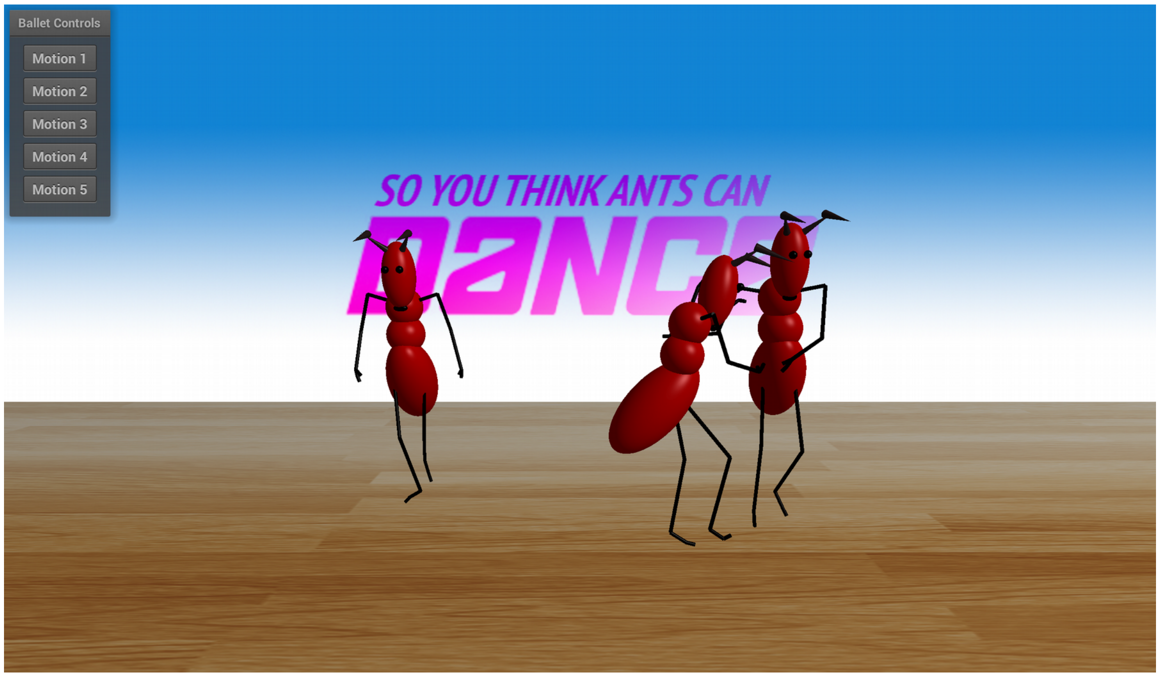
CSCI 4611: Programming Interactive Graphics and Games

Assignment 4: So You Think Ants Can Dance!

Handed out: Thursday, 10/25

Due: Wed, 11/7 11:59pm



Introduction

Animated characters are an important part of computer games and other interactive graphics. In this assignment you will learn how to animate computer graphics characters using data from motion capture systems. You will be working with data from the Carnegie Mellon motion capture database (http://mocap.cs.cmu.edu/), a great resource of free “mocap” data. The data will be formatted as text files, but with your programming and math skills, you will bring this data to life on the dance floor!

The CMU motion capture data is typical of all the skeleton-based motion capture data you’ll find in today’s games and movies. So, gaining some experience with this type of animation is one of the most important goals of the assignment. There are several important learning goals.

Learning Goals:

In completing this assignment, your goal should be to learn:

- How transformations can be composed in a hierarchy to create a scene graph and/or, in this case, an animated character within a scene.
- How transformations can be used to scale, rotate, and translate basic shapes (unit cubes, spheres, cylinders, and cones) into more complex scenes and characters (e.g., an ant!).
- How mocap data can be used and manipulated in multiple ways to create different types of animations. For example:
    - How to create a looping animation that smoothly interpolates between the beginning of the motion clip and the end to avoid any discontinuities.
    - How to overlay new motion clips onto a character at runtime, for example, in a game making your character punch or kick when you press a button, or in our case, perform one of a series of cool ballet moves.
- How to read and extend some fairly sophisticated C++ computer graphics code.

The amount of code you will need to write to complete this assignment is less than in the previous assignments. The support code provides quite a bit of infrastructure to deal with reading and playing back the mocap data. So, the key challenges will come in reading through and understanding the existing code as well as really thinking about the code that you do write. You will probably need to work out some of the math on paper before sitting down at the keyboard to program.

Background on CMU Mocap Data

The CMU Mocap database contains 2,605 different motions, most recorded at 120Hz, but some recorded at 60Hz or other speeds. These motions range from the simple (person walking straight forward), to the complicated (directing traffic), to the silly (someone doing the “I’m a little teapot” dance).

The motions in the CMU database use skeletons specified in .asf files and separate motions specified in .amc files. The .asf files specify bone names, directions, lengths, and the skeleton hierarchy for one specific human subject who came to the mocap lab. That person likely performed several motions during the capture session, so there is typically a 1 asf file to many amc files relationship. The subjects are numbered (e.g., subject #50) and the skeleton files are named accordingly (e.g., “50.asf”). Motion files names start with the subject ID, then have an underscore, then the number of the motion (e.g., “50_01.amc” is the first motion captured for subject 50). The support code comes with the data files we used in our solution to the assignment, but it can be fun to swap in other motions from the CMU database, and you are encouraged to experiment with this by downloading other asf and corresponding amc files from http://mocap.cs.cmu.edu/.

