

Hauptklausur Computergrafik

WS 2013/14

12. März 2014

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 22 Seiten (11 Blätter) mit 11 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	Gesamt
Erreichte Punkte												
Erreichbare Punkte	11	8	10	16	4	17	7	6	6	17	18	120

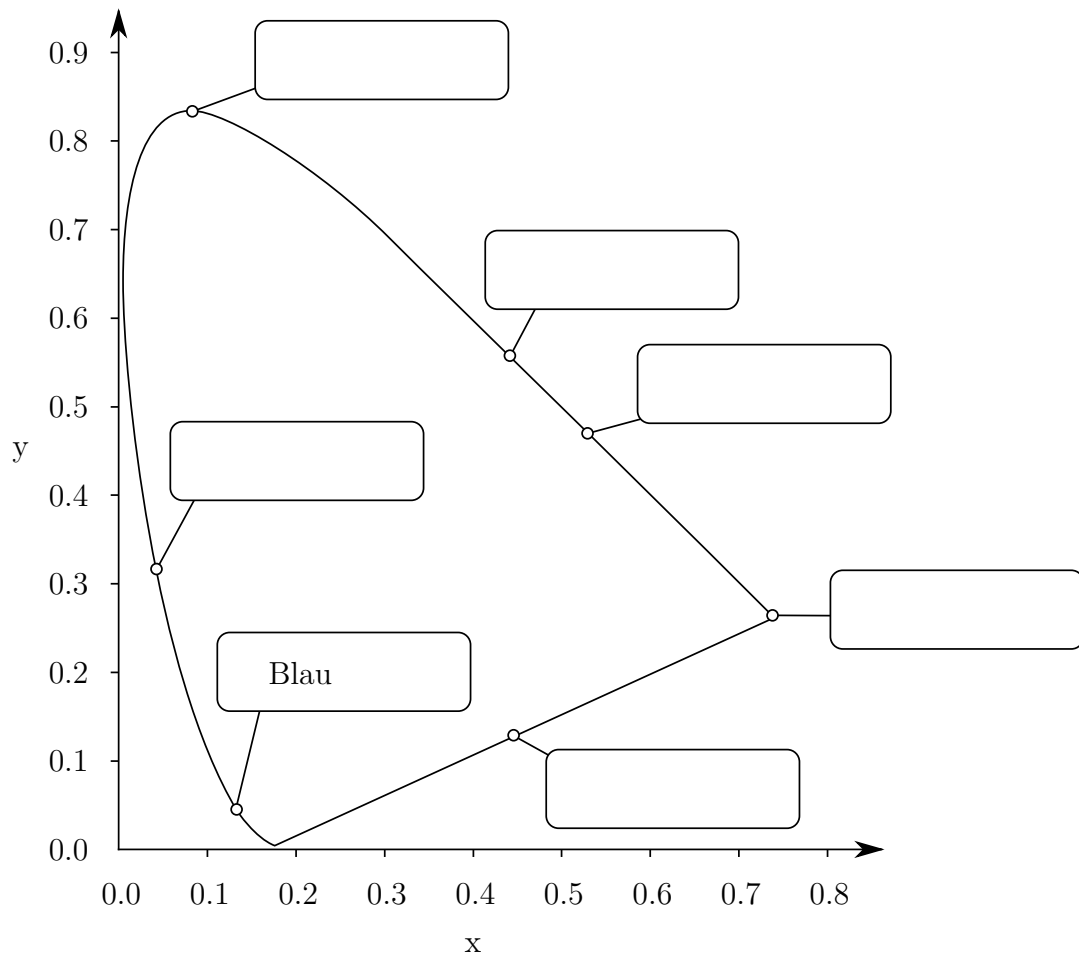
Note

Aufgabe 1: Farben und Farbwahrnehmung (11 Punkte)

a) Tragen Sie die Farben

Grün, Rot, Gelb, Orange, Cyan, Magenta

in die entsprechenden Felder im Chromatizitätsdiagramm ein. **(3 Punkte)**



b) Welcher der Farbeindrücke aus Aufgabe a) lässt sich nicht durch monochromatisches Licht erzeugen? **(1 Punkt)**

Name: _____

Matrikelnummer: _____

- c) Die Farbräume xyY und XYZ sind eng verwandt. Wie ist der mathematische Zusammenhang zwischen der Chromatizität (x, y) und der passenden Farbe (X, Y, Z) ? **(2 Punkte)**

$x =$

$y =$

- d) Gegeben sind

- 1) der Farbraum **XYZ**,
- 2) ein physikalisch realisierbarer **RGB**-Farbraum,
- 3) sowie der Raum aller Farben, die durch **100 monochromatische Leuchtdioden** mit den äquidistanten Wellenlängen $\{380\text{nm}, 384\text{nm}, \dots, 776\text{nm}\}$ darstellbar sind.

