

CG Programming II – Assignment #3 (BRDFs)

Due on 27-February-2013 by the end of class

In this assignment you will enhance the lighting algorithm implemented in the previous assignment. Using the previous assignment (tangent-space normal mapping) as a starting point, you will

- Implement Schlick's approximation to the Fresnell function. Use this function to generate (gray scale) colors for the rendered object.
- Implement the modified Cook-Torrance lighting algorithm using the Beckmann distribution instead of the weighted Gaussian distribution from the paper.
- Implement the isotropic Ward lighting model. Your C++ should implement a simple user interface to switch between the two lighting models *at run time*. There are several ways to accomplish this with different trade-offs.
- As a separate text document, describe your implementation of switching lighting models and explain the trade-offs of that choice and at least one other possible implementation.
- Implement the anisotropic Ward lighting model. Implement a simple user interface to adjust the α_x , α_y , and rotation of the tangent frame. You only need to support switching between the anisotropic Ward model and the modified Cook-Torrance model. The isotropic Ward model is just a stepping stone to the anisotropic model.

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>).