Karlsruher Institut für Technologie

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

Nachklausur Computergrafik
SS 2017

15. September 2017

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 10 Aufgaben.
- Es sind **keine Hilfsmittel** zugelassen.
- Sie haben **90 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:

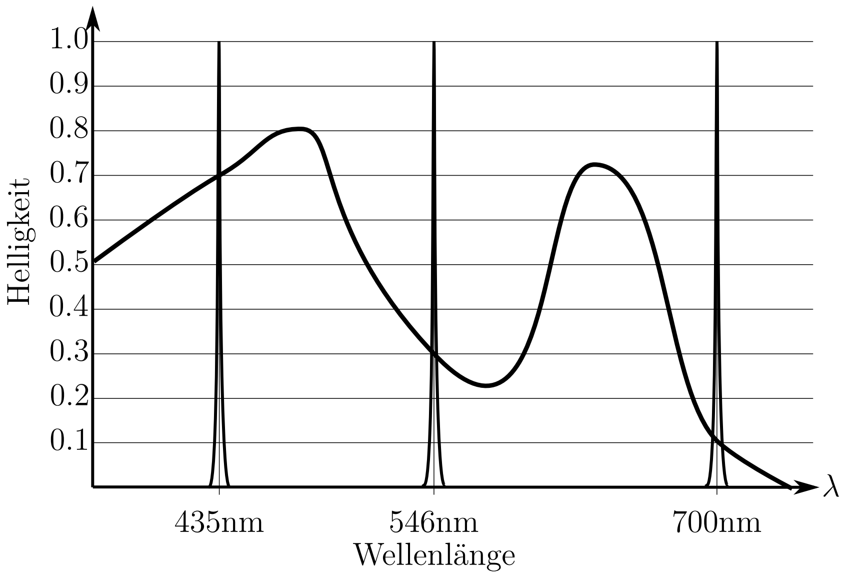
- Falsche Kreuze bei Wahr-Falsch Multiple-Choice-Aufgaben führen zu Punktabzug. Jede Teilaufgabe wird mit mindestens 0 Punkten bewertet.

Aufgabe	1	2	3	4	5	6	7	8	9	10	**Gesamt**
Erreichte Punkte											
Erreichbare Punkte	12	16	22	21	22	20	15	16	16	20	**180**

Note

Aufgabe 1: Farben (12 Punkte)

In diesem Diagramm ist ein Lichtspektrum dargestellt. Die drei Color Matching Funktionen des CIE RGB Farbraums sind jeweils drei Dirac-Deltafunktionen, die im Diagramm angedeutet sind.



1. Als welche Farbe würde das Spektrum ungefähr von einem menschlichen Betrachter wahrgenommen (rot, grün, gelb, blau, cyan oder magenta)? **(3 Punkte)**

2. Wie berechnen Sie für das Spektrum $s(\lambda)$ allgemein einen Tristimuluswert X für eine gegebene Color Matching Funktionen $c(\lambda)$? **(3 Punkte)**

3. Bestimmen Sie die CIE RGB Tristimuluswerte für das eingezeichnete Spektrum aus dem Diagramm. **(3 Punkte)**

4. Zeichnen Sie ein beliebiges anderes Spektrum in das Diagramm ein, welches in CIE RGB die selben Tristimuluswerte wie das gegebene Spektrum hat. **(3 Punkte)**

Name: ______________________ Matrikelnummer: ______________________

Aufgabe 2: Transformationen (16 Punkte)

1. Mit homogenen Koordinaten lassen sich affine Abbildungen als Matrixmultiplikationen darstellen. Geben Sie die 3×3-Transformationsmatrizen an, die Punkte $\boldsymbol{p} \in P(\mathbb{R}^3)$ in homogenen Koordinaten

   (a) um einen Vektor $\boldsymbol{t} = \begin{pmatrix} t_x \\ t_y \end{pmatrix}$ verschieben: **(2 Punkte)**

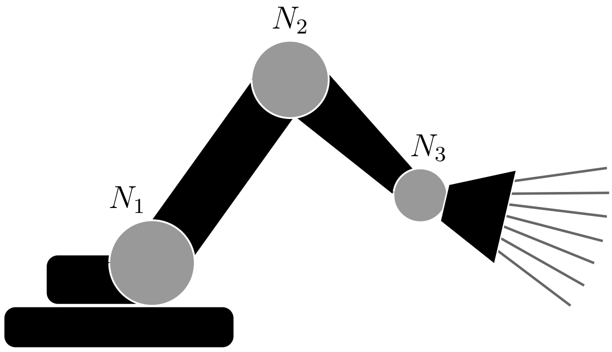
   $T(\boldsymbol{t}) =$

   (b) um Faktoren $\boldsymbol{s} = \begin{pmatrix} s_x \\ s_y \end{pmatrix}$ skalieren: **(2 Punkte)**