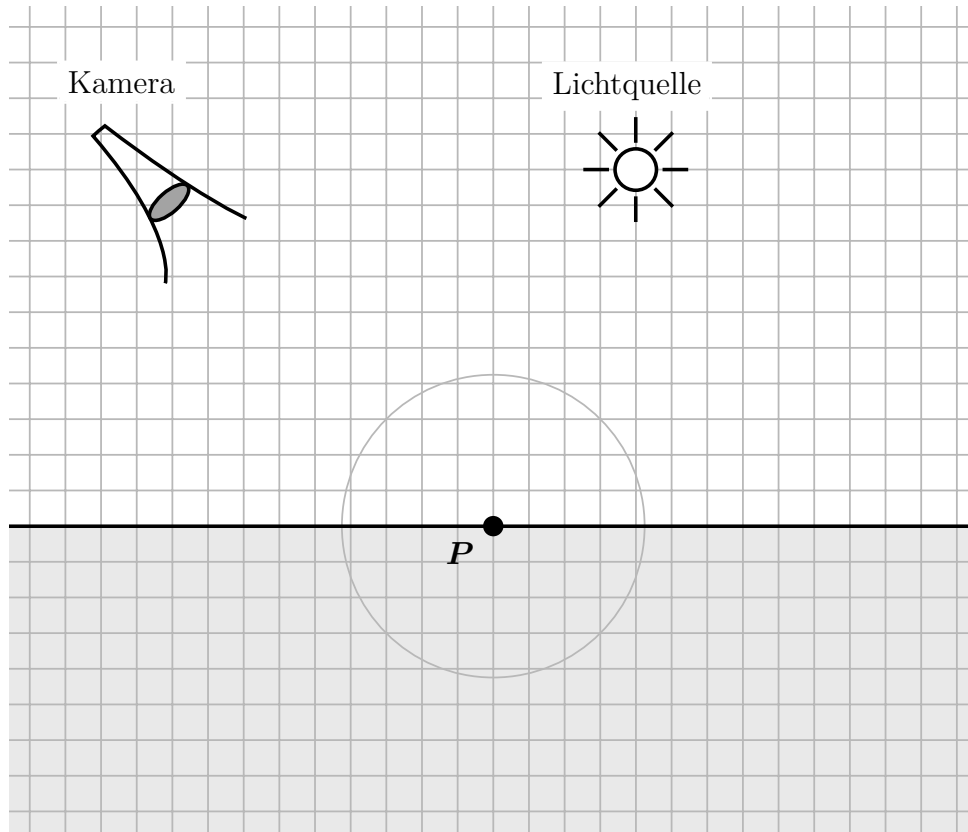
Ordnen Sie diese Räume aufsteigend nach der Größe ihres für den Menschen sichtbaren Gamut. **(2 Punkte)**

- e) Bewerten Sie die folgenden Aussagen, indem Sie *Wahr* oder *Falsch* ankreuzen. **(3 Punkte)**

Aussage	Wahr	Falsch
Den Weißpunkt eines Farbraums bezeichnet man auch als Tristimuluswert.	<input type="checkbox"/>	<input type="checkbox"/>
Die subjektiv empfundene Stärke von Sinneseindrücken ist proportional zum Logarithmus ihrer Intensität.	<input type="checkbox"/>	<input type="checkbox"/>
Jeder Farbeindruck für den Menschen kann mit drei Grundgrößen beschrieben werden.	<input type="checkbox"/>	<input type="checkbox"/>

Aufgabe 2: Whitted-Style Raytracing (8 Punkte)

Diese Skizze zeigt eine Kamera, einen Punkt P auf einer Oberfläche mit Brechungsindex $n_2 = 1.5$ und eine Lichtquelle. Kamera und Licht befinden sich im Vakuum (Brechungsindex $n_1 = 1$). Zur besseren Orientierung ist ein Gitter hinterlegt, sowie ein Kreis mit Radius 1 skizziert, der Ihnen beim Zeichnen hilft.



Ergänzen Sie die Skizze, indem Sie die folgenden Vektoren und Winkel einzeichnen und beschriften:

- Den Augstrahl \mathbf{v} zur Kamera und den Lichtstrahl \mathbf{l} (1 Punkt)
- Den Normalenvektor \mathbf{n} bei P (1 Punkt)
- Den Reflexionsstrahl \mathbf{r} , der auch für rekursives Raytracing verwendet wird (1 Punkt)
- Den Winkel α , der benötigt wird, um den spekularen Anteil des Phong-Modells zu berechnen (1 Punkt)
- Zeichnen Sie qualitativ die Richtung des transmittierten Strahls \mathbf{t} ein (1 Punkt).

Name: _____

Matrikelnummer: _____

f) Geben Sie die Formel für den *spekularen Anteil* des Phong-Modells unter Verwendung von α an! Wie wird α aus den eingezeichneten Vektoren berechnet? **(2 Punkte)**

g) Wie ist der Name des Gesetzes, welches zur Konstruktion des transmittierten Strahls aus Teilaufgabe e) benutzt werden muss? **(1 Punkt)**

Aufgabe 3: Transformationen (10 Punkte)

Im Folgenden sind homogene Transformationsmatrizen in 2D gegeben. Skizzieren Sie die jeweiligen Auswirkungen der Transformationen auf das Rechteck in das Koordinatensystem in der mittleren Spalte. Beschreiben Sie jeweils in der rechten Spalte *kurz* die Transformation.

