Winter 2020: CSI4130
Assignment 1

Due: Friday, March 6th, 2020, 11:00pm in Virtual Campus

University of Ottawa - Université d'Ottawa

Jochen Lang

1 Viewing and Multiple Viewports [10 in total]

In this assignment, you will build an interactive animation of a scene shown from two viewpoints in two viewports. The basis of this assignment is *Laboraory 5 - Object Loading and Viewing*. As in this laboratory, you will need to load an object file with materials and texture and display the object. This part will be largely the same as in the laboratory but see Question 1.3 for improvements. The laboratory showed a rendering from the camera but in this assignment you will need to show two views: one from the camera as usual and an orthographic view of the scene, camera and viewing volume. A similiar OpenGL 1.1 application can be found in Nate Robins OpenGL tutors.

This is an individual assignment. You are not allowed to use any other library except `Three.js` and `dat.gui.js`.

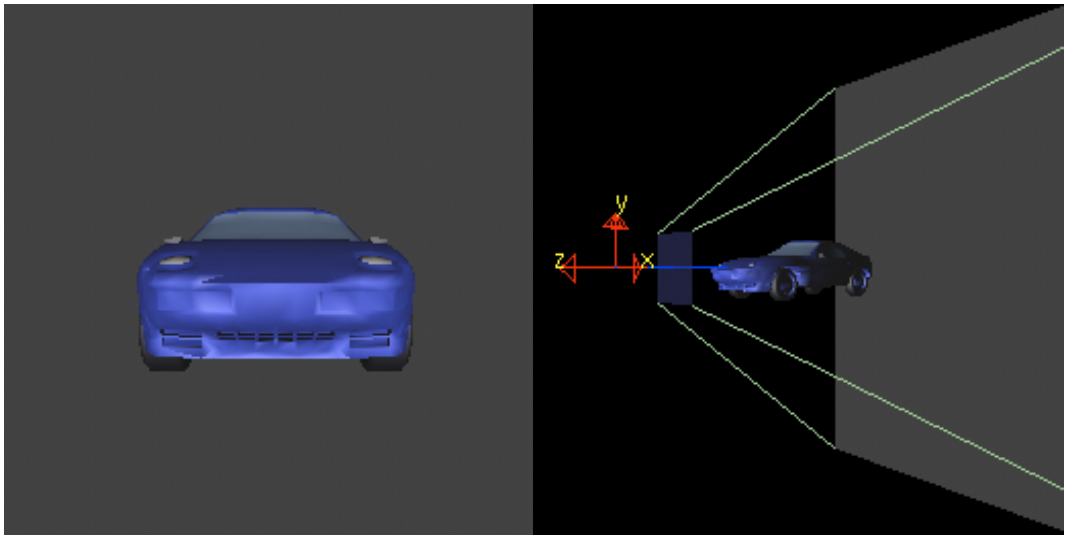

Figure 1: Sketch of Application based on Nate Robins Open GL Tutorials (Source code and model available at https://user.xmission.com/ñate/tutors.html)

1.1 Two Viewports [4]

Starting with the lab, add a second viewport to create a world view in addition to the camera view. Your world view needs to be always an orthographic view showing the camera, the loaded object

and all of the viewing volume. Note for this part of the assignment, you will not need to update your world view. You will find an example for using two camera views for two different viewports in the `THREE` documentation in the camera example `https://github.com/mrdoob/three.js/blob/master/example`
Note: Please do not add the animation changing the viewing from the example.

1.2 Viewing Volume [3]

Add lines marking the viewing volume in the world view. Your lines must update based on the parameters set with the dat.gui interface. Make sure to change the world view such that it always shows the complete viewing volume. The scene view will have to remain an orthographic view. Keep the a tight margin around the viewing volume (i.e., your scene view must adjust). Let the user rotate the view around the center of the scene and vary the radius large enough such that the viewing volume, scene and camera always remain in view.

1.3 Material and Texture Loading [3]

Currently, the object is rendered with the default material and a hard-coded texture map. This will obviously only work for a specific object file. Instead wavefront `obj` specify these parameters in a material file (extension `*.mtl`). This is a text file, see `tiger.mtl`. `THREE` has a built-in material laoder, use it to properly load different textured and non-textured obj files. I provide three example obj files with this assignment. Your code has to work with these three files. Your code does not need to handle arbitrary `obj` files containing more than one object and/or many material files.

1.4 Bonus: User Interface [2]

Give the user the option to drag the borders of the viewing volume to interactively set view parameters.

2 Submission

Your assignment submission must consist of your Javascript and html file. As you are working with the current version of `Three.js` and `dat.gui.js`, you will not submit these.

Filename
sceneViewer.js
sceneViewer.html

You must submit the files listed above and no library files via Virtual Campus.