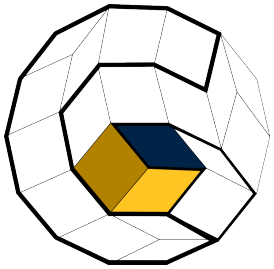
TU BRAUNSCHWEIG
Dr.-Ing. Martin Eisemann
Institut für Computergraphik
Matthias Überheide

January 5, 2015



Real-time Computer Graphics WS 14/15
Assignment 7

Present your solution to this exercise on Monday, January 12th, 2015.

The exercises are going to take place in the CIP pool, room G40 in Mühlenpfordstraße 23. Please make sure your solutions compile and run on the CIP pool computers. Note that you need a y-number, which can be obtained at the Gauß-IT-Zentrum, to use the computers. If for some reason you are not able to attend the exercise, you may send your solution to ueberheide@cg.tu-bs.de instead.

In this assignment you will setup your OpenGL and shader programs for texture usage. You will then be able to load OBJ meshes including texture coordinates and render textured models in the later tasks of this exercise.

7.1 Extending the OBJ Loader (20 Points)

Complete the ObjectLoader class. When the mesh information has been imported properly and the information is passed to the `MeshObj`, you will have to create a `VBO` for the texture coordinates, upload the values to the GPU and enable the proper vertex attribute for texture coordinates in the `VAO` of your `MeshObj`.

7.2 Loading Textures (20 Points)

Implement the loading process of the texture bitmap file in the function `loadTextureData(...)`. You can use the 'Open Computer Vision' (OpenCV) library for this task, which is installed on the computers in the CIP pool already. You can also use other smaller libraries (e.g. FreeImage) or a self-implemented function. Note that you have to adapt the CMakeLists to include the necessary folders or link any necessary libraries.

7.3 Using Textures (20 Points)

To texture a rendered object properly, you need to pass the texture itself to your shader program and set texture coordinates as vertex attributes. In the first task you've already setup everything to import texture coordinates and pass them as vertex attributes in form of a VBO to your GPU.
Now you need to create a texture structure, that you can upload to your graphics card and adapt your shader to retrieve the texel information for each rendered fragment.

7.4 Render textured meshes (20 Points)

Load, position and render the attached meshes `trashbin` and `ball` in a single scene (Fig. 1). Try to keep the texture handling nice and simple and avoid redundant code.

7.5 Autobots and Decepticons ! (30 Points)

The meshes in the previous task have been rather simple. More complex meshes are often assembled by more than one sub-meshes or use multiple textures. Your task is to load the models of Megatron and Optimus Prime and render both together in an adequate pose (e.g. Fig. 2). Find a way to render the respective vertices with the corresponding textures as defined in the mesh and material files (`*.obj`, `*.mtl`). You can hard-code the loading and rendering method for both models for this task, but it is nicer to parse the material file in the OBJ loader to keep the main file short and simple. Estimate a nice lighting situation, camera view and and pose for both models.

7.6 Improving the background (10 Points)

For model exposition it is more aesthetic to have some kind of gradient in the background. Find an efficient way to render a dark-gray to light-gray gradient as in the figure 3.

Figure 1: Textured trash bin and soccer ball shaded by three light sources

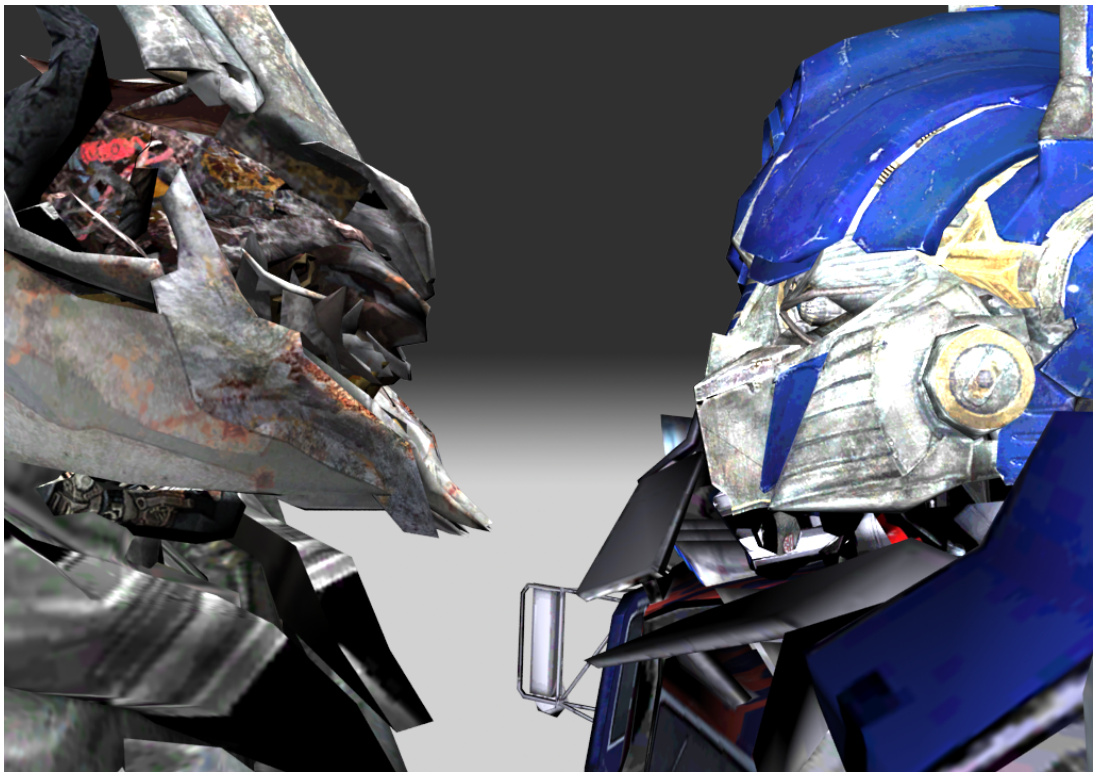


Figure 2: Megatron vs. Optimus Prime (models from http://thefree3dmodels.com/)

Figure 3: Megatron and Optimus Prime rendered with a gradient background