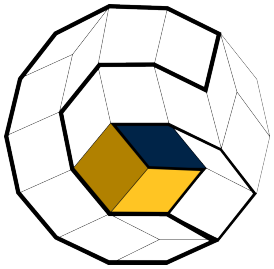
TU BRAUNSCHWEIG
Dipl. Inform. Martin Eisemann
Institut für Computergraphik
Lorenz Rogge (rogge@cg.tu-bs.de)

April 12, 2011



Echtzeit Computergraphik SS 2011
Assignment 1

Present your solution to this exercise on Monday, April 18th, 2011.

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 rogge@cg.tu-bs.de instead.

In this first assignment you will be introduced to the build system CMake and programming in C++ in general.
For additional information refer to `http://www.cmake.org/` or `http://www.cplusplus.com/`.

1.1 Create CMake Project to compile sources (20 Punkte)

You have been given a code framework which needs to be completed and compiled to run on the CIP pool computers (room G40). While keeping the given folder structure within `ex01`, complete the following files:

- `ex01/CMakeLists.txt`
- `ex01/src/CMakeLists.txt`

First edit the file `CMakeLists.txt` and give the build project a proper name, e.g. *Exercise01*. This name will be used throughout the configuration files of CMake and allows to use macros to refer to special locations. For example use `${Exercise01_SOURCE_DIR}` to refer to the source root path of the project *Exercise01* or use `${Exercise01_BINARY_DIR}` as the binary root path. Further configure the following:

- The build path should be the folder `ex01/` itself, so that the compiled executable can be run from that directory.
- Setup the path to search for included files to the subfolder `ex01/include/`.
- Also setup the path to search for existing libraries to the subfolder `ex01/lib/`.
- Tell CMake to process the subdirectory `ex01/src/` containing files to compile.

Having this done, edit the file `ex01/src/CMakeLists.txt`:

- Configure CMake to build an executable called `ex01` from the files `Ex01.cpp` and `ObjLoader.cpp` which are located in `ex01/src/`.
- To compile correctly, tell CMake to link against the library `openGLTools`, which can be found as `libopenGLTools.a` in `ex01/lib/`. Also tell the linker to use `${OpenGL_LIBRARIES}`, `${GLUT_LIBRARIES}`, and `${GLEW_LIBRARIES}`, which are needed for displaying a window, rendering geometry and so on. These environment variables are already provided by the CMake system. You just need to use them.

After the CMake setup create a make file using the command `"cmake ."` from the root folder. CMake will now process the definitions in all `CMakeLists.txt` files and setup an appropriate makefile. Now you can build your project by calling `"make"`. Executing `ex01` should result in a 512×512 black window titled *Exercise 01*. Close it using *ESC* or click the *X* on top.

1.2 Load a 3D mesh from file (20 Punkte)

Implement the missing method `loadObjFile(...)` defined in `ObjLoader.h`. The task of this method is, to open a Wavefront OBJ file (`http://en.wikipedia.org/wiki/Wavefront_.obj_file`) containing vertex data for a 3D mesh object and to import data into a vertex list and a list of vertex indices used by faces. These lists are used to render the imported object as a *VertexArrayObject*. For now, you don't have to implement the rendering itself. Just use the method `renderVertexArray(...)` provided by the library `openGLTools`. To implement a proper OBJ-Parser, do as follows:

- Clear the passed lists `vertexList` and `indexList` of eventually existing content.
- Access the file given by the parameter `fileName` and read it in line by line.
- Scan each line for the key defining the contents of the line. Until now, only the keys `"v"` and `"f"`, indicating a vertex resp. a face definition, are of interest.
- For each vertex definition, read in the three coordinates as `float` and append a `Vertex` object using these coordinates to the `vertexList`.
- For each face definition, read in the vertex indices used by the face and append them to `indexList`. Be sure to use correct indices, since `renderVertexArray(...)` starts indexing at 0.

Now modify `Ex01.cpp` to load the file `HelloOpenGL.obj` using your just implemented method and call `renderVertexArray(...)` to render the object to screen.
After correct implementation you may compile your project from task 1.1 again. Now, running `ex01` from your exercise folder should result in an image like this:

Figure 1: This is how your screen should look like after task 1.2