CG Programming I – Assignment #4 (Slinky)

Due 21-November-2012

In this assignment, a skinned mesh vertex shader and forward kinematic solver. The provided base code implements a large portion, and the sections where code must be implemented are marked with `FINISHME` comments. This applies to both the C++ code and the vertex shader.

The vertex shader receives an array of transformation matrices as uniform inputs. In addition, each vertex receives indices of two matrices and a single weight as inputs. The specified weight value is the weight of the first transformation matrix. The weight of the second transformation matrix can be derived using $1 - w$.

I recommend implementing this is three main steps.

- All vertices use matrix 0 as the transformation. Modify the vertex shader to use one of the matrix indices specified as the transformation matrix.
- Currently the C++ specifies the same transformation for all of the transformation matrices sent to the vertex shader. Modify the C++ to generate a separate transformation for each matrix. Each matrix should rotate around a separate pivot point with the same rotation angle. This will be very similar to the cube arch from the previous assignment.
- Modify the vertex shader to both specified transformations with the specified weights. Average the matrices together to generate a new transformation matrix. Then use this new transformation matrix to transform the point.

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).