

Nachklausur Computergrafik SS 2018

12. September 2018

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 20 Seiten (10 Blätter) mit 9 Aufgaben.
- Es sind **keine Hilfsmittel** zugelassen.
- Sie haben **90 Minuten** Bearbeitungszeit.
- Schreiben Sie Ihre Matrikelnummer oben auf jedes bearbeitete Aufgabenblatt.
- Schreiben Sie Ihre Lösungen auf die Aufgabenblätter. Bei Bedarf können Sie weiteres Papier anfordern.
- Wir akzeptieren auch englische Antworten.

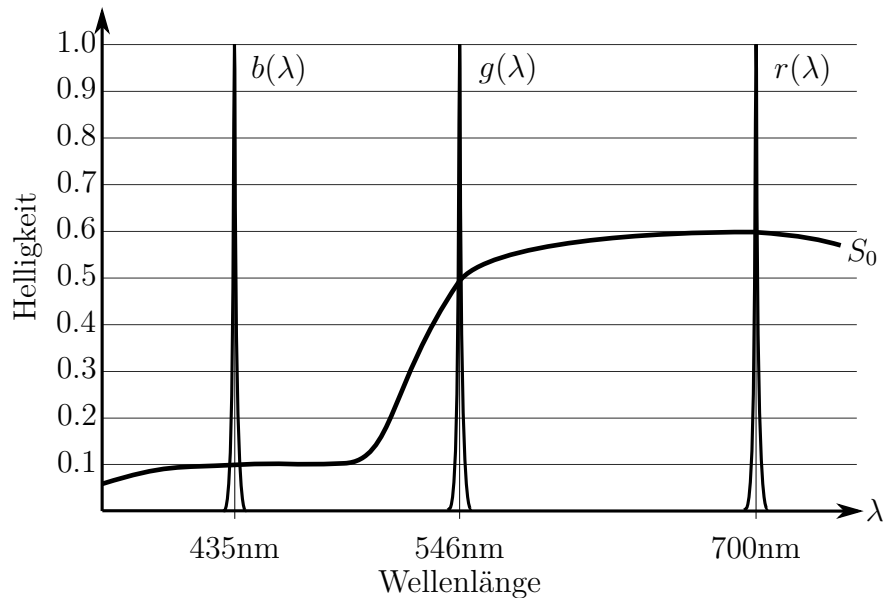
Aufgabe	1	2	3	4	5	6	7	8	9	Gesamt
Erreichte Punkte										
Erreichbare Punkte	11	27	23	22	8	16	20	22	31	180

Note



Aufgabe 1: Farben und Perzeption (11 Punkte)

- a) In diesem Diagramm ist ein Lichtspektrum S_0 dargestellt. Die drei Color Matching-Funktionen $r(\lambda)$, $g(\lambda)$, $b(\lambda)$ des CIE RGB Farbraumes sind die drei Dirac-Deltafunktionen, die im Diagramm angedeutet sind.



- i) Als welche Farbe würde das Spektrum ungefähr von einem menschlichen Betrachter wahrgenommen werden (rot, grün, blau, cyan, gelb, magenta)? **(2 Punkte)**



- ii) Zeichnen Sie ein Spektrum ein, sodass die additive Mischung der Farbe dieses Spektrums mit der Farbe des gegebenen Spektrums in CIE RGB den Tristimuluswert (0.7, 0.8, 0.9) hat! Beschriften Sie Ihr Spektrum mit S_1 ! **(3 Punkte)**




- iii) Zeichnen Sie ein anderes Spektrum ein, das im CIE RGB Farbraum ein Metamer zu dem gegebenen Spektrum S_0 ist! Beschriften Sie Ihr Spektrum mit S_2 ! **(3 Punkte)**

