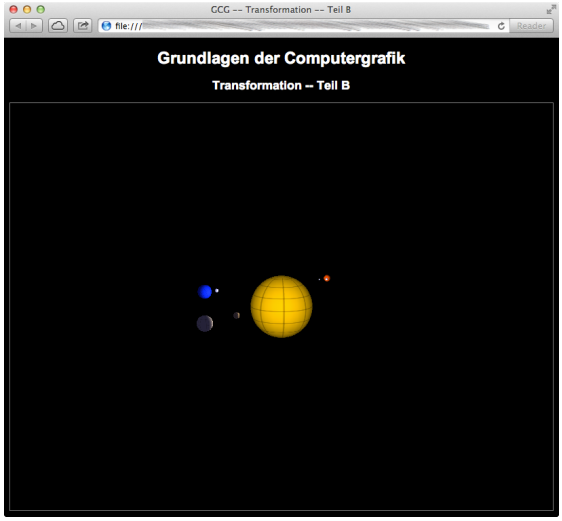
Grundlagen der Computergrafik — Praktikum

Transformation

Ziel

Am Ende dieses Praktikumsteils haben Sie einen Szenengraphen samt zugehörigen Transformationen für ein animiertes Planetensystem selbst implementiert:



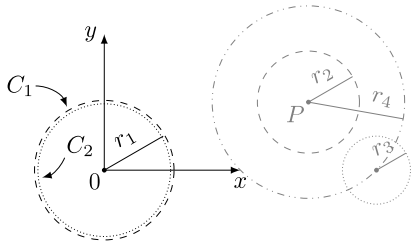
Vorbereitung

Wiederholen Sie die Inhalte aus dem Vorlesungsabschnitt „Transformation“. Richten Sie besonderes Augenmerk auf die verschiedenen Transformationsmatrizen, die Multiplikationsreihenfolge sowie den Unterschied zwischen Punkt- und Normalentransformation.

Übung (Rechnen)

Diese Aufgaben dienen zur **Vorbereitung** der Programmieraufgaben. Nutzen Sie sie zur Vorbereitung auf den Praktikumstermin.

Die folgende Abbildung dient zur Erläuterung der Aufgaben:



1. Transformation Teil 1

Gegeben sei der Kreis C_1 (gestrichelt) mit Mittelpunkt im Koordinatenursprung und Radius r_1.

Gesucht ist die Transformationsmatrix M_1 (für Spaltenvektoren), die diesen Kreis so transformiert, dass sich sein Mittelpunkt an Position $P = (x_p, y_p)$ befindet und sein Radius r_2 nur noch drei Viertel des ursprünglichen Radius r_1 beträgt.

a) Stellen Sie die Transformationsmatrizen für die einzelnen Transformationsschritte auf.

b) Stellen Sie die Berechnungsvorschrift für M_1 auf. Berechnen Sie diese aber nicht.

c) Stellen Sie die Berechnungsvorschrift für eine Transformationsmatrix auf, mit der zugehörige Normalen transformiert werden können. Berechnen Sie diese nicht.

2. Transformation Teil 2

Gegeben sei der Kreis C_2 (gepunktet) mit Mittelpunkt im Koordinatenursprung und ebenfalls Radius r_1.

Gesucht ist die Transformationsmatrix M_2 (für Spaltenvektoren), die diesen Kreis so transformiert, dass sein Radius r_3 nur noch die Hälfte des ursprünglichen Radius r_1 beträgt und sein Mittelpunkt auf einer Kreisbahn mit Radius r_4 um Position $P = (x_p, y_p)$ rotiert werden kann. Verwenden Sie als Rotationswinkel die Variable α.

a) Stellen Sie die Transformationsmatrizen für die einzelnen Transformationsschritte auf.

b) Stellen Sie die Berechnungsvorschrift für M_2 auf. Berechnen Sie diese aber nicht.

c) Stellen Sie die Berechnungsvorschrift für eine Transformationsmatrix auf, mit der zugehörige Normalen transformiert werden können. Berechnen Sie diese nicht.

3. Hierarchische Transformation

Welchen Transformationsschritt samt zugehöriger Transformationsmatrix beinhalten sowohl M_1 als auch M_2?

4. Szenengraph

Skizzieren Sie einen Szenengraphen für die beiden Kreise samt zugehörigen Transformationen. Nutzen Sie gemeinsame Transformationsschritte um die Szenenhierarchie abzubilden.

5. Gesamttransformation

Überlegen Sie, wie sich die Transformationsmatrizen für die Kreise mit Hilfe des Szenengraphen bilden lassen.

Es kann sein, dass sich eine andere Multiplikationsreihenfolge als in den vorherigen Aufgaben ergibt.

