Simulating Fusion Algorithms with Gaming Platforms

Lundy Lewis
Engineering Dept
Altusys Corp.
Manchester, NH USA
lewis@altusystems.com

Chris Wright
IT Dept
Southern New Hampshire University
Manchester, NH USA
christopher.wright@snhu.edu

Nolan DiStasio
IT Dept
Southern New Hampshire University
Manchester, NH USA
Nolan.Distasio@snhu.edu

Abstract – We discuss issues in testing various cognitive fusion algorithms for situation management. We provide a proof-of-principle discussion and demo showing how gaming technologies and platforms could be used to devise and test various fusion algorithms, including input, processing, and output, and we look at how the proof-of-principle could lead to more advanced test beds and methods for high-level fusion in support of situation management. We develop four simple fusion scenarios and one more complex scenario in which a simple rule-based system is scripted to govern the behavior of battlespace entities. We conclude that the proof-of-principle warrants further work to experiment with various high-level fusion algorithms and more complex scenarios for simulation exercises.

Keywords: situation management, information fusion, simulation, game engine, game editor

1 Introduction

It has been argued that traditional military simulation such as the Joint Conflict and Training Simulator (JCATS) and the One Semi-Automated Forces System (OneSAF)) fall short with respect to realism, scope, richness, and complexity of the battlespace [1]. Further, the consensus is that the next generation of users of simulations for the military will have grown up playing complex games in which the user becomes immersed in the game [2,4]. The new user will expect a degree of richness and visualization not found in the traditional simulations.

In this paper we experiment with a gaming platform, Neverwinter Nights 2, to simulate the Collection Management function in battlespace operations [3]. The work reported here is an extension of prior work [4].

Collection Management (CM) encompasses the set of procedures that orchestrate organizations and systems to focus on the acquisition of intelligence in support of war fighting and other operations. CM results in a *collection plan* that is followed during the course of an operation.

Three sub-functions of CM are as follows:

Requirements Management (RM) focuses on what information needs to be collected, and when and where to collect it, which is incorporated into the collection plan. At various points in the operation, the plan dictates the information that is needed to make appropriate decisions to carry out the operation. During the execution of the operation and the accompanying collection plan, the plan may be modified based upon collection results, unforeseen intelligence, and changes in higher-level operational goals. Thus, the collection plan is dynamic rather than static.

Mission Management (MM) focuses on how to deploy resources to collect information specified in the collection plan. MM evaluates the suitability of systems, units, and agencies based upon capability and availability. MM continually monitors the readiness and performance of resources needed to execute the collection plan.

Asset Management (AM) executes the collection plan to offer support for a commander's operational decisions. AM combines the what, when, and where in RM with the how in MM, and proceeds to execute the collection plan with available assets and resources.

Clearly, high-level data fusion is employed at several levels in the CM task. The work presented here is an exploratory effort to determine the extent to which such high-level fusion can be simulated with recent gaming platforms, including experimentation with various fusion technologies and collection strategies.

The paper is structured as follows: First, we summarize the ideal capabilities of a forward-looking simulation environment for fusion experiments. Second we examine several architectures for military gaming-based simulations. Third, we describe four simple fusion scenarios based on rule-based reasoning, and then we describe increasingly complex fusion scenarios that mimic part of the CM function. We close with a Summary and Outlook.

2 Ideal Capabilities for Forward-Looking Simulations

A prior two-year study [1] reported shortfalls with traditional modeling and simulation technologies, workarounds, how modeling and simulation capabilities support R&D in higher-level fusion, and the ideal capabilities of a modeling and simulation environment. In sum, the ideal capabilities of a simulation environment for CM are as follows:

Create Ground-Truth Scenarios

C1: Be able to implement realistic ground truth scenarios that produce intelligence reports detailed enough to drive higher-level fusion applications.

C2: Be able to modify the ground truth scenarios during run-time to reflect changes in tactics, rules of engagement, and asset performance.

