## Teilthema 1: Grundlagen und homogene Koordinaten

**a) Erläutern Sie anhand der gewünschten Wirkweise einer Translation die Notwendigkeit zwischen absoluten und relativen Positions- bzw. Richtungsangaben zu unterscheiden. (VZ, 2 P)**

absoluten Positionen – Punkt im Raum
- Punkte z.B. Mittelpunkt der Ellipse, Ecken des Würfels

relative Größen – Verschiebung von Punkten
- Vektoren z.B. Radien etc.
- sind Positionsunabhängig, absolute Punkte jedoch nicht

Wirkweise:
- Translation: Punkte verändern, aber Vektoren nicht
- Skalierung: Vektoren verändern, aber Punkte nicht

**b) Geben Sie die Definition von „Homogene Koordinaten" an und erläutern Sie wie diese in der Praxis für Punkte und Vektoren umgesetzt wird. (RP, 3 P)**
- Das Quadrupel $[x,y,z,\lambda]^T$ stellt die homogenen Koordinaten des Punktes $[x/\lambda,y/\lambda,z/\lambda]^T \in R_3$ dar.
- Bei homogenen Koordinaten ist ein Flag (**Lambda λ**) was einen Punkt oder einen Vektor definiert.
- $\lambda = 1$ → Punkt || $\lambda = 0$ → Vektor

**c) Erläutern Sie – als Erklärung ohne Beispielrechnung – warum die Ausnutzung der Assoziativität der Matrizenmultiplikation bei der Transformation von vielen Punkten im Raum deutliche Performance-Vorteile bietet. (VZ,3P)**
- Das kostenaufwendigste bei der Matrizenmultiplikation ist die Anwendung von Transformationsmatrizen auf alle Punkte. Diese Transformationsmatrizen zusammenzufassen und dann auf alle Punkte anzuwenden ist weniger kostenintensiv. Da Nutzten wir die Assoziativität aus. Sonst müssen wir pro Punkt immer dieselbe Matrix neu ausrechnen.

**d) Angenommen Sie benutzen eine aus 5 Matrizen zusammen- gesetzte Gesamttransformation um 100.000 Punkte in homogenen Koordinaten (!) zu transformieren. Berechnen Sie die Anzahl der benötigten float-Multiplikationen für folgende Berechnungsvarianten. (RO, 4 P)**

a.) Ausnutzung der Assoziativität der Matrizenmultiplikation und Anschließende Matrix-Vektor Multiplikation (ich gehe von einer 4x4 Matrix aus)
- **Aufwand, um x Matrizen zu einer zu machen**
  = Anzahl der Operationen pro Element * Anzahl Spalten * Anzahl Zeilen * (Anzahl Matrizen - 1)
  = 4 *4 * 4 * (5-1) = **256** (wir nutzten die Assoziativität aus)

- **Matrix-Vektor-Multiplikation** = (Anzahl der Spalten * Anzahl der Zeilen) * Punkte
  = (4 * 4) * 100.000 = **1.6 Mio**. (4x4) Matrix

- Aufwand für Matrix-Vektor-Multiplikation = Aufwand Matrix * (**Matrix-Vektor-Multiplikation**)
  = **256** + **1.6 Mio**.

b.) Abarbeitung jeder einzelnen Matrix-Vektor-Multiplikation nacheinander.
- Aufwand = Anzahl Matrizen * **Matrix-Vektor-Multiplikation**
  = 5 * **(4 * 4 * 100.000)**
  = 8 Mio

**e) Wie wird sich der Faktor zwischen günstiger und ungünstiger Berechnungsweise ändern wenn man die Anzahl der Punkte deutlich erhöht? (TR, 2 P)**
- Er wird sich der Anzahl der benutzten Transformationen (5) annähern, denn der Term (4*4*4)*(5-1) wächst nicht mit, d.h. man vergleicht 4*4* #Punkte mit 5*(4*4)* #Punkte

