# Integrating Learning in Interactive Gaming Simulators

**David W. Aha[1] and Matthew Molineaux[1,2]**

[1]Intelligent Decision Aids Group; Navy Center for Applied Research in Artificial Intelligence;
Naval Research Laboratory (Code 5515); Washington, DC 20375
[2]ITT Industries; AES Division; Alexandria, VA 22303
{aha,molineau}@aic.nrl.navy.mil

**Abstract**

Many developers of simulations for computer-generated forces and real-time strategy games seek to incorporate learning or learned behaviors in their systems. Likewise, many researchers seek to evaluate their learning systems in these simulators. However, these integrations require great effort. We describe our initial work on a testbed, named *TIELT* that we are designing to facilitate these integrations.

## 1. Motivation and Related Work

A key objective of DARPA's Information Processing Technology Office is to encourage research on learning in cognitive systems (Brachman, 2002). This thrust could significantly impact the machine learning (ML) community by shifting from an emphasis on isolated studies (e.g., on supervised learning) towards integrated studies in the context of ambitious reasoning systems. This requires providing the ML community with tools that facilitate these integrations. While current availability of standardized data representations and data sets (e.g., UCI, 2004) has met the needs of some isolated benchmarking studies (e.g., on classification tasks), no similar interface exists for integrating with cognitive systems.

Interactive computer games are excellent candidates for studying learning in cognitive systems for several reasons. First, some attempt to simulate cognitive behaviors. This is particularly true for military simulators of computer-generated forces (Laird & van Lent, 2001); they share some similarities with real-time strategy games, although the objectives for incorporating learning differ among military (exploration, analysis) and commercial (enhance the game playing experience) simulators. Second, they are popular. For example, Americans spent $7B on video and computer games in 2003 (ESA, 2004), and *America's Army: Operations* was downloaded 2.5M times in its first two months of release (Zyda *et al*., 2003). Third, AI researchers recognize them as a "killer" application (Laird & van Lent, 2001), and several have studied integrations of ML techniques in gaming engines/simulators (e.g., Laird, 2001; Geisler, 2002; Sweetser & Dennis, 2003; Ponsen, 2004). Also, several game engines exist that encourage ML development (e.g., RoboCup, 2004; Kaminka *et al*., 2002; Buro, 2003; Houk, 2004), as does some commercial AI middleware (e.g., MASA's *DirectIA*, LearningMachine's *NOMAD*, SHAI's *SimBionic*).

Although middleware applications can greatly simplify the design and development of new interactive games, they handicap researchers in several ways: they were not necessarily designed to address the full range of potential learning and performance tasks in interactive games; they don't encourage the sharing of learning systems and game engines for use by other researchers in subsequent investigations; and they are not free. For example, GameBots (Kaminka *et al*., 2002) can also be used to study learning systems, but it is committed to a single game engine (the Unreal Tournament Server) and a single reasoning activity (i.e., a sense-act loop), while TIELT shall support integrations with many game engines and several types of reasoning activities (e.g., display predicted opponent behaviors, update a game model, incorporate advice) in which learning can be studied. Thus, it's still difficult for ML researchers to conduct benchmarking and related tests that compare their system's ability vs. alternatives across a set of learning and performance tasks for multiple gaming engines. Similarly, game developers cannot easily compare state-of-the-art learning systems to determine whether any address their design needs.

We address the challenge of developing a middleware testbed that facilitates and encourages the evaluation of learning systems in the context of interactive computer games. Our system, named *Testbed for Integrating and Evaluating Learning Techniques* (TIELT), shall support the ML research community by providing composable interfaces to game engines and reasoning systems, the ability to select a wide variety of learning and performance tasks, and an editor for specifying and conducting an evaluation methodology. By providing access to challenging learning and performance tasks definable in these simulators, TIELT should encourage the creation of knowledge rich learning strategies that learn from only a few examples, and learn over an extended period of time. In addition, it should spur research on problems related to learning systems that are frequently cited by the commercial and military gaming communities (e.g., Woodcock, 2002; Petty, 2001), such as that some costs (e.g., cpu time, number of training examples required) associated with training can be prohibitively high, and that learned behavior may be unrealistic and/or unpredictable.

