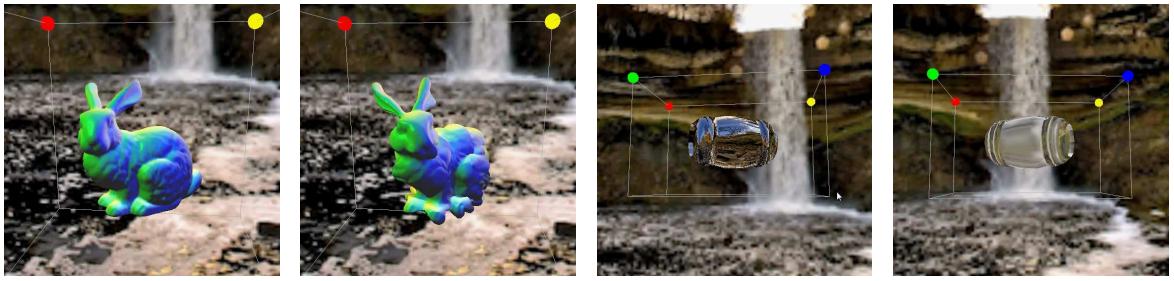
# Praktikum Computergraphik

## – Übungsblatt 2 –

**Fachgruppe für Computergraphik**
**und Multimediasysteme**
Julian Bader, Björn Labitzke

### 1. Themen

GPU-Programmierung (Vertex- und Fragment-Shader)



(a) (b) (c) (d)

**Figure 1:** *Anschauungsbeispiele: Beleuchtung 1(a), Reflektion 1(c), Brechung 1(d) und Vertexanimation 1(b).*

### 2. Überblick

In dieser Aufgabe sollen Sie sich mit der Nutzung programmierbarer Grafikhardware in OpenGL befassen. Hierzu soll ein Szenerie modelliert werden in der verschiedene Modelle eingeladen werden können, die in ihrer Farb- und Geometrieeigenschaften verändert werden sollen.

Sie erlernen folgende Techniken: GLSL-Shadersyntax, Kompilieren und Einbinden von Vertex- und Fragment-Shadern

### 3. Projektbeschreibung

Es soll eine auf OpenGL basierende Szenerie mittels der Fixed-Function-Pipeline (ohne Shader) umgesetzt werden. In dieser Szenerie sollen verschiedene Modelle eingeladen werden können. Diese eingeladenen Modelle sollen mittels Vertex- und Fragmentshader in ihrer Farbgebung und Geometrieeingenschaften veränderbar sein (vgl.Abb. 1).

### 4. Aufgabe

Das Programmgerüst, ein OBJ-Loader, sowie die Maus-Interaktionen sind bereits implementiert.

Ihre Aufgabe gliedert sich wie folgt:

**Modellierung der Szene [Fixed-Function-Pipeline]**

Ein einfacher OBJ-Loader ist bereits im Programmgerüst enthalten und soll zum einladen von vordefinierten Modellen genutzt werden. Das einladen kann z.B. wie folgt durchgeführt werden

```
ObjLoader* objMesh = new ObjLoader(tr("../resources/Barrel.obj") .
```

Ebenfalls enthält das Gerüst bereits drei Modelle, es steht Ihnen jedoch auch frei weitere Modelle Ihrem Projekt hinzuzufügen. Der OBJ-Loader bestimmt zudem auch die Axis Aligned Boundingbox (AABB) der geladenen Geometrie. Nutzen Sie die AABB, um die Geometrie mit einer linierten Box zu umhüllen. Skalieren Sie diese Box so, dass die Geometrie beim rotieren um die Y-Achse nicht aus der Box ragt.

