Assignment 3: Skeleton animation

1 Objective

The main objective is to create a complete system for keyframe animation of articulated objects. You will need to create keyframes to define movements of the objects using interactive controls and generate in-between motion using proper interpolation. The main goals of this assignment are:

1. Implement a 2D keyframe animation system to animate a pure translation motion.
2. Understand arc-length parameterization and implement speed control for keyframe animation.
3. Implement interactive controls for skeletal joints.
4. Implement keyframe animation for a hierarchical skeleton model.
5. Generate animation using quaternions for smooth interpolation of keyframes.

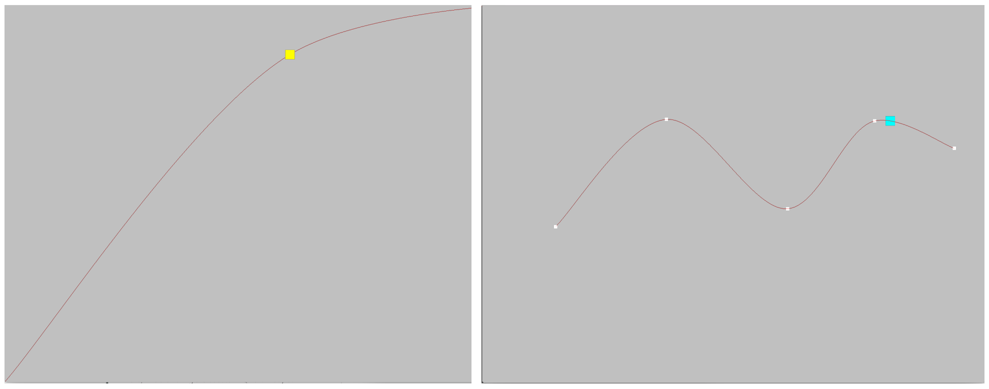

Figure 1: Stage 1. Animating a square, where the speed curve is on the left and the trajectory is on the right.

2 Tasks

2.1 Stage 1 (total 60%)

1. Core (40%)
   The core part of this stage is to implement a 2D keyframe system to animate a pure translation motion.

   - Let the user add keyframes (more than two). For that, open a window that represents a plane that would show the trajectory (a curve) of a moving object. The user would click on the window to add points to define the curve.
   - Display the trajectory of the movement (connecting keyframes) using the Catmull-Rom (CR) splines. Two additional (dummy) keyframes may be required, one at the very beginning of the motion, one at the very end (no special treatment of the first and last dummy keyframe segments would be needed).

2. Extension (20%)

   - Display a speed curve in an additional window and let the user edit the speed curve (use a CR spline to define the speed curve as well. Make sure the speed curve is a monotonically increasing function; hint: use x-axis as the time and y-axis as the arc-length).
   - Run a rigid object (e.g. teapot, bunny, etc) animation with the correct path and speed as designed by the user.
   - Implementation details:
     - Make sure your code could be used for 3D as well by just changing the type of vectors you manipulate.
     - For the re-parameterization, subdivide the curve into 200 segments, for example.

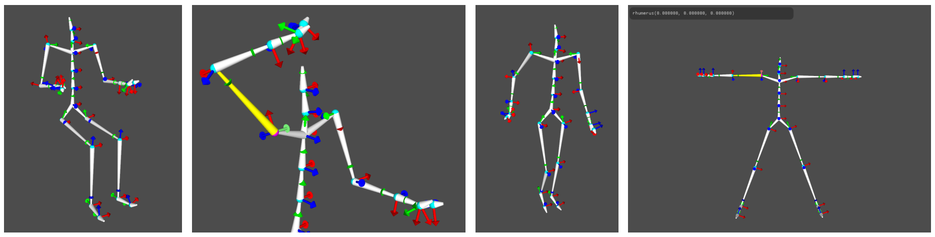
Figure 2: Stage 2. Animating a skeleton.

2.2 Stage 2 (25%)

Write a program that will read skeleton files, allow the user to make arbitrary poses using interactive controls, and then animate the skeleton using saved poses as the key frames.

Note that doing the animation is a lot more work than any of the other tasks in this stage. If you choose not to use quaternions for joint rotation in this stage, then more complex animations may end up looking poor, and you'll lose points. The program will have the following controls.

- On start up, it will read and render an ASF file passed to it on the command line (using the rules from Assignment 2). You should use 'priman.asf' for the assignment. However, it will work with any ASF file from the CMU database once you are finished.
- If the program is run with the wrong number of arguments, the program should print what the correct arguments are to the console and terminate.
- "Ctrl + Left mouse" click on a deselected joint select the given joint, deselecting any other currently selected joints.
- A selected joint should have its color changed (originally a cyan sphere) to magenta, its bone color changed to yellow, and highlight its x-axis (e.g., by setting the 0 parts of its RGB color to 0.5)
- "Ctrl + Left mouse" click on a selected joint should deselect it. This should change the colors back, as originally specified in assignment 2.
- Pushing 'x' while a joint is selected should change the currently selected axis. Repeated pushing of the 'x' key should cycle through the axes in the order $x \to y \to z \to x$. The currently selected axis should be highlighted, while all others retain their standard colors. Only one axis can be selected at any one time.
- While a joint is selected, holding right click and dragging the mouse should rotate the joint around the selected axis.
- While a joint is selected, the joint name and the current rotation around each axis would be displayed at the bottom of the window, e.g., rfemur (30,60,90).
- If the 's' key is pressed, it should save the current pose to a file. This should prompt the user for a file name.
- File names submitted for the assignment should have a key frame number and a pose number; e.g 0pos1.pos
- Define a configuration file to describe your keyframes and extend the program to accept the configuration file as an additional parameter to the program. The file should contain a list of the frame numbers and corresponding pose files to act as keyframes. These files are then used to define animations by interpolating between given poses to make a keyframe animation.
- When the 'a' key is pressed, the animation specified by the configuration file is played. This should interpolate all the joints between the given poses. Create basic player controls: 'Play', 'Pause', 'Stop', 'Rewind', and 'Fast Forward'. Note that this is the biggest part of Stage 2.

After implementing the program, complete the following tasks for creating a key frame animation of the primitive man.

- Using your interactive controls, construct and save three key frames for a walking animation of the primitive man.
- Create a configuration file that defines a single animation for walking in terms of the three saved poses.
- These poses act as keyframes for the resulting animation.
- Create at least three additional poses. The first two poses are sitting down and standing. The third pose (and any additional poses) is up to you to decide (e.g. punching or jumping).
- Use these files to make the animation file that will play the key frame animation that has the following sequence: sitting, standing, walking pose 1, walking pose 2, walking pose 3, standing, and your custom pose.

2.3 Advance Stage (15%)

Combine Stage 1 and 2 to generate an animation that follows a user-specified speed.

3 Marking

Total Marks: 100.

Complete the README provided. You should detail what you have completed as well as any additional modifications and instructions on how to run your code. Submit your source code (without the build dir) and necessary files. Please verify that your submission runs on the ECS machines. If it does not compile or execute, you will get 0 marks. Submit a video capture that demonstrates all the implemented features. Also, show and explain in the video important parts of the code. Any feature that is not demonstrated would be treated as missing. The video should be no more than 5 minutes.