Karlsruher Institut für Technologie

Institut für Betriebs- und Dialogsysteme
Lehrstuhl für Computergrafik
Prof. Dr.-Ing. Carsten Dachsbacher

Nachklausur Computergrafik WS 2012/13

8. April 2013

Kleben Sie hier **vor Bearbeitung der Klausur** den Aufkleber auf.

Beachten Sie:

- Trennen Sie vorsichtig die dreistellige Nummer von Ihrem Aufkleber ab. Sie sollten sie gut aufheben, um später Ihre Note zu erfahren.
- Die Klausur umfasst 25 Seiten (13 Blätter) mit 12 Aufgaben.
- Es sind **keine Hilfsmittel** zugelassen.
- Vor Beginn der Klausur haben Sie 5 Minuten Zeit zum *Lesen* der Aufgabenstellungen. Danach haben Sie **60 Minuten** Bearbeitungszeit.
- Schreiben Sie Ihren Namen und Ihre Matrikelnummer oben auf jedes bearbeitete Aufgabenblatt.
- Schreiben Sie Ihre Lösungen auf die Aufgabenblätter. Bei Bedarf können Sie weiteres Papier anfordern.
- Wenn Sie bei einer Multiple-Choice-Frage eine falsche Antwort angekreuzt haben und diesen Fehler korrigieren möchten, füllen Sie das betreffende Kästchen ganz aus:

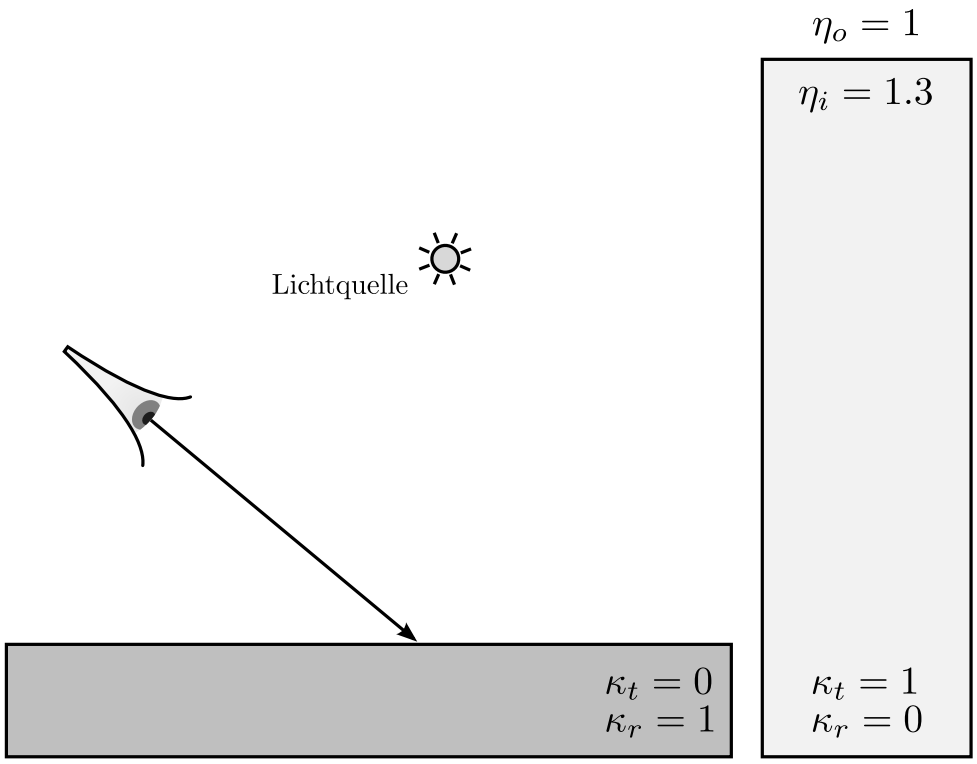
- Falsche Kreuze bei Multiple-Choice-Aufgaben führen zu Punktabzug. Jede Teilaufgabe wird mit mindestens 0 Punkten bewertet.

Aufgabe	1	2	3	4	5	6	7	8	9	10	11	12	**Gesamt**
Erreichte Punkte													
Erreichbare Punkte	6	13	8	12	15	5	10	10	10	5	16	10	**120**

Note

Aufgabe 1: Raytracing (6 Punkte)

a) Zeichnen Sie in der folgenden Grafik für den eingezeichneten Primärstrahl alle Sekundärstrahlen ein, die beim Whitted-Style-Raytracing erzeugt werden (auch die Strahlen, die keine in der Grafik sichtbaren Objekte treffen). Die dunklere Box soll, wie ein perfekter Spiegel, sämtliches Licht *reflektieren*. Die hellere Box soll, wie perfektes Glas, sämtliches Licht *transmittieren*. Der Brechungsindex der helleren Box ist mit η_i gegeben, die Brechzahl des umgebenden Mediums ist η_0. **(4 Punkte)**



