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

Bearbeitungszeitraum: 31.03.-20.04.2020

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

Prof. Dr. Marco Block-Berlitz, Rainer Uhlemann

Im zweiten Teil der Veranstaltung werden wir etwas in **OpenGL** einsteigen und die ersten Vorbereitungen für GLSL starten. Dazu müssen wir Eclipse beibringen, mit LWJGL in der Version 2 zu arbeiten.

Hinweis: Die Lösungen der praktischen Übungsaufgaben sind spätestens am **20.04.2020** einzureichen.

1 Einarbeitung in die Grundlagen

Lesen und verstehen Sie die Inhalte zur Einführung in OpenGL und den Zusammenhang zu LWJGL in Kapitel 1 [1]. Es gibt dazu einen ergänzenden Kapitelteil bei Opal zu finden. Das Passwort erhalten Sie per Email von mir. Nehmen Sie sich weiterhin das Red-Book bei Fragen zu OpenGL zu Hilfe [2]. Ein Foliensatz mit zusätzlichen Hilfestellungen zur Einrichtung von LWJLG in Eclipse befindet sich bei Opal. Ein weiterer Foliensatz zu den OpenGL-Beispielen ist ebenfalls vorhanden. Für die weitere Programmierarbeit wird es notwendig sein, den zur Verfügung stehenden Programmcode[1] herunterzuladen und startklar zu machen. Probieren Sie die Beispiele aus.

2 Übungsaufgaben

1. Beschaffen Sie sich das Red- und das Orange-Book für die Arbeit mit OpenGL [2] und GLSL [3]. Die Links zu den PDFs haben Sie bereits per Email erhalten.
2. Stellen Sie eine Recherche zum Thema **Buffer-Objects** in OpenGL an und machen Sie sich damit vertraut, dass es seit OpenGL 3.1 besser ist, Vertex-Daten blockweise zu übergeben und auf der Grafikkarte zu verwalten.
3. [10 Zusatzpunkte] Geben Sie auf ein bis zwei Seite ein systematisches Rezept an, wie sich sich klassische OpenGL-Beispiele, wie:

```
1 glBegin(GL_TRIANGLES);
2    glVertex3d(-.5f, -.5f + 0.3 * Math.sin(t), 0);
3    glVertex3d(.5f, 0 + 0.3 * Math.sin(t * 1.3), 0);
4    glVertex3d(+0.3 * Math.sin(t * 2), .5f, 0);
5 glEnd();
```

   jetzt mit Buffer-Objekts (VBOs, VAOs) in Java und LWJGL notieren lassen.

3 Praktischer Teil

1. [20 Punkte] In Kapitel 1 [1] gibt es die Beispielprogramme „Bewegtes Dreieck“, „Drehender Würfel“, „Würfelprojektion auf eine Fläche“ und „Wehende Fläche“. Analog zu Ihrem erstellten Rezept aus Übungaufgabe 2 sollen diese vier Beispiele nun mit Buffer-Objekts realisiert werden.

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

2. Die bisherigen Visualisierungsbeispiele aus Kapitel 1 [1] sollen um andere geometrische Formen erweitert werden. Wir hatten bspw. einen Würfel in OpenGL konstruiert und in der Funktion `renderWuerfel` in der Klasse `POGL` implementiert:

```
1  public static void renderWuerfel() {
2     glBegin(GL_QUADS);
3     glVertex3f(-1, -1, -1);
4     glVertex3f(1, -1, -1);
5     glVertex3f(1, 1, -1);
6     glVertex3f(-1, 1, -1);
7
8     glVertex3f(-1, -1, 1);
9     glVertex3f(1, -1, 1);
10    glVertex3f(1, 1, 1);
11    glVertex3f(-1, 1, 1);
12
13    glVertex3f(-1, -1, -1);
14    glVertex3f(-1, 1, -1);
15    glVertex3f(-1, 1, 1);
16    glVertex3f(-1, -1, 1);
17
18    glVertex3f(1, -1, -1);
19    glVertex3f(1, 1, -1);
20    glVertex3f(1, 1, 1);
21    glVertex3f(1, -1, 1);
22
23    glVertex3f(-1, -1, -1);
24    glVertex3f(1, -1, -1);
25    glVertex3f(1, -1, 1);
26    glVertex3f(-1, -1, 1);
27
28    glVertex3f(-1, 1, -1);
29    glVertex3f(1, 1, -1);
30    glVertex3f(1, 1, 1);
31    glVertex3f(-1, 1, 1);
32    glEnd();
33 }
```

