CG Programming III – Assignmnet #1 (planar projected soft shadows) Due on 04/08/2008

Using the supplied skeleton code you will implement the following:

- Implement a ground plane class derrived from `model`.
- Implement a simple shader to draw a ground plane using the new class. To give the impression of depth, add some sort of simple fog effect.
- Draw *multiple* animated objects in the scene along with the ground plane. These can either be simple solids (e.g., the torus or sphere) or models loaded from disk.
- Implement a light source class also derrived from `model`. This class should store the position, direction, size, and field-of-view of the light. Use this information to draw some representation of the light source.
  - Add functions to set and query these values.
- Implement simple, planar projected shadows using the equations from the lecture notes.
  - Create a function called `ProjectToPlaneThroughPoint` that takes a point and a plane equation (n and d) as parameters. This function will create the projection matrix and multiply it with the top of the current stack (i.e., it will will `glMultMatrixf`). Isolating this code in its own function will make the next step much easier.
- Once hard shadows are working correctly, implement simple soft shadows. Use either Gooch's method or Heckbert and Herf's method.
  - The fundamental challenge in *either* method will be to generate different alpha values in the umbra and penumbra areas.

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).