## Teilthema 2: Transformationen und deren Eigenschaften

**a) Leiten Sie anhand folgender Skizze die Rotationsmatrix für eine 2D-Roation um den Ursprung her. D.h. stellen Sie die Position von P` durch eine entsprechende Berechnung in Abhängigkeit von P dar. (RP, 4 P)**

$x = l \cdot \cos(\alpha)$

$x' = l \cdot \cos(\alpha+\beta)$

$= l \cdot (\cos(\alpha)\cdot\cos(\beta) - \sin(\alpha)\cdot\sin(\beta))$

$= x \cdot \cos(\beta) - y \cdot \sin(\beta)$

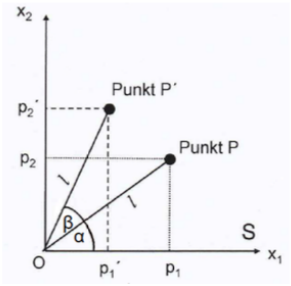
$y = l \cdot \sin(\alpha)$

$y' = l \cdot \sin(\alpha+\beta)$

$= l \cdot (\sin(\alpha)\cdot\cos(\beta) - \cos(\alpha)\cdot\sin(\beta))$

$= x \cdot \sin(\beta) + y \cdot \cos(\beta)$

$\Rightarrow \begin{bmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$




**b) Die Matrix-Multiplikation ist im Allgemeinen nicht kommutativ. Beweisen Sie, dass für eine 2D-Rotation R(a) um den Winkel a und eine 2D-Rotation R(b) um den Winkel b um den Ursprung trotzdem gilt: R(a) • R(b) = R(a + b) = R('b) • R(a) (RO, 4P)**
**Hinweis: Führen Sie dazu eine Rechnung in Matrix-Darstellung in homogenen Koordinaten durch.**

$R(\alpha)\cdot R(\beta) = R(\alpha+\beta) = R(\beta)\cdot R(\alpha)$

$R(\alpha+\beta) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\beta & -\sin\beta & 0 \\ \sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(\alpha+\beta) & -\sin(\alpha+\beta) & 0 \\ \sin(\alpha+\beta) & \cos(\alpha+\beta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$

| | | | $\cos\beta$<br>$\sin\beta$<br>$0$ | $-\sin\beta$<br>$\cos\beta$<br>$0$ | $0$<br>$0$<br>$1$ |
|---|---|---|---|---|---|
| $\cos\alpha$ | $-\sin\alpha$ | $0$ | $\cos\alpha\cdot\cos\beta - \sin\alpha\cdot\sin\beta$ | $\cos\alpha\cdot(-\sin\beta) - \sin(\alpha)\cdot\cos\beta$ | $0$ |
| $\sin\alpha$ | $\cos\alpha$ | $0$ | $\sin\alpha\cdot\cos\beta + \cos\alpha\cdot\sin\beta$ | $\sin(\alpha)\cdot(-\sin(\beta)) + \cos\alpha\cdot\cos\beta$ | $0$ |
| $0$ | $0$ | $1$ | $0$ | $0$ | $1$ |

$= \begin{bmatrix} \cos(\alpha+\beta) & -\sin(\alpha+\beta) & 0 \\ \sin(\alpha+\beta) & \cos(\alpha+\beta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} \cos\beta & -\sin\beta & 0 \\ \sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$

| | | | $\cos\alpha$<br>$\sin\alpha$<br>$0$ | $-\sin\alpha$<br>$\cos\alpha$<br>$0$ | $0$<br>$0$<br>$1$ |
|---|---|---|---|---|---|
| $\cos\beta$ | $-\sin\beta$ | $0$ | $\cos\beta\cdot\cos\alpha - \sin\beta\cdot\sin\alpha$ | $\cos\beta\cdot(-\sin\alpha) - \sin(\beta)\cdot\cos\alpha$ | $0$ |
| $\sin\beta$ | $\cos\beta$ | $0$ | $\sin\beta\cdot\cos\alpha + \cos\beta\cdot\sin\alpha$ | $\sin\beta\cdot(-\sin(\alpha)) + \cos\beta\cdot\cos\alpha$ | $0$ |
| $0$ | $0$ | $1$ | $0$ | $0$ | $1$ |