C3: Be able to implement alternative fusion algorithms to study the output resulting from modifying ground truth scenarios.

C4: Be able to *easily* implement ground truth scenarios and alternative fusion algorithms by using wizards, dialog windows, mouse-overs, drilldowns, and help screens.

C5: Be able to simulate entities as realistic representations of dismounted infantry, guerillas, vehicles, civilians, organizations, communications networks, computer systems, buildings, utilities, road networks, and other aspects of operational environments.

C6: Be able to have full control over the specific characteristics and behaviors of the entities, where the default state represents a believable individual in actions, maneuvers, and self-preservation.

Insert a Collection Plan into the Scenario

C7: Be able to insert an information collection plan into a scenario driver.

C8: Be able to allocate sensor assets to one or more areas of interest according to the collection plan.

C9: Be able to have sensor assets provide intelligence reports that are tied to a specific collection plan in support of commanders.

C10: Be able to explore error values and other limitations of sensor assets.

C11: Be able to generate reports from sensor assets and entities, store them in a retrievable database, and display them graphically on a map.

C12: Be able to modify or change the values of the reports in order to reflect a degradation or inaccuracy of the information.

C13: Be able to time-delay the reports to simulate pre-processing of information, network communication latency, adversary jamming, etc.

An example is this: As enemy forces maneuver across the battlespace, they would be detected by sensor models and spotted by friendly observers. The sensor data generated and spot reports issued, which are both linked to the collection plan, would allow a test subject to gain insight into what actions the simulated enemy forces are taking.

In the following sections, we experiment with capabilities C1 – C6, C9, C10 and the example given above.

3 Alternative Approaches to Forward-Looking Simulations

Traditional Modeling and Simulation (M&S) applications for battlespace operations lack realism and richness of the actual battlespace. The design of content for traditional M&S applications, from a semantic point of view, includes the essence of entities, strategies, and other battlespace objects to inform the user. The problem, however, is how to transform the essential semantics of traditional M&S technologies into the rich visualization semantics offered by recent gaming technologies so that the user feels immersed in the simulation and experiences the real complexities of the battlefield [6].

There have been a number of approaches to lessen the gap between traditional M&S and the required visualization to stimulate today's type of users, briefly discussed below:

1. Build custom visualization techniques for the M&S products from scratch. The problem with this approach is that it takes considerable time and engineering resources to develop the visualization toolkits.

2. Build a uni-directional gateway between M&S and modern gaming platforms to take advantage of the advanced visualization capabilities of the gaming

platforms. This approach is like #1, except that there must be a correspondence between M&S objects and game platform objects. The problem with this approach is that much art investment is required to enhance objects in the M&S space with detailed features that are available for corresponding objects in the gaming platform space.

3. Build a bi-directional gateway between M&S and modern gaming platforms, wherein objects in the game platform space are controlled by the game interface, and then state information in the game-platform is sent back to the M&S for further processing [2,7]. For example, prior research describes a bi-directional gateway between the Unreal Game Engine (UGE) v2.5 and the US Army's OneSAF system [2]. This approach suffers the same problem as that in #2, viz. it requires much artist investment, and further, the processing required for the continuous interaction between the two systems can be prohibitive.

4. Build a universal, standardized application programming interface (API) to simplify the integration between multiple M&S applications and multiple gaming platforms [5]. This approach is a generalization of #3, where the goal is to allow plug-and-play with many different gaming engines and engine subcomponents and allow them to communicate with diverse military simulation applications. The requirement is that M&S and game platform vendors follow the API, which is prohibitive.

Our approach is of a different sort. Using CM as the task domain, we look at the ideal capabilities of forward-looking simulations and explore the extent to which we can achieve them in an extensible, customizable gaming platform NeverWinter Nights 2 (NWN2) developed by Obsidian Entertainment and published by Atari [8, 9].

