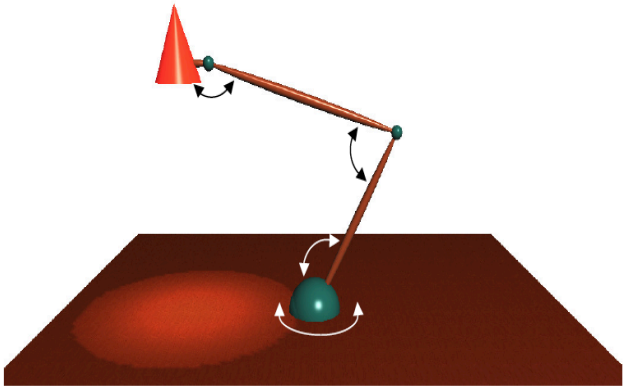
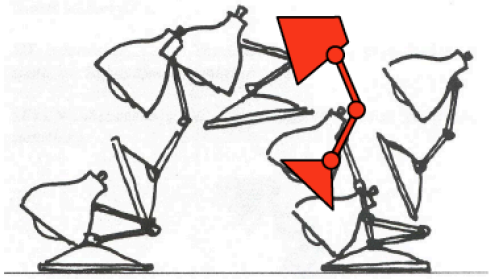
Computer Graphics

Interpolation and Animation





Your goal in this assignment is to animate the hierarchical object from your previous assignment using interpolation. To be specific, we will use the Luxo lamp as an example.

1. The right image above shows 6 keyframes for the Luxo lamp – similarly, you will need to create a small number of keyframes for your own model. Each keyframe is defined by a number of parameters (the degrees of freedom for your model) – in the Luxo example, the position and the joint angles. Pose your model and save all its degrees of freedom to a file – this saves a keyframe for your animation. You may define your own format for the keyframe file, but a natural choice would be to have one keyframe per line as follows:

   ```
   keyframe <time> <param1> <param2>, ...
   ```

   Here **<time>** is needed to define the range for the parameter ***t*** used in interpolating the subsequent parameters. In the case of the Luxo lamp, one keyframe would correspond to a line such as

   ```
   keyframe  0.5 2 5 90 70 85
   ```

   The first parameter `0.5` indicates the time elapsed between the previous keyframe and this (current) keyframe. The next 2 parameters (2, 5) may indicate the position of the lamp base, and the last 3 parameters may indicate the angles at the three lamp joints. This is just an example – your model will likely have different parameters.

   Your solution to the previous assignment should simplify the task of determining these parameters. Simply use the keyboard to move various parts of your model in the desired position, and record the parameter values – you may even assign the recording (printing on the screen, or saving into a file) action to some other key on the keyboard.

2. Once you've decided on the keyframes to use, write code to read in these keyframes and interpolate them to determine he inbetween frames. You will likely need to use linear interpolation for the joint angles, but in the case of the Luxo jump depicted in the right image, cubic interpolation is necessary to capture the curve trajectory of the lamp. Sample code for reading the keyframes in and using linear interpolation between them is available on the class website, so you won't have to start from scratch. The sample code uses a piece of the hierarchical robot for animation, so that it is not entirely new to you.

You can decide how many interpolation steps to take between each keyframe. This will decide the timing of your animation. This can be constant or different across all frames.

3. Dump PPM (Portable Pixel Map) images from your application during animation (the online sample code shows you how to do this) and then compile all the individual frames into a single movie. In Unix, you may convert these PPM image files to a GIF movie using the command

   ```
   convert *.ppm movie.gif
   ```

   I have not tried a similar method in Windows – you might need to convert the PPM images into an intermediate format, such as JPEG, and compile the JPEG images into a movie using Images To Video (http://www.addictivetips.com/windows-tips/images-to-video/) or the Windows Movie Maker. Find information online on how to do this and share it on Piazza with your peers.

   If you want to keep your movies reasonably small in size, I recommend choosing a fairly small window size (for example, 500 x 500) for your animation.

4. Do not worry about colors in this assignment. You will be graded on several things:

   a. Your animation is creative and interesting.
   b. Your code uses linear and cubic interpolation.
   c. Good use of the timer callback function to pace your animation.
   d. Your code is tidy.
   e. Your readme file is comprehensive.

Hand-in instructions:

At the beginning of your code, write a clear description of what the program does. Insert explanatory comments throughout your code. Email to your instructor your movie, your source code and a readme file explaining the amount of time spent on this project, any known bugs, and any suggestions for future improvements to the assignment.