Paper: ASAT-13-MO-13

13th International Conference on
AEROSPACE SCIENCES & AVIATION TECHNOLOGY,
ASAT- 13, May 26 – 28, 2009, E-Mail: asat@mtc.edu.eg
Military Technical College, Kobry Elkobbah, Cairo, Egypt
Tel : +(202) 24025292 – 24036138, Fax: +(202) 22621908

Dead Reckoning in a Distributed Interactive Tank Gunner Simulator

A. F. Seleem[*], H. A.Hussein[**], I.Talkhan[***]

Abstract: Distributed Interactive Simulation (DIS) is a way that provides a method of communicating entity state and other information through Protocol Data Units (PDUs). In distributed systems, bandwidth consumption is a major problem and the most difficult task is to provide the players with a consistent view in any virtual world despite the network delays. In this paper we use an algorithm called Dead Reckoning (DR) algorithm that estimates future movements based on past behavior to reduce the data exchange through the network to overcome the problem of bandwidth consumption. The environment used is an interactive Tank Gunner Simulator (TGS) model which involves a highly modular collection of software packages (VEGA, CREATOR, and real-time application program). Finally, we examine the impact of DR prediction algorithm on the bandwidth consumption.

Keywords: Distributed Interactive Simulation (DIS), Dead Reckoning (DR), Virtual Environment (VE), VEGA, CREATOR, Application Program Interface (API).

1. Introduction

Interactive visual simulation through a computer network is an important tool that the military is using to prepare troops and their commanders for battlefield training. This simulation setup usually has two topologies either centralized systems (star topology) or distributed systems (mesh topology). For n users the star topology usually needs n messages broadcast on every update in the real-time which can be reduced by multi-casting to be only one message ; however it has the major problem of having a single point of failure (simulation server). To ameliorate the single point of failure problem, distributed simulation appeared to operate in a way that each node (computer) acts as a server and client for the player simultaneously. However for an n-players system each player sends a status update message to all n-1 players, that is why it is called mesh topology. When all players interact, a total of n(n-1) messages are communicated, which introduces a network bandwidth problem that needs to be solved.

DIS standard produced by IEEE [1] defines the exchange of data through Protocol Data Units (PDUs). DIS provides 27 types of PDU and the most commonly used ones are the Entity State PDU (ESPDU). It describes the physical state of an entity: entity's (x,y,z) position, velocity, acceleration, angular velocity, angular acceleration, orientation , and contains "articulation parameters", which handles details such as turret orientation on a tank Network traffic-reduction techniques usually fall into two broad categories: reduction connection- oriented

[*] Egyptian Armed Forces, alaa_el_deken@hotmail.com
[**] Egyptian Armed Forces, h_aly@excite.com
[***] Associate professor, Dept. of Computer Engineering, Cairo University, Giza, Egypt/italkhan@cu.edu.eg

(data aware) and connectionless-oriented (data independent). In our work, we focus on connection oriented (data aware) techniques because the data exchanges through the network is very important (describes the physical state of the entity: position, velocity, acceleration…) [2]. The data aware techniques are: Simple approach in which the originating entity determines the set of entities which may be interested in a particular piece of data, and replicates separate messages to each target. In DIS [3] a predictor needs to determine which set of entities are interested. This predictor is the Dead reckoning that estimates future movements based on past behavior, and is updated using infrequent broadcast messages from the process being modeled.

This paper is organized as follows. Section 2 explains the concept and mechanism of dead reckoning algorithm. In section 3, proposed simulation model is introduced. Section 4 describes the experimental results of the proposed simulation model with and without using Dead reckoning algorithm. Finally, the paper is concluded in section 5.

2. Dead Reckoning

Dead reckoning is an algorithm that is appropriate when simulation entities are slow-moving or when network bandwidth is relatively high [4].This algorithm is used to limit the rate at which simulated entities need to send their update entity attributes such as position and velocity to others in DIS. It is a way to decrease the amount of messages communicated among the participants.

In distributed systems with n players when a player moves it has to send a message to all n-1 players containing the new position. When Dead Reckoning is used, the other participants will use previous velocity information to extrapolate the next position of the participant X. This extrapolation operation is named dead-reckoning [5].

In Dead Reckoning, each player extrapolates its own position (ghost model) and compares it with the real position. If the difference between the real position and the extrapolated one is greater than a predefined maximum threshold, then it sends the real position and velocity to other participants, so that they can correct their copy of participant X's object as shown in Fig.1.