NWN2 is a commercial off-the-shelf (COTS) computer game that has been recognized as a viable testbed for studying situation awareness and decision-making in domains such as team behavior, speech understanding, and education [10]. The key features of NWN2 include [11]:

- Multiplayer capability across local-area-networks and the Internet.
- Authoring tools enabling researchers to construct their own scenarios.
- GUI and scripting interfaces to facilitate entity control by computational models.
- Automated data capture mechanisms to collect data for post-experiment analysis.

The approach presented in this paper likewise demands scenarios to be programmed; however, the advantage is that users employ the authoring tools and GUI/scripting interfaces. The game-based testbed provides an immersive environment in which the actions and behaviors of entities intertwine with the experimenter. Scenario data can be observed or collected for post-analysis. In experimental scenarios, entity behaviors can range from being well-structured and deterministic to being unstructured and random, subject to the control of the experimenter [11].

4 Four Simple Scenarios

In our first scenario, Entity-1 wakes up and reasons about his abilities and how well he can fare in a fight against other entities with various skills and abilities. In a second scenario, the entity discovers Entity-2 who is in a neutral state of readiness and is waving. Entity-1 recognizes that he might have met a possible friendly character, and thus will walk up to Entity-2 and talk. In a third scenario, Entity-1 sees Entity-3 and understands that he has an advantage over Entity-3 in terms of skills and capabilities, and thus attacks. During the attack, Entity-3 takes on additional capabilities and thus gains an advantage over Entity-1, upon which Entity-1 reasons that his best option is to take flight. The fourth scenario is the same as the third, save that the skills and capabilities of entities are set randomly by a random-number generator.

The knowledge of Entity-1 is coded with the game's scripting language as a simple rule-based system. See the development interface in Figure 1 and the following simple snippet of code in which an entity takes stock of his capabilities, discovers some other entity, takes stock of the other entity, and decides whether fight or flight is in order.

