Aufgabe 1: Raytracing

Teilaufgabe 1a

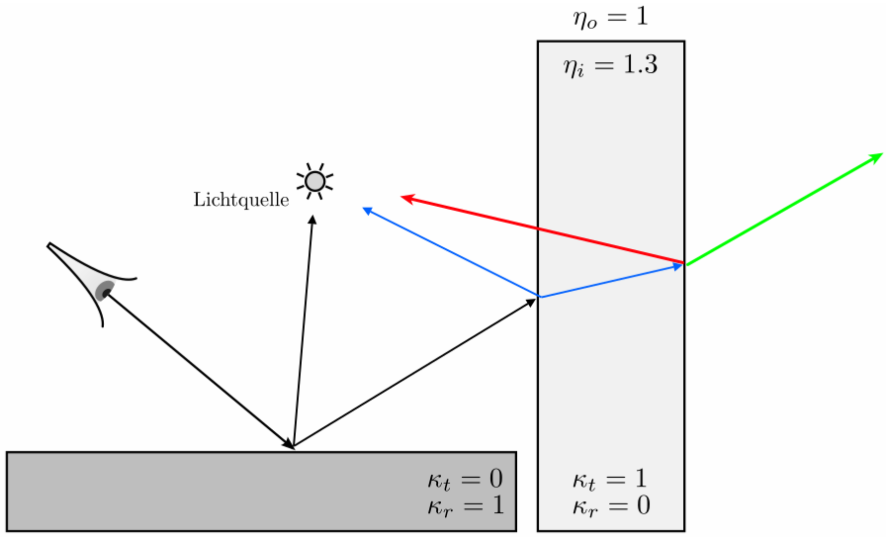


Abbildung 1: Reflexionsstrahl, Schattenstrahlen und Transmissionstrahl

Anmerkung: da es sich um perfekte Spiegel handelt, sind die eingezeichneten Schattenstrahlen unnötig.

Teilaufgabe 1b

Wie nennt man das physikalische Gesetz oder Prinzip, welches die Richtungsänderung eines Lichtstrahls beim Übergang in ein anderes Medium beschreibt?

Snellsches Gesetz ($\eta_0 \cdot \sin\theta_0 = \theta_1 \cdot \sin\delta_1$)

Teilaufgabe 1c

Welche Bedingung muss gelten, damit beim Übergang eines Lichtstrahls von einem Medium mit Refraktionsindex η_0 in ein Medium mit Refraktionsindex η_1 Totalreflexion auftreten kann?

Der Einfallswinkel muss einen Grenzwinkel $\theta = \arcsin\frac{\eta_1}{\eta_0}$ überschreiten (also besonders flach auf das Material sein).

Aufgabe 2: Beleuchtung und Wahrnehmung

Teilaufgabe 2a

- Bild 1: Nicht mögliche Kombination aus Bild 2 und Bild 3.
- Bild 2: Nur spekular (glossy)
- Bild 3: Entspricht einem Glanzlicht in Richtung N, aber das ist nur in Richtung R_L möglich.
- Bild 4: Komplett diffus.

Teilaufgabe 2b

Aussage	Wahr	Falsch
Von den drei Grundfarben der additiven Farbmischung sind Menschen gegenüber blau in der Regel am unempfindlichsten.	☑	☐
Es gibt keine zwei unterschiedlichen Lichtspektren im sichtbaren Bereich, die der Mensch als dieselbe Farbe wahrnimmt.	☐	☑
Genau drei Grundgrößen reichen (nach Graßmann) aus, um einen menschlichen Farbeindruck zu beschreiben.	☑	☐
Es entspricht nicht der menschlichen Farbempfindlichkeit, wenn die Helligkeit (Luminanz) einer Farbe als das arithmetische Mittel der RGB-Anteile berechnet wird.	☑	☐
Gammakorrektur mit dem Parameter γ wird üblicherweise durch die Abbildung $L' = \gamma^L$ beschrieben.	☐	☑

Aufgabe 3: Transformationen

Teilaufgabe 3a

Homogene Koordinaten

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

Teilaufgabe 3b

Geben Sie zeichnerisch ein einfaches Beispiel an, das deutlich zeigt, dass man die Normalen eines Primitivs im Allgemeinen nicht mit derselben Matrix transformieren kann wie die Vertizes. Um was für eine Art Transformation handelt es sich dabei?

Normale auf Kreis; Skalierung in x-Richtung.

