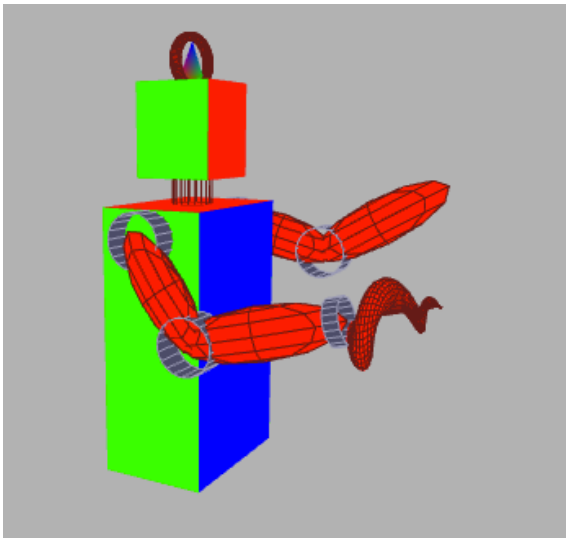
Computergrafik 2: Aufgabe 2.3

Der Roboter: Szenengraph und Animation



Lernziele / Motivation

In dieser Übung machen Sie sich mit der Transformation von Modellen und Szenen durch Matrizen vertraut und modellieren und animieren eine Szene hierarchisch.

Erweiterung des Übungsframeworks

Erweitern Sie Ihren Code aus Aufgabe 2 wie folgt beschrieben. Laden Sie von Moodle die Datei `scene_node.js` herunter und kopieren Sie diese neben die `scene.js` in Ihrem Projektverzeichnis. Studieren Sie den Aufbau von `SceneNode` anhand der Moduldatei und des SU-Handouts.

Legen Sie ein neues Modul `models/robot.js` an, in welchem Ihr Robotermodell definiert wird. Das Modul wird für seine Arbeit potentiell `scene_node.js`, `gl-matrix.js` sowie die anderen Modelle aus `models/` benötigen. Wie jedes andere Modell benötigt der Roboter einen Konstruktor und eine `draw()`-Methode.

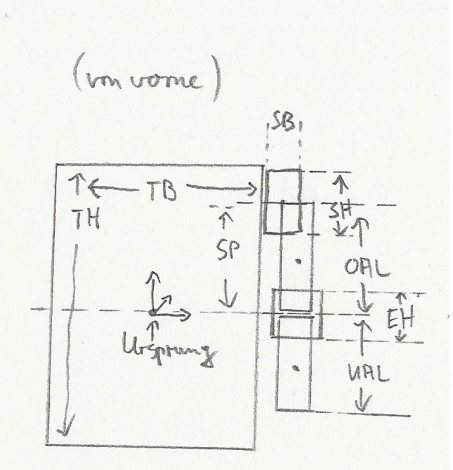
Aufgabe 2.3.1: Konstruktionsplanung

Planen Sie einen Fantasie-Roboter, der mindestens eine Extremität mit mindestens zwei Gliedmaßen und Gelenken besitzt. Fügen Sie der Extremität noch ein Handgelenk mit einer Hand oder einem Greifer oder rotierenden Gerät hinzu. Der Roboter soll sich mindestens aus Würfeln, Bändern und parametrischen Flächen Ihrer Wahl

zusammensetzen. Für eine sehr gute Note sollte Ihr Roboter möglichst nicht-trivial und sehr systematisch konstruiert sein.

Fertigen Sie für die Bearbeitung und Demonstration der Aufgabe einen Konstruktionsplan Ihres Roboters an und planen Sie darin, welches Körperteil seinen Koordinatenursprung wo haben wird. Aus der Zeichnung soll hervorgehen, welche Einzelteile der Roboter besitzt und wie diese Einzelteile relativ zu Ihren Abmessungen zueinander transformiert sind (*Skelett*).