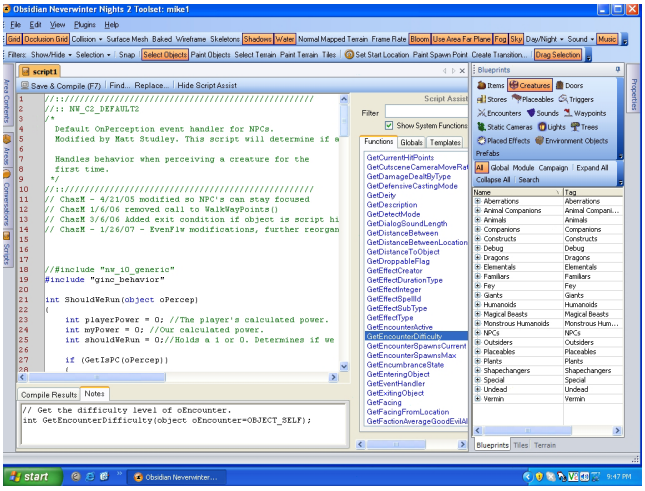

Figure 1. Scripting Interface

```
/*
Default OnPerception event handler for NPCs. Modified by Matt Studley. This script will
determine if an entity is strong than a 2nd entity, then decide if he  should run. Handles
behavior when perceiving an entity for the first time.
*/

#include "ginc_behavior"

int ShouldWeRun(object oPercep)
{
    int playerPower = 0; //The player's calculated power.
    int myPower = 0; //Our calculated power.
    int shouldWeRun = 0;//Holds a 1 or 0. Determines if we should run.

    if (GetIsPC(oPercep))
    {
    //Calculate the average power of myself.
        myPower += GetAbilityScore(OBJECT_SELF,ABILITY_STRENGTH,0);
        myPower += GetAbilityScore(OBJECT_SELF,ABILITY_INTELLIGENCE,0);
        myPower += GetAbilityScore(OBJECT_SELF,ABILITY_DEXTERITY,0);
        myPower += GetAbilityScore(OBJECT_SELF,ABILITY_CONSTITUTION,0);
        myPower += GetAbilityScore(OBJECT_SELF,ABILITY_CHARISMA,0);
        myPower += GetAbilityScore(OBJECT_SELF,ABILITY_WISDOM,0);

    //Now Calculate the average power of the player
        playerPower += GetAbilityScore(oPercep,ABILITY_STRENGTH,0);
        playerPower += GetAbilityScore(oPercep,ABILITY_INTELLIGENCE,0);
        playerPower += GetAbilityScore(oPercep,ABILITY_DEXTERITY,0);
        playerPower += GetAbilityScore(oPercep,ABILITY_CONSTITUTION,0);
        playerPower += GetAbilityScore(oPercep,ABILITY_CHARISMA,0);
        playerPower += GetAbilityScore(oPercep,ABILITY_WISDOM,0);

        SpeakString("Power Calculations: " + IntToString(myPower) + " v.s." +
         IntToString(playerPower) + ".",TALKVOLUME_TALK);

        if (myPower < playerPower)
                {
                  shouldWeRun = 1;
                 }
    }
                return shouldWeRun;
}
```

In this script, if an entity perceives a 2nd entity, he sums the values of strength, intelligence, dexterity, constitution, charisma, and wisdom of himself and the 2nd entity and performs a comparison, based on which he decides whether to run. One can imagine the GetAbilityScore function replaced by a function doing something complex such as retrieving data from external data stores and fusing the data, and one can imagine more complex reasoning than the one rule. An interesting exercise would be to explore the belief-desire-intention (BDI) agent to model the behavior of the entities and their interactions [12].

During runs of the scenarios, a window is shown with detailed information about the entities and entity relationships in the scene. The window shows physical states, attack states, a challenge metric for opponents, and a comparison of total capabilities among entities. We used the console commands at the bottom left of the screen to insert skills and capabilities into entities on the fly. See Figures 2 and 3. Figure 2 is a screenshot in which state of entities is shown at the beginning of the second scenario. Figure 3 is a screenshot in which the comparison is made and the message “You are stronger than me. I don’t want to attack you” appears.

Figure 2. States of Entities in the Scene

Figure 3. Comparison of States and Decision Reporting

5 Increasingly Complex Scenarios

A more complex scenario using this environment includes obstacles such as hills and water. It also features units which can be commanded and given an objective. In this scenario, Entity-1 must stop Entity-2 from reaching a waypoint. Entity-1 wakes up in the world and finds that it must cross over a water obstacle before continuing into an urban environment. It then must navigate around hills and buildings to find and neutralize Entity-2. Entity-2 has knowledge encoded in it to evade and judge potential hostile targets it encounters. If a hostile target is weaker than Entity-2, Entity-2 will attempt to eliminate the hostile threat to its mission. Otherwise, it will attempt to evade the

hostile entity. There is also a sensor entity that can report to Entity-1 on the whereabouts of Entity-2 by displaying overlaying text on the screen. The scenario is successful when Entity-1 navigates through the urban environment, finds Entity-2, and eliminates it. The scenario fails if Entity-1 fails in preventing Entity-2 from reaching its waypoint or if Entity-1 is incapacitated.

This scenario can be re-configured to reflect different urban layouts, unit strengths, and obstacles by using the NWN2 toolset to edit the environment. Figure 4 shows Entity-1 in the beginning of scene. Figure 5 shows the addition of hills, obstacles, and likeness to an urban environment, and Figure 6 shows the editor interface.

