Multi-Level Crane

How to start?

- Copy the *03_crane* folder from */opt/vr_exercises/WS_15_16* in the vr-lab cluster to a local repository.
- The accompanying video *demo_video.mp4* in the *doc* folder illustrates the final result.
- Execute the application by running ./start.sh in a terminal.
- Proceed with the assignments.

Assignment Tasks (**graded**):

The maximum number of students per group is two. The presentation of the results takes place on **Friday 20.11.2015** in the lab class.

Figure 1: A multi-level crane, consisting of three segments. Keyboard input is used to rotate the individual hinges (red discs), thus moving the whole sub-structures accordingly.

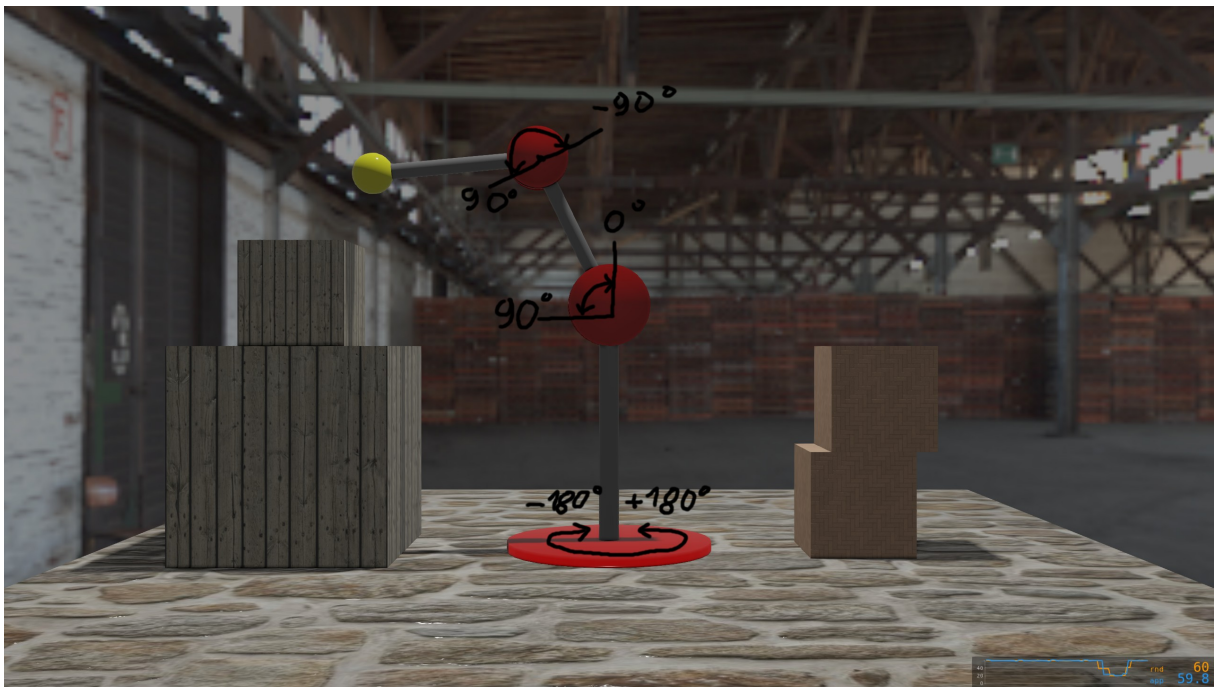

Figure 2: Rotation-axis and rotation-constraints of each hinge.

Figure 3: The hook at the final level of the crane (yellow ball) checks for box intrusion. If inside, the respective box is recolored to red.

1. **[30%]** Set-up a multi-level crane, as shown in Figure 1. The crane should consist of three levels, This includes three hinges (red discs) as well as three arm-segments (grey cylinders). The first hinge (I in Figure 1) should enable rotations arround it's local y-axis. The second and the third one (II && III in Figure 1) allow for rotation arround the local z-axis. The classes *Hinge* and *ArmSegement* are already given. Include respective scenegraph nodes as member variables into these classes. Use geometry nodes to load and parametrize basic cylinder shapes. Eventually init further transform nodes, that help you to structure the geometric aspects of the elements. Keep in mind that the global geometric transformation of a node is the accumulation of all local transformations of all it's parent nodes on the path towards the scengraph root. Init and parametrize the different hinge and arm-segments in the class *Crane*. Tip: Proceed step-wise with one segement after the other to check your results.
2. **[25%]** The *KeyboardInput* class provides three rotation input values as field variables. Internally the four arrow keys, the page-up and page-down keys are used to control the three rotations. Connect these input fields into the respective hinge classe's input fields. This can be done in the class *Crane* where the keyboard input and all the hinges are instanciated. Then accumulate the connected input in the *field-has-changed* callback of the *Hinge* class to the respective hinge node.
3. **[10%]** At the final level of the crane, a hook (represented by a yellow sphere in Figure 3) is attached. The *Hook* class is already given. It provides a callback function, that checks if the position of the hook is inside of any of the target boxes. Init the necessary node(s) and connect the final hook transformation to the callback variable.
4. **[25%]** Include rotation-constraints for the hinges. These constraints can be defined as constructor parameters in the *Hinge* class. Use these constraints to limit the rotation input accumulation to the min- and max- values given in Figure 2.
5. **[10%]** Include a frame-rate independent input mapping for all hinges. Therfore the rotation input values in the class *KeyboardInput* have to be adapted to the actual frame-rate. Pressing the CTRL key toogles the application frame-rate between 20 and 60Hz.