Aufgabe 1: Raytracing

Teilaufgabe 1a

TODO

Teilaufgabe 1b

TODO

Aufgabe 2

Teilaufgabe 2a

Abbildung 1: Whatever

TODO

Teilaufgabe 2a

TODO

Aufgabe 3

TODO

Aufgabe 4

TODO

Aufgabe 5

TODO

Aufgabe 6

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

TODO

Aufgabe 8

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

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
}
```

Aufgabe 11: Bézierkurven

TODO