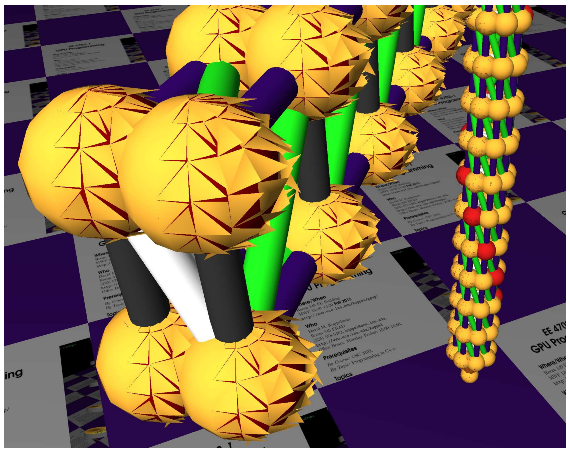
LSU EE 4702-1 Homework 4 Due: 11 November 2015



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 start with an 8-arm gyroscope. Press the `+` and `-` keys to change the number of armds. Press digits `1` through `3` to initialize different scenes, the program starts with scene 3.

Pressing 'v' cycles between shaders.

The code in this package includes the friction model asked for in Homework 2. The links however are unbreakable and there's no skin. Also, shadows are deactivated by default to make the solution to this assignment easier.

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.

Note: There is nothing to turn in for this first problem.

Problem 1: The geometry shader `gs_main_simple` in file `hw04-shdr.cc` passes triangles through unmodified. We know that when rendering spheres and cylinders (at least the cylinders that we use for links) a triangle that's not facing the user will be behind one that is facing the user. It makes sense to discard such triangles early.

(*a*) Modify the geometry shader so that it does not emit triangles based on whether they are facing the user and the value of uniform variable `tri_cull`. If `tri_cull` is zero always emit the triangle. If it is 1, emit the triangle only if the triangle front is facing the user. If `tri_cull` is 2 emit the triangle only if the back is facing the user. Base this on the triangle coordinates, not the normals. The value of `tri_cull` can be changed by pressing `c`.

(*b*) Estimate the amount of work saved. Use the following symbols: Let c_v, c_g, and c_f be the amount of work done by an invocation of the vertex, geometry, and fragment shaders respectively. For example, if we rendered one triangle covering 22 pixels the total work would be $3c_v + c_g + 22c_f$ units. For your answer let n_t denote the number of triangles, and make reasonable choices for other quantities.

(*c*) Try to determine actual values for c_v, c_g, and c_f by running your code. Pause the simulation to make sure the physics isn't hogging CPU. The number of vertices is shown on the green text near the top, the total number is at the end of the line. Triangle strips are used for both the balls and the links. Pressing the `+` and `-` keys when setup 3 is visible will change the number of arms on the gyroscope, use this to vary the number of balls and links.

Problem 2: The instance shader code rendering links shows all links the same color.

(*a*) Modify the shaders in `hw04-shdr-links.cc` so that that the fragment shader obtains the color using a value based on `glVertex_ID` rather than reading the color itself from on of its inputs. Only the vertex shader has access to `glVertex_ID`, define shader inputs and outputs as needed to send its value to the fragment shader.

Modify other shaders in this file consistently to remove the old code passing color data through the pipeline.

Note: The original version of the problem was to use pre-defined fragment shader variable `glVertex_ID`*. As a student pointed out, there is no such variable in the fragment shader.*

Problem 3: Modify `gs_main_simple` so that it can emit a second triangle that looks like it is being peeled off a moving sphere by the rushing air. The second triangle should be the same shape as the original triangle and should share one edge with the original triangle. See the illustration at the beginning of this assignment. The choice of shared edge and the angle of the triangle should be based in some way on the velocity of the ball.