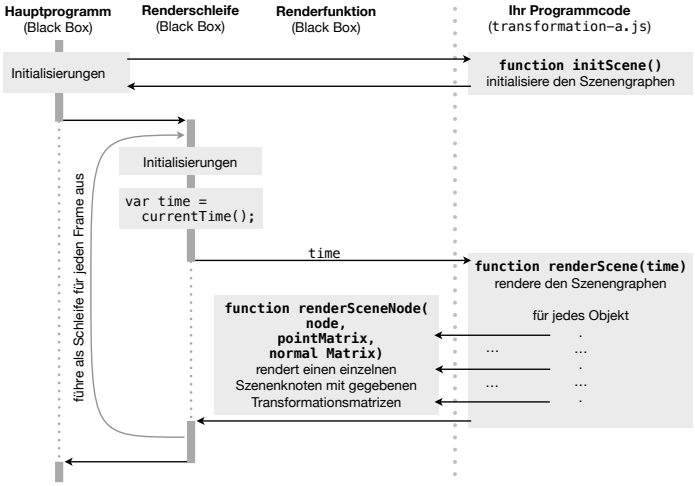
Prüfen Sie, ob die geänderte Reihenfolge zum selben Ergebnis wie vorher führt.

Zusammenspiel von Black Box und Ihrem Code

In diesem Praktikumsteil beinhaltet die Black Box ebenfalls eine Renderschleife, sodass eine Animation des Planetensystems möglich ist. Zusätzlich wird eine Renderfunktion bereitgestellt, die Sie aufrufen müssen, um einen Planeten zu rendern. Während der Initialisierung können Sie in der Funktion `initScene()` den vollständigen Szenengraphen anlegen.

Zum Rendering eines Frames ruft die Renderschleife der Black Box Ihre Funktion `renderScene(…)` auf. Die verstrichene Zeit seit dem Start in Sekunden wird als Parameter übergeben. In der Funktion `renderScene(…)` bestimmen Sie für jeden Szenenknoten die Transformationsmatrizen für Punkte und Normalen. Anschließend rufen Sie für jedes einzelne Objekt der Szene die Funktion `renderSceneNode(…)` der Black Box auf und übergeben den jeweiligen Szenenknoten und die beiden zugehörigen Transformationsmatrizen als Parameter.

Einen Überblick über diesen Ablauf gibt die folgende Abbildung.



Szenenknoten, Matrizen etc.

Datenformat

Knoten in einem Szenengraphen enthalten eine (zeitabhängige) Transformation sowie ggf. ein oder mehrere Kinder und ggf. ein Geometrieobjekt. In der Funktion `initScene()` in der Datei `transformation-b.js` wird bereits ein Szenenknoten für die Sonne erstellt (*Hinweis:* Auf die Anführungszeichen um die Attributnamen wurde verzichtet.):

```
var sun = {
    animator: animateSun,
    shape: CreateSun(),
    children: []
};
```

Dem Attribut `.animator` wird die Funktion `animateSun` als Attributwert zugewiesen. Diese kann später aufgerufen werden. Im Attribut `.shape` wird der Rückgabewert der Funktion `CreateSun()` aus der Black Box gespeichert. Diese Funktion legt für die Sonne eine Kugel in der richtigen Größe mit dem passenden Material an. Das Attribut `.children` enthält zu Beginn lediglich ein leeres Array `[]`.

Wichtig: Nach dieser Initialisierung kann die Funktion `sun.animator(…)` tatsächlich aufgerufen werden, da der Funktionsname von `animateSun(…)` abgelegt wurde. Im Gegensatz dazu enthält `sun.shape` lediglich ein Objekt[1], da die Funktion `CreateSun()` bereits aufgerufen und ihr Rückgabewert abgelegt wurde.

Animator-Funktionen

Für jeden Szenenknoten sollten Sie eine Animator-Funktion implementieren, die eine zeitabhängige Transformation bestehend aus zwei Matrizen zurückgibt. Die Datei `transformation-b.js` enthält mit der Funktion `animateSun(…)` bereits eine solche Animator-Funktion. Wenn Sie den jeweiligen Funktionsnamen im Attribut `.animator` eines Szenenknoten speichern, können Sie sie später über diesen Szenenknoten aufrufen (s.o.).

Nützliche Bedingungen

Es sei `var node = { … }` ein Szenenknoten (s.o.).

Die Bedingung `(node.animator != undefined)` ist wahr, wenn der Knoten das Attribut `.animator` enthält. Sie überprüft nicht, ob es einen sinnvollen Wert enthält!

[1] Für Interessierte, die einen Blick in die Black Box werfen: Tatsächlich handelt es sich nur um einen Indexwert, der das Objekt identifiziert.

Die Bedingung `(node.shape != undefined)` ist wahr, wenn der Knoten das Attribut `.shape` enthält. Sie überprüft nicht, ob es einen sinnvollen Wert enthält!

