# Human Body Positions

## Programming Assignment #1
## CS253 Spring 2014
## Due 6:00pm Wednesday, February 3rd, 2016

### Motivation

This semester you will be analyzing sequences of human body positions, as detected by the Microsoft Kinect II sensor. Although this sensor is currently used mostly as an input device for gaming consoles, it is potentially much more powerful. Being able to read people's gestures and body postures has the potential to fundamentally change the way people and computers interact.

For the first assignment, you will write a program that reads in a sequence of body poses and draws them to the screen. The good news is that you don't have to write the graphics. You will be supplied with two C++ classes written by the instructor. One is a simple Point3D class that stores 3D coordinates, i.e. (x, y, z) points. As provided, this class is minimal, and as the semester goes on you may want to add new data fields and/or methods to it. The version provided is sufficient for the first assignment, however (although feel free to modify it). The second class is called PoseDisplay. It displays body positions to the screen so that you can see them, and also stores a file of the poses that have been drawn as a record that we can grade. You should not modify the PoseDisplay class.

Each body pose contains the 3D positions of 25 body parts. The first point is the position of the base of the spine, the second point is the middle of the spine, and so on[1]. The PoseDisplay class takes a body position as a vector of 25 Point3Ds. Note that the order is important. The input file is an ASCII file with 75 floating point numbers per row, separated by white space. As such, each row contains the (x, y, z) positions of 25 points, or one body position. People move over time, so each row is a different time step (separated by approximately 1/30th of a second).

### Task

For the first assignment, you will write a program that takes two command line arguments. The first is the name of the input file, as described above. The second is the name of the output file; pass this name to the constructor of the PoseDisplay class. Your task is to read in and display the body poses in order. As you

[1] You don't need to know what every point corresponds to, but if you are curious you can find the complete mapping at https://msdn.microsoft.com/en-us/library/microsoft.kinect.jointtype.aspx.

do this, you will see what looks like a human stick figure moving on your screen. The same poses will also be written to the output file, which is what we will grade.

Pedagogically, this is almost a warm-up exercise. As a student in this class, you are expected to know Java, and to have been exposed to C. You should be able to read formatted data in either of these languages. But C++ is different. You need to get used to calling the compiler, splitting your code in .h and .cpp files, using #include statements, and other details of the C++ language. This assignment is an opportunity to get over these hurdles.

This assignment is also a chance to get used to new expectations. We are giving you one test input file so that you know what the input format looks like, but we are only giving you one. It is up to you to generate additional test cases. This is part of programming. You should assume that we will make up and test examples of every boundary case we can think of. A program that works on the supplied input but fails on all the test cases may earn a grade of zero.

Moreover, never trust a user or a file. The input file is supposed to contain 75 numbers per row. But it might not. It might contain 26, or 24, or the letter 'a'. You might not be able to read the file. All kinds of illegal input are possible. If the input is not legal, you program should print an error message to std::cerr (not std:cout) and return the value -1 from main. We will test for error cases, and if your program crashes or fails to print an error message (with the word "error" in it) or fails to return -1, you will not get credit for that error test case.

## Submitting your program

You will submit your program through the class web site, as demonstrated in the first recitation. You should submit a single tar file, and this file should contain all the source files of your program, but no object or executable files. To grade your programs, the GTAs will untar your file in a clean directory, compile your program, and run it on test files. In particular, they will compile it using the following linux command:

```
g++ -Wall -g -std=c++11 -I. *.cpp -lX11 -o pa1
```

If it does not compile on the department Linux machines using the command above, you will receive no points for the assignment. Among other things, this means that your non-header source files should use the .cpp extension, not .ccc or some other convention.

## Hints

1. Follow the specifications above carefully. If you are unsure of part of a specification, ask the instructor for clarification.
2. Test your program on the department linux machines before submitting it if you develop it anywhere else. Any machine you develop on must have X11 for the PoseDisplay class to compile. Also, if you develop on your own machines, you are responsible for your own backups. "My machine crashed" is not a valid excuse.
3. Un-tar your submission in a clean directory and test it before you submit it. If you have forgotten to include a file in your tar file (a common mistake), this will catch it.

4. Keep your main program short. Create classes, and put most of your code in the methods of these classes.