LSU EE 4702-1   **Homework 5**   **Due: 30 November 2012**

Follow the Programming Homework Work Flow *instructions on the procedures page, substituting* hw5 *for* hw1*. Also, update the include directory.*
Use http://www.ece.lsu.edu/gp/refs/CUDA_C_Programming_Guide.pdf*, the NVIDIA CUDA Compute Unified Device Architecture Programming Guide, as a reference. The code in this assignment requires OpenGL 4.3 and CUDA, which should be installed on* snow*,* ice*, and* frost*.*

Problem 0: Follow the class procedures for homework but substitute hw05 for the assignment. Compile and run the code. It should display our familiar helix, but this time it's a simulated spring, and it should be drooping under gravity, see the illustration to the right. The physics can be performed by both CPU and GPU, by default it runs on the CPU, pressing 'a' will toggle it.
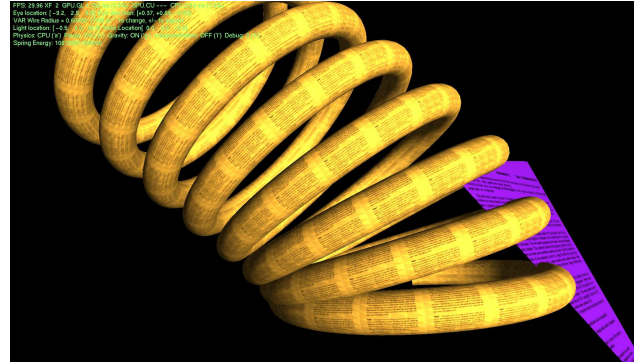
The CPU code can prevent interpenetration, this option is initially off, it can be toggled by pressing 'i'. The option was turned off when the image above was captured, notice that the bottom loops of the spring interpenetrate. The code for interpenetration can be found in `time_step_cpu`, it uses a brute-force approach, comparing every segment of the helix to every other, making it very slow. The goal of this assignment is **not** to use a clever broad-phase collision detection scheme to avoid $O(n^2)$ comparisons, but instead use the GPU to do the calculations faster. (In "real life" one might use both broad-scale collision detection and a GPU to do calculations.)

Problem 1: The routine `time_step_cpu` contains the CPU interpenetration code, clearly identified by comments. For this problem put the interpenetration code in CUDA kernel `time_step`, in file `hw5-cuda.cu`. The `time_step` kernel is launched in a configuration with one thread per helix segment. Each thread should test "its" segment against other segments. A correct solution will be faster than the CPU, but should still be slow.

Problem 2: By GPU parallelism standards, the number of threads launched for the `time_step` routine is not large. In the default configuration there are 160 segments, just five warps. That will underutilize any GPU.

For this problem, add a new kernel to perform the interpenetration test. The new kernel should be launched with more threads than segments, taking advantage of more parallelism. Modify the existing `time_step` kernel so that it does not modify `helix_position` (but it still updates `helix_velocity` and the other two state variables). The new kernel should compute forces based on interpenetration tests, use that to update velocity, and then update the position.

A good solution of this problem requires the use of shared memory, atomic operations, and synchronization. That will be covered next week.