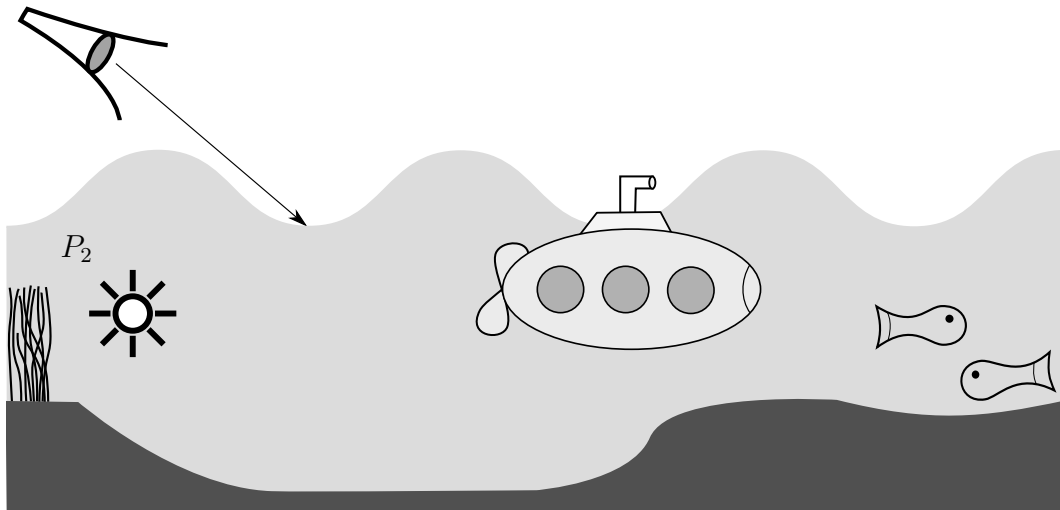


- b) Sie möchten die Helligkeit eines Bildes erhöhen. Nennen Sie einen Farbraum, in welchem Sie diese Operation möglichst einfach ausführen können, und wie! **(3 Punkte)**

Aufgabe 2: Whitted-Style Raytracing und das Phong-Beleuchtungsmodell (27 Punkte)



P_1 



Eine Unterwasserszene mit rein diffusem Meeresboden ($k_a = k_s = 0$) und einem rein diffusen, opaken U-Boot soll mit einem Whitted-Style Raytracer gerendert werden.

a) Der Whitted-Style Raytracer unterstützt Dispersion. Die Szene soll zunächst nur durch die eingezeichnete Punktlichtquelle P_1 beleuchtet werden. Die Wasseroberfläche sei rein transmittierend (Reflektivität = 0). Für das Wasser seien die wellenlängenabhängigen Brechungsindizes $1 < \eta_R < \eta_G < \eta_B$. Für Luft sei der Brechungsindex $\eta_{Luft} = 1$ für alle Wellenlängen.

i) Zeichnen Sie alle Sekundärstrahlen in die Skizze ein, die für den gegebenen Primärstrahl in einem Whitted-Style Raytracer verfolgt werden! Geben Sie an, um was für eine Art Strahl es sich jeweils handelt, indem Sie die Strahlen sinnvoll beschriften! Lassen Sie Strahlen weg, für die aufgrund der Materialeigenschaften bereits ohne Verfolgen des Strahls sicher ist, dass sie keinen Einfluss auf das Ergebnis haben können! **(6 Punkte)**



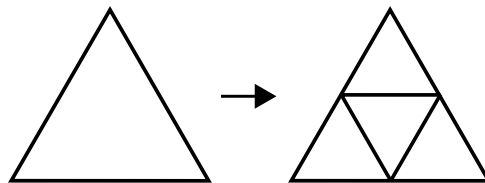
ii) Was ist die resultierende Pixelfarbe für den eingezeichneten Primärstrahl? Begründen Sie kurz! **(3 Punkte)**



- b) Nun soll die Szene nur durch die Punktlichtquelle P_2 beleuchtet werden. Die Farbe der Lichtquelle P_2 sei I_L und der zu beleuchtende Oberflächenpunkt sei nicht verschattet. Geben Sie den Term des Phong-Beleuchtungsmodells an, mit dem die Beleuchtung des rein diffusen Meeresgrundes berechnet wird! Beschreiben Sie kurz alle weiteren Komponenten des Terms! **(3 Punkte)**



- c) Nun wird der Meeresboden tesseliert, indem jedes Dreieck entsprechend der Abbildung in mehrere kleinere Dreiecke in der selben Ebene unterteilt wird. Die Normalen der neuen Eckpunkte werden aus den Normalen der beiden angrenzenden ursprünglichen Eckpunkte interpoliert. Hat diese Tessellierung des Meeresgrundes im Allgemeinen einen Einfluss auf die berechnete Beleuchtung, wenn Gouraud-Shading und das volle Phong-Beleuchtungsmodell verwendet wird? Begründen Sie kurz, indem Sie Bezug auf das Phong-Beleuchtungsmodell nehmen! **(4 Punkte)**



