Institut für Visualisierung
und Interaktive Systeme

Dr. Guido Reina Dr. Michael Krone

9. Übungsblatt zur Vorlesung Computergraphik im WS 2017/18

Abgabe am Mittwoch, 10.01.2018 (12:00 Uhr)
Besprechung am Montag, 15.01.2018 (17:30 - 19:00 Uhr)

Hinweis: *Sie können in diesem Aufgabenblatt 2 Bonuspunkte bekommen, d.h. 10 der maximal 12 zu erreichenden Punkte entsprechen wie bisher bereits der vollen Punktezahl.*

Aufgabe 1 *Transformationen, Instancing und Billboards in OpenGL [7 Punkte]*

In dieser Aufgabe arbeiten Sie mit den in der Rasterisierung üblichen Transformationen. Über ILIAS erhalten Sie das Visual-Studio-Projekt Geometrie, in dem die zu bearbeitenden Stellen mit `TODO` gekennzeichnet sind.

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
- lodepng: http://lodev.org/lodepng/

Wenn Sie das Projekt öffnen und ausführen, öffnet sich ein Fenster, in dem aber noch nichts gezeichnet wird.

1. In `main.cpp` finden Sie die Funktion `UpdateScene`. Setzen Sie darin die *View*- und die *Projection*-Matrix.

   Dabei ist `g_cam_pos` die Position der Kamera und `cam_dir` die Blickrichtung, wobei die y-Achse nach oben zeigen soll. Für die Projektion soll das richtige Seitenverhältnis zwischen der Fensterbreite `g_window_width` und der Fensterhöhe `g_window_height` verwendet werden. Wählen Sie außerdem passende Werte für das *Field of View* sowie die Entfernungen zur *Near*- und *Far-Plane*.

   Hinweis: Anstatt die Matrizen von Hand zu berechnen, empfiehlt es sich, die entsprechenden Funktionen der `glm`-Bibliothek zu verwenden.

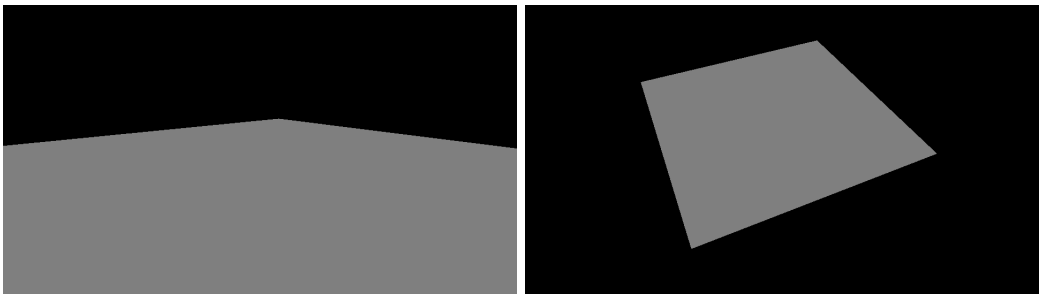

Abbildung 1: Ausgabe nach korrekter Lösung von Teilaufgabe 1.

Mit den Tasten *W, A, S, D, Shift* und *Space* kann man sich im Raum bewegen. Die Pfeiltasten erlauben eine Änderung der Blickrichtung.

2. In der Funktion `InitScene` befindet sich der Code, der die Geometrie für den Boden erstellt. Fügen Sie Code hinzu, der analog die Geometrie eines zweidimensionalen Baumes in der xy-Ebene erstellt und die zugehörige globale Variable initialisiert (vgl. Abbildung 2).

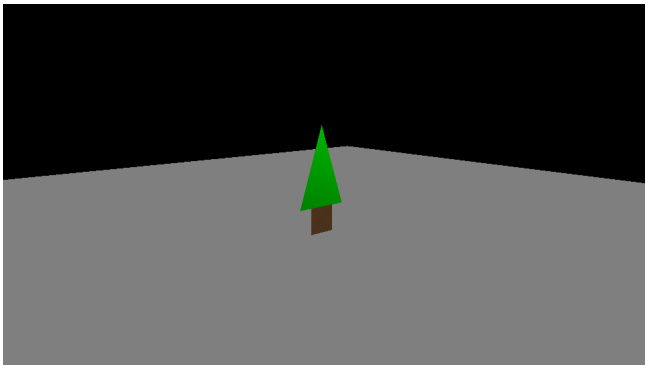

Abbildung 2: Ausgabe nach korrekter Lösung von Teilaufgabe 2.

3. Tatsächlich werden vom Hauptprogramm ganze 192 Instanzen Ihres Baumes gezeichnet. Dabei wird der Vertex-Shader `vert_trees.glsl` verwendet. An diesen wird für jeden der Bäume eine eigene *Model*-Matrix übergeben. Da diese nicht verwendet werden, erscheinen alle Bäume an der selben Stelle. Modifizieren Sie den Shader, sodass die *Model*-Matrizen korrekt verwendet werden (vgl. Abbildung 3).

   Hinweis: Mit `gl_InstanceID` können Sie im Vertex-Shader auf den Index der Instanz des Baums zugreifen. Dieser hat einen ganzzahligen Wert zwischen 0 und 191.

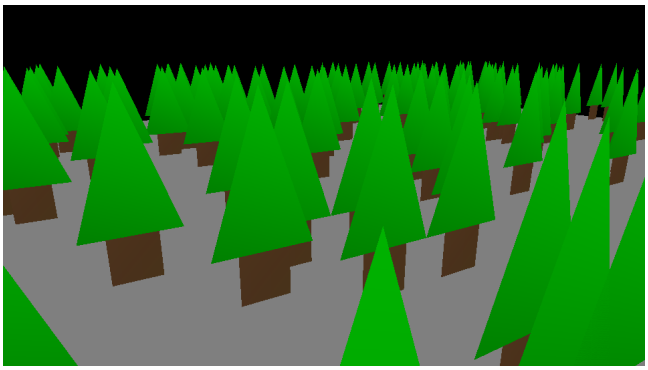

Abbildung 3: Ausgabe nach korrekter Lösung von Teilaufgabe 3.

