Klausur Computergraphik (SS 2018)

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:	13. Juli 2018

Name:_____ Vorname: _____

Matr.-Nr. _____

_____
Unterschrift

Hinweise:

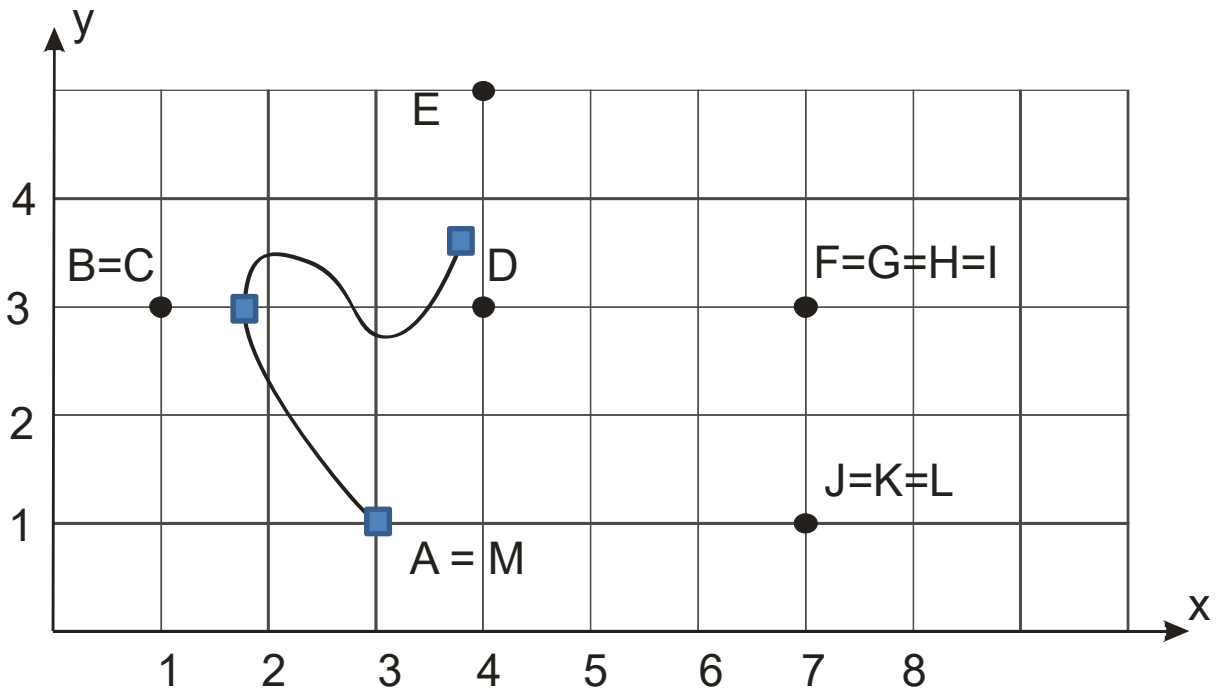
- Überprüfen Sie Ihr Klausurexemplar auf Vollständigkeit (Umfang: 8 Blätter)
- Lösen die Aufgaben im dafür vorgesehenen Raum. Wenn der Platz nicht ausreicht, verwenden Sie die Rückseiten - wenn alle Rückseiten beschrieben sind, fordern Sie ein leeres Blatt bei der Aufsicht an. Schreiben Sie im vorgesehenen 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.
- Die Klausur ist in jedem Fall bestanden mit **37 Punkten.**

Es wurden _____ Punkte erreicht.

Note, Handzeichen:

Aufgabe 1

Gegeben ist eine kubische B-Spline-Kurve Q(t) mit den Stützpunkten A, B, C, …, M (siehe Zeichnung) und dem (unvollständigen) Knotenvektor T = [22, 24, …].



(a) Vervollständigen Sie den Knotenvektor T so, dass Q(t) ein uniformer B-Spline wird

2 P.

T = [22, 24, _____]

(b) In der Zeichnung sind die ersten beiden Segmente von Q(t) skizziert (die Knoten sind als kleine Quadrate gezeichnet). Nennen Sie zwei Fehler, die in der Skizze gemacht wurden und verbessern Sie die Skizze entsprechend!

4 P.

1. _____

2. _____

6 P.

(c) Vervollständigen Sie die obige Skizze um die restlichen Kurvensegmente von Q(t), heben Sie dabei die Knoten hervor.

(d) Geben Sie eine Formel für die Berechnung von Q(25) an (setzen Sie in die Formel so weit wie möglich die aktuellen Zahlen ein – es genügt die Angabe der Formel, die Koordinaten von Q(25) müssen nicht ausgerechnet werden). Die Basismatrix für ein Kurvensegment einer uniformen, kubischen B-Spline Kurve sei M.

