## Aufgabe 1

a)

x,y Farbe

Y Helligkeit

b)

saftiges Grün

c)

Nein, die Farben im unteren Rand sind nicht darstellbar durch monochromatische LQ/sind keine Spektralfarben.

d)

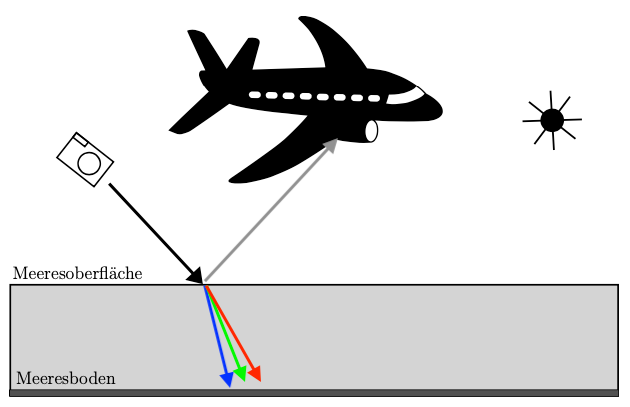
Aufgrund der menschlichen Empfindlichkeit; Weber-Fechner-Gesetz

## 

## Aufgabe 2

a)

Grau: Reflexionsstrahl mit allen Wellenlängen



b)

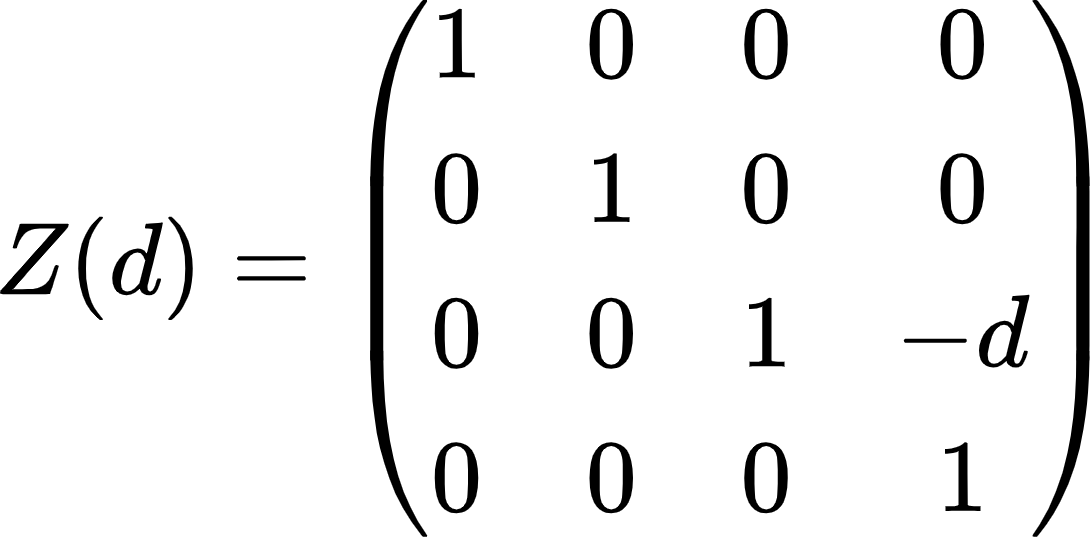
Snellsches Gesetz

c)

2^n. Pro Schnittpunkt werden 2 Sekundärstrahlen erzeugt

## Aufgabe 3

a)



Wenn positiv wird der Abstand größer kleiner.

b)

