Graphics Programming II – Assignment #3
Due on 3/21/2012 (at the final)

For this assignment, you will have a choice: BRDFs or post-processing. Pick *one*.

1 BRDFs

Implement an anisotropic BRDF

- Render a scene with at least four objects and a moving camera. The sphere scene from the previous assignments is acceptable.
- Apply the Cook-Torrance BRDF and the new anisotropic BRDF to at least two objects.
- Each object in the scene must have different BRDF control parameters. Control parameters include things such as the m parameter to the Cook-Torrance BRDF or the α_x and α_y parameters to the Ward BRDF. Surface diffuse or specular color do *not* count.

Implementing the Ward anisotropic BRDF from the Cook-Torrance BRDF is a good choice. Much of the Cook-Torrance shader code can be reused for the Ward BRDF.

Extra credit will be given for implementing a BRDF that was not covered in class. The Oren-Nayar BRDF is a good choice for diffuse objects. Please let me know what extra credit BRDF you plan to implement *before* you start. There are a few BRDFs that we did not cover in class that are too similar to ones that we did cover to qualify.

2 Post-processing

Implement a full-screen post-processing effect on the scene from assignment #2.

- Render the scene as normal, then copy it to a texture.
- Implement a shader that will perform the desired post-processing effect.
- Using the texture containing the rendered scene, apply the post-processing effect and draw to the window.

Implementing a simple box filter or Gaussian blur is a good choice. Extra credit will be given for implementing either filter as two $O(n)$ passes instead of a single $O(n^2)$ pass.

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).