

Aufgabe 1

Teilaufgabe 1a

Der Gamut ist der Raum aller vom Monitor darstellbaren Farben.

Teilaufgabe 1b

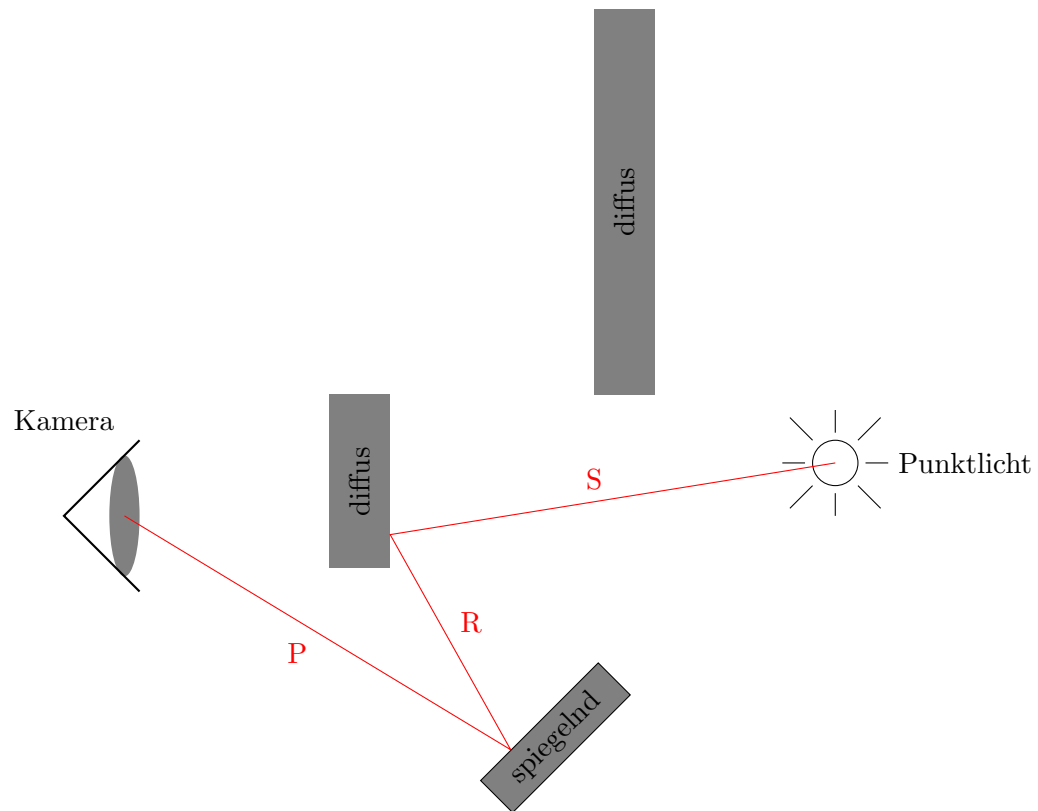
Aussage	RGB	CMY	HSV	CIE xyY
Der Farbraum ist additiv.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Der Farbraum ist subtraktiv.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Der Farbraum trennt Helligkeit von Farbe.	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Der Farbraum kann alle für den Menschen sichtbaren Farben repräsentieren.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Der Farbraum wird nativ auf Peripheriegeräten verwendet.	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Teilaufgabe 1c

