

## Aufgabe 1: Beleuchtung

### Teilaufgabe 1a

Welche 4 Vektoren werden benötigt, um die reflektierte Farbe an einem Vertex bzw. Oberflächenpunkt mit dem Phong-Beleuchtungsmodell zu berechnen? Illustrieren Sie Ihre Antwort mit einer Skizze, die den Oberflächenpunkt, die Betrachter- und die Lichtquellenposition beinhaltet.

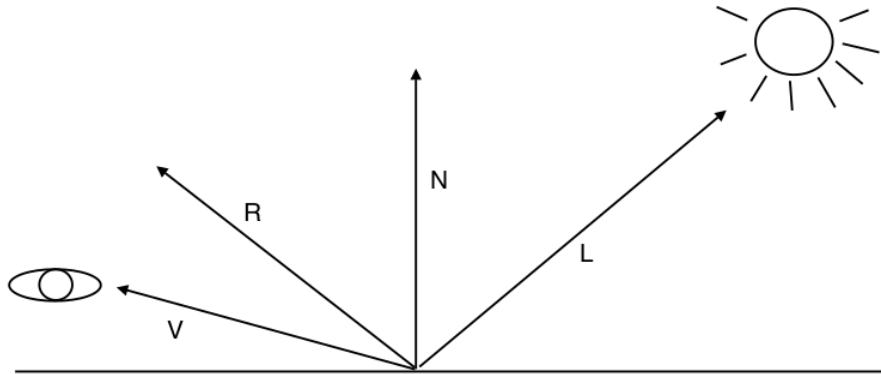


Abbildung 1: Whatever

1. Lichtvektor  $L$
2. Normale  $N$
3. View-Vektor  $V$
4. Reflektiertes Licht  $R_L$

### Teilaufgabe 1b

Beim Phong-Beleuchtungsmodell setzt sich das reflektierte Licht aus 3 Komponenten zusammen: ambient, diffus und spekulär.

Das Phong-Beleuchtungsmodell lautet:

$$I = \underbrace{k_a \cdot I_L}_{\text{ambient}} + \underbrace{k_d \cdot I_L \cdot (N \cdot L)}_{\text{diffus}} + \underbrace{k_s \cdot I_L \cdot (R_L \cdot V)^n}_{\text{spekulär}}$$

Erläutern Sie kurz, welche dieser Komponenten ihren Wert aus welchem Grund ändert bzw. ändern, wenn:

- der Vertex verschoben wird, aber die Lichtquelle- und Betrachterposition unverändert bleibt?

- ⇒ Diffus und Spekular ändern sich, da die diffuse Komponente von der Lichtrichtung abhängig ist und die spekulare Komponente vom Winkel des View-Vektors zum reflektierten Lichtvektor abhängig ist.
- *sich die Betrachterposition ändert, aber Lichtquelle und Vertex unverändert bleiben?*
- ⇒ Die spekulare Komponente ändert sich weil sie vom Winkel des View-Vektors zum reflektierten Lichtvektor abhängig ist.

### Teilaufgabe 1c

*Welche Aufgabe hat der Phong-Exponent? Welche Auswirkungen hat es, wenn sein Wert größer bzw. kleiner gewählt wird?* Imperfekte Spiegelung: Je größer der Phong-Exponent desto genauer die Spiegelung, heißt der spekulare Lichtfleck wird kleiner aber dafür intensiver.

### Teilaufgabe 1d

*Warum verbessert eine Unterteilung von Primitiven in kleinere die Darstellungsqualität, wenn Gouraud-Shading verwendet wird?*

Bei Gouraud-Shading wird die Farbe an den Vertices berechnet und dazwischen interpoliert. Dadurch können Glanzlichter verloren gehen, die in der Mitte von Primitiven liegen. Bei kleineren Primitiven ist die Wahrscheinlichkeit, dass dies passiert geringer. Außerdem ergibt die Interpolation genauere Ergebnisse, da die Vertices näher an den Punkten liegen, für die interpoliert wird

## Aufgabe 2: Blending

### Teilaufgabe 2a

$$\text{SRC\_ALPHA} \quad \text{(I) } \overbrace{0.5} \cdot 0.2 + \underbrace{(1 - 0.5)}_{1-\text{SRC\_ALPHA}} \cdot 0.4 = 0.3$$

(II) Auch 0.3, der der Alpha-Wert 0.5 ist.

(III) *Welche Konsequenzen hat dies für das Zeichnen von sich überlappenden semi-transparenten Objekten?*

Erfordert Tiefensortierung. Im allgemeinen ist Blending nicht kommutativ.

