CG1 WS17/18 - Exercise 2

Technische Universität Berlin - Computer Graphics
Handout 16. November 2017, **Due** 29. November 2017
Prof. Dr. Marc Alexa, David Lindlbauer, Xi Wang

Exercise 1 : Projective transformations / Canonical viewing volume

This exercise is built on the Nate Robins' OpenGL projection tutorial and it extends the visualization with the canonical viewing volume (normalized device coordinates). Skeleton code using Three.js is provided on the course page and a possible solution can be found on the server (`http://cybertron.cg.tu-berlin.de/p/cg1-ws17/ex2/index.html`). The skeleton code contains three sub-windows which are (from left to right) world space view, canonical viewing space and screen space view. A teddy bear is rendered as the main object in the scene and a GUI interface is included using the library dat.GUI. The tasks in detail are:

1. A perspective camera together with its frustum is visualized in world space view. Several related parameters are listed in the nested camera menu in the GUI including the field of view of the camera, aspect ration, near and far planes and so on. Implement a function that update the perspective camera when any of the parameters is changed. (*1 point*)

2. Implement a function that apply the corresponding transformations to the teddy bear when variables in the model manipulation menu is changed. (*0.5 point*)

3. Change the camera used for rendering screen space view so that it shows the image seen by the perspective camera in world space. (*0.5 point*)

4. Implement a function that transforms the object into the canonical viewing space (normalized device coordinates, NDC) and visualize it in the second window. (*2 points*)

   Hints:

   - Camera coordinate system is left-handed.
   - Camera position as well as its projection are stored in matrices which are properties of the camera object in Three.js .
   - Object in Three.js also have several matrices which store its location as well as its relative position to the root node (if certain hierarchy exists).
   - Use `updateMatrix()` and `updateMatrixWorld()` properly.
   - `verticesNeedUpdate` needs to be true if you want to update the vertices.

5. Add six clipping planes in clip space to simulate the canonical viewing volume (i.e. anything outside the viewing volume are removed). Extend the GUI so that the clipping planes can be toggled on and off. (*1.0 point*)

6. Extend the GUI with another boolean variable which toggles the orbit control over the clip space. (*0.5 point*)

7. Modify the `onWindowResize()` function in the *htmlcontroller* so that the sub-windows are scaled properly when the size of the browser window changes. (*0.5 point*)

   Bonus points (optional) :

8. Implement a function to interpolate between the current camera and a destination which is defined by its position and orientation. Related parameters can be defined by user in GUI. *Hint:* Tween.js (`https://github.com/tweenjs/tween.js/`) can be used to interpolate the camera positions. `THREE.Quaternion.slerp` can be used to interpolate between quaternions. (*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 **29. November 2017, 23:59**.
- *Naming convention*: {firstname}_{lastname}_cg1_ex{#}.zip (for example: jane_doe_cg1_ex2.zip).
- The zip must be standalone and include the whole code and all libraries (i.e. all html and js files).