Eine Transformationsmatrix M wird dabei durch $\mathbf{P}' = M\mathbf{P}$ auf einen Punkt \mathbf{P} angewendet. (10 Punkte)

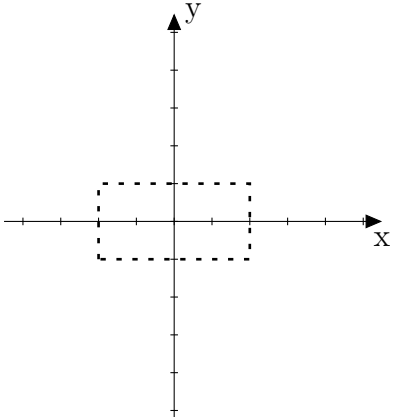
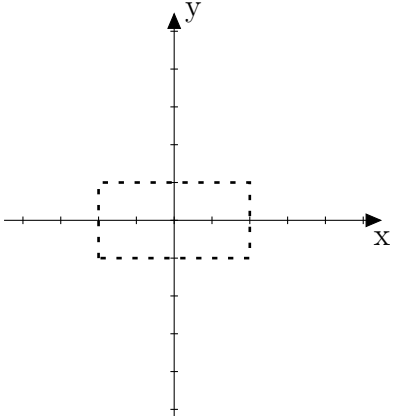
Hinweis:

Die erste Tabellenzeile gibt Ihnen ein Beispiel, wie die Tabelle ausgefüllt werden soll.

Transformation	Koordinatensystem	Beschreibung
$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{pmatrix}$		<p>Translation um 1 in x und 3 in y-Richtung.</p>
$\begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$		
$\begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$ <p>mit $\theta = \frac{\pi}{4}$</p>		

Name: _____

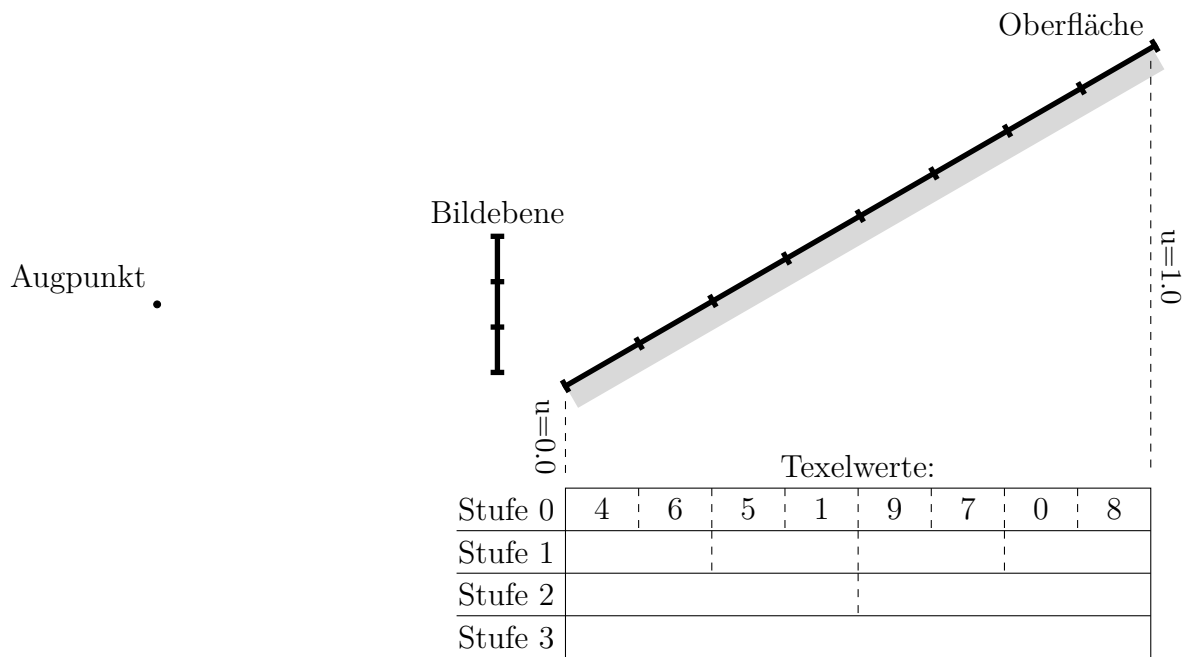
Matrikelnummer: _____

Transformation	Koordinatensystem	Beschreibung
$\begin{pmatrix} \frac{1}{2} & 0 & 4 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{pmatrix}$		
$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 3 \\ 0 & 0 & 1 \end{pmatrix}$		

Aufgabe 4: Mipmapping und Texturen (16 Punkte)

Wir betrachten Texturierung mit Mipmapping, hier am Beispiel einer eindimensionalen Textur mit Mipmaps, in einer zweidimensionalen Welt.

In der Abbildung sind eine Oberfläche, die Bildebene und der Augpunkt einer perspektivischen Projektion eingezeichnet. Auf die Oberfläche wird eine Textur gelegt, die die Oberfläche vollständig bedeckt. Für das Texturemapping wird Mipmapping verwendet. Die Texel der höchsten Auflösungsstufe (Stufe 0) dieser Mipmap sind in der Abbildung durch Unterteilungen gekennzeichnet und die jeweiligen Texel-Werte befinden sich in der Tabelle darunter.