Name: _____ Matrikelnummer: _____

b) Wie nennt man das physikalische Gesetz oder Prinzip, welches die Richtungsänderung eines Lichtstrahls beim Übergang in ein anderes Medium beschreibt? **(1 Punkt)**

Gesetz oder Prinzip	
Brewstersches Gesetz	☐
Snellsches Gesetz	☐
Fresnelsche Formeln	☐
Fermatsches Prinzip	☐

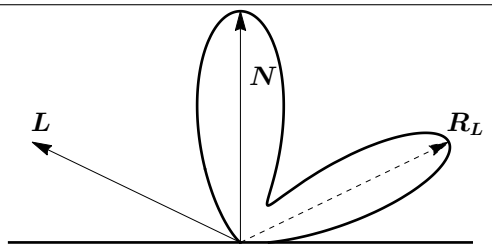
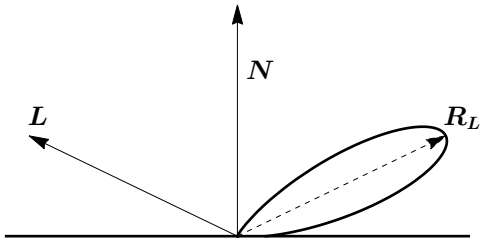
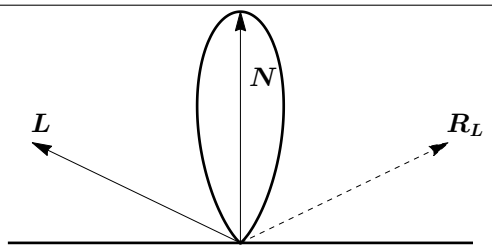
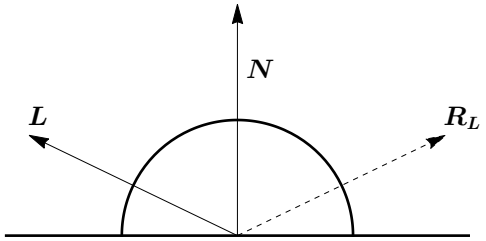
c) Welche Bedingung muss gelten, damit beim Übergang eines Lichtstrahls von einem Medium mit Refraktionsindex η_0 in ein Medium mit Refraktionsindex η_1 Totalreflexion auftreten kann? **(1 Punkt)**

Aufgabe 2: Beleuchtung und Wahrnehmung (13 Punkte)

In dieser Aufgabe geht es zunächst um das *Phong-Beleuchtungsmodell*, das Sie aus der Vorlesung kennen. Es beinhaltet eine ambiente, diffuse und spekulare Komponente.

a) Im Folgenden trifft ein Lichtstrahl aus der Richtung L auf eine Oberfläche mit der Normalen N. Die unten stehenden Polardiagramme zeigen, wie viel Licht in die jeweilige Richtung reflektiert wird (R_L ist die Spiegelung von L an N).

Geben Sie an, welche Diagramme durch das Phong-Beleuchtungsmodell entstehen können. Wenn dies der Fall ist, nennen Sie die abgebildete(n) Komponente(n). Andernfalls geben Sie eine Begründung an, warum das Phong-Beleuchtungsmodell das abgebildete Polardiagramm nicht erzeugen kann. **(8 Punkte)**

Name: _____ Matrikelnummer: _____

b) Bewerten Sie die folgenden Aussagen, indem Sie *Wahr* oder *Falsch* ankreuzen. **(5 Punkte)**

Aussage	**Wahr**	**Falsch**
Von den drei Grundfarben der additiven Farbmischung sind Menschen gegenüber blau in der Regel am *unempfindlichsten*.	☐	☐
Es gibt *keine* zwei unterschiedlichen Lichtspektren im sichtbaren Bereich, die der Mensch als dieselbe Farbe wahrnimmt.	☐	☐
Genau drei Grundgrößen reichen (nach Graßmann) aus, um einen menschlichen Farbeindruck zu beschreiben.	☐	☐
Es entspricht *nicht* der menschlichen Farbempfindlichkeit, wenn die Helligkeit (Luminanz) einer Farbe als das arithmetische Mittel der RGB-Anteile berechnet wird.	☐	☐
Gammakorrektur mit dem Parameter γ wird üblicherweise durch die Abbildung $L' = \gamma^L$ beschrieben.	☐	☐

Aufgabe 3: Transformationen (8 Punkte)

a) Welches Konzept verwendet man in der Computergrafik, um die Abbildung

$$f(\boldsymbol{x}) = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} \boldsymbol{x} + \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

als *lineare* Abbildung darzustellen? Geben Sie eine Matrix an, welche die Abbildung beschreibt. **(2 Punkte)**

b) Geben Sie zeichnerisch ein einfaches Beispiel an, das deutlich zeigt, dass man die Normalen eines Primitivs im Allgemeinen nicht mit derselben Matrix transformieren kann wie die Vertizes. Um was für eine Art Transformation handelt es sich dabei? **(2 Punkte)**

Name: _____ Matrikelnummer: _____

c) Kreuzen Sie die richtigen Aussagen an! Es können auch mehrere Aussagen zutreffen. Für jeden vollständig richtigen Block erhalten Sie einen Punkt. **(4 Punkte)**

