Department of Mathematics
Faculty of Mechanical Engineering
Slovak University of Technology in Bratislava

2nd International Conference
APLIMAT 2003

TRACING OF DYNAMIC OBJECTS IN DISTRIBUTED INTERACTIVE SIMULATION SYSTEMS

ORŁOWSKI Tomasz, (Poland), WISZNIEWSKI Bogdan, (Poland)

Abstract. Distributed interactive simulation systems require integration of several areas of computer science and applied mathematics to enable each individual simulation object to visualize effectively dynamic states of other objects. Objects are unpredictable, i.e., controlled by their local operators, and are remote, i.e., must rely on some transmission media to visualize dynamic scene from their local perspectives. The paper reports on the ongoing project aimed at developing a distributed simulator for mission training of helicopters and tanks interacting in a 3D dynamic scene. Based on a generic physical model of a material point some effective communication and visualization methods are proposed and illustrated in the paper with the working prototype.

1 Introduction

A *Distributed Interactive Simulation* system (*DIS*) combines several independently running simulation programs, real objects and human operators into one coherent computational process. Such a process is unpredictable, usually has no algorithmic representation, and because of humans and real objects participating in the computation, all events that may occur must be handled in real-time. Unpredictability is implied by the inherent non-determinism of simulation participants and dynamism of their internal states. Participating simulation objects, which are usually located at various geographically separated sites are changing their states accordingly to asynchronous events occurring in their local environment and messages irregularly flowing from other objects.

If the message flow is too intensive *performance* problems of the communication infrastructure may cause delay or loss of information, affect individual object behaviour and in consequence spoil the entire simulation process. In order to avoid that, simulation objects limit the volume of outgoing messages to the absolute minimum, sending to other objects only the updates on their most important state changes. However, if the messages are sent too rarely, *reliability* of the simulation may be affected; this is because some objects may require updates on specific states of other objects - important from the local object perspective but not sent, as the reporting objects may consider them unimportant.

Performance and reliability problems of DIS are addressed by researching various models of state related information exchange. Typical models are based on either *dead reckoning* or *relevance filtering* [1]. Dead reckoning models are based on the assumption that states of the reporting objects may be estimated with some function of time with fixed parameters, e.g. speed or distance in a straight line accelerated movement has just two constants characterizing it, namely the initial speed and the acceleration rate. Reporting object has then to send out messages on its state changes only when one of these parameters changes. Observing objects are receiving this information only when they need to correct their local estimates of these parameters. The relevance filtering models are more advanced and are able to estimate remote object states when relations between them are more complex, e.g. some states of the remote object are important, while others are not. Reporting objects are sending out state updates important from their point of view, while observing objects are selecting only the specific information of interest, relevant from their local perspective. Owing to this the observing object visualization system may achieve higher *fidelity*, i.e., reporting objects that are supposed to be at a greater distance in a virtual scene may be displayed at a lower level of graphical detail than objects that are supposed to be closer to the observer.

The model presented throughout the rest of this paper has been developed for a distributed mission training application of helicopters and tanks interacting in a 3D dynamic scene, started at the Technical University of Gdansk in 2001. Applications of this class are rapidly gaining popularity, not just because of their attractive, often highly entertaining interface, but first of all as a cheap alternative to costly, and sometimes impossible to implement in a real world, realistic exercises involving large groups of users, expert teams and equipment operators in crisis situations, massive rescue operations, environmental catastrophes, etc. [2,3,4]. The key issue in any such application is the proper selection of an object behavioural model to enable effective filtering of messages and assure high quality of simulation with regard to performance, reliability and fidelity.

2 Simulation object model

In general objects moving in a virtual scene are unpredictable. Below we introduce a set of parameters that may be used to predict their behaviour according to the laws of dynamics. Let us first distinguish three classes of non-deterministic events in a DIS system regarding moving objects:

- *decision event* initiated by the object's human operator, e.g., a new manoeuvre; such an event can be only partially predicted by using a predefined simulation scenario, by monitoring of some representative physical parameters of an object, or by analysing past behaviour of the operator stored in a log;
- *interactive event* initiated by other participants of a simulation experiment and implied because of their presence in a scene, e.g. a collision of objects; they can be predicted or detected based on the current scene context analysis and reported to other participants;
- *random event*, which is not foreseeable at all, like the loss of messages, incorrect order of messages, or network malfunction; instead of predicting them the observer should try to detect them to react on time; usually upon their occurrence extrapolation of the observed object movement is applied until correct messages start to come again.