$= \begin{bmatrix} \cos(\alpha+\beta) & -\sin(\alpha+\beta) & 0 \\ \sin(\alpha+\beta) & \cos(\alpha+\beta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$

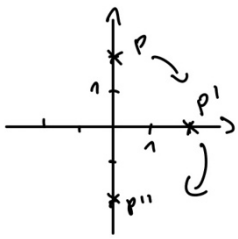
**Additionstheoreme**

$$\sin(x \pm y) = \sin x \cdot \cos y \pm \cos x \cdot \sin y$$

$$\cos(x \pm y) = \cos x \cdot \cos y \mp \sin x \cdot \sin y$$

**c) Geben Sie eine Beispiel-Konfiguration als Skizze an, bei der die Hintereinander-Ausführung zweier Rotationen nicht kommutativ ist. (TR, 3 P)**

- Man nehme zwei Rotationen, die nicht um denselben Punkt gehen Rotiere z.B. einen Punkt bei 2,0 um 90° um den Ursprung und danach um 90° um den Punkt (0,2) => Punkt liegt bei 0,2. Wende die Rot umgekehrt an => Punkt liegt irgendwo im 2. Quadranten mit neg x und pos y Koordinate




**d) Erläutern Sie warum eine Translation in homogenen Koordinaten einen Punkt verschiebt, einen Vektor aber unverändert lässt. Geben Sie dazu die strukturelle Funktionsweise der entsprechenden Matrix- Vektor Multiplikation an. (RP, 2 P)**

- Bei homogenen Koordinaten ist ein **Lambda λ** was einen Punkt oder einen Vektor definiert ( 0 = Vektor || 1 = Punkt).
- Da die Translationsangaben in der rechten Spalte sind werden diese mit dem **λ** multipliziert.
  Falls **λ** = 0 ist wird wie gewollt nichts geändert.
  Bei einem Punkt mit **λ** = 1 wird etwas geändert

$$\begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ \lambda \end{bmatrix} = \begin{bmatrix} x + t_x \cdot \lambda \\ y + t_y \cdot \lambda \\ z + t_z \cdot \lambda \\ \lambda \end{bmatrix}$$

$\lambda = 0$, falls Vektor
$\lambda = 1$, falls Punkt

**Welche Vorteile ergeben sich aus der Verwendung von homogenen Koordinaten hinsichtlich der Effizienz? Begründen Sie.**

- Man spart sich die if-Abfrage isRelative() ob es ein Punkt oder ein Vektor ist, da die if-Abfrage eine elementare Operation ist und sehr sehr oft aufgerufen wird→ viel Schneller

**e) Geben Sie die prinzipielle Vorgehensweise einer Rotation um eine beliebige Achse an. Listen Sie die notwendigen Arbeitsschritte in der richtigen Reihen- folge auf und Erklären Sie jeden Arbeitsschritt in einem Satz. (RP, 4 P)**

Annahme: befindet sich im Ursprung

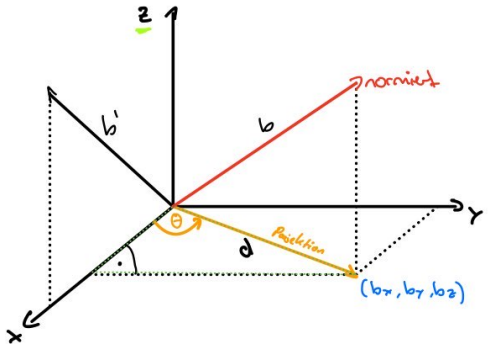
1. **Schritt 1:** Dreh den Vektor b in die zx-Ebene (damit es etwas sortierter ist)
2. **Schritt 2:** Dreh den resultierenden Vektor b` auf die z-Achse (die neue Richtung fällt in die z-Achse)
3. **Schritt 3:** Dreh den resultierenden Vektor b`` um den Winkel alpha
4. **Schritt 4:** Alles am Ende wieder Rückwärts rechnen

$$R_b(\alpha) = R_z(\theta) * R_y(\phi) * R_z(\alpha) * R_y(-\phi) * R_z(-\theta)$$

