

## Aufgabe 1: Wahrnehmung und Farbräume

### Teilaufgabe 1a

Eine Grafikkarte ist an ein Anzeigegerät mit einem Gamma-Wert von 2.0 angeschlossen und muss eine entsprechende Gamma-Korrektur durchführen. Berechnen Sie den Intensitätswert, den die Grafikkarte an das Anzeigegerät senden muss, um eine Ausgabe mit der Hälfte der Maximalintensität zu erreichen. (Der Wertebereich der Koeffizienten reicht von 0 bis zur Maximalintensität 1.0.)

Es gilt

$$I_{\text{out}} = I_{\text{in}}^{\gamma}$$

für  $I_{\text{out}} = 1/2$  und  $\gamma = 2$  muss also gelten:

$$I_{\text{in}} = \frac{1}{\sqrt{2}}$$

### Teilaufgabe 1b

Sie haben ein Bild im RGB-Farbraum gegeben und wollen den Helligkeitskontrast erhöhen. In welchen der in der Vorlesung vorgestellten Farbräume wandeln Sie es um, um diese Kontrasterhöhung möglichst einfach durchführen zu können? Welche Berechnung(en) führen Sie dazu auf den Koeffizienten dieses Farbraums aus?

HSV (oder HSI, HSL). Dann wird einfach der V-Wert (I-Wert, L-Wert) erhöht.

### Teilaufgabe 1c

#	Aussage	Wahr	Falsch	Begründung
1	Um den Farbeindruck für einen Menschen eindeutig zu beschreiben, genügt ein Farbmodell mit 3 Koeffizienten.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Graßmansche Gesetze
2	Durch diese 3 Koeffizienten ist dann das Spektrum ebenso eindeutig festgelegt.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Metamerie
3	Der RGB-Einheitswürfel enthält alle sichtbaren Farben.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Magenta / Purple-Line
4	Der RGB-Einheitswürfel enthält Farben, die sich im CIE XYZ-Farbmodell nicht darstellen lassen.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Es ist umgekehrt

## Aufgabe 2: Raytracing und prozedurale Modelle

### Teilaufgabe 2a

- (1) Schattenstrahl zu Lichtquelle verschießen um Verschattung zu testen
- (2) Grundfarbwert anhand eines Beleuchtungsmodells (z.B. Phong) bestimmen
- (3) Reflektionsstrahl verschießen und ermittelten Farbwert auf Farbwert im Punkt addieren.
- (4) Refraktionsstrahl verschießen.

### Teilaufgabe 2b

*Was ist der konzeptuelle Unterschied zwischen Raymarching und Raycasting bzw. Raytracing? Nennen Sie einen Anwendungsfall, bei dem Sie Raymarching verwenden würden, und begründen Sie, warum Sie für diesen Fall Raytracing können*

Bei **Raymarching** wird durch jeden Pixel ein Strahl geschossen und mit einer bestimmten Größe „abmarschiert“. Im Gegensatz zu Raytracing werden keine Sekundärstrahlen verschossen.

Raymarching wird zur Verarbeitung von Volumendaten (Nebel, Wolken, ...) eingesetzt, wobei Raytracing und Raycasting für Oberflächen verwendet wird.

**Anwendungsfall für Raymarching:** Hypertextures. Die Berechnung mittels Raytracing wäre zu aufwendig, da es sehr viele Sekundärstrahlen gäbe.

TODO: Unterschied zwischen Raycasting und Raytracing?

## Teilaufgabe 2c

#	Aussage	Wahr	Falsch	Begründung
1	Prozedurale Modelle erlauben eine kompakte Beschreibung von Texturen oder Objekten, aber die Resultate sind oft nur schwer zu kontrollieren.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
2	Eine ideale Noise-Funktion sollte weder bandbegrenzt sein noch räumliche Korrelationen aufweisen, um erkennbare Periodizitäten und monotone Strukturen zu vermeiden.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
3	Um Turbulenz-Texturen zu erstellen, werden immer höhere Oktaven von Noise-Texturen aufsummiert und dabei immer stärker gewichtet, damit die Resultate so weit wie möglich gegen Weißes Rauschen konvergieren.	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
4	Constructive Solid Geometry ist eine Technik zum Modellieren von Festkörpern, bei der Objekte durch boolesche Operatoren kombiniert werden.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

## Aufgabe 3: Transformationen

### Teilaufgabe 3a

*Um welchen Typ Transformation handelt es sich dabei?*

TODO (Basiswechsel?)

