CG Programming I – Assignmnet #1 (moving objects w/stencil)
Due on 10/16/2007

For this assignment, you will need to implement a simple scene containing multiple flat objectes facing the camera. Each object must move and rotate. In addition, the objects must be partially "obscured" using an animated stencil.

- Draw multiple animated objects.
  - The animation must run at a rate independent of the rendering frame rate.
- Obscure portions of the scene using the stencil buffer.

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.
- An input must be implemented to pause the animation.

I recommend implementing the project in several discrete steps.

- Draw a single static object. This should look pretty much like the sample program from last week.
- Draw additional static objects.
- Create an array of data structures to track the position / orientation of each object. Update the program to draw the objects based on the data stored in these structures.
- Animate the objects by updating the data stored in the structures.
- Draw an additional object. This is the object that will eventually become the stencil. Be sure to draw this object first.
- Update the program to create a stencil buffer for the window, and clear the stencil buffer when the color buffer is cleared.
- Enable stencil updates when drawing the stencil object.
- Enable stencil test when drawing the remaining objects.

Future assignments won't have as much hand-holding. :)

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).