Benennen Sie zunächst die Einzelteile des Roboters und geben Sie den wichtigen Größen symbolische Bezeichner (z.B. OAL = Oberarm-Länge). Später in Ihrem Code sollen sich bei den Transformationen im Skelett möglichst wenige absolute Zahlenwerte, sondern Formeln unter Verwendung dieser Bezeichner wiederfinden (z.B. "transformiere um die halbe Oberarmlänge", **nicht** "transformiere um 2.374"). Wird im Code der Bezeichner auf einen anderen Wert gesetzt, soll das Modell sich entsprechend konsistent anpassen.



Bitte beachten: eine elektronische Kopie dieses Plans gehört in Ihre `.zip`-Datei zur formalen Abgabe via Moodle. Anhand des Plans sollen Sie bei der Abnahme den Aufbau des Roboters / Szenengraphen erläutern.

Aufgabe 2.3.2: Hierarchischer Szenenaufbau mit Skelett und Haut

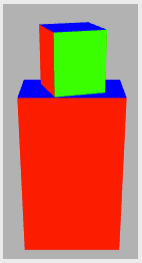
Fügen Sie in Ihrer Szene eine neue Option "Show Robot" hinzu, die den Roboter in der Szene zeichnet. Setzen Sie Ihren Konstruktionsplan im Roboter-Modul in Code um. Erstellen Sie mit Hilfe von `SceneNode` Skelett und Haut des Modells getrennt, so dass z.B. Skalierungen der Haut sich nicht auf das Skelett und die übrigen Haut-Teile auswirken. Beachten Sie die Hinweise aus 2.2.2.

Wie Sie dem Modul `SceneNode.js` und dem unten folgenden Beispiel entnehmen können, können Sie alle Knoten in beliebiger Reihenfolge erzeugen und dann einen Knoten einem anderen mittels `SceneNode.add()` als Kind zuweisen (z.B. um das Skelett zu konstruieren, oder um die Skins an das Skelett zu hängen).

Über den Setter/Getter `SceneNode.transform()` können Sie die Transformations-matrix eines jeden Knoten direkt manipulieren. Diese Transformation versteht sich in einem Szenengraphen immer relativ zum Elternknoten. Um entsprechende Transformationsmatrizen zu erzeugen, verwenden Sie bitte die Funktionen des Moduls `gl-matrix.js`

Folgendes Beispielmodul erzeugt Roboter aus zwei Quadern (Torso und Kopf) mit Skelett und Haut:

```
var Robot = function(gl, programs, config) {

    // Komponenten zum Bau des Roboters
    var cube    = new Cube(gl);

    // Dimensionen der in der Zeichnung benannten Teile
    var headSize  = [0.3, 0.35, 0.3 ];
    var torsoSize = [0.6, 1.0,  0.4 ];

    // Skelett für Torso und Kopf (ups, Hals vergessen...)
    this.head = new SceneNode("head");
    mat4.translate(this.head.transform(), [0, torsoSize[1]/2+headSize[1]/2, 0]);
    this.torso = new SceneNode("torso");
    this.torso.add(this.head);

    // Skins
    var torsoSkin = new SceneNode("torso skin");
    torsoSkin.add(cube, programs.vertexColor);
    mat4.scale(torsoSkin.transform(), torsoSize );
    var headSkin = new SceneNode("head skin");
    headSkin.add(cube, programs.vertexColor)
    mat4.rotate(headSkin.transform(), 0.6*Math.PI, [0,1,0]); // wegen der Farben!
    mat4.scale(headSkin.transform(), headSize);

    // Verbindung Skelett + Skins
    this.torso.add(torsoSkin);
    this.head.add(headSkin)


}; // constructor

Robot.prototype.draw = function(gl, program, transformation) {
    this.torso.draw(gl, program, transformation);
};
```



Wie Sie im Beispiel sehen, erwartet die `draw()`-Methode dieses Modells ein Programm und eine Transformation, welche an den Wurzelknoten des Szenengraphen (hier: Torso) weitergereicht werden. Die `Scene` sollte die Welt-Transformation an `Robot.draw()` übergeben (in diesem Fall `Scene.transformation`), damit die Rotation der Welt auch auf den Roboter übertragen wird.