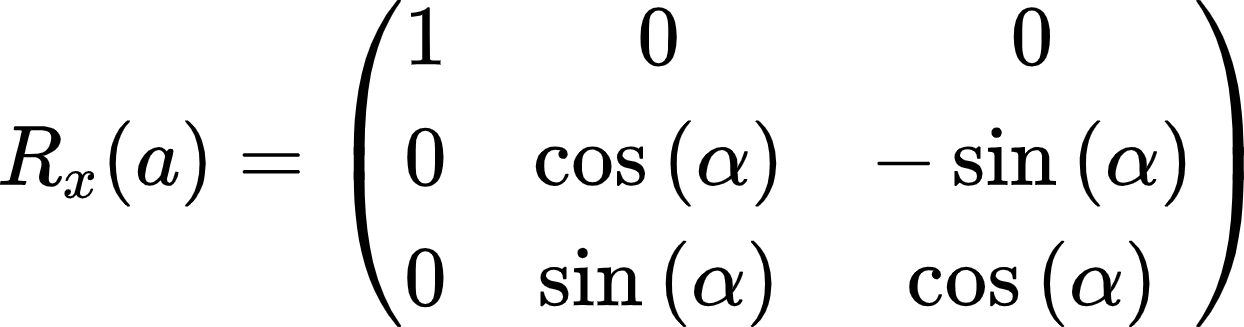
1 0 0 0

0 ca -sa 0

0 sa ca 0

0 0 0 1

R\_x(a) = V\* [1 0 0 0; 0 ca -sa 0; 0 sa ca 0; 0 0 0 1]\*inv(V)



c)

Mit N bleibt die Normale senkrecht zur Oberfläche

## Aufgabe 4

a)

Koeffizient (Material)

Lichtintensität

Reflexion Lichtrichtung auf der Oberfläche

Kamerarichtung

Phong-Exponent

b)

Reflexion findet in alle Richtungen statt.

Reflexion abhängig von der Rotation des Objekts. (2-80)

c)

i) Ja,

ii) Nein,

iii) Nein, , aber der Punkt ist verschattet, spek Term würde wegfallen

## Aufgabe 5

a)

Um mehrere Schnitttests mit demselben Primitiv zu vermeiden

b)

Nein, bei BVH tritt nicht der Fall auf dass ein Primitiv in 2 Hüllkörpern ist

c)

ABDE Der nähere Kindknoten wird zuerst betrachtet, also muss E oben liegen.

D Kann nicht ausgeschlossen werden da der Schnittpunkt nicht vor D liegt.

A und B enthalten keine Primitive da sie keine Blätter sind, da gibts also nichts zu finden.

Ich habe einfach CD geschrieben, denn A und B schon traversiert wurden. (Lege A auf den Stack, traversiere A. Also poppe A, lege C und B auf den Stack. Poppe B, traversiere B, also lege D und E auf den Stack. Traversiere E (also poppe), finde den Schnittpunkt. Also aktuell noch nicht traversierte D und C auf dem Stack. Ich bin mir hier aber nicht sicher, ob ein Knoten vor oder nach dem Traversieren gepoppt werden soll. Vermutlich davor, siehe 05-100, es ist zwar für kd Bäume, das Prinzip sollte aber gleich sein)

Das stimmt wohl, also AB werden auf jeden Fall gepoppt, hab ich vergessen.C Hab ich ebenfalls komplett vergessen. Ich schätze mal, einen Knoten traversieren heißt ihn poppen, und damit alle primitive die “raus kommen” testen

D - ausgeschlossen werden könnte, weil keine Primitive enthält

C - ausgeschlossen werden könnte, weil Schnittpunkt mit C ferner als gefundenen Schnittpunkt ist.

(E), D, C

C: kann ausgeschlossen werden da der Schnitt mit der AABB weiter weg als mit dem Dreieck ist

D: kann noch nicht ausgeschlossen werden da der Schnitt mit der AABB näher als mit dem Dreieck ist

+1 +1 +1 +69420

d)

Ja; da die Kugel idR zu groß ist, würden viele Hüllkörper überschneiden + Hüllkörperschnitte können außerhalb der Szene liegen, also insbesondere hinter dem Strahlursprung/der Kamera

Es wird viel leerer Raum eingeschlossen.

Kugelförmige Hüllkörper sind aufwendiger zu erzeugen

e)

Inkrementell Hüllkörper zusammenbauen mit SAH?

Einfach Bäume als Primitive behandeln beim Aufbau einer BVH?

Würde das mit SAH gar nicht erwähnen. Einfach eine BVH für die Bäume aufbauen, und bei Schnitt mit einem Baum die Baumstruktur traversieren.

So wie ich die Aufgabe verstehe, ist die BVH eines Baumes im Koordinatensystems des Baumes gegeben.

Ich würde die AABB jedes Baumes in Weltkoordinaten überführen und dann darüber ein globales BVH aufbauen. Man traversiert also zuerst die globale BVH in Weltkoordinaten, landet bei einem Blatt, überführt den Strahl in die Modellkoordinaten und traversiert dann die BVH eines Baumes dort.

BVH ist semigut bei der Aufgabe. Dadurch dass bei einem Wald die Bäume nah beieinander stehen, würden sich die AABBs oft überlappen. Besser wäre eine Uniform Space Subdivision(reguläres Gitter) mit den BVHs der Bäume als Objekte.

## Aufgabe 6

### a)

i)

ii)

iii)

3D-Primitiv. Nach der Projektion gilt keine Linearität mehr.

3D-Primitiv, da perspektivische Projektion Längenverhältnisse ändert

### b)

### c)

Clamp, repeat, clamp to border, mirrored repeat

### d)

Textur wird unverändert angezeigt?

Bei der bilinearen Interpolierung haben die Punkte die Koordinaten (i + 0.5)/b, (j + 0.5)/b wodurch die Texturkoordinaten immer zwischen zwei Pixeln liegen werden. Dadurch sieht das Bild verschwommen aus.

### e)

i)

Hohe Auflösung = größeres Spektrum. Da sie keine Werte in höheren Frequenzen haben, muss nicht so dicht abgetastet werden.

ii)

“Trilineare” Interpolation innerhalb Mipmap-Stufe (wie bilinear aber für die 3 Achsen), danach lineare Interpolation zwischen Stufen n und n+1

iii)

Spekulare Reflexionen lassen sich approximieren

Es wird nur über den Richtungsvektor auf die Map zugegriffen, nicht über die Position

Bei der Schattierung?? Idee: Schattierung der diffusen Oberfläche vorberechnen und dann darauf zugreifen, falls Schattenstrahl nicht LQ erreicht. Ebenfalls mit Richtung?

iv)

z-Aliasing

## Aufgabe 7

a)

n(a) = ((0.86 \* 0.3) + ((1-0.86) \* 0.2)) \* 0.25

* ((0.86 \* 0.1) + ((1-0.86) \* 0.6)) \* 0.75

b)

n’(p) = n(2p)

n’(a) = 0.8 \* 0.5

* 0.4 \* 0.5

n’(a) = n((1.72, 3.5)) = 1\*0.4\*0.5 + 1\*0.8\*0.5 = 0.6

## Aufgabe 8

a)

b)

Ray marching, genauer Sphere Tracing. Der Distanzwert bestimmt die Schrittgröße und wenn d < 0 bedeutet dass die Position innerhalb der Geometrie liegt.

Ausgenutzte Eigenschaft: man kann immer eine Strecke f(x) zurücklegen (also auch entlang des Strahls), ohne einen Schnittpunkt zu verpassen

## Aufgabe 9

### a)

(b) Primitive assembly, ~~Tesselierung~~;

(d) Clipping, Rasterisierung;

(f) Stencil test, Depth test, Blending

### b)

* glEnable(GL\_DEPTH\_TEST);
  + Tiefentest
* glVertexAttribPointer(...);
  + Vertex-Shader
* glEnable(GL\_CULL\_FACE);
  + Rasterierung? Primitive Assembly!

### c)

a und c. Im VS wird jeweils ein Vertex transformiert. Im GS werden jeweils alle Vertices eines Primitives transformiert, wodurch manche doppelt transformiert werden -> mehr Aufwand im Geometry Shader.

### d)

Im GS hat man schon die Primitive und könnte die Normalen beliebig berechnen.

(Über Face culling kann man auch vorder und rückseite bestimmen?)

### e)

Gouraud: vertex shader, Farben werden interpoliert

Phong: fragment shader, Normalen werden interpoliert

## 

## Aufgabe 10

(sR*dA) + (dR*1)

(sG*dA) + (dG*1)

(sB*dA) + (dB*1)

(sA*dA) + (dA*1)

glDisable: ? (0.18, f0.5, 0.64, 0.3) Stencil: 1

glEnable ? (0.1,0.3,0.5,0.2) Stencil: 0

Ohne Gewähr

Mit Begründung:

**Disable**:   
*Farbe:* 0.2 \* (0.4, 1.0, 0.7, 0.5) + 1 \* (0.1, 0.3, 0.5, 0.2) = (0.18, 0.5, 0.64, 0.3)  
*Stencil* = 1, da der Test immer als erfolgreich zählt, der Tiefentest klappt auch (da aus) und wir somit im zpass landen (3. Argument). Das ist “GL\_KEEP”, damit bleibt die 1  
  
**Enabled**:  
*Stencil* = 2. Der Stencil test muss fehlschlagen, da “(stencil & bitmask) op refstencil” = “(1 & 1) > 1” nicht wahr ist. Damit landen wir direkt in “fail” und dekrementieren.