The threshold is an important parameter in the DR algorithm [6]. It is used to control the accuracy of extrapolation, and affect the number of entity state update packets generated. A small threshold makes DR algorithm generate state update packets at a higher frequency, but results in higher accuracy in the estimation of the entity's position. On the other hand, the DR algorithm using larger threshold value generates fewer update packets, but its accuracy is also lower. In order to maintain an adequate accuracy, the adaptive dead reckoning algorithm is used where the threshold is dynamically changed [7] according to the relative distances between simulation entities. To determine the levels of threshold, the area of interest (AOI) of an entity, is defined as a circle with a constant radius around the entity. The length of radius is usually defined according to the entity type and its function and weaponry. Hence, the AOI of all entities (either local or remote) can be easily determined by a simulator. In addition to AOI, an entity also has its sensitive region (SR). If one entity moves into another entity's SR, a collision will likely to happen. By using AOI and SR the distance between entities has four threshold levels which can be defined as level 1 being the smallest threshold and level 4 the largest. They are listed in Table.1 and illustrated in Figure.2. In the figure, the circle with a solid line represents the entity's AOI, and the circle with a dotted line represents the entity's

SR. When an entity A's AOI is not overlapped with any other entity's AOI, level 4 threshold will be used in entity A's extrapolation. Large errors are allowed in the extrapolation, and update packets will be sent at a low frequency. In this case, entity A's movement is not of interest to other entities, and extrapolation can be less accurate. When the entity A's AOI is overlapped with the entity B's AOI, the extrapolation of entity A's movement needs to be more accurate because interaction level becomes higher [7].

2.1 Dead Reckoning Implementation Methods

The following methods provide an overview of the different dead reckoning implementation methods [8] used in virtual environments (VEs): (1) Extrapolation equations, (2) Pre-Reckoning, (3) PHBDR (Position History Based Dead Reckoning), (4)Kalamn filter, Which are described below.

2.1.1 Extrapolation

Current extrapolation equations can be one of two groups [9]: (1) first order extrapolation equations which are a function of the previous packet’s position, velocity of the object, and the elapsed time, (2) second order extrapolation equations which is a function of the previous packet’s position, velocity, acceleration of the object, and the elapsed time, [10]. Extrapolation equations can be categorized into two interpolation steps: one-step formulas and multi-step formulas. One-step formulas only use the last state update packet to extrapolate an entity's Position, whereas multi-step formulas use the last two or more state update packets in the extrapolation which is shown in Table.2.

In Table2, $\boldsymbol{X_{t'}}$, $\boldsymbol{v_{t'}}$ and $\boldsymbol{a_{t'}}$ represent the position, velocity and acceleration of the entity respectively, as found in the last state update packet. Similarly, $\boldsymbol{X_{t''}}$, $\boldsymbol{v_{t''}}$ and $\boldsymbol{a_{t''}}$ are the position, velocity and acceleration of the second last state update packet. $\boldsymbol{\tau}$ is the elapse time from the last update. The formulas are used to extrapolate the position of the entity at time $\boldsymbol{t} = \boldsymbol{t_0} + \boldsymbol{\tau}$. [11]

2.1.2 Pre-Reckoning

The objective of the technique is to eliminate foreseeable error with a negligible increase in update packets. The algorithm as shown in Fig.3 seeks to anticipate the potential for a violation of the error threshold and issues an status update packet immediately rather than waiting for the threshold to be violated while this can result in update packets being generated unnecessarily. The pre-reckoning technique was introduced in [12] as a variation of the standard DR algorithm to decrease prediction errors without significantly increasing network traffic. Pre-reckoning is effective when entity makes sudden changes.

2.1.3 Position History Based Dead Reckoning (PHBDR)

This overcomes the objectionable discontinuities produced by DIS. Position History-Based Dead Reckoning is a technique designed by [13] to overcome some of the limitations of the basic DIS approach. DIS uses only the most recent information about an object's position to predict future positions. Position History-Based Dead Reckoning maintains a history of object positions, and uses curve-fitting techniques to predict future object positions. Higher order polynomial curves are fitted to the history of object positions and are used to estimate future positions. The same techniques are used to smoothly correct for error in past predictions, and produce continuous natural movement of objects.

2.1.4 Kalman filter

A Kalman filter [14] is a recursive procedure to estimate the states S_k of a discrete-time controlled process governed by the linear stochastic difference equation from a set of measured observations t_k. The mathematical model is shown in the following two Equations:

$$s_k = AS_{k-1} + W_{k-1} \quad (1)$$

$$t_k = HS_k + r_k \quad (2)$$

The NxN matrix A represent an state transition matrix, [11], W_k is an Nx1 process noise vector with $N(0, б^2_w)$, t_k is Mx1 measurement vector, H is MxM measurement matrix,and r_k is Mx1 measurement noise vector with $N(0, б^2_r)$. To estimate the process, Kalman filter uses a form of feedback control. The Kalman filter, [14] is a set of mathematical equations that dose not store all system history but keeps knowledge in the system states. Based on the above discussion, when simulation entities are slow-moving the standard dead reckoning algorithm appears to provide a good approximation to Entity position with the Extrapolation implementation method. Since tanks motion represents low scenario dynamics then we use Extrapolation method in our interactive tank gunner simulation model.