*Nennen Sie die effizienteste Methode, die Inverse dieser Transformation zu finden.*

Man kann leicht feststellen, dass  $u \times v = u \times w = v \times w = 0$ . Es handelt sich also um eine Orthogonalmatrix. Die inverse  $M^{-1}$  einer Orthogonalmatrix  $M$  ist gleich ihrer Transponierten  $M^T$ :  $M^{-1} = M^T$ .

### Teilaufgabe 3b

*Nennen Sie zwei Gründe für die Verwendung hierarchischer Modellierung.*

- (1) Anordnung von Objekten in hierarchischen Gruppen
- (2) Anwendung von Transformationen auf ganze Objektgruppen
- (3) Natürliche Beschreibung eines Objekts / einer Szene

*Welche Datenstruktur haben Sie in der Vorlesung kennengelernt, die dabei hilft, Transformationen beim Traversieren des Szenengraphen zu verwalten?*

Matrixstack

### **Teilaufgabe 3c**

(Homogene 2D-Matrizen)

*Nennen Sie jeweils, um welche Transformation es sich handelt und wie deren Parameter sind.*

- (A) Rotation um  $45^\circ$  gegen den Urzeigersinn.
- (B) Translation um 2.5 in Richtung der  $y$ -Achse.
- (C) Spiegelung an der  $x$ -Achse.
- (D) Scherung um 1 in Richtung der  $x$ -Achse.

## **Aufgabe 4: Texturen**

### **Teilaufgabe 4a**

*Was ist eine Environment-Map?*

Eine Environment-Map ist eine Textur zur Darstellung der Umgebung.

*Wofür wird sie eingesetzt?*

Durch eine Environment-Map kann die Reflektion/Beleuchtung eines Objekts bestimmt werden, ohne aufwendiges Ray-Tracing zu betreiben.

*Welche Annahmen trifft man dabei?*

Die Umgebung ist weit genug entfernt, sodass die Position keine Rolle spielt. Es wird nur die Blickrichtung verwendet.

### **Teilaufgabe 4b**

*Welches Problem beim Texture-Mapping löst Mip-Mapping? Erklären Sie kurz die Idee!*

Mip-Mapping ist eine Vorfilterung der Textur um Aliasing-Artefakte vorzubeugen. Aliasing tritt durch Unterabtastung des Bildsignals auf.

Man erzeugt Mip-Maps durch „zusammenfassen“ von jeweils  $2 \times 2$  Texeln. Es werden solange neue Mip-Map-Stufen generiert, bis man eine Mip-Map erzeugt hat, die nur noch aus einem Texel besteht.

### Teilaufgabe 4c: Bilineare Interpolation

$$c' = (1 - s) \cdot c_{01} + s \cdot c_{11} \quad (1)$$

$$c'' = (1 - s) \cdot c_{00} + s \cdot c_{10} \quad (2)$$

$$c = (1 - t) \cdot c'' + t \cdot c' \quad (3)$$

## Aufgabe 5: Hierarchische Datenstrukturen

### Teilaufgabe 5a

Nennen Sie vier Arten von Hüllkörpern (bounding volumes), die Sie in der Vorlesung kennengelernt haben, und sortieren Sie sie danach, wie eng sie den in der Darstellung gegebenen Zylinder umschließen können.

(A) Slabs (B) OBB (Orientierte Bounding Box) (C) AABB (D) Kugel

### Teilaufgabe 5b

Für welche Datenstrukturen stimmt folgende Aussage? Begründen Sie.