Um Normalen korrekt von Objekt- in Kamerakoordinaten zu transformieren, verwendet man die	☐ inverstransponierte Model-View-Matrix. ☐ dehomogenisierte Model-View-Matrix. ☐ inverse Projektionsmatrix.
Die Matrix $\begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix}$ ist eine	☐ Translationsmatrix in homogenen 2D-Koordinaten. ☐ Translationsmatrix in homogenen 3D-Koordinaten. ☐ Rotationsmatrix in $\mathbb{R}^{3\times3}$.
Die Matrix $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ beschreibt eine	☐ Rotation. ☐ Spiegelung an der ersten Winkelhalbierenden. ☐ nichtlineare Abbildung.
Matrixmultiplikation ist stets kommutativ, wenn	☐ nur Skalierungsmatrizen multipliziert werden. ☐ nur Scherungsmatrizen multipliziert werden. ☐ nur Rotationsmatrizen multipliziert werden.

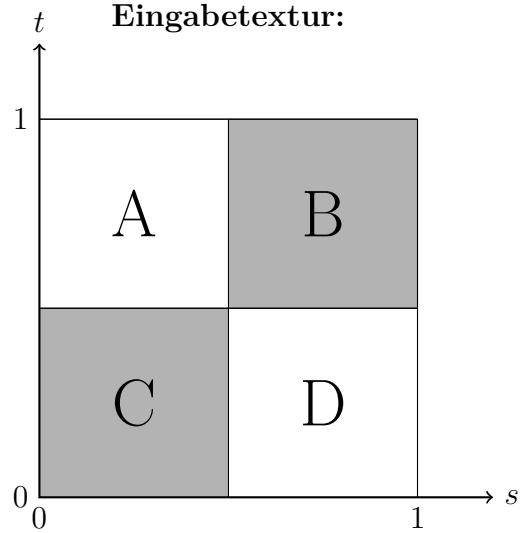
Aufgabe 4: Texturen und Texture-Mapping (12 Punkte)

a) Die folgenden Zeilen eines OpenGL-Programms zeichnen ein `GL_QUAD`-Primitiv, auf das eine Textur aufgebracht wird:

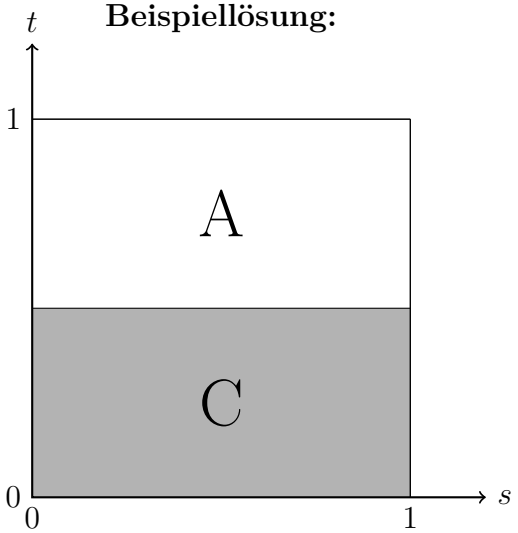
```
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);

glBegin(GL_QUADS);
    glTexCoord2f(-0.5, -0.5);
    glVertex2f(0.0, 0.0);

    glTexCoord2f(1.0, -0.5);
    glVertex2f(1.0, 0.0);

    glTexCoord2f(1.0, 1.5);
    glVertex2f(1.0, 1.0);

    glTexCoord2f(-0.5, 1.5);
    glVertex2f(0.0, 1.0);
glEnd();
```

Die Eingabetextur ist links auf dem folgenden Bild dargestellt, die Buchstaben dienen zum Identifizieren der Schachfelder. Skizzieren Sie die Vorderseite des `GL_QUAD`-Primitivs (inkl. der Buchstaben, analog zum Lösungsbeispiel), wie es durch obiges Programm gezeichnet wird! **(6 Punkte)**

Eingabetextur:

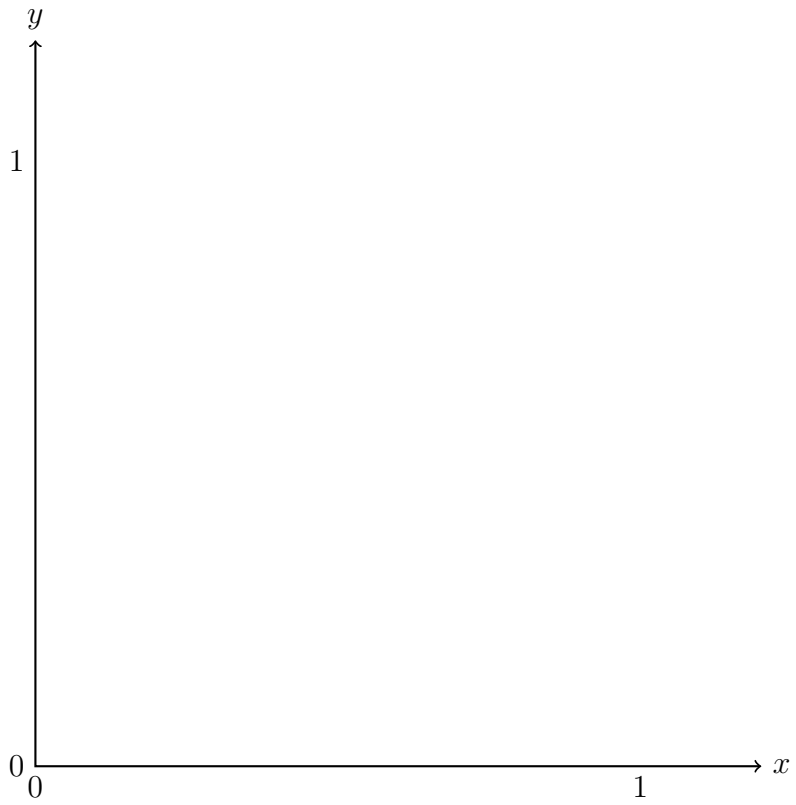


Beispiellösung:



Name: _____ Matrikelnummer: _____

Skizzieren Sie hier die Ausgabe:



Platz für Überlegungen/Zeichnungen (nicht bewertet):

b) *Wann* und *wofür* wird die bilineare Interpolation beim Texture-Mapping verwendet? **(2 Punkte)**

c) Was ist die Grundidee von vorgefilterten Environment Maps? Nennen Sie zwei Beispiele für Beleuchtungseffekte, die damit erzielt werden können! Welche grundlegende Annahme wird dabei gemacht? **(4 Punkte)**

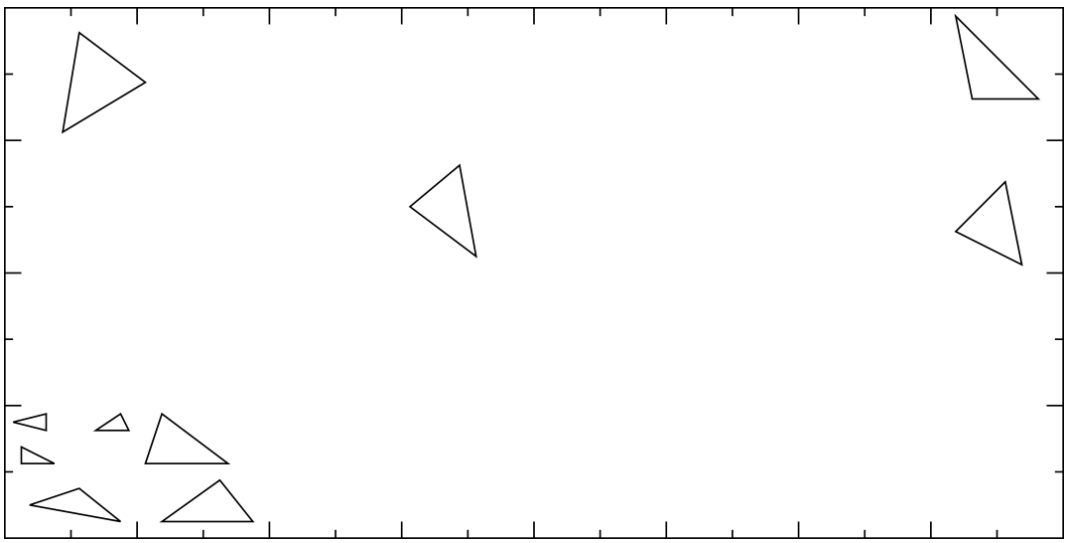
Name: _____ Matrikelnummer: _____

Aufgabe 5: Räumliche Datenstrukturen (15 Punkte)

a) In dieser Aufgabe sollen Sie die zutreffenden Aussagen über die räumlichen Datenstrukturen, wie Sie sie in der Vorlesung kennengelernt haben, identifizieren. Kreuzen Sie dazu jeweils die passende(n) Kästchen(n) an! Sie erhalten für jede vollständig richtige Zeile 1.5 Punkte. **(6 Punkte)**

Aussage	**BVH**	**Octree**	**kD-Baum**	**Gitter**
Die Raumunterteilung kann mit Hilfe der Surface Area Heuristic durchgeführt werden.	☐	☐	☐	☐
Die Anzahl der Schnitttests mit Primitiven kann durch Mailboxing weiter reduziert werden.	☐	☐	☐	☐
Die Raumunterteilung ist geschickt für Szenen mit einer Geometrie, die dem sogenannten Teapot-in-a-Stadium-Problem ähnelt.	☐	☐	☐	☐
Der Datenstruktur liegt ein Binärbaum zugrunde.	☐	☐	☐	☐