- Berechnen Sie die Texelwerte aller weiteren Mipmap-Stufen und tragen Sie diese in die obige Tabelle ein! **(3 Punkte)**
- Ermitteln Sie – *zeichnerisch in der obigen Abbildung* – die Größe des Footprints der drei eingezeichneten Bildpixel in der Textur! **(2 Punkte)**

Footprint des Pixels oben

mitte

unten

Name: _____

Matrikelnummer: _____

- c) Es soll nun ein Zugriff auf diese Textur mit der Texturkoordinate $u = \frac{3}{8}$ erfolgen. In der Abbildung oben sind die 2 Texturkoordinaten $u = 0$ und $u = 1$ eingezeichnet, an denen Sie sich orientieren können. Gehen Sie für diesen Zugriff von einem Textur-Footprint von $f = 3$ Texel (gemessen in Stufe 0) aus.

Sie sollen nun das Ergebnis dieses Mipmap-Zugriffs berechnen:

- I) Aus welchen Mipmap-Stufen werden bei diesem Zugriff Daten benötigt? **(2 Punkte)**

- II) Innerhalb jeder Mipmap-Stufe wird linear interpoliert. Führen Sie diese Interpolationen mit den Texel-Werten in der Tabelle aus und geben Sie jeweils den Rechenweg und das Ergebnis an! **(3 Punkte)**

Gehen Sie bei der Interpolation davon aus, dass die Texel-Werte für die Mitte der Texel gültig sind.

- III) Zuletzt soll das Endergebnis t des Mipmap-Zugriffs berechnet werden, wenn ein Verfahren analog zur trilinearen Filterung bei 2D-Texturen verwendet wird.

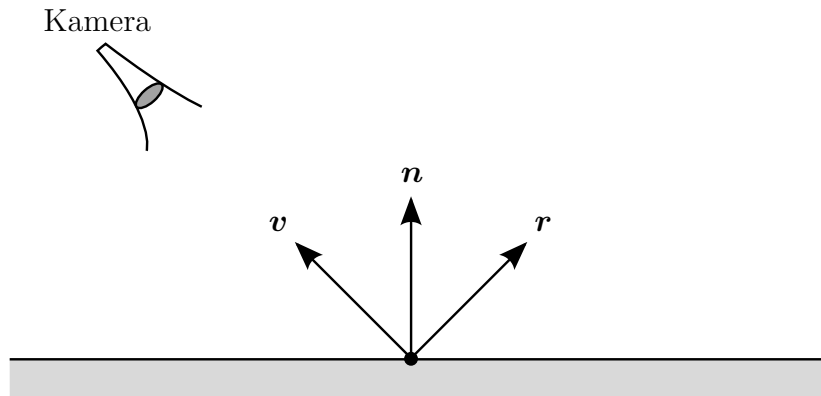
Geben Sie eine Formel zur Bestimmung der Gewichte der beteiligten Stufen an! Geben Sie nun mit Hilfe dieser Gewichte eine Formel zur Bestimmung des Endergebnisses t an! **(3 Punkte)**

d) Bewerten Sie die folgenden Aussagen, indem Sie *Wahr* oder *Falsch* ankreuzen. (3 Punkte)

Aussage	Wahr	Falsch
Texturkoordinaten müssen sich immer im Intervall $[0; 1]$ befinden.	<input type="checkbox"/>	<input type="checkbox"/>
Texturkoordinaten können als Attribute der Eckpunkte (Vertices) übergeben werden und werden als solche interpoliert.	<input type="checkbox"/>	<input type="checkbox"/>
Texturkoordinaten müssen für die Darstellung wie Eckpunktkoordinaten der Model-View-Transformation unterzogen werden.	<input type="checkbox"/>	<input type="checkbox"/>

Aufgabe 5: Vorgefilterte Environment-Maps (4 Punkte)

Es soll eine Oberfläche mit vorgefilterten Environment-Maps schattiert werden. Dafür sind eine vorgefilterte Environment-Map für diffuse Beleuchtung und eine vorgefilterte Environment-Map für imperfekte Spiegelung gegeben. In der folgenden Abbildung ist ein sichtbarer Oberflächenpunkt illustriert.



a) Welche Richtungsvektoren werden für den Zugriff in die jeweiligen Environment-Maps verwendet? (2 Punkte)

