Lab 4

- Modular development using WebGL
- Reading a model from a file
- Projection type and definition of the view volume

1.1 Reading models from file

Analyze the incomplete example **WebGL_example_16.html**.

Notice how the code is organized into several files. In particular, analyze:

- The file **WebGL_example_16.html**. Note: (1) for ease of development, each shader's code is still defined in this file; (2) how the **.js** files are included.

- The file **WebGL_example_16.js**. Verify the additional functionalities that have been defined: in particular, the possibility of reading the content of a **text file**, with a simple format, defining a **3D model**.

- The file **maths.js**, which allows instantiating and operating with matrices and vectors to, for instance, define the global transformation matrix e apply elementary transformations.

- The file **initShaders.js**, which allows reading in the shaders' code and instantiate the shaders.

Identify the new functionalities that are now available:

- Reading the file containing the model to be visualized. Analyze the file format being used.

- Visualization of the model in three different modes: only the vertices, the edges or the triangular faces. **Attention: the code for this functionality has to be completed in the function drawScene().**

Tasks:

- Complete the code and test the three visualization modes for a given model.

- Create additional files containing the definition of simple 3D models: triangular prism, quadrangular pyramid, etc.

- Visualize the new models created; ensure that all triangular faces are correctly defined.

Suggestion:

- Represent the 3D models using JSON.

1.2 3D Projections and definition of the view volume

Analyze the incomplete example **WebGL_example_17.html**.

Identify the main changes regarding the previous example:

- The additional functions in the **maths.js** file, which allow instantiating the orthogonal projection matrix and the perspective projection matrix, by defining the associated view volumes.

- The possibility of choosing the type of projection used for visualizing the 3D models.

Analyze the code of this example, in particular:

- How the choice of the projection type is done and how the projection matrix is instantiated.

- How the projection matrix is passed as an argument to the vertex-shader.

- The concatenation of the projection matrix with the global transformation matrix – note that the projection is the last transformation to be applied to each vertex.

Tasks:

- Visualize each model using orthogonal projection and parallel projection.

- Ensure, when using perspective projection, that the model to be visualized is inside the view volume.

- Apply different transformations to each model and verify their effect.

- Develop functionalities that allow modifying the features of the view volume, for each projection type.

Suggestion:

- Modify the code in order to simultaneously visualize two models.