Graphics Programming II – Assignment #1
Due on 10/13/2010

In this assignment you will generate camera control in a simple scene. Initially, a set of positions that the camera must "hit" are supplied. In the first iteration, linearly interpolate among these position.

In the second iteration create a set of Bézier curves that also pass through these end points. Evaluate the curves through time to generate camera positions. To do this, you will have to implement the `bezier_curve::position` method.

In the third iteration generate a new set of Bézier curves from the original set. The new set of curves must be G^1 continuous. Evaluate the G^1 curves through time to generate camera positions.

In the fourth iteration generate a new set of Bézier curves from the original set. The new set of curves must be C^1 continuous. Evaluate the C^1 curves through time to generate camera positions.

In addition, modify the supplied shaders, `phong.vert` and `phong.frag`, to perform per-fragment lighting. Extra credit will be given if lighting is performed in surface-space. This will also give you a start on the next assignment.

There are three main programming elements to this exercise:

- Evaluation of positions on a Bézier curve.
- Algorithmic modification of piecewise Bézier curves to attain higher orders of continuity.
- Per-fragment lighting.

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).