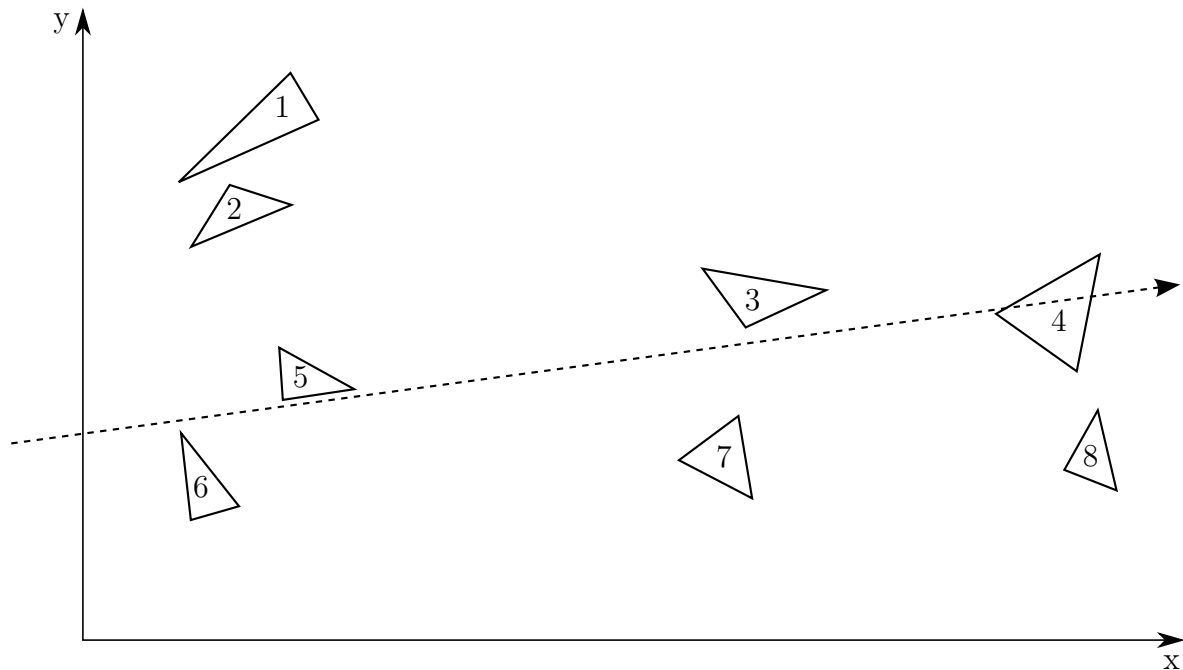
- Für die Environment-Map für diffuse Beleuchtung:

- Für die Environment-Map für imperfekte Spiegelung:

b) Geben Sie eine Berechnungsvorschrift für den Reflexionsvektor r in Abhängigkeit von v und n an (die Vektoren sind normiert). (2 Punkte)

Aufgabe 6: Hierarchische Datenstrukturen (17 Punkte)

Gegeben sei die folgende Menge von Dreiecken, sowie der eingezeichnete Strahl:



- a) Zeichnen Sie die Hüllkörper für eine Hüllkörperhierarchie mit achsenparallelen Boxen (Bounding Volume Hierarchy, BVH) auf dem Aufgabenblatt ein!

Verwenden Sie dazu die **Objektmittel-Methode (object median)** entlang der maximalen Ausdehnung in Achsenrichtung des umschließenden Hüllkörpers. Teilen Sie die Teilbäume so oft auf, bis nur noch maximal **zwei Primitive** in jedem Blatt vorhanden sind. (3 Punkte)

- b) Geben Sie die Reihenfolge der Dreiecke an, die bei der Traversierung der Hüllkörperhierarchie für den eingezeichneten Strahl auf Schnitt getestet werden! (3 Punkte)

Name: _____

Matrikelnummer: _____

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

Aussage	Wahr	Falsch
Beim Traversieren eines kD-Baums müssen immer beide Kinder in Betracht gezogen werden.	<input type="checkbox"/>	<input type="checkbox"/>
Das Traversieren einer Hüllkörperhierarchie mit achsenparallelen Boxen (Bounding Volume Hierarchy, BVH) erfordert Mailboxing, um mehrfache Schnitttests mit einem Dreieck zu verhindern.	<input type="checkbox"/>	<input type="checkbox"/>
Der Speicheraufwand einer BVH hängt logarithmisch von der Anzahl der Primitive ab.	<input type="checkbox"/>	<input type="checkbox"/>
kD-Bäume sind eine Verallgemeinerung von BSP-Bäumen.	<input type="checkbox"/>	<input type="checkbox"/>
BSP-Bäume sind adaptiv und leiden nicht unter dem „Teapot in a Stadium“-Problem.	<input type="checkbox"/>	<input type="checkbox"/>

d) Kreuzen Sie jeweils die passenden Kästchen an! Sie erhalten für jede vollständig richtige Zeile 1.5 Punkte. (6 Punkte)

Aussage	Hüllkörperhierarchie (BVH)	Oktalbaum (Octree)	kD-Baum	Gitter
Die Datenstruktur partitioniert den Raum.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Der Aufwand für den Aufbau der Datenstruktur ist linear in der Anzahl der Primitive.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eine effizientere Traversierung wird erreicht, wenn die Surface Area Heuristic bei der Konstruktion verwendet wird.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Die Datenstruktur eignet sich am besten für Szenen, in denen die Geometrie gleichmäßig verteilt ist und kaum leere Zwischenräume vorhanden sind.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Aufgabe 7: Rasterisierung und OpenGL (7 Punkte)

- a) Bewerten Sie die folgenden Aussagen über eine moderne OpenGL-Pipeline, indem Sie *Wahr* oder *Falsch* ankreuzen. (4 Punkte)

