ECSE 420 – Parallel Computing – Fall 2019

Lab 3 – CUDA Musical Instrument Simulation

Due: December 1, 2019 at 11:59 pm

In this lab, you will write code for a musical instrument simulation, parallelize it using CUDA, and write a report summarizing your experimental results.

Finite Element Music Synthesis

In a "finite element" model, a complex physical object is modelled as a collection of simple objects (finite elements) that interact with each other and behave according to physical laws. It is possible to simulate electromagnetic fields, calculate stresses in the structure of a building, or synthesize the sounds of a musical instrument using finite elements. In this lab, you will synthesize drum sounds using a two-dimensional grid of finite elements.

The particular finite element method we will be using is similar to but slightly different from the ocean simulator we examined in the class. In this method, each interior element performs the following update at every iteration:

$$u(i,j) = \frac{\rho[u1(i-1,j) + u1(i+1,j) + u1(i,j-1) + u1(i,j+1) - 4u1(i,j)] + 2u1(i,j) - (1-\eta)u2(i,j)}{1+\eta}$$

$$1 \leq \mathrm{i} \leq \mathrm{N} - 2,\ 1 \leq \mathrm{j} \leq \mathrm{N} - 2$$

where u is the displacement of the element at position *(i, j)*, $u1$ is the displacement of that element at the previous time step, $u2$ is the displacement of that element at the previous time step, and η *(0.0002)* and ρ *(0.5)* are constants related to the size of the drum, sampling rate, damping, etc.

Then, we ensure that the boundary conditions are met by updating the side elements:

$$u(0,i) := Gu(1,i)$$

$$u(N-1,i) := Gu(N-2,i)$$

$$u(i,0) := Gu(i,1)$$

$$u(i,N-1) := Gu(i,N-2)$$

$$1 \leq i \leq N-2$$

and then the corner elements:

$$u(0,0) := Gu(1,0)$$

$$u(N-1,0) := Gu(N-2,0)$$

$$u(0,N-1) := Gu(0,N-2)$$

$$u(N-1,N-1) := Gu(N-1,N-2)$$

where *G (0.75)* is the boundary gain.

Finally, at the end of the iteration, we set *u2* equal to *u1* and *u1* equal to *u*.

To simulate a hit on the drum, simply add 1 to *u*1 at some position (*only once before the first iteration*). To record the sound of the drum, collect the value of *u* at some position for each iteration in an array of length *N*, where *T* is the number of iterations to perform. For this lab, you should use (*N/2, N/2*) as the position on the drum for both the hit and the recording.

1. Write code that implements a 4 by 4 finite element grid **sequentially**. The user will provide a single command line argument, *T*, which specifies the number of iterations to run the simulation. Your code should print *u* (*N/2, N/2)* (which in this case is u (2, 2)) to the terminal at each iteration.

2. Parallelize your implementation in **Part 1** by assigning **each node** of this grid to a single **thread** using CUDA. The user will provide a single command line argument, *T*, which specifies the number of iterations to run the simulation. Your code should print *u*(*N/2, N/2)* (which in this case is u(2, 2)) to the terminal at each iteration. (The output must match the output from Part 1, otherwise please debug).

3. Using one **thread** per finite element can require an enormous amount of communication and GPU overhead, and even in large clusters there may not be enough hardware available to fully parallelize the computation. It is usually better to use a decomposition that assigns multiple elements to each processor.

   Parallelize your code using such a decomposition (with rows, columns, blocks, etc.) and simulate a 512 by 512 grid. Try to parallelize with *different combinations* of **threads, blocks and finite elements per thread**. Ex – 1024 threads/block and 16 blocks can allow each thread to handle 16 finite elements (Total elements = 512x512 = 1024x16x16). In your report, discuss and analyze your parallelization schemes and experimental results.

To help debug your code, Figure 1 shows the entire u grid (4x4) for the first three iterations:

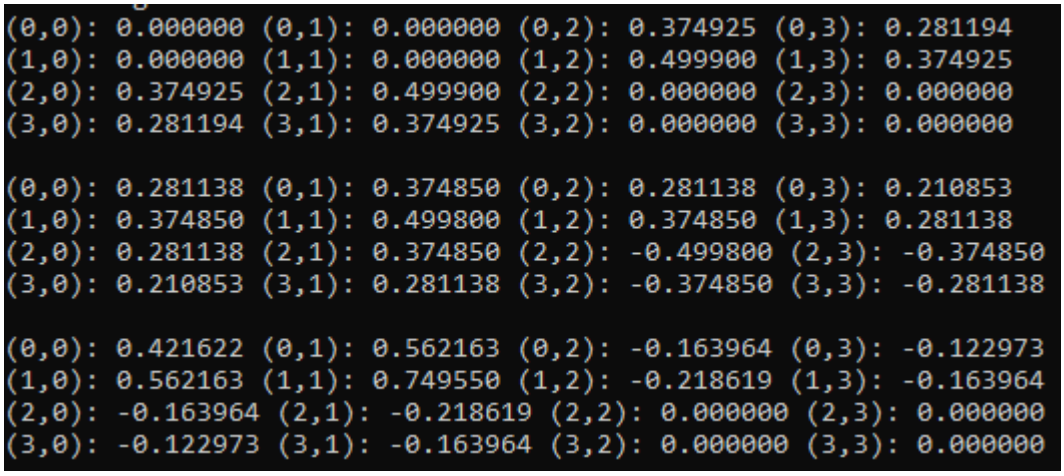

Figure 1: u for first three iterations

Submission Instructions:

NO LAB DEMO. There will not be any more demos for the labs. Please submit your entire solution (code) and the report in a zip file.

Each group should submit a single zip file with the filename Group <your group number> _Lab3.zip. (Ex – Group04_Lab3.zip).

Format for Report:

1. Must be a PDF only (No word document).
2. Must be named Group<your group number>_Lab3_Report. (Ex – Group03_Lab3_Report.pdf).
3. Must have a cover page.
4. Must follow the logical order for the lab discussions.
5. Have an appendix with **your own** code inside. Copy paste and merge the format so that the code alignment remains the same as in your IDE (MS Visual Studio).