# CG1 WS17/18 - Exercise 1

Technische Universität Berlin - Computer Graphics
**Date** 26. October 2017. **Deadline** 15. November 2017
Prof. Dr. Marc Alexa
Xi Wang <xi.wang@tu-berlin.de>
David Lindlbauer <david.lindlbauer@tu-berlin.de>

## EXERCISE 1 : Scene graph / Affine transformation

Implement a robot with moving parts using WebGL and Three.js based on the provided `Javascript` skeleton code. This includes constructing a scene graph and applying affine transformations based on existing functionality in the Three.js framework. A possible solution is shown in the video that is included in the .zip archive (*robot.mp*4). It's not mandatory to use this skeleton code and feel free to use your own framework. The tasks in detail are:

1. Construct a scene graph that is suitable to represent a robot. It should be possible to add child and sibling nodes to existing nodes. It is not necessary to match your robot the one shown in the video, however the scene graph of the constructed robot should have a depth of at least two. *(1 point)*

2. Implement selecting individual nodes in the scene graph using keyboard input. You should be able to traverse the scene graph by pressing $w$ (select parent node) and $s$ (select first child node) and select sibling nodes by pressing $a$ (select previous sibling) and $d$ (select next sibling). (*1 point*)

3. Implement transforming selected nodes in the scene graph using keyboard input (e.g., cursor keys). Note that the center of rotation does not have to be the centroid of the node. *Hint:* Rotation around arbitrary points in Three.js can be achieve through nesting geometry. (*1 point)*

4. Display the local coordinate system of each node in its center of rotation. Axes should be displayed as red, green, and blue for $x$, $y$ and $z$, respectively. You should be able to switch the visibility of the coordinate systems by pressing $c$ The coordinate systems should rotate with the nodes, however should not be considered when selecting child nodes. (*1 point*).

5. Implement a reset functionality that restores the initial pose of each node. This functionality should be triggered by pressing $r$. (*1 point*).

**Bonus points (optional) :**

6. Implement picking which can select and transform object based on mouse input. You can use the existing Three.js raycasting functionality for the selection. *Hint:* the existing orbit control can be toggled with the property *enabled*. (*1 point*)

7. Animate the robot. By pressing $k$, the robot should move based on predefined rotations. Feel free to use an existing library such as Tween.js (`https://github.com/tweenjs/tween.js/`). (*2 point*)

8. In addition to the provided orbit interaction, implement your own version of a trackball camera. Do not use the existing Three.js implementation of the trackball camera. This functionality should be triggered by pressing $t$. (*1 point*)

**Requirements**

- Exercises must be completed individually. Plagiarism will lead to exclusion from the course.
- Please submit a zipped version of your solution through ISIS by **15. November 2017, 23:59**.
- *Naming convention*: {firstname}_{lastname}_cg1_ex{#}.zip (for example: jane_doe_cg1_ex1.zip).
- The zip must be standalone and include the whole code and all libraries, including html and js files.