„Der erste während der Traversierung gefundene Schnittpunkt ist zugleich der nächste Schnittpunkt zum Ursprung des Strahls.“

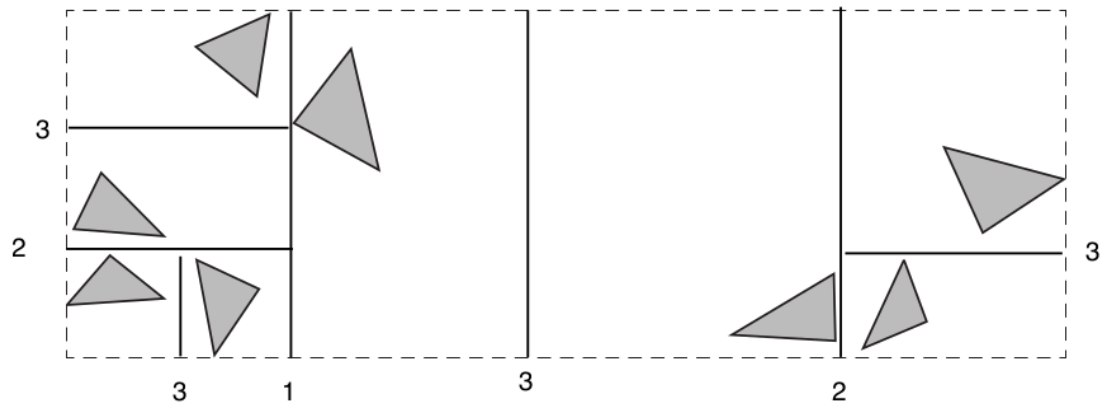
**Octree:** Falsch. Da keine Reihenfolge festgelegt ist in der der Baum traversiert wird.

**BVH:** Falsch.

**kD-Baum:** Wahr. Die Traversierungsreihenfolge des Schnittelalgorithmus ist so festgelegt, dass immer die naheste Ebene zuerst betrachtet wird.

Die Traversierungsreihenfolge ist also von vorne nach hinten.

### Teilaufgabe 5c



## Aufgabe 6: Projektive Transformationen und Clipping

### Teilaufgabe 6a

Transformieren Sie die Vertizes  $P_1$ ,  $P_2$  in homogene Clip-Koordinaten, und geben Sie die homogenen Koordinaten der transformierten Punkte an.

$$P_{1h} = M_{proj} \begin{pmatrix} 2 \\ 6 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ -2 \\ 6 \end{pmatrix} \quad P_{2h} = M_{proj} \begin{pmatrix} 8 \\ 4 \\ 1 \end{pmatrix} = \begin{pmatrix} 8 \\ -8 \\ 4 \end{pmatrix} \quad (4)$$

### Teilaufgabe 6b

Als nächstes müssen Sie die „window edge coordinates“ der projizierten Punkte in Clip-Koordinaten berechnen. Welche Kanten müssen potentiell auf Schnitt mit der Strecke überprüft werden?

$$WEC_{x=w}(P_1) = 4 \quad WEC_{x=w}(P_2) = -4 \quad (5)$$

$$WEC_{x=-w}(P_1) = 8 \quad WEC_{x=-w}(P_2) = 12 \quad (6)$$

$$WEC_{z=w}(P_1) = 8 \quad WEC_{z=w}(P_2) = 12 \quad (7)$$

$$WEC_{z=-w}(P_1) = 4 \quad WEC_{z=-w}(P_2) = -4 \quad (8)$$

$$WEC_{x=w} = w - x \quad WEC_{x=-w} = w + x \quad (9)$$

⇒ Untere rechte Kante überprüfen

### Teilaufgabe 6c

Wenden Sie nun  $\alpha$ -Clipping an. Geben Sie ebenfalls die Endpunkte  $P_1$  und  $P_2$  der Teilstrecke in der Sichtpyramide in normalisierten Device-Koordinaten an.