Matrikelnummer: _____

- d) Mit Hilfe welchen physikalischen Gesetzes lässt sich die gebrochene Strahlrichtung beim Übergang von einem Medium in ein anderes berechnen? Welche Größen benötigen Sie zur Berechnung der Richtung des gebrochenen Strahls? **(3 Punkte)**

☐

- e) Wie können vorgefilterte Environment-Maps in Kombination mit dem Phong-Beleuchtungsmodell in einem Whitted-Style Raytracer verwendet werden? Welche Vorberechnungen finden statt und wie wird auf die vorberechneten Daten zugegriffen? Nennen Sie einen Vorteil und eine Einschränkung von vorgefilterten Environment-Maps! **(8 Punkte)**

☐



Aufgabe 3: Beschleunigungsstrukturen (23 Punkte)

Beim Raytracing werden räumliche Datenstrukturen eingesetzt, die den Raum oder Primitive in Gruppen unterteilen, um die Anzahl an Schnittpunkten mit der Geometrie einer Szene zu reduzieren.



- a) Wenn eine Szene N Primitive enthält, wie viele Strahl-Primitive-Schnittpunkte müssen Sie erwartungsgemäß durchführen, um den nächsten Schnittpunkt eines zufälligen Strahls mit einem Primitive zu finden, wenn Sie **(2 Punkte)**

i) keine Beschleunigungsstrukturen verwenden?

ii) eine Hüllkörperhierarchie (Bounding Volume Hierarchy, BVH) verwenden?



- b) Beschreiben Sie exemplarisch eine nicht leere Szene oder deren Charakteristik, bei der die Verwendung eines regulären Gitters als Beschleunigungsstruktur keine Vorteile bringt, und begründen Sie, warum! **(3 Punkte)**



- c) Bei der Konstruktion mancher Datenstrukturen gibt es verschiedene Möglichkeiten, eine Unterteilung durchzuführen. Nennen Sie ein Kriterium, mit dem die Qualität einer Unterteilung heuristisch bewertet werden kann!

Nennen Sie außerdem zwei räumliche Datenstrukturen, für deren Konstruktion Sie dieses Kriterium verwenden können, und eine, für die dies nicht möglich ist! **(3 Punkte)**



- d) Nennen Sie drei Hüllkörper, die Sie in der Vorlesung kennengelernt haben, und ordnen Sie sie nach den Kosten eines Schnittpunkts zwischen einem Strahl und dem jeweiligen Hüllkörper! **(4 Punkte)**

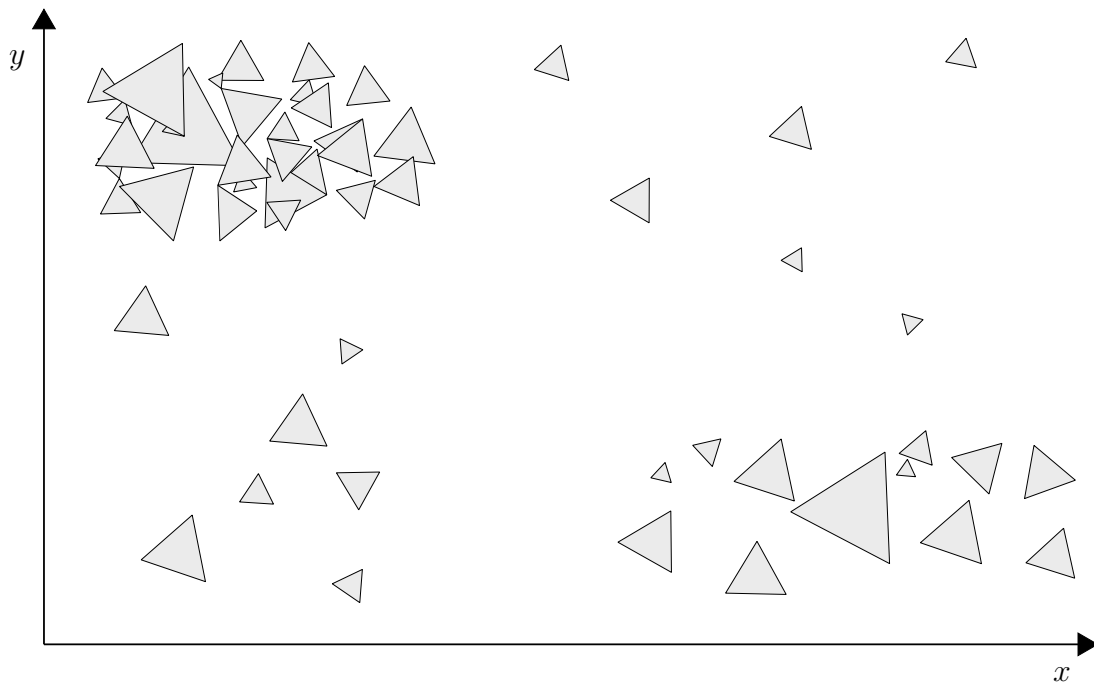
- e) Geben Sie eine objektunterteilende und zwei raumunterteilende Datenstrukturen an! **(3 Punkte)**