In this paper we describe TIELT's specification, our approach for implementing that specification, an illustrative example, and our progress and future goals.

## 2. Specification

Our goal is to open up the playing field of interactive gaming simulators to learning research. While the initially targeted beneficiaries are ML and cognitive systems researchers, our longer term goals include providing a useful investigation tool for the commercial gaming industry and developers of military simulators. With this vision in mind, along with our objective to streamline the process of integrating learning systems with gaming simulators, we have the following goals for TIELT:

1. *Integration*: TIELT should input a description of the game's model and state, a description of the inputs and communication medium required to communicate with the learning-embedded reasoning system, a learning task(s), and a performance task(s). During a game, it should interpret a sequence of game states, translate them for input to the reasoning system, interpret the system's response (e.g., a decision consisting of one or more actions for controlling a game engine agent), and translate it for display and/or as input to the game engine. It should also support a variety of empirical studies on the reasoning system's utility for the learning and performance task(s).

2. *Learning focus*: TIELT should support investigations for learning three types of planning-focused models:

   a. *Task model*: Given a task interpretation, its learned model could be used to execute an action or provide it as advice to a user or software agent.
   b. *Player model*: Given a player-focused state representation, its learned model could be used to predict a user's actions or suggest/execute an action.
   c. *Game model*: TIELT could also be used to improve or learn a model of the game's environment or its agents' behaviors, either for predicting behavior or prescribing response actions.

The reasoning system could also support other learning investigations (e.g., improving the representation of a game's state description).

3. *Learning methods*: TIELT should work with supervised, unsupervised, analytic, and reinforcement learning methodologies. It should support online and offline training by providing learning systems with a stream of game state descriptions or by recording sessions for later training. In addition, TIELT should support investigations on using a priori knowledge to constrain learned behavior models, preferably during the model-learning process but also for post-hoc approaches that correct learned models (e.g., by analyzing execution-time errors).

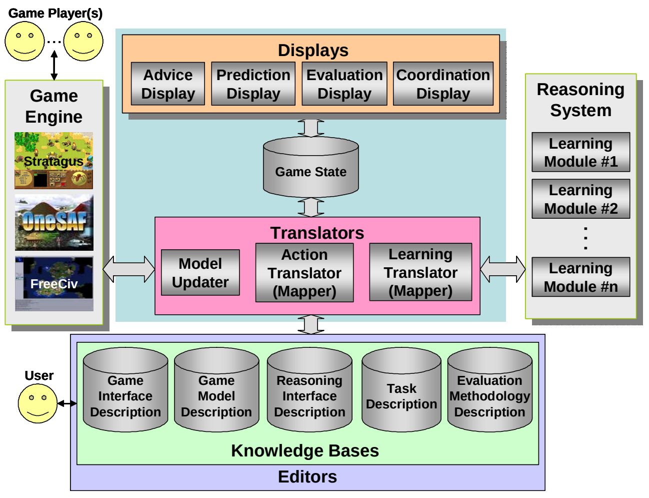

**Figure 1**: Simplified TIELT component architecture.

4. *Game engines*: TIELT should facilitate a researcher's access to simulators for strategy (real-time and discrete, God and first-person perspectives), role-playing (individual and massively-multiplayer), and team sports games. These include games involving learning tasks with large hypothesis spaces that can benefit from learning and learned strategies.

5. *Reuse*: TIELT should permit researchers to easily study a learning system's ability on tasks from several games, and permit game developers to study the comparative ability of multiple learning systems on a given task. For example, when applying a reasoning system to a different learning task but using the same game engine, only that task will require specification; the game description and interface descriptions may remain unchanged.

