# Aufgabe 1: Das Phong-Beleuchtungsmodell

## Teilaufgabe 1a

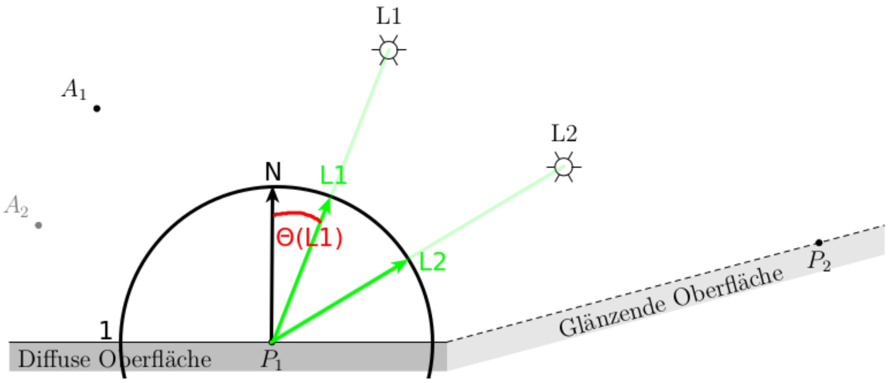



Abbildung 1: Skizze zu Aufgabe 1

## Teilaufgabe 1b

*Geben Sie mit Hilfe der benannten Vektoren die Formel an, mit der die Farbe der Oberfläche im Punkt $P_1$ bestimmt werden kann! Unterstreichen Sie dabei die Vektoren, die normiert sein müssen!*

$$I = \sum_{i=1}^{2} \overbrace{k_a \cdot I_{L_i}}^{=0} + k_d \cdot I_{L_i} \cdot (\underbrace{N \cdot L_i}_{\cos \Theta(L_i)}) + \overbrace{k_s \cdot I_{L_i} \cdot (R_{L_i} \cdot V)^n}^{=0}$$

Normalisierte Vektoren:

- TODO: Welche Vektoren müssen normalisiert sein? Nicht alle?

## Teilaufgabe 1c

*Welche der beiden Lichtquellen trägt mehr zur Intensität der Oberfläche an $P_1$ bei?*

$L_1$ trägt mehr zur Intensität $I$ von $P_1$ bei, weil $\cos \Theta(L_1)$ größer ist als $\cos \Theta(L_2)$.

## Teilaufgabe 1d

*Wie verändert sich die Intensität von $P_1$, wenn er statt von $A_1$ nun von $A_2$ betrachtet wird?*

$I$ ändert sich nicht, weil keiner der Terme die zu $I$ in diesem Fall beitragen vom $A_1$ bzw. $A_2$ abhängig ist.

### Teilaufgabe 1e

*Es soll nun die Farbe des Punktes $P_2$ bestimmt werden. Dieser befindet sich auf der glänzenden Oberfläche. Zeichnen Sie für $P_2$ die Vektoren ein, die zur Bestimmung seiner Farbe benötigt werden! Beschränken Sie sich dabei auf die Vektoren für $L_1$.*

TODO

### Teilaufgabe 1f

*Wie verändert sich die Intensität von $P_2$, wenn er statt von $A_1$ von $A_2$ betrachtet wird? Begründen Sie Ihre Antwort kurz!*

$I$ wird schwächer, da die spekulare Reflexion mit zunehmendem Winkel zwischen $\mathbf{v}$ und $\mathbf{r}_L$ abfällt.

## Aufgabe 2: Raytracing

### Teilaufgabe 2a

- Anstelle einen Punkt für einen Pixel abzutasten, tastet man $k^2$ mal in äquidistanten Intervallen ab.
- Aliasing wird dadurch verringert.

### Teilaufgabe 2b

- Maximale Rekursionstiefe erreicht
- Rekursion bis der Beitrag zur Farbe vernachlässigbar wird

### Teilaufgabe 2c

*Was ist der Unterschied zwischen Distributed Raytracing und Whitted-Style Raytracing?*

Beim Distributed Raytracing wird nicht nur ein Schattenstrahl verschickt, sondern viele. Damit sollen zu perfekte Spiegelungen/Transmissionen vermieden werden.

*Welchen Lichttransport kann man durch Distributed Raytracing berechnen, den Whitted-Style Raytracing nicht erfassen kann?*

Siehe Kapitel 2 (Raytracing), Folie 107 ff.:

- Kaustiken
- Weiche Schatten
- Tiefenunschärfe
- Bewegungsunschärfe

### Teilaufgabe 2d

*Nennen Sie kurz und stichpunktartig die zwei Schritte, die zur Berechnung von Vertex-Normalen bei einem Dreiecksnetz notwendig sind! Gehen Sie dabei davon aus, dass nur die Vertex-Positionen und die Topologie des Netzes gegeben sind!*

- Kreuzprodukt über $(P_2 - P_1)$ und $(P_3 - P_1)$ für drei Punkte eines Dreiecks
- Normalisieren

# Aufgabe 3: Farben und Farbwahrnehmung

## Teilaufgabe 3a

### Teilaufgabe 3a (I)

*Wie berechnet man die Sensorantwort a für ein Spektrum $S(\lambda)$?*

$$a(S(\lambda)) = \int_\lambda E(\lambda) \cdot S(\lambda) \mathrm{d}\lambda$$

### Teilaufgabe 3a (II)

Unter einem *Metamerismus* versteht man das Phänomen, das unterschiedliche Spektren den selben Farbeindruck vermitteln können. Es muss also

$$a_1 = a_2$$

gelten, damit $S_1(\lambda)$ und $S_2(\lambda)$ bzgl. der gegebenen Kamera Metamere sind.

## Teilaufgabe 3b

1. Das HSV-Farbmodell trennt Farbton von Helligkeit.

⇒ Richtig ($H$ue (Farbton), $S$aturation (Sättigung), $V$alue (Hellwert)).

2. Der Farbeindruck einer additiv gemischten Farbe hängt nicht vom Farbeindruck der Ausgangsfarben ab.

⇒ Falsch. Vgl. Superpositionsprinzip, Kapitel 1, Teil 2, Folie 6.

3. Farbige Flächen werden unabhängig von ihrer Umgebung vom menschlichen Auge immer gleich wahrgenommen.

⇒ Falsch. (z.B. durch Machsche Bandeffekte)

4. Der Machsche Bandeffekt ist vor allem bei Phong-Shading ein Problem.

⇒ Falsch. Machsche Bänder entstehen an den Grenzen von Flächen, die jeweils keine Farbgraduierung haben. Beim Phong-Shading gibt es wegen der interpolierten Normalen keine solchen Flächen. Der Effekt tritt z. B. beim Flat-Shading auf, ist also vor allem dort problematisch.

## Aufgabe 4: Bézier-Kurven

### Teilaufgabe 4a

*Gegeben sei die Bézier-Kurve* $\mathbf{b}(u) = \sum_{i=0}^{3} \mathbf{b}_i B_i^3(u)$ *mit den Kontrollpunkten* $\mathbf{b}_i$*, wobei* $u \in [0, 1]$ *und* $B_i^3$ *das* $i$*-te Bernstein-Polynom vom Grad 3 ist.*

#### Teilaufgabe 4a (I)

*Werten Sie die Bézier-Kurve zeichnerisch mit dem de-Casteljau-Algorithmus an der Stelle* $u = \frac{1}{3}$ *aus! Markieren Sie den Punkt* $\mathbf{b}\left(\frac{1}{3}\right)$*!*

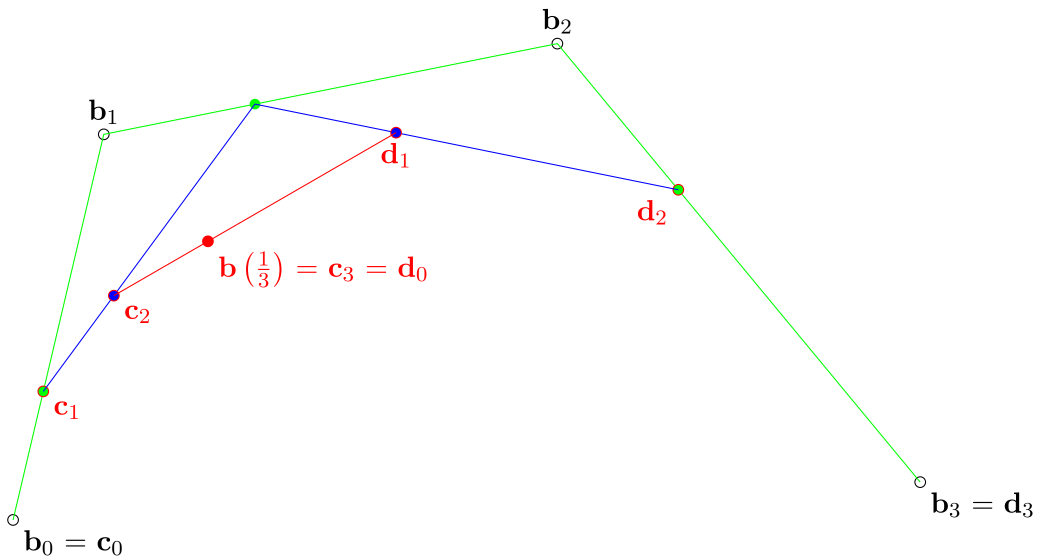



Abbildung 2: Skizze zu Aufgabe 4a und 4b

#### Teilaufgabe 4a (II)

vgl. Abbildung 2

### Teilaufgabe 4b

Siehe Nachklausur 2015, Aufgabe 11b für eine detaillierte Erklärung.

1. Nein, da die Kontrollpunkte auf den Ecken eines Rechtecks liegen, aber die Kurve nicht symmetrisch ist.
2. Nein, da die Kurve nicht in der konvexen Hülle der Kontrollpunkte liegt.

3. Ja
4. Nein, da die Kurve nicht tangential an $b_0 b_1$ ist.

## Aufgabe 5: Transformationen

Der Basiswechsel ist eine Verschiebung um $\begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix}$ und dann eine Rotation um 180° um die $x$-Achse. Daher:

$$M = \overbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 180^\circ & -\sin 180^\circ & 0 \\ 0 & \sin 180^\circ & \cos 180^\circ & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}^{\text{Rotation}} \cdot \overbrace{\begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}}^{\text{Translation}} \tag{1}$$

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2}$$

$$M = \begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3}$$

## Aufgabe 6: Texturierung

### Teilaufgabe 6a

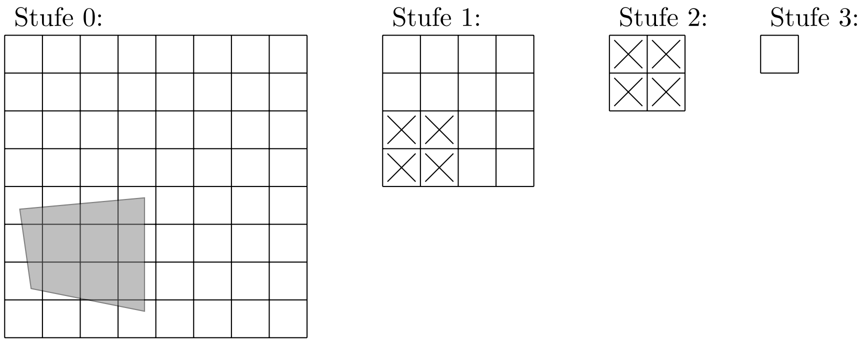
$$\lambda_A = \frac{A_\Delta(P,B,C)}{A_\Delta(A,B,C)} = \frac{3}{6} \qquad \lambda_B = \frac{A_\Delta(P,A,C)}{A_\Delta(A,B,C)} = \frac{1}{6} \qquad \lambda_C = \frac{A_\Delta(P,A,B)}{A_\Delta(A,B,C)} = \frac{2}{6} \tag{4}$$