   $S(\boldsymbol{s}) =$

   (c) um $90°$ im Uhrzeigersinn um den Ursprung rotieren: **(3 Punkte)**

   $R(90°) =$



2. Für die Beleuchtungsberechnung mit einer an einem Roboterarm befestigten Lichtquelle soll Shadow Mapping durchgeführt werden. Der Roboterarm besteht aus drei Gelenken. Die Lichtquelle ist am dritten Gelenk befestigt. Die Matrix N_1 ist die Transformation von Weltkoordinaten in das Koordinatensystem des ersten Gelenks. Die Matrix N_2 transformiert vom Koordinatensystem des ersten Gelenks in das Koordinatensystem des zweiten Gelenks. Die Matrix N_3 gibt die Transformation vom Koordinatensystem des zweiten Gelenks in das Koordinatensystem des dritten Gelenks an. Die Matrix V_C transformiert Punkte von Weltkoordinaten in das lokale Koordinatensystem der Kamera und P_L ist die Projektionsmatrix der Lichtquelle.

   Für Shadow Mapping sei nun ein Punkt $p \in \mathbb{R}^4$ im lokalen Koordinatensystem der Kamera gegeben. Wie setzt sich die Transformationsmatrix M zusammen, die p in den Clip-Space der Lichtquelle transformiert? **(5 Punkte)**

   $M =$

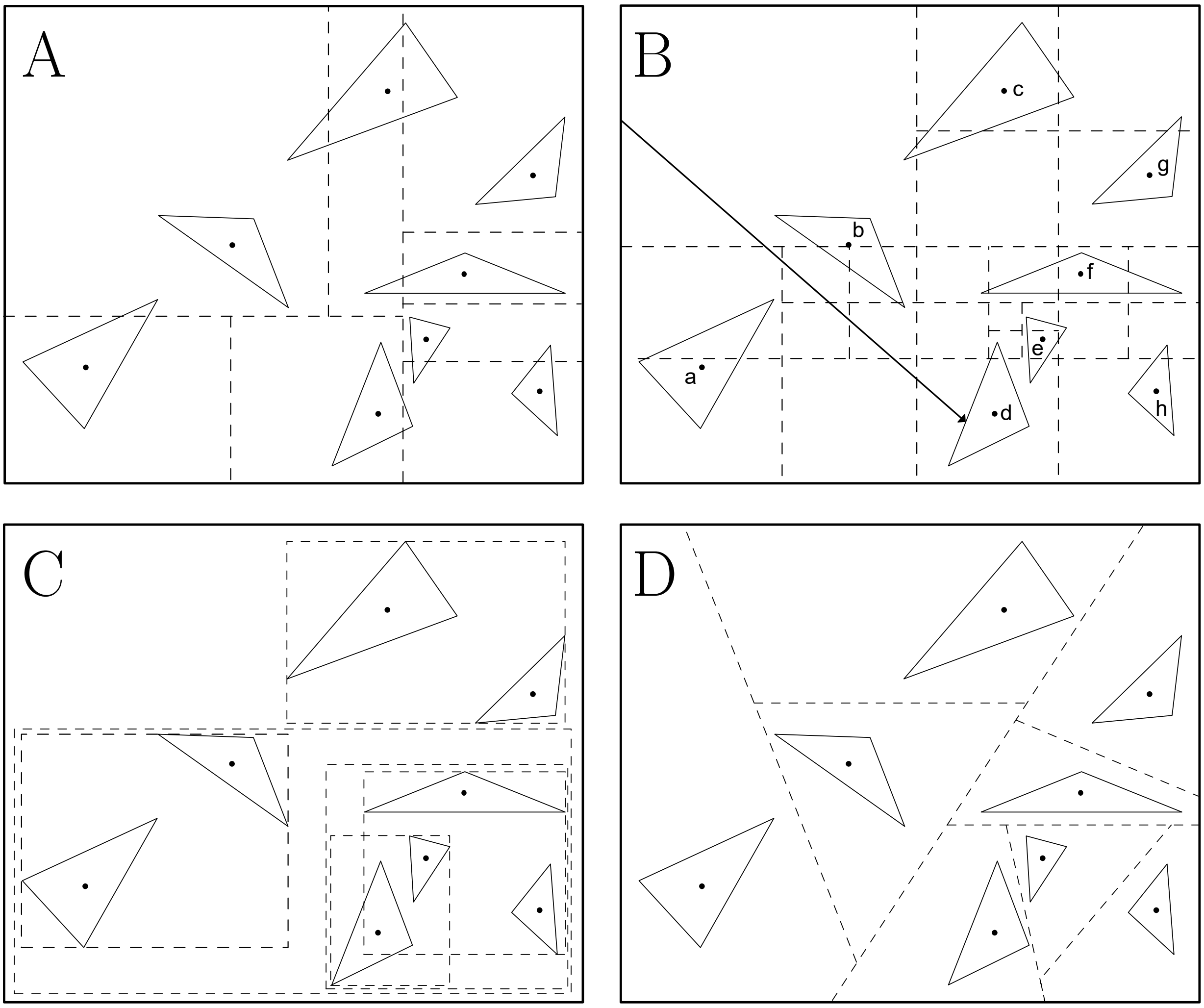
3. Wie müssen normalisierte Gerätekoordinaten transformiert werden, um auf die Shadow Map zuzugreifen? **(2 Punkte)**