Des Weiteren setzen Sie bitte die folgenden Aufgaben um:

- Die Szene soll vier Lichtquellen enthalten, die jeweils durch eine Kugel in der jeweiligen Lichtfarbe repräsentiert werden sollen. Die Lichter sollen an den oberen Ecken der linierten Box positioniert werden (vgl.Abb. 1). Die Lichtquellen sollen individuell per GUI de- und aktivierbar sowie in ihrer Farbgebung veränderbar sein. Das deaktivieren eines Lichts soll sich auch auf die Farbgebung der zugehörigen Repräsentation auswirken.
- Umschließen sie die gesamte Szene in einer *Skybox* welche mit der zur Verfügung gestellten Cube-Map textuiert werden soll.
- Das Modell soll sich um die eigene Y-Achse rotieren, wenn die Animation aktiv ist. Die GUI soll es einem Nutzer ermöglichen den Status der Animation zu ändern und zwischen den Geometrien zu wechseln.
- Die zur Darstellung gewählten Parameter sind sinnvoll zu wählen, so dass die aktuelle Geometrie im initialen Zustand der Anwendung fensterfüllend und ohne geclippte Objekte angezeigt wird.

**Farbgebung und Geometrieeingenschaften [Shader]**

Setzen Sie sich zunächst mit dem erstellen von Shader-Programmen und dem einbinden von Shadern auseinander. Das Programmgerüst enthält bereits zwei Test-Sahder die Sie hierzu nutzen können. Anschließend setzten Sie bitte die nachfolgenden Aufgaben um.

- Implementieren Sie Gouraud- und Phong-Shading (vgl. Abb. 1(a)) und ermöglichen Sie über die GUI den Wechsel zwischen beiden Beleuchtungsarten. Verwenden Sie zur Berechnung der Farbwerte alle aktiven Lichter. Der ambiente Term soll aktiv sein, solange mindestens ein Licht aktiv ist.
- Implementieren Sie ein Environment Mapping mittels einer Cube-Map (GL_TEXTURE-_CUBE_MAP). Nutzen Sie die GLSL-Funktionen `reflect(...)` sowie `refract(...)`, um den Texture-Lookup mittels `rextureCube(...)` vorzunehmen. Die resultierende Farbe soll zwischen dem Reflektions- und Brechungswert interpoliert werden (veränderbar mittels GUI). Hierduch sollen Glaseffekte nachgestellt werden (vgl. Abb. 1(c) und 1(d)).
- Die finale Farbe des Fragments soll zwischen der Beleuchtungsfarbe und der Farbe des Environment Mappings interpoliert werden. Auch hier soll der Interpolationsparameter via GUI änderbar sein.
- Die Geometrieeigenschaften sollen mittels des Vertex-Sahders in Abhängigkeit von der Anima-

tionszeit und der beigefügten Textur `sine.png` geändert werden (vgl. Abb. 1(b)). Die Amplitude und Frequenz der Vertex-Verschiebungen sollen auch mittels der GUI änderbar sein.

- Das Verwenden von Shadern ermöglicht Grafik-Entwicklern viele Freiheiten. Nutzen Sie Ihre Kreativität, um neben den genannten Aufgaben auch noch einen individuellen Effekt zu entwickeln.

**GUI**

Erweitern Sie das Qt-Hauptfenster durch einen Einstellungsdialog für die vorhergehend genannten Programm-Parameter.

### 5. Bewertung

Bei der Bewertung werden folgende Kriterien berücksichtigt:

1. Die Aufgaben sind gemäß der Aufgabenstellung gelöst.
2. Das gesamte Projekt unterliegt einer objektorientierten Denkweise, genügt den Grundlagen der modularen Gestaltung und ist übersichtlich editiert.
3. Der Quellcode ist in englischer Sprache kommentiert und enthält sämtliche Hilfen, die zum Verständnis der Programmstruktur bzw. der verwendeten Algorithmen notwendig sind.
4. In einem Gespräch sollte jeder Teilnehmer in der Lage sein, den Programmcode kurz zu erläutern und allgemeine Fragen zu OpenGL beantworten zu können.