Figure 4. Scenario Initiation

Figure 5. Hills and Urban Environment

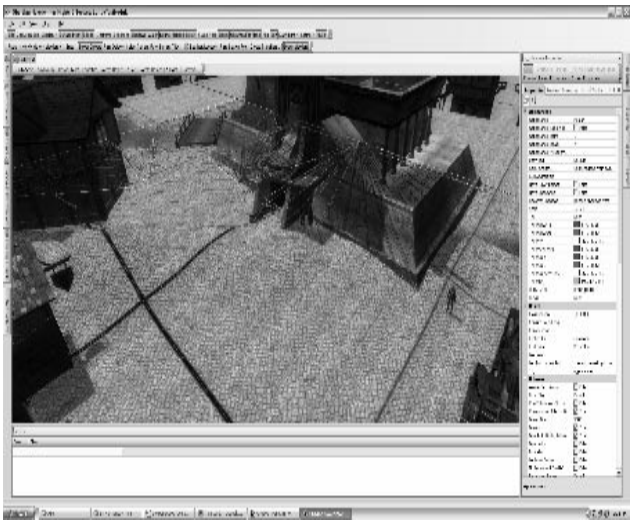

Figure 6. The Editor Interface

In our final scenario, Entity-1 is attempting to make its way to a home base. As an extension to the work reported in [4], we cast the scenario in terms of situation management, where Entity-1 tries to understand his current situation and where new intelligence causes him to formulate new situations. Environmental and intelligence events are fused with a simple fusion algorithm and the results of the fusion cause Entity-1 to invoke a situation template. The instantiated situation template may be used for further reasoning and for guidance to seek out additional environment or intelligence data.

Being provided with real time intelligence forces Entity-1 to come up with strategic plans and make decisions based on the new intelligence. The enemy is spotted by a friendly observer and reported to Entity-1 during his attempt to make it home. With this new information, Entity-1 has to make real time decisions about how he should proceed. This scenario demonstrates capabilities C1 – C6 described in Section 2.

Entity-1 and the friendly observer discuss the new intelligence whereupon Entity-1 decides it is a good idea to take an alternate path to home base – one that will avoid the enemy forces. Without this data, Entity-1 may have run directly into the enemy. In Figure 7, the default path to home base is through a forest. Upon receiving intelligence of enemy locations by the friendly observer, an alternative path is chosen over a mountain range, as shown in Figure 8. Upon receipt of further intelligence, an alternative path is chosen through a cave, by which Entity-1 makes it to home base, Figure 9. And interesting extension to this scenario would be to give Entity-1 the intelligence to set up a trap for the enemy forces by devising a plan with its peers.

Figure 7. Reaching Goal via Forest

Figure 8. Reaching Goal via Mountain Range

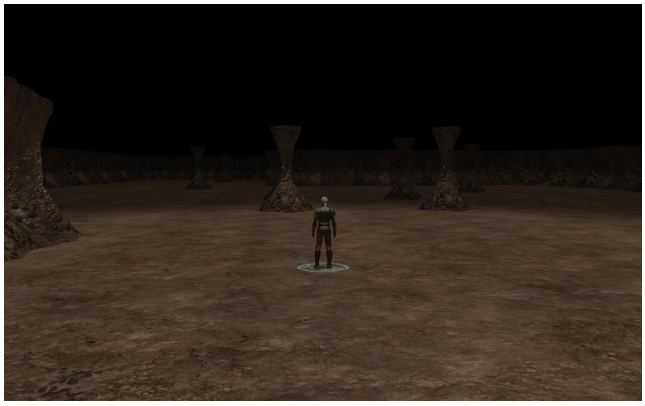

Figure 9. Reaching Goal via Cave

6 Summary and Outlook

In this paper we have discussed proof-of-principle work towards using extensible gaming platforms to experiment with high-level fusion and simulation. We discussed the domain of Collection Management, outlined the ideal capabilities of future simulation environments, and experimented with the implementation of these capabilities with an advanced gaming platform, NeverWinter Nights 2. We developed several simple fusion scenarios for the proof-of-principle.

