Karlsruher Institut für Technologie

Institut für Visualisierung und Datenanalyse
Lehrstuhl für Computergrafik
Prof. Dr.-Ing. Carsten Dachsbacher

Nachklausur Computergrafik SS 2016

04. Oktober 2016

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 21 Seiten (11 Blätter) mit 10 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 Wahr-Falsch 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	**Gesamt**
Erreichte Punkte											
Erreichbare Punkte	7	11	19	6	10	9	10	16	23	9	**120**

Note

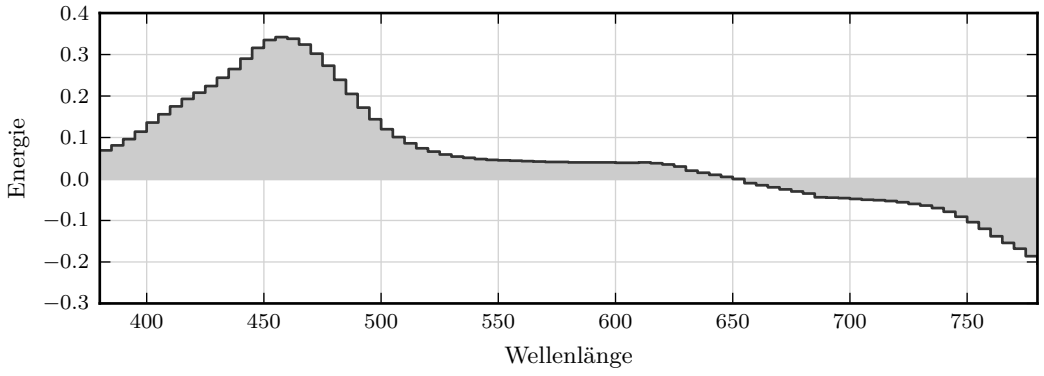
Aufgabe 1: Wahrnehmung und Farben (7 Punkte)

a) Was ist der Gamut eines Monitors? **(1 Punkt)**

b) Kreuzen Sie die Aussagen an, die auf die jeweiligen Farbräume zutreffen. Kreuzen Sie jeweils die passenden Kästchen an. Sie erhalten für jede vollständig richtige Zeile einen Punkt. **(5 Punkte)**

Aussage	**RGB**	**CMY**	**HSV**	**CIE xyY**
Der Farbraum ist additiv.	☐	☐	☐	☐
Der Farbraum ist subtraktiv.	☐	☐	☐	☐
Der Farbraum trennt Helligkeit von Farbe.	☐	☐	☐	☐
Der Farbraum kann alle für den Menschen sichtbaren Farben repräsentieren.	☐	☐	☐	☐
Der Farbraum wird nativ auf Peripheriegeräten verwendet.	☐	☐	☐	☐

c) Kann das folgende Spektrum von einer real existierenden Lampe stammen? Begründen Sie in Stichpunkten! **(1 Punkt)**



Name: _____ Matrikelnummer: _____

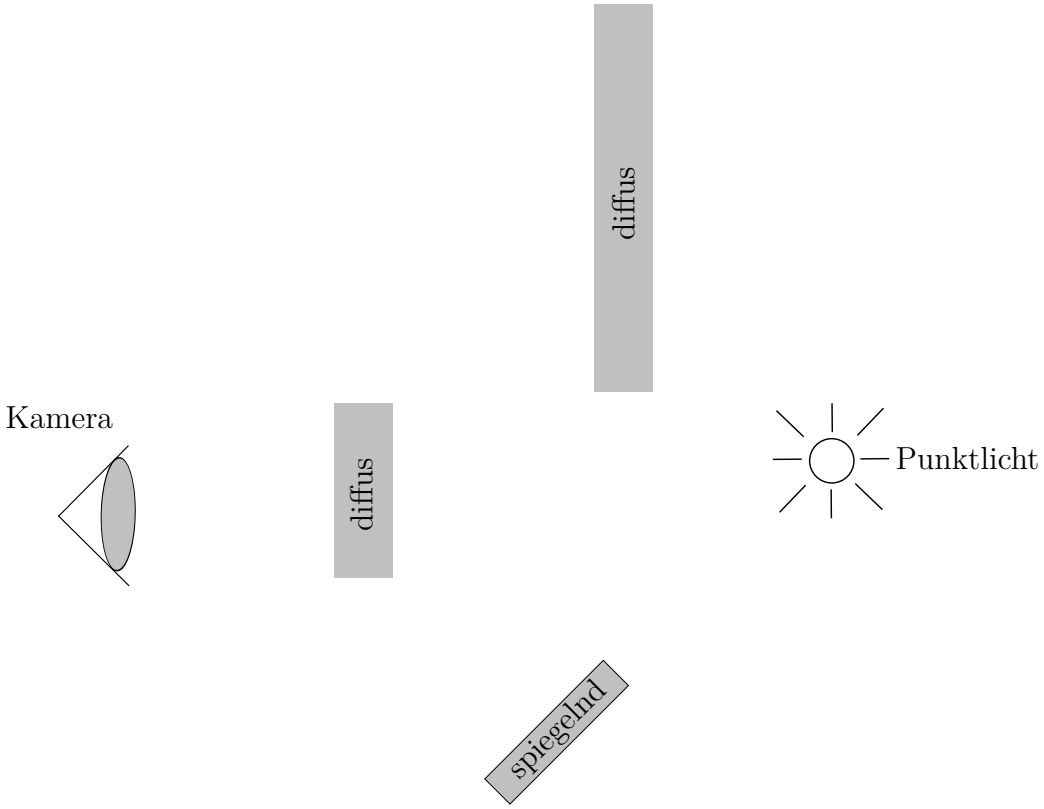
Aufgabe 2: Raytracing (11 Punkte)