Objektunterteilend:

Raumunterteilend:

- f) Die Abbildung zeigt eine zweidimensionale Szene mit einigen Dreiecken. Zeichnen Sie zwei rekursive Unterteilungsschritte der Menge der Dreiecke in je zwei Teilmengen ein, wie Sie sie von der Heuristik aus c) erwarten würden! Führen Sie die erste Unterteilung entlang der x -Achse, die zweite entlang der y -Achse durch! Begründen Sie, warum eine so konstruierte hierarchische Datenstruktur zu höherer Raytracing-Performanz führen kann! **(8 Punkte)**





Aufgabe 4: Texturen (22 Punkte)

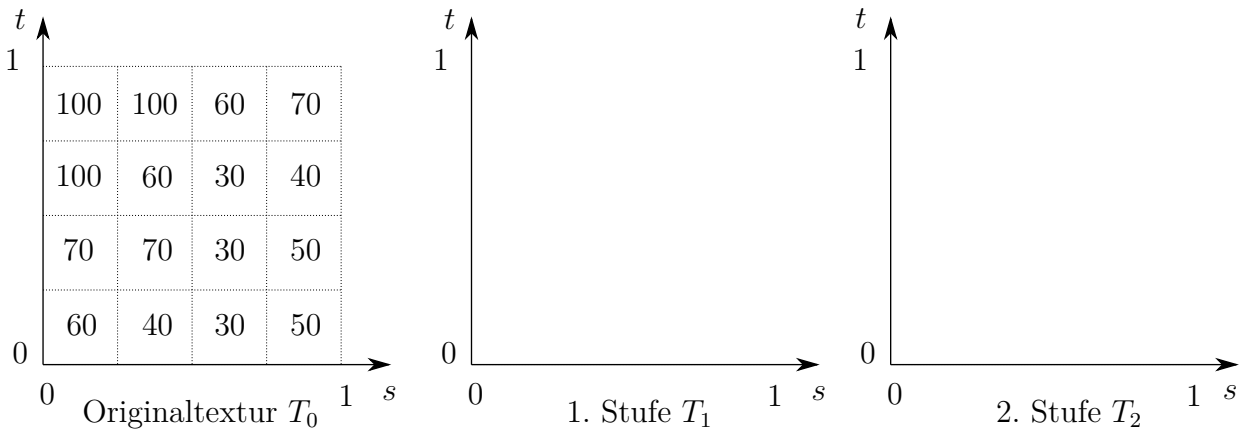


- a) Wie sehr erhöht sich der Speicheraufwand für Texturen, wenn Mip-Mapping eingesetzt wird? **(2 Punkte)**

- b) Gegeben sei die 4×4 Grauwerttextur T_0 .



- i) Zeichnen Sie die nächsten beiden Mipmap-Stufen von T_0 in die gegebenen Diagramme ein! **(5 Punkte)**



Im Folgenden soll die Textur mit verschiedenen Verfahren ausgelesen werden:



- ii) Geben Sie den Grauwert G an, der sich beim Auslesen der Originaltextur T_0 für $(s, t) = (\frac{1}{5}, \frac{1}{5})$ mit Nearest-Neighbor ergibt! **(2 Punkte)**

