Exercise 5-a: Loading a json model

Make a copy of the previous *Exercise 4* (see *Figure 1*) and start from that point. This way you will have available interactive manipulation of the camera transformations.

Following the instructions given in the slides, make the appropriate changes and add the necessary code to load a model from a .json file. The result should look similar to *Figure 2*. You can use the two models available in the virtual campus: teapot.json and laptop.json.

In a nutshell, these are the steps you will have to follow:

- Add the model loading functions provided in the slides (*loadModelOnGPU* and *handleLoadedModel*). The function *handleLoadedModel* will create and fill the vertex and index buffers handled by the previously declared global variables.
    - Store the buffer handlers in global variables so that they are later available from the *drawScene* function.
- In the *webGLStart* method, call the function *loadModelOnGPU*.
- Modify the function drawScene to draw the model geometry.
    - In this case, as we use an element buffer to store vertex indices, you will use the function *glDrawElements* instead of *glDrawArrays*.
    - **NOTE: By default, the teapot is too big. Scale it down so you can see it!**
- Modify the shaders conveniently to remove all texture-related stuff and paint the fragments with a plain color. Also, remove all texture-related stuff also from the webGL / javascript code.

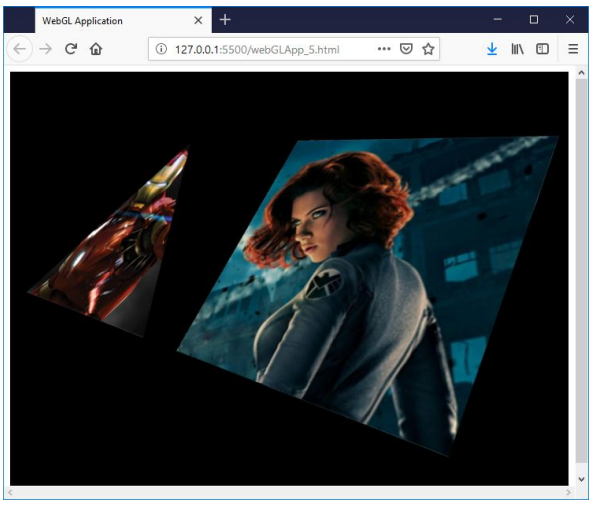

Figure 1. At the end of Exercise 4 you could inspect two objects interactively by manipulating the camera transformations with the mouse.

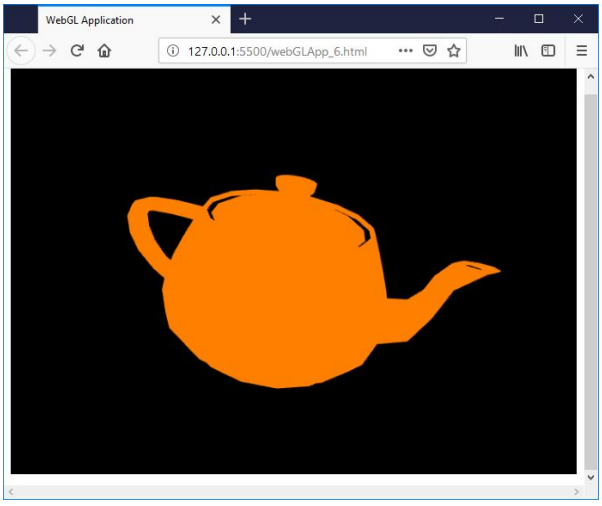

Figure 2. Result after loading the teapot model (teapot.json) and showing it with a plain orange color.

Exercise 5-b: Loading several models

Extend *Exercise 5-a* to load two models on the GPU and render both at the same time as shown in *Figure 3*.

- Add the necessary code to load the second model.
    - Can we reuse the loading function implemented in Exercise 5-a? Research how you could do it.
- Pass the object color as a uniform variable to the shader. You will have to modify the shader itself, and add the necessary code into *initShaders* and *drawScene*.
    - Where will you store this color property? Do you have an entity type of object?