a) Die folgende Szene besteht aus einem Spiegel, zwei Objekten mit diffuser Oberfläche, einer Punktlichtquelle und einer perspektivischen Kamera. Der Spiegel *reflektiert* sämtliches Licht perfekt.

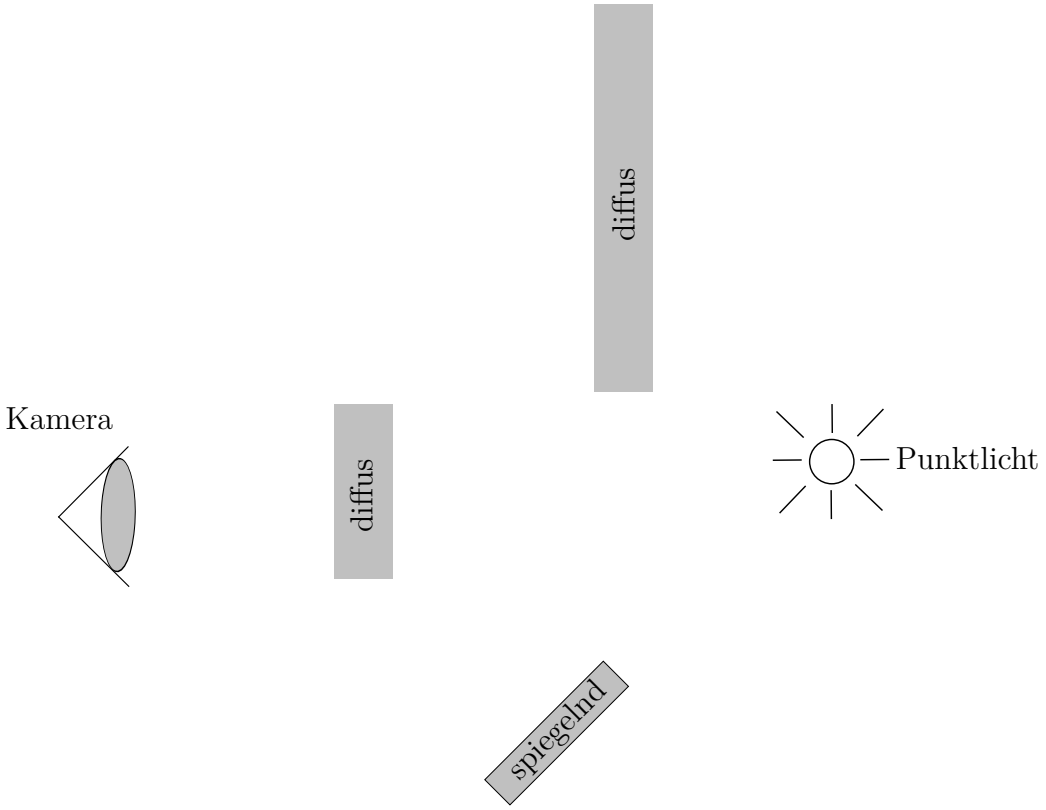
Zeichnen Sie einen Lichtpfad ein, der beim Whitted-style Raytracing erzeugt wird und die Kamera mit dem Punktlicht verbindet! **(3 Punkte)**

Kennzeichnen Sie außerdem die einzelnen Pfadsegmente als

- Primärstrahl (**P**),
- Schattenstrahl (**S**),
- Reflexionsstrahl (**R**),
- Transmissionsstrahl (**T**)!



b) Zeichnen Sie nun einen Pfad ein, der in der Realität Licht in die Kamera bringen würde, aber von einem Whitted-style Raytracer nicht erzeugt werden kann! Begründen Sie in Stichworten, warum Whitted-style Raytracing diesen Pfad nicht erzeugen kann! **(2 Punkte)**