The model of a mobile object dynamics we use in our project is three-dimensional, general enough to simulate any realistic movement, and suitable for any class of mobile airborne, land or naval object that may appear in a dynamic scene. A set of parameters describing state of an

object, reported to an observing object to allow the latter to correctly visualise and extrapolate movements of the former is introduced in Figure 1. Directions of speed, angular acceleration and moments are defined using the right hand's rule.

Parameter	*Description*
$\vec{L} =< L_x, L_y, L_z >$	Position
$\vec{O} =< \theta_x, \theta_y, \theta_z >$	Orientation
$\vec{V} =< V_x, V_y, V_z >$	Linear velocity
$\vec{a} =< a_x, a_y, a_z >$	Linear acceleration
$\vec{\omega} =< \omega_x, \omega_y, \omega_z >$	Angular velocity
$\vec{\gamma} =< \gamma_x, \gamma_y, \gamma_z >$	Angular acceleration
$\vec{F} =< F_x, F_y, F_z >$	Force
$\vec{M} =< M_x, M_y, M_z >$	Moment

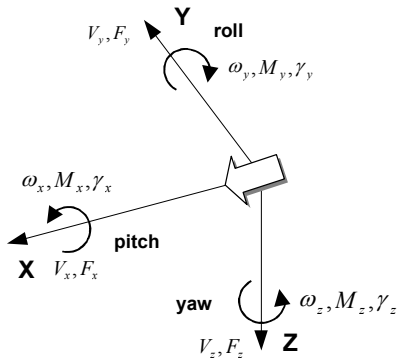

Figure 1. State of a moving object: (a) parameters, (b) axes and local coordinates

The model of a mobile object dynamics is used by all participants of the simulation experiment to visualize one another's behaviour in a global dynamic scene. More data is sent out by object less computation is required at the receiving side to perform visualization. On the other hand, frequent updates reported to remote objects increases the network load. Therefore less data should be sent by reporting objects and receiving objects must be able to extrapolate movements of remote (observed) objects – correctly and just in time. In order to reduce the volume of data being sent out by any simulation object let us first consider the following dependencies between parameters describing the state of a moving object shown in Figure 1:

1. If reporting object *W* sends updates at the rate not less than the framing rate of a graphical system at the side of a visualising object *T* only position $\vec{L}_w$ and orientation $\vec{O}_w$ of *W* are needed by *T* to correctly visualize *W*. This approach is straightforward but bandwidth consuming, so in many cases it is not acceptable.
2. If at a given moment object *T* does not have current values of $\vec{L}_w$ or $\vec{O}_w$, but has their values sent by *W* earlier it:
   a) must possess recent values of $\vec{V}_w$ or $\vec{\omega}_w$, otherwise it must assume that neither $\vec{V}_w$ nor $\vec{\omega}_w$ has changed from the last reported values; in such a case only changes of $\vec{V}_w$ or $\vec{\omega}_w$ will be transmitted;
   b) alternatively, *T* has to extrapolate value of $\vec{V}_w$ or $\vec{\omega}_w$ based on at least two recent values of $\vec{L}_w$ or $\vec{O}_w$;
3. If at a given moment object *T* does not have current values of information about $\vec{V}_w$ or $\vec{\omega}_w$, but has their values sent by *W* it:
   a) must possess recent values of $\vec{a}_w$ or $\vec{\gamma}_w$, otherwise it must assume that neither $\vec{a}_w$ nor $\vec{\gamma}_w$ has changed from the last reported values; in such a case only changes of $\vec{a}_w$ or $\vec{\gamma}_w$ will be transmitted;
   b) alternatively, *T* has to extrapolate value of $\vec{a}_w$ or $\vec{\gamma}_w$, based on at least two recent values of $\vec{V}_w$ or $\vec{\omega}_w$;

In a properly initiated simulation system, where each receiver (observer) has once got full information about each participant, for objects associated with decision events (manoeuvres initiated by their human operators) only changes in their acceleration shall be transmitted. For interactive events (collisions) other parameters shall be transmitted as well. The worse

situation is when messages are lost (random events). State updates of some objects are not delivered, so their visualisation may be discontinued. Sending back retransmission requests (like in lose less network protocols) is not acceptable, due to additional network load and in consequence a yet more delayed update. Instead of that the receiving object shall perform "smoothing", i.e., until the next successful update a few "extra" generated consecutive frames may attempt to asymptotically reach the final correct one.