**f) Erläutern Sie die erste notwendige Teilrotation einer Rotation um eine beliebige Achse gemäß der in der VL vorgestellten Reihenfolge im Detail. Leiten Sie also die entsprechenden Berechnungsformeln her und ergänzen Sie ihre Ausführungen mit einer beschrifteten Skizze. (RP, 4 P)**

**Schritt 1: Dreh den Vektor b in die zx-Ebene (etwas sortierter)**

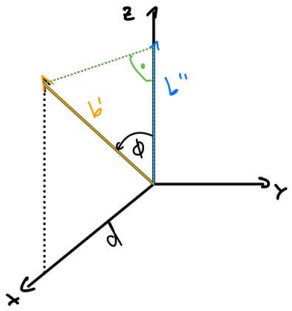
- Annahme: Achse geht durch den Ursprung **und** b normiert!
- Die Achse b um einen Winkel -tetta um die z-Achse rotieren. Ergebnis b‘




$$\cos(-\theta) = \cos(\theta) = \frac{b_x}{d}$$
$$\sin(-\theta) = \sin(\theta) = \frac{b_y}{d}$$
$$\boldsymbol{R_z(-\theta)}$$

**Schritt 2: Dreh den resultierenden Vektor b` auf die z-Achse (die neue Richtung fällt in die z-Achse)**

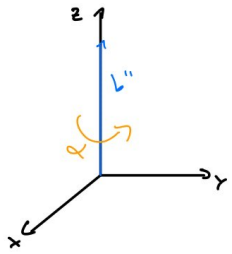
- Die nächste Rotation geht auf die z-Achse um die y-Achse.




$$\cos(-\phi) = \cos(\phi) = \frac{b_z}{1} = \boldsymbol{b_z}$$
$$\sin(-\phi) = \sin(\phi) = -\frac{d}{1} = \boldsymbol{-d}$$
$$\boldsymbol{R_y(-\phi)}$$

**Schritt 3: Dreh den resultierenden Vektor b`` um die z Achse um den Winkel alpha**

- Nun folgt die eigentliche Rotation um alpha **(Drehung um die z-Achse)**




$$\cos(\alpha)$$
$$\sin(\alpha)$$
$$\boldsymbol{R_z(\alpha)}$$

**Schritt 4:** Alles am Ende wieder Rückwärts rechnen

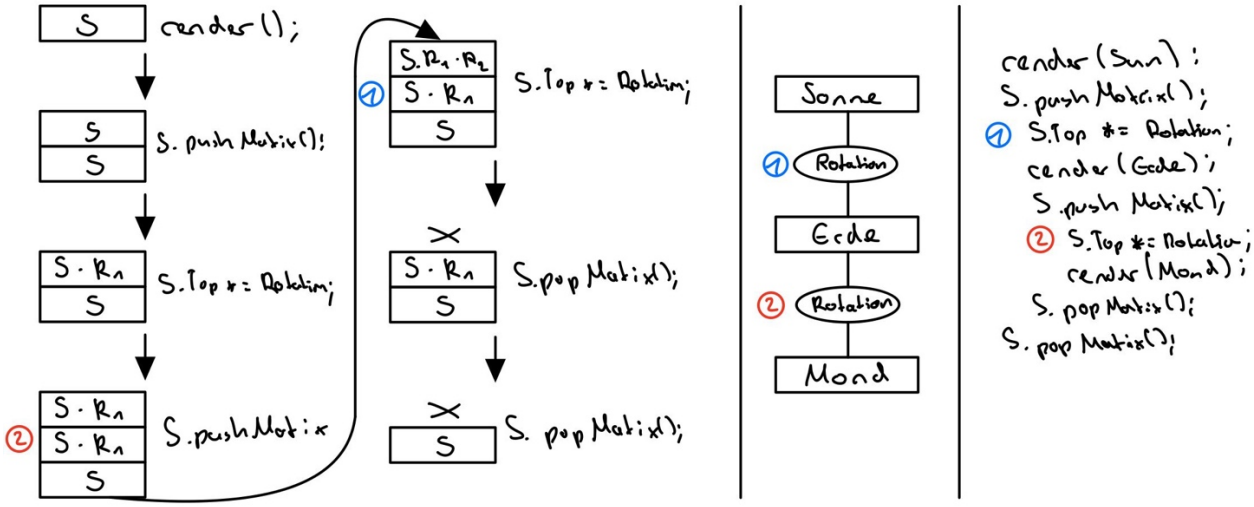
$$R_b(\alpha) = R_z(\theta) * R_y(\phi) * R_z(\alpha) * R_y(-\phi) * R_z(-\theta)$$