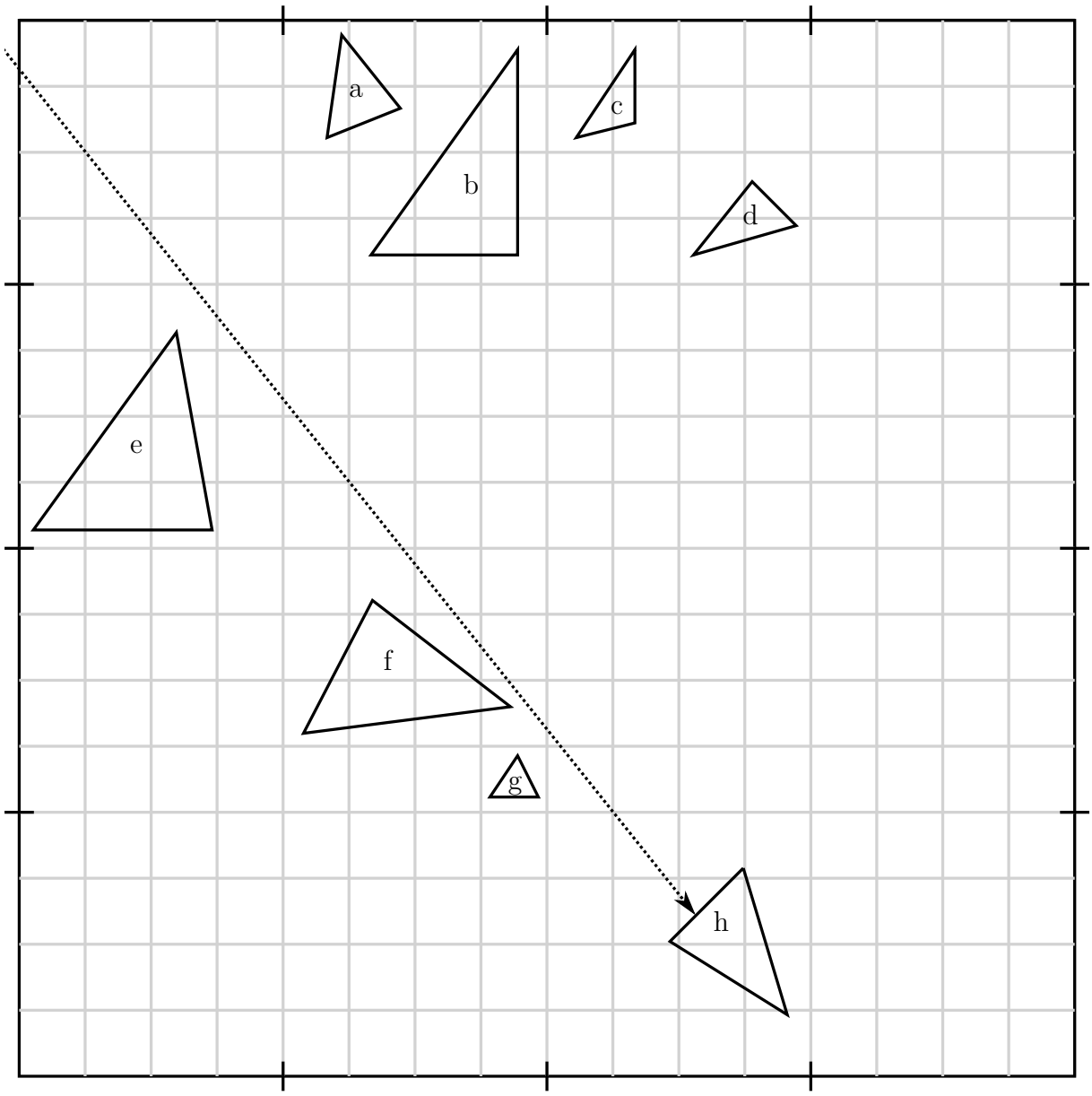
Name: _____ Matrikelnummer: _____

c) Bewerten Sie die folgenden Aussagen, indem Sie *Wahr* oder *Falsch* ankreuzen. **(6 Punkte)** ☐

Aussage	**Wahr**	**Falsch**
Wenn ein Lichtstrahl aus einem optisch dichteren Medium in ein optisch dünneres Medium übergeht, kann Totalreflexion auftreten.	☐	☐
Die Zeitkomplexität von Whitted-style Raytracing ist polynomial in der Rekursionstiefe.	☐	☐
Es gibt eine Klasse von Materialien, die beim Whitted-style Raytracing mehrere Sekundärstrahlen per Schnittpunkt erzeugen.	☐	☐
Der Whitted-style Raytracer überprüft die direkte Beleuchtung eines Oberflächenpunktes mithilfe von Schattenstrahlen.	☐	☐
Whitted-style Raytracing konvergiert immer zum physikalisch korrekten Ergebnis.	☐	☐
Whitted-style Raytracing ist das Bildsyntheseverfahren, das auf der GPU in Hardware implementiert ist.	☐	☐

Aufgabe 3: Beschleunigungsstrukturen (19 Punkte)