Matrikelnummer: _____

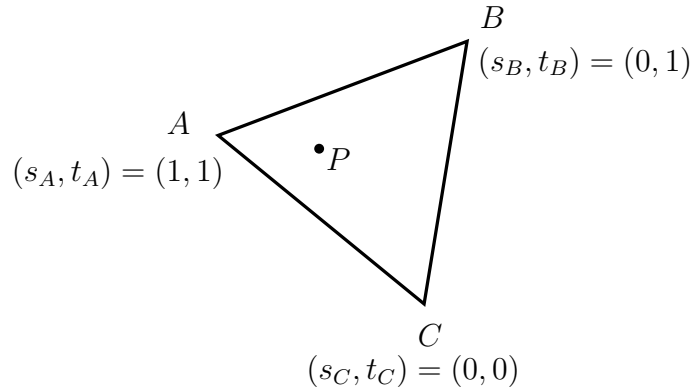
- iii) Berechnen Sie die Grauwerte G_0, G_1, G_2 , die sich beim Auslesen der gegebenen Textur (T_0) sowie der ersten (T_1) und zweiten (T_2) Mipmap-Stufe für $(s, t) = (\frac{5}{8}, \frac{3}{4})$ mit bilinearer Interpolation ergeben! **(4 Punkte)**



- iv) Berechnen Sie den Grauwert, der sich ergibt, wenn die Textur mithilfe der Mipmaps und trilinearer interpolation für $(s, t) = (\frac{5}{8}, \frac{3}{4})$ (*Tipp: Gleiche Koordinaten wie in iii)*) ausgelesen wird! Dabei sollen für die trilineare Interpolation die gegebene Textur T_0 und die 1. Stufe T_1 verwendet werden, das Gewicht für die gegebene Textur T_0 sei $\frac{1}{3}$. **(3 Punkte)**



- c) Die Texturkoordinaten der Eckpunkte des unten skizzierten Dreiecks ABC seien $(s_A, t_A) = (1, 1)$, $(s_B, t_B) = (0, 1)$, $(s_C, t_C) = (0, 0)$. Bestimmen Sie die baryzentrischen Koordinaten $\lambda_A, \lambda_B, \lambda_C$ des Punktes P im Dreieck, der die Texturkoordinaten $(s_P, t_P) = (\frac{1}{2}, \frac{3}{4})$ hat! **(6 Punkte)**



Aufgabe 5: Prozedurale Modellierung (8 Punkte)



Sie sollen Wolken mit Hilfe eines prozeduralen Modells generieren und darstellen. Die komplexen Strukturen in einer dreidimensionalen Wolke sollen mit Rauschfunktionen modelliert werden.

- a) Nennen Sie 3 Eigenschaften, die eine Rauschfunktion $n(\mathbf{x})$ für solch eine Aufgabe erfüllen sollte! **(3 Punkte)**



- b) Geben Sie einen sinnvollen Definitions- und Wertebereich von $n(\mathbf{x})$ an! Geben Sie zusätzlich an, wie diese Werte für die Modellierung der Wolke interpretiert werden! **(2 Punkte)**



- c) Mit einer Turbulenzfunktion können grobe und feine Strukturen modelliert werden. Geben Sie die Definition der Turbulenzfunktion über $n(\mathbf{x})$ mit 4 Oktaven an! **(3 Punkte)**



$turbulence(\mathbf{x}) =$



Aufgabe 6: Blending (16 Punkte)



- a) In einem Pixel im Framebuffer steht bereits die Farbe $c_0 = (c_{0,R}, c_{0,G}, c_{0,B}, c_{0,A})$ und soll im Folgenden mit der Farbe c_1 kombiniert werden. Geben Sie die Gleichung für die resultierende Farbe c an, die bei Kombination mit Alpha Blending (die übliche Art, in OpenGL semitransparente Flächen zu zeichnen) entsteht! (4 Punkte)



- b) Alpha Blending ist nicht kommutativ. Welche Konsequenz hat das für das Zeichnen von semitransparenten Partikeln? (2 Punkte)



- c) Im Folgenden soll eine Szene mit opaken und semitransparenten Primitiven dargestellt werden. Semitransparenz soll mittels Alpha Blending realisiert werden. In der Tabelle haben Sie mehrere Funktionen zur Auswahl, die den OpenGL-Zustand ändern oder Primitive zeichnen. Wählen Sie alle notwendigen Funktionen aus und bringen Sie sie in die richtige Reihenfolge! Sie dürfen anstelle der Funktionen direkt deren Nummern angeben. (10 Punkte)

