CG1 WS 22/23 - Exercise 1: Robot

Technische Universität Berlin - Computer Graphics
Date 27. October 2022 **Deadline** 09. November 2022
Prof. Dr. Marc Alexa, Dimitris Bogiokas, Ugo Finnendahl, Max Kohlbrenner, Markus Worchel

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 the local coordinate system to the objects parent coordinate system stored in the `matrix` property. The matrix relevant for the GPU is the transformation from local coordinates to world coordinates. This matrix is stored in the `matrixWorld` property. **Restriction:** In this exercise we only allow direct manipulation of these transformation matrices. Rotation, translation and scaling of objects must be performed via matrix multiplication with an appropriate matrix on to `matrix`. And after that you have to update the `matrixWorld` of that object and all its children. As `Three.js` conveniently does that all implicitly for us, we need to set `THREE.Object3D.DefaultMatrixAutoUpdate` to `false` in the beginning of the application to ensure that restriction. Do not remove that line otherwise you will lose points. 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`, `scale`, `setPosition`...). You can however use functions for matrix multiplication and the matrix inverse.

Tasks:

1. Construct a scene graph that consists of nodes, where each node represents one part of a robot. It is not necessary that your implementation matches the robot shown in the presentation video, however, the constructed scene graph must have at least a depth of two and there must exist a node that has at least two siblings. **Hint:** A node may consist of multiple `THREE.Object3d` instances. **Note:** You have to define the pose of each robot part by setting the `matrix` property by hand and then updating the `matrixWorld` by traversing the scene graph up to the root. *(2 point)*

2. Select individual nodes visibly in the scene graph using the keyboard. Traverse the scene graph 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. Display the local coordinate system of the selected node (or all nodes) 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 should be switched by **pressing** *c*. (*1 point*)

4. Allow changes (e.g. rotations) of the `matrix` property in the current selected node and update all `matrixWorld`s of the children depending on that `matrix`. (*1 point*)

5. 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. (*1 point*)

6. 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 transformations. (*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 **09. November 2022, 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.