3. Proposed Simulation Model

The software components used in our proposed simulation model are: software visualization environment (VEGA), the modeling package (MultiGen Creator), and real-time application program (C++ program) as shown in Fig.4.

First, we use CREATOR package to design the 3-D models which we will use in the simulator scene (such as: land; terrain; tank models such as: M1A1, Mirkava, Tank after destruction; Gunner's primary sight reticule…etc) and save the CREATOR file with .flt file extension. After that, we import the .flt CREATOR files (3-D models) into VEGA package to built the simulator scene and determine its coordinate frame positions relative to the terrain position in 3-D world dimensions. The VEGA package is a visualization packet for the whole scene. Finally, we use a C++ program to control all the parameters of objects, terrain, weather, and whole scene in the simulation. These steps will be described in the following sections.

3.1 The Modeling Package

The Create Object function is used to create different objects in the visual scene as required. The 3-D model is built in our simulation either by 3-DMAX or MultiGen Creator as shown in Fig.5, [15] but the later is better because it presents small size models with a lot of details, which indicate better performance in visualization. The CreateObject is also used to set the initial position of the created object in all three axes and its orientations.

3.2 Software Visualization Environment (VEGA)

VEGA is MultiGen-Paradigm's premier software environment, [16] for the creation and deployment of real-time visual and audio simulation, sensor, virtual reality, and general visualization applications. By combining advanced simulation functionality, VEGA provides the basis for the most productive process for building, editing. VEGA is intended to be used in conjunction with LynX (graphical user interface) for defining and previewing VEGA applications as shown in Fig.6. Although VEGA contains all of the application programming interface, API necessary to create an application. The purpose of DIS protocol in VEGA is to allow players from multiple separate applications to be interconnected such that they share a common virtual world. Players are represented as network entities which communicate entity state information, weapons fire and detonation information, and other information with entities participating in the same simulation.

The dead reckoning algorithm used for Vega outbound entities is set in the Lynx Outbound Entity Information panel for a given object. The dead reckoning algorithm used for inbound entities is set in the ESPDU (entity sate PDU) sent by those entities.

3.3 Proposed Simulation Algorithm

This program controls the graphical scene, how the user moves through the scene, as well as a wide variety of other dynamic events during the scene. Tank driving, collision detection, laser range finder, firing, and special effects such as explosions are also included in this real-time application program which generally consists of a C or C++ code. The Tank simulator program flowchart can be described as follows as in Fig.7:

1-Define all constant parameters: BLENGTH, THRESHOLD PARAMETERS, RESTART_WAIT, FIRE_WAIT……..etc.
2-Start the program with initiating the DIS module to operate in SEND AND RECEIVE Mode.
3-If the observer (player) sees all entities that exist in the simulation then DIS is ready to work else reinitiate DIS for the entities that are not exist in the simulation scenarios.
4-Get the motion model (predefined path or actual path) of the player to move, and then apply Dead Reckoning (DR) Model for our player after defining all the threshold parameters in the simulation scenarios.
5-Compare between real path and predict path which is predicted by DR algorithm. If the difference is greater than predefined threshold, send update packet to all the entities in the simulation.
6-After the player gets his motion model the player gets the distance between his object and the target object by laser range finder function. If the distance is between 200 m and 4000 m (effective area for the gun), then wait for the FIRE-WAIT time to fire the target object (enemy tank).
7-If firing parameters (azimuth, elevation, ammunition types, gun rotation angle …etc) are true and firing is in the damaged area (circle) of the target object (enemy tank) then destroy the target object else repeat the steps to re-fire the target object for the second time.

4. Experiments and Results

4.1 Experiment setup

In our experiment we connect the simulation entities (computers) in a private network in a lab of 12 computers (entities). We run the c++ application program file in each simulation entity in the network which itself calls the .pdf file that is made by VEGA package. In the application program we initiate each entity to work in DIS mode (by DIS_INIT function) and ensure that each entity sees the other entities (self test) that are in the range of field of view (FOV) in the simulation scene.

Each entity gets its motion model for movement which we implement to work with Dead Reckoning algorithm or to work without Dead Reckoning algorithm. If the entity works *without Dead reckoning algorithm*, then we measure the average of traffic on the network in three setups. The first case, when the simulation scenario has only 3 entities (tanks) exchanging data at the same time through the network. The other two cases are with 6 and 12 entities respectively.