[1] draw_opaque()	[2] draw_semitransparent()
[3] glEnable(GL_DEPTH_TEST)	[4] glDisable(GL_DEPTH_TEST)
[5] glDepthMask(GL_TRUE)	[6] glDepthMask(GL_FALSE)
[7] glEnable(GL_CULL_FACE)	[8] glDisable(GL_CULL_FACE)
[9] glEnable(GL_BLEND)	[10] glDisable(GL_BLEND)
[11] glBlendEquation(GL_FUNC_ADD)	
[12] glBlendEquation(GL_FUNC_SUBTRACT)	
[13] glBlendEquation(GL_FUNC_MAX)	
[14] glBlendFunc(GL_ZERO, GL_ONE)	
[15] glBlendFunc(GL_ONE, GL_ONE)	
[16] glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)	
[17] glBlendFunc(GL_ONE_MINUS_DST_ALPHA, GL_DST_ALPHA)	

Aufgabe 7: OpenGL Pipeline (20 Punkte)



- a) Bringen Sie die hier genannten Stufen der OpenGL Pipeline in die richtige Reihenfolge, indem Sie die Stufen mit 1 bis 5 nummerieren! **(5 Punkte)**



Geometry Shader	
Fragment Shader	
Rasterisierung	
Vertex Shader	
Blending	

- b) An welcher Stelle findet das Clipping statt? Begründen Sie, aus welchem Grund Clipping durchgeführt wird, und warum es an dieser Stelle durchgeführt wird! Begründen Sie außerdem, warum es nicht davor oder danach durchgeführt werden kann! **(7 Punkte)**



- ☐ c) In welcher Stufe der Pipeline wird das Beleuchtungsmodell jeweils beim Phong-Shading und beim Gouraud-Shading ausgewertet? Begründen Sie kurz! **(4 Punkte)**

Phong-Shading:

Gouraud-Shading:

- ☐ d) Was ist Z-Fighting, was ist die Ursache und wie kann Z-Fighting vermieden werden? **(4 Punkte)**

Aufgabe 8: GLSL Shader: Real-time Ray Tracing (22 Punkte)

In dieser Aufgabe soll Echtzeitgrafik um einige Ray Tracing-Effekte erweitert werden, um realistischere Reflexionen leuchtender Objekte zu erreichen.

- a) Implementieren Sie die Funktion `intersectLuminaries`, mit der Sie die Farbe und Entfernung des nächsten Schnittpunktes mit einer texturierten leuchtenden Oberfläche ermitteln können! Drei aufeinanderfolgende Vertices in `vPos` bilden ein Dreieck, die nächsten drei Vertices bilden das nächste usw. In `vTex` sind die Texturkoordinaten der Vertices aus `vPos` gespeichert. Die Emission der Oberflächen ist in `textureAtlas` gespeichert. *Hinweis: Die Funktion `intersectTri` schneidet Dreiecke und wird als gegeben betrachtet.* (15 Punkte)



```
// Daten der leuchtenden Oberflächen:
uniform vec3 vPos[MAX_VERTICES]; // Vertex-Positionen
uniform vec2 vTex[MAX_VERTICES]; // Vertex-Texturkoordinaten
uniform int vCount; // Anzahl der Vertices, 3 pro Dreiecksfläche
uniform sampler2D textureAtlas; // Emission der leuchtenden Oberflächen

#define MAX_FLOAT 2.e32

// Gibt vec4(x, y, z, t) mit t > 0 zurück, sodass gilt:
// Schnittpunkt == o + t * d == x*v0 + y*v1 + z*v2 ODER t == MAX_FLOAT
vec4 intersectTri(vec3 o // Strahlursprung
                , vec3 d // Strahlrichtung
                , vec3 v0, vec3 v1, vec3 v2 // Vertices des Dreiecks
                );

// Gibt vec4(vec3(r, g, b), t) zurück, also die RGB-Farbe des
// emittierten Lichts und die Distanz des nächsten Schnittpunkts,
// sonst schwarz
vec4 intersectLuminaries(vec3 o // Strahlursprung
                        , vec3 d // Strahlrichtung
                        ) {
    vec4 result = vec4(0, 0, 0, MAX_FLOAT);
```

```
    return result;  
}
```


