CPSC 426: Computer Animation
Assignment 3 (25%) **Due Fri, April 4, 2014, 11:59pm**.
This last assignment consists of two parts.
Part one: The final coding project, described in question 1, can be done individually or in groups of two or three.
Part two: This consists of questions 2, 3, and 4, and can also be done individually or in groups of two or three. If done individually, you need only do one of these questions. If done as a group, you should do two of the three questions. My own suggestion is to consider working as a group, as you will learn more.

1. (15 points) Final Coding Project

   Develop a project of your own choosing, to be completed by Friday April 2. We will use the last day of class (Monday April 7) to show off some of the results. Submit your project by emailing a URL to the instructor `van@cs.ubc.ca`. The URL can point to web pages, a zip file, or whatever is necessary to describe your project. You will also have a chance to demonstrate and discuss your work in person, so focus on the final result rather than writing a long report. Design your project to be scalable, i.e., begin modestly and then extend your work, so that you will always have something to show for your work. Some example ideas:

   - Develop your own idea!
   - Build a 3D geometric model of an animated motion and have this printed by a 3D printing service such as Shapeways. The animated motion can be of any object, e.g., ball, block, person, etc., and come from any source, e.g., procedural animation, hand animation, motion capture data. motion capture.
   - Develop software that helps explain and demonstrate quaternions.
   - Develop software that helps explain and demonstrate the principles behind cubic parametric curves.
   - Create a procedurally-animated character using some of the "animation texture" ideas described in the paper "Real Time Responsive Animation with Personality".
   - Implement a simple physics-based particle system or rigid body simulation system. We will be discussing the basic methodology in class.
   - Use freely available modeling, rendering, and animation tools (or simply OpenGL) in order to develop your own animated short film. If using OpenGL, there exist a variety of tools that will capture and record the window output as a video.
   - Using the OBJ file reader from assignment 2, develop a basic demonstration of character skinning with linear blend skinning.
   - Implement a model of flocking birds, using the ideas in the paper "Flocks, herds and schools: A distributed behavioral model."
   - Elaborate on one of the past assignments.

   Submit an initial one-page description of your ideas and team members by Wed March 19, in class. This initial description is worth 4 of the 15 marks.

2. (5 points) The Uncanny Valley

   The film "The Polar Express" is often cited as an example of a film that was adversely impacted because of the "uncanny valley". Was this related to the rendering, the motion, or both? What is your personal opinion regarding the human likeness of its characters? Do film critics believe that this has impacted its success? How might it have been alleviated? List the web pages or other material that you used to help form your opinion. Keep your answers to two pages or less.

3. (5 points) Topics in Computer Animation Research

   Recent advances in computer animation are published as research papers in conferences that include ACM SIGGRAPH, ACM SIGGRAPH ASIA, and the Symposium on Computer Animation (SCA), as well as journals such as the ACM Transactions on Graphics. The web site `http://kesen.realtimerendering.com` provides a good starting point for exploring the conferences papers.

   Select and read an animation research paper that has been published in 2010 or later, on a topic that you find to be interesting. Write a 2-page summary that responds to the following questions.

   What problem is it solving? What is the proposed solution? How are the method or ideas evaluated? Can the method be implemented from the ideas described in the paper? What are the limitations? Comment on at least two issues or features that are not discussed in the paper – these could be limitations, inadequate documentation, or possible directions for future work.

4. (5 points) Creating a Motion Graph

   In this question you will design a motion graph from some of the many motions that can be found in the CMU motion capture library (`mocap.cs.cmu.edu`). You will be provided with a program called `MocapPlayer` which reads a motion graph as specified using the `play.txt` file. The specification consists of a set of motion clips and a sequence for playing these motion clips. You should also submit a visual depiction of motion graph as an image. This can be an image created using your favorite drawing package or even simply a photo of a hand-drawn sketch of the motion graph.