Gegeben sind ein Strahl und eine Menge von Dreiecken (a, b, ..., h).



a) Sie sollen in der Abbildung zeichnerisch einen Quadtree erstellen, *in dem Primitive nur in Blattknoten gespeichert werden.* Unterteilen Sie so lange, bis jeder Blattknoten maximal *ein* Dreieck enthält. **(5 Punkte)**

b) Annotieren Sie in der Abbildung aus a) die Dreiecke mit der Anzahl der Blattknoten in denen sie enthalten sind. **(2 Punkte)**

Name: _____ Matrikelnummer: _____

c) Der eingezeichnete Strahl soll mit dem Quadtree aus Aufgabe a) geschnitten werden. Geben Sie *alle* durchgeführten Schnittests als Liste der entsprechenden Dreiecke an! Ordnen Sie die Liste nach der Reihenfolge der ausgeführten Schnittests, wenn der Quadtree ohne Mailboxing traversiert wird. **(2 Punkte)** ☐

d) Die folgenden Aussagen beziehen sich auf die Datenstrukturen, wie sie in der Vorlesung behandelt wurden. BVH ist eine Hüllkörperhierarchie mit achsenparallelen Quadern. Das Gitter ist regulär.

Kreuzen Sie jeweils die Datenstruktur an, für die die Aussage zutrifft! Wenn die Aussage für keine der Datenstrukturen gilt, kreuzen Sie **Keine** an. Sie erhalten für jede vollständig richtige Zeile einen Punkt. Nicht vollständig richtige Zeilen werden mit Null Punkten bewertet. **(4 Punkte)** ☐

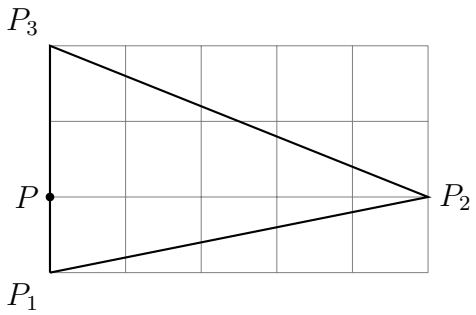
Aussage	**BVH**	**kD-Baum**	**Gitter**	**Keine**
Primitive können in mehr als einem Blattknoten/einer Gitterzelle vorhanden sein.	☐	☐	☐	☐
Die Datenstruktur kann zur Beschleunigung von Nachbarschaftssuchen verwendet werden.	☐	☐	☐	☐
Die Datenstruktur eignet sich besonders für Szenen, in denen die Geometrie gleichmäßig verteilt ist.	☐	☐	☐	☐
Der Aufbau-Algorithmus passt die Datenstruktur an die Normalen der Geometrie an.	☐	☐	☐	☐

☐ e) Bewerten Sie die folgenden Aussagen, indem Sie *Wahr* oder *Falsch* ankreuzen. **(6 Punkte)**

Aussage	**Wahr**	**Falsch**
Die Surface Area Heuristic minimiert die Anzahl der Primitive, die sich in der Beschleunigungsstruktur befinden.	☐	☐
Der Aufbau einer BVH mithilfe der Surface Area Heuristic ist im Allgemeinen aufwändiger als der Aufbau mittels Objektmittel-Methode (object median).	☐	☐
BSP-Bäume sind kD-Bäume mit achsparallelen Unterteilungsebenen.	☐	☐
Die Objektmittel-Methode (object median) kann beim Erstellen von BVH und kD-Baum verwendet werden.	☐	☐
Die Raummittel-Methode (spatial median) kann beim Erstellen von BVH und kD-Baum verwendet werden.	☐	☐
Eine optimale objektorientierte Box (OOBBs) hat höchstens das Volumen der entsprechenden achsenparallelen Box (AABB).	☐	☐

Name: _____ Matrikelnummer: _____

Aufgabe 4: Baryzentrische Koordinaten und Shading (6 Punkte)



a) Für ein Dreieck seien die Eckpunkte P_1, P_2, P_3 mit Farben c_1, c_2, c_3 gegeben. Ein Punkt P innerhalb des Dreiecks habe die baryzentrischen Koordinaten λ_1, λ_2, λ_3. Geben Sie eine Formel zur Berechnung der interpolierten Farbe c im Punkt P an. **(2 Punkte)**

$c =$

b) Was müssen Sie beachten, wenn Sie auf diese Weise Normalen interpolieren, um die Beleuchtung für Punkte in einem Dreieck zu berechnen? **(1 Punkt)**

