CG1 WS14/15 - Übungsblatt 4

Technische Universität Berlin - Computer Graphics
Ausgabe 19. Dezember 2014, **Abgabe** 23. Januar 2015
Prof. Dr. Marc Alexa, Katrin Lang, Xi Wang.

AUFGABE 1 : Texturierung

Auf der Webseite des Kurses steht Ihnen ein `C++` Programmskelett zur Verfügung, das zu ergänzen ist:

1. Erweitern Sie die dem Skelettcode beiliegende Klasse `Image` um die Generierung von 2D-Texturen für OpenGL. Geladene Texturen sollen im linken Unterfenster als Bild angezeigt werden. Definieren Sie hierfür ein sog. "Full Screen Quad" als `.OFF`-Modell (Zum Laden und Anzeigen kopieren Sie bitte die nötigen Funktionen aus dem letzten Aufgabenblatt in den neuen Skelettcode). Berechnen Sie in einem Vertex-Shader aus den Vertex-Positionen Texturkoordinaten, und texturieren Sie das Modell im Fragment-Shader. Auf der rechten Seite soll die Textur in perspektivischer Ansicht auf das Quad aufgebracht werden, das der Benutzer jedoch beliebig rotieren, skalieren, und translieren kann. Als Option soll der Benutzer zwischen den Interpolationsmethoden `GL_NEAREST`, `GL_LINEAR`, und, für `GL_TEXTURE_MIN_FILTER`, `GL_NEAREST_MIPMAP_NEAREST`, `GL_LINEAR_MIPMAP_NEAREST`, `GL_NEAREST_MIPMAP_LINEAR`, sowie `GL_LINEAR_MIPMAP_LINEAR` wählen können. (1 Punkt)

2. Erweitern Sie Ihren `OFF`-Loader aus Übungsblatt 3 um die Berechnung sphärischer Texturkoordinaten (und ein weiters Vertex Array). Visualisieren Sie texturierte Modelle wiederum unter Verwendung eines Shaders im rechten Unterfenster. Beleuchtung und Texturierung sollen zu- und abwählbar sein. Dazu verwenden Sie bitte die in Aufgabenblatt 3 implementierte Funktion `blinnPhongReflection`. Stellen Sie sicher, dass ihr Algorithmus für die Texturen aus der Rubrik "`SPHERICAL TEXTURES`" korrekte Ergebnisse liefert. (1 Punkt)

3. Stellen Sie Funktionalität zum Editieren der Textur zur Verfügung. Die Modifikation soll interaktiv auf der sphärischen Textur ebenso wie auf dem 3D-Modell sichtbar werden. Als Mindestfunktionalität sollen das Setzen von Punkten und das "Ausradieren" gesetzter Punkte ermöglicht werden. Um die neu gesetzten Pixel an das graphische System zu übergeben stehen `glTexSubImage2D(...)` bzw. `glCopyTexSubImage2D(...)` zur Verfügung. Implementieren Sie die Editierfunktionalität in `Texture::mousePressed(...)`, `Texture::mouseDragged(...)` und `Texture::mouseMoved(...)`. Um dem Benutzer die Orientierung zu erleichtern, soll in der 3D Ansicht zusätzlich ein Cursor in Form einer Linie, ausgehend vom Mittelpunkt des Modells, angezeigt werden. (1 Punkt)

4. Erstellen Sie ein `GLSL`-Shaderprogramm, das Environment-Mappping mittels Indizierung einer Sphere Map durch Reflektionsvektoren implementiert. Beleuchtung und Texturierung sollen wieder zu- und abwählbar sein. Stellen Sie sicher, dass Ihr Algorithmus für die Texturen aus der Rubrik "`ENVIRONMENT`" `TEXTURES`" korrekte Ergebnisse liefert. Der Benutzer soll sowohl die ganze Umgebung, als auch das Modell innerhalb der Umgebung bewegen können! Für ersteres steht Ihnen die Matrix `Texture::mirrorMatrix` zur Verfügung. (1 Punkt)