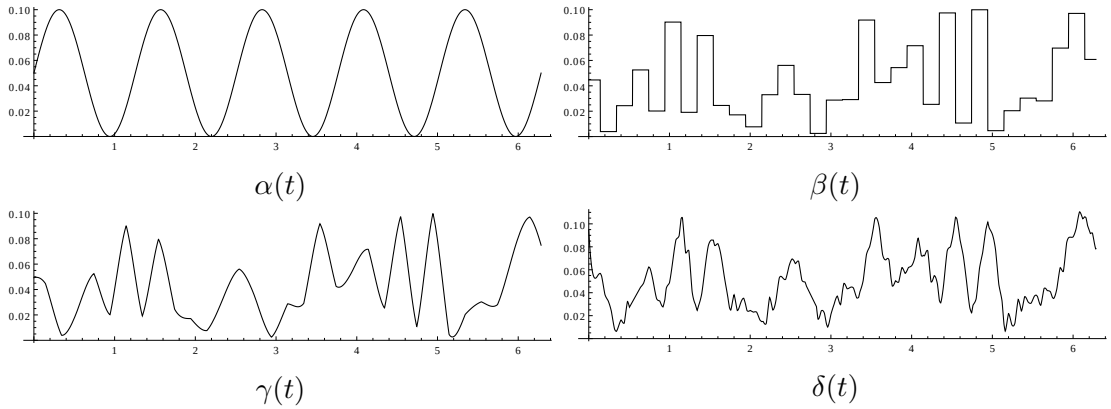
b) Konstruieren Sie, indem Sie Unterteilungslinien zeichnen, für die unten gezeichneten Primitive einen Quadtree mit einer regelmäßigen Unterteilung und maximal einem Primitiv pro Blatt: **(5 Punkte)**



c) Sie wollen in der obigen Szene Raytracing durchführen und dies mit einer räumlichen Datenstuktur beschleunigen. Ist dafür ein regelmäßiges Gitter oder ein kD-Baum besser geeignet? Begründen Sie Ihre Entscheidung! **(4 Punkte)**

Name: _____ Matrikelnummer: _____

Aufgabe 6: Rauschen und Turbulenz (5 Punkte)



$\alpha(t)$ $\beta(t)$

$\gamma(t)$ $\delta(t)$

In dieser Aufgabe sollen sie die oben abgebildeten Funktionen α, β, γ und δ einordnen. Benutzen Sie dabei die Eigenschaften von Rauschfunktionen, die Sie in der Vorlesung kennen gelernt haben.

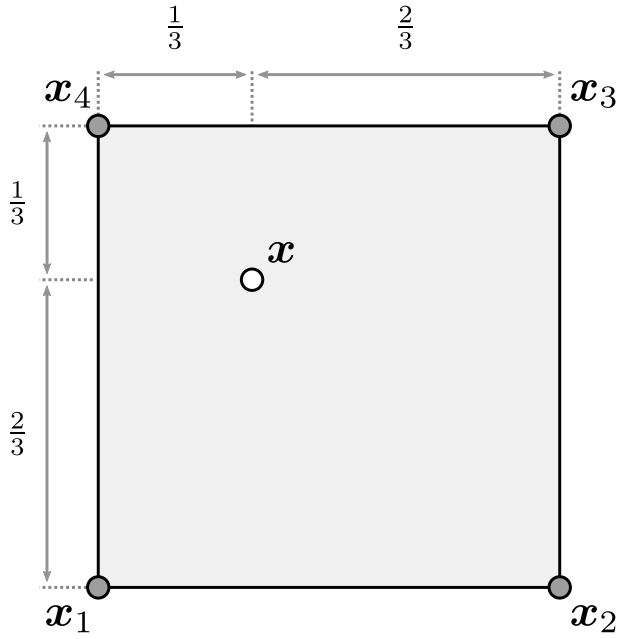
a) Welche der Funktionen ist eine Rauschfunktion (*value noise*)? **(1 Punkt)**

b) Welche der Funktionen ist eine Turbulenzfunktion? Wie kann man eine Turbulenzfunktion aus einer Rauschfunktion wie der aus Aufgabe a) erzeugen? Geben Sie eine Formel an! **(2 Punkte)**

c) Welche zwei der Funktionen sind weder Rausch- noch Turbulenzfunktionen? Nennen Sie jeweils eine Eigenschaft von Rauschfunktionen, die nicht erfüllt wird. **(2 Punkte)**

Aufgabe 7: Interpolation (10 Punkte)

In der Vorlesung haben Sie die Interpolation von Vertex-Attributen mit baryzentrischen Koordinaten bei Dreiecksnetzen kennen gelernt. Analog kann man Vertex-Attribute auch für Viereckснetze interpolieren.



In dieser Aufgabe sollen beispielhaft die Farbwerte $\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3$ und $\boldsymbol{c}_4$ für das abgebildete *Rechteck* interpoliert werden. Die Eckpunkte $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$ und $\boldsymbol{x}_4$ sind koplanar, liegen also in einer Ebene.

a) Wie berechnet man die baryzentrische Koordinate λ_2 des Punktes $\boldsymbol{x}$ bezüglich des abgebildeten Rechtecks? Analog zur Interpolation in Dreiecken soll λ_2 den Einfluss der Farbe $\boldsymbol{c}_2$ an der Stelle $\boldsymbol{x}$ beschreiben. Sie können davon ausgehen, dass $\boldsymbol{x}$ innerhalb des Rechtecks liegt.