3 P.

Σ2:

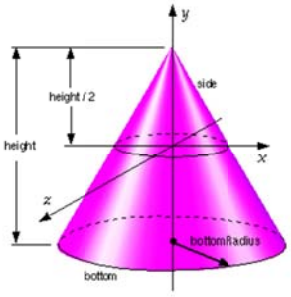
Aufgabe 2

Gegeben ist folgende VRML-Szene:

```
DEF T1 Transform{
  children [
    DEF S1 Shape{ geometry IndexedFaceSet{ coord Coordinate {
                                                    point [  0 0 0, # Punkt A
                                                             1 1 4, # Punkt B
                                                             1 0 0, # Punkt C
                                                           } # Coordinate
                                          coordIndex [ 2 0 1 -1 ]
                                } # IndexedFaceSet
    } # S1
    DEF T2 Transform{




      children[ Shape{ geometry Cone {height 8} }  ] } # T2
] } # T1
```



(a) Berechnen Sie die Normale auf die im IndexedFaceSet-Node definierte Fläche.

4 P.

(b) Ergänzen Sie Werte im Transform-Node T2 so, dass der Grundkreis des Kegels parallel zu der im IndexedFaceSet bestimmten Fläche und die Spitze des Kegels auf der positiven y-Achse liegt (Sie können dazu auch Ausdrücke verwenden, die trigonometrische Funktionen beinhalten).

4 P.

Σ3:

Aufgabe 3

Gegeben ist folgender Ausschnitt aus einem WebGL-Javascript, wobei die in der Lehrveranstaltung vorgestellten Hilfsfunktionen verwendet werden sowie die Konstante PI mit Wert π:
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 Skalierungsfaktoren (s_x,s_y,s_z)“*,
perspective(fov, aspect, near, far) *„erzeugt eine Projektionsmatrix“*

```
// Projektionsmatrix
var projection = perspective(90.0, 1.0, 0.5, 100.0);

// 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 =
```

(a) Ergänzen Sie das Programm nach Zeile A so, dass alle Modelle auf die gleiche Weise transformiert werden, wie es in folgendem VRML-Transform Node spezifiziert wird:

```
Transform{      scaleOrientation    1 2 3 4
                center              1 2 3
                scale               1 2 3
                rotation            2 3 4 5
                children [ # …
]}
```

5 P.

Σ4:

(b) Ergänzen Sie das Programm nach Zeile B so, dass die Kamera so orientiert und platziert wird, wie es in folgendem VRML-Viewpoint-Node spezifiziert wird (verwenden Sie dabei nur die oben angegebenen Hilfsfunktionen):

```
Viewpoint{      orientation     1 0 0 -1.57
                position        1 2 3
}
```

3 P.

(c) Mit welchem lookAt-Befehl würde man die selbe Kameraposition spezifizieren wie mit dem VRML-Viewpoint-Node aus Teilaufgabe (b)?

3 P.

lookAt(_____, _____, _____, _____, _____, _____, _____, _____, _____)

(d) Ein Punkt P hat die Koordinaten (1, 0, -2) in Viewkoordinaten. Berechnen Sie seine Koordinaten in Weltkoordinaten:

4 P.

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

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

2 P.

Σ5:

(g) Wie ändert sich das Bild, wenn perspective(90.0, 1.0, 0.5, 100.0) abgeändert wird in: perspective(100.0, 1.0, 0.5, 5.0); ?

4 P.

(h) Ergänzen Sie den unten stehenden GLSL Vertex-Shader und Fragment-Shader möglichst einfach, um eine 2D-Textur aufzubringen. Dabei soll die als attribute-Variable übergebene Farbe mit Gouraud-Shading auf das Objekt aufgebracht werden und anschließend im Verhältnis 2:1 mit der Texturfarbe gemischt werden. Führen Sie dazu auch neue benötigte uniform- und attribute-Variable ein.

```
void main(){  // Vertex-Shader
                uniform mat4 matrix;     //  Matrix zur Umrechnung von Objekt- in Clippingkoordinaten
                attribute vec4 color;      // die dem Vertex zugeordnete Farbe
                attribute vec4 position;  //  die dem Vertex zugeordnete Position
```

```
}

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

```
}
```

8 P.

Σ6:

Aufgabe 4

Gegeben ist ein Dreieck D mit den Koordinaten A(1, 2, 2), B(0, 0, 0) und C(1, 0, 0). Die Fläche liegt wie z.B. in VRML üblich im Umlaufsinn links. Am Punkt (5, 0, 1) befindet sich der Augpunkt der Kamera K. Alle Koordinaten sind in Weltkoordinaten angegeben.

(a) Eine Beleuchtungsrechnung ergibt, dass an Punkt B die diffuse Reflektion die Farbe (120°, ½, 1) und an Punkt C die Farbe (240°, ½, 1) vorliegt (Farbangaben sind im HLS-Farbsystem). Wie lautet die Farbe (im RGB-Farbsystem) am Punkt P(0,3; 0; 0), wenn Gouraud-Shading verwendet wird?

5 P.

(b) Berechnen Sie, ob der Punkt A durch Backface-Culling entfernt wird oder nicht.

4 P.

Σ7:

Aufgabe 5

(a) Nennen Sie zwei Beispiele für Per-Fragment-Ops bei Fragment-Shadern:

1. _____

2 P. 2. _____

(b) Worin liegt der Vorteil bei der Verwendung einer MipMap?

3 P.

(c) Wohin wird der Viewing-Reference-Point im Bild abgebildet?

2 P.

(d) Worin liegt der Vorteil bei der Verwendung von TriangleStrips?

2 P.

(e) 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.baba;
v.q = u.t;
```

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

Σ8: