CG Programming III – Term Project
Due on 6/11/2007 - day of the final exam

This term we have studied several advanced algorithms that enhance either shadow map or shadow volume based techniques. Up to this point you have not been required to implement any of these algorithms. For the term project you will need to enhance either the shadow mapping assignment or the shadow volume assignment to include one of the advanced algorithms listed below.

As usual, projects will be presented to the class on the day of the final exam. Be prepared to describe how you implemented the selected algorithm. Also be prepared to explain and defend your design choices.

In *addition to* the requirements for assigment #3 (shadow maps) you must implement at least one of the following:

- Soft shadows using the algorithm described in the paper "Percentage-Closer Soft Shadows" by Randima Fernando.
  - The paper is fairly vague on several implementation details. If you choose to implement this algorithm, we can discuss some strategies on Monday 6/4.
- Dual-paraboloid shadow maps as described in "Shadow Mapping for Hemispherical and Omnidirectional Light Sources" by Stefan Brabec, et. al.
  - For this option, there should be several shadow casters and receivers surrounding the light source. In addition, the objects and light should be placed in side a "room" of some sort. Figure 3 and 4 at the end of the paper show some examples.
- Implement the light frustum optimizations described in "Practical Shadow Mapping" by Stefan Brabec, et. al.
  - There are two changes from the paper should be implemented. First, the texture shader in section 3 should be implemented using a fragment shader. The complex texture mechanism should be implemented using shader calculations. Second, the light frustum should be limited to the view frustum (as seen on the right of figure 3) and NV_depth_clamp should be used.
  - Scene rendered must include objects that are *outside* the view frustum and inside the light frustum. The light must also be outside the view frustum.

If shadow maps do not suit you, one of the following may be implemented in *addition to* the requirements for assignment #4 (shadow volumes).

- Implement the *complete* version of ZP+.
  - The implementation will include the crack avoidance algorithm described in section 5 of the ZP+ paper.
  - The scene must include at least one light that shows the failings of the Z-pass algorithm (e.g., the light and an occluder are positioned such that the camera is inside a shadow volume).
- Implement the "simple" version of ZP+ described in assignment #4.
  - The scene must include multiple *spot lights.*
  - It must be possible to toggle the display of the cone-like shape that defines the spot light. Part of your project presentation must include a description of how the spot light clamping is implemented.
  - The scene must include at least one light that shows the failings of the Z-pass algorithm (e.g., the light and an occluder are positioned such that the camera is inside a shadow volume).
- Implement shadow volumes using the alpha buffer instead of the stencil buffer as described in "Shadow Volumes Revisited" by Stefan Roettger, et. al.

Criteria	**Excellent**	**Good**	**Satisfactory**	**Marginal**	**Unacceptable**
Code Function	Program correctly implements all required graphical elements in a manner that is readily apparent when the program is executed. Appropriate algorithms and data structures are used.	Program implements all required graphical elements, but the operation of some elements may not be obvious. Appropriate algorithms and data structures are used.	Program implements all required graphical elements in some fashion. Algorithms and data structures are used that perform the required function, but may be less than ideal.	Program implements most required graphical elements in some fashion.	Most or all of the required graphical elements are missing or do not function correctly.
Code Mechanics	Program code is formatted in a consistent manner, variables and data structures are named in a consistent, logical manner. Code is commented adequately.	Program code is mostly consistent, but contains some occasional inconsistencies.	Program code is readable. Some functions or code blocks show consistent formatting, but that formatting does not carry through the entire program.	Program code is not consistently formatted, but is still somewhat readable.	Program code is a mess and may be more suitable as an entry to the International Obfuscated C Coding Competition.
User Interface	The program is responsive to input. All required inputs are implemented, and the user is informed, by the program, what the inputs are. The program can be terminated by the user.	The program is responsive to input. All required inputs are implemented. Some of the inputs are documented by the program.	The program is unresponsive under some circumstances. All required inputs are implemented. Some of the inputs are documented by the program.	The program is unresponsive under some circumstances. Some required inputs are either not implemented or are not implemented correctly. Some inputs are documented by the program.	Many of the required inputs are either not implemented or are not implemented correctly. The program lacks documentation for the inputs.