4. Benötigt man die Projektionsmatrix der Lichtquelle auch, wenn in OpenGL auf eine Cube Map als Shadow Map zugegriffen werden soll? Begründen Sie kurz. **(2 Punkte)**

Name: ______________________ Matrikelnummer: ______________________

Aufgabe 3: Beschleunigungsstrukturen (22 Punkte)

Die vier Abbildungen A, B, C und D visualisieren vier unterschiedliche Beschleunigungsstrukturen, die über der selben Dreiecksmenge aufgebaut wurden.

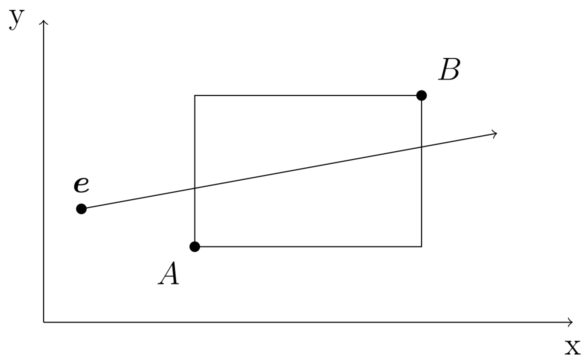


1. Wie heißen die dargestellten Beschleunigungsstrukturen? **(4 Punkte)**

Abbildung	Name
A	
B	
C	
D	

2. Traversieren Sie die Struktur B um den eingezeichneten Strahl mit den Dreiecken zu schneiden und notieren Sie alle durchgeführten Schnitttests mit Dreiecken in der Reihenfolge, in der sie stattfinden. Gehen Sie dabei davon aus, dass die Dreiecke in den Blattknoten des Baums gespeichert werden und kein Mailboxing verwendet wird. **(5 Punkte)**

3. Gegeben sind ein Strahl $\boldsymbol{r}(t) = \boldsymbol{e} + t\boldsymbol{d}$ mit $\boldsymbol{e} = (e_x, e_y)$ und $\boldsymbol{d} = (d_x, d_y)$, sowie ein zweidimensionaler achsenparalleler Hüllquader (2D-AABB). Die AABB ist beschrieben durch eine linke untere Ecke $A = (a_x, a_y)$ und eine rechte obere Ecke $B = (b_x, b_y)$. Der Strahl soll, so wie in der Vorlesung besprochen, auf den nächsten Schnittpunkt in Strahlrichtung ($t \geq 0$) mit der AABB getestet werden.



(a) Geben Sie an, wie die Strahlparameter t_1 und t_2 für die Schnittpunkte mit den Begrenzungsebenen senkrecht zur x-Achse bestimmt werden. **(4 Punkte)**

(b) Wie werden t_{near} und t_{far} nach diesem ersten Schritt initialisiert? **(2 Punkte)**

(c) Der nächste Schnitt mit den Ebenen parallel zur y-Achse ergibt die Strahlparameter t_3 und t_4. Wie werden t_{near} und t_{far} nun aktualisiert? **(3 Punkte)**

(d) Geben Sie nun ausgehend von t_{near} und t_{far} an, unter welche(n/r) Bedingung(en) es keinen Schnittpunkt in Richtung $\boldsymbol{d}$ gibt. Wenn ein Schnittpunkt existiert, welchen Strahlparameter t hat dieser? **(4 Punkte)**

	Bedingung(en) / Strahlparameter
Kein Schnitt	
Schnitt	$t =$

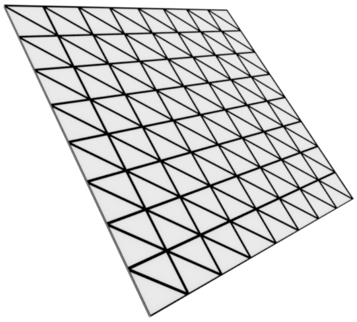
Aufgabe 4: Surface Area Heuristic (21 Punkte)