$$\begin{pmatrix} u \\ v \end{pmatrix} = \lambda_A \cdot \begin{pmatrix} 4 \\ 0 \end{pmatrix} + \lambda_B \cdot \begin{pmatrix} 12 \\ 6 \end{pmatrix} + \lambda_C \cdot \begin{pmatrix} 0 \\ 12 \end{pmatrix} = \begin{pmatrix} 4 \\ 5 \end{pmatrix} \tag{5}$$

### Teilaufgabe 6b

Siehe Kapitel 4, Folie 56

*Gegeben ist eine Textur mit $8 \times 8$ Texeln und 4 Mipmap-Stufen sowie ein Pixel-Footprint und dessen Mittelpunkt (siehe Abbildung). Kreuzen Sie in der Mipmap-Pyramide die Texel an, die für eine trilineare Interpolation zur Bestimmung des Farbwertes verwendet werden! Begründen Sie kurz Ihre Wahl der Mipmap-Stufen!*




Die Größe des Footprints liegt zwischen der Texelgröße der Stufen 1 und 2.

### Teilaufgabe 6c

*Welchen Vorteil haben Summed-Area-Tables gegenüber Mipmaps bei der Texturfilterung?*

Die Textur kann mit einem beliebigen rechteckigen Filter in konstanter Zeit gefiltert werden (Kapitel 4, Folie 66).

### Teilaufgabe 6d

Das Interpolationsschema heißt *Bilineare Interpolation* und funktioniert wie folgt:

$$t_{12} = (1 - a) \cdot t_1 + a \cdot t_2 \tag{6}$$
$$t_{34} = (1 - a) \cdot t_3 + a \cdot t_4 \tag{7}$$
$$t = (1 - b) \cdot t_{34} + b \cdot t_{12} \tag{8}$$

## Aufgabe 7: Cube-Maps und Environment-Mapping

### Teilaufgabe 7a

#### Teilaufgabe 7a (I)

*Wie wird die Cube-Map-Seite bestimmt, auf die zugegriffen wird? Welche ist es für* $\mathbf{r}$*?* Es wird die betragsmäßig größte Komponente gewählt. Diese bestimmt ob es oben/unten oder links/rechts oder vorne/hinten wird. Das Vorzeichen bestimmt dann per Konvention die konkrete Fläche.

Im vorliegenden Fall ist $|r_z|$ am größten, also ist es Fläche 1.

### Teilaufgabe 7a (II)

*Berechnen Sie die Texturkoordinaten des Zugriffs auf der für* **r** *ausgewählten Cube-Map-Seite!*

$$s = \frac{1}{2} + \frac{r_x}{2 \cdot r_z} = \frac{1}{4} \tag{9}$$

$$t = \frac{1}{2} + \frac{r_y}{2 \cdot r_z} = \frac{3}{4} \tag{10}$$

### Teilaufgabe 7a (III)

Vorteile von Cube-Maps gegenüber Sphere-Maps (Quelle):

- Keine Bildverzerrung
- Unabhängigkeit vom Viewpoint $P$
- Schneller Berechenbar

$\Rightarrow$ Besser für Echtzeit-Rendering geeignet.

## Teilaufgabe 7b

### Teilaufgabe 7b (I)

*Was wird in einer Environment-Map gespeichert?*

Ein Bild der Umgebung in einer Textur.

