Lab 2: Programming Haptic Environments

Due: Monday 3/2 by Midnight

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.

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. God-object/proxy based virtual wall
This exercise is meant to allow you to learn a little more about programming graphical objects in OpenGL and learning to program a virtual wall using a God-object proxy
- Your first task is to modify the graphics and the virtual wall from your last lab so that the wall is on the bottom of the screen as shown to the right.
- Next, you’ll want to add a new point that represents the proxy point that stays at the surface of the virtual wall. The prior point that was displayed in the last assignment is the Haptic Interaction Point (HIP), that was discussed by Dr. David Johnson in class. Ensure that the proxy and HIP are displayed appropriately as you move along and into your virtual wall.
- Finally, add a line that is displayed at the point of the proxy that indicates the orientation and relative magnitude of force that is currently being applied by the haptic device.
Hints:
  a) I suggest that you take a look at the OpenGL online tutorial that Aman found and was forwarded to the class in early February. I will also put a copy of the “Red Book” in the small robotics lab for your reference. Please do not remove this book.
b) Remember to think about coding this assignment in vector form so that it could still work if you changed the location or orientation of the wall.

2. Friction using 2-proxy method
Please add friction to your virtual wall from part 1 in the form: friction = $\mu N$ or similar. You may try to add static and dynamic friction mode switching, but this can be tricky and is not required. A student from Dr. Provancher’s research group has used a modified Karnopp friction model. I have you implement the “static” part of this model below. You may find a paper on rendering friction and skin stretch using a modified Karnopp friction by Nick Sylvester on Dr. Provancher’s lab site. For your reference, Vincent Hayward’s group often uses a modified Dahl Friction model. You can probably find a paper on Dr. Hayward’s publication page that describes their friction model

- One way of rendering friction is through the use of a virtual spring. This can be a simply linear spring. This is a convenient way to represent the force generated by “static friction” and realistically represents the increase of force you would naturally feel as you slide your finger over a surface. Generally speaking this force will increase till the friction force exceeds $\mu_s N$. Once the static friction limit is exceeded, sliding ensues… typically accompanied by a reduction in friction force corresponding to kinetic friction $= \mu_k N$. Since this mode switching as well as the transition back to static friction can be tricky, simply saturate the friction experienced by the user through the virtual friction spring at $f_{\text{max}} = \mu_s N$.

- To implement your virtual friction spring, create a second proxy point/circle/sphere. This friction proxy will be coincident with the HIP and God-object proxy when you are moving in free space, so please choose the relative size of these 3 spheres so that they all can be seen when moving in free space. Make it so that you start to experience normal forces when the God-object proxy touches the virtual wall.

- When in contact with the wall, the friction proxy will remain fixed/anchored. As you move to one side, you will stretch the virtual spring, generating increasing amounts of force (displayed as friction force tangent to the virtual wall). This force will increase until the static friction limit is exceeded ($f_{\text{max}} = \mu_s N$, hence you will need test for this condition). Once you exceed this limit, the friction proxy will begin to translate at the same rate as the God-object proxy (keeping the lateral friction force constant).
  - If you go in the opposite direction, this will reverse the stretch in the spring.
  - Also note that if you push hard on the surface and move to the side this will stretch the friction spring. If you then pull your finger straight out of the surface, note that the friction proxy will actually move laterally towards the God-object proxy until the friction force is zero when you leave the surface.
  - Please make the friction spring constant small enough to be able observe the relative displacement between the God-object proxy and friction proxy.
3. Virtual Textures
Textures can be used to help a user identify given regions in a virtual environment or for other purposes. Please try your hand at creating your own virtual texture.

- Please take the virtual wall from part 1 and add a virtual texture to it. There are a variety of ways to render virtual textures. Please implement your own method for rendering a texture, or else go to one of several haptics researchers websites and search for “texture” and look for how these researchers have implemented texture. Vincent Hayward’s group has implemented textures in several recent papers using a periodic sine function that would be appropriate for this assignment (http://www.cim.mcgill.ca/~haptic/publications.html).

4. Planar detents
It can be helpful to guide a user to a point (e.g., a push button) or around a point (e.g., a blood vessel). Please implement a planar field of attracting and repulsing circular detents.

- Please constrain the Falcon to move in the plane of the LCD screen (in the y-z plane and any x coordinate location you like).
- Draw/create several circular regions of various size and distribution on the screen.
- Create potential fields within these circular regions that either attract or repulse the user from the center of these regions (at least 2 sizes of attracting and repulsing of each. Please also play with the spring stiffness of your potential fields).

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