Klausur Computergraphik (WS 2017/18)

Prüfer:	Prof. Dr. R. Dörner, Prof. Dr. C. Schulz, HS RheinMain
Bearbeitungszeit:	90 min
Zugelassene Hilfsmittel:	ein beidseitig handbeschriebenes DIN A4 Blatt, Stifte. (insbesondere Taschenrechner und eigenes Papier ist verboten)
Datum:	22. Februar 2018

Name:_____ Vorname: _____

Matr.-Nr. _____

_____
Unterschrift

Hinweise:

- Überprüfen Sie Ihr Klausurexemplar auf Vollständigkeit (Umfang: 8 Blätter)
- Lösen die Aufgaben im dafür vorgesehenen Raum. Wenn der Platz nicht ausreicht, verwenden Sie die Rückseiten - wenn alle Rückseiten beschrieben sind, fordern Sie ein leeres Blatt bei der Aufsicht an. Schreiben Sie im vorgesehenen Raum einen Hinweis der Art "weiter siehe S. 3 Rückseite". Fehlt dieser Hinweis, ist die Lösung unleserlich oder gibt es mehrere Lösungen zu derselben Aufgabe, so werden keine Punkte vergeben.
- Wer einen Täuschungsversuch begeht oder einem Täuschungsversuch Vorschub leistet erhält die Note "nicht bestanden".
- Es darf nicht mit Bleistift geschrieben werden. Es sind nur Schreibfarben „blau“ oder „schwarz“ zulässig.
- Starten Sie mit der Bearbeitung der Klausur nur, wenn Sie prüfungsfähig sind.
- Die Klausur ist in jedem Fall bestanden mit **44 Punkten.**

Es wurden _____ Punkte erreicht.

Note, Handzeichen:

Aufgabe 1

Gegeben sind die Stützpunkte A, B, C, D einer Beziér-Kurve Q(t), t∈[0,1].

2 P. (a) Zeichnen Sie die konvexe Hülle der Stützpunkte ein.

3 P. (b) Skizzieren Sie den Verlauf der Kurve.

(c) Nennen Sie vier Aspekte, die man bei der Skizze beachten muss:

1. _____

2. _____

3. _____

4 P. 4. _____

(d) Die Koordinaten der Punkte lauten: A(1,2,3), B(4,5,6), C(7,8,9), D(10,11,12). Geben Sie eine Formel aus nicht ausmultiplizierten Matrizen an, mit der man Q(0,4) berechnet.
Hinweis: Die Basismatrix der Beziér-Kurven lautet:

$$M_{Bezier} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

3 P.

Σ2:

Aufgabe 2

Gegeben ist folgende VRML-Szene:

```
DEF T1 Transform {
 translation 1 2 3
 children[
            DEF T2 Transform{
             scale 3 1 1
             children[
                        DEF T3 Transform{
                         translation -1 0 1
                         children[
                                    DEF S Shape{
                                     geometry Sphere{}  }
                        ] }
            ]}
            DEF T4 Transform{
             translation  1 0 1
             rotation     1 0 0 3.14
             children[
                        DEF K Viewpoint{
                         fieldOfView 0.7
                         position 5 0 1
                         orientation 1 0 0 1.57}
            ]}
]}
```

(a) Zeichnen Sie den Szenengraph (nur Transform-, Shape- und Viewpoint-Nodes, keine Fields)

4.5 P.

(b) Wie lauten die Koordinaten des Mittelpunkts der Kugel S in Weltkoordinaten?

5.5 P.

Σ3:

(c) Wie lauten die Koordinaten des Mittelpunkts der Kugel S in Kamerakoordinaten der Kamera K?

6 P.

(d) Wie lauten die Koordinaten des Augpunkts der Kamera K in Objektkoordinaten der Kugel S?

7 P.

Σ4:

Aufgabe 3

Gegeben ist folgender Ausschnitt aus einem WebGL-Javascript, wobei die in der Lehrveranstaltung vorgestellten Hilfsfunktionen verwendet werden:
mat4() *„erzeugt eine 4x4 Einheitsmatrix“*,
mult(m1, m2) *„berechnet das Matrixprodukt der Matrizen m1 und m2“*,
transpose(m1) *„transponiert die Matrix m1“*, inverse(m1) *„invertiert die Matrix m1“*,
rotate(alpha, [x,y,z]) *„erzeugt eine 4x4 Rotationsmatrix um die Achse $(x,y,z)^T$ um den Winkel alpha“*,
translate(x,y,z) *„erzeugt eine 4x4 Translationsmatrix für den Translationsvektor $(x,y,z)^T$“*,
scale(sx,sy,sz) *„erzeugt eine 4x4 Skalierungsmatrix für die Skalierungswerte sx, sy, sz“*,
perspective(fov, aspect, near, far) *„erzeugt eine Projektionsmatrix“*

```
// Projektionsmatrix
var projection = perspective(50.0, 1.0, 0.01, 300.0);

// Zeile A: hier die Model-Matrix mA anlegen

var mA =




// Zeile B: hier die View-Matrix mB anlegen

var mB =




// Zeile C: hier Matrix mC anlegen, die Objektkoordinaten in Clipping-Koordinaten umrechnet

var mC =



// Zeile D: hier Matrix mD anlegen, die Normalen von Objektkoordinaten in Kamerakoordinaten
// umrechnet

var mD =
```

4 P. (a) Ergänzen Sie das Programm nach Zeile A so, dass alle Modelle zuerst um die Achse durch die Punkte A(2,2,2) und B(4,2,1) um 30° gedreht und dann um das 2-fache in z-Richtung skaliert werden.