### Teilaufgabe 7b (II)

*Nennen Sie ein Anwendungsbeispiel für Environment-Maps!*

Berechnung von Reflexion mit Spiegelung der Umgebung ohne geometrische Repräsentation

### Teilaufgabe 7b (III)

*Welche grundlegende Annahme wird bei Environment-Mapping gemacht?*

Umgebung ist weit entfernt, nur die Richtung $\mathbf{r}$ wird verwendet, der Ausgangspunkt $P$ wird ignoriert.

### Teilaufgabe 7b (IV)

*Was bzw. welcher Effekt kann mit vorgefilterten Environment-Maps nicht korrekt dargestellt werden?*

Selbstverschattung

## Aufgabe 8: Hierarchische Datenstrukturen

### Teilaufgabe 8a

Reihenfolge der Schnittests:

A B 14 15 C 5 6 11

### Teilaufgabe 8b

1. räumliches Mittel (Spatial Median)

→ Nein, da $B$ nicht in der Mitte liegt

2. Objektmittel (Object Median)

→ Nein, da $B$ genau 8 Objekte auf der einen Seite und nur 2 Objekte auf der anderen Seite hat.

3. Kostenfunktion (Surface Area Heuristic)

→ Ja.

### Teilaufgabe 8c

*Nennen Sie jeweils eine Stärke und eine Schwäche der Aufteilung mittels Kostenfunktion (Surface Area Heuristic)!*

Vorteil Gut konstruierte kD-Bäume können deutlich schneller sein als schlecht konstruierte

Nachteil Konstruktion aufwendig

### Teilaufgabe 8d

| Aussage | BVH | Octree | kD-Baum | Gitter |
|---|---|---|---|---|
| Datenstruktur wird an Geometrie angepasst | Ja (Hüllkörper) | Ja (Rekursionstiefe) | Ja (wo Ebenen liegen) | Nein |
| Raum wird immer achsenparallel unterteilt | Nein (nur mit AABBs) | Ja | Ja | Ja |
| Datenstruktur ist Binärbaum[1] | Nein | Nein | Ja | Nein |
| Speicherplatz der Datenstruktur ist abhängig von der Anzahl der Primitive | Ja | Ja | Ja | (Ja) (je nachdem wie gespeichert wird) |
| Bei der Konstruktion kann die SAH sinnvoll eingesetzt werden | Ja[2] | Nein | Ja | Nein |

## Aufgabe 9: Rasterisierung und OpenGL

| # | **Aussage** | **Wahr** | **Falsch** |
|---|---|---|---|
| 1 | Die Präzision des Tiefenpuffers wird verringert, wenn man die Distanz zwischen Near-Plane und Far-Plane vergrößert. | ☑ | ☐ |
| 2 | Die OpenGL-Pipeline nutzt den Vertex-Cache beim Zeichnen ohne Index-Puffer. | ☐ | ☑ |
| 3 | OpenGL-Puffer vom Typ `GL_ELEMENT_ARRAY_BUFFER` werden für das Rendering mit der Shared-Vertex-Repräsentation verwendet. | ☑ | ☐ |
| 4 | Ein Alpha-Test kann im Fragment-Shader mit dem Befehl `discard` implementiert werden. | ☑ | ☐ |
| 5 | Die OpenGL-Funktion `glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)` kann verwendet werden, um additives Blending zu implementieren. | ☐ | ☑ |
| 6 | Shadow-Mapping benötigt den Stencil-Puffer. | ☐ | ☑ |
| 7 | Für die Transformation einer Normalen im Vertex-Shader kann immer dieselbe Matrix wie zur Transformation der entsprechenden Vertizes verwendet werden. | ☐ | ☑ |
| 8 | Beim Gouraud-Shading werden die im Vertex-Shader berechneten Farben pro Fragment linear interpoliert. | ☑ | ☐ |

