Lab 3: Programming Dynamic Haptic Interactions

Due: Tuesday 3/24 by 10:45AM

Please read this entire document before starting the lab.

This project must be completed and submitted individually. You are welcome to discuss the assignment with other students in the class, but you do need to program your own solution.

To submit this assignment, please ZIP and email just your “main.cpp” file to Drs. Abbott and Provancher.

For your reference, you may wish refer to several past lectures from Dr. Allison Okamura’s haptics course (posted on the readings page for your reference)

- Refer to pp. 5-6 of Dr. Okamura’s lecture10.pdf for reference intro info on dynamic equation formulation
- Refer to pp. 17-28 of Dr. Okamura’s lecture12.pdf for reference info on haptic rendering (relevant for Lab 2 and Lab 3)
- Refer to pp. 28-36 of Dr. Okamura’s lecture14.pdf for reference info on dynamic simulation

We would like to acknowledge Francois Conti, Federico Barbagli, and Kenneth Salisbury at Stanford University. The lab exercises presented herein largely come directly from their Experimental Haptics Course, CS277, and is based on their CHAI-3D environment.
For this lab, you will simply build upon the software files that were given to you in Lab 1. Therefore, all the files you need can be found on the wiki (posted from Lab 1) or on the computers in the lab in the directory `C:\Haptics\Lab1\Lab1\`. Again, you will only need to modify the main.cpp file, which is located in the “Source Files” folder. Refer to other instructions from the Lab 1 handout as necessary.

**Remember, before you run your code.**

**FIRMLY GRASP THE HANDLE/KNOB OF THE FALCON TO PREVENT IT FROM ACCIDENTALLY DESTROYING ITSELF BY SLAMMING INTO ITS OWN HARD STOPS OR OTHER OBJECTS IF IT HAS AN UNSTABLE OR UNINTENDED PROGRAMMED BEHAVIOR!!!**

**Coordinate Frames**

For all of the graphical and haptic virtual environments, the coordinate frames used are defined as follows:
- Right = +y, Left = -y
- Up = +z, Down = -z
- Out = +x, In = -x

**Your task for this lab is to create 4 new haptic virtual environments.**
1. **1-D Mass-spring dynamic simulation**

   This exercise is meant to allow you to learn a little more about programming dynamic interactions for haptic display and utilize simple graphical objects in OpenGL.

   Please implement a 1-DOF mass spring system (either horizontal or vertical) that has a mass attached to one end of a spring and the HIP attached to the other end of the spring. You will need to solve $F = ma$ and use numerical integration to determine the motion of the mass. Please refer to Dr. David Johnson’s lecture and your undergraduate physics or dynamics books as necessary.

   You do not need to render any collision (i.e., the mass can pass right through the HIP), just the feeling of the inertia of the mass connected to the HIP with the spring. You may choose the stiffness and mass(es) as you like. Please represent the mass and hip with circles or spheres, and the spring with a line. If you can, please also represent the force experience at the hip with a vector pointing out of the center of the hip that has a length that is proportional to the rendered force.

2. **1-D Collision**

   This exercise is meant to allow you to learn a little more about programming dynamic interactions including collision for haptic display and utilize simple graphical objects in OpenGL.

   Please now add mass to your HIP and remove the spring from part 1 of this lab. Now you will haptically render collision between the mass and HIP (i.e., the mass can no longer pass through the HIP). Both the HIP and mass will also collide with the ends of travel (what is referred to as the ceiling and floor boundaries in the figure to the right) (you may also make your 1-DOF motion left-right if you like).

   Please graphically render the mass and HIP with circles or spheres, and the length of your track with a line or a box (as shown).

   You will need to solve $F = ma$ and use numerical integration to determine the motion/forces on the masses. Please refer to Dr. David Johnson’s lecture and your undergraduate physics or dynamics books as necessary.

   Hint: you will likely need to make some assumptions about the “coefficient of restitution” in order to deal with the dynamic collision event. You may want to play with different coefficient values.
3. Virtual Fixture (linear track)

This exercise is meant to allow you to learn a little more about using proxies (such as what you used in the last lab to implement a virtual wall) for other useful applications such as virtual fixtures (i.e., constraining the trajectory to a desired motion). This exercise constrains the user to move on one of two lines, but other virtual fixtures can use combinations of other shapes or virtual walls, etc.

Please implement a virtual fixture that constrains the user to move on one of two lines as shown in the figure on the right. Utilize a virtual spring to “encourage” the user to not deviate from this right angled path.

Please graphically render the Proxy and HIP with circles or spheres, and the virtual fixture track with lines (as shown – the lines don’t need to be dashed).

Hint: you will recognize the behavior of this as being similar to the “attracting line” from Lab 1. However, you will need to render both the proxy and HIP.

4. Virtual fixture/coupling with proxy dynamics

This exercise is meant to allow you to learn a little more about using proxies (such as what you used in the last lab to implement a virtual wall) for other useful applications such as virtual fixtures and virtual couplings. This exercise builds on part 3 of this lab and adds dynamic behavior to the proxy. You will add some type of “drag” to the proxy, but also realize that you could generally also add damping in parallel with your virtual spring or other desirable behaviors to suite your application.

Please add some sort of “drag” to the proxy from part 3. The drag may be in the form of coulomb friction \( f = \mu \cdot N \) or viscous damping (or both… whichever is easier for you to implement).

Please graphically render the Proxy and HIP with circles or spheres, and the virtual fixture track with lines (as shown – the lines don’t need to be dashed). Please also display a line vector indicating the relative magnitude of the “drag force” on the proxy.

Hint: you will recognize that you can use your work from part 2 of Lab 2 to implement
coulomb friction. Alternatively if you implement viscous drag, that you will need to store the previous position of the HIP (and perhaps filter) to get a velocity estimate to use in your viscous drag formulation.

**Hints:**
Whenever possible, try to think in terms of vector math, rather than breaking vectors into their individual components.

All of the code for the OpenGL objects came from online examples linked below.

**OpenGL Libraries:**
Information about the OpenGL libraries (Red Book) can be found here: 
http://fly.cc.fer.hr/~unreal/theredbook/

As well as some source code examples:
http://www.opengl.org/resources/code/samples/redbook/

Also, Aman found some nice video tutorials for OpenGl and you can learn it very fast. It's introductory videos shows the installation of OpenGl and how to setup it's path to interact with MSVC++.

The Lab1 package contains everything installed (OpenGL, Glut & Microsoft SDK), but you will need to figure out how to display a line and perhaps some circles. You can follow the methods shown in video tutorial in the link below to help you code your lab.

http://www.videotutorialsrock.com/opengl_tutorial/get_opengl_setup_windows/video.php
www.videotutorialsrock.com