If the entity works *with Dead reckoning algorithm*, then we measure the average of traffic on the network in the pervious three cases. Then, we plot the average traffic using performance monitor (as DU Meter software) to measure the average traffic per node on the network (as in Fig.8). Also, in Fig.9 we plot the average traffic on 2, 4, 6,8,10 and 12 users.

4.2 Results

We examined the effect of the Dead Reckoning prediction algorithm in our interactive tank gunner simulator model. The values of average traffic in every node in the simulation model increase when increasing the number of nodes at any time and this happens if we use Dead Reckoning algorithm or not.

When number of nodes in the simulation model increases, average traffic increases, and the need of Dead Reckoning Algorithm increases. When the number of nodes is 12 the traffic when using Dead Reckoning is nearly half that without using Dead Reckoning. So, the Dead Reckoning algorithm is more effective when increasing the number of nodes in DIS.

5. Conclusion

we can see that the values of average traffic per node in the DIS model increase when increasing the number of nodes O(nodes^2) at any time and this happens if we use Dead Reckoning algorithm or not. So, the Dead Reckoning algorithm is more effective when increasing the number of nodes in the simulation without adversely increasing the network traffic.

6. Refrences

[1] IEEE Computer Society (1995). “Standard for Distributed Interactive Simulation-Application Protocols”, IEEE Std 1278.1-1995. New York, NY: Institute of Electrical and Electronics Engineers, Inc..

[2] Bassiouni, M.A., M.-H. Chiu, M. Loper, M.Garnsey, and J.Williams. 1997.Performance and Reliability Analysis of Relevance Filtering for Scalable Distributed Interactive Simulation. ACM Transactions on Modeling and Computer Simulation, 7 (3) (July): 293-331.

[3] VanHook, D. J., Calvin, J. O., and Fusco,“An Approach to DIS Scalability”, 11th Workshop on Standard for Inter-operability of Distributed Simulation, Vol. 2, pp.347-356, 1994.

[4] P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W.Evans, eds."Developing Interest Management Technique In Distributed Interactive Simulation Using Java", Proceedings of the 1999 Winter Simulation Conference, pp.518-523 .

[5] Kuo-Chi Lin. "Dead Reckoning and Distributed Interactive Simulation". In Proc. of SPIE Conference(AeroSense'95), Orlando Florida, April 1995.

[6] Lee, B-S, Cai, W., Turner, S, and L. Chen, “Adaptive Dead Reckoning Algorithms for Distributed Interactive Simulation”, International Journal of Simulation Systems Science, &Technology, Vo1. 1, 2003.

[7] W. Cai, F. B. S. Lee, and L. Chen, “An auto-adaptive dead reckoning algorithm for distributed interactive simulation,” in Proccedings of the 13th Workshop on Parallel and Distributed Simulation (PADS), 1999, pp. 82–89.

[8] Mcauliffe, M., Long, R., Liu, J., Lyou, W., and D. Nocera, “Testing the Implementation of DIS Dead Reckoning Algorithms”, Proc. 14th DIS Workshop, 94-14- 086,1994.

[9] Peter Ryan, "Effectiveness Of Dead Reckoning Algorithms In Advanced Distributed Simulation",Defence Science & Technology Organisation (DSTO) 2004, PO Box 4331, Melbourne, Victoria, 3001, Australia,2004.

[10] Aggarwal, S., Banavar, H., Khandelwal, A., Mukherjee, S., Rangarajan, " accuracy in dead-reckoning based distributed multi-player games". Proc. ACM SIGCOMM 2004 Workshops on Net-Games. Network and System Support for Games,2004.

[11] Corentin Durbach and Jean-Michel Fourneau. "Performance Evaluation of a Dead Reckoning Mechnism".In IEEE 2nd Int. Workshop on Distributed Interactive Simulation and Real-time Applications, pp.23-29, July 1998.

[12] Xiaoyu Zhang, Denis Graˇcanin, Thomas P. Duncan, " Evaluation of a Pre-Reckoning Algorithm for Distributed Virtual Environments" Proceedings of the Tenth International Conference on Parallel and Distributed Systems(ICPADS’04),IEEE COMPUTER SOCIETY,1521-9097,2004.

[13] Singhal, S.K., Cheriton, D.R." Exploiting Position History for Efficient Remote Rendering in Networked Virtual Reality". Tele-operators and Virtual Environments. vol. 4. no. 2,169-193, 1995.

