CSCI2951-G: Homework 0

Paul Valiant

Due: Next class.
Turn in: A screenshot of proteins, from the provided code, running on your own computer (Windows or Linux), or a lab computer you have easy access to.

Apologies in advance to those of you who have to wrestle with their computers to get this working; once you do, the rest of the assignments will be much more fun.

The source code is at `www.cs.brown.edu/courses/csci2951-g/proteins.zip`. Your job is to compile and run it. You will need a copy of MATLAB (lab computers have them installed, and I think Brown students get free copies in general). If you've never used Matlab before, you can google for help ("matlab introduction"), or just follow instructions here.

Very brief Matlab introduction: type a Matlab command and press enter for it to be evaluated – for example, try:

```
1+1
```

Matlab also has built-in functions, like

```
sin(100)
```

If you see spaces instead of parentheses, that means every argument is interpreted as having quotes around it:

```
myfunction hello world
```

is the same as

```
myfunction('hello', 'world')
```

(Matlab uses single quotes). Semicolons (";") suppress output, so `1+1;` will produce no output. You can put together a bunch of Matlab commands, save it in a file with the extension ".m", and run all the commands in the file together; you can define your own functions in a similar way. Matlab also works well with compiled C/C++ code, as you'll find out.

Getting Proteins Running: The code here should work on Linux and Windows. If you find problems or issues, let me know, particularly if you think other people might have similar trouble.

1. Unzip all the files into a directory.

2. Start Matlab, and change Matlab to that directory (click the "..." next to the directory location at the top of the Matlab window).

3. Make sure you have a C++ compiler installed, and then type `mex -setup` and follow instructions.

4. Compile something easy first: `mex myang2pos.cpp`

5a. Compile the main routine. On Windows the command is

```
clear mex; mex -I. -L. -lfreeglut proteins.cpp
```

5b. On Linux: `mex proteins.cpp -lglut -lGL -lGLU`

6. Try loading protein data into Matlab:

```
d=loaddat;d2=loadprotein(d,{'asn','leu','tyr','ile','gln','trp','leu'
,'lys','asp','gly','gly','pro'},'b');p=loadphys;p2=applyphys(d2,p,'i');
```

(that should all be on one line)

7. Run it! `proteins(d2.p,d2.edges,d2.types,p2);`

In Matlab, press up-arrow to bring up previous commands you have entered; if you partially type a command and then press up-arrow, Matlab will fill in the rest of the command for you from commands you have previously entered. (Pressing tab asks Matlab to search for valid commands, in general.

If you run into problems, let me know, since I've probably encountered them before, and the point of this class *isn't* to get into the details of C++ linkage errors. But if you really want to, you can add a `-v` flag to the mex command to make it output more verbosely.

I'm not going to go into details about how to tweak things yet, but, you can probably guess that the long list inside the `loadprotein` command is a list of amino acids, so, feel free to google "amino acid" and add your own three letter abbreviations to the list. If you want things to run faster, get rid of the `'i'` at the end, since that adds water simulation. If you change the `'b'` to an `'a'`, that'll start the protein in a helix configuration.

When you have the thing running, press space bar to unpause it. The bottom slider in the bottom right controls friction, so once the simulation is started, you might want to set it down to 0. The middle slider controls temperature – by default set to body temperature (in Kelvins). Pressing "b" toggles between whether the backbone atoms are displayed larger than the rest of the atoms or not, which can help visibility. You can also try pressing "c" or "w". And, click and drag anywhere in the window to rotate the view.

3D stereo is a rather delicate thing, so if you want to set that up properly, right click in the window, and make sure two things are set right: the distance from you to the screen, and the size of the screen. Also, if you are viewing this on a laptop, be sure to turn the screen brightness all the way up, as the red/blue glasses really dim the image.

If you are feeling ambitious, my admin assistant, Saara Moskowitz, has a 3D mouse for you to borrow if you drop by her cubicle on the 5th floor (546, by the center stairwell, though she may move later this semester). If you want to use this on Windows, you will have to install the driver from `www.3dconnexion.com/service/drivers.html` (it's the "Space-Navigator"), install it, and then run "Trainer" before you try to use it with Matlab, to initialize the driver. On Linux, I'm still working on getting the driver package (spacenavd) installed on the lab computers, but when it is, go into a terminal window, change to the directory you unzipped `proteins.zip` to, go to the `spnav` folder, and type `./configure` and then `make` to compile it. You'll also need to recompile the protein code, (in Matlab again) as

```
mex -DLINUX_INPUT proteins.cpp -lglut -lGL -lGLU -lpthread -Ispnav -Lspnav -lspnav -lX11
```

and it should work. Press Enter to toggle between whether you're pulling on an atom or re-selecting atoms. Shift-enter selects multiple atoms. Press "g" when the 3D cursor (the spinning octahedron) is visible to glue atoms; shift-"g" glues multiple atoms. Make sure that the wire coming out of the 3D mouse is pointing "into the screen" from your perspective.

Next week we'll start looking at fancier ways to manipulate the protein (and examine data from it), but we'll get to that later.

Have fun!