4. Implementieren Sie ebenfalls in `vert_trees.glsl` den sogenannten *Billboard*-Effekt für die Bäume. Bei diesem Effekt drehen sich 2D-Objekte immer zum Betrachter, egal aus welcher Richtung dieser blickt. Modifizieren Sie dazu die Rotationskomponente der `model_view`-Matrix, sodass die x-Achse bei dieser Transformation unverändert bleibt.

   Hinweis: Da der Baum in der xy-Ebene liegt (mit $z = 0$), kann die Transformation der z-Achse ignoriert werden.

Aufgabe 2 *Texturen und Imposter [5 Punkte]*

Ein möglicher Einsatzzweck für Bildboards ist die Darstellung entfernter Objekte, für die sich eine detaillierte Geometrie weder lohnt noch im Polygon-Budget der Szene Platz findet. Da die Blickrichtung zu ausreichend weit entfernten Objekt zudem auch bei Bewegung nur begrenzt variiert, fallen Billboards oftmals kaum auf. Die bisher verwendete Geometrie der Bäume in der Szene ist allerdings auch auf weite Distanz nur eingeschränkt glaubwürdig. Deswegen sollen nun die Bäume als texturierte Quads (sog. Imposter) gerendert werden. Dabei soll der Alpha-Kanal der Textur verwendet werden, um unnötige Fragmente des Quads zu verwerfen (vgl. Vorlesungsfolien OpenGL S. 142).

1. Erzeugen Sie in der Funktion `InitScene` analog zu Aufgabe 1.2 die Geometrie für ein Quad aus zwei Dreiecken. Das Quad soll im Bereich $[-1.5,1.5] \times [0,3] \times [0,0]$ liegen, mit Texturkoordinaten im Bereich $[0,1] \times [1,0]$. Verwenden Sie die ersten beiden Einträge der Vertex-Color um die Texturkoordinaten zu speichern.

2. Kommentieren Sie in der Funktion `RenderFrame` das Rendern der Bäume aus Aufgabe 1 aus und kopieren Sie ihre Lösung von Aufgabe 1.3 und 1.4 in den Vertex-Shader `vert_tree_imposters.glsl`. Anschließend sollten Sie die Quad-Imposter mit dem farblichen Verlauf der Texturkoordianten sehen.

3. In der Funktion `CreateSpriteTexture` wird bereits eine Bilddatei geladen und die notwendigen Informationen zur Verwendung als Textur angegeben. Vervollständigen Sie die Funktion mit der Erzeugung eines OpenGL Textur-Objektes in das die Bilddaten geladen werden. Die Kommentare im Code liefern Ihnen die Details zu den notwendigen OpenGL API-Calls. Schauen Sie sich außerdem in der Funktion `RenderFrame` an, wie die Textur bereits an den Shader weitergeben wird, um einen vollständiges Eindruck der notwendigen API-Calls für Texturen zu erhalten.

4. Um die Textur auf den Imposter-Quads anzuzeigen, müssen Sie nun noch im Fragment-Shader `frag_tree_imposters.glsl` die Textur mit den übergegebenen Texturkoordiante

(`in vec2 uv;`) auslesen und als Ausgabefarbe verwenden. Wenn der Alpha-Kanal der Textur unterhalb eines gewissen Grenzwertes liegt, verwerfen Sie das Fragment mit dem `discard` Befehl.

5. Bisher haben alle Instanzen des Baums die gleiche Größe. Berechnen Sie im Vertex-Shader `vert_tree_imposters.glsl` einen zufälligen Skalierungsfaktor, mit dem die Position der Eingabe-Vertices multipliziert wird.

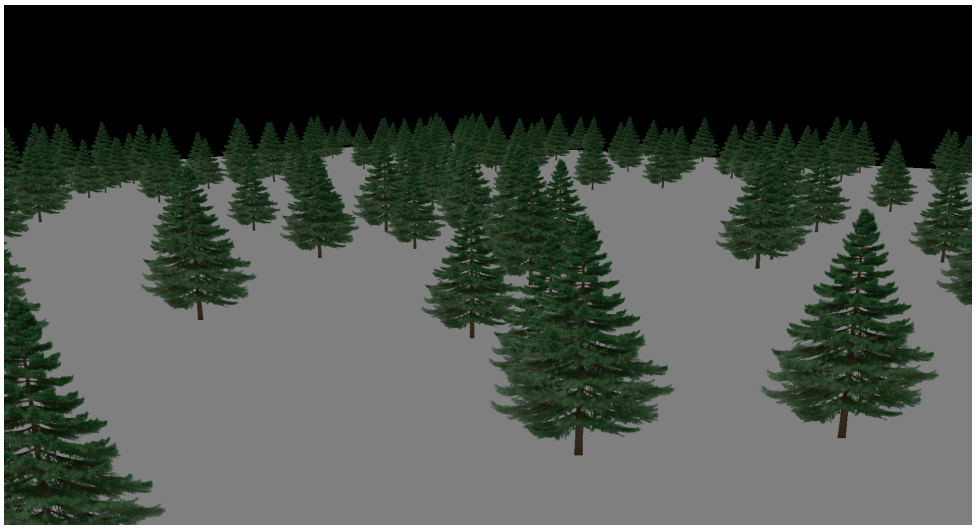

Abbildung 4: Ausgabe nach korrekter Lösung von Aufgabe 2.5.