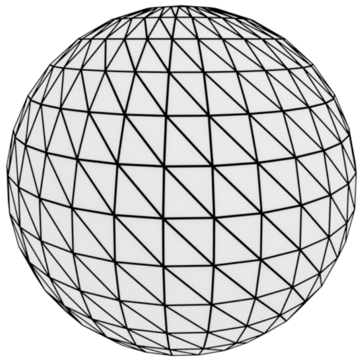
Eine Hüllkörperhierarchie soll mit der sogenannten Surface Area Heuristic (SAH) aufgebaut werden. Dazu muss in jeder Iteration eine Unterteilungsebene bestimmt werden, die bezüglich der SAH optimal ist. Vervollständigen Sie auf der nächsten Seite die Funktion `getPartitionIdx`, die die Dreiecke in zwei optimale Teilmengen aufteilt. Verwenden Sie die bereitgestellten Hilfsfunktionen! Als Eingabe erhalten Sie die AABBs der Dreiecke. Gehen Sie davon aus, dass diese bereits entlang der Unterteilungsachse sortiert sind. Berechnen Sie die AABBs für jede mögliche Unterteilungsebene senkrecht zu dieser Achse möglichst laufzeiteffizient, wie Sie es in der Vorlesung gelernt haben! **(21 Punkte)**

```
struct AABB {...};  // Achsenparalleler Hüllkörper

// Berechnet die AABB der übergebenen AABBs
AABB unionAABB(AABB aabb1, AABB aabb2);

// Berechnet die SAH für eine Unterteilung
float evalSAH(AABB left, int leftNumTris, AABB right, int rightNumTris);

// Gibt den Index idx zurück, der die AABBs aabb in zwei Teilmengen
// {aabb[0] ... aabb[idx]} und {aabb[idx+1] ... aabb[aabb.size()-1]}
// unterteilt, die optimal bezüglich der SAH sind.
int getPartitionIdx(const std::vector<AABB> &aabb)
{
   int idx = 0;
   int n = aabb.size(); // Anzahl Dreiecke im Knoten




   return idx;
}
```

Aufgabe 5: Beleuchtungsmodelle & Shading (22 Punkte)

Eine tesselierte Ebene und Kugel werden mit dem *Phong-Beleuchtungsmodell* gezeichnet. Je höher der Grad der Tesselierung, desto besser approximiert das Dreiecksnetz das analytische Modell der Körper. Abhängig davon, welche Art von Lichtquelle und ob Gouraud- oder Phong-Shading benutzt wird, kann das Aussehen der Objekte vom Grad der Tesselierung abhängen.





1. Kreuzen Sie die an, welche Beiträge des Beleuchtungsmodells sich abhängig von der Tesselierung ändern, wenn die *Ebene* mit einer Punkt- bzw. direktionalen Lichtquelle gerendert wird! Für jede richtig ausgefüllte Spalte erhalten Sie 1 Punkt. **(4 Punkte)**

	Punktlichtquelle		direktionale Lichtquelle	
Komponente	Phong-Shading	Gouraud-Shading	Phong-Shading	Gouraud-Shading
diffus	☐	☐	☐	☐
spekular	☐	☐	☐	☐
keine	☐	☐	☐	☐

2. Verändert sich die Schattierung im inneren Bereich der Kugel abhängig von der Tesselierung bei Phong-Shading mit einer direktionalen Lichtquelle? Begründen Sie ihre Antwort! **(6 Punkte)**

Name: ______________________ Matrikelnummer: ______________

3. Eine Kugel wird mit Phong-Shading und Phong-Beleuchtungsmodell gerendert.

(a) (b) (c) (d) (e)

(a) Geben Sie zu den jeweiligen Bildern an, um welche Komponente(n) des Beleuchtungsmodells es sich handelt. **(2 Punkte)** ☐

Alle Komponenten des Beleuchtungsmodells	Keine Komponente des Beleuchtungsmodells	ambient	diffus	spekular
(a)				

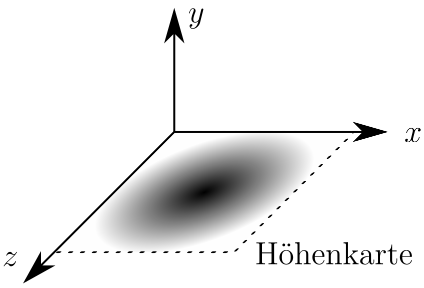
(b) Welche Lichtquelle(n) könnte(n) beim obigen Rendering der Kugel verwendet worden sein? **(2 Punkte)** ☐

