Graphics Programming II – Assignment #2 (BRDFs)
Due on 05/06/2009

In this assignment you will implement four BRDFs. Each BRDF is will be used on the same model under identical lighting conditions. Draw a grid of objects. So that each object will be drawn with the same viewing and lighting conditions, partition the display using `glViewport`. Each BRDF will be represented by a separate row in the grid. In the columns of the grid, parameters of each BRDF will be modified. For example, if the Cook-Torrance BRDF is used, the columns would show different values of m. Figures 5.5 and 5.6 at `http://wiki.gamedev.net/index.php/D3DBook:(Lighting)_Cook-Torrance` show examples of what I mean. Naturally, each object should rotate around its center, and the light should orbit the object.

Since there will be so many objects on the screen, you may want to create a full-screen window. It is safe to assume that 1280x1024 is the minimum available window size. However, it is possible to query SDL for the set of available window sizes (i.e., screen modes).

The BRDFs you choose must meet the following criteria:

- At least one BRDF must have a Fresnel term.
- At least one BRDF must have anisotropic reflection.
- At least one BRDF that represents metals (hint: it will *not* include a Fresnel term.)
- All BRDFs must use normal mapping.

Phong or Blinn lighting, even recast as a BRDF, will not count as one of the four BRDFs. However, including a row of objects using Phong or Blinn lighting may be a useful reference point.

There are several ways to implement the shaders for this assignment. The method chosen will have different performance and implenetation difficutly trade-offs. In addition to your code, you must submit a short descirption of the choice you have made. You must also defend your choice. Your write-up should be on the order of half a page to a page. Please include the write-up as either MS Word `.doc` (*not* `.docx`!), OpenOffice `.odt`, plain ASCII text, or PDF.

Listing the trade-offs and the relative merits of those trade-offs is a good way to defend your choice. There are some trade-offs that you may have to guess about (i.e., the relative performance of doing one operation multiple times per-frame versus doing a different operation per-fragment). Explicitly state these cases in your write-up.

Two obvious methods (there are others) are:

- Implement each BRDF as a separate fragment program. All BRDFs should be able to share the same vertex program. Link the four separate programs into four separate shaders. During each frame, make a shader active and draw a single object, make the next shader active and draw the next object, etc.
- Make a single "super shader" that implements all BRDFs. The output generated by the shader is selected by the setting of one or more uniforms.

Criteria	Excellent	Good	Satisfactory	Unacceptable
Completion	Program correctly implements all required elements in a manner that is readily apparent when the program is executed. User interface is complete and responsive to input. Program documents user interface functionality.	Program implements all required elements, but some elements may not function correctly. User interface is complete and responsive to input.	Program implements most required elements. Some of the implemented elements may not function correctly. User interface is complete and responsive to input.	Many required elements are missing. User interface is incomplete or is not responsive to input.
Correctness	Program executes without errors. Program handles all special cases. Program contains error checking code.	Program executes without errors. Program handles most special cases.	Program executes without errors. Program handles some special cases.	Program does not execute due to errors. Little or no error checking code included.
Efficiency	Program uses solution that is easy to understand and maintain. Programmer has analysed many alternate solutions and has chosen the most efficient. Programmer has included the reasons for the solution chosen.	Program uses an efficient and easy to follow solution (i.e., no confusing tricks). Programmer has considered alternate solution and has chosen the most efficient.	Program uses a logical solution that is easy to follow, but it is not the most efficient. Programmer has considered alternate solutions.	Program uses a difficult and inefficient solution. Programmer has not considered alternate solutions.
Presentation & Organization	Program code is formatted in a consistent manner. Variables, functions, and data structures are named in a logical, consistent manner. Use of white space improves code readability.	Program code is formatted in mostly consistent with occasional inconsistencies. Variables, functions, and data structures are named in a logical, mostly consistent manner. Use of white space neither helps or hurts code reability.	Program code is formatted with multiple styles. Variables, functions, and data structures are named in a logical but inconsistent manner. Use of white space neither helps or hurts code reability.	Program code is formatted in an inconsistent manner. Variables, functions, and data structures are poorly named. Use of white space hurts code reability.
Documentation	Code clearly and effectively documented including descriptions of all global variables and all non-obvious local variables. The specific purpose of each data type is noted. The specific purpose of each function is noted, as are the input requirements and output results.	Code documented including descriptions of most global variables and most non-obvious local variables. The specific purpose of each data type is noted. The specific purpose of each function is noted, as are the input requirements and output results.	Code documented including descriptions of the most important global variables and the most important local variables. The specific purpose of each data type is noted. The specific purpose of each function is noted.	No useful documentation exists.

This rubric is based loosely on the "Rubric for the Assessment of Computer Programming" used by Queens University (http://educ.queensu.ca/ compsci/assessment/Bauman.html).