Several important relations between parameters listed in Figure 1 are defined with equations of the classic Newtonian model of a material point dynamics. First of all, linear acceleration of a moving object can be determined based on the resultant force applied to the object. Similarly, angular acceleration can be determined based on the resultant moment of the object. Linear and angular velocities can be determined then by integrating the respective accelerations. Integration of the respective time dependent parameters is numerical, based on the rule of trapezium, known also as the modified Euler's method or "predictor-corrector" approximation:

$$P_n = P_{n-1} + \left(\frac{\Delta P_{n-1} + \Delta P_n}{2} \right) \Delta t \tag{1}$$

where P is a parameter of interest, P_n is a new value of P, P_{n-1} is the last value of P, ΔP_{n-1} is the last change of the value of P, ΔP_n is the predicted change of the value of P, and Δt is the integration step.

Given current linear velocity $\vec{V} =< V_x, V_y, V_z >$ and the last reported position $\vec{L}_{n-1} =< L_x^{n-1}, L_y^{n-1}, L_z^{n-1} >$ of the observed remote object , the observing object may calculate the new position of the former in one step as:

$$\vec{L}_n = \vec{L}_{n-1} + \vec{V} \Delta t \tag{2}$$

Note that according to Figure 1b angular velocity of a moving object is defined with regard to its local coordinate system. Because updating of an object position requires adjustment of its linear velocities to global scene coordinates (one global cube), updating of object orientation needs similar adjustment of its angular velocity. There are several methods for determining a moving object orientation in a global cube based on its angular velocity; one is the Euler's method, another one is the *quaternion method*. The latter, however, enables rotations in a more elegant manner [1].

Let a system of local coordinates of a given object W correspond to the global cube coordinates with a single rotation D around the global axes with angles A, B and C. These four parameters define orientation of an object in the global system of coordinates, and the transformation of local position $\vec{L}$ of object W to its global position $\vec{L}_w$ can be expressed as:

$$\vec{L}_w = \begin{bmatrix} 1 - 2\sin^2 A \sin^2 \frac{1}{2} D & \begin{matrix} 2\cos A \cos B \sin^2 \frac{1}{2} D - \\ - 2\cos C \sin \frac{1}{2} D \cos \frac{1}{2} D \end{matrix} & \begin{matrix} 2\cos A \cos C \sin^2 \frac{1}{2} D + \\ + 2\cos B \sin \frac{1}{2} D \cos \frac{1}{2} D \end{matrix} \\ \begin{matrix} 2\cos A \cos B \sin^2 \frac{1}{2} D + \\ + 2\cos C \cos \frac{1}{2} D \sin \frac{1}{2} D \end{matrix} & 1 - 2\sin^2 \frac{1}{2} D \sin^2 B & \begin{matrix} 2\cos B \cos C \sin^2 \frac{1}{2} D - \\ - 2\cos A \cos \frac{1}{2} D \sin \frac{1}{2} D \end{matrix} \\ \begin{matrix} 2\cos A \cos C \sin^2 \frac{1}{2} D - \\ - 2\cos B \sin \frac{1}{2} D \cos \frac{1}{2} D \end{matrix} & \begin{matrix} 2\cos B \cos C \sin^2 \frac{1}{2} D - \\ - 2\cos A \sin \frac{1}{2} D \cos \frac{1}{2} D \end{matrix} & 1 - 2\sin^2 C \sin^2 \frac{1}{2} D \end{bmatrix} \vec{L} \tag{3}$$

The transformation matrix above uses four angles and may look more complicated than the usual Euler's turn matrix; however, by using the following substitutions:

$$q_0 = \cos\frac{1}{2}D \qquad q_1 = \cos A \sin\frac{1}{2}D \qquad q_2 = \cos B \sin\frac{1}{2}D \qquad q_3 = \cos C \sin\frac{1}{2}D \tag{4}$$

it can be simplified into (normalised quaternion form):

$$\vec{L}_w = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_0q_2 + q_1q_3) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \vec{L} \tag{5}$$

The normalized quaternion form is next used for updating location and orientation. In the case when angles A, B, C and D are not initially known, values q_1, q_2, q_3 and q_4 can be get from the initial cosine matrix for the angles. For quaternion updating based on angular velocities the following equations shall be used:

$$\dot{q}_0 = -\frac{1}{2}(q_1\omega_x + q_2\omega_y + q_3\omega_z) \quad \dot{q}_1 = \frac{1}{2}(q_0\omega_x + q_2\omega_z - q_3\omega_y)$$
$$\dot{q}_2 = \frac{1}{2}(q_0\omega_y + q_3\omega_x - q_1\omega_z) \quad \dot{q}_3 = \frac{1}{2}(q_0\omega_z + q_1\omega_y - q_2\omega_x) \tag{6}$$