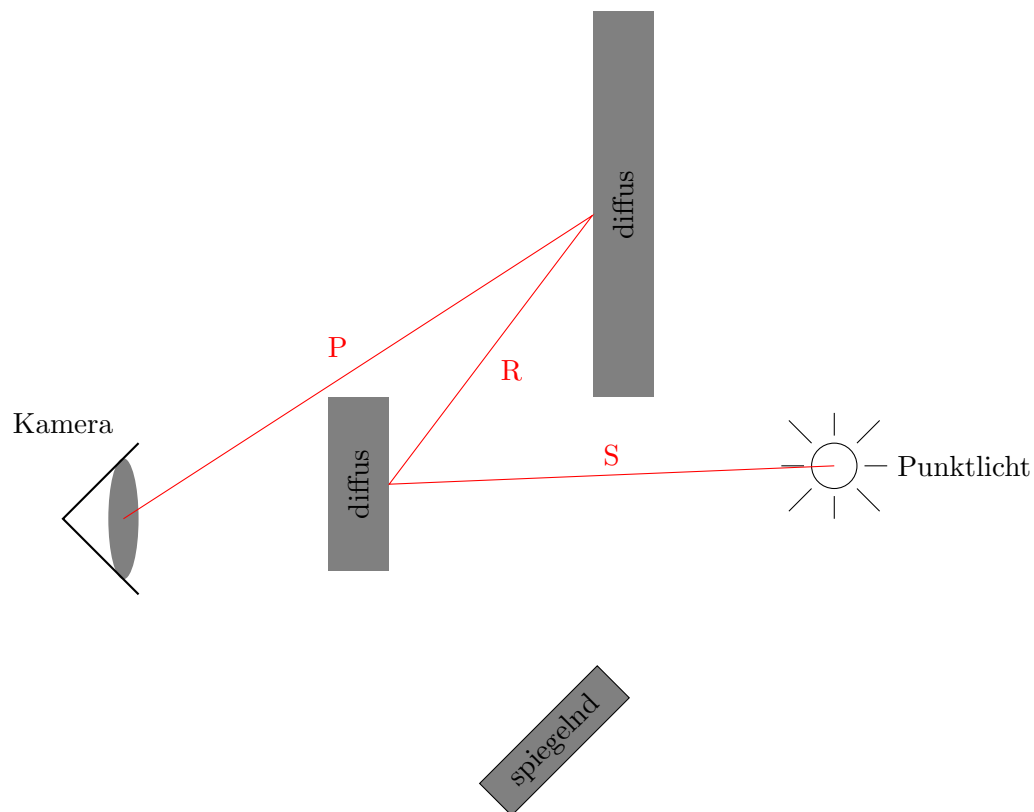
Nein (negative Energieabstrahlung nicht möglich)

Aufgabe 2

Teilaufgabe 2a



Teilaufgabe 2b



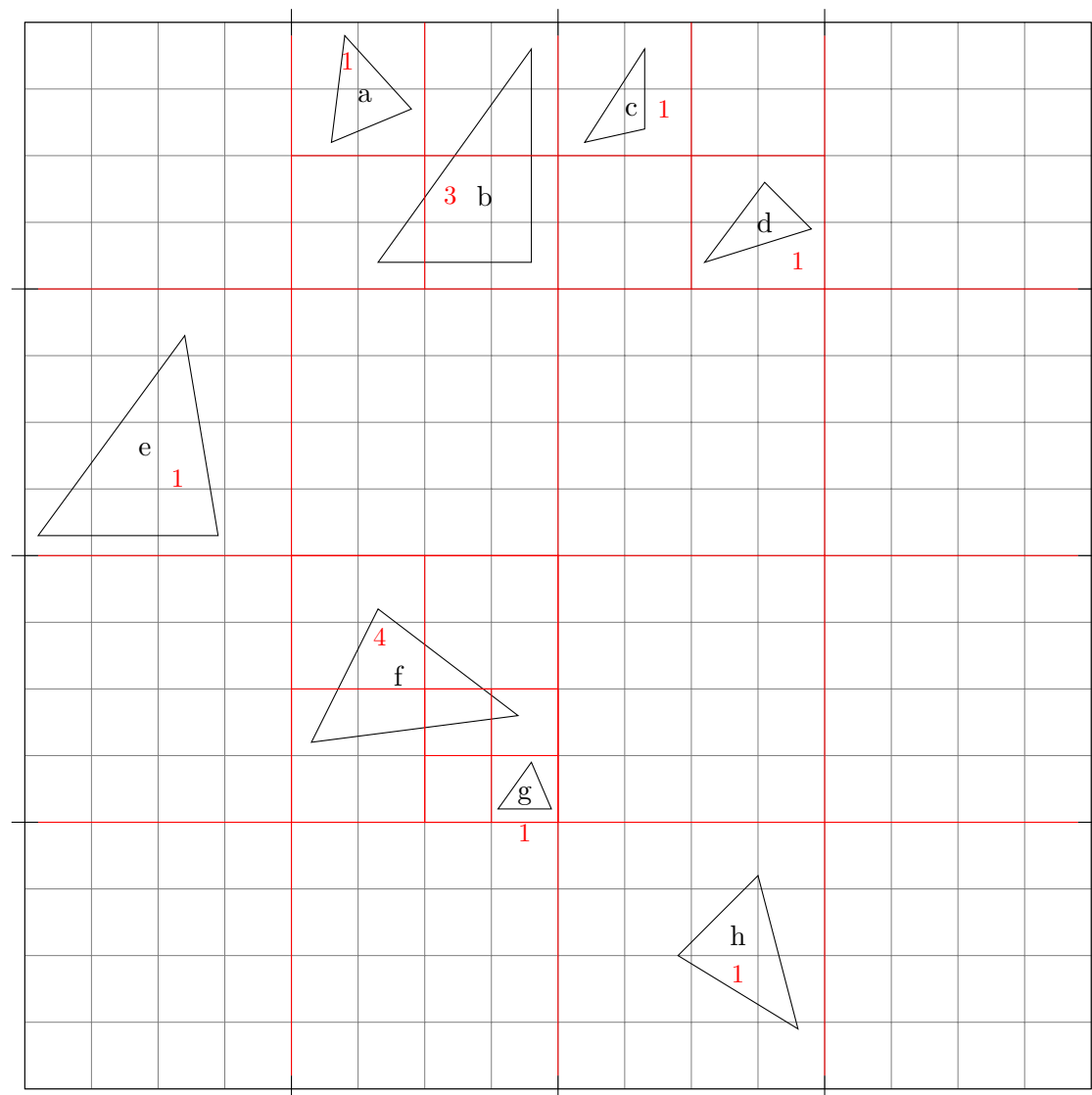
keine Verfolgung von Reflektionsstrahlen an diffusen Oberflächen (indirekte Beleuchtung)

