# 3. Aufgabenblatt zur Veranstaltung *Computergrafik und Visualisierung II* im Sommersemester 2020

**Bearbeitungszeitraum: 21.04.-08.05.2020**

Hochschule für Technik und Wirtschaft Dresden, Friedrich-List-Platz 1, 01069 Dresden

Prof. Dr. Marco Block-Berlitz, Rainer Uhlemann

Im dritten Teil der Veranstaltung werden wir mit **GLSL** und der Shaderprogrammierung beginnen. Hinweis: Die Lösungen der praktischen Übungsaufgaben können als Gruppe (max. 3) bearbeitet werden und sind spätestens am **08.05.2020** einzureichen.

## 1 Einarbeitung in die Grundlagen

Eclipse ist bereits eingerichtet und die Buchprogramme[1] sind entsprechend vorbereitet. Jetzt wollen wir das **Kapitel 2 „Eigene Shader in GLSL entwickeln“** im Buch zur Veranstaltung durcharbeiten [1]. Spezielle Fragen zu GLSL können sicherlich durch das Orange-Book beantwortet werden [3]. Bei Opal befinden sich weiterhin Folien zu dem Kontext, die Sie ebenfalls noch zu rate ziehen können.

## 2 Übungsaufgaben

Zur Erinnerung: Die Übungsaufgaben haben den Zweck, Sie auf die Klausur vorzubereiten. Alle Aufgaben sollten durch das Buch zur Veranstaltung beantwortet werden können [1].

1. Erläutern Sie die vereinfachte Renderpipeline von OpenGL anhand einer Skizze.
2. Welche Aufgabe hat ein Vertex-Shader? Geben Sie zur Erläuterung ein möglichst kurzes Code-Beispiel an.
3. Welche Aufgabe hat ein Fragment-Shader? Geben Sie zur Erläuterung ein möglichst kurzes Code-Beispiel an:
4. Wie lassen sich zusätzliche Informationen vom Vertex- zum Fragment-Shader übertragen, die dann den Rasterisierungsprozess passieren? Erläutern Sie das an einem sinnvollen Beispiel. Dafür müssen Sie keinen Programmcode angeben, es genügt eine aussagekräftige Skizze.
5. Welches Konzept unterstützt Sie dabei, die im Shader verwendeten Parameter direkt zu manipulieren? Geben Sie dafür ein kleines Code-Beispiel an.
6. Warum ist die ebenfalls gebräuchliche Bezeichnung Pixel-Shader statt der korrekten Bezeichnung Fragment-Shader irreführend? Erläutern Sie den Sachverhalt.
7. Wie lässt sich eine einfache Wasserdynamik über Konvolution realisieren? Ist das Vorgehen geeignet, um es in einem Shader berechnen zu lassen? Wenn ja, im Vertex- oder Fragmentshader?
8. Beschreiben Sie, wie für Refraktion und Spiegelung aus der Höhenkarte des Wassers die Normale ermittelt werden kann.

[1] http://www.vividus-verlag.de/beleuchtung_und_rendering/index.html

# 3 Praktischer Teil

1. [0 Punkte] In Kapitel 2 gibt es verschiedene Beispielprogramme und Projektzwischenstufen [1]. Führen Sie alle Kapitelbeispiele aus und probieren Sie dabei verschiedene Parameter. So ist bspw. der Dämpfungsfaktor bei der Wassersimulation ein interessanter Faktor.
2. Wir wollen die Beispielprogramme ein wenig anpassen und damit die Gelegenheit bekommen, ein paar Zusatzpunkte einzusammeln. Ihr Dozent hat in der Programmsammlung, die bei OPAL zu finden ist, neue Klassen angeboten. Die Klasse `POGL` hält bspw. eine neue Methode `loadModel` bereit, mit der Objekte im OBJ-Format eingelesen werden können. Mit der Klasse `ObjektLadenUndDrehen` können wir Objekte laden und direkt anzeigen lassen.

   **a)** [10 Zusatzpunkte] Modellieren Sie in einem Tool Ihrer Wahl (Maya, Cinema 4D, 3DS Max, Blender, ...) ein interessantes, untexturiertes Objekt (bestehend aus **Dreiecken**), exportieren es im OBJ-Format und laden es mit der Klasse `ObjektLadenUndDrehen`. Falls Sie Schwierigkeiten beim Einlesen des Formats haben, ergründen Sie die Problematik und passen den Code gegebenenfalls an.
   **b)** [30 Zusatzpunkte] Modellieren Sie in einem Tool Ihrer Wahl (Maya, Cinema 4D, 3DS Max, Blender, ...) ein interessantes, untexturiertes Objekt (bestehend aus **Vierecken**), exportieren es im OBJ-Format und laden es mit der Klasse `ObjektLadenUndDrehen`. Jetzt müssen Sie verstehen, wie ein Objekt aufgebaut ist und eine Klasse für die Repräsentation eines Vierecks analog zur Klasse `Triangle` implementieren und entsprechend Parser und Visualisierung anpassen.
   **c)** [15 Punkte] Mit der Klasse `QuadGelb` haben wir das erste vollständige Shaderprogramm in Kapitel 2 erlebt. Passen Sie das Programm so an, dass ein Osterei mit bunten Punkten statt des gelben Vierecks gezeigt wird. Hinweis: Dazu müssen Sie sich Gedanken machen, wie die Komposition der OpenGL-Objekte mit Ihren individuellen Eigenschaften, im Shader dargestellt werden können. Sie können selbst entscheiden, ob Sie Buffered-Objects verwenden.

## Literatur


[1] Block-Berlitz, M.: *„Warum sich der Dino furchtbar erschreckte: Lehrbuch zu Beleuchtung und Rendering mit Java, LWJGL, OpenGL und GLSL“*, vividus Wissenschaftsverlag, 2019

[2] Shreiner D., Woo M., Neider J., Davis T.: *„OpenGL Programming Guide“*, 6. Auflage, Pearson Education, Addison-Wesley Verlag, 2008

[3] Rost R.J., Licea-Kane B.: *„OpenGL Shading Language“*, 3. Auflage, Pearson Education, Addison-Wesley Verlag, 2010