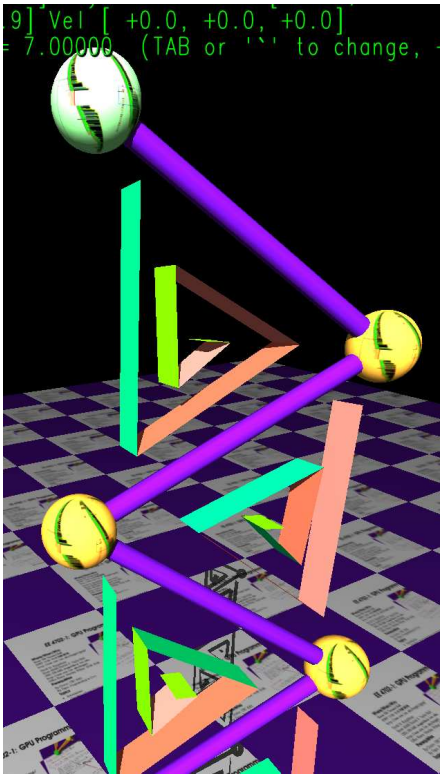
LSU EE 4702-1 **Homework 2** **Due: 21 September 2018**



Problem 0: Follow the instructions on the `https://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 connected in a zig-zag pattern. Pressing `h` (head) will grab or release one end (to be precise, the ball at one end) and pressing `t` (tail) will grab or release the other end. (Actually, those keys toggle the fixed-in-space status of their respective balls.)

General User Interface

Press digits `1` through `4` to initialize different scenes, the program starts with scene 1. Scene 1 starts with the balls arranged almost vertically. The other scenes are not used in this assignment.

Press `Ctrl=` to increase the size of the green text and `Ctrl-` to decrease the size. 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.

Keys `y` and `Y` toggle the value of Boolean variables `opt_tryout_1` and `opt_tryout_2`. They are intended for debugging and trying things out.

Look at the comments in the file `hw02.cc` for documentation on other keys.

Code Generation

The compiler generates two versions of the code, `hw02` and `hw02-debug`. Use `hw02` to measure performance, but use `hw02-debug` for debugging. The `hw02-debug` version was compiled with optimization turned off and with OpenGL error checking turned on.

Physics Model, Graphics, and Problem-Specific UI

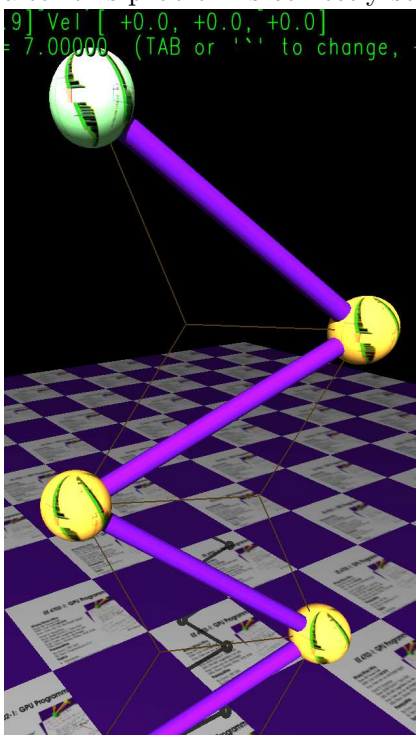
The physics model in the Homework 2 code is similar to the one implemented in Homework 1 Problem 1, in which the cord was given some stiffness by adding a spring between balls at distance 2, except that the relaxed length of all the springs are the same. This has the effect of pulling the cord into a zig-zag pattern.

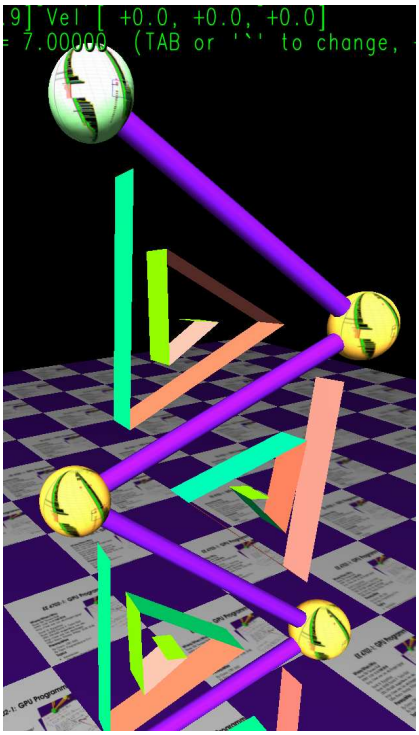
Pressing `f` will cycle through different methods of filling the triangles formed by the balls. The possibilities are none, wye, and louvers, the screen shots in Problem 1 show the intended appearances for wye and louvers. The code in routine `hw02_render` provides the needed fill. The

routine already correctly renders the wye option, in this assignment the louver option will be implemented.

Pressing `w` will toggle shadows on and off. Pressing `W` will toggle the visibility of shadow volumes. Shadow volumes are used to render shadows, see Problem 2.

Problem 1: The two screen shots below show Scene 1 rendered with fill option set to Wye and Louver. None and Wye should work with unmodified code, the screen shot for louver shows the appearance after this problem is correctly solved.