Die Bedingung `(node.children != undefined)` ist wahr, wenn der Knoten das Attribut `.children` enthält. Sie überprüft nicht, ob es sinnvolle Werte enthält!

Die Bedingung `(node.children.length > 0)` ist wahr, wenn das Array im Attribut `.children` des Knotens Einträge enthält. Sie überprüft nicht, ob es sinnvolle Werte enthält!

Hinweis: Mithilfe dieser Bedingungen können Sie reine Transformationsknoten erstellen. Fügen Sie diesen Knoten kein Attribut `.shape` hinzu. Gleiches gilt für reine Geometrieknoten. Fügen Sie diesen Knoten kein Attribut `.animator` hinzu.

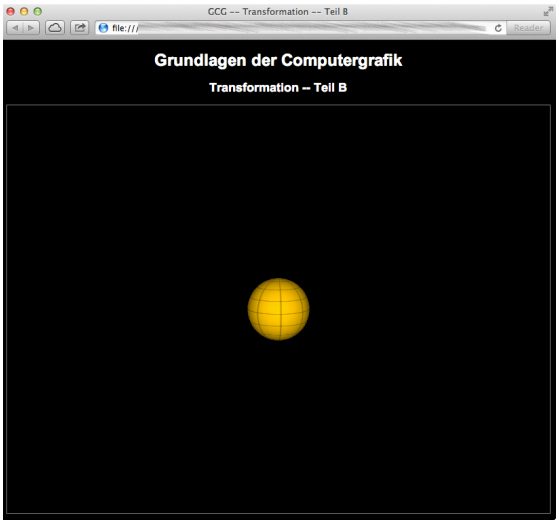
Wichtig: Die Matrix `matrixA` muss zuvor als Matrix angelegt und initialisiert worden sein.

Aufgaben (Programmierung)

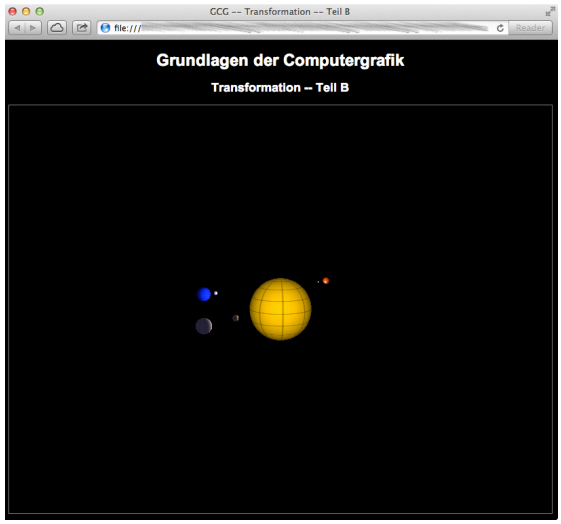
Öffnen Sie die Datei `transformation-b.html` in Ihrem Browser. Sie sollten auf der Html-Seite einen Rahmen sehen, in dem sich die Sonne als gelbe Kugel im Koordinatenursprung befindet. Der Äquator der Sonne liegt in der x/z-Ebene, die beiden Pole liegen auf der y-Achse.

Durch Klicken und Ziehen mit der Maus im Rahmen können Sie die Kamera um die Szene bewegen.

Die Szene wird immer direkt von der Sonne aus beleuchtet.



zu Beginn



am Ende dieses Praktikumsteils

Für die Planeten sowie für die Monde gibt es wie für die Sonne jeweils eine Erstellungsfunktion. Abgesehen von der Sonne werden alle Planeten als Einheitskugel im Ursprung erstellt.

Die folgende Tabelle listet die Erstellungsfunktionen sowie Durchmesser, Eigenumdrehungszeit, Bahnradien, Umlaufzeiten, sowie für die Zusatzaufgaben auch die Achs- und Bahnneigung auf. Die Planeten- und Mondradien sind relativ zueinander korrekt skaliert. Gleiches gilt für die Bahnradien, sowie Eigenumdrehungszeiten und Umlaufzeiten. Die Planeten- und Mondradien stehen allerdings nicht im richtigen Verhältnis zu den Bahnradien.

Planet / Mond	Erstellungs-Funktion	Radius	Bahn-radius	Eigenum-drehungs-zeit	Umlaufzeit	Achs-neigung	Bahn-neigung
Merkur	`CreateMercury()`	5	76	58,7	88	0°	7°
Venus	`CreateVenus()`	12	145	243	224,7	177°	3°
Erde	`CreateEarth()`	13	200	1	365,2	23°	0°
Mars	`CreateMars()`	7	305	1	687	25°	2°
Mond (Erde)	`CreateMoon()`	3,5	22	27,3	27,3	7°	5°
Phobos (Mars)	`CreateMoon()`	2,5	15	0,3	0,3	0°	1°
Deimos (Mars)	`CreateMoon()`	1,5	20	1,3	1,3	0°	1°

1. Szenegraphen zeichnen

Erstellen Sie einen Szenegraphen auf dem Papier, welcher die Konstellation des Planetensystems mit seinen zugehörigen Transformationen wiedergibt. Vergessen Sie dabei nicht „Root“ als Wurzel des Szenegraphen!