[14] Seong-Whan Kim and Ki-Hong Ko, "Kalman Filter Based Dead Reckoning Algorithm for Minimizing Network Traffic Between Mobile Nodes in Wireless GRID", IFIP International Federation for Information Processing 2006, E.Shaet al. (Eds.): EUC 2006, LNCS 4096, pp. 62 – 170, 2006.

[15] http://www.paradigmsim.com/products/database/creator/index.html is the home page of MultiGen-paradigm's Creator product.

[16] http://www.paradigmsim.com/products/runtime/vega/index.html is the home page of MultiGen-paradigm's Vega product.

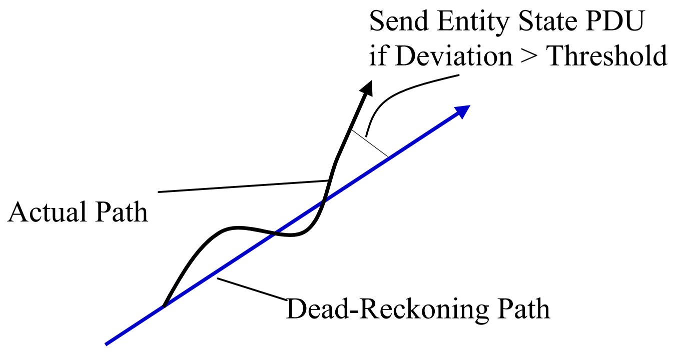


Fig. 1 Dead Reckoning algorithm

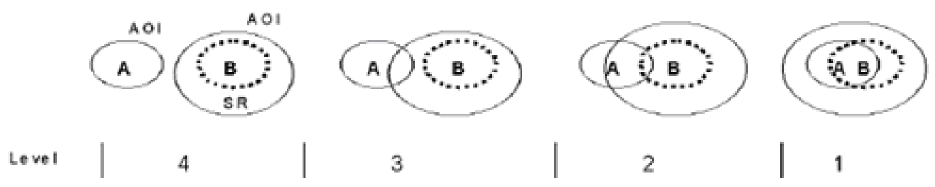


Fig. 2 Multi-level threshold levels

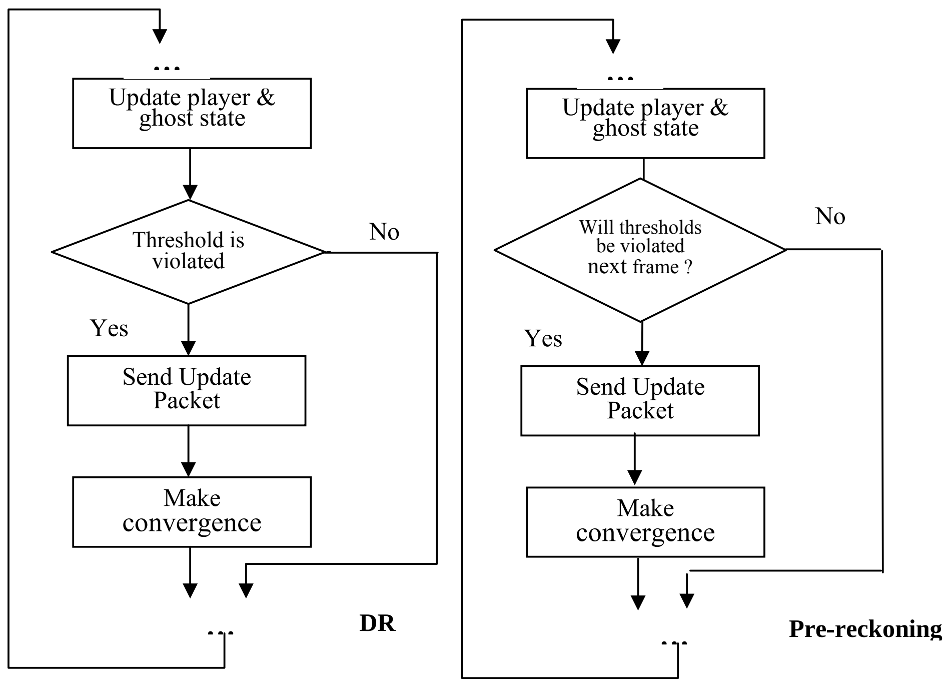


Fig. 3 Pre-Reckoning algorithm

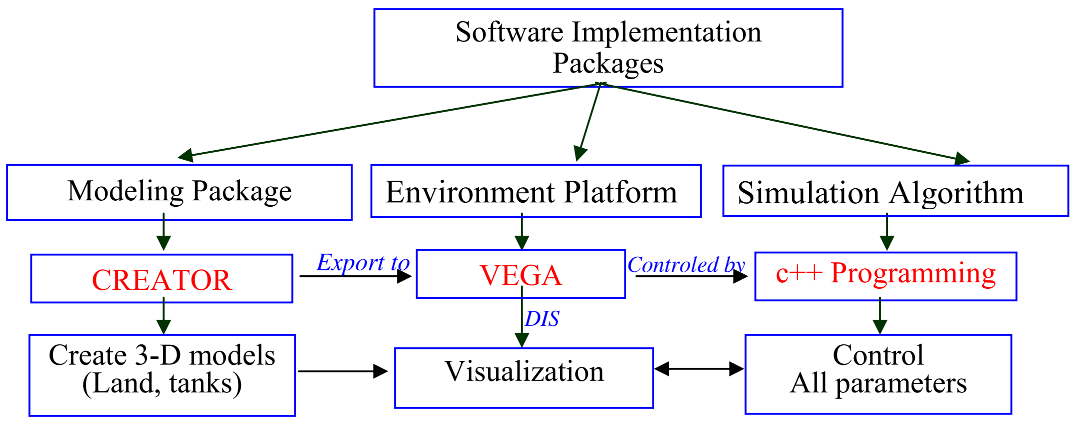
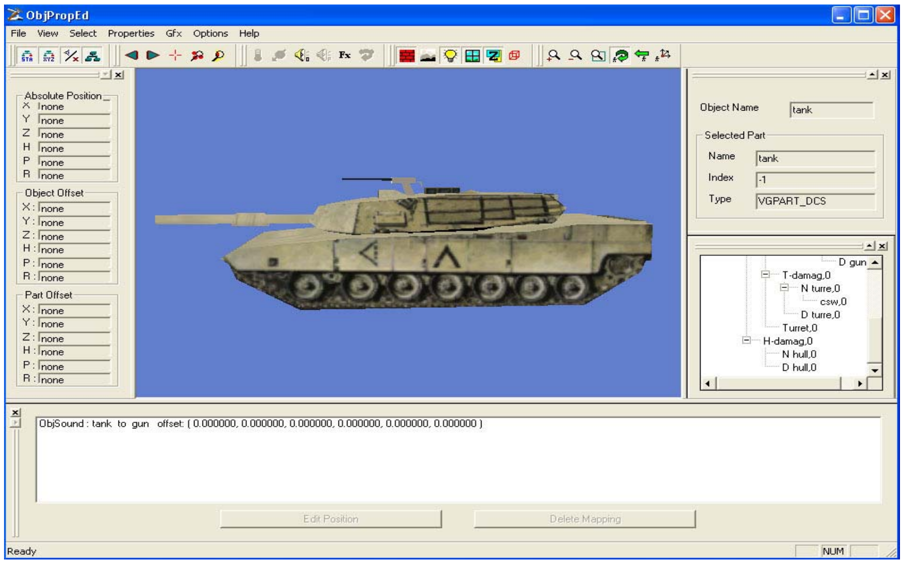