3 P. (b) Ergänzen Sie das Programm nach Zeile B so, dass entsprechend lookAt(0,5,1,0,1,1,0,0,-1) die Kamera positioniert wird (verwenden Sie dabei nur die oben angegebenen Hilfsfunktionen)

2 P. (c) Ergänzen Sie das Programm nach Zeile C so, dass eine Matrix mC angelegt wird, die Vertices von Objektkoordinaten in Clipping-Koordinaten umrechnet

2 P. (d) Ergänzen Sie das Programm nach Zeile D so, dass eine Matrix mD angelegt wird, die Normalen von Objektkoordinaten in Kamerakoordinaten umrechnet

Σ5:

(e) Wie ändert sich das Bild, wenn perspective(50.0, 1.0, 0.01, 300.0) abgeändert wird in: perspective(20.0, 1.0, 100.0, 300.0); ?

3 P.

(f) Ergänzen Sie den unten stehenden GLSL Vertex-Shader und Fragment-Shader möglichst einfach, um eine Phong-Beleuchtungsrechnung zu realisieren, die nur den diffusen Anteil berücksichtigt. Ist der Punkt mehr als eine Distanz d von der Lichtquelle entfernt, soll der Punkt schwarz erscheinen. Basierend auf den durch die Beleuchtungsrechnung ermittelten Farbwerten soll ein Gouraud-Shading durchgeführt werden.

```
void main(){  // Vertex-Shader
              uniform mat4  clipMat;      //  Matrix zur Umrechnung von Welt- in Clippingkoordinaten
              uniform mat4 worldMat;      // Matrix zur Umrechnung von Objekt- in Weltkoordinaten
              uniform vec3 lightPos;      // Position einer Punktlichtquelle L in Weltkoordinaten
              uniform float d;            // maximale Reichweite von L in Weltkoordinaten
              uniform vec 3 lightColor;   // Farbe/Intensität des Lichts (in RGB) von L
              attribute vec3 diffColor;   // Koeffizient für diffuse Reflektion (in RGB)
              attribute vec3 normal;      // die dem Vertex zugeordnete Normale
              attribute vec3 position;    //  die dem Vertex zugeordnete Position in Objektkoordinaten
```

```
}

void main() {  // Fragment-Shader
```

8 P.

```
}
```

Σ6:

Aufgabe 4

Der Punkt P(2, 1, 0) soll mit einer Kamera, die sich an Punkt A(0,5,0) befindet, auf die Projektionsebene mit der Gleichung y = - 2 perspektivisch projiziert werden. Die Bildkoordinaten P' von P sind mit der aus der Vorlesung bekannten Matrix $M_{per}(d)$ zu berechnen.

(a) Um M_{per} anwenden zu können, muss eine Standardsituation eingehalten werden:
Wo muss sich die Kamera befinden?

Wohin muss die Kamera schauen?

Wo muss sich die Projektionsebene befinden?

3 P.

(b) Wie kann man die Standardsituation für $M_{per}(d)$ erreichen?

3 P.

(c) Berechnen Sie die Bildkoordinaten von Punkt P.

4 P.

(d) Geben Sie die Ebenengleichung der verbotenen Ebene an.

2 P.

Aufgabe 5

(a) Nennen Sie zwei prinzipielle Ansätze zur Realisierung von Anti-Aliasing

1. _____

2 P. 2. _____

Σ7:

(b) Welche Informationen benötigt man um normalisierte Gerätekoordinaten in Bildkoordinaten umzurechnen?

2 P.

(c) Nennen Sie zwei Vorteile der Parallelprojektion gegenüber der perspektivischen Projektion:

1. _____

2 P. 2. _____

(d) Was bezeichnet man in der Computergraphik mit „spekularem Licht“?

3 P.

(e) Was versteht man in der Computergraphik unter „Culling“? Nennen Sie zwei Beispiele und erläutern Sie diese kurz.

4 P.

(f) Gegeben ist folgender Ausschnitt eines GLSL – Shaders:

```
vec4 v = vec4(1.0, 2.0, 3.0, 4.0);
vec4 u = vec4(1.5, 2.5, 3.5, 4.5);
v = u.baba;
v.s = u.q;
```

2 P. Welchen Wert hat v nach Ausführung der letzten Zeile? v = (_____, _____, _____, _____)

(g) Nennen Sie zwei Möglichkeiten, wie das Texturmapping realisiert werden kann:

1. _____

_____

2. _____

3 P. _____

Σ8: