LSU EE 4702-1 Homework 6 Due: 24 November 2014

Problem 0: Follow the class account instructions for obtaining the assignment, substitute hw06 where needed. This homework code is based on Homework 4/3, and the physics simulation is identical. Like those assignments it renders a spiral. The spiral can be rendered using two routines, `render_spiral2` and `render_spiral3`; the `m` key switches between them. Routine `render_spiral2` is based on the Homework 4 solution, and initially `render_spiral3` is identical to `render_spiral2`, it will be modified for this assignment.

Routine `render_spiral2`, rather than sending actual vertex coordinates to the GPU, sends indices. The x component of a vertex "coordinate" is the ball number (the value of `i` in `render_spiral1`), the y component is the relative position between ball `i-1` and ball `i`, ranging from 0 to `opt_segments-1`, and the z component indicates whether the vertex is on the inner or outer edge of the spiral. The code in vertex shader `vs_main` in file `hw06-shdr-triangles.cc` uses these values to compute the actual vertex coordinate and normal, and to compute the texture coordinates.

The level of detail used for the spiral is controlled by variable `opt_segments`. Its value can be modified by first pressing the `TAB` key until "Number of segments per spiral" appears, and then pressing `+` and `-` to adjust its value. Check out and compile the code, and make sure that it runs correctly.

For this assignment modify files `hw06.cc` and `hw06-shdr-lines.cc`. Solutions to the non-code questions can be provided in any of the following forms:

- As comments in one of the files above.
- As plain text in a separate file named `hw06.txt` in the `hw06` directory.
- In portable document format in a separate file named `hw06.pdf` in the `hw06` directory.
- Submitted on paper.
- E-mailed. If E-mailing, plain text and PDF are preferred.

Problem 1: The code in `render_spiral2` is based on the solution to Homework 4. As discussed in Homework 5 Problem 3, the vertex shader used for `render_spiral2` in Homework 4 wastes computation because it computes information both for inner and outer vertices but only sends one of these to the geometry shader. The problem suggests using line strips to solve that problem. The solution to Homework 5 presented some details on how this would be done, but it did not show any code. For this problem, modify `render_spiral3` and the shaders so that it avoids the waste.

For this problem you must modify code both in `render_spiral3` in file `hw06.cc` and the code in `hw06-shdr-lines.cc`, both will be collected by the TA-bot. Initially the code in `render_spiral3` is nearly identical to `render_spiral2` and `hw06-shdr-lines.cc`, except that it invokes shaders from file `hw06-shdr-lines.cc` (which you should modify). Routine `render_spiral2` invokes shader code from `hw06-shdr-triangles.cc`, which you should not modify.

Here are some things to watch out for:

- Be sure to adjust the maximum number of vertices specified for the geometry shader output. If this number is too low execution will end with an error.
- When you modify an interface block (such ad `Data_to_GS`) be sure to modify the other interface block with the same name (one is a shader output, the other is a shader input).

Here are some debugging tips:

- Check for shader code compilation errors when your program starts. The errors are sent to `stdout` and should appear in the shell or in gdb, depending on how you started the program.
- If execution ends with an OpenGL error, you can get a more detailed error message by turning on OpenGL debugging. To do this change `false` to `true` in `popengl_helper.ogl_debug_set(false);` near the end of `hw04.cc`.
- Three UI-controlled variables are available for debugging. They are `debug_bool.x`, `debug_bool.y`, and `debug_float`. The Booleans can be toggled with `d` and `D`. The float can be adjusted by pressing `Tab` until "Debug Float" appears and then press `+` and `-` to adjust the value.

Problem 2: Compare the performance of `render_spiral3` to `render_spiral2`. To see any difference at all the value of `opt_segments` will need to be increased, so do this if necessary.

(*a*) Show a table comparing the performance of the two methods versus `opt_segments`. In the table show CPU and GPU OpenGL time. Also indicate which GPU you are using.

(*b*) Explain how varying `opt_segments` changes the proportion of the work performed by the three shader stages (vertex, geometry, and fragment).

(*c*) In a solution to an exam problem, the computation of `sin(theta)` and `cos(theta)` was moved out of the vertex shader. Instead the pre-computed sine and cosine values would be read from a buffer object. Would this increase or decrease the relative benefit of `render_spiral3`?

Problem 3: When the Homework 6 code starts it prints information about the available GPUs, using a CUDA API. CUDA isn't used for anything other than printing this information. Here is some sample output:

```
GPU 0: Tesla K20c @ 0.71 GHz WITH 5119 MiB GLOBAL MEM
GPU 0: CC: 3.5  MP: 13  CC/MP: 192  TH/BL: 1024
GPU 0: SHARED: 49152 B  CONST: 65536 B  # REGS: 65536
GPU 0: L2: 1280 kiB   MEM to L2: 208.0 GB/s  SP 1760.9 GFLOPS  OP/ELT 33.86
GPU 1: GeForce GTX 780 @ 1.02 GHz WITH 3071 MiB GLOBAL MEM
GPU 1: CC: 3.5  MP: 12  CC/MP: 192  TH/BL: 1024
GPU 1: SHARED: 49152 B  CONST: 65536 B  # REGS: 65536
GPU 1: L2: 1536 kiB   MEM to L2: 288.4 GB/s  SP 2348.9 GFLOPS  OP/ELT 32.58
```

In the example above information on two GPUs is given. (My machine and all of the graphic lab computers have two GPUs.) The first GPU is a Tesla K20c and has Compute Capability 3.5. The number of streaming multiprocessors (abbreviated `MP` in the output) is 13. The maximum number of threads per block is 1024.

Suppose the time step routine (a version of `time_step_cpu`) were going to be run on the GPU.

(*a*) Based on the program's default parameters and a GPU on one of the lab machines, how many blocks should be launched and how many threads per block? (The number of blocks and number of threads per block are collectively called the *launch configuration*.) *Please* indicate which GPU and its relevant statistics when answering this question.

(*b*) Do you expect the code to run efficiently with this launch configuration? Explain.