Punktlichtquelle	parallele/direktionale Lichtquelle	konstante Environment-Map	Flächenlichtquelle
☐	☐	☐	☐

4. Wie unterscheidet sich die Beleuchtungsberechnung mit Punktlichtquellen von der mit parallelen/direktionalen Lichtquellen in einem Whitted-Style Raytracer? **(4 Punkte)** ☐

5. Beschreiben Sie stichpunktartig, wie Sie Flächenlichtquellen in einem Whitted-Style Raytracer implementieren würden? **(4 Punkte)** ☐

Aufgabe 6: OpenGL - Shader (20 Punkte)

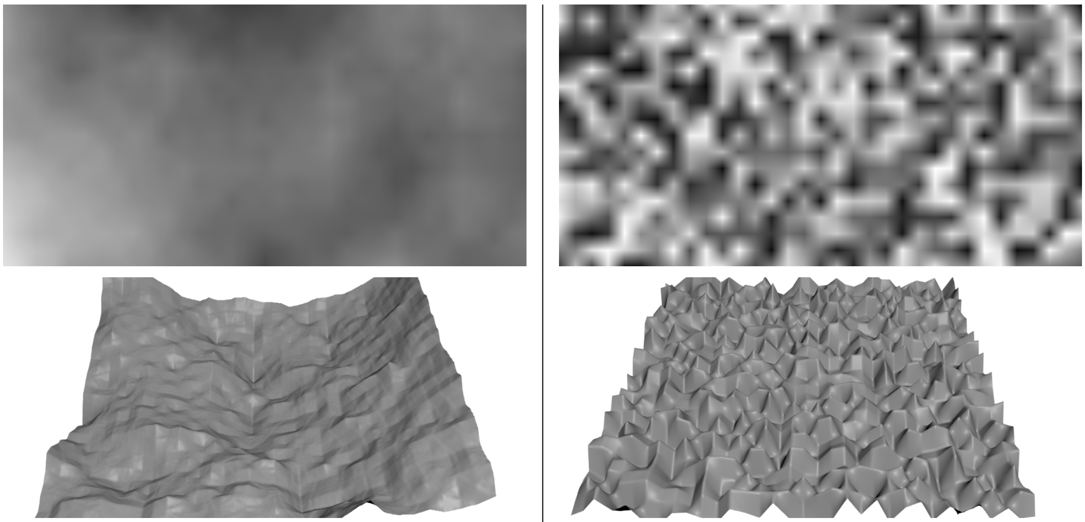


1. Gegeben ist ein Terrain in Form einer Höhenkarte. Vervollständigen Sie den Fragment Shader auf der nächsten Seite, um einen einfachen Ray Marching-Algorithmus umzusetzen!

   Benutzen Sie für das Ray Marching eine feste Schrittweite von `s` *in Weltkoordinaten*. Der Strahl schneidet das Terrain, sobald der aktuelle Punkt entlang des Strahls unter der aus der Höhenkarte ausgelesenen Höhe liegt. Auf die Höhenkarte kann über die xz-Koordinate des Punktes *im lokalen Koordinatensystem* des Terrains zugegriffen werden. Verwerfen Sie das Fragment falls der Strahl das Terrain nicht schneidet. Schreiben Sie andernfalls den Wert aus der Textur `tex_color` in `frag_color`. **(12 Punkte)**

```
in vec3 ray_dir;              // Strahlrichtung in Weltkoordinaten
uniform vec3 camera_position; // Kameraposition in Weltkoordinaten

uniform mat4 worldToTerrain;  // von Weltkoordinaten zu Terrainkoordinaten
uniform mat4 terrainToWorld;  // von Terrainkoordinaten zu Weltkoordinaten

uniform sampler2D tex_height; // speichert die Höhenkarte
uniform sampler2D tex_color;  // speichert die Terraintextur

uniform float s;              // Schrittlänge in Weltkoordinaten
uniform int max_steps;        // maximale Anzahl von Schritten

out vec4 frag_color;          // Ausgabefarbe

void main()
{
    vec3 dir_world = normalize(ray_dir);
    vec3 pos_world = camera_position;









}
```

2. In der folgenden Abbildung sind zwei Höhenkarten mit zugehörigem Terrain dargestellt. Welche der Höhenkarten wurde durch eine Rauschfunktion und welche durch eine Turbulenzfunktion erstellt? Geben Sie in Stichpunkten an, wie die Funktionswerte berechnet werden. **(4 Punkte)**



3. Wie können durch Turbulenzfunktionen generierte Texturen *effizient* gefiltert werden, um durch Unterabtastung entstehende Artefakte (Aliasing) zu verringern? **(2 Punkte)**