   Verwenden Sie die Schreibweise $\square(A, B)$ für den Absolutwert des Inhalts der Rechtecksfläche, die durch die Eckpunkte A und B aufgespannt wird, zum Beispiel $\square(\boldsymbol{x}_1, \boldsymbol{x}_3)$ bzw. $\square(\boldsymbol{x}_3, \boldsymbol{x}_1)$ für die Fläche des gesamten Rechtecks. **(3 Punkte)**

b) Berechnen Sie anhand der in der Zeichnung angegeben Längen die 4 baryzentrischen Koordinaten des Punktes $\boldsymbol{x}$ mit der Formel aus Aufgabenteil a). Nutzen Sie dann die baryzentrischen Koordinaten, um anzugeben, wie der interpolierte Farbwert $\boldsymbol{c}$ am Punkt $\boldsymbol{x}$ berechnet wird. **(3 Punkte)**

c) Für die Rasterisierung mit modernem OpenGL zerlegt man Rechtecke üblicherweise in je zwei Dreiecke. Hierdurch können sich allerdings interpolierte Größen ändern.

Zeigen Sie dies, indem Sie annehmen, dass das Rechteck in die Dreiecke $\boldsymbol{x}_1\boldsymbol{x}_2\boldsymbol{x}_3$ und $\boldsymbol{x}_1\boldsymbol{x}_3\boldsymbol{x}_4$ zerlegt wurde. Berechnen Sie zunächst die baryzentrischen Koordinaten von Punkt $\boldsymbol{x}$ bezüglich des *Dreiecks* $\boldsymbol{x}_1\boldsymbol{x}_3\boldsymbol{x}_4$. Nutzen Sie diese dann, um anzugeben, wie der interpolierte Farbwert $\boldsymbol{c}'$ am Punkt $\boldsymbol{x}$ berechnet wird. **(4 Punkte)**

Aufgabe 8: OpenGL und OpenGL-Shader (10 Punkte)

Bewerten Sie die folgenden Aussagen, indem Sie *Wahr* oder *Falsch* ankreuzen. **(10 Punkte)**

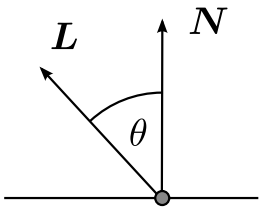
Aussage	**Wahr**	**Falsch**
Beleuchtung mit Schattenberechnung kann bei OpenGL mit Hilfe von `glEnable(GL_LIGHTING | GL_SHADOWS)` aktiviert werden.	☐	☐
Der Vertex-Cache funktioniert *nicht* mit indizierten Vertizes.	☐	☐
Im Fragment-Shader kann auf benachbarte Fragmente zugegriffen werden.	☐	☐
Im Fragment-Shader kann man die Tiefe eines Fragments verändern.	☐	☐
Die Fixed-Function-Pipeline berechnet Beleuchtung in normalisierten Gerätekoordinaten.	☐	☐
OpenGL führt Clipping in normalisierten Gerätekoordinaten durch.	☐	☐
Der Maler-Algorithmus benutzt einen Tiefenpuffer, um das Sichtbarkeitsproblem zu lösen.	☐	☐
Im Vertex-Shader kann man durch Erzeugung zusätzlicher Vertizes Primitive unterteilen.	☐	☐
Der Fragment-Shader ist in der OpenGL-Pipeline vor den Fragment-Operationen und nach dem Geometrie-Shader.	☐	☐
Beim Tiefenpuffer-Verfahren ist die Reihenfolge des Zeichnens auch bei opaken Primitiven wichtig für die Korrektheit.	☐	☐

Name: _____ Matrikelnummer: _____

Aufgabe 9: Toon-Shading in OpenGL (10 Punkte)

In dieser Aufgabe sollen Sie sogenanntes Toon-Shading in OpenGL implementieren. Ziel ist es, dadurch eine Darstellung im Comic-Stil zu erreichen. Dafür wird bei der Beleuchtungsberechnung der Winkel zwischen Lichtrichtung und Oberflächennormale diskretisiert. Sie können die Funktion `toonShade(int c)` nutzen, die für den `c`-ten Winkelbereich eine Farbe zurückliefert. Insgesamt soll es 3 Bereiche geben, die wie folgt begrenzt sind:

$$\begin{aligned} \texttt{c} = 0: &\quad \theta \in \left[0, \frac{\pi}{6}\right) \\ \texttt{c} = 1: &\quad \theta \in \left[\frac{\pi}{6}, \frac{\pi}{3}\right) \\ \texttt{c} = 2: &\quad \theta \in \left[\frac{\pi}{3}, \pi\right] \end{aligned}$$



a) Berechnen Sie im folgenden Vertex-Shader `gl_Position`. Transformieren Sie außerdem die Eingaben `position` und `normal` von Objekt- nach Weltkoordinaten und speichern Sie das Ergebnis in `posWorld` bzw. `normWorld`. Sie müssen dabei *nicht* dehomogenisieren. **(5 Punkte)**