In diesem Beispiel übernimmt und kennt Konstruktor des Roboters die Programm-Objekte, die in der Szene bereits angelegt wurden.

Aufgabe 2.3.3: Bewegung der Gelenke

Wie Sie vielleicht schon bemerkt haben, kann die gesamte Szene in dieser Aufgabe mittels der Tasten Y und Shift-Y bzw. X und Shift-X um die Y- bzw. X-Achse der "Welt" rotiert werden (falls es mit dem Roboter nicht gleich geht, versuchen Sie es mit Triangle/Cube/Band). Welche Taste welche Rotation bewirkt, ist in `main.js` definiert. Der `HtmlController` ruft bei jedem Tastendruck die Methode `Scene.rotate()` auf. In der JavaScript-Konsole sehen Sie den Tastencode der gedrückten Taste.

Erweitern Sie den Code in `main.js` für alle Gelenke Ihres Roboters, und rufen Sie in `scene.js` aus `Scene.rotate()` heraus für diese Gelenke eine neue Methode `Robot.rotate()` auf (denn die Szene kennt ja den Roboter), welche Roboter-intern die verschiedenen Gelenke kennt und die entsprechenden Transformationsmatrizen im Szenengraph verändert.

Aufgabe 2.3.4 : Animation (optional, nur für eine sehr gute Note)

In `main.js` sehen Sie die Animationsfunktion, welche die Szene um die Welt-Y-Achse rotiert, indem Sie die Methode `Scene.rotate()` aufruft.

Erweitern Sie diese Funktion und animieren Sie einige Gelenke Ihres Roboters, indem Sie abhängig von der Animationszeit entsprechende `Scene.rotate()`-Aufrufe tätigen. Mindestens eine der Bewegungen sollte oszillieren (z.B. Kopfnicken, Arm auf und ab, ...).

Tipp: der Parameter **`t`** in der Animationsfunktion ist die **Zeit seit Start der Animation** in Millisekunden. Der Parameter **`deltaT`** enthält die **seit dem letzten Aufruf der Animationsfunktion verstrichene Zeit**. Mit Hilfe von `deltaT` können Sie recht einfach dafür sorgen, dass Ihre Animation eine gleichmäßig schnell läuft, auch wenn die Animationsfunktion nicht in exakt regelmäßigen Abständen aufgerufen wird.

Noch ein Tipp: falls es ruckelt, kommentieren Sie in der `main()` den Aufruf der Funktion `WebGLDebugUtils.makeDebugContext()` aus. Der Code wirft dadurch weniger aussagekräftigere Fehlermeldungen, wird aber deutlich effizienter.

Abgabe

Diese Aufgaben ist einer von mehreren Teilen der Aufgabe 2. Die Abgabe der gesamten Aufgabe 2 soll via Moodle bis zu dem dort angegebenen Termin erfolgen. Verspätete Abgaben werden wie in den Handouts beschrieben mit einem Abschlag von 2/3-Note je angefangener Woche Verspätung belegt. Geben Sie bitte pro Gruppe jeweils nur eine einzige `.zip`-Datei mit den Quellen Ihrer Lösung sowie **mit einer elektronischen Kopie des geforderten Konstruktionsplans** ab.

Demonstrieren und erläutern Sie dem Übungsleiter Ihre Lösung *in der nächsten Übung nach dem Abgabetag*. Die Qualität Ihrer Demonstration ist, neben dem abgegebenen Code, ausschlaggebend für die Bewertung! Es wird erwartet, dass alle Mitglieder einer Gruppe anwesend sind und Fragen beantworten können.