Fig. 4 Software Implementation Packages



Absolute Position
X none
Y none
Z none
H none
P none
R none

Object Offset
X : none
Y : none
Z : none
H : none
P : none
R : none

Part Offset
X : none
Y : none
Z : none
H : none
P : none
R : none

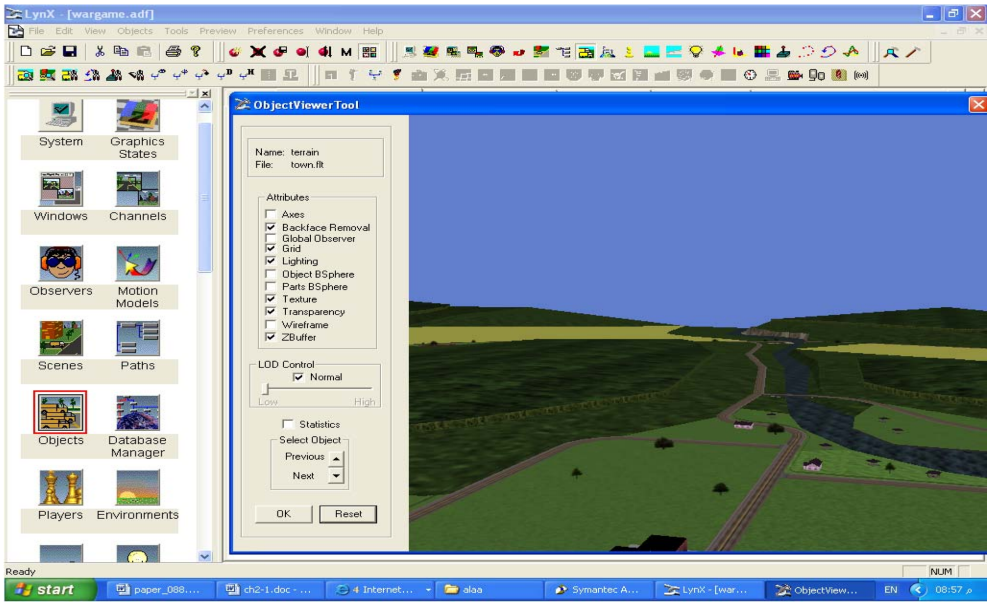
Object Name tank

Selected Part
Name tank
Index -1
Type VGPART_DCS

```
D gun
T-damag,0
  N turre,0
    csw,0
  D turre,0
Turret,0
H-damag,0
  N hull,0
  D hull,0
```

```
ObjSound : tank to gun offset: ( 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000 )
```

Fig. 5 Modeling package (CREATOR)



Name: terrain
File: town.flt

Attributes
- [ ] Axes
- [✓] Backface Removal
- [ ] Global Observer
- [✓] Grid
- [✓] Lighting
- [ ] Object BSphere
- [ ] Parts BSphere
- [✓] Texture
- [✓] Transparency
- [ ] Wireframe
- [✓] ZBuffer

LOD Control
- [✓] Normal

Low High

- [ ] Statistics

Select Object

Fig. 6 VEGA graphical user interface (Lynx)

