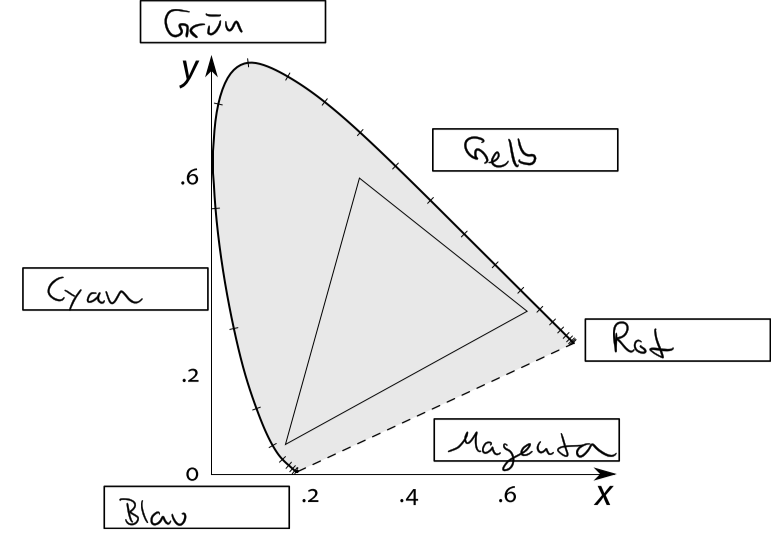
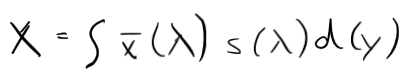
Disclaimer

Nach unserem besten wissen erstellt, Korrekturen sind gerne gesehen.

Aufgabe 1

a) 

b) x,y = Chromazität

Integral bilden um X,Y,Z zu berechnen 

d(y) ist müll sollte d\_lambda sein

Y,Z analog

transformation in den xyY Farbraum

x= X/X+Y+Z, y= Y/X+Y+Z

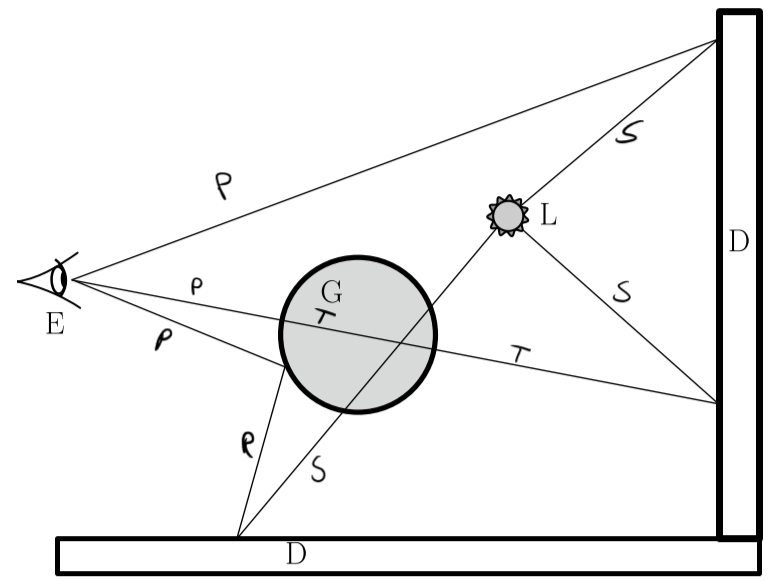
c) Formel Folien 01/56

Aufgabe 2

a)

1. kann nicht erzeugt werden, keine Transmission von Schattenstrahlen
2. kann erzeugt werden, Strahl wird vom Schnitt mit der diffuse Oberfläche richtung Lichtquelle verschossen ohne blockiert zu werden
3. kann erzeugt werden, siehe b falsch: Primärstrahl sollte beim Austritt aus G gleichen Winkel haben wie vor dem Eintritt
4. kann nicht erzeugt werden, Primärstrahl wird an diffuser Oberfläche reflektiert

b)



Aufgabe 3

a) k\_s \* I\_l \* dot(R\_l, V)^n

k\_s = Spekulare Koeffizient

I\_l = Lichtintensität

R\_l = reflektierter Lichtvektor

V = Sichtvektor

n = Phong-Exponent

b) Gouraud-Shading: Berechnung von Farbe und Beleuchtung im Vertex Shader, interpoliere diese später im Fragment Shader

Phong-Shading: Interpoliert über die Normale im Fragment-Shader.

Kein Unterschied bei Oberflächen ohne scharfe Kanten (sind uns hier nicht sicher)

c) 1,2,3,4 (fühlt sich falsch an)

Aufgabe 4

a) r1: A, B, E, C, F

r2: A, C, G, F, B, E

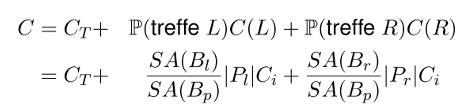
Alternativ:  
r1: A,B,E,C (F muss nicht traversiert werden, da seine AABB hinter dem Schnittpunkt mit E liegt)

r2: A,C,F,B,E

b)

1. Überlappung von AABBs
2. OBBs, Slabs
3. größerer Aufwand der Schnittberechnung

c)



e) BVH: 4

kD-Baum: 1, 2, 3, 4

reguläres Gitter: 1, 2, 3

Octree: 1, 2, 3, 4

Aufgabe 5

a) cos(phi) -sin(phi) 0 0

sin(phi) cos(phi) 0 0

0 0 1 0

0 0 0 1

b) 1 0 0 4

0 1 0 0

0 0 1 -3

0 0 0 1

c) 2 da nach dem dehomogenisieren q\_0 = q\_2 =/= q\_1 gilt

d) M\*(x,y,z,w)T = (x,y,-z,-z)T

Spiegelung an xy-Ebene in homogenen Koordinaten

Projektion auf die um 1 nach vorne verschobene xy-Ebene + Skalierung um Faktor z

Für alle Punkte gilt z’ = w’

Aufgabe 6

a)

A für Flächeninhalt (Area).

A(A,B,C) = 4\*2 = 8

A(P1,B,C) = 2\*2 = 4 => lambda\_a = ½

A(A,P1,C) = (8-4)/2 = 2 => lambda\_b = 1/4

A(A,B,P1) = A(A,P1,C) => lambda\_c = 1/4

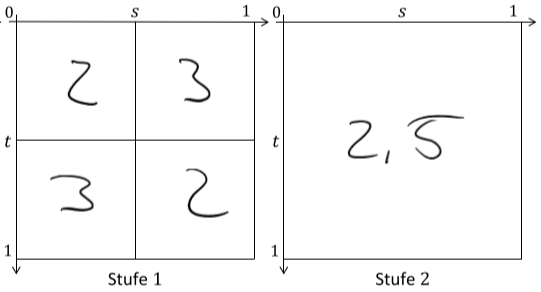
P\_{1,st} = ½ \* A\_st + ¼ \* B\_st + ¼ C\_st = (-¼, -½)T + (¼, 0)T + (¼, ¼)T

= (¼, -¼)T

b) |P3\_st - P4\_st| = sqrt((⅜)^2 + (⅜)^2) = sqrt(18/64) = sqrt(9/32) = 3/sqrt(32)

|y2\_st - y4\_st| =1/8 - (-1/8) = ¼

c)



Stufe : 2

Ausgelesener Wert: 2,5

Müsste 1 sein weil

Texelgröße(MipMap-Stufe n) <= Pixelfootprint < Texelgröße(MipMap-Stufe n+1)

Footprint aus Aufgabe b) 3/sqrt(32) = 0.53

Damit ½ (Stufe 1) <= 0.53 < 1 (Stufe 2). Deswegen Stufe 1.

Der ausgelesene Wert ist dann 3.

Wenn man die größte Seitenlänge der AABB als Footprint nimmt, kommt 5/8 raus

s

Aufgabe 7

a)

1. Vertex finalisieren der Vertex Attribute z.B. world\_normal berechnen

c) Geometrie geometrie erstellen

e) Fragment Farbe des Fragments berechnen

b) Triangle\_strip, IndexBuffer

c) Clip Koordinaten da man clipping vor der normalen transformation durchführen möchte und nach dem man weiß was alles Teil des zu sehenden Bildes ist.

Vor der Rasterisierung, da wir sonst die ganze Szene rasterisieren und nicht nur den sichtbaren Bereich.

d) Modell-Koordinaten, da durch die Projektion in Clip-Koordinaten die Szene verzerrt ist.

Aufgabe 8

a)

1. glEnable(GL\_DEPTH\_TEST), glDepthMask(GL\_TRUE)
2. glEnable(GL\_BLEND), glDepthMask(GL\_FALSE)

b) Da die Farbe sonst falsch berechnet wird.

Sortierkriterium: von hinten nach vorne (Abstand zur Kamera)

c) O, 3, 5, 8, T, 4

Aufgabe 9

a)

Aufgabe 10

a) t(x) = sum\_i((1/f)^i \* n(f^i \* x))

i = Oktaven

(1/f)^i = amplitude

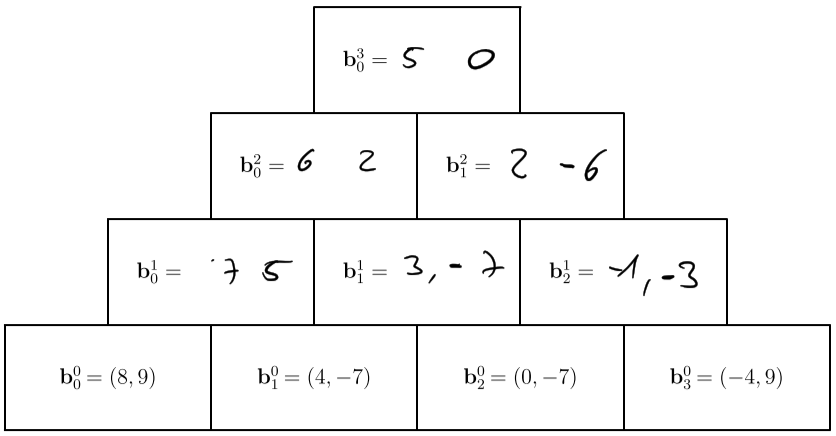
n(f^i \* x) = noise

b) man kann sich die für 2x und 3x berechneten noise-Werte speichern und kann diese später wieder verwenden.

c) wähle je das nächst kleiner/größere x und y berechne h(w) w = (x,y) spanne eine Ebene auf und berechne deren normale

Aufgabe 11

a)



b) b21 - b20 = (-4,-8)

c)