4. Anstatt die Höhenkarte zu speichern, ist es auch möglich die Rausch- bzw. Turbulenzfunktion zur Laufzeit auszuwerten. Welche Eigenschaft der Rauschfunktionen gewährleistet, dass sich das Gelände nicht in jedem Bild ändert? **(2 Punkte)**

Name: ______________________ Matrikelnummer: ______________________

Aufgabe 7: OpenGL - Blending (15 Punkte)

1. In dieser Aufgabe soll eine Szene mit mehreren Lichtquellen gezeichnet werden. Dazu wird die Szene mehrfach gezeichnet wobei immer die Beleuchtung einer Lichtquelle akkumuliert wird. Weiterhin soll auch ein Partikelsystem mit Rauch- und Feuerpartikeln richtig gezeichnet werden.

   Im Folgenden sollen Sie nun den OpenGL Blending Operator einstellen. Sie können aus den folgenden Argumenten wählen und in den Aufgaben deren Nummern eintragen:

0 GL_ZERO	5 GL_ONE_MINUS_DST_ALPHA
1 GL_ONE	6 GL_SRC_COLOR
2 GL_SRC_ALPHA	7 GL_ONE_MINUS_SRC_COLOR
3 GL_ONE_MINUS_SRC_ALPHA	8 GL_DST_COLOR
4 GL_DST_ALPHA	9 GL_ONE_MINUS_DST_COLOR

   (a) Konfigurieren Sie OpenGL, um die Beleuchtung mehrerer Lichtquellen zu akkumulieren. **(2 Punkte)**

   ```
   glBlendFunc(      ,      )
   ```

   (b) Setzen Sie den korrekten Blending Operator um Rauchpartikel zu zeichnen, die kein Licht emittieren. **(2 Punkte)**

   ```
   glBlendFunc(      ,      )
   ```

   (c) Bestimmen Sie nun den Blending Operator für die Feuerpartikel, welche Licht emittieren. **(2 Punkte)**

   ```
   glBlendFunc(      ,      )
   ```

Weiterhin sollen Sie den Tiefentest konfigurieren. Mit der Funktion `glDepthMask` können Sie das Schreiben in den Tiefenpuffer aktivieren (`GL_TRUE`) oder deaktivieren (`GL_FALSE`).

Für die Funktion `glDepthFunc`, welche nur bei eingeschaltetem Tiefentest von Bedeutung ist, stehen Ihnen die folgenden Argumente zur Verfügung.

0	`GL_FALSE`	5	`GL_LEQUAL`
1	`GL_TRUE`	6	`GL_GREATER`
2	`GL_NEVER`	7	`GL_NOTEQUAL`
3	`GL_LESS`	8	`GL_GEQUAL`
4	`GL_EQUAL`	9	`GL_ALWAYS`

(a) Wie muss dazu der Tiefentest konfiguriert werden, wenn Sie alle opaken Objekte wiederholt für jede Lichtquelle zeichnen? **(2 Punkte)**

```
gl__________(GL_DEPTH_TEST);
glDepthFunc(       );
glDepthMask(       );
```

(b) Stellen Sie nun den Tiefentest zum Zeichnen der Partikel ein. **(2 Punkte)**

```
gl__________(GL_DEPTH_TEST);
glDepthFunc(       );
glDepthMask(       );
```

(c) Mit welcher Blendingvariante könnte man Rauch und Feuerpartikel in einem Draw-Aufruf zeichnen und in welcher Reihenfolge müssten die Partikel dann gezeichnet werden? **(2 Punkte)**

Blendingvariante:

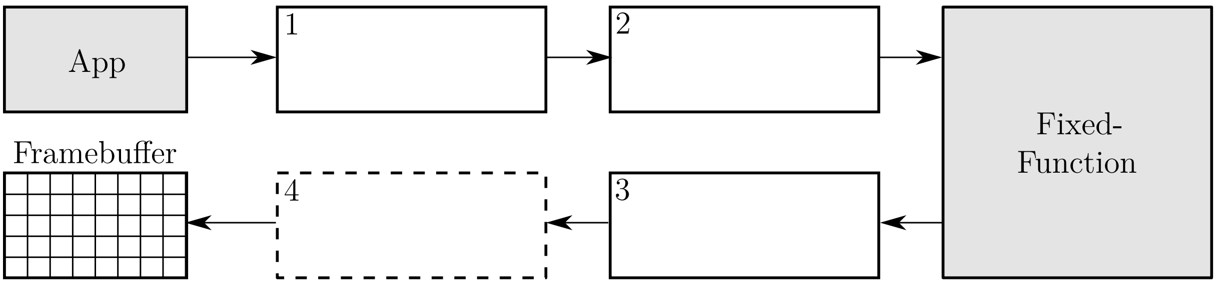
Kreuzen Sie die richtige Option an	
Die Reihenfolge ist wegen des Tiefenpuffers egal.	☐
Von hinten nach vorne.	☐
Von vorne nach hinten.	☐

