Klausur Computergraphik (WS 2018/19)

Prüfer:	Prof. Dr. R. Dörner, HS RheinMain
Bearbeitungszeit:	90 min
Zugelassene Hilfsmittel:	ein beidseitig handbeschriebenes DIN A4 Blatt, Stifte. (insbesondere Taschenrechner und eigenes Papier ist verboten)
Datum:	7. März 2019

Name:______________________________ Vorname: _________________________

Matr.-Nr. ___________________

____
Unterschrift

Hinweise:

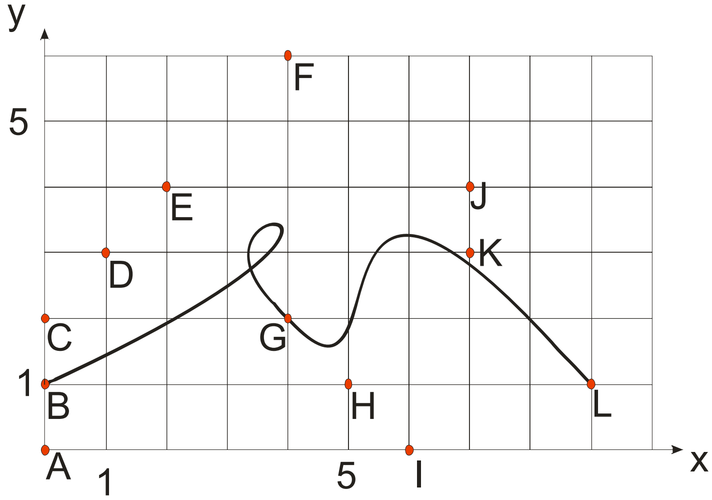
- Überprüfen Sie Ihr Klausurexemplar auf Vollständigkeit (Umfang: 8 Blätter)
- Lösen die Aufgaben im dafür vorgesehenen Raum. Wenn der Platz nicht ausreicht, verwenden Sie die Rückseiten - wenn alle Rückseiten beschrieben sind, fordern Sie ein leeres Blatt bei der Aufsicht an. Schreiben Sie im vorgesehenen 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 **43 Punkten.**

Es wurden __________ Punkte erreicht.

Note, Handzeichen:

Aufgabe 1

Gegeben sind Punkte A, B, C, …L sowie ein **Bézier-Spline**, der aus zwei kubischen Bézier-Kurven $Q_1(t)$ und $Q_2(t)$, jeweils $t \in [0,1]$, zusammengesetzt ist, die G^1-stetig ineinander über gehen, aber nicht C^1-stetig.



(a) Ergänzen Sie jeweils einen der Punkte A, B, …, L (mit Begründung)?

Stützpunkt 1 von Q_1 ist ____, weil ______________________________

Stützpunkt 2 von Q_1 ist ____, weil ______________________________

Stützpunkt 3 von Q_1 ist ____, weil ______________________________

Stützpunkt 4 von Q_1 ist ____, weil ______________________________

Stützpunkt 1 von Q_2 ist ____, weil ______________________________

Stützpunkt 2 von Q_2 ist ____, weil ______________________________

Stützpunkt 3 von Q_2 ist ____, weil ______________________________

8 P. Stützpunkt 4 von Q_2 ist ____, weil ______________________________

(b) Die Punkte A, B, C, …, L sollen die Stützpunkte für einen kubischen uniformen B-Spline R(t) werden. Ergänzen Sie den Knotenvektor:

2 P. T = [2,5, ____]

(c) Berechnen Sie R(17). Die Basismatrix für uniforme kubische B-Splines lautet dabei:

$$M_{uniformer\_B-Spline} = \frac{1}{6} \cdot \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$$

Σ2:

6 P.

Aufgabe 2

Gegeben ist folgende VRML-Szene:

```
DEF T1 Transform {
 scale 1 1 3
 children[
           DEF T2 Transform{
            scale 3 1 1
            children[
                      DEF K Viewpoint{
                       fieldOfView 0.7
                       position 2 2 0
                       orientation 0 0 1 1.57}

                      DEF T3 Transform{
                       rotation     0 0 2 1.57
                       translation  1 1 1
                       children[
                                 DEF S1 Shape{
                                  geometry Sphere{}  }
                      ] }
           ]}
           DEF T4 Transform{
            translation 2 2 2
            children[
                      DEF S2 Shape{
                              geometry Sphere{}  }
           ]}
]}
```

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

5 P.

Σ3:

(b) Wie lauten die Koordinaten des Mittelpunkts der Kugel S1 in Weltkoordinaten?

5 P.

(c) Wie lauten die Koordinaten des Mittelpunkts der Kugel S2 in Kamerakoordinaten der Kamera K?

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(60.0, 1.0, 2.0, 3.0);

// 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 =
```

4 P. (a) Ergänzen Sie das Programm nach Zeile A so, dass alle Modelle zuerst um 3 in x-Richtung transliert und danach um die Achse durch die Punkte A(5,18,17) und B(39,0,1) um 30° gedreht werden.

4 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

(d) Wie ändert sich das Bild, wenn perspective(60.0, 1.0, 2.0, 3.0) abgeändert wird in: perspective(60.0, 1.0, 20.0, 30.0); ?

2 P.

Σ5:

(e) Ergänzen Sie den unten stehenden GLSL Vertex-Shader und Fragment-Shader möglichst einfach, um dem Vertex die Farbe Blau zuzuordnen, wenn er mehr als eine Distanz *d* (im Kamerakoordinatensystem) von der Kamera entfernt ist und die Farbe Rot sonst. Basierend auf den ermittelten Farbwerten soll ein Gouraud-Shading durchgeführt werden.

```
void main(){  // Vertex-Shader
          uniform mat4  clipMat;       //  Matrix zur Umrechnung von Objekt- in Clippingkoordinaten
          uniform mat4  viewMat;       // Matrix zur Umrechnung von Objekt- in Kamerakoordinaten
          uniform float d;             // Distanz für die Farbberechnung
          attribute vec3 position;     //  die dem Vertex zugeordnete Position in Objektkoordinaten

}

void main() {  // Fragment-Shader

}
```

8 P.

(f) Beschreiben Sie, wie der Shader-Code in WebGL kompiliert und zur Ausführung gebracht wird.

3 P.

Σ6:

Aufgabe 4

Der Punkt P(1, 2, 0) soll mit einer Kamera, die sich an Punkt A(-1,0,0) befindet, auf die Projektionsebene mit der Gleichung x = - 8 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 Koordinaten eines Punktes Q an, der sich nicht projizieren lässt.

2 P.

Aufgabe 5

(a) Nennen Sie zwei innere Parameter einer Kamera

1. ______________________________

2 P. 2. ______________________________

Σ7:

(b) Warum sollte das Clipping nach dem Culling durchgeführt werden?

3 P.

(c) Nennen Sie einen Vorteil und einen Nachteil von Rastergrafiken gegenüber Vektorgrafiken:

1. ____

2 P. 2. ____

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

3 P.

(e) Wie verhält sich der Platzbedarf einer MipMap zur Ausgangstextur (wobei diese quadratisch mit einer Zweierpotenz als Breite und Länge sein soll)?

3 P.

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

```
vec4 v = vec4(1.0, 2.0, 3.0, 4.0);
vec4 u = vec4(5.0, 6.0, 7.0, 8.0);
v = u.bara;
v.q = u.t;
```

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

(g) Nennen Sie zwei Möglichkeiten, wie man einen nicht-uniformen B-Spline durch einen bestimmten Stützpunkt zwingen kann:

1. ____

____

2. ____

2 P. ____

Σ8: