Graphics Programming I – Assignment #1 (2D plasma)
Due on 10/14/2009

In this assignment, you will implement a simple 2D graphics effect using GLSL shaders. Using the fragment shader, implement a simple "plasma" effect. Some examples of this effect can be found on YouTube by searching for "plasma demo".

It is recommended that this assignment be implemented in three phases.

First, start with the sample code provided in class to open an window for OpenGL rendering. The program should terminate when the escape is pressed. The program should also use a time to control the rate of rendering.

Second, implement set of utility routines to compile a shader, generate a program from a pair of shaders, and create an empty buffer object. Use the routines to load a simple vertex and fragment shader. Use these shaders to render a simple triangle strip in the shape of a rectangle. The "GLSL Hello World" tutorial at `http://people.freedesktop.org/~idr/OpenGL_tutorials/` will be very helpful for this stage.

Finally, modify the simple fragment shader to implement a plasma type effect. The fragment shader built-in `gl_FragCoord` gives the X, Y, and Z position at each fragment. Scaling this value and using as input to the `sin` function can produce interesting results. In addition, your program can provide an angle offset uniform that can be used to change the input to the `sin` function. Play around and come up with something interesting.

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).