Teilaufgabe 3c

Um Normalen korrekt von Objekt- in Kamerakoordinaten zu transformieren, verwendet man die…

- ☑inverstransponierte Model-View-Matrix.
- ☐ dehomogenisierte Model-View-Matrix.
- ☐ inverse Projektionsmatrix.

Die Matrix $\begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix}$ *ist eine…*

- ☑Translationsmatrix in homogenen 2D-Koordinaten.
- ☐ Translationsmatrix in homogenen 3D-Koordinaten.
- ☐ Rotationsmatrix in $\mathbb{R}^{3\times 3}$.

Die Matrix $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ *beschreibt eine…*

- ☐ Rotation.
- ☑Spiegelung an der ersten Winkelhalbierenden.
- ☐ nichtlineare Abbildung.

Matrixmultiplikation ist stets kommutativ, wenn…

- ☑nur Skalierungsmatrizen multipliziert werden.
- ☐ nur Scherungsmatrizen multipliziert werden.
- ☐ nur Rotationsmatrizen multipliziert werden.

2D-Rotationsmatrizen sind kommutativ:

$$\begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \cdot \begin{pmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{pmatrix} = \begin{pmatrix} \cos\alpha\cos\beta - \sin\alpha\sin\beta & -\cos\alpha\sin\beta - \sin\alpha\cos\beta \\ \sin\alpha\cos\beta + \cos\alpha\sin\beta & -\sin\alpha\sin\beta + \cos\alpha\cos\beta \end{pmatrix} \tag{1}$$

$$= \begin{pmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{pmatrix} \cdot \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \tag{2}$$

3D-Rotationsmatrizen nicht:

$$R_x(\frac{3}{2}\pi) \cdot R_z(\frac{3}{2}\pi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3}$$

$$= \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \neq \begin{pmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix} \tag{4}$$

$$= \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \tag{5}$$

$$= R_z(\frac{3}{2}\pi) \cdot R_x(\frac{3}{2}\pi) \tag{6}$$

Aufgabe 4: Texturen und Texture-Mapping

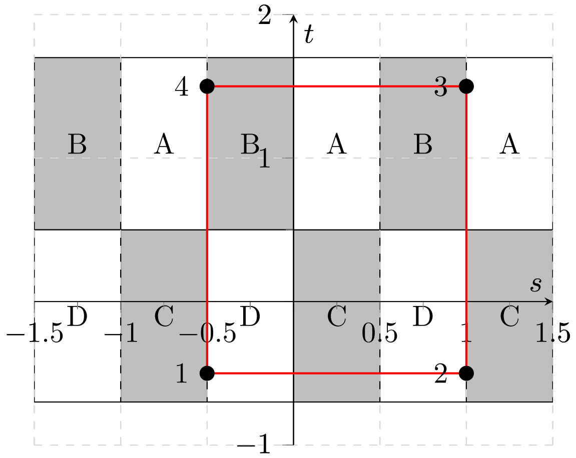
Teilaufgabe 4a

Skizzieren Sie hier die Ausgabe

Für die t-Koordinate ist `GL_CLAMP` eingestellt. Das bedeutet, dass nach oben die letzte Texel-Reihe einfach wiederholt wird und nach unten die erste Texel-Reihe wiederholt wird.

Für die s-Koordinate ist `GL_REPEAT` eingestellt. Das bedeutet, dass nach rechts und links die komplette Textur nochmals hinkopiert wird.

Damit ergibt sich das Gesamtbild:



Teilaufgabe 4b

Wann und wofür wird die bilineare Interpolation beim Texture-Mapping verwendet?

Bei Magnification um Unschärfe zu kompensieren.

Teilaufgabe 4c

Was ist die Grundidee von vorgefilterten Environment Maps?

Die Grundidee ist die Darstellung einer Umgebung mit nur geringem Aufwand.

Nennen Sie zwei Beispiele für Beleuchtungseffekte, die damit erzielt werden können!

- Diffuse Beleuchtung
- Spekulare Beleuchtung

Welche grundlegende Annahme wird dabei gemacht?

