INFO6015 – Animation 1: Final Exam (c)

Thursday, December 10th, 2012, 9:00 AM to 9:00 PM

Instructor: Michael Feeney

The exam format:

- The questions are ***not*** of equal weight. There are three (3) pages with seven (7) questions.
- CLEARLY indicate which answer goes to which question. My suggestion is that you place each answer in its own folder, named "Question_01", "Question_02" and so on (or something equally clear). Another option is to create a Visual Studio solution and add a number of projects – one per question – to it. Please don't combine questions.
- PLEASE clean all the "extra" files from your solutions before submitting. You can manually delete these, or use the "CleanStupidVisualStudioFiles.bat" program that's contained in almost every in class example.
- The intent of the coding questions is for you to demonstrate that you can code an answer. Simply uploading an existing solution (i.e. an old project, or a solution from class) demonstrates that you can upload a file, not that you can code anything.
- Place any written answers into a Word, RTF, or text file. Again, *clearly* indicate which question you are answering.
- For applications: If it doesn't build and run; it's like you didn't answer it. I'll correct trivial, obvious problems, but you need to be sure that it compiles and/or runs.
- This is an "open book" style exam, but keep in mind that you are supposed to be doing this exam completely on your own (with *absolutely no communication or collaboration*):
    - **There is no possible way that the basic structure of your code and/or the approach you take would be similar. This would indicate collaboration, in other words "plagiarism" or "cheating."**
    - **I will be aggressively looking out for this kind of thing; it is a clear violation of the college plagiarism policy.**
    - Please refer to Policy 2-G-04 on Fanshawe Online or in the Student Handbook for more information.
- There may be a number of versions of the exam.

Questions:

Create a class that has the following functionality:

1. (10 marks) Has a method, called “Initialize” which takes a vector of strings as a parameter. Each string represents a separate BVH motion capture file. The method should return true if is possible to load all the files, or false if any one of the files fails to load. Write code to test this method.

2. (20 marks) Has a method called “GetJointNames” that takes a string, representing the name of a BVH file, and returns a vector of strings, listing all the joint names as listed in that BVH file.

    - If there is an error, an empty vector (of strings) must be returned
    - The BVH file must have been one that was included in the Initialize method call, otherwise there is an error
    - The order of the returned vector doesn’t matter (i.e. the joint names don’t have to be in an particular order, but must include all the joint names)

3. (40 marks) Create a method called “GetJointAngle” that takes a string (the joint name) and an int (the frame number), and returns a class containing all the joint information (both offset and/or angle) at that frame.

    - The return class contain six float values (xOffset, yOffset, zOffset, xAngle, yAngle, zAngle) that contain the values contained in BVH file. You must take into account that the BVH file may list these in a different order (z first, for instance).
    - The return class must also list the channel name *as listed in the BVH file*. In other words, not only must the z rotation angle value be returned, we also need to know that it’s called “zRotation” (or whatever) inside the file. There also has to be a “sensible” way to associate the value of an axis with the name.
    - There should also be a variable (i.e. property) in the class called “bIsValid”. If the joint name and/or frame number isn’t valid, this should be “false.”

4. (30 marks) Override the method in question 3) so that the frame number is passed as a float, and a LERP angle value is returned. You can assume that the animation will only go in the “forward” direction here.

5. (30 marks) Further override the method in question 3) by adding two strings, each representing one of the BHV files from question 1). Your method should still take a string to indicate the joint name and a *float* to indicate the desired number. This method should return the LERP values at that frame, also LERPed between the two animations.

    - Since the number of frames will almost certainly be different between animations, assume that the animations are being continuously looped. For example: Assume BVH1 has 25 frames and BVH2 has 80 frames. If you request frame 30 (which is past the "end" of BVH1), assume that BHV1 has "looped" and return frame 5 of BVH1 (i.e. 30 – 25 = 5).
    - Note that, since the animations are being "looped," it is impossible to return an error based on the frame number; you can only return an error if you are requesting a "bad" joint name.
    - Assume the animations will only go in the "forward" direction.

6. (10 marks) Why it is generally a Good Idea (or why is your life as a programmer easier) when you omit scaling in a scene graph?

7. (10 marks) What's the advantage of using quaternions to represent angles, rather than just using Euler angles?

That's it.