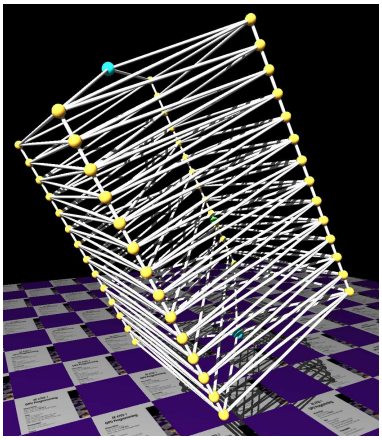
LSU EE 4702-1 **Homework 1** **Due: 14 September 2015**

Some of the effort for this homework assignment is in learning to use the various pieces of software, such as a text editor. Those less familiar with Linux software development procedures might seek out a more knowledgeable classmate to minimize frustration and wasted time.

Problem 0: Follow the instruction on the `http://www.ece.lsu.edu/koppel/gpup/proc.html` page for account setup and programming homework work flow. Compile and run the homework code unmodified. It should show a swinging string of beads. Press digits `1` through `4` to initialize different scenes, the program starts with scene 1. The illustration to the right is from scene 3. *Promptly report any problems.*

Use the arrow keys, PageUp, and PageDown to move around the scene. Use key `h` to toggle between the first (head) ball being locked in place and free. Use key `t` to do the same for the last (tail) ball. Press (lower-case) `b` and then use the arrow and page keys to move the first ball around. Press `l` to move the light around and `e` to move the eye.

Look at the comments in the file `hw01.cc` for documentation on other keys. One fun thing to do is to lock both the first and last ball, move the head ball until the spring is stretched tight, then release one of the balls. Press `p` to pause, then the space bar to single step. *Note: There is nothing to turn in for this first problem.*



Problem 1: Pressing `2` shows scene 2, a truss and three marker balls. The truss should fall but the marker balls should stay in place. Modify the code in `balls_setup_2` so that the ends of the truss (the peak of the pyramid at each end) are initially in the same position as the cyan marker balls and so that a longitudinal (long direction) link passes through the green marker ball, see the screen shot to the right.

The locations of the marker balls are declared at the top of `balls_setup_2`, and a little further below there is a place for a solution. The problem can be solved by initializing variables `spacing`, `delta_unit`, `loc_x`, and `loc_z` correctly.

Of course, the truss must be between the head and tail balls and should have `chain_length` units. The modified code should make good use of the coordinate classes. For example, DON'T set `x`, `y`, and `y` members individually.

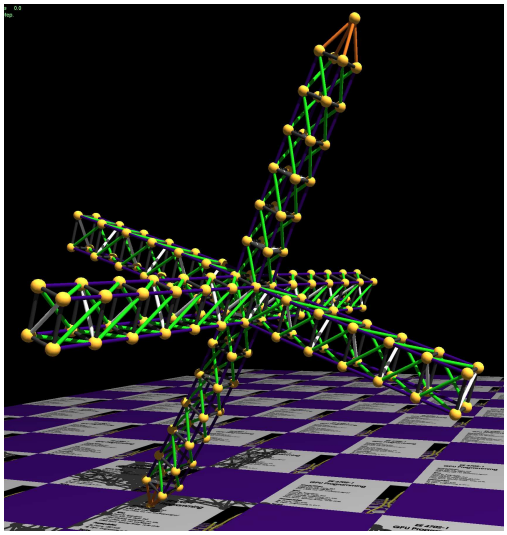
The mathematics for this problem is fairly simple: you need to find the vector pointing from `head_pos` to `tail_pos`, call that vector the *axis*. The vector `delta_unit` should be some fraction of the axis vector. Find the closest point on the axis to `surface_point`, call it S. Use S to find `loc_x`, `loc_y` and `spacing`.

To test out a solution it might be helpful to pause simulation before switching to scene 2. Then switch to scene 2 and check whether the cyan balls are at the head and tail locations and whether the green ball is on a longitudinal link.

Problem 2: When `w` is pressed the simulator will call routine `balls_twirl`, which currently does nothing. Modify the routine `balls_twirl` so that it adds velocity to balls in a direction around the axis defined by the position of `head_ball` and `tail_ball` (if these are defined). The amount of velocity should be based on the distance from the axis, so that collection of balls rotates as a unit.

The mathematics for solving this problem are similar to the mathematics for the first problem. Write an equation for the line connecting the head and tail balls (the axis), say $S = B_0 + tv$, where B_0 is the first ball and v is a vector pointing towards the last ball. If we are choosing a velocity change for ball B, we need to find the point on the axis closest to B. For such a point $\overrightarrow{SB}$ will be orthogonal to v. One can use a property of the dot product to solve for S and another operation to find the force direction, a vector orthogonal to both v and $\overrightarrow{SB}$.

Problem 3: Modify `ball_setup_4` so that it adds two or four more trusses to the existing truss, these should be neatly attached to the center of the original truss and should fall on a plane normal to the original truss. See the illustration to the right.



The truss should be well balanced, so that when it is twirled (using `w` from the previous problem) it balances like a gyroscope.

To solve this problem examine the code in routine `make_truss`, and also pay attention to the comments that describe how balls and links are added to the lists `balls` and `links`.