Aussage	Wahr	Falsch
In der OpenGL-Pipeline wird View Frustum Clipping <i>vor</i> der perspektivischen Division durchgeführt.	<input type="checkbox"/>	<input type="checkbox"/>
Vertex-Shader können auf Texturen zugreifen.	<input type="checkbox"/>	<input type="checkbox"/>
Bei Gouraud-Shading muss man die Normale im Fragment-Shader erneut normalisieren.	<input type="checkbox"/>	<input type="checkbox"/>
Gouraud-Shading mit dem Phong-Beleuchtungsmodell kann im Geometry-Shader implementiert werden.	<input type="checkbox"/>	<input type="checkbox"/>
Phong-Shading kann man alleine mit einem Vertex-Shader und einem Geometry-Shader implementieren; letzterer gibt dann die Farbe aus.	<input type="checkbox"/>	<input type="checkbox"/>
Bei beliebig feiner Tessellierung ist kein Unterschied zwischen Gouraud- und Phong-Shading erkennbar.	<input type="checkbox"/>	<input type="checkbox"/>
Selbst wenn der Tiefentest für ein Fragment fehlschlägt, kann der Stencil-Puffer verändert werden.	<input type="checkbox"/>	<input type="checkbox"/>
Instanziierung von Geometrie kann man sowohl mit dem Vertex- als auch dem Geometry-Shader durchführen.	<input type="checkbox"/>	<input type="checkbox"/>

- b) Warum zieht man das Tiefenpuffer-Verfahren (Z-Buffering) dem Sortieren von Dreiecken vor? Nennen Sie drei Gründe! (3 Punkte)

Aufgabe 8: OpenGL-Primitive (6 Punkte)

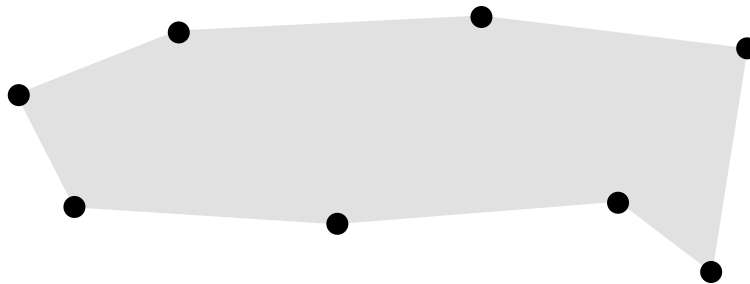
Verbinden Sie die eingezeichneten Vertizes zu OpenGL-Primitiven, so dass die schattierten Flächen gezeichnet werden.

Benutzen Sie dabei das jeweils effizienteste OpenGL-Primitiv! Schreiben Sie an jeden Vertex auch seine Übergabereihenfolge an OpenGL (beginnen Sie die Nummerierung mit 1)! Nennen Sie außerdem den Namen des verwendeten OpenGL-Primitivs!

Beachten Sie bei der Reihenfolge der Eckpunkte, dass Backface-Culling mit den Standardeinstellungen aktiviert ist: Dreiecke werden nur gezeichnet, wenn Ihre Eckpunkte im Bild *gegen den Uhrzeigersinn* angeordnet sind.

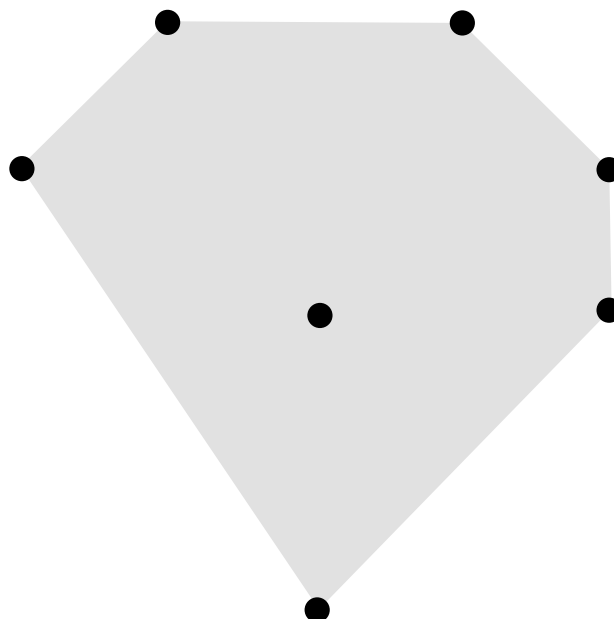
a) Fläche 1 (3 Punkte)

OpenGL-Primitiv:



b) Fläche 2 (3 Punkte)

OpenGL-Primitiv:



Aufgabe 9: OpenGL und Blending (6 Punkte)

In dieser Aufgabe sollen Sie die OpenGL-Pipeline für korrektes Blending in verschiedenen Szenarien konfigurieren.

Die folgenden Zustände sind bereits gesetzt:

```
glEnable (GL_DEPTH_TEST)
glDepthMask (GL_FALSE)
glDepthFunc (GL_LESS)

glEnable (GL_BLEND)
glBlendEquation (GL_FUNC_ADD)
glBlendFunc (  ,  )
```

Geben Sie die passenden Argumente , für `glBlendFunc()` für die unten beschriebenen Anwendungsfälle an. Wählen Sie dazu aus der folgenden Liste aus:

- | | | | |
|----------------------------|------------------------|----------------------------|------------------------|
| <input type="checkbox"/> 0 | GL_ZERO | <input type="checkbox"/> 5 | GL_ONE_MINUS_DST_ALPHA |
| <input type="checkbox"/> 1 | GL_ONE | <input type="checkbox"/> 6 | GL_SRC_COLOR |
| <input type="checkbox"/> 2 | GL_SRC_ALPHA | <input type="checkbox"/> 7 | GL_ONE_MINUS_SRC_COLOR |
| <input type="checkbox"/> 3 | GL_ONE_MINUS_SRC_ALPHA | <input type="checkbox"/> 8 | GL_DST_COLOR |
| <input type="checkbox"/> 4 | GL_DST_ALPHA | <input type="checkbox"/> 9 | GL_ONE_MINUS_DST_COLOR |

a) Zur Darstellung von Rauch soll ein Partikelsystem mit Alpha-Blending gezeichnet werden. Die Rauchpartikel sind texturierte Impostors (oder Billboards), wobei der Alphakanal der Textur die Opazität bestimmt.

