CS334: Assignment #3 – GPU it!

Out: February 25, 2021
Back/Due: March 16, 2021 (2.5 weeks)

Objective:
This objective of this assignment is to obtain a good understanding of modern GPU programming. To this end, you will implement a water simulator. There are many ways to simulate water but your implementation should be at least somewhat physically based and use vertex/fragment shaders to implement the phenomena. Your program will make use of OpenGL but most of the work will be GLSL programming.

We will provide you with a working template in the GLUT/GLUI environment. It implements reaction diffusion – you can change it to one of the below.

Algorithm:
There are multiple ways to simulate water. Below are three options:
  a. Sine wave summation: add a few sine waves of different frequencies/phases/directions (e.g., 4 waves); instead of “changing the height” you can change the pixel color from less to more intense.
  b. Gerstner waves: use this specific form of summing waves that in addition to computing a height/color, also moves the xy coordinates. Thus, you should consider implementing this as a vertex program (and not a fragment program) that processes a dense triangular mesh.
  c. Wave equation (PDE): instead of reaction-diffusion use the space-time varying wave equation; in addition you will need to have to pixel values “reflect” off the boundary of the image rather than simply dissipate beyond the edges.

Choose from amongst these or you can create a similar one but it should fall into one of the above categories. Please indicate which option you used in the code in a very clear manner.

Specifics:
(0) (10%) Setup. Define an initial condition and a GUI to control simulation parameters.
(1) (60%) Water algorithm. Implement the water algorithm, mostly GLSL code though it might need some communication with the CPU and GUI. The minimum is defining the parameters at the beginning of the execution and then pressing a “start” button.
(2) (20%) Visualization. Produce a meaningful and visually interesting visual result looking like water. The minimum is a top-down 2D view of the simulation. However, you will need to compute surface normals in any solution.
(3) (10%) Islands. Add at random, or just chosen locations, 1 to 3 small islands in the middle of the image/terrain. Water should not enter or appear on top of these islands and should “bounce off” the shoreline.
(4) Extra Credit (up to 10%)
  a. (2%) Extend so that the parameters can be changed on the fly using a GUI
  b. (5%) Extend “algorithm C” so that the result is a heightfield simulating water – you will need to move the camera to a decent bird’s eye rendering so that the heightfield can be appreciated.
  c. (3%) Extend in some other impressive way
Grading:
Your program will be tested against the aforementioned functionality and your code will be inspected – PLEASE DO NOT COPY CODE FROM ANY SOURCE!

NOTE: you must turn-in a precompiled executable of your program, but no debug output!

If you implement the extra credit, please ensure instructions are given on how to use it --- put such in the GUI or have clear instructions printout – do not assume we will read your code and decipher how to use your extra credit.