   The file `play.txt` allows you to load a skeleton, load multiple motion files, identify a set of motion clips within the motion data, and to play an ordered set of these motion clips. The following is an example `play.txt` file:

```
# load skeleton
skeleton data/05.asf

# load motions
motion data/07_01.amc
motion data/05_02.amc
motion data/05_03.amc

# clip <startFrame> <endFrame> <characterLabel> <stringLabel>
# Note: endFrame is clamped to not exceed the max # of frames available,
clip 0   99999 C all
clip 0      50 A clipA
clip 450   500 B clipB

# designate the list of clips to be played
playlist CAAABBBABAB
```

   In the CMU library, the motion files are associated with the particular skeleton that they were captured with. Because the skeletons share the same structure, i.e., joints are numbered and named in the same way, it is possible to play back a given motion using another skeleton. `MocapPlayer` simply plays the joint angle data on whatever skeleton is currently loaded. However, the resulting motion might exhibit some artifacts. A more general solution to this problem is to process the motion data to fit the new skeleton, which is known as *motion retargeting*, but `MocapPlayer` does not do this.

   All frames of motion data are concatentated into a single dedicated chunk of memory, the 'frame buffer'. The number of frames and the their placement into the frame buffer is output to the console window as the motion files are read in. Motion clips can then be identified within the frame buffer. Each motion `clip` is defined by a start frame, an end frame, a single character label, and a longer text name. Motion clips act as convenient references to designated subsequences of frames in the frame buffer. The

`playlist` specifies a particular sequence of motion clips that is to be played in a loop. Comment lines begin with the `#` character.

When `MocapPlayer` is started, it will read the current `play.txt` file and then begin playing the designated `playlist` in a loop. The spacebar can be used to toggle between pause and play. If this is not working, ensure that the current window focus is on the playback window. Typing '<' and '>' will rewind and advance by 100 frames in the play list. The text displayed on screen gives the name of the current clip and the number of the current frame, i.e., its position in the frame buffer.

`MocapPlayer` does not yet implement any blending between the end of one motion clip and the start of the next. As such, you will not be able to achieve perfectly seamless transitions between motion clips. However, you will be able avoid large visual discontinuities by carefully choosing your motion clips and controlling which transitions are allowed.

Suggested Plan

- Browse the motion capture library to see which types of motions are available. View all *Subjects*, i.e., motions grouped by capture subject, provides the most concise overview of what is available. Identify at least three subject areas that are interesting to you. One of these should include some kind of locomotion.
- Download the motions. Identify the subset of files that contain motions of interest to you by viewing using `MocapPlayer` and an appropriate `play.txt` file. Note that the player is limited to loading 100,000 frames at a time. Setup a `play.txt` file that includes all the relevant motions and identify the specific motion clips of interest to you. I suggest having motion clips that represent a minimum of 8 different actions, and that some (but less than half) of these actions involve locomotion.
- Sketch an initial motion graph design. Given your motion capture clips and the transitions that you wish to support, create a play sequence that tests all these transitions. Note which transitions work well, i.e., only small discontinuities at the transition points, and which ones do not.
- Iterate on your design. You can adjust the start and end points of clips in order to achieve better transitions. You can look for motion files that contain an example of a transition that you would like to support but that currently does not work well (in a game studio this would usually result in a new motion capture request). You can change your motion graph design to disallow certain transitions. Some motions may exhibit foot-skate or other artifacts because the capture subject had significantly different dimensions than the skeleton you have selected. In this case you might choose motions from subjects which are better matched with the skeleton you have chosen.
- Create a final motion sequence that demonstrates all the transitions in your motion graph. This will be the final `play.txt` file that you submit.

What to submit:

Create a `README.txt` file that includes your names, student numbers, and login IDs, and includes a few paragraphs that describe what challenges you faced when creating your motion graph and what tools would help speed the design process.

Create a folder called 'a3' under your cs426 directory and put your play.txt, `README.txt`, the motion graph image, and all your motion capture data files there. Do not use further sub-directories. The assignment should be handed in using: `handin cs426 a3`

Hand in printed copies of your `README.txt` and `play.txt` files. Also, in the space below, give your login ID, and sketch out the graph structure of your particular motion graph. Label your graph structure using the same names as in your `play.txt` file. Use meaningful names.

Submission checklist:
README.txt, play.txt, skeleton files, motion files, and motion graph image