wieder zurück rotieren auf z in xy-Ebene

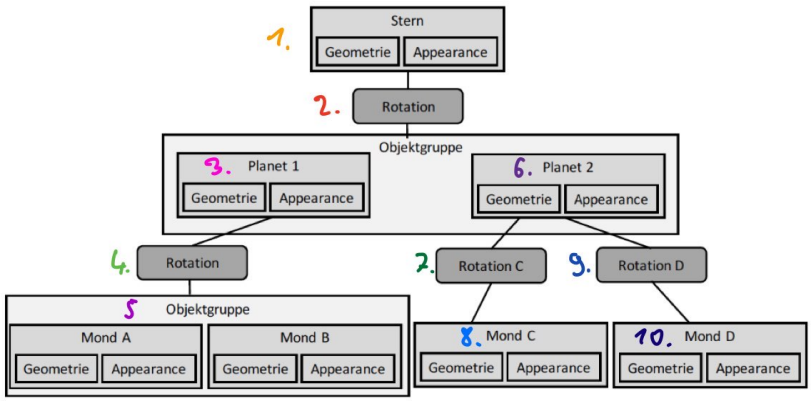
Teilthema 3: Szenegraph

**a) Erläutern Sie die Funktionsweise des Matrix-Stacks bei der Rendering-Operation basierend auf einem gegebenen Szenegraphen. (RP, 4 P)**

- Am Anfang besteht der Matrix-Stack s aus einer Matrix, die Transformationsmatrix des Sterns
- Der Graph wird nach Depth First Search die Kinder durchiterieren bei der Abarbeitung des Stacks
- Während jeder Iterationsstufe wird s.top() gepusht und die Transformationsmatrix des aktuellen Knotens drauf multipliziert und dieser Knoten dann gerendert, solange es Kinder gibt
- Wenn es keine Kinder gibt, wird s.pop() gemacht und die Transformationsmatrix quasi "vergessen"




**b) Geben Sie die Prinzipielle Abarbeitung des folgenden Szenegraphen beim Rendering mit Hilfe eines Matrix Stacks als Pseudocode an (RP, 4 P)**




```
1.) render(Stern, S.top());
    S.pushMatrix();
        2.) S.Top() *= Rot1;
        3.) render(Planet1, S.top());
            S.pushMatrix();
                4.) S.Top() *= Rot2 ;
                5.) render(Mond A, S.top());
                    render(Mond B, S.top());
            S.popMatrix();
        6.) render(Planet2, S.top());
            S.pushMatrix();
                7.) S.Top() *= RotC ;
                8.) render(Mond C, S.top());
            S.popMatrix();
            S.pushMatrix();
                9.) S.Top() *= RotD ;
                10) render(Mond D, S.top());
            S.popMatrix();
    S.popMatrix();
```

**c) Erläutern Sie welche Struktur ein Szenegraph haben muss, damit die Skalierung eines Planeten-Durchmessers unabhängig von der Skalierung seines Mondes (der im Baum relativ zum Planeten positioniert ist) möglich wird. Hinweis: Mit Übung RO, sonst TR.**

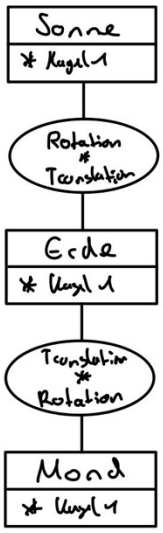
- Man muss eine leere Objektgruppe haben, die am Zentrum des Planeten platziert ist. Der Planet ist als Kind davon aufgehangen aber nicht verschoben.
- Der Mond ist auch als Kind von der leeren Objektgruppe angehangen, hängt also an der Mitte des Planeten aber nicht an dem Planeten selbst.
- Skalierung des Planeten wirkt sich also nicht auf den Mond aus. Trotzdem kann man beide gemeinsam um die Sonne rotieren, wenn die Rotation in der Matrix für die Objektgruppe realisiert wird.

