CG Programming I – Assignment #3 (texture world)
Due on 11/13/2007

For this assignment, you will implement a simple 3D world. The world will consist of a ground plane and several objects scattered around the plane. The user will be able to move around the ground plane.

- Draw a ground plane.
  - The ground plane should be at least 100x100 units, assuming that the typical object is 1x1x1 units.
  - The ground plane should be textured.
  - Correct lighting of the ground plane requires that it be made from numerous smaller polygons in a grid pattern. Either `GL_QUAD_STRIP` or `GL_TRIANGLE_STRIP` must be used for this purpose.
- Draw multiple objects.
  - Each object should be animated, but may remain generally stationary (e.g., the object rotates around its center, the object moves up and down, etc.).
  - The objects must be 3D dimensional.
  - The objects must be lit by at least one light source.
  - The objects must be colored.
- The user can navigate the world.
  - Directional movement (i.e., forward, backward, strafe left, strafe right) and rotational movement (i.e., turn left and turn right) must be implemented. Using W-A-S-D for movement and Q-E for turning is recommended, but not required.
  - The user's movement should be constrained to the region of the ground plane.
- Light objects in the world.
  - At least one light source, which follows the user (i.e., is located at some constant position in eye-space) must be implemented.
  - If additional lights are implemented, they should be visually represented.

The following inputs must be implemented. In addition, the program must, in some way, communicate to the user how to use it.

- Escape must terminate the program.
- Inputs must be implemented for directional movement.
- Inputs must be implemented for rotational movement.

Code will be provided for loading texture files.

Objects in the environment may be platonic solids from the previous assignment or other objects of your choosing.

Collision detection (i.e., preventing the user from moving through objects) need not be implemented. In future assignments where this is required, additional code will be provided.

Extra credit will be given for implementing shadows projected on the ground plane.

Extra credit will also be given for implementing a level-of-detail effect for the ground plane. The ground plane is made from numerous small polygons. There should be more polygons close to the viewer, and fewer (larger) polygons farther away. Simplicity is the key here. To show that LOD is being used, there should be way to toggle normal rendering of the ground plane and wire-frame rendering of the ground plane.

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).