5. Wie könnten die behandelten Techniken dazu eingesetzt werden, in Echtzeit Silhouetten zu zeichnen? Erstellen Sie eine passende Textur, und erweitern Sie Ihr Programm geeignet! (1 Punkt)

   Zusatzpunkt (freiwillig):

6. Die applizierten Texturen weisen an den Rändern Artefakte auf. Was ist die Ursache? Implementieren Sie in der Klasse `TriMesh` eine geeignete Lösung des Problems. (1 Punkt)

7. Implementieren Sie trilineare Texturfilterung in Software. Entwickeln Sie hierfür in der Klasse `Image` eine geeignete Datenstruktur für Mipmaps, und applizieren Sie die Texturen perspektivisch korrekt in einem Shaderprogramm. Ihre Resultate können Sie vergleichen mit denjenigen aus Unteraufgabe 1. (1 Punkt)

Achtung: Diese Aufgabe wird erneut gestellt beim nächsten Aufgabenblatt 5. In Vorgriff auf den Raytracer-Contest ist die Lösung der Zusatzaufgabe jedoch bereits jetzt zu empfehlen!

AUFGABE 2 : Theoriefragen

1. Da die Rasterisierung eines texturierten Dreiecks im (2D) Bildraum durchgeführt wird, kann es zu Abbildungsfehlern kommen wenn zwischen den Eckpunkt-Texturkoordinaten linear interpoliert wird. Beschreiben Sie (in Worten) wie und wann dieser Fehler entsteht. (1 Punkt)

2. Unter welchen Umständen ist der Einsatz einer MipMap zur Texturierung sinnvoll? (1 Punkt)

3. Die Zweischritt-Verfahren zur Bestimmung von Texturkoordinaten bilden das Modell zumeist nicht bijektiv auf den umschreibenden Körper ab. Beschreiben Sie die Familie von Objekten, die sich durch Zentralprojektion bijektiv auf einen Kugel abbilden lassen! (1 Punkt)

4. Warum kommt es bei der Parametrisierung einer Environment Map in planaren (u,v) Koordinaten zwangsläufig zu Verzerrungen? (es kommt auch zu Singularitäten, danach ist aber nicht gefragt) (1 Punkt)

5. Projektives Texture Mapping ist eine Technik, die eingesetzt wird, um Texturkoordinaten dynamisch zu generieren, indem die 3D-Positionen der gerenderten Geometrie in eine 2D-Textur abgebildet werden. Eine mögliche Anwendung ist die Projektion von Bildern auf 3D-Objekte in einer Szene, ähnlich einem Tageslichtprojektor (siehe Abbildung).

   - Beschreiben Sie die Transformationen, die auf Geometrie angewendet werden müssen, um diese Technik zu implementieren. (Hinweis: Diese sind sehr ähnlich zu denen, die bei der Abbildung auf den Bildschirm angewendet werden.) (1 Punkt)
   - Erklären Sie, wie diese Technik zum Zweck der Schattenberechnung eingesetzt werden kann (sog. Projective Shadow Mapping). (1 Punkt)

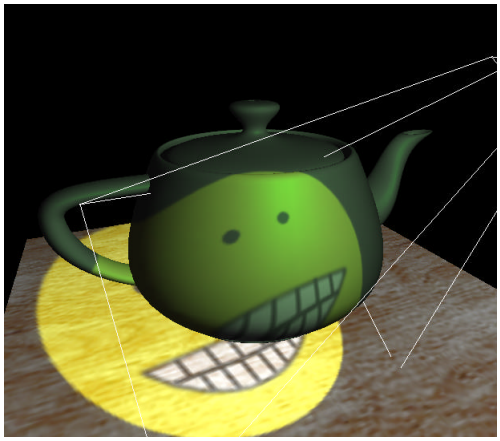

Abbildung 1: „Projective Texture Mapping“[Eve01]

Literatur

[Eve01] Cass Everitt. Projective texture mapping. *White paper, NVidia Corporation*, 4, 2001.