# Lab 5

- Animation
- Instantiating more than one model in the scene
- Composing model transformations with global scene transformations
- Removing hidden faces using the *"Depth-Buffer"*

## 1.1 Animation

Analyze the incomplete example **WebGL_example_18.html**.

Identify the main changes regarding the previous example:

- The auxiliary functions in the **webgl-utils.js** file, which allow developing code that is independent from the used browser.

- The function **animate(),** which allows updating the parameters of the applied transformations as a function of elapsed time.

- The parameters controlling the applied transformations are defined as global variables.

- The function **tick(),** which allows periodically redrawing the contents of the *canvas*.

- In this way, function **drawScene()** is no longer called after processing each event.

- When a model is visualized using perspective projection, a translation is applied to place it inside the view volume – see **drawScene()**.

- Controlling the rotation around the YY axis using buttons – see the respective *event listeners*.

**Task:**

- Add the possibility of rotating around the XX and ZZ axes, and controlling those transformations using buttons.

Suggestions:

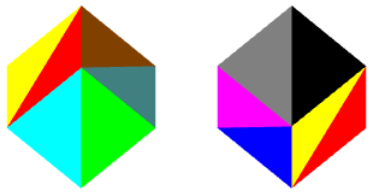
- Animate a horizontal movement.

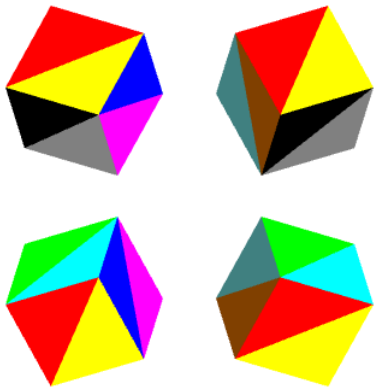
- Use the keyboard to control the transformations that can be applied to a model.

### 1.2 Instantiating more than one model in the scene

Based on the previous one, develop a new example: **WebGL_example_19.html**.

It should be possible to visualize more than one cube: for instance, two cubes, in a first version, and four cubes, in a second version.





For the visualization using perspective projection, it has to be ensured that all models are inside the view volume.

**Tasks:**

- Modify, in an appropriate way, the **drawScene()** function.
- Develop an animate scene where each cube has a different behavior.
- Start with a simple scenario, where each cube rotates around its vertical axis, possibly with different rotation speeds and directions.
- Afterwards, add the possibility of rotating around the other coordinate axes.

**Question:**

- Is it enough to use ***"Back-Face Culling"*** to correctly visualize the various cubes? Why?

Suggestion:

- Develop an auxiliary function that allows instantiating and applying local transformations to a cube.

### 1.3 Composing model transformations with global scene transformations

Analyze the example **WebGL_example_20.html**.

Identify the main change regarding the previous example:

- In addition to their own transformations, the two models display a global behavior.

**Questions:**

- What is the behavior of each model?
- Is any one of the models always closer to the camera?
- Is there any "odd" behavior during the animation?

**Tasks:**

- Enable/disable the **depth test** (*"Depth-Buffer"*), so that the visibility of each model face is correctly computed.
- Understand the usefulness of the **depth test** (*"Depth-Buffer"*), so that models are correctly rendered.
- Understand when hidden faces can be removed using ***"Back-Face Culling"***, and when using the **depth test** (*"Depth-Buffer"*) is mandatory.

Analyze the example code, in particular:

- The function **drawScene()**, which allows visualizing and applying different transformations to the models.
- The function **drawModel()**, which allows visualizing and applying a transformation to a model, that results from the concatenation of the global scene transformation and the particular model transformations.
- Note how global scene transformations are defined and applied.

**Tasks:**

- Control the rotation direction and the rotation speed around the vertical axis.
- Apply other global scene transformations and control their parameters. For instance, allowing rotations around the other coordinate axes.