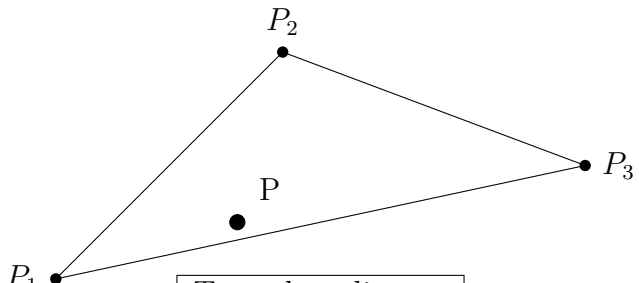
c) In der obigen Abbildung sind die Koordinaten der Eckpunkte $P_1 = (0, 0)$, $P_2 = (5, 1)$ und $P_3 = (0, 3)$. Bestimmen Sie die baryzentrischen Koordinaten des Punktes $P = (0, 1)$. **(2 Punkte)**

- $\lambda_1 =$
- $\lambda_2 =$
- $\lambda_3 =$

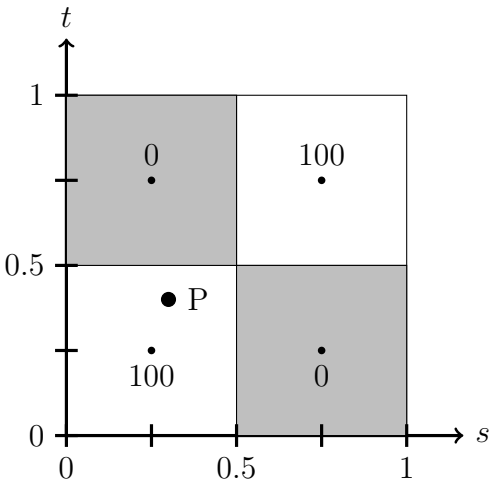
d) In der Vorlesung haben Sie Phong-Shading, Gouraud-Shading und Flat-Shading kennengelernt. Für welche Verfahren können Sie das Phong Beleuchtungsmodell einsetzen? **(1 Punkt)**

Aufgabe 5: Texturen (10 Punkte)

Es soll ein texturiertes Dreieck gezeichnet werden. Die Eckpunkte des Dreiecks P_1, P_2, P_3 sind mit Texturkoordinaten versehen. Neben dem Dreieck ist die zu verwendende 2×2 Grauwerttextur mit Schachbrettmuster abgebildet. Dort sind auch jeweils die Grauwerte und Koordinaten der Texelmitten angegeben. Die Farbe des Randtexels wird bei Zugriffen außerhalb [0,1] fortgesetzt (`CLAMP_TO_EDGE`).



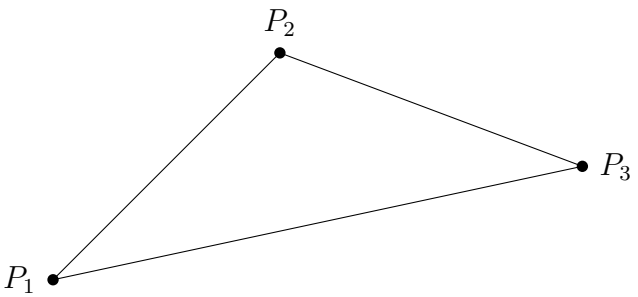
Texturkoordinaten	
P_1	(0,0)
P_2	(0,1)
P_3	(1,1)



a) Die Texturkoordinate am Punkt P sei $T_p = (0.3, 0.4)$. Verwenden Sie diese, um die oben abgebildete Textur auszulesen! Bestimmen Sie den Farbwert $c_{BL}(T_p)$ an dieser Stelle durch bilineare Interpolation! **(2 Punkte)**

$c_{BL}(T_p) =$

b) Skizzieren Sie auf der folgenden Abbildung, wie das texturierte Dreieck mit der obigen Textur bei Verwendung der *Nearest-Neighbor-Interpolation* aussehen würde! **(2 Punkte)**



Name: _____ Matrikelnummer: _____

c) Es soll nun Mip-Mapping eingesetzt werden, dazu wird die Mip-Map-Pyramide der obigen Textur aufgebaut.

i) Berechnen Sie alle Texelwerte der fehlenden Mip-Map-Stufen. Geben Sie die durchgeführten Rechenschritte an! **(1 Punkt)** ☐

ii) Führen Sie den Zugriff aus Aufgabenteil a) erneut aus, diesmal mit Mip-Mapping und trilinearer Interpolation! Welchen Wert ergibt der Texturzugriff nun? Geben Sie die durchgeführten Rechenschritte an! *Gehen Sie hierbei davon aus, dass der Footprint exakt in der Mitte zwischen zwei Stufen liegt.* **(1 Punkt)** ☐

$c_{MM}(T_p) =$

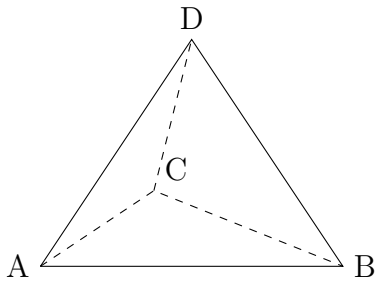
iii) Bei der Texturfilterung unterscheidet man zwei Fälle, je nachdem wie viele Texel auf ein Pixel abgebildet werden. Benennen Sie diese beiden Fälle in der untenstehenden Tabelle! **(2 Punkte)** ☐