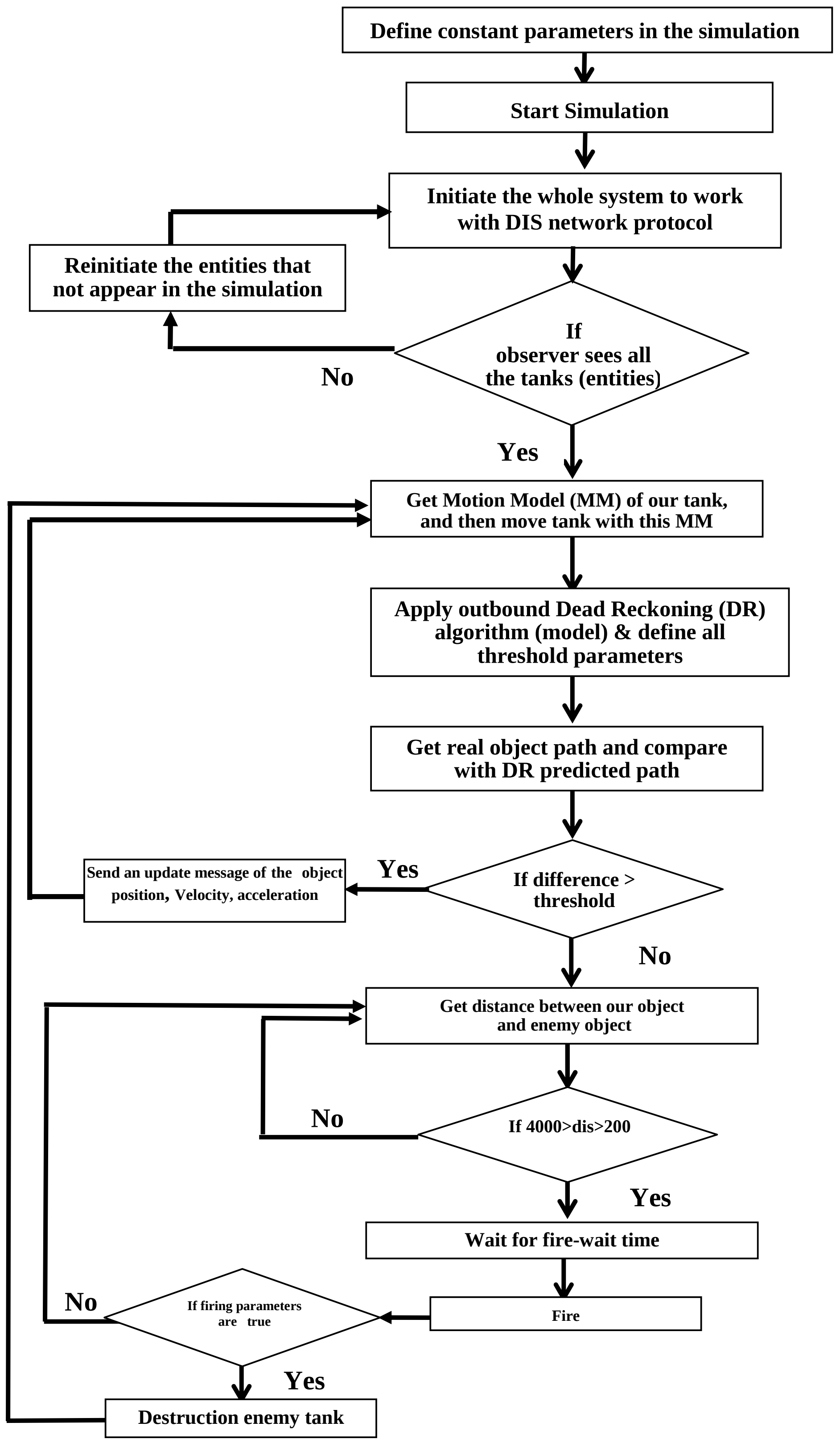

Define constant parameters in the simulation
Start Simulation
Initiate the whole system to work with DIS network protocol
Reinitiate the entities that not appear in the simulation
If observer sees all the tanks (entities)
No
Yes
Get Motion Model (MM) of our tank, and then move tank with this MM
Apply outbound Dead Reckoning (DR) algorithm (model) & define all threshold parameters
Get real object path and compare with DR predicted path
Send an update message of the object position, Velocity, acceleration
Yes
If difference > threshold
No
Get distance between our object and enemy object
No
If 4000>dis>200
Yes
Wait for fire-wait time
No
If firing parameters are true
Fire
Yes
