendet werden sollen. Dabei entspricht der Index im Array dem Mip-Map Level. (D.h. die erste Textur ist die hochauflösendste mit 256x256 Pixeln, die nächste hat in beiden Richtungen die halbe Auflösung und so weiter.) Das Datenformat der Texturdaten ist vier `float`s.
   - Nutzen Sie anschließend die Funktion `glGenerateMipmap` um die noch fehlenden Mip-Map Level zu generieren. Achten Sie dabei darauf, dass diese aus der am niedrigsten aufgelösten Textur, d.h. aus dem höchsten bereits initialisierten Mip-Map Level, generiert werden.

   Hinweis: `glGenerateMipmap` verwendet das aktuell gesetzte Base-Level als Quelle und generiert alle höheren Mip-Map Level. Um das höchste bereits initialisierte Mip-Map Level als Quelle zu verwenden, müssen Sie zuerst das Base Level mithilfe der Funktion `glTexParameteri` auf das entsprechende Level setzen. Vergessen Sie nicht, das Base Level danach wieder auf 0 zurückzusetzen.

   - Nutzen Sie die Funktion `glTexParameteri` um folgende Effekte zu erreichen: Als Minification Filter soll der übergebene Parameter `min_filter` verwendet werden, der Magnification Filter soll linear sein und beim Sampling außerhalb des Bereichs $[0, 1]^2$ soll der Texturinhalt wiederholt werden.

3. Vervollständigen Sie die gekennzeichnete Stelle in `final_frag.glsl`, sodass die Textur auch tatsächlich verwendet wird. Verwenden Sie die mit dem Fragment übergebenen Texturkoordinaten, um auf die Textur `tex_ground` zuzugreifen und die Farbe auszulesen. Die Farbe befindet sich in den ersten drei Komponenten und soll in der lokalen Variable `color` abgelegt werden.

Nach der Lösung von Teilaufgabe 3 sollte das Ergebnis wie in Abbildung 1 aussehen. Sie können die Tasten 3 bis 8 nutzen, um zwischen unterschiedlichen Minification Filtern umzuschalten:

3	`GL_NEAREST` (Gesetzt beim Programmstart)
4	`GL_LINEAR`
5	`GL_NEAREST_MIPMAP_NEAREST`
6	`GL_LINEAR_MIPMAP_NEAREST`
7	`GL_NEAREST_MIPMAP_LINEAR`
8	`GL_LINEAR_MIPMAP_LINEAR`

Nach Betätigen der Taste 8 sollten die Ausgabe wie in Abbildung 2 aussehen. Zur besseren Veranschaulichung, welche Textur wo verwendet wird, verwenden wir Texturen mit unterschiedlichen Farben (wobei Rot am höchsten aufgelöst ist). In der Praxis würde man natürlich die gleiche Textur in unterschiedlichen Auflösungen verwenden.