Teilaufgabe 2c

	Aussage	Wahr	Falsch
1	Wenn ein Lichtstrahl aus einem optisch dichteren Medium in ein optisch dünneres Medium übergeht, kann Totalreflexion auftreten.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2	Die Zeitkomplexität von Whitted-style Raytracing ist polynomial in der Rekursionstiefe.	<input type="checkbox"/>	<input checked="" type="checkbox"/>
3	Es gibt eine Klasse von Materialien, die beim Whitted-style Raytracing mehrere Sekundärstrahlen per Schnittpunkt erzeugen.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
4	Der Whitted-style Raytracer überprüft die direkte Beleuchtung eines Oberflächenpunktes mithilfe von Schattenstrahlen.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
5	Whitted-style Raytracing konvergiert immer zum physikalisch korrekten Ergebnis.	<input type="checkbox"/>	<input checked="" type="checkbox"/>
6	Whitted-style Raytracing ist das Bildsyntheseverfahren, das auf der GPU in Hardware implementiert ist.	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Aufgabe 3

Teilaufgabe 3ab



Teilaufgabe 3c

e, f, f, f, h

Teilaufgabe 3d

Aussage	BVH	kD-Baum	Gitter	Keine
Primitive können in mehr als einem Blattknoten/einer Gitterzelle vorhanden sein.	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Die Datenstruktur kann zur Beschleunigung von Nachbarschaftssuchen verwendet werden.	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Die Datenstruktur eignet sich besonders für Szenen, in denen die Geometrie gleichmäßig verteilt ist.	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Der Aufbau-Algorithmus passt die Datenstruktur an die Normalen der Geometrie an.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Teilaufgabe 3e

Aussage	Wahr	Falsch
1 Die Surface Area Heuristic minimiert die Anzahl der Primitive, die sich in der Beschleunigungsstruktur befinden.	<input type="checkbox"/>	<input checked="" type="checkbox"/>
2 Der Aufbau einer BVH mithilfe der Surface Area Heuristic ist im Allgemeinen aufwändiger als der Aufbau mittels Objektmittel-Methode (object median).	<input checked="" type="checkbox"/>	<input type="checkbox"/>
3 BSP-Bäume sind kD-Bäume mit achsparallelen Unterteilungsebenen.	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4 Die Objektmittel-Methode (object median) kann beim Erstellen von BVH und kD-Baum verwendet werden.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
5 Die Raummittel-Methode (spatial median) kann beim Erstellen von BVH und kD-Baum verwendet werden.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
6 Eine optimale objektorientierte Box (OOBBs) hat höchstens das Volumen der entsprechenden achsenparallelen Box (AABB).	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Aufgabe 4

Teilaufgabe 4a

$$c = \lambda_1 c_1 + \lambda_2 c_2 + \lambda_3 c_3$$

Teilaufgabe 4b

Die interpolierte Normale muss erneut normalisiert werden.