Name: ______________________ Matrikelnummer: ______________________

2. Erklären Sie kurz den Verwendungszweck des *Alpha Tests* und erläutern Sie dessen Implementation mit modernem OpenGL. In welchem Shader muss dies stattfinden? **(3 Punkte)**

Aufgabe 8: OpenGL - Pipeline (16 Punkte)

1. In folgendem Diagramm ist die OpenGL-Pipeline ohne Unterstützung für Hardware-Tesselierung dargestellt. Der Block `Fixed-Function` enthält nicht modifizierbare Funktionalität. Die durchgezogenen Kästen sind drei frei programmierbare Stufen, der gestrichelte Kasten lediglich konfigurierbar. Tragen Sie ein, um welche Stufen es sich handelt! **(4 Punkte)**



2. Nennen Sie drei Schritte die zwischen den Stufen 2 und 3 durchgeführt werden und nicht programmierbar sind. **(3 Punkte)**

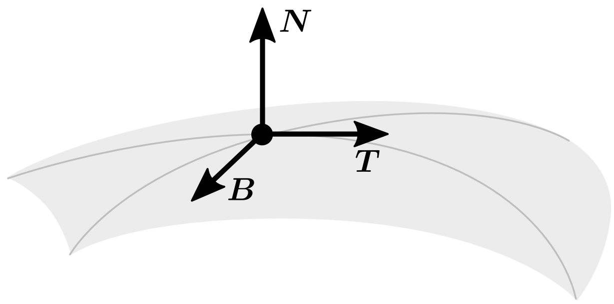
3. Angenommen Sie möchten im Rahmen dieser Pipeline eine Unterteilung der Eingabegeometrie in Dreiecke durchführen. Begründen Sie kurz, in welcher Stufe Sie diese Unterteilung vornehmen würden. **(2 Punkte)**

Name: ____________________ Matrikelnummer: ____________________

4. Welche Koordinatentransformationen und Berechnungen führen Sie in welcher Stufe der Pipeline aus, um in Objektkoordinaten gegebene Geometrie in Weltkoordinaten mit Phong-Shading zu beleuchten? Wo werten Sie das Beleuchtungsmodell aus? **(3 Punkte)**

5. Nennen Sie zwei mögliche Operationen, die Sie nach der letzten programmierbaren Stufe vor Schreiben in den Framebuffer durchführen können und beschreiben Sie deren Zweck stichpunktartig. **(4 Punkte)**

Aufgabe 9: OpenGL - Tangentenraum (16 Punkte)



Für eine Beleuchtungsberechnung mit Normal-Mapping ist an einem Oberflächenpunkt ein orthonormaler Tangentenraum durch die Tangente $\boldsymbol{T}$, Normale $\boldsymbol{N}$, und Bitangente $\boldsymbol{B}$ gegeben. Die Blickrichtung $\boldsymbol{V}$ und die Lichtrichtung $\boldsymbol{L}$ sind bereits normalisiert und in Weltkoordinaten gegeben. Vervollständigen Sie das folgende GLSL Code Fragment, um das Phong-Beleuchtungsmodell im Tangetenraum auszuwerten! Verwenden Sie die gegebenen Texturkoordinaten, um die im Tangentenraum gespeicherte Normale aus der Normal-Map auszulesen! Die Oberflächennormale im Tangentenraum ist $(0, 1, 0)$. **(16 Punkte)**

```
uniform sampler2D tex_normal; // Normal-Map

vec3 computePhong(
    vec2 uv, // Texturkoordinaten zum Auslesen der Normal-Map
    vec3 T,  // normalisierte Tangente
    vec3 N,  // normalisierte Normale
    vec3 B,  // normalisierte Bitangente
    vec3 V,  // Blickrichtung in Weltkoordinaten
    vec3 L,  // Lichtrichtung in Weltkoordinaten
    vec3 Kd, // Diffuser Reflexionskoeffizient
    vec3 Ks, // Spekularer Reflexionskoeffizient
    float n) // Phong Exponent
{









}
```

Name: ______________________ Matrikelnummer: ______________________

Aufgabe 10: Bézier-Kurven (20 Punkte)