Dieser Würfel kann bspw. durch folgendes Programm erzeugt, gedreht und visualisiert werden:

```
1  package kapitel01;
2
3  import static org.lwjgl.opengl.GL11.GL_FRONT_AND_BACK;
4  import static org.lwjgl.opengl.GL11.GL_LINE;
5  import static org.lwjgl.opengl.GL11.glColor3d;
6  import static org.lwjgl.opengl.GL11.glFrustum;
7  import static org.lwjgl.opengl.GL11.glLoadIdentity;
8  import static org.lwjgl.opengl.GL11.glPolygonMode;
9  import static org.lwjgl.opengl.GL11.glRotatef;
10 import static org.lwjgl.opengl.GL11.glScaled;
11 import static org.lwjgl.opengl.GL11.glTranslated;
12
13 import java.awt.Canvas;
14
15 import javax.swing.JFrame;
16
17 import org.lwjgl.LWJGLException;
18 import org.lwjgl.opengl.Display;
19
20 public class WuerfelDrehtSich extends LWJGLBasisFenster {
21    public WuerfelDrehtSich() {
22       super(640, 480);
23
24       JFrame f = new JFrame();
25       Canvas c = new Canvas();
26       f.add(c);
27       f.setBounds(100, 100, 600, 600);
28       f.setTitle("Würfel dreht sich");
29       f.setVisible(true);
30       f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
31
32       initDisplay(c);
33    }
34
35    @Override
36    public void renderLoop() {
37       glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
```

```
38        long start = System.nanoTime();
39        while (!Display.isCloseRequested()) {
40           double t = (System.nanoTime() - start) / 1e9;
41           POGL.clearBackgroundWithColor(1, 1, 1, 1);
42
43           glLoadIdentity();
44           glFrustum(-1, 1, -1, 1, 4, 10);
45           glTranslated(0, 0, -5);
46           glRotatef((float) t * 100, 1, 0, 0);
47           glRotatef((float) t * 100, 0, 1, 0);
48           glScaled(.5, .5, .5);
49
50           glColor3d(0, 0, 0);
51           POGL.renderWuerfel();
52
53           Display.update();
54        }
55     }
56
57     public static void main(String[] args) throws LWJGLException {
58        new WuerfelDrehtSich().start();
59     }
60  }
```

a) [5 Punkte] Entwickeln Sie für die Klasse `POGL` die Funktion `renderKugel`, die eine Kugel im Bereich von -1 bis 1 darstellt.
b) [7 Punkte] Entwickeln Sie für die Klasse `POGL` die Funktion `renderStern(int n)`, die einen Stern mit n Zacken im Bereich von -1 bis 1 darstellt.
c) [9 Punkte] Entwickeln Sie für die Klasse `POGL` die Funktion `renderPyramide`, die eine Pyramide im Bereich von -1 bis 1 darstellt.
Hinweis: Bei den Aufgabenteilen a), b) und c) gibt es insgesamt 10 Zusatzpunkte, wenn Sie die klassische Variante und die Variante mit Buffer-Objects implementieren!

3. Bei der vorhergehenden Aufgabe haben wir die Rotation eines Objekts über die Systemzeit realisiert. Erzeugen Sie weitere komplexere Bewegungsmuster.
a) [5 Punkte] Realisieren Sie ein Bewegungsmuster ohne Rotation, das einer Acht entspricht.
b) [5 Punkte] Realisieren Sie ein Bewegungsmuster mit beliebigen Rotationen um verschiedene Achsen, das einer Acht entspricht.
c) [8 Punkte] Realisieren Sie ein weiteres komplexes Bewegungsmuster, das sich eignet, um erstellte 3D-Objekte aus verschiedenen Blickrichtungen darzustellen.

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