

# Aufgabe 1: Farben und Farbwahrnehmung

## Teilaufgabe 1a: Chromatizitätsdiagramm

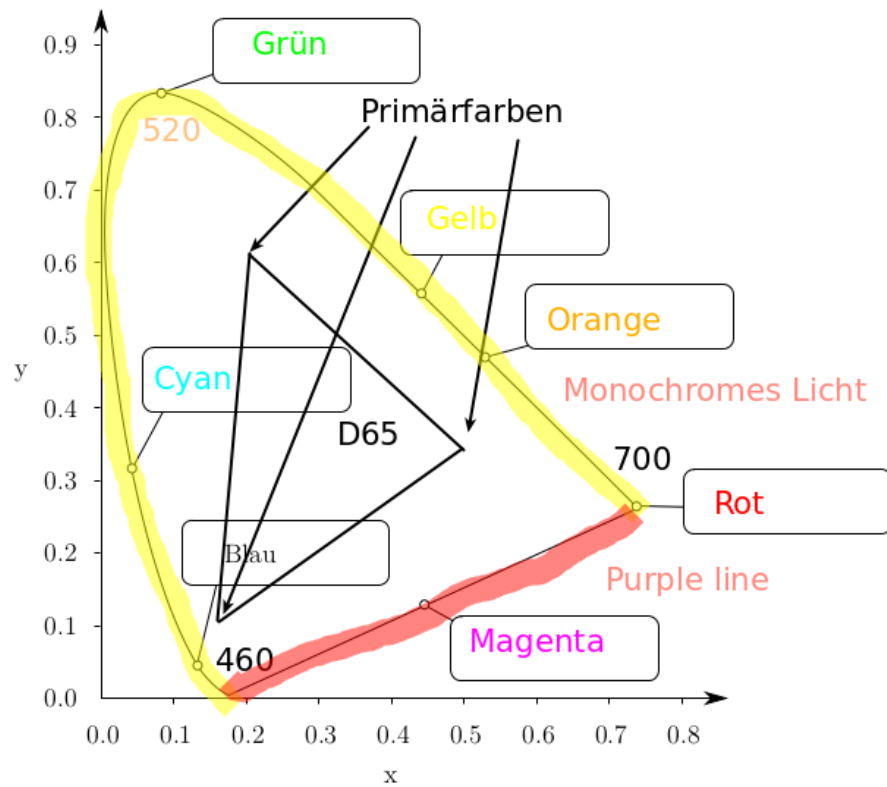


Abbildung 1: Aufgabe 1a

## Teilaufgabe 1b

Alles auf der Purple line. Also insbesondere **Magenta**.

## Teilaufgabe 1c

$$x = \frac{X}{X + Y + Z} \quad (1)$$

$$y = \frac{Y}{X + Y + Z} \quad (2)$$

Aussage	Wahr	Falsch	Begründung
Den Weißpunkt eines Farbraums bezeichnet man auch als Tristimuluswert.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Die RGB-Werte sind die Tristimulus-Werte. Der Weißpunkt heißt üblicherweise $D[\text{Zahl}]$ , wobei die Zahl die Temperatur angibt. D65 hat eine Farbtemperatur von ca. 6504K.
Die subjektiv empfundene Stärke von Sinneseindrücken ist proportional zum Logarithmus ihrer Intensität.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Jeder Farbeindruck für den Menschen kann mit drei Grundgrößen beschrieben werden.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	vgl. 1 (b)

### Teilaufgabe 1d

(2) < (3) < (1), also

RGB < Raum aller Farben die durch 100 monochromatische Leuchtdioden darstellbar sind < XYZ

### Teilaufgabe 1e

## Aufgabe 2: Whitted-Style Raytracing

### Teilaufgabe 2a-d

Siehe Abbildung 2.

