Aufgabe 1

Teilaufgabe 1a

1. Lichtvektor L
2. Normale N
3. View-Vekto V
4. Reflektiertes Lich R_L

Teilaufgabe 1b

- Diffus und Spekular ändern sich, da die diffuse Komponente von der Lichtrichtung abhängig ist und die spekulare Komponente vom Winkel des View-Vektors zum reflektierten Lichtvektor abhängig ist.
- Die spekulare Komponente ändert sich weil sie vom Winkel des View-Vektors zum reflektierten Lichtvektor abhängig ist.
- Imperfekte Spiegelung: je größer der Phong-Exponent desto genauer die Spiegelung, heißt der Spekulare Lichtfleck wird kleiner aber dafür intensiver
- Bei Gouraud-Shading wird die Farbe an den Vertices berechnet und dazwischen interpoliert. Dadurch können Glanzlichter verloren gehen, die in der Mitte von Primitiven liegen. Bei kleineren Primitiven ist die Wahrscheinlichkeit, dass dies passiert geringer. Außerdem ergibt die Interpolation genauere Ergebnisse, da die Vertices näher an den Punkten liegen, für die interpoliert wird

Teilaufgabe 1c

TODO

Teilaufgabe 1d

TODO

Aufgabe 2

Teilaufgabe 2a

- $0.5 * 0,2 + (1 - 0,5) * 0,4 = 0,3$
- $0,3$
- erfordert Tiefensortierung. Im allgemeinen ist Blending nicht kommutativ

Teilaufgabe 2b

Der Tiefenalgorithmus sieht nicht, dass der Zaun an manchen Stellen durchsichtig ist und zeichnet deswegen Objekte die dahinter liegen nicht. Mit Alpha-Testing werden dahinterliegende Objekte gezeichnet

Teilaufgabe 2c

Rufe DrawScene auf, Setze Source- und Destination-Faktor auf die 1/Anzahl Lichtquellen, verschiebe Lichtquelle, Rufe DrawScene auf (Blending der Ergebnisse), setze Source auf 2/Anzahl, verschiebe Lichtquelle, Rufe DrawScene auf (Blending der Ergebnisse) usw bis alle Lichtquellen durch sind

Aufgabe 3

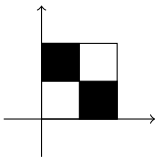
Teilaufgabe 3a

- löst das Aliasing-Problem bei Minification
- Man berechnet Mipmap-Stufen, indem die Höhe und Breite der Textur jeweils halbiert wird
- Zugriff auf die Textur: Texelgröße(n) $\leq$ Größe des Pixelfootprints auf der Textur ¡ Texelgröße(n+1)

Teilaufgabe 3b

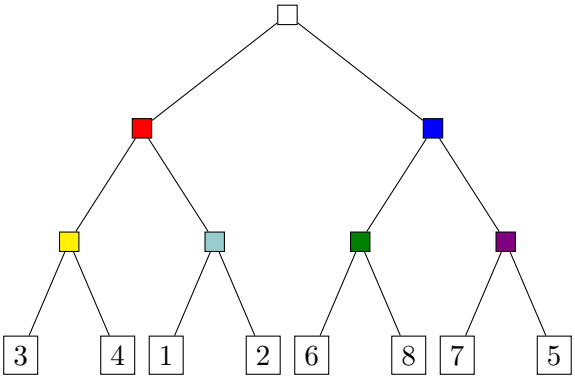
Darstellung reflektierender Objekte mit Spiegelung von Umgebung ohne geometrische Repräsentation. Annahme: nur Richtung des Strahls wird berücksichtigt, man geht davon aus, dass die Umgebung relativ weit weg ist

Teilaufgabe 3c



Aufgabe 4

Teilaufgabe 4a



Teilaufgabe 4b

Ziel: Bounding Boxes mit möglichst geringem Volumen

Die Wahrscheinlichkeit dafür dass man ein Primitiv trifft, wenn man die Bounding Box getroffen hat, soll groß werden. Konstruktion bezieht Kostenfunktion (Kosten für Schnitt mit Knoten) mit ein, damit die Kosten minimiert werden. Surface Area der Primitive soll einen großen Anteil der Bounding Box ausmachen. Eingeflossene Größen: Primitiv-Flächeninhalt, Surfache Area der Box, Kosten der Traversierung eines Knotens im Verhältnis zu Kosren eines Strahl-Primitiv-Tests

Teilaufgabe 4c

Suboptimal, wenn es Stellen gibt, an denen wenige Primitive liegen und welche mit vielen Primitiven auf wenig Raum → es gib Boxen mit viel leerem Raum drin. Median ist gleich gut wie SAH wenn Primitive gleichmäßig verteilt sind.