Teilaufgabe 4c

- $\lambda_1 = \frac{2}{3}$
- $\lambda_2 = 0$
- $\lambda_3 = \frac{1}{3}$

Teilaufgabe 4d

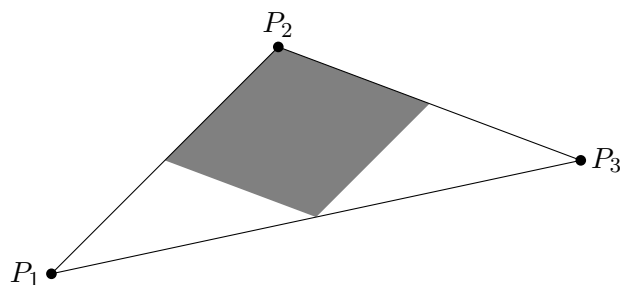
Das Phong-Beleuchtungsmodell kann für alle Verfahren eingesetzt werden.

Aufgabe 5

Teilaufgabe 5a

$$c_{BL}(T_p) = 0.4 \cdot (100 \cdot 0.1) + 0.8 \cdot (100 \cdot 0.9) = 4 + 72 = 76$$

Teilaufgabe 5b



Teilaufgabe 5c (i)

Stufe 1 (1 Texel): $\frac{1}{4} \cdot 0 + \frac{1}{4} \cdot 100 + \frac{1}{4} \cdot 0 + \frac{1}{4} \cdot 100 = 50$

Teilaufgabe 5c (ii)

$$c_{MM}(T_p) = \frac{1}{2} \cdot 50 + \frac{1}{2} \cdot 76 = 25 + 38 = 63$$

Teilaufgabe 5c (iii)

Fall	Nearest	Bilinear	Mip-Map-Nearest	Mip-Map-Trilinear
Minification	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Magnification	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Aufgabe 6

Teilaufgabe 6a

GL_TRIANGLES 0,1,3,0,2,3,1,2,3
GL_TRIANGLE_STRIP 0,1,3,2,0
GL_TRIANGLE_FAN 3,1,0,2,1

Teilaufgabe 6b

Vertex-Shader → Geometry-Shader → Fragment-Shader

Aufgabe 7

Teilaufgabe 7a

```
shader7a.frag
1 uniform int W;    // Breite des Bildes in Pixeln
2 uniform int H;    // Höhe des Bildes in Pixeln
3 uniform vec2 C0;   // Zentrum der 1. Kreisscheibe in Pixeln in  $[0, W) \times [0, H)$ 
4 uniform vec2 C1;   // Zentrum der 2. Kreisscheibe in Pixeln in  $[0, W) \times [0, H)$ 
5 uniform float R;   // Radius der Kreisscheiben in Pixeln
6
7 in vec2 uv;        // interpolierte Texturkoordinaten des Fragments in  $[0, 1) \times [0, 1)$ 
8
9 in vec3 inColor;    // Eingabefarbe des Fragments
10 out vec3 outColor; // Ausgabefarbe des Fragments
11
12 void main()
13 {
14     vec2 P = vec2(W, H) * uv;
15
16     if (distance(C0, P) > R && distance(C1, P) > R)
17         discard;
```

```
18
19     outColor = inColor;
20 }
```

Teilaufgabe 7b

TODO

Aufgabe 8

Teilaufgabe 8a

TODO

Teilaufgabe 8b

```
1 void renderScene()
2 {
3     glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
4     // Initialisieren Sie hier den OpenGL-Zustand!
5     glDepthMask( GL_TRUE );    /* (1) */
6     glEnable( GL_DEPTH_TEST ); /* (3) */
7     glDisable( GL_BLEND );     /* (6) */
8
9     // Zeichnen Sie die Szene ab hier!
10    drawOpaque();               /* (12) */
11    glDepthMask( GL_FALSE );    /* (2) */
12    glEnable( GL_BLEND );       /* (5) */
13    sortTransBackToFront();     /* (7) */
14    drawTrans();                /* (9) */
15 }
```