By normalizing a quaternion and assuming that integration step is less than 1, the above equations change to:

$$\dot{q}_0 = -\frac{1}{2}(q_1\omega_x + q_2\omega_y + q_3\omega_z) + \lambda q_0 \quad \dot{q}_1 = \frac{1}{2}(q_0\omega_x + q_2\omega_z - q_3\omega_y) + \lambda q_1$$
$$\dot{q}_2 = \frac{1}{2}(q_0\omega_y + q_3\omega_x - q_1\omega_z) + \lambda q_2 \quad \dot{q}_3 = \frac{1}{2}(q_0\omega_z + q_1\omega_y - q_2\omega_x) + \lambda q_3 \tag{7}$$

where λ is a correction factor of the numerical integration error, given as:

$$\lambda = 1 - (q_0^2 + q_1^2 + q_2^2 + q_3^2) \tag{8}$$

Alternatively to using factor λ, period normalisation of the resulting quaternion can be performed. Many simulation supporting computations need Euler's angles (what does not require creating a turn matrix). Because *pitch* and *yaw* angles are normally within the range of (-π/2,+π/2), the values of their cosine functions are positive and the respective angles θ_x, θ_y, θ_z may be calculated as:

$$\theta_x = \arctan(-\frac{m_{12}}{m_{11}}) \qquad \theta_y = \arcsin(m_{13}) \qquad \theta_z = \arctan(-\frac{m_{23}}{m_{33}}) \tag{9}$$

where m_{ij}, i, j=1..3, are the respective elements of the transformation matrix in formula (5).

3 Dynamic state sampling

Having defined a model for determining a reporting object position in a global cube, let us now relate it to the local visualization capabilities of the observing object. Let assume first that two visualization parameters are given a'priori for the simulation system: maximum *resolution* and *refreshing frequency* of all visualization devices used by observers. These parameters determine the upper bound of the visualization capability across the system. In many cases frequency of updating state of visualised objects may be less than the refreshing frequency. The lower the quality of a visualising device, the less frequent the updates of the reporting objects state should be. Similarly, lower velocity of a visualised object, or a greater distance from the observer will also require lower refreshing frequency. A question arises –

how many times lapses (samples) may be skipped before the visualised object will change its state so much that it can be noticed by the observer?

The minimum realistic change is *one pixel*. Therefore, for the reporting object the following information about the observer's display is needed: horizontal ($X_{\max}$), and vertical ($Y_{\max}$) screen *resolution* at the observer's side, horizontal (α_x) and vertical (α_y) *angles of view* at the observer's side, and distance h between the observed (reporting) and observing (receiving) objects (see Figure 2).

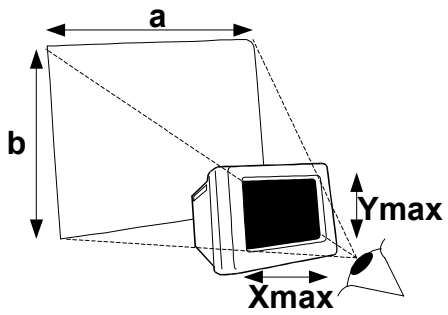

Figure 2. A view base pyramid

With these parameters it is possible to calculate the size (a,b) of a view base pyramid seen by the observer from distance h:

$$a = 2h\,\mathrm{tg}(\frac{1}{2}\alpha_x) = 2h\frac{(1-\cos(\alpha_x))}{\sin(\alpha_x)} \qquad b = 2h\,\mathrm{tg}(\frac{1}{2}\alpha_y) = 2h\frac{(1-\cos(\alpha_y))}{\sin(\alpha_y)} \tag{10}$$

Then one can calculate how big observed object area could be contained in one pixel of the image rendered at the observer's side:

$$1p_x = \frac{a}{X_{\max}} \qquad 1p_y = \frac{b}{Y_{\max}} \tag{11}$$

The formulas above clearly indicate that if the rendered image of a moving object is smaller than $1p_x$ and $1p_y$, it is pointless to update object's state because it will not be noticed by the observer. The screen refreshing frequency at the observer side f_{obs} can be also used as an additional constraint; so if the frequency of sending updates is f we get:

$$f_{pxpy} = \frac{1}{T_{pxpy}} \le f \le f_{obs} \tag{12}$$

where T_{pxpy} is the time of scope transgression $1p_x$ or $1p_y$ by the visualized object.

If $f_{pxpy} > f_{obs}$ then the reporting object should send messages with just the f_{obs} frequency. It is particularly important to offline observers (which may refresh their screens once in a couple of seconds).

