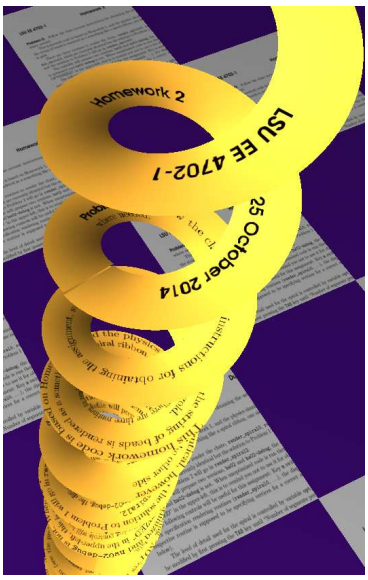
LSU EE 4702-1 **Homework 2/3** **Due: 25 October 2014**



Problem 0: Follow the class account instructions for obtaining the assignment, substitute hw02 where needed.

This homework code is based on Homework 1, and the physics simulation is identical, however the string of beads is rendered as a something like a spiral ribbon, one side is purple, the other side is gold.

There are three routines to render the chute, `render_spiral0`, `render_spiral1`, and `render_spiral2`, which are currently identical but the solution to Problem 1 will go in `render_spiral1` and the solution to Problem 2 will go in `render_spiral2`.

The makefile will prepare two versions, `hw02` and `hw02-debug`, the difference being that `hw02` is optimized and `hw02-debug` is not. When unoptimized code is run there will be a flashing "NOT OPTIMIZED" in the upper-left, this is to remind you not to use it for obtaining performance data.

The following controls will be useful for this assignment: Key `m` switches between the different vertex specification rendering routines (`render_spiral0`, ...), the display will indicate how the respective routine is *supposed to be* specifying vertices for a correct solution (see the problems below).

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. The image on the upper right shows the appearance of the spiral after a correct solution to Problem 2.

Problem 1: Examine the code in `render_spiral1`. There you will find that individual triangles are used to render the chute.

(*a*) Modify the code so that it uses triangle strips. Be sure to do this correctly, the ribbon should be gold on one side and purple on the other and fewer vertices should be sent to the GPU.

(*b*) Measure how much of a performance difference triangle strips make. (Do this by comparing to `render_spiral0` using the `m` key.)

(*c*) Estimate the amount of data sent from CPU to GPU for the two versions of the code.

(*d*) Performance should be better with the triangle strips. How much was the improvement from `render_spiral0` to `render_spiral1` affected by:

- reduction in computation by the CPU and GPU? *Note: The original question did not include the phrase "by the CPU and GPU".*
- reduction in data sent to the GPU?
- reduction in work performed by the fragment shader (or fixed function equivalent)?

Problem 2: The code in `render_spiral0` and `render_spiral1` apply a texture to the spiral. The texture is a draft of this homework assignment. The code applies the texture so that the top and

bottom of the page is mapped to the inner- and outer- part of the spiral. Modify `render_spiral1` so that the width of the page is one over the chain length and as one goes down the spiral one goes down the page. (As though the page were cut into horizontal strips and those strips were connected to form a spiral.) See the illustration above.

The problem below was originally assigned as Homework 2 Problem 3 but was later reassigned as Homework 3 Problem 1.

Problem 1: In this problem use buffer objects to store vertices for one spiral section and transformation matrices to render all of them.

(*a*) Modify `render_spiral2` so that it uses buffer objects. (See the code in `demo-7-vtx-arrays.cc` for example of how to do that.)

A spiral section is a spiral going from one ball to the next. It is what is constructed inside of the `t` loop. Use buffer objects to store just one spiral section. Use transformation matrices to transform the spiral section in the buffer object into the ones that are needed. For your convenience a routine `make_transform(f1,f2,t1,t2)` is available which will transform a spiral between points `f1` and `f2` to a spiral between points `t1` and `t2`.

(*b*) Measure the performance improvement over the two methods used in Homework 2.