### Teilaufgabe 2e

$$\eta_i \sin \theta_i = \eta_t \sin \theta_t \quad (3)$$

$$1 \cdot \frac{4}{10} = 1.5 \sin \theta_t \quad (4)$$

$$\Leftrightarrow \sin \theta_t = \frac{4}{15} = \frac{2}{7.5} \quad (5)$$

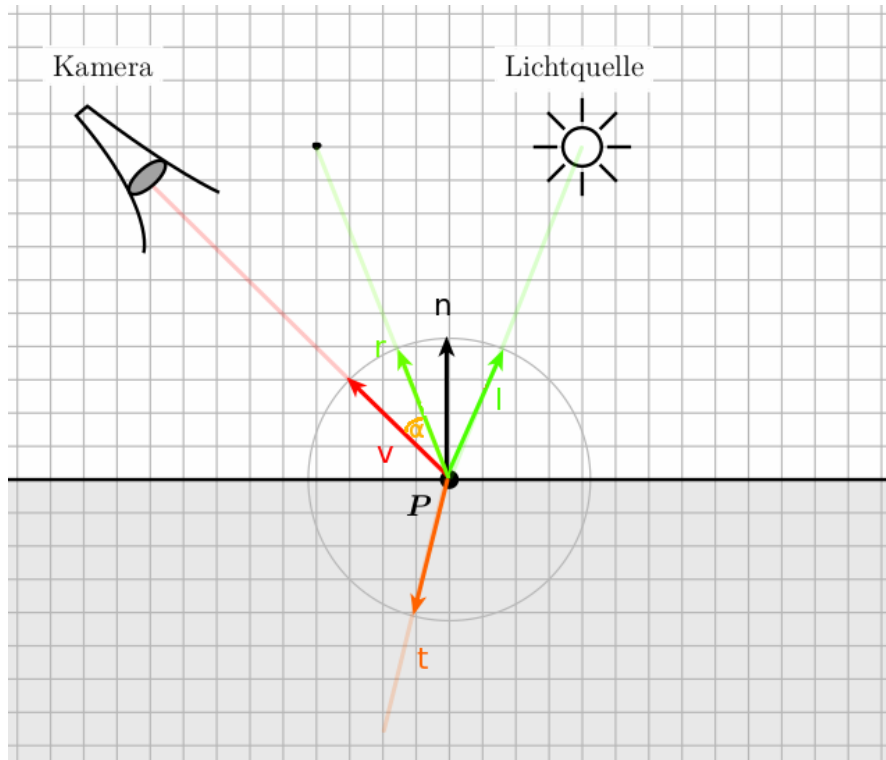


Abbildung 2: Aufgabe 2a-d;  $n_1 = 1, n_2 = 1.5$

### Teilaufgabe 2f

$$I_s = k_s \cdot I_L \cdot \cos^n \alpha \quad (6)$$

$$\alpha = r_L \cdot v \quad (7)$$

wobei  $k_s$  ein Materialparameter und  $I_L$  die intensität der Lichtquelle ist.  $n$  wird der Phong-Exponent genannt (TODO: woher kommt der?)

### Teilaufgabe 2g

Snellsches Brechungsgesetz

$$\eta_i \sin \theta_i = \eta_t \sin \theta_t$$

## Aufgabe 3: Transformationen

$$\begin{pmatrix} s_x & h_x & t_x \\ h_y & s_y & t_y \\ a & b & c \end{pmatrix}$$

- Die Parameter  $s_x, s_y$  skalieren in Richtung der  $x$  bzw.  $y$  Achse.
- Die Parameter  $h_x, h_y$  scheeren in Richtung der  $x$  bzw.  $y$  Achse.
- Die Parameter  $t_x, t_y$  führen eine Translation in  $x$  bzw.  $y$  Richtung aus.
- Die Parameter  $a, b, c$  skalieren.

Die Matrix

$$\begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

rotiert um  $\theta$  um den Ursprung (gegen den Uhrzeigersinn.)

