CG1 WS 20/21 - Exercise 1: Robot

Technische Universität Berlin - Computer Graphics
Date 12. November 2020 **Deadline** 25. November 2020
Prof. Dr. Marc Alexa, Ugo Finnendahl, Max Kohlbrenner

Scene graph / Affine transformation (7 Points)

In this exercise, you learn how to work with a scene graph on the example of a simple robot model. The graph is built and traversed using the `Three.js` framework and you have to apply affine transformations on the objects in the scene. We provide a `Typescript` skeleton project and a screencast where the demanded functionalities are demonstrated which will be available on ISIS.

In `Three.js` the `THREE.Scene` represents the scene graph and consists of multiple `THREE.Object3D` objects. Their local coordinate system is implicitly defined by the homogeneous transformation (a 4×4 matrix) from it to the objects parent coordinate system stored in the `matrix` property. **Restriction:** If not otherwise stated, we only allow direct manipulation of this transform matrix in this exercise. Rotation, translation and scaling of objects must be performed via matrix multiplication with an appropriate matrix. When creating each object, it is mandatory to set its `matrixAutoUpdate` property to `False` and you are not allowed to call the `updateMatrix` method. Additionally all transformation matrices need to be built from scratch, you are not allowed to use the convenience functions of the `THREE.MatrixX` class (`X` $\in \{3, 4\}$) that enables you to create rotation, translation or scaling matrices without setting their elements by hand (e.g. `makeRotationX`, `setPosition`...). You can however use functions for matrix multiplication and the matrix inverse.

Tasks:

1. Construct a scene graph to represent a robot, which allows you to add child and sibling nodes to existing ones. It is not necessary to match your implementation to the robot showed in the presentation video, however, the constructed scene graph should have at least a depth of two. *(1 point)*

2. Select individual nodes visibly in the scene graph using the keyboard. The scene graph can be traversed by pressing

   - w, which selects the parent node,
   - s, which selects the first child node,
   - a, which selects the previous sibling node and
   - d, which selects the next sibling node. (*1 point*)

3. Implement a functionality that allows you to rotate the selected node using the arrow keys. The origin of each rotation should be at the joints of the objects. Depending on how you created your scene graph up to now, this might require some refinements. You need to move the local coordinate systems of each object to the desired rotation center. It may be necessary to change the position of the objects geometry (`THREE.Geometry`) within its coordinate system. As this only changes the local position of the vertices it is independent from the transformation matrix and can therefore be solved using helper functions from `Three.js`. (*2 point*)

4. Display the local coordinate system of the selected node at its origin using axes. The axes should be displayed as red, green and blue lines for x, y and z, respectively (you can use `THREE.AxesHelper`). The visibility of the axes can be switched by **pressing** c. **Restriction:** First, the axis position and orientation must be defined in *global* (world) coordinates, the axis must therefore be a direct child of the main `THREE.Scene` object. Second, only the local transformation matrix is allowed for the calculation (i.e. especially the usage of the `matrixWorld` property is forbidden). *Hint:* Choose the right place for the update function call, as the axes need to be updated whenever the node of the robot is rotated. (*2 point*)

5. Implement a reset functionality that restores the initial pose of each node. This functionality should be triggered by **pressing** r. Do not construct a new scene graph but traverse the existing graph and reset all changes in the matrices of the local transforms. (*1 point*)

Requirements

- Exercises must be completed individually. Plagiarism will lead to exclusion from the course.
- Submit a `.zip` file of the `src` folder of your solution through ISIS by **25. November 2020, 23:59**.
- *Naming convention*: {firstname}_{lastname}_cg1_ex{#}.zip (for example: jane_doe_cg1_ex1.zip).
- You only hand in your `src` folder, make sure your code works with the rest of the provided skeleton.