Destruction enemy tank

Fig. 7 Proposed Simulation Algorithm applied on each node

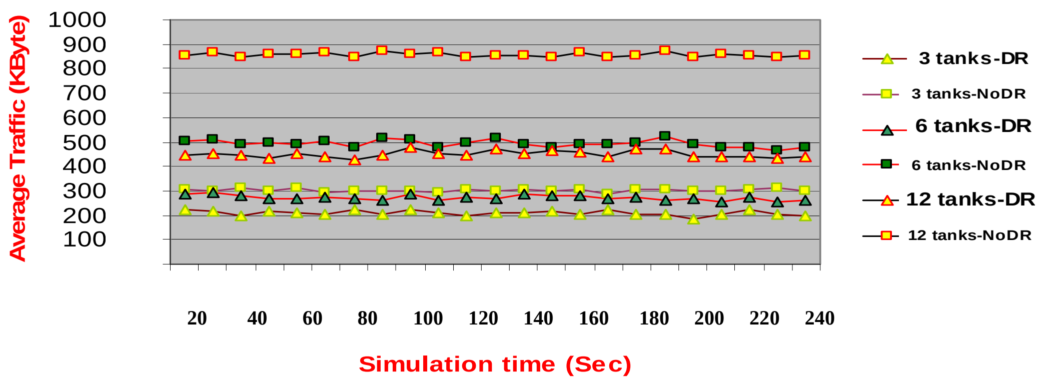


Fig. 8 recorded experimental results

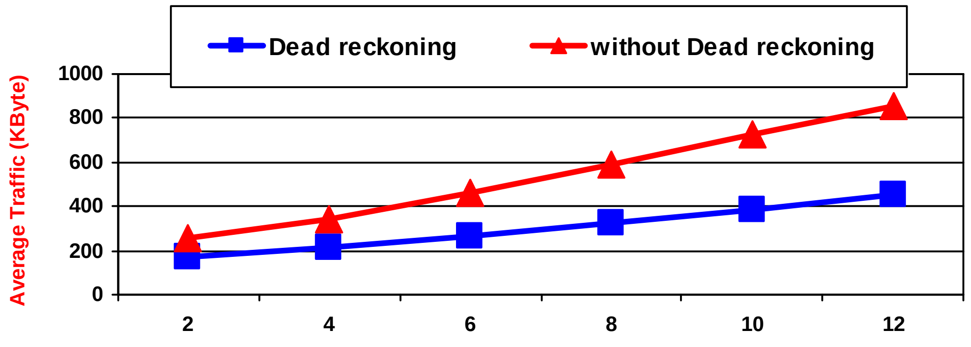


Fig. 9 Enhancement in network traffic using Dead reckoning

Table 1 Threshold levels

level	1	2	3	4
description	In another entity's SR	In another entity's AOI	Overlap of AOIs with another entity	No overlap of AOIs

Table 2 Extrapolation equations

	One-step	*Two-Step*
1st Order	$X_t = X_{t'} + \upsilon_{t'} \tau$	$X_t = X_{t'} + \dfrac{X_{t'} - X_{t''}}{t' - t''} \tau$
2nd Order	$X_t = X_{t'} + \upsilon_{t'} \tau + 0.5 a_{t'} \tau^2$	$X_t = X_{t'} + \upsilon_{t'} \tau + 0.5 \dfrac{\upsilon_{t'} - \upsilon_{t''}}{t' - t''} \tau^2$