Scannen Sie diesen Szenegraphen und fügen Sie die Datei Ihrer Abgabe hinzu.

2. Erde auf Umlaufbahn hinzufügen

Erstellen Sie in der Funktion `initScene(…)` einen neuen Szenenknoten für die Erde und fügen Sie ihn als Kind dem Szenenknoten der Sonne hinzu.

Schreiben Sie eine geeignete Animator-Funktion, die bewirkt, dass sich die Erde um die eigene Achse und sich in der richtigen Größe auf der Umlaufbahn um die Sonne bewegt (siehe Tabelle).

Fügen Sie der Funktion `renderScene(…)` einen Codeblock hinzu, der die Transformationsmatrizen für die Erde mit Hilfe der Animator-Funktion ermittelt und die Erde anschließend korrekt transformiert rendert.

Möglicherweise müssen Sie den Inhalt der Variable `timeScale` anpassen, um die Eigenrotation der Erde visuell überprüfen zu können.

Stellen Sie sicher, dass im Bildergebnis zu sehen ist, dass die Sonne die Erde beleuchtet.

Hinweis: Verwenden Sie für diese Aufgabe das Wissen, dass die Erde das erste Kind (Nummer 0) der Sonne (also des Szenenknoten) ist.

Wichtig: Achten Sie darauf, dass die Matrizen `pointMatrix` und `normalMatrix` zu Beginn jedes Frames die richtigen Wert enthalten.

3. Szenengraph / Hierarchische Transformation

Implementieren Sie für die aktuelle Szene den Szenengraphen aus Aufgabe 1. Jeder Szenenknoten sollte dabei zunächst lediglich einen einzelnen Transformationsschritt implementieren.

Ergänzen Sie Ihren Programmcode um eine neue Funktion, die die einzelnen Knoten des Szenengraphen **rekursiv** verarbeitet, die benötigten Transformationen ermittelt und schließlich die Objekte rendert. Rufen Sie diese neue Funktion aus der Funktion `renderScene(…)` auf.

Erstellen Sie für jeden Transformationsschritt, den Sie in der vorherigen Aufgabe verwendet haben eine eigene Funktion, die die zugehörige Transformation erstellt – also wie zuvor auch ein Paar aus Transformationsmatrix für Punkte und Transformationsmatrix für Normalen. Verwenden Sie diese Funktionen.

4. Mond auf Umlaufbahn hinzufügen

Fügen Sie Ihrem Szenengraphen den Mond hinzu, sodass er korrekt um die Erde kreist.

Wichtig: Achten Sie darauf, dass sich die Transformationen der Kindknoten nicht gegenseitig beeinflussen.

5. Verbleibende Planeten und Monde hinzufügen

Fügen Sie die verbleibenden Planeten und Monde hinzu.

Zusatzaufgaben (freiwillig, ohne Wertung)

6. Bahnneigungen

Fügen Sie Szenengraphknoten für die jeweiligen Bahnneigungen hinzu.

7. Achsneigungen

Fügen Sie Szenengraphknoten für die jeweiligen Achsneigungen hinzu.

8. Jahreszeiten

Sorgen Sie dafür, dass wechselnde Jahreszeiten entstehen, falls noch nicht geschehen..

Hinweis: Überlegen Sie, wie sich die Planeten um die Sonne bewegen.

Bearbeitung, Abgabe, Bewertung

Die Abgabe erfolgt gruppenweise. Ein Gruppenmitglied lädt dazu eine Zip-Archiv der Dateien (`transformation.js` und `planeten_szenegraph.jpg`) auf der Moodle-Seite des Kurses hoch.

Wichtig: Tragen Sie unbedingt Ihre Gruppe, den Studiengang sowie die einzelnen Gruppenmitglieder im führenden Kommentarblock der Datei ein. Benennen Sie die Datei folgendermaßen:

`STUDIENGANG-GRUPPE-NAME1-NAME2-transformation.zip` (Beispiel: `BMT-A-MayerRichter-transformation.zip`)

Punkteverteilung (10 Punkte)

Aufgabe 1:	1 Punkt
Aufgabe 2:	1 Punkt
Aufgabe 3:	2 Punkte
Aufgabe 4:	2 Punkte
Aufgabe 5:	2 Punkte
Aussagekräftige Kommentierung:	1 Punkt
Symmetrie/Performance:	1 Punkt