13 December 1998

DIS Java VRML Entity State Protocol Data Unit Smoothing

by

Scott David Heller, LT USN
Naval Postgraduate School

Reference:

(a) P1278.1 IEEE Standard for Distributed Interactive Simulation -- Communication Service and Profiles, September 1995. Appendix B sections 3.8 and 3.9.

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)

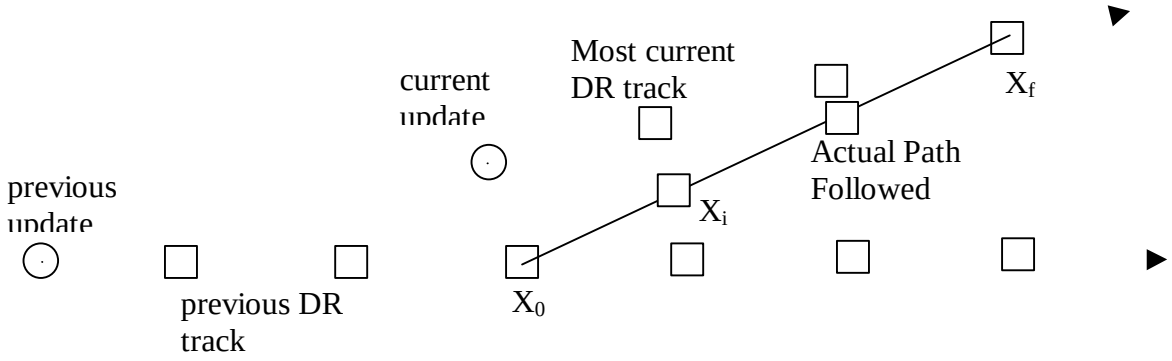


Figure 1

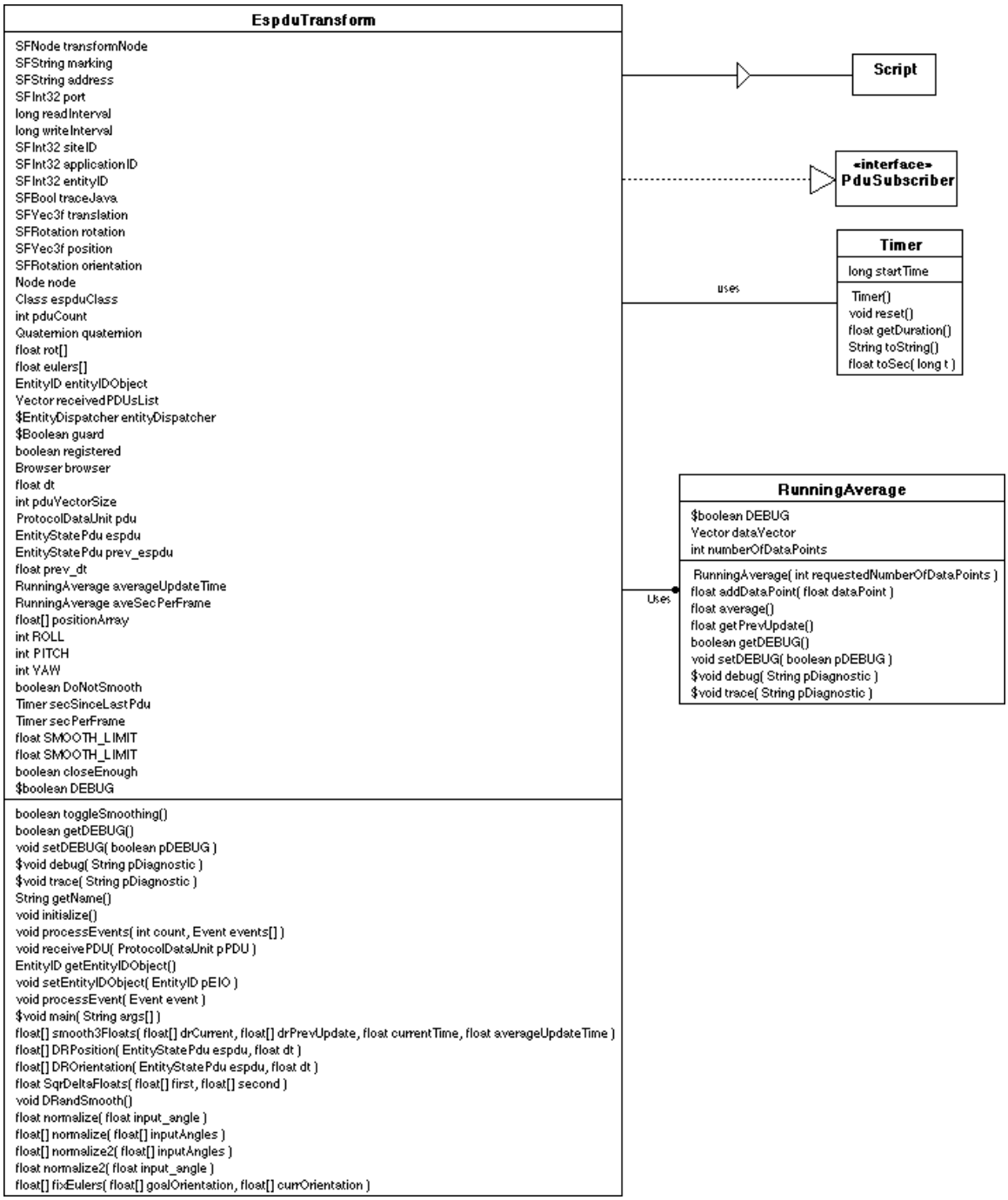
Smoothing is best thought of as the result of the linear interpolation between the deadreckoned positon of the previous entity state pdu and the deadreckoned postion of the current entity state pdu. Both Espdu(s) are deadreckoned to the current time.