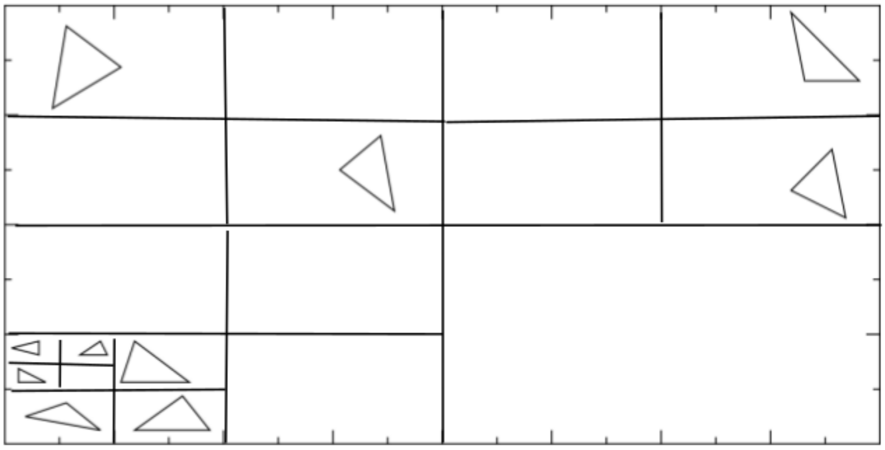
Die Umgebung ist weit genug entfernt, sodass die Position keine Rolle spielt und nur die Richtung genommen werden muss.

Aufgabe 5: Räumliche Datenstrukturen

Teilaufgabe 5a

Aussage	BVH	Octree	kD-Baum	Gitter
Die Raumunterteilung kann mit Hilfe der Surface Area Heuristic durchgeführt werden.	☑	☐	☑	☐
Die Anzahl der Schnitttests mit Primitiven kann durch Mailboxing weiter reduziert werden.	☑	☐	☑	☑
Die Raumunterteilung ist geschickt für Szenen mit einer Geometrie, die dem sogenannten Teapot-in-a-Stadium-Problem ähnelt.	☑	☑	☑	☐
Der Datenstruktur liegt ein Binärbaum zugrunde.	☑	☐	☑	☐

Teilaufgabe 5b



Teilaufgabe 5c

Sie wollen in der obigen Szene Raytracing durchführen und dies mit einer räumlichen Datenstuktur beschleunigen. Ist dafür ein regelmäßiges Gitter oder ein kD-Baum besser geeignet? Begründen Sie Ihre Entscheidung!

Ein kD-Baum eignet sich für das genannte Szenario besser, da bei Verwendung eines Gitters mehr als die Hälfte des raumes leer ist. Der Großteil der Geometrie befindet sich links unten.

Aufgabe 6: Rauschen und Turbulenz

Teilaufgabe 6a

Welche der Funktionen ist eine Rauschfunktion (value noise)? $\gamma(t)$

Teilaufgabe 6b

Welche der Funktionen ist eine Turbulenzfunktion? Wie kann man eine Turbulenzfunktion aus einer Rauschfunktion wie der aus Aufgabe a) erzeugen? Geben Sie eine Formel an!

$\delta(t)$ wird erzeugt durch spektrale Synthese (vgl. Folie 14):

$$\text{turbulence}(x) = \sum_{k}^{n} ({}^{1}\!/{}_{2})^k \cdot n(2^k \cdot x)$$

Teilaufgabe 6c

Welche zwei der Funktionen sind weder Rausch- noch Turbulenzfunktionen? Nennen Sie jeweils eine Eigenschaft von Rauschfunktionen, die nicht erfüllt wird.

α Eine Rauschfunktion darf nicht sichtbar periodisch sein
β Räumliche Korrelation ist verletzt

Aufgabe 7: Interpolation

Teilaufgabe 7a

Wie berechnet man die baryzentrische Koordinate λ_2 des Punktes $\mathbf{x}$ bezüglich des abgebildeten Rechtecks?

$$\lambda_2 = \frac{\square(\mathbf{x}, \mathbf{x}_4)}{\square(\mathbf{x}_1, \mathbf{x}_3)}$$

Teilaufgabe 7b

Berechnen Sie anhand der in der Zeichnung angegeben Längen die 4 baryzentrischen Koordinaten des Punktes $\mathbf{x}$ mit der Formel aus Aufgabenteil a). Nutzen Sie dann die baryzentrischen Koordinaten, um anzugeben, wie der interpolierte Farbwert $\mathbf{c}$ am Punkt $\mathbf{x}$ berechnet wird.