Figure 3 illustrates the relevance filtering method based on the model presented above. It allows for reducing the volume of messages in two ways. Outgoing messages are filtered out before letting them to propagate throughout the underlying Run-Time Infrastructure (RTI) of the DIS platform. Only messages that report on manoeuvring (decision events) and on detected collisions (interactive events) are allowed into the RTI. Incoming messages are further filtered out before receiving them by the observing object, according to the local perspective of the visualized dynamic scene and the local graphical system capability.

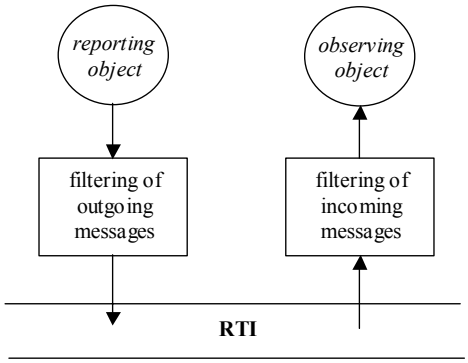

Figure 3. Relevance filtering model

4 Conclusions

The relevance filtering model proposed in the paper is flexible and independent on implementation details of individual simulators (simulation objects) connected to the system. This shall be a principal feature of any DIS application, as simulators connected to the system are usually stand-alone applications with different hardware and software components. The only implementation cost is conversion of messages between the format used locally by a simulator and a format used by RTI. In the project mentioned before we use the common standard introduced by the High Level Architecture (HLA) platform [6].

In the project we currently have been concentrating on the improvement of predicting specific manoeuvres of the reporting object, based on context analysis and its past behaviour stored in a log file. There are several methods of gathering information needed in such a case:

- *physical features of a simulation object*; one can eliminate some possible manoeuvres of a simulation object with its present state and its physical features, by considering the ability to accelerate, turning rate, maximum/minimum velocity, inertia, ability to shoot, changes of locations of object parts, etc.
- *predefined simulation scenario*; potential intentions of a given object may be read from a script specifying its target to be reached, manoeuvres planned and orders received.
- *past behaviour of simulation objects*; observer may read out in what manner object executes some specific manoeuvres (i.e., individual manner of doing manoeuvres by the human operator of the object) and in the case of similar behaviour in the future the manoeuvre can be predicted.

Our initial experiments with the presented model implementation indicate that simulation scenarios involving several (up to eight) flying objects and ground vehicles give satisfactory results with regard to performance, reliability and fidelity in a modest lab settlement, including 16 Pentium-3 class workstations connected with a Fast Ethernet cable.

References

1. BASSIOUNI, M.A., CHIN, M.H., LOPER, M., GARNSEY, M., WILLIAMNS, J.: Performance and Reliability Analysis of Relevance Filtering for Scalable Distributed Interactive Simulation. *ACM Trans. on Modeling and Computer Simulation*, vol. 7, no. 3, pp.293-331, 1997.

2. MUSSE, S.R.: Crowd modeling in collaborative virtual environments. *proc. ACM Symp. on Virtual Reality Software and Technology,* Taipei, Taiwan, pp. 115-123, 1998.
3. BUKOWSKI, R., SEQUIN, C.: Interactive simulation of fire in virtual building environments. *Proc. 24th Annual Conf. on Comp. Graphics and Interactive Techniques,* Los Angeles, USA, pp. 35-44, 1998.
4. KELIN, U., SCHULZE, T., STRASSBURGER, S.: Traffic simulation based on the High Level Architecture. *Proc. 1998 Winter Simulation Conf.*, Washington, USA, pp.1095-1104, 1998.
5. DAM, E., KOCH, M., LILLHOLM, M.: Quaternions, Interpolation and Animation. *Technical Report, University of Copenhagen*, Copenhagen, Denmark, 1998.
6. US DEPT. OF DEFENCE: High Level Architecture Interface Specification, Version 1.3, IEEE P1516.1, http://hla.dmso.mil, 1998

Current address

ORŁOWSKI Tomasz, Faculty of Electronics, Telecommunications and Informatics, Technical University of Gdansk, ul. Narutowicza 11/12, 80-952 Gdansk, Poland, phone: +48 58 347-1089
email: eaglet@eti.pg.gda.pl,

WISZNIEWSKI Bogdan, Faculty of Electronics, Telecommunications and Informatics, Technical University of Gdansk, ul. Narutowicza 11/12, 80-952 Gdansk, Poland, phone: +48 58 347-1089
email: bowisz@eti.pg.gda.pl