In order to perform this linear interpolation a ground truth must be esablished 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 amout of latentacy in the network and in the local applicaion, 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 noticable benifts. 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.


EspduTransform
SFNode transformNode
SFString marking
SFString address
SFInt32 port
long readInterval
long writeInterval
SFInt32 siteID
SFInt32 applicationID
SFInt32 entityID
SFBool traceJava
SFVec3f translation
SFRotation rotation
SFVec3f position
SFRotation orientation
Node node
Class espduClass
int pduCount
Quaternion quaternion
float rot[]
float eulers[]
EntityID entityIDObject
Vector receivedPDUsList
$EntityDispatcher entityDispatcher
$Boolean guard
boolean registered
Browser browser
float dt
int pduVectorSize
ProtocolDataUnit pdu
EntityStatePdu espdu
EntityStatePdu prev_espdu
float prev_dt
RunningAverage averageUpdateTime
RunningAverage aveSecPerFrame
float[] positionArray
int ROLL
int PITCH
int YAW
boolean DoNotSmooth
Timer secSinceLastPdu
Timer secPerFrame
float SMOOTH_LIMIT
float SMOOTH_LIMIT
boolean closeEnough
$boolean DEBUG
boolean toggleSmoothing()
boolean getDEBUG()
void setDEBUG(boolean pDEBUG)
$void debug(String pDiagnostic)
$void trace(String pDiagnostic)
String getName()
void initialize()
void processEvents(int count, Event events[])
void receivePDU(ProtocolDataUnit pPDU)
EntityID getEntityIDObject()
void setEntityIDObject(EntityID pEID)
void processEvent(Event event)
$void main(String args[])
float[] smooth3Floats(float[] drCurrent, float[] drPrevUpdate, float currentTime, float averageUpdateTime)
float[] DRPosition(EntityStatePdu espdu, float dt)
float[] DROrientation(EntityStatePdu espdu, float dt)
float SqrDeltaFloats(float[] first, float[] second)
void DRandSmooth()
float normalize(float input_angle)
float[] normalize(float[] inputAngles)
float[] normalize2(float[] inputAngles)
float normalize2(float input_angle)
float[] fixEulers(float[] goalOrientation, float[] currOrientation)
Script
«interface»
PduSubscriber
Timer
long startTime
Timer()
void reset()
float getDuration()
String toString()
float toSec(long t)
uses
RunningAverage
$boolean DEBUG
Vector dataVector
int numberOfDataPoints
RunningAverage(int requestedNumberOfDataPoints)
float addDataPoint(float dataPoint)
float average()
float getPrevUpdate()
boolean getDEBUG()
void setDEBUG(boolean pDEBUG)
$void debug(String pDiagnostic)
$void trace(String pDiagnostic)
Uses