I) In welcher Reihenfolge müssen die Partikel gezeichnet werden? (1.5 Punkte)

Kreuzen Sie die richtige Option an.	
Die Reihenfolge ist wegen des Tiefenpuffers egal.	<input type="checkbox"/>
Von hinten nach vorne.	<input type="checkbox"/>
Von vorne nach hinten.	<input type="checkbox"/>

II) Konfigurieren Sie nun die OpenGL-Blending-Faktoren für die Rauchpartikel! (1.5 Punkte)

`glBlendFunc (,)`

Name: _____

Matrikelnummer: _____

- b) Statt Rauch sollen nun Feuerpartikel gezeichnet werden. Feuerpartikel emittieren Licht und müssen daher mit anderen Blending-Faktoren gezeichnet werden. Welche sind das?
(1.5 Punkte)

glBlendFunc(,)

- c) Ein Teil der Szene ist *durch eine semi-transparente Glasfläche* zu sehen. Diese moduliert die Farben der dahinterliegenden Flächen mit einer farbigen Textur. Hat diese beispielsweise einen RGB-Wert von (1, 0.5, 0), so wird die Hälfte des grünen und der gesamte blaue Lichtanteil absorbiert. Mit welchen Blending-Faktoren wird dies erreicht?
(1.5 Punkte)

glBlendFunc(,)

Aufgabe 10: Bézier-Kurven und Bézier-Splines (17 Punkte)

a) Nennen Sie **drei** wichtige Eigenschaften der Bézier-Kurven, die Sie in der Vorlesung kennengelernt haben! **(3 Punkte)**

b) Ein Bézier-Spline soll für eine Animation verwendet werden. Dazu wird in einem GLSL-Vertex-Shader ein Spline in Abhängigkeit des Parameters t ausgewertet und die Eingabegeometrie anschließend entsprechend verschoben. **(14 Punkte)**

Es handelt sich um einen Bézier-Spline $S(t)$ mit t in $[0, 3)$, der aus drei kubischen Bézier-Kurven zusammengesetzt ist. Liegt t im Intervall $[0, 1)$, $[1, 2)$ bzw. $[2, 3)$, so ist jeweils die erste, zweite bzw. dritte Bézier-Kurve auszuwerten. Die Teilkurven sind dabei über s in $[0, 1)$ definiert; Sie müssen also s aus t bestimmen.

Der Spline ist durch zwölf Kontrollpunkte gegeben, die im Array `b[]` gespeichert sind; jeweils vier aufeinanderfolgende Kontrollpunkte beschreiben eine der Bézier-Kurven.

Ergänzen Sie in folgendem Vertex-Shader die Funktionsrümpfe der Funktionen `bezierspline3(..)` und `bezier3(..)`! Beachten Sie die Kommentare zu den Funktionen im Code.

Name: _____

Matrikelnummer: _____

```
uniform mat4 matrixMVP; // Model-View-Projection-Matrix
in vec3 position;      // Koordinaten des Eingabe-Vertex

uniform vec3 b[12];    // Array der Kontrollpunkte
uniform float time;    // Zeitpunkt für die Animation in [0;3)

// bezier3(..) soll die Bezier-Kurve an der Stelle s
//   auswerten und das Resultat als vec3 zurückgeben.
//   Sie können die Bernstein-Polynome oder den
//   Algorithmus von de Casteljau verwenden.
vec3 bezier3(float s, // Parameter s in [0;1)
             const vec3 b0, const vec3 b1, // Kontrollpunkte b0, b1, b2, b3
             const vec3 b2, const vec3 b3) {
    // Fügen Sie Ihren Code hier ein.

}

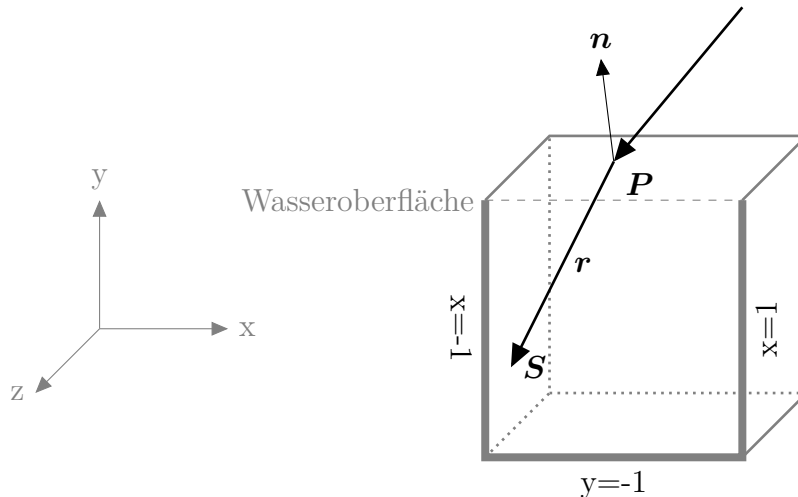
// bezierspline3(..) soll die Auswertung des Bezier-Splines an der
//   Stelle t als vec3 zurückgeben.
//   Verwenden Sie dazu die Funktion bezier3(..)!
vec3 bezierspline3(float t) {
    // Fügen Sie Ihren Code hier ein.

}

void main() {
    vec3 offset = bezierspline3(time);
    vec3 newpos = position + offset;
    gl_Position = matrixMVP * vec4(newpos, 1.0);
}
```