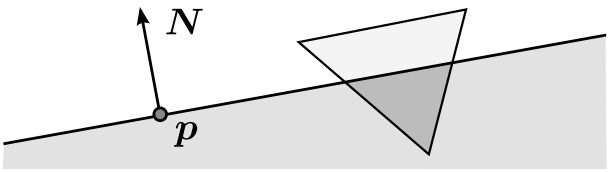
```
in vec4 position;   // Vertex-Position in Objektkoordinaten.
in vec4 normal;     // Vertex-Normale in Objektkoordinaten.
out vec3 posWorld;  // Vertex-Position in Weltkoordinaten.
out vec3 normWorld; // Vertex-Normale in Weltkoordinaten.
uniform mat4 M;     // Model-Matrix.
uniform mat4 V;     // View-Matrix.
uniform mat4 P;     // Projection-Matrix.

void main(void)
{









}
```

Name: _____ Matrikelnummer: _____

b) Berechnen Sie im folgenden Fragment-Shader den Richtungsvektor zur Lichtquelle. Berechnen Sie dann aus dem Winkel zwischen Lichtrichtung und Normale den passenden Winkelbereich `c` und nutzen Sie `toonShade(int c)`, um die Ausgabefarbe zu bestimmen. Speichern Sie diese in `color`. **(5 Punkte)**

```
out vec4 color;           // Ausgabefarbe.
in vec3 posWorld;         // Vertex-Position in Weltkoordinaten.
in vec3 normWorld;        // Vertex-Normale in Weltkoordinaten.
uniform vec3 lightWorld; // Position der Lichtquelle in Weltkoordinaten.
uniform float pi;         // Der Wert der Konstante Pi.

vec4 toonShade(int c) {
    ...
}

void main(void)
{









}
```

Aufgabe 10: Clipping in einem OpenGL-Shader (5 Punkte)



In dieser Aufgabe ist eine Ebene in *Kamerakoordinaten* durch ihre Normale $\boldsymbol{N}$ und einen Aufpunkt $\boldsymbol{p}$ gegeben. Sie sollen Clipping im Fragment-Shader implementieren, sodass Fragmente verworfen werden, die *hinter* der Ebene liegen. In der Abbildung sind das Fragmente im grauen Bereich.

a) Vervollständigen sie den Vertex-Shader, indem Sie `posCamera` und `gl_Position` schreiben. Sie müssen nach der Model-View-Transformation nicht dehomogenisieren. **(3 Punkte)**

```
in vec3 position;      // Die Position des Vertex in Objektkoordinaten.
out vec3 posCamera;    // Die Position des Vertex in Kamerakoordinaten.

uniform mat4 M;        // Die Model-Matrix.
uniform mat4 V;        // Die View-Matrix.
uniform mat4 P;        // Die Projection-Matrix.

void main() {









}
```

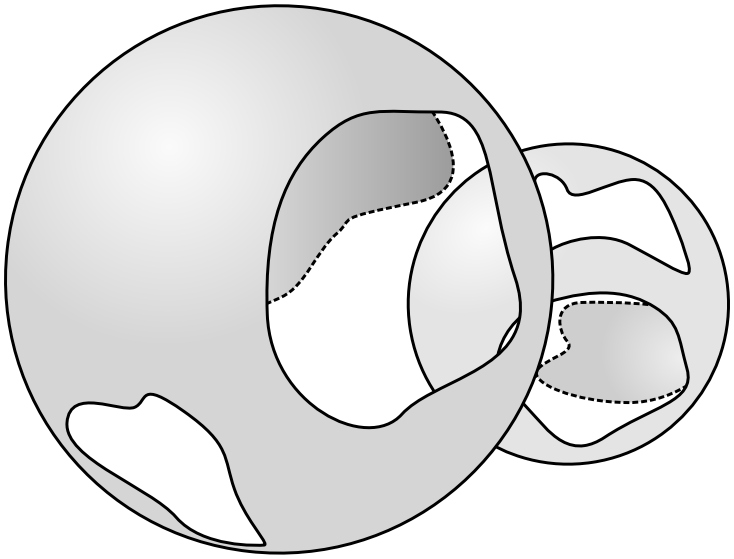
Name: _____ Matrikelnummer: _____

b) Vervollständigen Sie in diesem Fragment-Shader die Funktion `clip()`, welche entscheidet, ob ein Fragment verworfen werden soll. Die Variable `N` enthält die Normale der Ebene, während in `p` der Aufpunkt der Clipping-Ebene steht. Beide sind in Kamerakoordinaten gegeben. **(2 Punkte)**

```
in vec3 posCamera;  // Die Position des Vertex in Kamerakoordinaten.
out vec4 color;     // Die Ausgabefarbe.
uniform vec3 N;     // Die Normale der Clip-Ebene, in Kamerakoordinaten.
uniform vec3 p;     // Der Aufpunkt der Clip-Ebene, in Kamerakoordinaten.

bool clip() {









}

void main() {
    if (clip()) {
        discard;
    }
    color = vec4(1.0, 0.0, 0.0, 1.0);
}
```