The amc and asf files are actually from the Acclaim Motion Capture data format, and if you are interested, you can read more about the details of the format here: http://graphics.cs.cmu.edu/nsp/course/cs229/info/ACCLAIMdef.html

Requirements and Grading Rubric

Working from the support code, which is significant for this assignment, you will be required to write the code to meet the following specs:

1. Draw one (or more) animated characters that perform a motion in a continuous loop. The support code starts you down this path by loading mocap data for the male and female parts of a salsa dance.
2. Draw one (or more) animated characters that loop through a small motion clip when “at rest” and seamlessly transition to perform new motions when the “Motion 1”, “Motion 2”, …, “Motion 5” buttons are clicked, similar to how a character in a game would jump or punch or kick whenever you press a button on your controller.
3. We suggesting starting #1 and #2 with a very simple character, like a stick figure, the final requirement is to make this character more interesting. Use transformation matrices to construct a more interesting character out of scaled spheres, cubes, cones, cylinders, or other simple shapes you can draw with the QuickShapes class.

To accomplish these tasks, you will need to add code to the places marked TODO in the AnimatedCharacter class and in the DanceApp class. You will probably not need to add any new classes to the project in this assignment.

Work in the “C” range will:

Focus on three tasks within the DrawBonesRecursive() function in animated_character.cc:

- Extend the AnimatedCharacter::DrawBonesRecursive() function to define the correct "current transformation matrix" to use for drawing each bone.
- Correctly set the "child_root_transform" before the recursive call at the end of the function in order to draw all of the bones in the skeleton.

If you do these things, then you should be rewarded by seeing a pair a figures with a coordinate frame drawn at each joint of their bodies dancing the salsa!

Work in the "B" range will:

Continue extending DrawBonesRecursive():

- Draw at least a stick figure (e.g., line segment or cylinder) representation for the each bone so that your dancing coordinate frames now turn into a dancing stick figure.

Create good motion clips for the ballet-dancing ant inside dance_app.cc:

- Follow the pattern demonstrated with the "ballet_special1_" example to define 4 additional special motions, load the motion clips from amc files, trim and uninteresting frames from the front and back of the clips, and then play them when the "Motion X" buttons in the GUI are pressed.
- Visually verify that the motion smoothly transitions from the ballet dancer's base loop motion into all of these special motions whenever a new motion is played.

If you do these things, then you should be rewarded by seeing a pair a stick figures dancing the salsa on the right hand side of the screen and an interactive stick figure that you can control on the left side of the screen.

Work in the "A" range will:

Return to the AnimatedCharacter::DrawBonesRecursive() function and extend it to draw an ant (or some other interesting character) rather than just a stick figure.

- Use QuickShapes to draw simple geometries like spheres, cubes, cylinders, cones, etc.
- Use transformation matrices (Matrix4s) to apply scales, translations, and rotations to the model matrix you use to draw each shape in order for it to create an interesting looking character. For example, notice how the "butts" of the ants that we have created are drawn using spheres that are scaled to create ellipsoids

and then rotated by about 45 degrees. When this shape is drawn with the transformation matrix for the lower back “bone”, it makes a pretty good ant.

- Adjust colors and use several geometries to make a convincing character. You are free to make the same ant character that we did, but you can also create your own character, just make sure that your character includes a few examples of applying transformation matrices before drawing simple shapes so that we can be sure that you have learned that skill.

Useful Math

Refer to the notes in class in the next couple of weeks. We will be talking about rendering hierarchical models and about blending poses together via linear interpolation and how to interpolate angles smoothly.

Above and Beyond

All the assignments in the course will include great opportunities for students to go beyond the requirements of the assignment and do cool extra work. We don’t offer any extra credit for this work — if you’re going beyond the assignment, then chances are you are already kicking butt in the class. However, we do offer a chance to show off… While grading the assignments the TAs will identify the best 4 or 5 examples of people doing cool stuff with computer graphics. After each assignment, the selected students will get a chance to demonstrate their programs to the class!

There are great opportunities for extra work in this assignment. You can add more characters or change the characters and put them in a different scene. You can also pick different motion clips to use from the CMU database. All of these can lead to interesting wizardly aesthetical changes! For the more technically inclined wizards, you might consider turning the ballet character into a character that can be controlled with the mouse and keyboard rather than just buttons on the screen. And/or, try to make a character than can walk around the screen using mocap data but rather than following a pre-defined path, make it go wherever you command it with keyboard or mouse input. This is what you find in many computer games, and you will have the knowledge to do this at the end of this assignment.

Support Code

As in past assignments, you will need to download the support code for assignment 4. This can be done with the following commands:

```
cd repo-[your x500]
git pull upstream support-code
```

You should now see a4-dance and a4-dance-docs in your repo-[your x500]/dev directory.

Handing It In

To hand in your code, check in to the master branch of your Github repository by the deadline. Any commits past the deadline will be assessed according to the late penalties described in the syllabus.

As always, include a README file that describes the completed portions of the rubric for this assignment. Be sure to include details of any standing bugs or wizard-work.