Aufgabe 11: Wasseroberfläche mit GLSL (18 Punkte)

Für die Darstellung der Wasseroberfläche eines komplett mit Wasser gefüllten Gefäßes soll ein GLSL-Fragment-Shader geschrieben werden. Das Gefäß ist würfelförmig und nur nach oben offen. Der Fragment-Shader soll den Eindruck von Lichtbrechung des Sichtstrahls an der Wasseroberfläche erwecken.



Der Würfel hat die Kantenlänge 2. In der folgenden Tabelle sind die Seitenflächen und die Wasseroberfläche mit den Ebenen, in denen diese liegen, angegeben.

Index	Fläche	liegt in Ebene	Index	Fläche	liegt in Ebene
0	Linke Seite	$x = -1$	1	Rechte Seite	$x = 1$
2	Hintere Seite	$z = -1$	3	Vordere Seite	$z = 1$
4	Boden	$y = -1$	5	Wasseroberfläche	$y = 1$

Sie sollen die Funktionen **determineIntersection** und **determineTextureCoordinate** implementieren, die im vorgegebenen Hauptprogramm aufgerufen werden:

```
uniform sampler2D wallMap; // Textur für die Gefäßflächen
uniform float eta; // Verhältnis der Brechungsindizes

in vec3 P; // Punkt auf der Wasseroberfläche in Weltkoordinaten
in vec3 n; // Normale an der Wasseroberfläche in Weltkoordinaten
in vec3 E; // Position des Betrachters in Weltkoordinaten
out vec4 color; // Ausgabefarbe

void main()
{
    vec3 v = normalize(P - E); // Einfallrichtung berechnen
    vec3 r = refract(v, n, eta); // Transmissionsrichtung bestimmen

    // Schnittpunkt mit Fläche und deren Index bestimmen
    int idx;
    vec3 S = determineIntersection(P, r, idx);

    // Texturkoordinate für Zugriff auf Textur aus dem Schnittpunkt
    // und dem Index bestimmen
    vec2 UV = determineTextureCoordinate(S, idx);

    color = texture(wallMap, UV); // Zugriff auf die Textur
}
```

Name: _____

Matrikelnummer: _____

- a) Implementieren Sie die Funktion **determineIntersection**, die den Schnittpunkt des Transmissionsstrahls mit den Gefäßflächen berechnet! Die Funktion muss auch den Index der Fläche zurückliefern, auf der der Schnittpunkt liegt. **(10 Punkte)**

Hinweise:

Ihnen steht hierzu die Funktion **intersect ()** zur Verfügung, die den Schnitt eines gegebenen Strahls mit einer der Ebenen durchführt. Die Ebene wird anhand eines Index identifiziert (siehe Tabelle).

intersect () hat folgende Signatur (sie muss *nicht* implementiert werden):

```
// Rückgabewert: true, wenn es einen Schnittpunkt gibt, sonst false
bool intersect( in int i, // Index der zu testenden Ebene (siehe Text)
               in vec3 X, // Ausgangspunkt des Strahls
               in vec3 d, // Strahlrichtung
               out float t // Strahlparameter des Schnittpunktes S: S=X+t*d
             );
```

```
vec3 determineIntersection(in vec3 P, in vec3 r, out int index)
{
    // Ermitteln Sie hier den Schnittpunkt mit der nächsten Gefäßfläche
    // und geben Sie ihn zurück. Zusätzlich muss 'index' auf den Index
    // der entsprechenden Seitenfläche gesetzt werden.
```

```
}
```

- b) Vervollständigen Sie nun die Funktion **determineTextureCoordinate**! Diese soll aus der Koordinate und dem Ebenenindex des Schnittpunkts die Texturkoordinate bestimmen, mit der dann auf die Textur der Fläche zugegriffen wird.

Jede Seitenfläche und auch der Boden wird mit derselben 2D-Textur dargestellt. Die Textur soll auf allen diesen Flächen vollständig dargestellt werden und diese auch komplett bedecken. Die Orientierung der Textur spielt dabei keine Rolle. **(8 Punkte)**

```
vec2 determineTextureCoordinate(in vec3 S, in int index)
{
    vec2 UV;

    switch(index)
    {
        // Vervollständigen Sie die Fälle entsprechend der Aufgabenstellung
        case 0:

        case 1:

        case 2:

        case 3:

        case 4:

        case 5:

    }

    // Fügen Sie ggf. notwendige weitere Anweisungen hier ein

    return UV;
}
```