We believe that the proof-of-principle warrants further work to experiment with various high-level fusion algorithms and more complex scenarios for simulation exercises. For example, references [12,13] describe increasingly complex scenarios involving BDI agents, multi-agent systems, fusion algorithms, and situation management on several levels in urban environments. Reference [14] describes an effect-based modeling paradigm based on the agent metaphor which may provide groundwork for experimentation as well. In addition to more complex scenario development and algorithm development, there are further issues to address: (i) create realistic artistic content for urban and battlespace environments, (ii) increase the number of entities into the 100s or 1000s to test for performance and control, (iii) evaluate the degree of immersion experienced by domain experts, and (iv) evaluate the degree of cognitive aid for domain experts.

More game editors are being shipped with popular games whereby users can create content and design new scenarios and games for experimentation. The authors are evaluating these frameworks against the ideal capabilities outlined in Section 2. Examples include MyCryENGINE [15], UNREAL Development Kit [16], and Age of Empires III [17].

Finally, we are especially interested in situation management, whereby entities are able to develop plausible situations based on the fusion of lower-level observations, INTEL, messages from collaborating agents, and past experiences. Further, we wish to equip entities with the reasoning power to imagine plausible futures, to make choices of desirable and undesirable futures, and to take actions in current situations towards realizing desirable futures.

References

[1] C. Pizzo, G. Powell, C. Brown III, and J. May, Modeling and Simulation Support for Answering Commanders' Priority Intelligence Requirements, *10th International Command and Control Research and Technology Symposium*, June 2005.

[2] R. McAlinden and W. Clevenger, Using Commercial Game Technology for the Visualization and Control of Constructive Simulations, *Spring Simulation Multiconference 2007*, March 2007.

[3] Collection Management and Synchronization Planning, FM 34-2, Department of the Army, Washington DC, March 8 1994.

[4] L. Lewis, N. DiStasio, and C. Wright. Using Gaming Engines and Editors to Construct Simulations of Fusion Algorithms for Situation Management. Proceedings of the SPIE Defense, Security, and Sensing Conference. Orlando FL April 5 - 9, 2010.

[5] K. Doris, M. Larkin, M. Zieniesicz, and R. Szymanski. Applying Gaming Technology to Military Visualization – Games Where You Only Live Once!, *Simulation Technology Magazine*, Vol. 8, Issue 1, April 28 2005.

[6] G. Frazier, and P. Blemberg. Intelligence Cells in Large Scale Exercises, 99S-SIW-194. *Proceedings of the Spring Simulation Interoperability Workshop*, 1999.

[7] P. Prasithsangaree, J. Manojlovich, J. Chen, and M. Lewis. UTSAF: A Simulation Bridge between OneSAF and the Unreal Game Engine, *IEEE International Conference on Systems, Man and Cybernetics*, October 2003.

[8] http://www.atari.com/nwn2/

[9] http://en.wikipedia.org/wiki/Neverwinter_Nights_2

[10] Leung, A., Diller, D., & Ferguson, W. (2005). SABRE: A Game-Based Testbed for Studying Team Behavior. *Proceedings of the Fall Simulation Interoperability Workshop (SISO*). Orlando, FL, September 18-23, 2005.

[11] J. Sutton and V. Edelmann. Leader and Team Adaptability in Multinational Coalitions (LTAMC): An International Research Project. *10th International Command and Control Research and Technology Symposium.* June 2005.

[12] L. Lewis, J. Buford, and G. Jakobson. Inferring Threats in Urban Environments with Uncertain and Approximate Data: An Agent-Based Approach. *Journal of Applied Intelligence*: *The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies*. Published online: 28 August 2007.

[13] G. Jakobson, J. Buford, and L. Lewis. Collaborative Agents for C2 of Tactical Urban Combat Operations. *Defense Transformation and Net-Centric Systems 2008, edited by R. Suresh, Proceedings of the SPIE*, Volume 6981, 2008.

[14] A. Tolk, R. Bowen, and P. Hester: Using Agent Technology to Move from Intention-based to Effect-based Models. *Proceedings of the 2008 Winter Simulation Conference.*

[15] http://mycryengine.com/

[16] http://www.udk.com/

[17]http://aoe3.heavengames.com/modding/tutorials/index.shtml