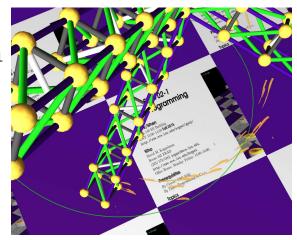
Problem 0: Follow the instruction on the http://www.ece.lsu.edu/koppel/gpup/proc.html page for account setup and programming homework work flow. Compile and run the homework code unmodified. It should show a swinging string of beads. Press digits 1 through 3 to initialize different scenes, the program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

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. Notice that this starts like the code for Homework 2, except that ball at the bottom of the chain hits the platform as it swings by.



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 1 to move the light around and e to move the eye.

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

Problem 1: Notice how the platform is always perfectly clean. That's not realistic! In this problem modify the code so that when the balls slide across the platform they leave scuffs. The shape of a scuff should be the area of the platform touched by the ball (base the width on the intersection of the sphere and the platform) and the color should match the ball. See the illustration to the right. The ball at the base of the gyroscope is green, it leaves a neat green path. The arm balls are yellow. The balls at the arm ends striking the platform leave scuffs that start out thick and get thinner.

The code should be added to class My_Piece_Of_The_World and elsewhere as needed. The places in which code needs to be added start with the comment Homework 3 and are followed by checklists of things to be done.

Scuffs are to be implemented using textures. Rather than using one large texture covering the entire platform, the textures are broken up into a grid of *overlays*. The texture corresponding to an overlay should only be initialized when the area is first scuffed, and textures should only be sent to OpenGL when the textures change. This uses vastly less data then sending a texture for the entire platform at every frame.

(a) Code has already been added to time_step_cpu to retrieve an overlay, actually a Platform_Overlay object, for a ball if it is colliding with the platform. Add code to this routine and elsewhere to write the appropriate texels with scuff marks. For sample code see sample_tex_make in the homework file. This routine creates a texture containing a big red ex. (The code in time_step_cpu only needs to modify the texels in po->data, it should not send the data to OpenGL!) (b) Routine My_Piece_Of_The_World::render() is supposed to render the platform overlays. For each scuffed overlay it should emit primitive(s) with the corresponding texture attached. The texture application, alpha test, and blending modes need to be set so that only the scuff marks cover the underlying platform (and semi-transparently if possible). The routine has some code. Finish it, and make changes elsewhere as needed.

See gpup/demo-8-textures.cc for examples of how to do texturing, blending, and alpha testing. If you are not sure if your own texture is working you can use the texture created in sample_tex_make.

Problem 2: Answer each question below about performance aspects of texturing.

(a) The texel array that we send to OpenGL is an array of floats, because that lets us use our familiar pColor objects. What would be the impact of using a color representation that matched the depth of our displays, eight bits per color?

(b) Suppose we vary nx and nz, and choose twid_x and twid_z so that the total number of texels across the platform, nx * twid_x, is constant. Describe the impact on performance as nx and nz change vary from 1 to their maximum (say, 10240 for the assignment code).