Consider a group of three balls forming a triangle. In the Wye image there are faint yellow lines from each ball to the center of the triangle. This is drawn by the code in routine `hw02_render` guarded by `opt_fill == Fill_Wye` .

The louver fill option renders what looks like a strip folded into a triangular spiral. Each fold on the strip occurs on a line between a ball and the triangle center (the yellow lines shown with the wye fill option). The strip starts in the triangle center.

Let s denote the number of segments, which is the value of variable `opt_n_segs`. Let l_0, l_1, and l_2 be the distance from ball 0, 1, and 2, to the triangle center, where ball 0, 1, and 2 refer to *any* three adjacent balls forming a triangle. Let *fold 0* refer to the end of the strip in the center (so there's no real fold). The distance from fold j, $0 \geq j \geq s$ to the triangle center is $\frac{0.8j}{s} l_{(j \bmod 3)}$.

Here are some other details:

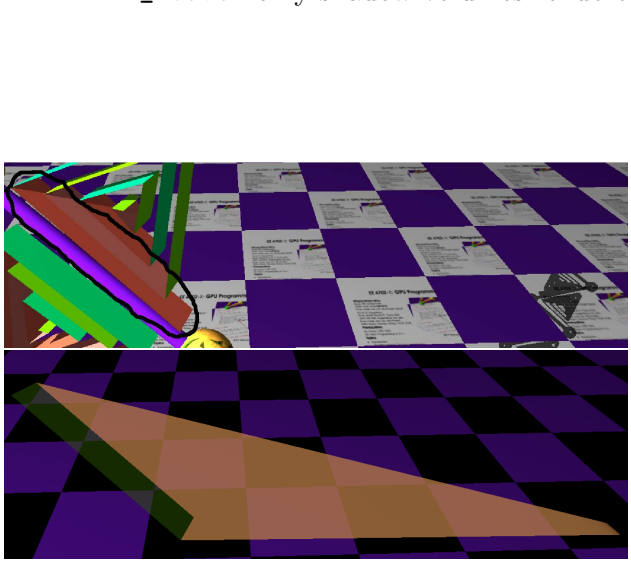
- The width of the louvers should be 0.1 units.
- The front of the louvers should be facing the center of the triangle.
- Use colors in the array `fcolors` for the front colors and `bcolors` for the back colors. Alternate the colors in these arrays to enhance visibility.
- Be sure to set the surface normals correctly.

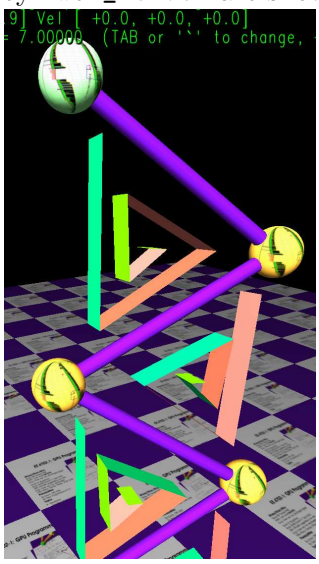
- Use `opt_n_segs` for the number of segments. Make sure that the code reacts when its value is changed with the UI.
- Before solving Problem 2, don't render anything when `shadows` true.

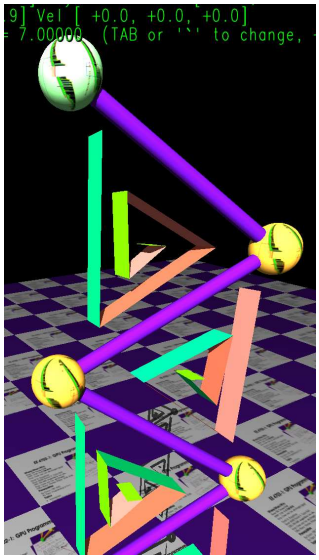
Modify `hw02_render` so that when `opt_fill == Fill_Louver` it renders the louvers as described above.

Problem 2: One way to render shadows is by using shadow volumes. A point is in the *shadow volume* of some light and some object if a line from the point to the light passes through the object. That is, everything in the shadow volume is shaded from the light by the object.

One way to create shadows in OpenGL uses triangles (or other primitives) placed on the surface of the shadow volume and facing outside of the volume. The surfaces start at the object being shaded and end some large distance away. Consider the triangle defined by points P_1, P_2, P_3, and a light at location L. One surface would be the quadrilateral defined by points P_1, P_2, $P_2 + d\overrightarrow{LP_2}$, and $P_1 + d\overrightarrow{LP_1}$, where d is some large distance. Similar quadrilaterals can be defined for points P_2 and P_3 and for points P_3 and P_1. The pair of screenshots below on the left show a shadow volume for one segment of the louver. Pressing `W` (upper-case) toggles the visibility of shadow volumes. When `opt_fill != Fill_Louver` all shadow volumes are shown, when `opt_fill == Fill_Louver` only shadow volumes rendered by `hw02_render` are shown.







Modify the code in `hw02_render` so that when `shadows` is true and `opt_fill == Fill_Louver` it emits shadow volumes for the louvers. The two screenshots above right show the louvers without and with shadows.

It is important that the shadow volume triangles (or other primitives) face outside the volume.