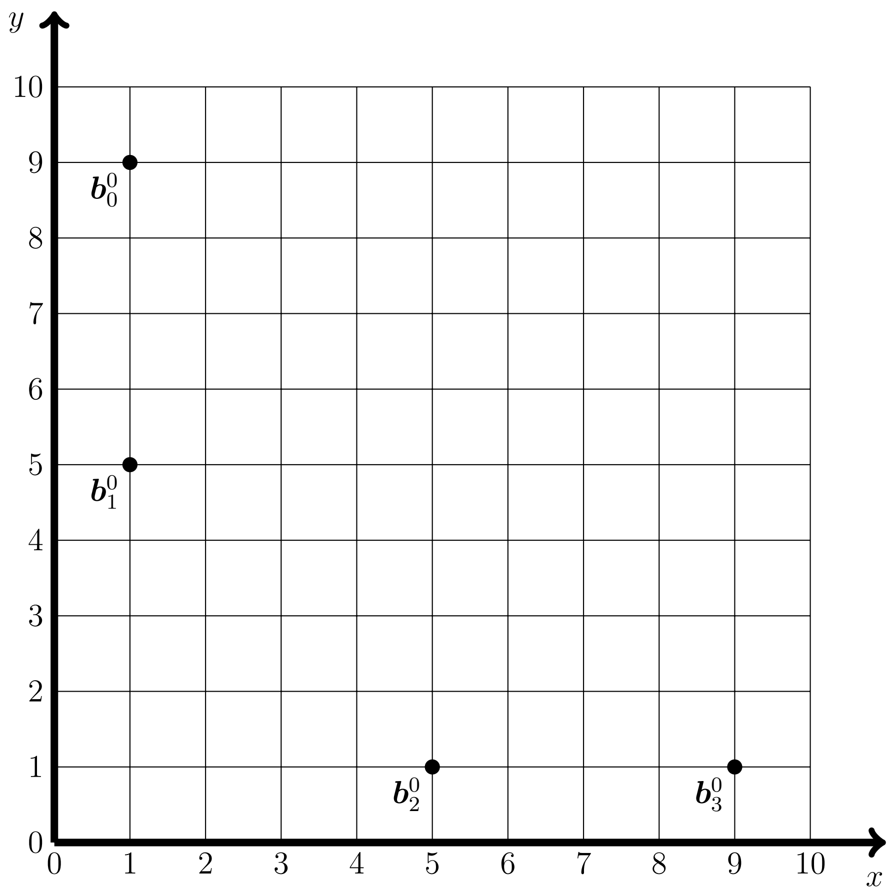
1. Wie zeichnet man eine Bézier-Kurve möglichst effizient, wenn man sie durch einen Linienzug approximiert? **(2 Punkte)**

2. Werten Sie die Bézier-Kurve mit Kontrollpunkten

$$\boldsymbol{b}_0^0 = \begin{pmatrix} 1 \\ 9 \end{pmatrix}, \ \boldsymbol{b}_1^0 = \begin{pmatrix} 1 \\ 5 \end{pmatrix}, \ \boldsymbol{b}_2^0 = \begin{pmatrix} 5 \\ 1 \end{pmatrix}, \ \boldsymbol{b}_3^0 = \begin{pmatrix} 9 \\ 1 \end{pmatrix}.$$

für $u = 0.5$ mit dem Algorithmus aus 1 in folgender Abbildung zeichnerisch aus! Skizzieren Sie zusätzlich die Bézier-Kurve! **(6 Punkte)**

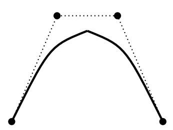
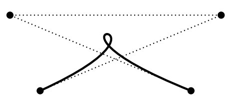
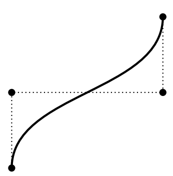
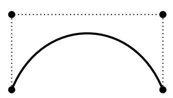
$P(0.5) = \boldsymbol{b}_0^3 =$



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

Aussage	**Wahr**	**Falsch**
Bei Bézier-Kurven beliebigen Grades haben sämtliche Kontrollpunkte einen globalen Einfluss auf die Kurve.	☐	☐
Zur Auswertung einer beliebigen Bézier-Kurve kann man mit linearen Interpolationen auskommen.	☐	☐
Bézier-Kurven von Grad n können immer in zwei Kurven von Grad $\lceil n/2 \rceil$ aufgeteilt werden.	☐	☐
Jede Bézier-Kurve ist auch ein Bézier-Spline.	☐	☐
Die Randkurven einer Tensorprodukt-Bézier-Fläche sind Bézier-Kurven.	☐	☐
Mit dem de Casteljau-Algorithmus benötigt die Unterteilung einer kubischen Bézier-Kurve nicht mehr Rechenoperationen als die Auswertung, um einen Funktionswert zu bestimmen.	☐	☐

4. 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**
	☐	☐	
	☐	☐	
	☐	☐	
	☐	☐	