6. *Platforms and programs*: TIELT should be available for use on all major platforms, and provide support for slower reasoning systems (e.g., for studying real-time simulators).

## 3. Approach

Given this specification, Figure 1 displays a subset of TIELT's components for integrating learning-embedded reasoning systems with interactive game engines. These components will include displays, a model of the (current) game state, translators for communicating between the learning and gaming systems, a set of knowledge bases and their respective GUI editors (not shown), and a Controller (not shown) for monitoring and managing communications. The editors will facilitate interchangeability of the five knowledge bases, which are described immediately below.

The *Game Interface Description* defines how TIELT communicates during a game simulation and the method of communication. This will include definitions of *message templates*, which describe the parameters of a message and its connection to the Game Model. Two types of messages will be defined: *action* messages from the reasoning system, which TIELT can use to affect the game, and *percept* messages from the game engine, which provide information to the reasoning system. TIELT will have built-in support for communication via a socket-based TCP/IP network connection, console I/O, interprocess messaging, and dynamic linking so as to be able to communicate with many different systems. Each *percept* message should identify what changes in the state it signifies, and each *action* message must identify how it changes the state. These changes will be stated in terms of the Game Model Description - the second knowledge base.

The *Game Model Description* provides an explicit, abstract description of a game. It consists of an initial state that defines all objects of potential player interest, operators that describe ways to interact with the game, and rules that govern how the game may be played and the consequences likely to arise from a particular game state. This information is kept separate from the Game Interface Description because different reasoning systems might benefit from viewing the same game in different ways. Also, this abstraction frees a learning system from a game engine's timetable; rather than responding directly to the information when supplied, the learning modules can consult the model as needed, which simplifies integration with a real-time game. A third benefit of this abstraction is that a single game model may potentially be applicable to an entire category of games (e.g., first-person shooters), so that the learning system would view each the same way. Thus, a trained system could potentially transfer knowledge learned from one game to another.

The *Learning Interface Description* uses message templates similar to those in the Game Interface Description to define communication with the learning-embedded reasoning system. Each *learning input* message allows a particular mode to be identified so as to enable separate training and test messages, gives a trigger that causes the message to be sent, and provides a list of "slots" that game information can be "plugged" into. The slots, by providing conversions from basic types to the types used by the reasoning system, will ensure that this system's messages need only be defined once, and henceforth game attributes can be mixed and matched as needed.

Once a game and reasoning system's interfaces are described, defining learning and performance *Task Descriptions* in TIELT is straightforward. By matching up a learning system's inputs with state information and its outputs with model operators, a researcher can quickly describe a learning task. Configurable equations will measure system performance over a suitable range of mathematically expressible metrics and display the results.

Finally, the *Evaluation Methodology Description* allows a researcher to define exactly how to conduct an evaluation. For example, the user will be able to command TIELT to train a learning module live against an internal game AI for a specified number of sessions, or to record an online game between human opponents and later use the recorded information for cross-validation studies.

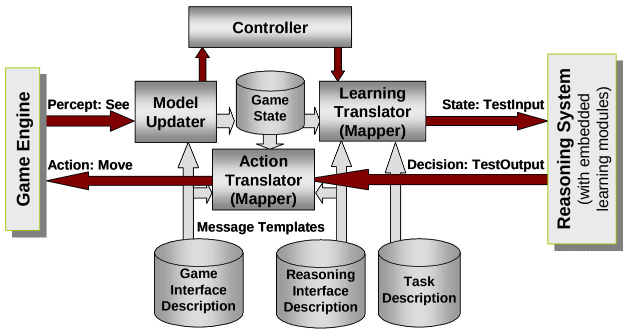
Controller
Game Engine
Percept: See
Model Updater
Game State
Learning Translator (Mapper)
State: TestInput
Reasoning System (with embedded learning modules)
Action: Move
Action Translator (Mapper)
Decision: TestOutput
Message Templates
Game Interface Description
Reasoning Interface Description
Task Description

**Figure 2**: Data flow in TIELT for the city placement task.

Over time, these five TIELT knowledge bases will accumulate. This will permit researchers to compare their learning-embedded reasoning systems vs. others across a variety of games and game engines. Likewise, game engine developers will be able to access multiple reasoning systems to compare their abilities on selected learning and performance tasks.