Teilaufgabe 8c

```
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
```


Teilaufgabe 8d

```
glBlendFunc(GL_ONE, GL_ONE);
```

Aufgabe 9

Teilaufgabe 9a

```
shader9a.frag
1 #define TERRAIN 1    // Index des Terrain-Objekts
2 #define WATER 2     // Index des Wasser-Objekts
3 #define NOTHING 0   // Kein Objekt
4
5 // Berechnet Abstand zwischen p und nächstgelegener Oberfläche ...
6 float distTerrain(vec3 P) {...}    // ... der Landschaft
7 float distWater(vec3 P) {...}      // ... der Wasseroberfläche
8
9 uniform float tmin; // Minimale Schnittdistanz, ab der gesucht wird
10 uniform float tmax; // Maximale Schnittdistanz, bis zu der gesucht wird
11 uniform float eps;  // Oberflächendistanz, ab der Objekt als geschnitten gilt
12
13 // Berechne Abstand des nächstgelegenen Schnittes mit Terrain oder Wasser
14 float intersect(
15     in vec3 O,          // Strahlursprung
16     in vec3 d,          // Normierte Strahlrichtung
17     out int oidx)       // Geschnittenes Objekts (NOTHING, TERRAIN oder WATER)
18 {
19     float t = tmin;
20
21     while (t < tmax) {
22         vec3 P = O + t * d;
23         float dt = distTerrain(P);
24         float dw = distWater(P);
25
26         float d;
27         if (dt < dw) {
28             d = dt;
29             oidx = TERRAIN;
30         } else {
31             d = dw;
32             oidx = WATER;
33         }
34     }
35 }
```

```

34
35     if (d < eps) return t;
36     t += d;
37 }
38
39 // Kein Schnittpunkt
40 oidx = NOTHING;
41 return tmax;
42 }

```

Teilaufgabe 9b

```

_____ shader9b.frag _____
1 // Berechnet Oberflächennormale der Distanzfelder
2 vec3 normalAt(vec3 P) {...}
3
4 // Berechnet Farbe ...
5 vec3 skyColor(vec3 viewDir) {...} // ... des Himmels
6 vec3 terrainColor(                // ... der Landschaft
7     vec3 P,                        // - Position
8     vec3 normal) {...}             // - Normale
9 vec3 waterColor(                  // ... des Wassers
10    vec3 normal,                   // - Normale
11    vec3 viewDir,                  // - eingehende Strahlrichtung
12    vec3 reflectionColor,          // - Farbe des Reflexionsstrahls
13    vec3 transmissionColor) {...} // - Farbe des Refraktionsstrahls
14
15 // Berechne Schnittfarbe
16 vec3 computeColor(
17     in vec3 O, // Strahlursprung
18     in vec3 d) // Strahlrichtung
19 {
20     int oidx;
21     float t = intersect(O, d, oidx);
22
23     if (oidx == NOTHING) {
24         return skyColor(d);
25     }
26
27     vec3 P = O + t * d;
28     vec3 n = normalAt(P);
29     if (oidx == TERRAIN) {
30         return terrainColor(P, n);

```

```

31     }
32
33     vec3 r = reflect(d, n); //Order: 2 * n * dot(-d, n) + d;
34     return waterColor(n, d, skyColor(r), computeColor(P, d));
35 }

```

Aufgabe 10

Teilaufgabe 10a

$$\begin{aligned}
 & (1 - u^2)\mathbf{b}_0 + 2u(1 - u)\mathbf{b}_1 + u^2\mathbf{b}_2 \\
 &= (1 - 2u + u^2)\mathbf{b}_0 + (2u - 2u^2)\mathbf{b}_1 + u^2\mathbf{b}_2 \\
 &= \mathbf{b}_0u^2 - 2\mathbf{b}_1u^2 + \mathbf{b}_2u^2 - 2\mathbf{b}_0u + 2\mathbf{b}_1u + \mathbf{b}_0 \\
 &= \underbrace{(\mathbf{b}_0 - 2\mathbf{b}_1 + \mathbf{b}_2)}_{=\mathbf{a}_2}u^2 + \underbrace{(2\mathbf{b}_1 - 2\mathbf{b}_0)}_{=\mathbf{a}_1}u + \underbrace{\mathbf{b}_0}_{=\mathbf{a}_0}
 \end{aligned}$$

Teilaufgabe 10b

