GLSL Hinweise zu Übungsblatt 4

Schon wieder eine andere Programmiersprache, mit welcher ihr während eures Studiums arbeiten müsst... Aber keine Sorge, GLSL ist im Vergleich zu C++ wirklich überschaubar. Und das, was ihr mit GLSL machen müsst, ist noch viel überschaubarer. Shader sind im Normalfall nur sehr kleine, kurze Miniprogramme.

Ein paar Keyfacts:

- Shader sind **nicht** objektorientiert.
- Shader unterstützen **keine** rekursiven Funktionsaufrufe.
- Shader unterstützen **keine** dynamischen Arraygrößen.

Shader werden üblicherweise erst während der Laufzeit des C++ Programms geladen und kompiliert – darum kümmert sich die vorgegebene `Shader`-Klasse. Das heißt allerdings auch, dass eure C++ Programme selbst dann kompilieren und starten, wenn ihr einen Fehler im Source Code eines Shaders habt. Wenn es beim Kompilieren eines Shaders zu einem Fehler kommt, gibt die vorgegebene `Shader`-Klasse euch die Fehlermeldung in der Konsole aus, auch wenn das Programm weiterläuft (und etwas falsches anzeigt).

Ein Shader-Programm, mit dem ein 3D-Objekt gerendert wird, besteht übrigens immer mindestens aus einem Vertex-Shader und Fragment-Shader. Die vorgegebene `Shader`-Klasse kompiliert dabei sowohl den Vertex- als auch Fragment-Shader und erstellt daraus ein gemeinsames Shader-Programm, welches auf der Grafikkarte liegt – wenn dieses Shader-Programm über OpenGL-Befehle aktiviert wird, werden Vertex- und Fragment-Shader also immer gemeinsam aktiv.

Typen

Die einzigen Typen, die im Übungsbetrieb von CG1 für euch relevant werden können, sind:
`bool`, `int`, `float`, `vec2`, `vec3`, `vec4` und `mat4`

Primitive Typen

Die primitiven Datentypen `bool`, `int` und `float` verhalten sich wie die Datentypen, die ihr bereits aus Java und C++ kennt – ihr könnt damit rechnen (z.B. `+`, `-`, `*` oder `/`) und sie in neue lokale Variablen speichern.

Vektoren

Die Vektortypen enthalten folgende Attribute, die jeweils vom Typ `float` sind:

- `vec2` besitzt die float-Attribute `x` und `y`.
- `vec3` besitzt die float-Attribute `x`, `y` und `z`.
- `vec4` besitzt die float-Attribute `x`, `y`, `z` und `w`.

Vektor-Beispiel:

Einen 4D-Vektor kann man z.B. wie folgt erstellen und Werte zuweisen (wichtig ist dabei, dass ihr bei einem vec4 auch immer vier Werte im Konstruktor bereitstellt, sonst gibt es einen Kompilierfehler):

```
// Create local variable:
vec4 position = vec4(1.0, 0.0, 0.0, 1.0);
```

Auf die einzelnen Elemente zugreifen kann man wie auf Attribute einer Klasse in C++ (auch wenn es keine Klasse ist):

```
float x = position.x;
```

GLSL-Shader erlauben es auch, gleichzeitig auf mehrere Werte zuzugreifen und diese als einen Vektor zurückzugeben:

```
vec2 x = position.xy;
```

Auf gleiche Weise könnt ihr die Werte einfach umdrehen, indem ihr einfach die Attribute in anderer Reihenfolge aufruft:

```
vec4 swappedComponents = position.wzyx;
```

Das macht hier zwar nicht so viel Sinn, aber kann manchmal ganz hilfreich sein, z.B. wenn man x- und y-Achse vertauschen will.

Matrix

Auf einzelne Werte einer Matrix müsst ihr in GLSL im Übungsbetrieb von CG1 nicht zugreifen. Ihr müsst auch keine Matrix per Hand anlegen.

In späteren Übungsblättern wird an wenigen Stellen lediglich die Matrix-Multiplikation sowie die Matrix-Vektor-Multiplikation verwendet. Diese funktionieren genauso, wie wir sie in C++ implementiert haben mithilfe des $\star$-Operators.

Wenn ihr im Vorfeld eine Matrix vom Typ `mat4` mit dem Namen `myMat` definiert habt, so könnt ihr z.B. schreiben:

```
// Define a vector:
vec4 myVec = vec4(1.0, 0.0, 0.0, 1.0);

// Multiplicate a defined matrix with this vector:
vec4 myResult = myMat * myVec;
```

Ein paar Keyfacts:

Bedingungen und Schleifen

If-Bedingungen

Wie in C++ könnt ihr in GLSL auch einfach if-Bedingungen nutzen, z.B.:

```
// If the x-component of myVec is bigger than 0.5, do something:
if(myVec.x > 0.5){
   // Code which is executed if the condition is true.
} else {
   // ...
}
```

Schleifen

In GLSL gibt es die typische `for`- und `while`-Schleife, wie ihr sie aus Java und C++ kennt. Da der Syntax wie bei der If-Bedingung identisch ist, gibt es hier keine Beispiele dazu.