### Teilaufgabe 2b

*Ein Zaun wird mit OpenGL gezeichnet, indem ein GL\_QUAD -Primitiv mit einer Textur gezeichnet wird, die den Wert 1 enthält, wo sich Lücken im Zaun befinden, und sonst*

den Wert 0 enthält. Warum sollte man in diesem Fall Alpha-Testing statt Blending verwenden?

Der Tiefenalgorithmus sieht nicht, dass der Zaun an manchen Stellen durchsichtig ist und zeichnet deswegen Objekte die dahinter liegen nicht. Mit Alpha-Testing werden dahinterliegende Objekte gezeichnet.

### Teilaufgabe 2c

- Rufe **DrawScene** auf, Setze Source- und Destination-Faktor auf 1/Anzahl Lichtquellen,
- verschiebe Lichtquelle
- Rufe **DrawScene** auf (Blending der Ergebnisse)
- setze Source auf 2/Anzahl
- verschiebe Lichtquelle
- Rufe **DrawScene** auf (Blending der Ergebnisse)
- usw. bis alle Lichtquellen durch sind

## Aufgabe 3

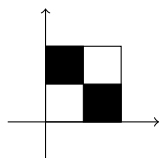
### Teilaufgabe 3a

- löst das Aliasing-Problem bei Minification
- Man berechnet Mipmap-Stufen, indem die Höhe und Breite der Textur jeweils halbiert wird
- Zugriff auf die Textur:  $\text{Texelgröße}(n) \leq \text{Größe des Pixelfootprints auf der Textur ; Texelgröße}(n+1)$

### Teilaufgabe 3b

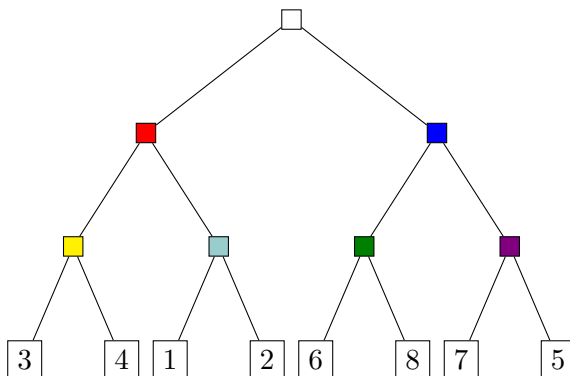
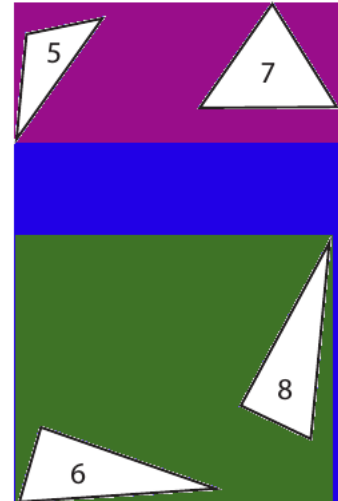
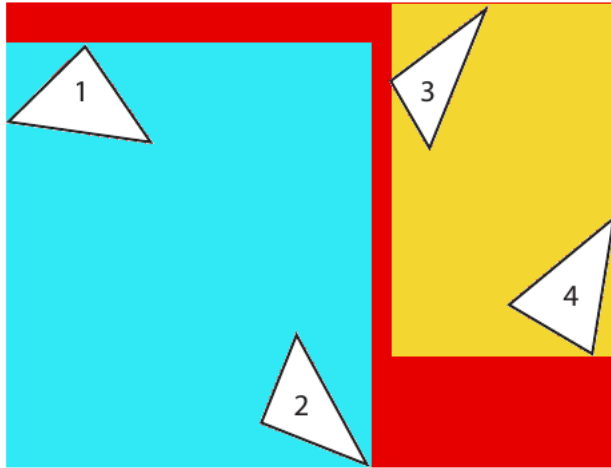
Darstellung reflektierender Objekte mit Spiegelung von Umgebung ohne geometrische Repräsentation. Annahme: nur Richtung des Strahls wird berücksichtigt, man geht davon aus, dass die Umgebung relativ weit weg ist

### Teilaufgabe 3c



## Aufgabe 4

### Teilaufgabe 4a



### Teilaufgabe 4b

Ziel: Bounding Boxes mit möglichst geringem Volumen

Die Wahrscheinlichkeit dafür dass man ein Primitiv trifft, wenn man die Bounding Box getroffen hat, soll groß werden. Konstruktion bezieht Kostenfunktion (Kosten für Schnitt mit Knoten) mit ein, damit die Kosten minimiert werden. Surface Area der Primitive soll einen großen Anteil der Bounding Box ausmachen. Eingeflossene Größen: Primitiv-Flächeninhalt, Surface Area der Box, Kosten der Traversierung eines Knotens im Verhältnis zu Kosten eines Strahl-Primitiv-Tests

### **Teilaufgabe 4c**

Suboptimal, wenn es Stellen gibt, an denen wenige Primitive liegen und welche mit vielen Primitiven auf wenig Raum  $\rightarrow$  es gib Boxen mit viel leerem Raum drin. Median ist gleich gut wie SAH wenn Primitive gleichmäßig verteilt sind.

