Brown University ENGN2912B Fall 2017

Scientific Computing in C++

Assignment 9 | The IfsViewer Application

This assignment is based on the notes for Lecture 14.

The homework assignment for this week is the IfsViewer described above. Download the file 09_Gabriel_Taubin.zip and expand it. You should have a directory named 09_Gabriel_Taubin containing four subdirectories: assets, bin, build, and src. The assets directory contains an Apple icon file IfsViewr.icns, and an image file named IfsViewer.png . The bin and build directories are empty. The src directory contains a CMakeLists.txt file, and four subdirectories: ifs, test, util, and viewer. Each of the four subdirectories contains its own CMakeLists.txt file, as well as class header and implementation files. The files contained in the directories ifs and util will be compiled into libraries. The ifstest.cpp file contained in the test directory will be compiled and linked with the two libraries into a command line application. The files contained in the viewer directory will be compiled and linked with the two libraries into an interactive Qt application named IfsViewer. Pay special attention at the structure of the CMakeLists.txt files, since this is the first example that we see of a complex build structure with subdirectories, libraries, command line, and interactive applications.

The main components of this assignment are the `Ifs` class, which is just a container class without major complications, the `IfsWrlSaver` class which extends code that you have written in previous assignments, and the `IfsWrlLoader` which is significantly more complex. You should implement this last class based on the `Tokenizer` approach described above. Your parser should be able to parse a complete `IndexedfaceSet` node, including texture coordinates, even though the image used for texturing such `IndexedfaceSet` node is not specified within the `IndexedfaceSet` node itself. We will come back to this problem in a subsequent assignment.

To help debug your `Ifs`, `IfsWrlLoader`, and `IfsSaver` classes, you will implement a command line application named `ifstest` which will take the following switches, followed by an input file name, and an output file name, as command line arguments. While the input and output file names are mandatory, all the other switches are optional. When run with the "-u" switch, the application should print the usage message and quit, independently of any other command line parameters. If a command line syntax error is detected, the application should print an error message, explaining where in the command line was the error detected, followed by the usage message, and quit. If the "`-d`" switch is specified the application should print messages as it progress through the loading, processing, and saving phases. If the "`-d`" switch is not specified the application should run without printing any messages at all. If the "`-cn`" switch is specified, the application should clear the `normal`, and `normalIndex` fields of the loaded `Ifs`, and set the `normalPerVertex` field to `true`, before saving it. If the "`-cc`" switch is specified, the application should clear the `color`, and `colorIndex` fields of the loaded `Ifs`, and set the `colorPerVertex` field to `true`, before saving it. And if the "`-ct`" switch is specified, the application should clear the `texCoord`, and `texCoordIndex` fields of the loaded `Ifs`, before saving it.

```
void usage() {
  cerr << "USAGE | ifstest" <<endl;
  cerr << "          [-d|-debug          ]" << endl;
  cerr << "          [-u|-usage          ]" << endl;
  cerr << “          [-cn|-clearNormal   ]” << endl;
  cerr << “          [-cc|-clearColor    ]” << endl;
  cerr << “          [-ct|-clearTexCoord ]” << endl;
  cerr << “          inputFile.wrl        ” << endl;
  cerr << “          outputFile.wrl       ” << endl;
}
```

After you finish debugging your code within the command line application, you can move on to compiling the `IfsViewer` application, which should be straightforward. Once you manage to get this application to work, you should modify it to add command line processing. As a minimum you should be able to read the input file name from the command line and load the file directly at start-up.

How to submit your work

Follow the same guidelines as for previous assignments.