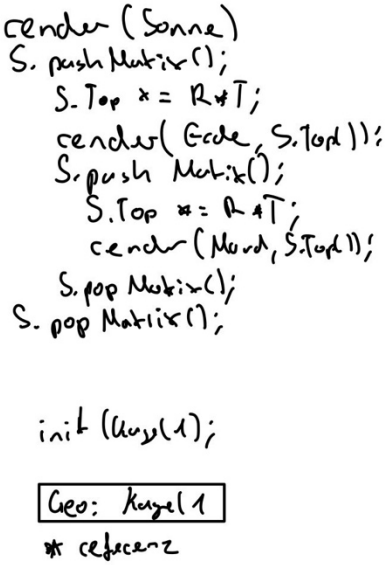
**d) Geben Sie einen Szenegraphen an, die folgende Szene nachbildet: (RO, 3 P)**

**I. Eine Sonne bildet das Zentrum.**
**II. Die Erde kreist um die Sonne**
**III. Ein Mond kreist um die Erde**
**IV. Alle Objekte sollen dieselbe Geometrie-Darstellung einer Kugel benutzen.**

**Hinweis**: Beachten Sie dazu abweichend von den vereinfachten Beispielen aus der VL auch die notwendigen Translationen

- Eigentlich easy, statt Rotations-Matrix R schreibt man überall R*T und packt so noch eine Translationsmatrix T mit drauf. Von rechts nach links wird gearbeitet (also R*T nicht T*R), da der Punkt von rechts an die Matrix multipliziert wird also erst aus der Mitte raus bewegen, dann um die Mitte rotieren. Referenz dann auf ein geometrisches Kugelobjekt dazu zeichnen und fertig.




**e) Erläutern Sie die beiden prinzipiellen Möglichkeiten wie beim Rendering der Objekte eines Szenegraphen die jeweils passende Auflösung eines mehrfach verwendeten geometrischen Grundobjekts ermittelt werden kann. (RP, 4 P)**

**Erweiterungsmöglichkeit A: Factory**

- Diese Factory erzeugt und verwaltet die Objekte gibt das Objekt in der passenden Auflösung und Detailliertheit zurück.

**Erweiterungsmöglichkeit B: Objektselektierung**

- Automatische Selektion der passenden Auflösung für ein geometrisches Objekt durch CgScenegraphEntity selbst

- Beide Methoden benötigen die Kameraposition und die Projektionsart, um die passende Auflösung zu ermitteln
- Durch eine Factory, die vom Szenengraphen gesteuert wird. Die Factory erstellt dann Objekte in der passenden Auflösung auf Nachfrage.
    - Vorteil: weniger Compilerzeit, da nicht mehrere Objekte in der GPU hinterlegt werden. Geeignet bei wenig Objektwechsel
    - Nachteil: größeres nachladen in Laufzeit beim Wechseln von Objekten
- Objektselektierung. jede Entity sind Referenzen von Objekten hinterlegt, die auch in der GPU initialisiert sind. Durch ein Selektionsmechanismus wird das passende Objekt ausgewählt.
    - Vorteil: schnellere Laufzeit bei vielen Objektwechseln
    - Nachteil: mehr Compilerzeit, weil alle Objekte am Anfang zur GPU gehen

Andere Fragen:
a) Erläutern Sie wie beim Rendering der Objekte in einem Szenegraph die Abarbeitung mit einem Matrix-Stack realisiert wird?

• Kann jede Skalierung invertiert werden? (1P)