Kreuzen Sie außerdem für beide Fälle an, welche der genannten Texturfilterungstechniken die höchste Qualität bietet, ohne unnötigen Zusatzaufwand hervorzurufen! **(2 Punkte)** ☐

Fall	**Nearest**	**Bilinear**	**Mip-Map Nearest**	**Mip-Map Trilinear**
	☐	☐	☐	☐
	☐	☐	☐	☐

Aufgabe 6: OpenGL-Pipeline (9 Punkte)

Ein Tetraeder mit den Stützpunkten A,B,C,D soll mit modernem OpenGL und Vertex- bzw. Indexpuffern gezeichnet werden. Das Dreieck ABC soll **nicht** gezeichnet werden.



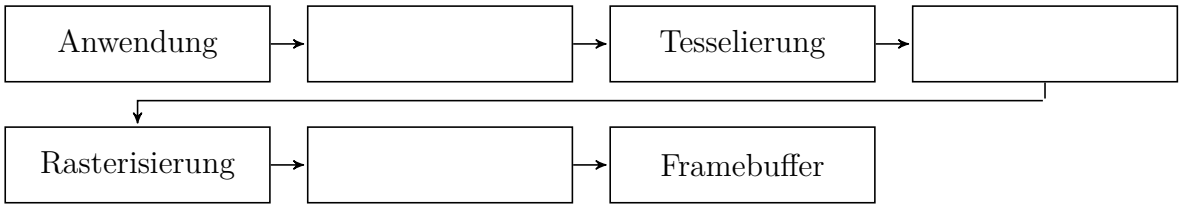
a) In der folgenden Tabelle sind verschiedene Kombinationen von Primitivtyp, Vertex- und Indexpufferinhalt aufgelistet. Vervollständigen Sie die Tabelle für das gegebene Tetraeder und fangen Sie dabei wenn möglich mit dem Vertex A an!

Das Tetraeder wird wie in der Abbildung von außen betrachtet. Der Drehsinn (winding order) ist *gegen den Uhrzeigersinn*.

Pro richtig ausgefüllter Zeile gibt es 2 Punkte. **(6 Punkte)**

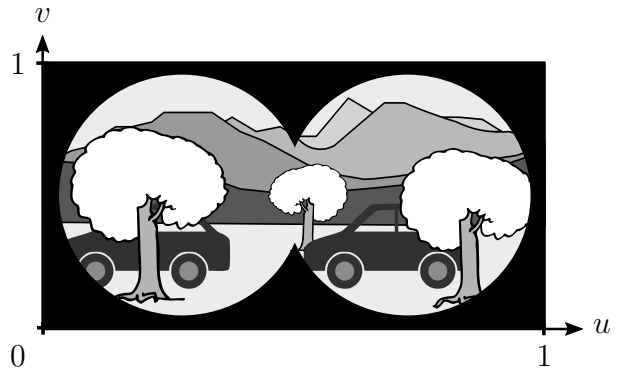
Primitivtyp	Vertexpufferinhalt	Indexpufferinhalt
`GL_TRIANGLES`	A,B,C,D	
`GL_TRIANGLE_STRIP`	A,B,C,D	
`GL_TRIANGLE_FAN`	A,B,C,D	

b) Da modernes OpenGL verwendet werden soll, kommen Shader zum Einsatz. Vervollständigen Sie die OpenGL-Pipeline, indem Sie die fehlenden Namen der Shader in die entsprechenden leeren Felder schreiben. **(3 Punkte)**



Name: _____ Matrikelnummer: _____

Aufgabe 7: Clipping im OpenGL-Fragment-Shader (10 Punkte)



a) Es soll der Eindruck erweckt werden, dass eine Szene durch ein Fernglas betrachtet wird. Dafür sollen in einem Fragment-Shader alle Fragmente außerhalb zweier gegebener Kreisscheiben *verworfen* werden. Andernfalls wird die gegebene Fragment-Farbe weitergereicht.

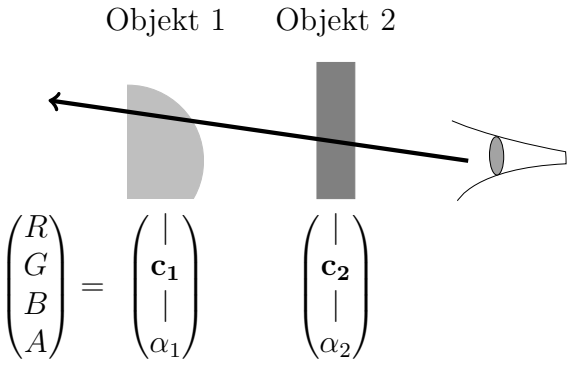
Impementieren Sie diesen Fragment-Shader! **(6 Punkte)**

```
uniform int W;    // Breite des Bildes in Pixeln
uniform int H;    // Höhe des Bildes in Pixeln
uniform vec2 C0; // Zentrum der 1. Kreisscheibe in Pixeln in [0,W) × [0,H)
uniform vec2 C1; // Zentrum der 2. Kreisscheibe in Pixeln in [0,W) × [0,H)
uniform float R; // Radius der Kreisscheiben in Pixeln

in vec2 uv; // interpolierte Texturkoordinaten des Fragments in [0,1) × [0,1)

in vec3 inColor;    // Eingabefarbe des Fragments
out vec3 outColor; // Ausgabefarbe des Fragments

void main()
{




}
```