Dadurch wird der Tiefentest nicht ausgeführt und auch die Farbe nicht angepasst. Ergo bleibt die Originalfarbe.  
*Farbe* = (0.1, 0.3, 0.5, 0.2)

## Aufgabe 11

a)

t = -1;

for (int i = 0; i < num\_spheres; ++i) {

vec3 center = spheres\_buffer.spheres[i].xyz;

float radius = spheres\_buffer.spheres[i].w;

float test\_t;

bool isect\_found = isect(eye, dir, center, radius, test\_t);

if ((test\_t < t || t < 0) && isect\_found) {

t = test\_t;

idx = i;

}

}

if (t >= 0) return true;

else return false;

b)

fragColor = vec3(0.0);

if (isect\_all(eye, dir, idx, t)) {

vec3 p = eye + t\*dir;

vec3 n = normalize(p - spheres\_buffer.spheres[idx].xyz);

for (int i = 0; i < num\_lights; ++i) {

vec3 light\_dir = normalize(p - l\_pos[i]);

int test\_idx;

float test\_t;

if (!isect\_all(p, light\_dir, test\_idx, test\_t) || test\_t >= length(l\_pos[i] - p) {

fragColor += shade(p, n, l\_pos[i], l\_col[i]);

}

}

} else {

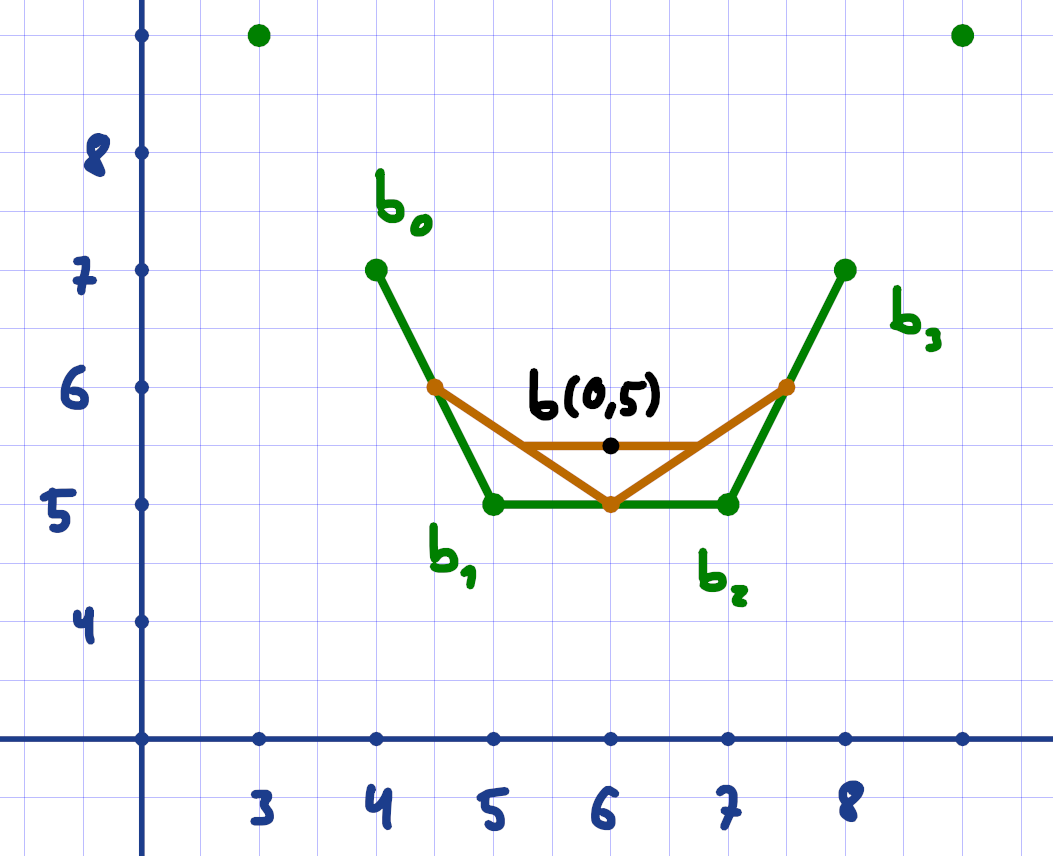
discard;