- b) Der folgende Fragment Shader berechnet bereits die Normale eines Punktes, seine Farbe `c` ohne Oberflächenreflexion, seinen Reflektivitätswert und seine Position `p` in Weltkoordinaten. Die Normale und der Reflektivitätswert werden gemeinsam in `nr` gespeichert – die Normale in den RGB-Kanälen und der Reflektivitätswert im Alpha-Kanal. Nun soll die Oberflächenreflexion mithilfe der Funktion aus a) für den Punkt berechnet und zu seiner Farbe ohne Oberflächenreflexion hinzugefügt werden. Vervollständigen Sie den Shader, sodass er für jeden Bildschirmpixel entsprechend die Reflexion der leuchtenden Oberflächen hinzufügt! **(7 Punkte)**



```
vec4 intersectLuminaries(vec3 o, vec3 d); // siehe Aufgabenteil a)

in vec3 camDir; // Unnormalisierte Kamerarichtung in Weltkoordinaten

out vec4 finalColor; // Finale Farbe der Szene mit Reflexionen

// Fragment Shader
void main() {
    vec4 c; // wird beschrieben mit Farbe ohne Reflexion
    vec4 nr; // Normale im RGB-Kanal, Reflektivitätswert im Alpha-Kanal
    vec3 p; // Koordinaten des Punktes in Weltkoordinaten

    ... // Füllt c, nr und p

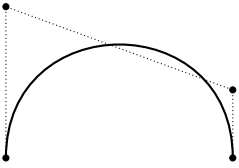
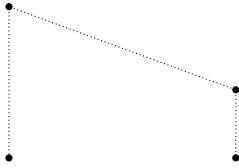
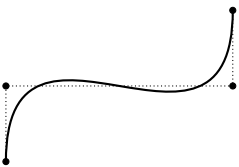

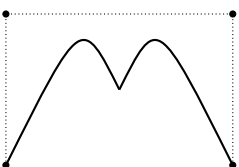

}
```



Aufgabe 9: Bézierkurven und Transformationen (31 Punkte)

- a) Die folgenden Kurven sind keine Bézierkurven für die gegebenen Kontrollpunkte. Geben Sie für die ersten beiden Kurven einen Grund und für die letzte Kurve zwei Gründe dafür an! Skizzieren Sie die korrekten Kurven in der rechten Spalte! (9 Punkte)

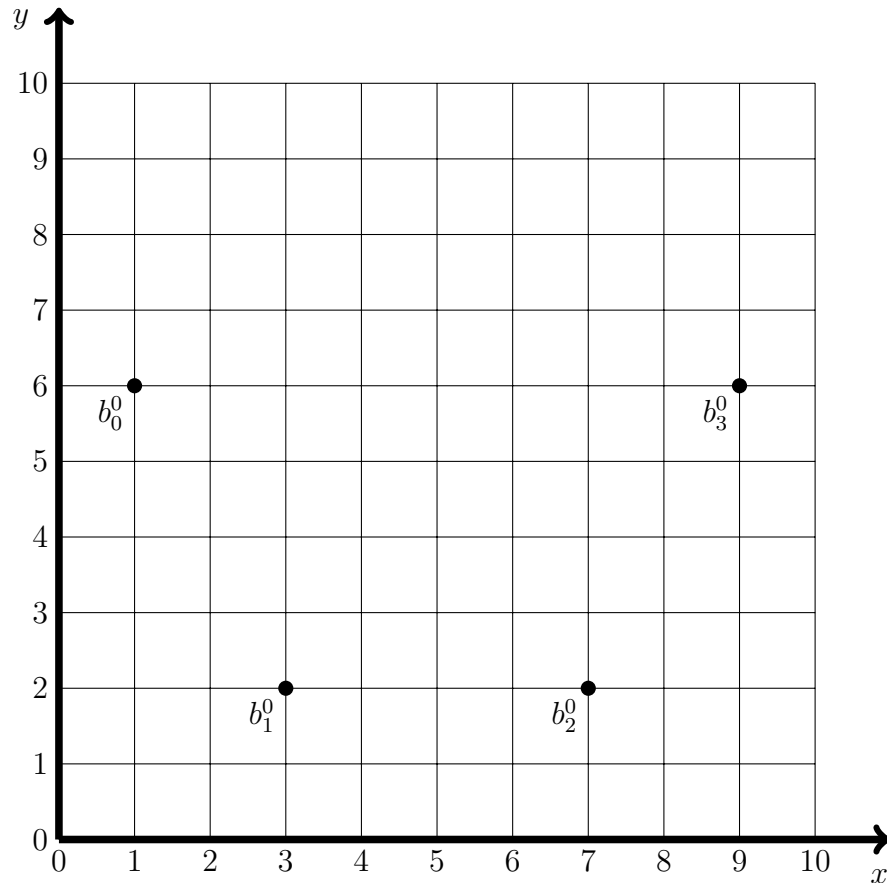


Kurve	Begründung	Bézierkurve
		
		
		

