LSU EE 7700-1 Homework 3 Due: 21 April 2010

To complete this assignment follow the setup instructions from the course Web page (if not already followed). The setup instructions bring you to the point where you can compile the cpu-only examples but for this assignment that won't be necessary. Then follow the instructions in the first problem to complete and test the setup.

Problem 0: Check out the balls project base code, a sample transcript appears below. Compile and run the code, and familiarize yourself with the following UI needed for this and follow-up assignments:

Press 't' to run a benchmark that drops thousands of balls on the platform with certain constants, such as friction set to special values. (Edit `benchmark_setup` to change the behavior.)

Press 'T' to run the benchmark with fewer balls, this is helpful when on a slower system or testing slower code.

Press 'a' to switch between CPU and GPU physics.

Press 'c' to toggle between natural colors and color based on CUDA block assignment. (See VAR 'Color by Bock in Pass'.)

Press 'q' to toggle the value of variable `World::opt_debug` between true and false. The current value is the first number after "Debug Options." The variable `World::opt_debug` currently does nothing, it can be used to test ideas for your solution.

Press 'Q' to toggle the value of variable `World::opt_debug2`. This also does nothing and is available for working on the assignment.

Timing for the scheduling pass is shown on the upper right (it may be necessary to make the window wider). Sched shows the total time for scheduling, SPart shows the time needed to find contact pairs. These timings are provided by `frame_timer.user_timer_X` calls, feel free to add your own calls to time other parts of the code.

```
# Transcript of commands to check out and run code for
# this assignment.

    [orion.ece.lsu.edu] % cd ~
    /home/faculty/koppel

# Check out a copy of the balls code, revision 240, into a directory named hw3.
    [orion.ece.lsu.edu] % svn -r 240 co https://svn.ece.lsu.edu/svn/gp/proj-base/balls
hw3
    A    hw3/balls-shdr.cc
    A    hw3/balls.png
# ...
    Checked out revision 240.

# Check out or update the include directory.

# Because the repository include directory has changed this step needs
# to be repeated unless the last checkout was very recent.
# (An easier way to update is to execute ‘‘svn update’’ in the include
# directory.)
    [orion.ece.lsu.edu] % svn -r 240 co https://svn.ece.lsu.edu/svn/gp/include
    A    include/gl-buffer.h
```

```
    A    include/misc.h
    A    include/texture-util.h
    A    include/util.h
    A    include/gp
    A    include/glextfuncs.h
    A    include/coord.h
    A    include/pstring.h
    A    include/shader.h
    A    include/cuda-util.h
    Checked out revision 236.


# Go to the balls directory (named hw3), compile, and run the code.
    [orion.ece.lsu.edu] % cd hw3
    /home/faculty/koppel/example/hw3
    [orion.ece.lsu.edu] % make -j 4
    /usr/local/cuda/bin/nvcc  -Xcompiler -Wall --ptxas-options=-v -I/usr/X11R6/include/
-I/opt/local/include -I../include -I../../include  -g -Xcompiler -Wno-strict-aliasing
-Xcompiler -Wno-parentheses -c balls-kernel.cu
# ...

    [orion.ece.lsu.edu] % balls
    S GL_VENDOR: "NVIDIA Corporation"
    S GL_RENDERER: "Quadro FX 3800/PCI/SSE2"
    S GL_VERSION: "3.2.0 NVIDIA 190.53"
```

Background:

The balls code used in class uses CUDA to compute ball and tile positions, but uses the CPU to prepare a work schedule for CUDA. The work schedule specifies which pairs of balls may be in contact, and because balls move it must be recomputed every few time steps (the default is every six time steps). CUDA kernel `pass_pairs` uses the schedule to determine which pairs of balls to resolve (test for contact and if so, apply contact forces).

The most time-consuming part of schedule computation is testing whether balls (called phys in the code) are nearby, taking roughly $O(n^{1\frac{2}{3}})$ time for n densely packed balls. This and other code is in routine `World::contact_pairs_find`. The first part of the code sorts the balls based on their maximum position on the z axis over some amount of time based on their size, shape, velocity. This sort presumably takes $O(n \log n)$ time. The proximity test occurs after that, in the `idx9` loop. If balls are in proximity a Contact structure for the pair is added to the list `contact_pairs`. Assuming a regular distribution of closely spaced balls the proximity test is performed about $n^{1\frac{2}{3}}$) times. (If the balls are more spread out the time complexity approaches $O(n)$.)

Problem 1: Modify `contact_pairs_find_gpu` and file `balls-kernel.cu` so that a proximity count for each ball is computed with CUDA. The CUDA code (in this first assignment of this series) should just count the number of lower-z balls a ball is in proximity to. (A lower-z ball is one with a lower z coordinate value, base this on the zsort that was already completed.) Count only ball/ball proximity, do not bother counting ball/tile proximity.

When `cuda_contact_pairs_find` starts CUDA has an up-to-date copy of the `cuda_balls` array. If necessary, your code should send any additional data needed by CUDA. Do this using

CUDA calls, not by using the `pCUDA_Memory` classes (but feel free to look at these for ideas).

After sending data to CUDA your code in routine `cuda_contact_pairs_find` should launch a CUDA kernel that determines the proximity count. It should then retrieve that data (using CUDA API calls) and assign the count to `Prop::proximity_cnt`.

The kernel itself should be placed in file `balls-kernel.cu` and common declarations placed in `balls.cuh`.

By default, the CUDA proximity code is turned off. Use 'F4' to toggle it on and off. An error count is displayed, this is the difference between the actual number of proximity pairs (excluding tiles) and the number counted by your code.

(*a*) Complete the code so that the error count is zero when there are no balls near any tiles (the staircase and wheel). The goal should be to choose a block size that maximizes performance. Issues such as data arrangement will be dealt with in future assignments.

(*b*) Provide several possible reasons why your code is not reaching its full potential.