### **Teilaufgabe 4d**

Erster Schnittpunkt mit einem Primitiv wird an Gitterzellen weitergegeben. Vorteil: man muss das gleiche Objekt nicht mehrfach schneiden.

## Aufgabe 5

---

```
1 uniform vec3 eye; //Position des Betrachters in Weltkoord
2 in vec3 dir;      //Strahlrichtung in Weltkoord
3 uniform vec3 spheres[10];
4 out vec4 fragColor;
5 int main() {
6     float t = -1;
7     vec3 pos;
8     vec3 normal;
9     for (i=0; i < 10; i++) {
10         vec2 isec = intersectRS(eye, dir, vec3(spheres[i]), spheres[i].z);
11         if (isec.x < t && isec.x > 0) {
12             t = isec.x;
13             pos = eye + t * dir;
14             normal = normalize(pos - vec3(spheres[i]));
15         }
16     }
17     if (t != -1) {
18         fragColor = computeShading(pos, normal);
19     }
20     // ...
21 }
```

---

## Aufgabe 6

### Teilaufgabe 6a

Polygone, die man nur von hinten sieht, werden nicht gezeichnet. Wird zur Beschleunigung verwendet, weil man dann weniger Polygone zeichnen muss.

### Teilaufgabe 6b

Ergebnisse der letzten Verarbeiteten Vertices werden gecacht. Nur sinnvoll, wenn Vertices mehrfach verwendet werden, z.B. durch Indizierung. Pipelinestufe: Primitive Assembly

### Teilaufgabe 6c

über Indizes

### Teilaufgabe 6d

Triangle Strips brauchen weniger Platz, weil sich Dreiecke Vertices teilen ( $n+2$  statt  $3n$  Vertices)

### Teilaufgabe 6e

Vertex Shader holt Textur, ordnet den Vertices Punkte auf den Texturen zu. Fragment Shader berechnet anhand dessen die Texturierung der Primitive

### Teilaufgabe 6f

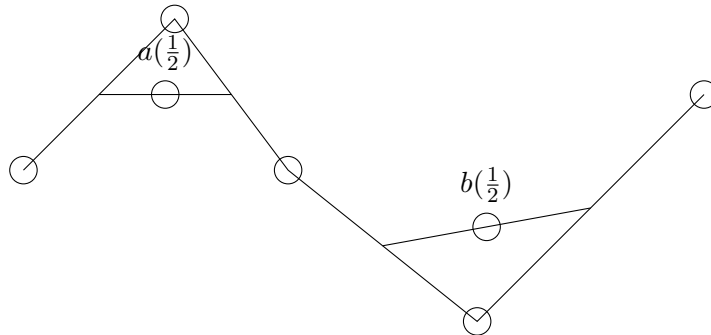
Aussage	Wahr	Falsch
Vertex Shader kann Vertices löschen und generieren		X
Vertex Shader kann Transformieren	X	
Sichtrichtung entlang negativer Y-Achse		X
mehrere Shader in einem Zeichenvorgang		X
inkonsistente Interpolation bei T-Vertices	X	

### Teilaufgabe 6g

1. B V
2. F V
3. E V
4. A N
5. C N
6. D F

## Aufgabe 7

### Teilaufgabe 7a



Tangenten an den Randpunkten zeigen auf den nächsten Punkt. 2fach stetig diffbar (runde Kurve, keine Ecken, keine Unterbrechungen). Kurve liegt innerhalb der konvexen Hülle der Kontrollpunkte.

### Teilaufgabe 7b

- $a_3 : C^1$ . Begründung:  $a_3 - a_2 = b_1 - b_0$ , aber  $a_2 + (a_2 - a_1) \neq b_1 + (b_1 - b_2)$
- $c_0 : C^0$ . Begründung:  $b_3 - b_2 \neq c_1 - c_0$

### Teilaufgabe 7c

1. bilden eine Basis des Polynomraumes
2. Rekursionsformel:

$$B_i^n(u) = u \cdot B_{i-1}^{n-1} + (1-u) \cdot B_i^{n-1}$$

3. symmetrisch
4. positiv ( $\geq 0$ ) auf  $[0,1]$