Teilaufgabe 4d

Erster Schnittpunkt mit einem Primitiv wird an Gitterzellen weitergegeben. Vorteil: man muss das gleiche Objekt nicht mehrfach schneiden.

Aufgabe 5

shader5.frag

```
uniform vec3 eye; //Position des Betrachters in Weltkoord
in vec3 dir;      //Strahlrichtung in Weltkoord
uniform vec3 spheres[10];
out vec4 fragColor;
int main() {
    float t = -1;
    vec3 pos;
    vec3 normal;
    for (i=0; i < 10; i++) {
        vec2 isec = intersectRS(eye, dir, vec3(spheres[i]), spheres[i].z);
        if (isec.x < t && isec.x > 0) {
            t = isec.x;
            pos = eye + t * dir;
            normal = normalize(pos - vec3(spheres[i]);
        }
    }
    if (t != -1) {
        fragColor = computeShading(pos, normal);
    }
    // ...
}
```

Aufgabe 6

Teilaufgabe 6a

Polygone, die man nur von hinten sieht, werden nicht gezeichnet. Wird zur Beschleunigung verwendet, weil man dann weniger Polygone zeichnen muss.

Teilaufgabe 6b

Ergebnisse der letzten Verarbeiteten Vertices werden gecacht. Nur sinnvoll, wenn Vertices mehrfach verwendet werden, z.B. durch Indizierung. Pipelinestufe: Primitive Assembly

Teilaufgabe 6c

über Indizes

Teilaufgabe 6d

Triangle Strips brauchen weniger Platz, weil sich Dreiecke Vertices teilen (n+2 statt $3n$ Vertices)

Teilaufgabe 6e

Vertex Shader holt Textur, ordnet den Vertices Punkte auf den Texturen zu. Fragment Shader berechnet anhand dessen die Texturierung der Primitive

Teilaufgabe 6f

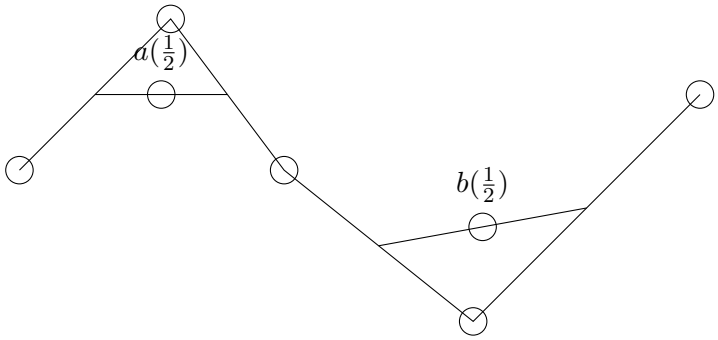
Aussage	Wahr	Falsch
Vertex Shader kann Vertices löschen und generieren		X
Vertex Shader kann Transformieren	X	
Sichtrichtung entlang negativer Y-Achse		X
mehrere Shader in einem Zeichenvorgang		X
inkonsistente Interpolation bei T-Vertices	X	

Teilaufgabe 6g

1. B V
2. F V
3. E V
4. A N
5. C N
6. D F

Aufgabe 7

Teilaufgabe 7a



Tangenten an den Randpunkten zeigen auf den nächsten Punkt. 2fach stetig diffbar (runde Kurve, keine Ecken, keine Unterbrechungen). Kurve liegt innerhalb der konvexen Hülle der Kontrollpunkte.

Teilaufgabe 7b

- $a_3 : C^1$. Begründung: $a_3 - a_2 = b_1 - b_0$, aber $a_2 + (a_2 - a_1) \neq b_1 + (b_1 - b_2)$
- $c_0 : C^0$. Begründung: $b_3 - b_2 \neq c_1 - c_0$

Teilaufgabe 7c

1. bilden eine Basis des Polynomraumes
2. Rekursionsformel:
$$B_i^n(u) = u \cdot B_{i-1}^{n-1} + (1-u) \cdot B_i^{n-1}$$
3. symmetrisch
4. positiv (≥ 0) auf [0,1]