Aufgabe 11: OpenGL und semitransparente Kugeln (16 Punkte)

In dieser Aufgabe sollen mit Hilfe von OpenGL *semitransparente* Kugeln *möglichst effizient* gezeichnet werden. Dabei soll ein nicht-kommutativer Blending-Operator verwendet und ein *möglichst korrektes Resultat* erzielt werden. Die Kugeln sind als Dreiecksnetze tesselliert. Es sollen sowohl die vom Betrachter abgewandten als auch die zum Betrachter zugewandten Primitive (hier: Dreiecke) einer Kugel gezeichnet werden. Sie können annehmen, dass sich die Kugeln nicht schneiden.



1. Ergänzen Sie das folgende OpenGL-Programmskelett, indem Sie die notwendigen Befehle oder deren Nummer eintragen. Der Blend-Operator ist bei Eintritt in den Programmteil schon konfiguriert; die anderen OpenGL-Zustände sind nicht definiert. **(8 Punkte)**

   Die folgenden OpenGL-Kommandos und Subroutinen stehen Ihnen zur Verfügung (von denen Sie nicht alle benötigen!):

   (1) `glDepthMask( GL_TRUE );`

   (2) `glDepthMask( GL_TRUE );`

   (3) `glEnable( GL_DEPTH_TEST );`

   (4) `glDisable( GL_DEPTH_TEST );`

   (5) `glEnable( GL_CULL_FACE );`

   (6) `glDisable( GL_CULL_FACE );`

   (7) `glCullFace( GL_BACK );`

   (8) `glCullFace( GL_FRONT );`

   (9) `glEnable( GL_BLEND );`

   (10) `glDisable( GL_BLEND );`

   (11) `sort( spheres );` sortiert alle Kugeln von hinten nach vorne.

   (12) `sortPrim( sphere );` sortiert Primitive einer Kugel von hinten nach vorne.

   (13) `draw( sphere );` zeichnet eine Kugel

Name: _____ Matrikelnummer: _____

```
void renderScene(std::vector<Sphere>& spheres)
{









    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );









    for (int i = 0; i < spheres.size(); ++i)
    {
        Sphere& sphere = spheres[i];









    }
}
```

2. Begründen Sie kurz warum Sie die Funktion `sort( spheres )` benutzt oder nicht benutzt haben. **(2 Punkte)**

3. Begründen Sie kurz warum Sie die Funktion `sortPrim( sphere )` benutzt oder nicht benutzt haben. **(2 Punkte)**

4. Bewerten Sie die folgenden Aussagen, indem Sie *Wahr* oder *Falsch* ankreuzen. **(4 Punkte)**

Aussage	**Wahr**	**Falsch**
Um eine Szene mit opaken und semitransparenten Objekten mit OpenGL möglichst korrekt zu zeichnen, muss man die opaken nach den transparenten Objekten zeichnen.	☐	☐
Der Alpha-Test erlaubt es, Fragmente anhand ihres Alpha-Wertes zu verwerfen, ohne den Tiefenpuffer zu modifizieren.	☐	☐
Alpha-Clipping ist eine Fragmentoperation und findet zwischen Alpha-Test und Alpha-Blending statt.	☐	☐
Der Blendingoperator definiert durch `glBlendEquation(GL_FUNC_ADD)` und `glBlendFunc(GL_SRC_ALPHA, GL_SRC_ALPHA)` ist kommutativ.	☐	☐

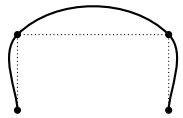
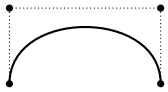
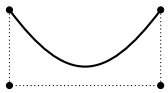
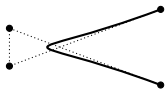
Name: _____ Matrikelnummer: _____

Aufgabe 12: Bézierkurven (10 Punkte)

a) Bewerten Sie die folgenden Aussagen, indem Sie *Wahr* oder *Falsch* ankreuzen. **(4 Punkte)**

Aussage	**Wahr**	**Falsch**
Bézierkurven liegen immer innerhalb der konvexen Hülle der Kontrollpunkte.	☐	☐
Bézierkurven interpolieren zwei Ihrer Kontrollpunkte und approximieren alle anderen.	☐	☐
Um eine Bézierkurve affin zu transformieren, genügt es, die Transformation auf ihre Kontrollpunkte anzuwenden.	☐	☐
Bézierkurven sind stetig differenzierbar.	☐	☐

b) Geben Sie an, ob es sich bei den folgenden Kurven mit gegebenem Kontrollpolygon um Bézierkurven handelt. Begründen Sie jeweils kurz Ihre Antwort, falls es sich *nicht* um eine Bézierkurve handelt! **(6 Punkte)**

Kurve	**Ja**	**Nein**	**Begründung**
	☐	☐	
	☐	☐	
	☐	☐	
	☐	☐	