$$\lambda_1 = {}^2/_9 \qquad \lambda_2 = {}^1/_9 \tag{7}$$
$$\lambda_3 = {}^2/_9 \qquad \lambda_4 = {}^4/_9 \tag{8}$$

$$c = {}^2/_9 \cdot c_1 + {}^1/_9 \cdot c_2 + {}^2/_9 \cdot c_3 + {}^4/_9 \cdot c_4$$

Teilaufgabe 7c

$$\lambda_1^\Delta = {}^1/_6 \qquad \lambda_3^\Delta = {}^1/_6 \tag{9}$$
$$\lambda_4^\Delta = {}^2/_3 \tag{10}$$

$$c' = {}^1/_6 \cdot c_1 + {}^1/_6 \cdot c_3 + {}^2/_3 \cdot c_4 \neq c$$

Aufgabe 8: OpenGL und OpenGL-Shader

	Aussage	Wahr	Falsch
1	Beleuchtung mit Schattenberechnung kann bei OpenGL mit Hilfe von `glEnable(GL_LIGHTING | GL_SHADOWS)` aktiviert werden.	☐	☑
2	Der Vertex-Cache funktioniert nicht mit indizierten Vertizes.	☐	☑
3	Im Fragment-Shader kann auf benachbarte Fragmente zugegriffen werden.	☐	☑
4	Im Fragment-Shader kann man die Tiefe eines Fragments verändern.	☑	☐
5	Die Fixed-Function-Pipeline berechnet Beleuchtung in normalisierten Gerätekoordinaten.	☐	☑
6	OpenGL führt Clipping in normalisierten Gerätekoordinaten durch.	☐	☑
7	Der Maler-Algorithmus benutzt einen Tiefenpuffer, um das Sichtbarkeitsproblem zu lösen.	☐	☑
8	Im Vertex-Shader kann man durch Erzeugung zusätzlicher Vertizes Primitive unterteilen.	☐	☑
9	Der Fragment-Shader ist in der OpenGL-Pipeline vor den Fragment-Operationen und nach dem Geometrie-Shader.	☑	☐
10	Beim Tiefenpuffer-Verfahren ist die Reihenfolge des Zeichnens auch bei opaken Primitiven wichtig für die Korrektheit.	☐	☑

Bemerkung zu 1: falsch, da `GL_SHADOWS` nicht existiert (`https://www.opengl.org/sdk/docs/man/html/glEnable.xhtml`)

Aufgabe 9: Toon-Shading in OpenGL

Teilaufgabe 9a

shader9a.vert

```
in vec4 position;    // Vertex-Position in Objektkoordinaten.
in vec4 normal;      // Vertex-Normale in Objektkoordinaten.
out vec3 posWorld;   // Vertex-Position in Weltkoordinaten.
out vec3 normWorld; // Vertex-Normale in Weltkoordinaten.
uniform mat4 M;      // Model-Matrix.
uniform mat4 V;      // View-Matrix.
uniform mat4 P;      // Projection-Matrix.

void main(void) {
    vec4 posWorld4 = M * position;
    gl_Position = P * V * posWorld4;

    posWorld = posWorld4.xyz / posWorld4.w;
    normWorld = normalize((transpose(inverse(M)) * normal).xyz);

}
```

Teilaufgabe 9b

Berechnen Sie im folgenden Fragment-Shader den Richtungsvektor zur Lichtquelle. Berechnen Sie dann aus dem Winkel zwischen Lichtrichtung und Normale den passenden Winkelbereich `c` *und nutzen Sie* `toonShade(int c)`*, um die Ausgabefarbe zu bestimmen. Speichern Sie diese in* `color`*.*

shader9b.frag

```
out vec4 color;            // Ausgabefarbe.
in vec3 posWorld;          // Vertex-Position in Weltkoordinaten.
in vec3 normWorld;         // Vertex-Normale in Weltkoordinaten.
uniform vec3 lightWorld; // Position der Lichtquelle in Weltkoordinaten.
uniform float pi;          // Der Wert der Konstante Pi.

vec4 toonShade(int c) {
    // ...
}

void main(void)
{
    vec3 l = normalize(lightWorld - posWorld);
    float theta = acos(dot(l, normWorld));
    int c;
```

```
    if (theta < pi / 6.) {
        c = 0;
    } else if (theta < pi / 3.) {
        c = 1;
    } else {
        c = 2;
    }
    color = toonShade(c);
}
```