$$a_{\min} = 0 \quad a_{\max} = 1 \quad (10)$$

$$a_1 = \frac{WEC_{x=w}(P_1)}{WEC_{x=w}(P_1) - WEC_{x=w}(P_2)} = \frac{4}{4 - (-4)} = 0.5 \quad (11)$$

$$\Rightarrow a_{\max} = \min(a_{\max}, a_1) = 0.5 \text{ (weil Optcode } P_2 \text{ gesetzt)} \quad (12)$$

$$a_2 = \frac{WEC_{z=-w}(P_1)}{WEC_{z=-w}(P_1) - WEC_{z=-w}(P_2)} = \frac{4}{4 - (-4)} = 0.5 \quad (13)$$

$$\Rightarrow a_{\max} = \min(a_{\max}, a_2) = 0.5 \text{ (weil Optcode } P_2 \text{ gesetzt)} \quad (14)$$

Neues Liniensegment:

$$(P_{1h} + a_{\min} \cdot (P_{2h} - P_{1h}), \quad P_{1h} + a_{\max} \cdot (P_{2h} - P_{1h})) \quad (15)$$

$$= ((2, -2, 6), \quad (2, -2, 6) + 0.5 \cdot (6, -6, -2)) \quad (16)$$

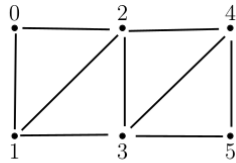
$$= ((2, -2, 6), \quad (5, -5, 5)) \quad (17)$$

$$P'_1 = (1/3, -1/3) \quad P'_2 = (1, -1) \quad (18)$$

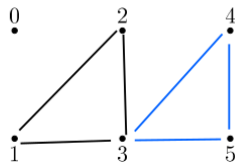
## Aufgabe 7: OpenGL und Rasterisierung

### Teilaufgabe 7a

I. *Triangle Strips* mit Indexbuffer := {0, 1, 2, 3, 4, 5}



II. *Triangle List* mit Indexbuffer := {1, 3, 2, 3, 5, 4}



### Teilaufgabe 7b

---

```
7b.cpp
1 glEnable(GL_DEPTH_TEST); // (1) Aktiviere den Tiefentest
2 glDepthMask(GL_TRUE);   // (2) Aktiviere Schreiben in Tiefenpuffer
3
4 // (3) Lösche Farb- und Tiefenpuffer
5 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
6
7 // (4) Zeichne alle opaken Dreiecksnetze in beliebiger Reihenfolge
8
9 // (5) Deaktiviere das Schreiben in den Tiefenpuffer:
10 glDepthMask(GL_FALSE);
11 glEnable(GL_BLEND); // (6) Aktiviere Blending
12
13 // (7) Sortiere alle transparenten Dreiecksnetze anhand ihrer Bounding-Boxen
14 // und zeichne sie in der Reihenfolge von hinten nach vorne
15
16 glDisable(GL_BLEND); // Deaktiviere Blending
17 glDisable(GL_DEPTH_TEST); // Deaktiviere Tiefentest
```

---



## Aufgabe 8: OpenGL-Shading-Language

### Teilaufgabe 8a: Vertex-Shader

---

```
shader8a.vert
1 # version 140
2 uniform mat4 matMV; // Modelviewmatrix zur Transformation
3                       // von Objekt - nach Kamerakoordinaten
4 uniform mat4 matP; // Projektionsmatrix
5 uniform mat3 matN; // Normalenmatrix
6
7 uniform vec3 I;      // Intensitaet der Lichtquelle I_L
8 uniform vec3 lightPos ; // Lichtposition in Kamerakoordinaten
9
10 in vec3 in_v ; // enthaelt Vertex in Objektkoordinaten
11 in vec3 in_n ; // enthaelt Normalenvektor mit Laenge 1
12 in float kd ; // Materialkoeffizient k_d
13
14
15 void main ( void )
16 {
17     // <solution>
18     gl_Position = //TODO;
19     // </solution>
20 }
```

