DIS Java VRML Entity State Protocol Data Unit Smoothing

by

Scott David Heller, LT USN
Naval Postgraduate School

Reference:

Background:

Currently when an update occurs to the location and/or orientation of an entity there is a discontinuous jump from position \( P_0 \) to position \( P_1 \) and from orientation, \( O_0 \) to \( O_1 \). This discontinuous motion can be visually disconcerting. To improve the quality of the visual experience smoothing can be used. Smoothing will sacrifice some accuracy of position and orientation at the time of EspduUpdate receipt, but will greatly reduce the discontinuous motion due to a EspduUpdate rate that is much slower than the screen update rate.

Methodology:

This smoothing will be accomplished using the following formula as specified in the IEEE Standard for Distributed Interactive Simulation -- Communication Service and Profiles:

\[
X_i = X_0 + \left( X_f - X_0 \right) \left( \frac{i}{p} \right)
\]

Where \( X_i \) is the current DR position,
\( X_0 \) is the previous update position dead reckoned to the current time,
\( X_f \) is the most recent update position dead reckoned to the current time,
\( i \) is \( (t_f - t_0) \), or the current time minus the time of current update.
\( p = \) (average update delta)
Smoothing is best thought of as the result of the linear interpolation between the deadreckoned position of the previous entity state pdu and the deadreckoned position of the current entity state pdu. Both Espdu(s) are deadreckoned to the current time.

In order to perform this linear interpolation a ground truth must be established for the time of the previous update and the time of the current update. The time used is when EspduTransform is called. This introduces an error by the amount of latency in the network and in the local application, however for the purposes of visual display these errors can be considered negligible.

In order to minimize the effects of variable display frame rates and pdu update rates an average of each is used to calculate the time between pdu updates. Originally this time was calculated by a call to the browser to get the frame rate. In this iteration of EspduTransform, the local system clock is used via a Timer class that can be reset to zero and returns the number of seconds since reset as a float.

The smoothing algorithm can be turned off via a call to toggleSmoothing(). Smoothing of acceleration may be enabled by setting SMOOTH_LINVELOCITY to true in the source code and recompiling. A set function was not provided since this feature provides no noticeable benefits. The reason smoothing the velocity does not improve the representation is that the smoothing algorithm already smooths between the previous and current dead reckoning tracks. Each dead reckoning track is calculated using the appropriate velocities, so the effects of these velocities are already accounted for. I left in the boolean switch just in case someone wanted to experiment with it a bit.

The next page shows EspduTransform class variables and methods at the time of this work.