}

## Aufgabe 12

a)

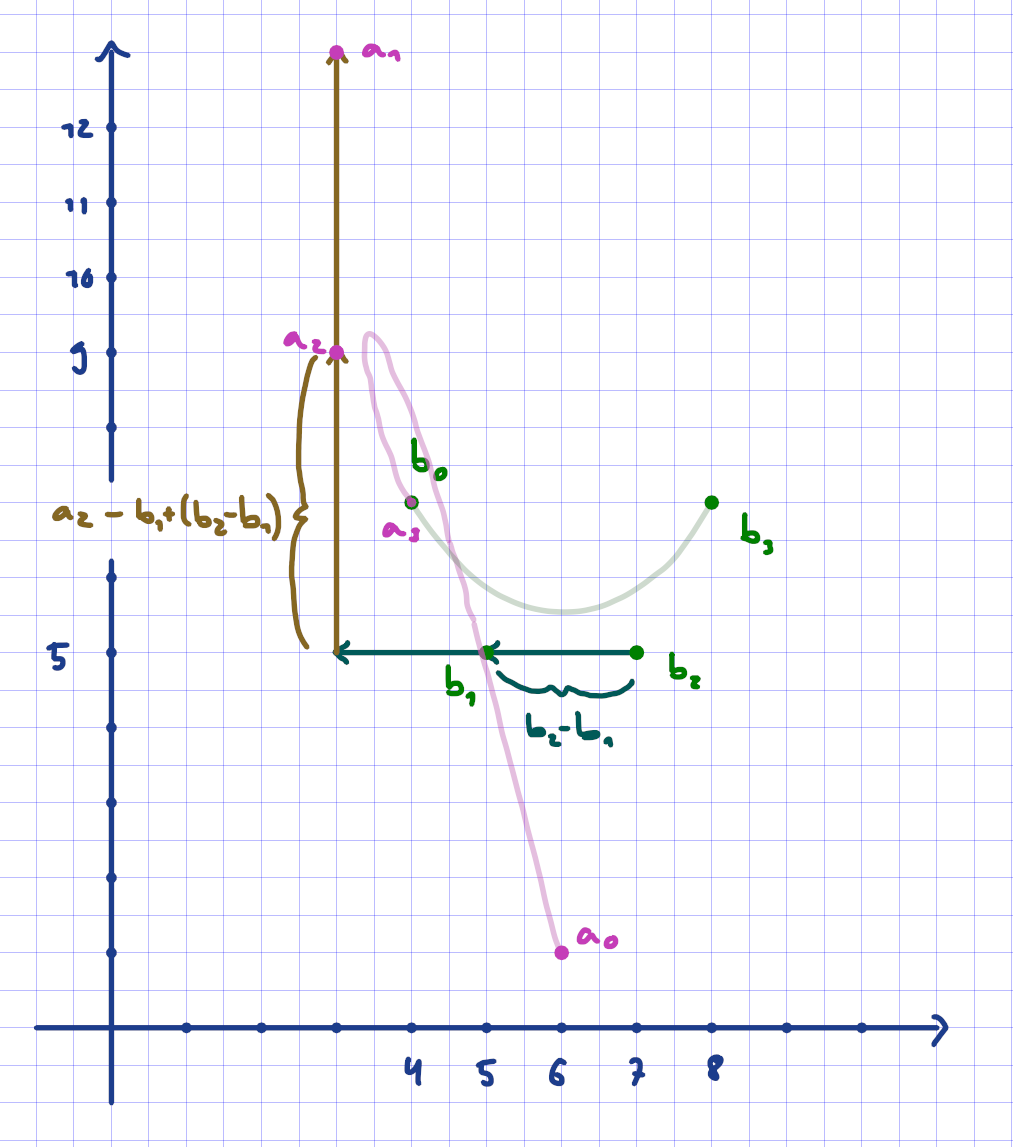
b)



c)

Man muss doch a und b C^2-stetig zusammensetzen:

Graphisch:



d)

a0 = c3, C0-Stetig