- Bild 1: Translation um 1 in  $x$  und 3 in  $y$ -Richtung.
- Bild 2: Scherung im  $-2$  in  $y$ -Richtung.
- Bild 3: Rotation um  $45^\circ$  gegen den Uhrzeigersinn.
- Bild 4: In  $x$ -Richtung um  $1/2$  stauchen, in  $y$ -Richtung um 3 Strecken und dann um 4 nach rechts verschieben.
- Bild 5: Projektion auf die zur  $x$ -Achse parallele Gerade durch  $(0, 3)$ .

## Aufgabe 4

### Teilaufgabe 4a

TODO

### Teilaufgabe 4b

TODO

### Teilaufgabe 4c

#### Teilaufgabe 4c (I)

TODO

### Teilaufgabe 4c (II)

TODO

### Teilaufgabe 4c (III)

TODO

### Teilaufgabe 4d

Aussage	Wahr	Falsch	Begründung
Texturkoordinaten müssen sich immer im Intervall $[0; 1]$ befinden.	<input type="checkbox"/>	<input type="checkbox"/>	
Texturkoordinaten können als Attribute der Eckpunkte (Vertizes) übergeben werden und werden als solche interpoliert.	<input type="checkbox"/>	<input type="checkbox"/>	
Texturkoordinaten müssen für die Darstellung wie Eckpunktkoordinaten der Model-View-Transformation unterzogen werden.	<input type="checkbox"/>	<input type="checkbox"/>	

## Aufgabe 5: Vorgefilterte Environment-Maps

### Teilaufgabe 5a

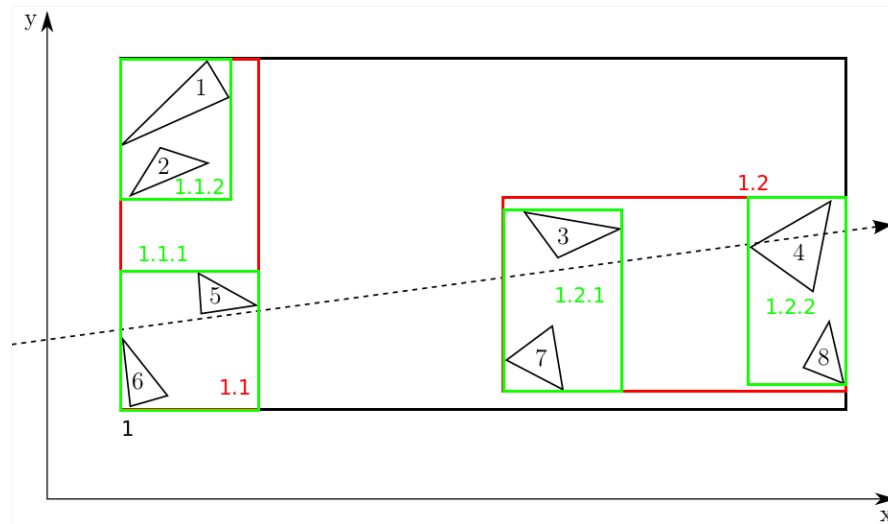
TODO

### Teilaufgabe 5b

TODO

## Aufgabe 6: Hierarchische Datenstrukturen

### Teilaufgabe 6a



### Teilaufgabe 6b

Inklusive Schnittests der AABB Hüllkörper:

1. 1
2. 1.1
3. 1.1.1
4. 5, 6
5. 1.1.2
6. 1.2
7. 1.2.1
8. 3, 7
9. 1.2.2
10. 4, 8

### Teilaufgabe 6c

Aussage	Wahr	Falsch	Begründung
Beim Traversieren eines kD-Baums müssen immer beide Kinder in Betracht gezogen werden.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	vgl. Folie 103
Das Traversieren einer Hüllkörperhierarchie mit achsenparallelen Boxen (Bounding Volume Hierarchy, BVH) erfordert Mailboxing, um mehrfache Schnittpunkte mit einem Dreieck zu verhindern.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Der Speicheraufwand einer BVH hängt logarithmisch von der Anzahl der Primitive ab.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
kD-Bäume sind eine Verallgemeinerung von BSP-Bäumen.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Es ist genau anders herum. kD-Bäume müssen Achsenparallele Trennebenen haben, BSP-Bäume jedoch nicht.
BSP-Bäume sind adaptiv und leiden nicht unter dem „Teapot in a Stadium“-Problem.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

### Teilaufgabe 6d

Aussage	BVH	Octree	kD-Baum	Gitter
Die Datenstruktur partitioniert den Raum.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Der Aufwand für den Aufbau der Datenstruktur ist linear in der Anzahl der Primitive.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Eine effizientere Traversierung wird erreicht, wenn die Surface Area Heuristic bei der Konstruktion verwendet wird.	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Die Datenstruktur eignet sich am besten für Szenen, in denen die Geometrie gleichmäßig verteilt ist und kaum leere Zwischenräume vorhanden sind.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

## Aufgabe 7: Rasterisierung und OpenGL

### Teilaufgabe 7a

Aussage	Wahr	Falsch	Begründung / Quelle
In der OpenGL-Pipeline wird View Frustum Clipping vor der perspektivischen Division durchgeführt. Vertex-Shader können auf Texturen zugreifen. Bei Gouraud-Shading muss man die Normale im Fragment-Shader erneut normalisieren. Gouraud-Shading mit dem Phong-Beleuchtungsmodell kann im Geometry-Shader implementiert werden. Phong-Shading kann man alleine mit einem Vertex-Shader und einem Geometry-Shader implementieren; letzterer gibt dann die Farbe aus. Bei beliebig feiner Tessellierung ist kein Unterschied zwischen Gouraud- und Phong-Shading erkennbar. Selbst wenn der Tiefentest für ein Fragment fehlschlägt, kann der Stencil-Puffer verändert werden. Instanziierung von Geometrie kann man sowohl mit dem Vertex- als auch dem Geometry-Shader durchführen.			

### Teilaufgabe 7b

*Warum zieht man das Tiefenpuffer-Verfahren (Z-Buffering) dem Sortieren von Dreiecken vor? Nennen Sie drei Gründe!*

- Dreiecke können nicht sortierbar sein (wenn ein Dreieck ein andere schneidet)
- TODO
- TODO



## Aufgabe 8: OpenGL-Primitive

- (a) `GL_TRIANGLE_STRIP`: Ganz links ist (1), (2) ist rechts unten davon, (3) ist rechts oben von (1). Dann im Zick-Zack-Muster weiter.
- (b) `GL_TRIANGLE_FAN`: Der mittlere Knoten ist (1), dann wird von ganz links gegen den Uhrzeigersinn nummeriert.

## Aufgabe 9: OpenGL und Blending

### Teilaufgabe 9a

#### Teilaufgabe 9a (I)

- Die Reihenfolge ist wegen des Tiefenpuffers egal: TODO
- Von hinten nach vorne: TODO
- Von vorne nach hinten: TODO

#### Teilaufgabe 9a (II)

`glBlendFunc(TODO, TODO)`

#### Teilaufgabe 9b

`glBlendFunc(TODO, TODO)`

#### Teilaufgabe 9c

`glBlendFunc(TODO, TODO)`

## Aufgabe 10

TODO

## Aufgabe 11: Wasseroberfläche mit GLSL

### Teilaufgabe 11a

\_\_\_\_\_ `shader.frag` \_\_\_\_\_

```
1 vec3 determineIntersection(in vec3 P, in vec3 r, out int index)
2 {
3     // Ermitteln Sie hier den Schnittpunkt mit der nächsten Gefäßfläche
4     // und geben Sie ihn zurück. Zusätzlich muss 'index' auf den Index
5     // der entsprechenden Seitenfläche gesetzt werden.
6
7
8     // TODO
9 }
```

---

## Teilaufgabe 11b

TODO