b) Wie kann man den Effekt aus Aufgabenteil a) mit der Fixed-Function-Pipeline von OpenGL, also ohne Verwendung von Shadern, realisieren? Erklären Sie eine Möglichkeit stichpunktartig! **(4 Punkte)**

Name: _____ Matrikelnummer: _____

Aufgabe 8: OpenGL Blending (16 Punkte)



Eine Szene mit opaken und transparenten Objekten soll möglichst korrekt und effizient gerendert werden. Das Blending wurde in OpenGL wie folgt initialisiert:

```
glClearColor(1.0, 1.0, 1.0, 1.0);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
glEnable(GL_BLEND);
glBlendEquation(GL_FUNC_ADD);
glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
```

a) Zeigen Sie, dass im Allgemeinen die resultierende Pixelfarbe davon abhängt, in welcher Reihenfolge die Objekte gezeichnet werden. Geben Sie dazu eine Formel für die Pixelfarbe $\mathbf{c}$ an für den Fall, dass Objekt 1 zuerst gezeichnet wird, sowie eine Formel für die Pixelfarbe $\mathbf{c}'$ für den Fall, dass Objekt 2 zuerst gezeichnet wird! Zählen Sie auf, in welchen Fällen $\mathbf{c} = \mathbf{c}'$ gilt! **(8 Punkte)**

b) Eine Szene mit opaken und semitransparenten Dreiecken soll *möglichst effizient* mit einem nicht-kommutativen Blending-Operator gezeichnet werden. Vervollständigen Sie dazu die nachfolgende Funktion mit den notwendigen OpenGL-Kommandos und Subroutinen (oder deren Nummern) aus der folgenden Liste:

(1) `glDepthMask( GL_TRUE );`

(2) `glDepthMask( GL_FALSE );`

(3) `glEnable( GL_DEPTH_TEST );`

(4) `glDisable( GL_DEPTH_TEST );`

(5) `glEnable( GL_BLEND );`

(6) `glDisable( GL_BLEND );`

(7) `sortTransBackToFront();` sortiert alle transparenten Dreiecke von hinten nach vorne.

(8) `sortTransFrontToBack();` sortiert alle transparenten Dreiecke von vorne nach hinten.

(9) `drawTrans();` zeichnet alle transparenten Dreiecke.

(10) `sortOpaqueBackToFront();` sortiert alle opaken Dreiecke von hinten nach vorne.

(11) `sortOpaqueFrontToBack();` sortiert alle opaken Dreiecke von vorne nach hinten.

(12) `drawOpaque();` zeichnet alle opaken Dreiecke..

Gehen Sie davon aus, dass der Zustand des Tiefentests, der Tiefenmaske und des Blendings undefiniert ist, und sich der Rest der OpenGL Pipeline im richtigen Zustand befindet.

☐ Vervollständigen Sie die Funktion auf der nächsten Seite! **(6 Punkte)**

Name: _____ Matrikelnummer: _____

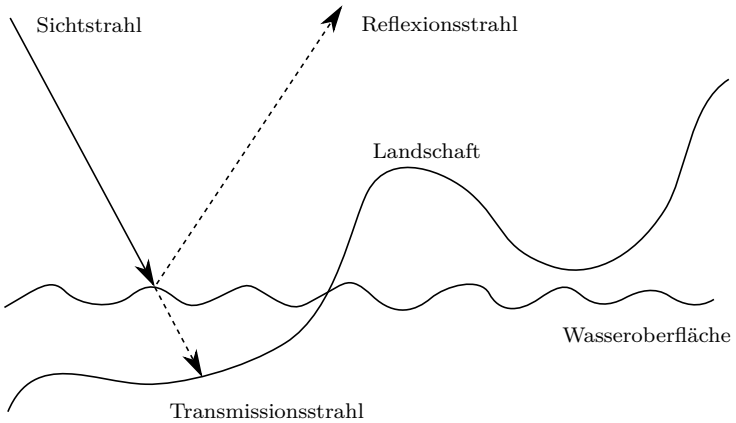
```
void renderScene()
{
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
    // Initialisieren Sie hier den OpenGL-Zustand!









    // Zeichnen Sie die Szene ab hier!









}
```

c) Nennen Sie ein Beispiel für die Verwendung eines nicht-kommutativen Blending-Operators! **(1 Punkt)** ☐

d) Nennen Sie ein Beispiel für die Verwendung eines kommutativen Blending-Operators! **(1 Punkt)** ☐

