Aufgabe 1: Raytracing

Teilaufgabe 1a

Raytracing nach Whitted, wie Sie es in der Vorlesung kennengelernt haben, folgt den Gesetzen der geometrischen Optik. Ergänzen Sie die folgende Liste um die 3 weiteren Strahltypen, die bei diesem Raytracing-Verfahren vorkommen!

(1) Primärstrahlen (2) Reflektionsstrahlen (rekursiv) (3) Transmissionsstrahlen (rekursiv) (4) Schattenstrahlen

Teilaufgabe 1b

Die folgenden Skizzen zeigen zwei Lichtstrahlen mit unterschiedlichem Einfallswinkel die an einer spekularen Glasoberfläche reflektiert werden (der Vektor N ist die Oberflächennormale).

In Bild 2, da dort der Winkel des Strahls auf die Oberfläche flacher ist.

Teilaufgabe 1c

Wie nennt man das physikalische Gesetz oder Prinzip, welches den Zusammenhang zwischen Einfallswinkel und Reflektivität beschreibt?

Snelliussches Brechungsgesetz. Es lautet

$$n_1 \cdot \sin(\theta_1) = n_1 \sin(\theta_2)$$

wobei die Winkel von der Oberflächennormale aus gemessen werden. n_1, n_2 sind Materialkonstanten.

Aufgabe 2

Teilaufgabe 2a

TODO

Teilaufgabe 2b

TODO

Abbildung 1: Whatever

Teilaufgabe 2c

TODO

Teilaufgabe 2d

TODO

Aufgabe 3

Teilaufgabe 3a

TODO

Teilaufgabe 3b

TODO

Aufgabe 4

Teilaufgabe 4a

TODO

Teilaufgabe 4b

TODO

Aufgabe 5

Teilaufgabe 5a

TODO

Teilaufgabe 5a

TODO

Aufgabe 6

Teilaufgabe 6a

TODO

Teilaufgabe 6b

TODO

Teilaufgabe 6c

spheretracing.frag

```
in vec3 A; // Ursprung des Strahls.
in vec3 D; // Die normalisierte Richtung des Strahls.
in float tMax; // Abbruchkriterium: maximale Suchdistanz.
uniform float epsilon; // Toleranz

// Distanzfunktion. Liefert den Abstand von x zur nächsten Fläche.
float DF( vec3 x ) { ... }

// Implementieren Sie Sphere Tracing in dieser Funktion.
bool sphereTrace( out vec3 pos, out int steps ) {
    pos = A;
    steps = 0;
```

```
    float t = 0.;
    while (t < tMax) {
        float d = DF(pos);
        pos += d * D;
        if (abs(d) < epsilon) {
            return true;
        }
    }
    return false;
}
```

Aufgabe 7

Teilaufgabe 7a

TODO

Teilaufgabe 7b

TODO

Aufgabe 8

Teilaufgabe 8a

TODO

Teilaufgabe 8b

TODO

Teilaufgabe 8c

TODO

Aufgabe 9

keyframing.vert

```
in vec4 p; // Position des Vertex in Objektkoordinaten.
uniform float t; // Aktueller Zeitpunkt.
uniform float t1; // Die Zeitpunkte der drei Keyframes.
uniform float t2;
uniform float t3;
uniform mat4 M1; // Die drei Transformationsmatrizen (Objekt->Welt).
uniform mat4 M2;
uniform mat4 M3;
uniform mat4 VP; // Die View-Projection-Matrix.

void main() {
    vec4 pWorld;
    if (t < t2) {
        pWorld = mix(M1 * p, M2 * p, (t - t1) / (t2 - t1));
    } else {
        pWorld = mix(M2 * p, M3 * p, (t - t2) / (t3 - t2));
    }

    gl_Position = VP * pWorld;
}
```

Aufgabe 10

Teilaufgabe 10a

shader.frag

```
void renderScene() {
    // Setup vor dem Löschen von Frame- und Tiefenpuffer
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
    // Zeichnen der Szene ab hier

    //TODO
}
```

Teilaufgabe 10b

TODO

Teilaufgabe 10c

TODO

Aufgabe 11: Bézierkurven

Teilaufgabe 11a

TODO

Teilaufgabe 11b

TODO