[2] Üblicherweise ist ein BVH ein Binärbaum, muss aber nicht so sein.
[2] Foliensatz 5 Folie 98

## Aufgabe 10: Tiefenpuffer und Transparenz

### Teilaufgabe 10a

*In welcher Reihenfolge müssen die drei Funktionen aufgerufen werden, um eine korrekte Darstellung des Hauses zu erhalten?*

1. `LöscheTiefenPuffer` (1)
2. `ZeichneWände` (3)
3. `ZeichneFenster` (2)

### Teilaufgabe 10b

| Sortierung | keine | vorne nach hinten | hinten nach vorne | Begründung |
|---|---|---|---|---|
| `ZeichneWände` | ☑ | ☐ | ☐ | Tiefentest sorgt für korrekte Verdeckung |
| `ZeichneFenster` | ☐ | ☐ | ☑ | Blending ist i. A. nicht kommutativ |

### Teilaufgabe 10c

| Sortierung | Tiefentest | `EQUAL` | `LESS` | `GREATER` | Tiefe schreiben | Blending |
|---|---|---|---|---|---|---|
| `ZeichneWände` | ☑ | ☐ | ☑ | ☐ | ☑ | ☐ |
| `ZeichneFenster` | ☑ | ☐ | ☑ | ☐ | ☐ | ☑ |

# Aufgabe 11: Phong-Shading und Phong-Beleuchtungsmodell

## Teilaufgabe 11a

shader.vert

```
uniform mat4 matN;   // Normalentransformation (Objekt -> Kamera)
uniform mat4 matM;   // Modelltransformation
uniform mat4 matV;   // Kameratransformation
uniform mat4 matP;   // Projektionstransformation
uniform mat4 matMV;  // Model-View-Matrix
uniform mat4 matMVP; // Model-View-Projection-Matrix

in vec3 P; // Eingabe-Vertex in Objektkoordinaten
in vec3 n; // Eingabenormale in Objektkoordinaten

out vec3 P_k; // Vertex-Position in Kamerakoordinaten
out vec3 n_k; // Vertex-Normale in Kamerakoordinaten

void main() {
    P_k = vec3(matMV * vec4(P, 1.0));
    n_k = vec3(matN * vec4(n_k, 0.0));
    gl_Position = matMVP * vec4(P, 1.0);
}
```

## Teilaufgabe 11b

shader.frag

```
uniform vec3 L; // Lichtposition in Kamerakoordinaten

// Materialparameter
uniform vec3 ka, kd, ks;
uniform float pexp; // Phong Exponent

// Intensität der Lichtquelle
uniform vec3 intensity;

in vec3 P_k;
in vec3 n_k;

void main()
{
    vec3 n = n_k;
    vec3 v = normalize(-P_k); /* Kamera im Ursprung vec3(0.0) */
    vec3 l = normalize(L - P_k);
    vec3 r = reflect( -l, n );
    float cosAlpha = max(0.0, dot(v, r));
    vec3 diffuse = kd * max(0.0, dot(l, n));
    vec3 specular = ks * pow(cosAlpha, pexp);
    gl_FragColor = vec4(intensity * (ka + diffuse + specular), 1.0);
}
```

## Aufgabe 12: Deformation mit Skelettsystemen

skelett.vert

```
in vec3 P; // Position des Vertex in Objektkoordinaten.
in float w[3]; // Einfluss der Knochen.

uniform mat4 matVP; // View-Projection-Matrix.
uniform mat4 M1; // Transformation des Fingers (Objekt -> Welt).
uniform mat4 M2; // Transformation von K2 relativ zu K1.
uniform mat4 M3; // Transformation von K3 relativ zu K2.

void main()
{
    vec4 P1 = M1 * vec4(P, 1.);
    vec4 P2 = M2 * P1;
    vec4 P3 = M3 * P2;
    gl_Position = matVP * (w[0] * P1 + w[1] * P2 + w[2] * P3);
}
```