4. Erklären Sie in eigenen Worten die Unterschiede zwischen den verschiedenen Minification Filtern.

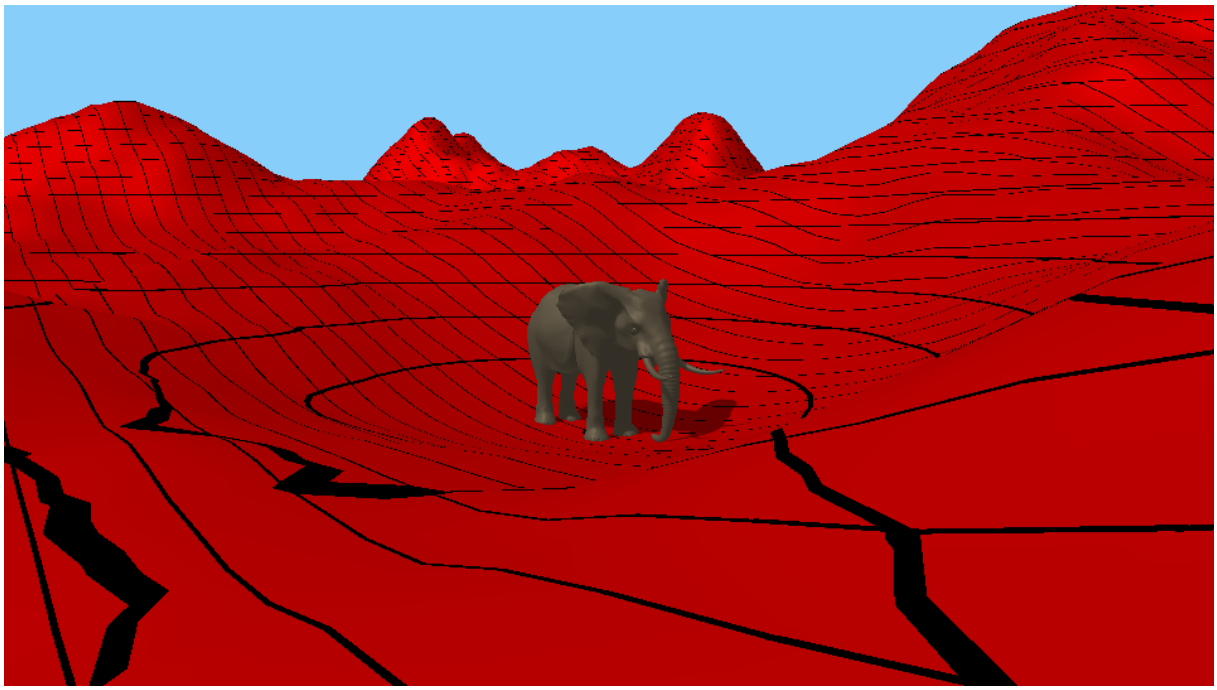

Abbildung 1: Ausgabe mit Minification Filter `GL_NEAREST`

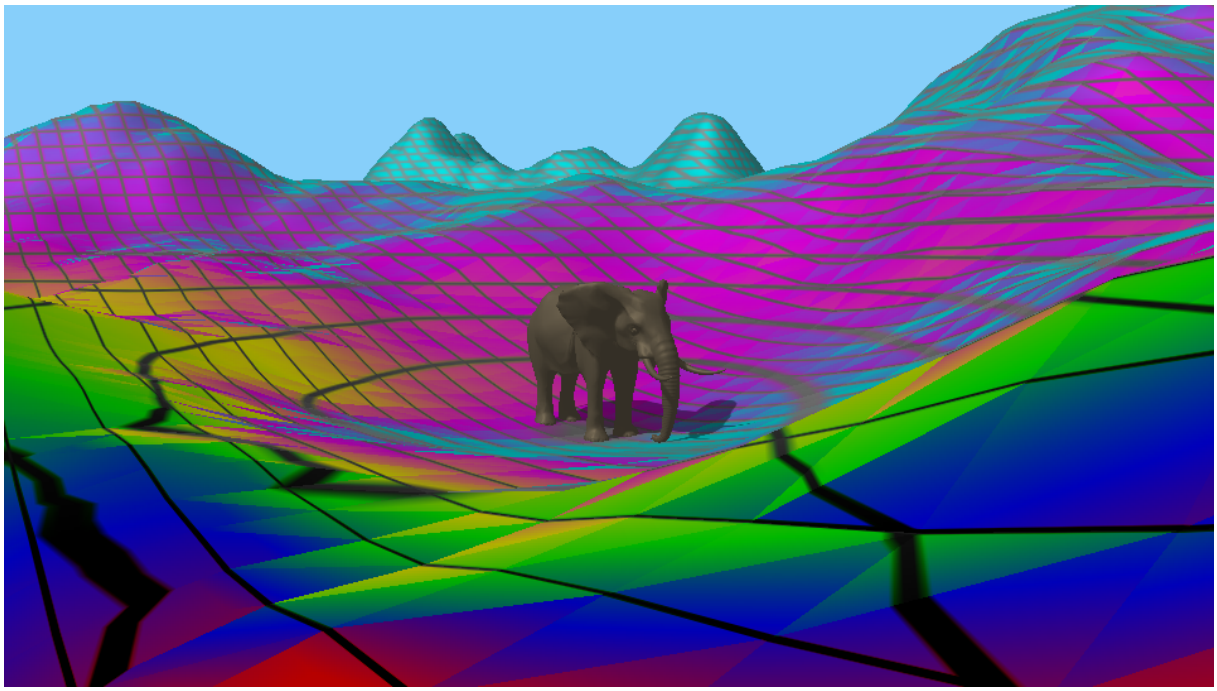

Abbildung 2: Ausgabe mit Minification Filter `GL_LINEAR_MIPMAP_LINEAR`

Aufgabe 2 *Tri-Planar Texture Mapping [4 Punkte]*

In Aufgabe 1 wird ein planares Mapping mit Projektion der Textur entlang der Z-Achse verwendet, d.h. als Texturkoordinaten wird die XY-Position der Terrain Oberfläche in Weltkoordinaten verwendet. Weicht die Normale der Oberfläche von der Projektionsachse ab, kommt es zu Verzerrungen der Textur. In der vorherigen Aufgabe wird dieser Verzerrungseffekt verwendet um Höhenlinien auf der Landschaft darzustellen. Wir wollen jetzt allerdings ein primitives, prozedurales Texture Mapping verwenden, das unterschiedliche Oberflächenmaterialien in Abhängigkeit der Höhe der Landschaft verwendet. Dazu verwenden wir nahtlos kachelbare (*engl.* tiling) Texturen von unterschiedlichen Oberflächen (Sand, Gras, Gestein, etc.). Eine übliche Technik um mit solchen Texturen eine Landschaft verzerrungsfrei zu texturieren, ist Tri-Planar Texture Mapping. Bei dieser Technik werden pro Oberflächenpunkt Samples aus drei orthogonalen Projektionsrichtungen aus den Texturen gelesen und gewichtet aufsummiert. Die Gewichtung erfolgt danach, wie groß die Verzerrung der jeweiligen Projektionsrichtung des Samples auf dem betrachteten Oberflächenpunkt ist.

Hinweis: Unter folgende Links finden Sie weitere Erklärungen zu Tri-Planar Texture Mapping:

- Use Tri-Planar Texture Mapping for Better Terrain
- Generating Complex Procedural Terrains Using the GPU (Kapitel 1.5)

1. Vervollständigen Sie die Funktion `UpdateGroundMaterialTexture` in `main.cpp`, in der die unterschiedlichen Material-Texturen für das Terrain initialisiert werden. Sie müssen hierbei nur ein Mipmap Level der Textur laden, die restlichen generieren Sie mit `glGenerateMipmap`. Sobald die Texturen korrekt initialisiert sind, können Sie sie mit der Taste 9 auf der Landschaft anzeigen. Die unterschiedlichen Oberflächenmaterialien werden bereits nach der Höhe der Landschaft ausgewählt und überblendet. Bisher wird jedoch ein planares Mapping mit Projektion entlang der Y-Achse verwendet. An steileren Hängen in der Landschaft können Sie entsprechend beobachten wie die Textur in der Höhe verzerrt wird.

2. Implementieren Sie in der Funktion `triplanarTextureMapping` in `ground_material_frag.glsl` Tri-Planar Texture Mapping. In `uvw_coords` wird die Position des Fragments in (skalierten) Weltkoordinaten übergeben. Kombinieren Sie jeweils die passenden Einträge von `uvw_coords` zu Texturkoordinaten, mit denen Sie die übergebenen Texturen `texture_y` und `texture_xz` mit planarem Mapping entlang der Hauptachsen auslesen. Verwenden Sie `texture_xz` für die Projektion entlang der X- und Z-Achse, und `texture_y` für die Projektion entlang der Y-Achse. Anschließend müssen Sie die Gewichtungen für die drei Farbwerte aus der übergebenen Oberflächen-Normalen berechnen und die gewichtete Summe zurückgeben. Folgen Sie dazu den Anweisungen in den Kommentaren.

Abbildung 3: Ausgabe mit Texturen aus Aufgabe 2.