---

## Teilaufgabe 8b

---

```
1  # version 140
2
3  // <solution>
4  // TODO
5  // </solution>
6
7  out vec4 out_color ; // Ausgabefarbe bestimmt unter Verwendung
8                        // von computeLighting ()
9  void main ( void )
10 {
11     // <solution>
12     // TODO
13     out_color = computeLighting(position, normal, lightPos, kd, I);
14     // </solution>
15 }
```

---

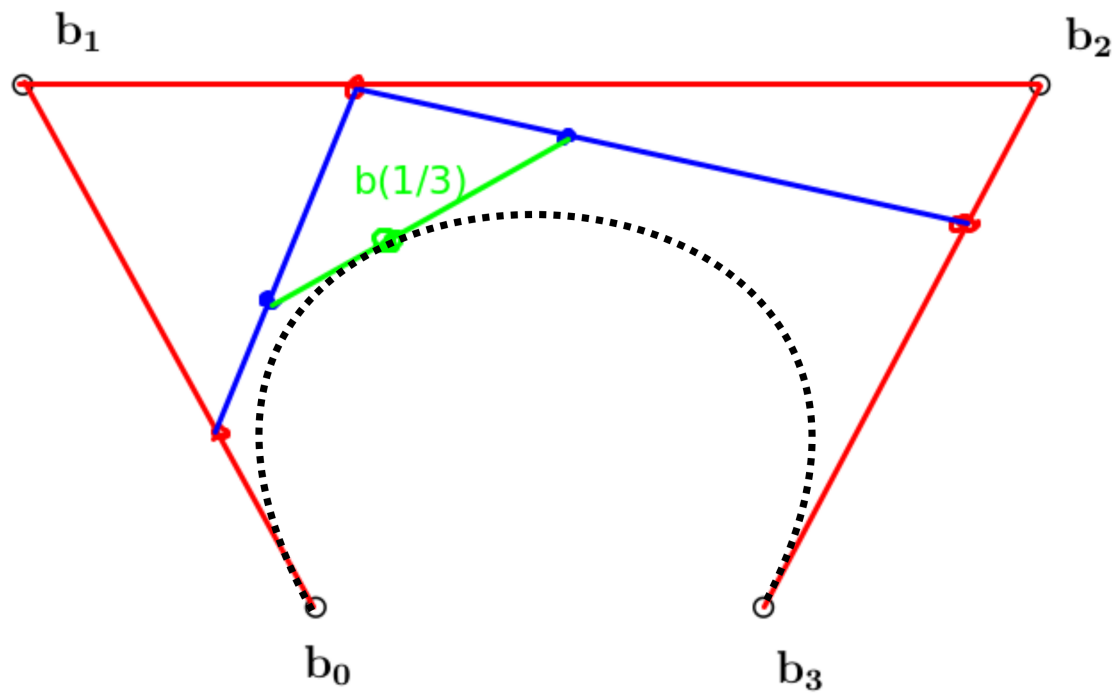
## Aufgabe 9: Bézierkurven und Béziersplines

### Teilaufgabe 9a

Nennen Sie drei wichtige Eigenschaften der Bézierkurven, die Sie in der Vorlesung kennengelernt haben.

- **Tangentenbedingung:**  $c_0c_1$  ist Tangential an die Bezierkurve am Anfang,  $c_2c_3$  ist Tangential an die Bezierkurve am Ende.
- **Wertebereich:** Bézierkurven liegen innerhalb der konvexen Hülle, die durch die 4 Kontrollpunkte gebildet werden.
- **Endpunktinterpolation:** Bézierkurven beginnen immer beim ersten Kontrollpunkt und enden beim letzten Kontrollpunkt.
- **Variationsreduktion:** Eine Bézierkurve  $F$  wackelt nicht stärker als ihr Kontrollpolygon  $B$  ( $\sharp(H \cap F) \leq \sharp(H \cap B)$ ).
- **Affine Invarianz**

### Teilaufgabe 9b



### Teilaufgabe 9c

Kubische Splines sind 2-mal stetig differenzierbar. Damit es immer noch ein kubischer Spline ist, müssen die Punkte  $b_1, b_2, b_3, c_1, c_2$  angepasst werden.