Aufgabe 10: Clipping in einem OpenGL-Shader

Teilaufgabe 10a

shader10a.vert

```
in vec3 position;    // Die Position des Vertex in Objektkoordinaten.
out vec3 posCamera; // Die Position des Vertex in Kamerakoordinaten.

uniform mat4 M; // Die Model-Matrix.
uniform mat4 V; // Die View-Matrix.
uniform mat4 P; // Die Projection-Matrix.

void main() {
    posCamera = vec3(V * M * vec4(position, 1.0));
    gl_Position = P * V * M * vec4(position, 1.0);
}
```

Teilaufgabe 10b

shader10b.frag

```
in vec3 posCamera; // Die Position des Vertex in Kamerakoordinaten.
out vec4 color;    // Die Ausgabefarbe.
uniform vec3 N;    // Die Normale der Clip-Ebene, in Kamerakoordinaten.
uniform vec3 p;    // Der Aufpunkt der Clip-Ebene, in Kamerakoordinaten.

bool clip() {
    // Solution starts here:
    vec3 toVertex = posCamera - p;
    return dot(N, toVertex) < 0.;
}

void main() {
    if (clip()) {
        discard;
    }
    color = vec4(1.0, 0.0, 0.0, 1.0);
}
```

Aufgabe 11: OpenGL und semitransparente Kugeln

Teilaufgabe 11.1

opengl.cpp

```
void renderScene(std::vector<Sphere>& spheres)
{
    glDisable( GL_DEPTH_TEST );  // (4)
    glDepthMask( GL_TRUE );      // (1)
    glDisable( GL_CULL_FACE );   // (6)

    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

    glDepthMask( GL_TRUE );  // (2)
    glEnable( GL_BLEND );    // (9)
    sort( spheres );         // (11) sortiert alle Kugeln von hinten nach vorne

    for (int i = 0; i < spheres.size(); ++i)
    {
        Sphere& sphere = spheres[i];

        // sortiert Primitive einer Kugel von hinten nach vorne.
        sortPrim( sphere );   // (12)
        draw( sphere );       // (13) zeichnet eine Kugel
    }
}
```

Teilaufgabe 11.2

Begründen Sie kurz warum Sie die Funktion `sort( spheres )` benutzt oder nicht benutzt haben.

Da die Sphären alle semi-transparent sind (nicht-kommuntativer Blendingoperator) müssen sie von hinten nach vorne gezeichnet werden um korrekt dargestellt zu werden.

Teilaufgabe 11.3

Begründen Sie kurz warum Sie die Funktion sortPrim(sphere) benutzt oder nicht benutzt haben.

Analog zu (11.2).

Teilaufgabe 11.4

	Aussage	Wahr	Falsch	Begründung
1	Um eine Szene mit opaken und semitransparenten Objek-ten mit OpenGL möglichst korrekt zu zeichnen, muss man die opaken nach den transparenten Objekten zeichnen.	☐	☑	
2	Der Alpha-Test erlaubt es, Fragmente anhand ihres Alpha-Wertes zu verwerfen, ohne den Tiefenpuffer zu modifizieren.	☑	☐	
3	Alpha-Clipping ist eine Fragmentoperation und findet zwischen Alpha-Test und Alpha-Blending statt.	☐	☑	
4	Der Blendingoperator definiert durch `glBlendEquation(GL_FUNC_ADD)` und `glBlendFunc(GL_SRC_ALPHA, GL_SRC_ALPHA)` ist kommutativ.	☐	☑	1)

1) Da GL_SRC_ALPHA den Alpha-Wert des Quell-Fragments multipliziert und das beim Tauschen von Quell- und Ziel-Fragment nicht berücksichtigt wird, ist der Operator nicht kommutativ.

Aufgabe 12: Bézierkurven

Teilaufgabe 12a

#	Aussage	Wahr	Falsch
1	Bézierkurven liegen immer innerhalb der konvexen Hülle der Kontrollpunkte.	☑	☐
2	Bézierkurven interpolieren zwei Ihrer Kontrollpunkte und approximieren alle anderen.	☑	☐
3	Um eine Bézierkurve affin zu transformieren, genügt es, die Transformation auf ihre Kontrollpunkte anzuwenden.	☑	☐
4	Bézierkurven sind stetig differenzierbar.	☑	☐

Teilaufgabe 12b

Kurve	Ja	Nein	Begründung
1	☐	☑	Kurve verlässt konvexe Hülle der Kontrollpunkte
2	☑	☐	
3	☐	☑	Tangentenbedingung verletzt
4	☑	☐	