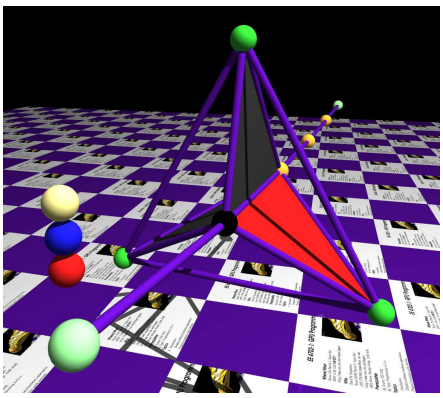
LSU EE 4702-1 **Homework 1** **Due: 20 September 2017**



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 initially show a string of beads stretched out horizontally with either end fixed in space by some invisible force, see the illustration to the right. Pressing `h` (head) will release one end (to be precise, the ball at one end) and pressing `t` (tail) will release the other end. (Actually, those keys toggle the fixed-in-space status of their respective balls.)

The cable includes a vaguely paddle-wheel like thing that includes triangular sheets of material which will be called blades. The blades are red on one side and dark gray on the other. Each corner of the blade connects to a different colored ball. The corner connected to the black ball is called the *base vertex*, the corner connected to the green ball is called the *circ vertex*, and the corner connected to the yellow ball is called the *apex vertex*.

The scene also includes (initially) three stationary balls, colored khaki, blue, and red. These are called *markers* and are intended for debugging purposes. See routine `World::render_p1` for an example of how to use the marker balls. Another aid for debugging are Boolean variables `opt_tryout1` and `opt_tryout2`. Pressing `y` toggles `opt_tryout1` between `true` and `false` and `Y` toggles `opt_tryout2`. The state of these variables is shown in the penultimate line of green text. Feel free to use the marker balls and tryout variables to help write and debug your code.

There three versions of the scene, *Experiment*, *Problem 1*, and *Problem 2*. The version currently being displayed is shown in the penultimate line of green text (not shown in the illustration). Pressing `v` cycles between the different versions. When the Experiment version is active the code in `World::render_p0` is executed. Changes to this routine won't be graded, it's intended to try things out or at least to have something clean to look at after having extensively modified other routines. The code in `World::render_p1` and `World::render_p2` are active when scenes Problem 1 and Problem 2 are active. The changes to be made in these routines are described in the problems below.

Press digits `1` through `2` to initialize different scenes, the program starts with scene 2. Scene 1 was described above, scene 2 shows a ... a ... make up your own name for it. Scene 2 is not part of this assignment.

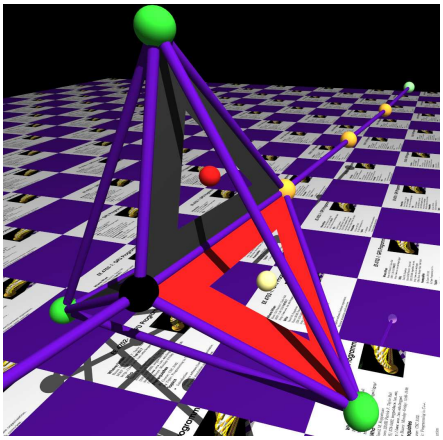
Initially the arrow keys, PageUp, and PageDown can be used to move around the scene. Press (lower-case) `b` and then use the arrow and page keys to move the first ball around. Press `l` to move the light around and `e` to move the eye (which is what the arrow keys do when the program starts).

The `+` and `-` keys can be used to change the value of certain variables to change things like the light intensity, spring constant, and variables needed for this assignment. The variable currently affected by the `+` and `-` keys is shown in the bottom line of green text. Pressing `Tab` cycles through the different variables. Those who want to increase the spring constant to the point that the scene explodes may be disappointed to learn that there is a protection mechanism that increases the mass of balls when the spring constant is high enough to make the system go unstable, such balls turn red.

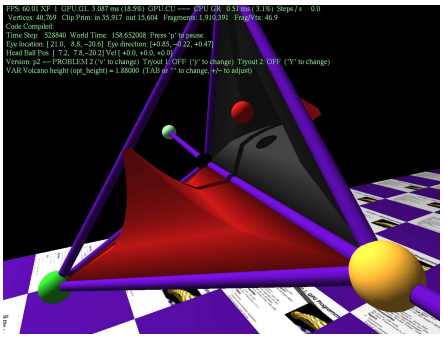
Look at the comments in the file `hw01.cc` for documentation on other keys.

There is a problem on the next page.

Problem 1: Version Problem 1 is initially the same as the Experiment version, except that a marker ball is placed at the center of each blade. Modify the code in `World::render_p1` so that the blade has a triangular hole in the center, see the illustration to the right. Do so by using a triangle strip for each blade.



Problem 2: Version Problem 2 is initially the same as the Experiment version, except that a marker ball is placed at the center of each blade. Modify the code in `World::render_p2` so that the blade is shaped sort of like a volcano, as in the illustration to the right and as described below.



Let a, b, and c denote the coordinates of the apex, base, and circ vertices. Let m denote the coordinate of the center of the triangle (shown as `mid` in the code). Let $\overrightarrow{n}$ denote the triangle normal and following our usual notation, $\overrightarrow{am}$ is a vector from a to the triangle center.

The shape of the volcano will be defined by the parametric line $P_a(t) = a + t^e\overrightarrow{am} + td\overrightarrow{n}$, where e and d are constants. Note that $P_a(0)$ is point a (the apex) if $e = 1$ and $d = 0$, point $P_a(1)$ is the center of the triangle. For other values of e and d the line ascends the volcano as t increases. Define $P_b(t)$ and $P_c(t)$ similarly. For this assignment set e to the value of `opt_e` (default, 0.3) and d to the value of variable `opt_dist`. The volcano is rendered by varying t from 0 to 0.9. (If t varied from 0 to 1 it would be a mountain, not a volcano.) Use variable `opt_layers` for the number of values of t (evenly spaced between 0 and 0.9).

(a) Modify the code in `World::render_p2` so that it renders each side of the volcano using a triangle strip. (One side is between points a and b, one side is between b and c, and one is between c and a.)

(b) Choose the normals based on the shape of the volcanoes defined by parametric equation. If this is done correctly the volcano will be smoothly colored consistent with the lighting, as illustrated.

Note that using the normal of one of the triangles adjacent to a vertex as the vertex's normal is an approximation. An exact surface normal can be found by taking the cross product of two vectors that lie on the surface. Taking the derivative of the parametric equation with respect to t provides a vector along the surface. So $\frac{dP_a(t)}{dt}$ gives a vector on the ca and ab sides.