Aufgabe 9: OpenGL - Sphere Tracing in Distanzfeldern (23 Punkte)

In dieser Aufgabe soll mittels Sphere Tracing eine durch Distanzfelder gegebene Landschaft dargestellt werden. Die zwei Funktionen `distTerrain` und `distWater` berechnen zu einem gegebenen Punkt jeweils den Abstand zur nächstgelegenen Oberfläche einer Landschaft und einer Wasseroberfläche.



a) Implementieren Sie **auf der nächsten Seite** die Funktion `intersect`, welche den nächstgelegenen Schnittpunkt eines gegebenen Strahls mit der Landschafts- oder Wassergeometrie, sowie das zugehörige Objekt bestimmt! **(13 Punkte)**

Name: _____ Matrikelnummer: _____

```
#define TERRAIN 1   // Index des Terrain-Objekts
#define WATER 2     // Index des Wasser-Objekts
#define NOTHING 0   // Kein Objekt

// Berechnet Abstand zwischen p und nächstgelegener Oberfläche ...
float distTerrain(vec3 P) {...}     // ... der Landschaft
float distWater(vec3 P) {...}       // ... der Wasseroberfläche

uniform float tmin; // Minimale Schnittdistanz, ab der gesucht wird
uniform float tmax; // Maximale Schnittdistanz, bis zu der gesucht wird
uniform float eps;  // Oberflächendistanz, ab der Objekt als geschnitten gilt

// Berechne Abstand des nächstgelegenen Schnittes mit Terrain oder Wasser
float intersect(
  in vec3 O,        // Strahlursprung
  in vec3 d,        // Normierte Strahlrichtung
  out int oidx)     // Geschnittenes Objekts (NOTHING, TERRAIN oder WATER)
{
    float t = tmin;









    // Kein Schnittpunkt
    oidx = NOTHING;
    return tmax;
}
```

b) Verwenden Sie nun die Funktion `intersect` aus Aufgabe a), um die Farbe eines Pixels des Ausgabebildes zu berechnen! Berechnen Sie dazu von der Wasseroberfläche je einen Reflexions- und einen Transmissionsstrahl! Der Transmissionsstrahl wird nicht gebrochen. Sie können davon ausgehen, dass Transmissionsstrahlen immer die Landschaft und Reflexionsstrahlen nur den Himmel treffen. **(10 Punkte)**

```
// Berechnet Oberflächennormale der Distanzfelder
vec3 normalAt(vec3 P) {...}

// Berechnet Farbe ...
vec3 skyColor(vec3 viewDir) {...}   // ... des Himmels
vec3 terrainColor(                  // ... der Landschaft
  vec3 P,                             // - Position
  vec3 normal) {...}                  // - Normale
vec3 waterColor(                    // ... des Wassers
  vec3 normal,                        // - Normale
  vec3 viewDir,                       // - eingehende Strahlrichtung
  vec3 reflectionColor,               // - Farbe des Reflexionsstrahls
  vec3 transmissionColor) {...}       // - Farbe des Refraktionsstrahls


// Berechne Schnittfarbe
vec3 computeColor(
  in vec3 O,        // Strahlursprung
  in vec3 d)        // Strahlrichtung
{









}
```

Name: _____ Matrikelnummer: _____

Aufgabe 10: Basiswechsel und Bézierkurven (9 Punkte)

a) Gegeben ist eine quadratische Bézierkurve mit Kontrollpunkten $\{\boldsymbol{b}_0, \boldsymbol{b}_1, \boldsymbol{b}_2\} \subset \mathbb{R}^n$. Bestimmen Sie die Koeffizienten $\{\boldsymbol{a}_0, \boldsymbol{a}_1, \boldsymbol{a}_2\} \subset \mathbb{R}^n$ der Kurve bezüglich der Monombasis $\{1, u, u^2\}$! **(3 Punkte)**

Die Bernsteinpolynome sind $B_0^2(u) = (1-u)^2, B_1^2(u) = 2u(1-u)$ und $B_2^2(u) = u^2$.

b) Gegeben sei die kubische Bézierkurve $\boldsymbol{F}(u) = \sum_{i=0}^{3} \boldsymbol{b}_i B_i^3(u)$ mit Kontrollpunkten $\boldsymbol{b}_i$.

I) Werten Sie die Bézierkurve zeichnerisch an der Stelle $u = \frac{1}{2}$ mit dem de Casteljau-Algorithmus aus! Zeichnen Sie alle Punkte der Konstruktion, inklusive dem endgültigen Kurvenpunkt, ein! **(4 Punkte)**

II) Skizzieren Sie den Verlauf der Bézierkurve! **(2 Punkte)**