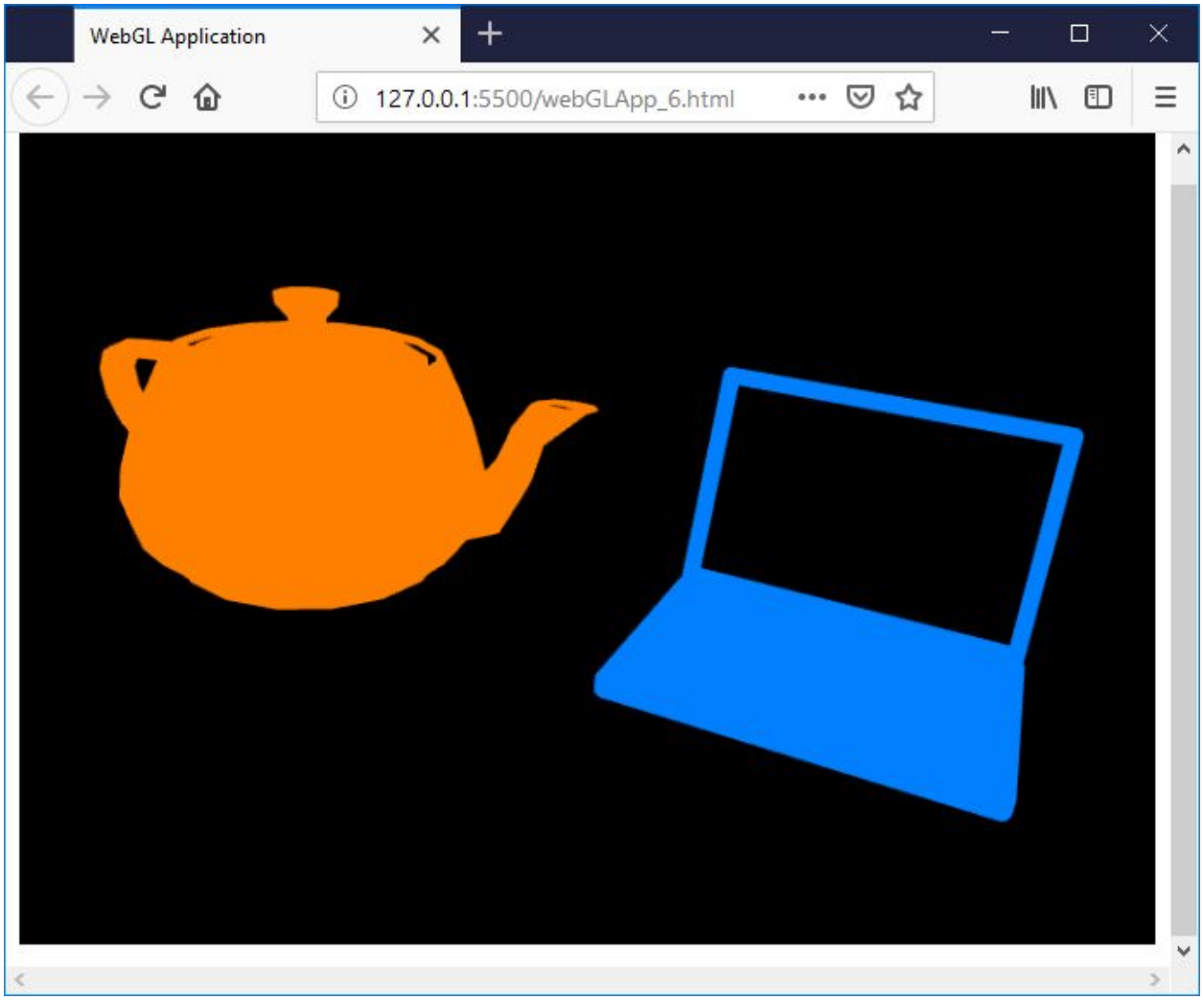

Figure 3. Both the teapot and the laptop models have been loaded on the GPU and are rendered at the same time.