- b) Gegeben sind die folgenden Kontrollpunkte einer kubischen Bézierkurve \mathbf{b} , die auch in dem Diagramm eingezeichnet sind:

$$b_0^0 = \begin{pmatrix} 1 \\ 6 \end{pmatrix}, b_1^0 = \begin{pmatrix} 3 \\ 2 \end{pmatrix}, b_2^0 = \begin{pmatrix} 7 \\ 2 \end{pmatrix}, b_3^0 = \begin{pmatrix} 9 \\ 6 \end{pmatrix}.$$

- i) Werten Sie die Kurve für $u = 0.5$ grafisch mit dem de Casteljau-Algorithmus aus! Skizzieren Sie die Kurve im Diagramm mithilfe Ihrer grafischen Auswertung! **(4 Punkte)**.



- ii) Berechnen Sie die Kontrollpunkte der rechten Teilkurve, wenn \mathbf{b} bei $u = 0.5$ unterteilt wird! **(6 Punkte)**

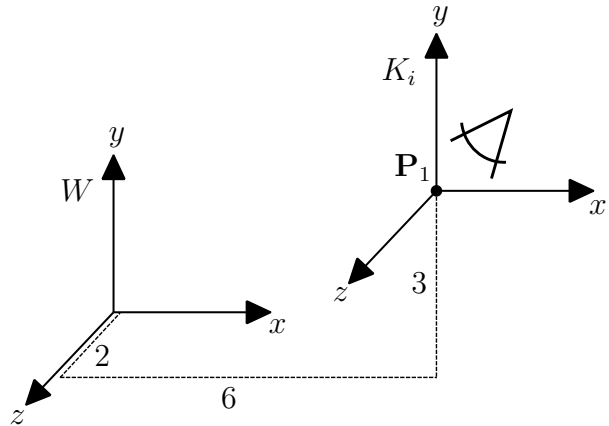


- c) Alle Koordinatensysteme in dieser Aufgabe seien rechtshändig. Eine Kamera befindet sich am Punkt $\mathbf{P}_1 = (6, 3, 2)$. In ihrem lokalen Kamerakoordinatensystem blickt die Kamera immer in z -Richtung, die y -Achse weist nach oben und die x - und y -Achse liegen parallel zur Bildebene. In Weltkoordinaten hat die Kamera den up-Vektor $(0, 1, 0)$.

- i) Die Kamera blickt in Richtung der globalen z -Achse (siehe Skizze). Geben Sie die 4×4 -Transformationsmatrix T an, die einen Punkt in homogenen Koordinaten vom Weltkoordinatensystem (W) in das Kamerakoordinatensystem K_i transformiert! **(4 Punkte)**



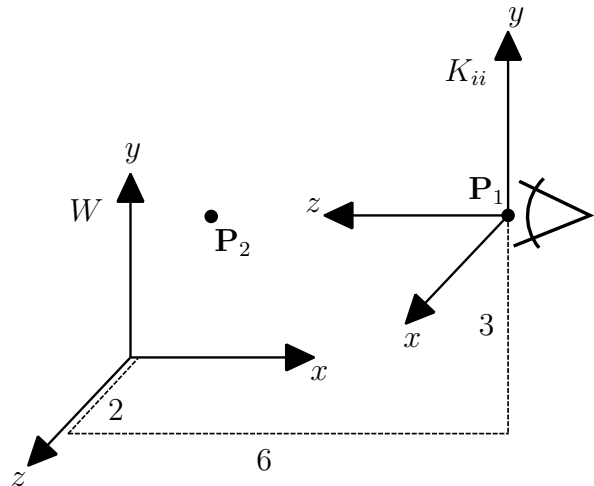
$T =$



- ii) Nun wird die Kamera so rotiert, dass sie zum Punkt $\mathbf{P}_2 = (2, 3, 2)$ blickt (siehe Skizze). Der up-Vektor ist unverändert $(0, 1, 0)$. Geben Sie die Transformationsmatrix R an, die einen Punkt aus dem Koordinatensystem K_i aus Aufgabe i) in das neue Kamerakoordinatensystem K_{ii} transformiert! **(6 Punkte)**



$R =$



- iii) Wie müssen Sie R aus ii) und T aus i) verknüpfen, um die Transformationsmatrix M zu erhalten, die von Weltkoordinaten W direkt in K_{ii} transformiert? **(2 Punkte)**



$M =$