## 4. Example

To highlight TIELT's operation, we now describe the system's behavior through a cycle during an on-line game (Figure 2). Our example involves a city placement task like those common to strategy games (e.g., Civilization®), similar to a task addressed by Houk (2004). We assume TIELT is given a map of the terrain on which a city may be placed. The map is a two-dimensional matrix of squares that vary in the amount and type of resources they can provide to a nearby city. The initial state provides TIELT with a settler and its starting location. The performance task is to maximize resource acquisition in the time provided, which we set to ten turns. The learning task concerns deciding which square to select for building a city. To acquire resources, a settler must first move to the desired location and build a city there. Thus, the system must learn how far to travel to find a better city location.

When our example game creates a settler, it sends a sensor message to TIELT's Model Updater module, namely *See(settler, {0,0})*. The Model Updater retrieves the corresponding message template *See(object, location)* in the Game Interface Description, and updates the Game State using instructions in the retrieved message template. The effect is that the Game State now includes the information that there is a settler at map location {0,0}. The Model Updater then notifies the Controller that the Game State was updated and a *See* message was received, and it waits for another message.

The controller finds that a *See* message has been received. If it were in *recording* mode, it would potentially notify the Database Engine (not shown) so that the updated state could be recorded. Instead, it sends state information to the Evaluator (not shown), which updates the

**Table 1**: A sampling of learning research with real-time gaming environments.

| Name + Citation | Engine | Learning Approach | Task Focus |
|---|---|---|---|
| (Goodman, 1993) | Bilestoad | Projective Visualization | Fighting maneuvers |
| CAPTAIN (Hieb *et al*., 1995) | ModSAF | Multistrategy (e.g., version spaces) | Platoon placement |
| MAYOR (Fasciano, 1996) | SimCity | Case-Based Reasoning | City development |
| (Fogel *et al*., 1996) | ModSAF | Genetic Programming | Tank movements |
| KnoMic (van Lent & Laird, 1998) | ModSAF | Rule condition learning in SOAR | Aircraft maneuvers |
| (Agogino *et al*., 1999) | Peon (inspired by Warcraft II) | Neuroevolution | Search and evasion strategies |
| (Geisler, 2002) | Soldier of Fortune | Multiple (e.g., Boosting Backprop) | FPS decision tasks |
| (Sweetser & Dennis, 2003) | Tubby Terror | Regression | Advice generation |
| (Chia & Williams, 2003) | TankSoar | Naïve Bayes Classifier | Tank behaviors |
| (Guestrin *et al*., 2003) | Freecraft | Relational MDPs | Strategic/tactical battle |
| (Spronk *et al*., 2004) | Neverwinter Nights | Reinforcement Learning (Dynamic Scripting) | Adaptation of opponent AI |
| (Ponsen, 2004) | Wargus/Stratagus | Reinforcement Learning (Dynamic Scripting) | Strategic rule selection |

performance values for all tasks currently running, and returns. Then the Controller notifies the Learning Translator that a *See* message has been received.

The Learning Translator checks the knowledge base of user-specified tasks to determine which (if any) provides a response triggered by a *See* message. The city location task specifies that a *TestInput* message may be sent when a *See* message is received that concerns a settler unit. The Learning Translator confirms this update by checking the Game State, and then retrieves the *TestInput* message template from the Learning Interface Description. TIELT has information about the map and the settler in the slots of the *TestInput* message. Therefore, it constructs a new message, formatted according to the *TestInput* message template, using map and settler information to create the parameters, and sends the resulting message to the reasoning system.

After the reasoning system receives the message, its behavior is not constrained. The reasoning system is expected to reply with a behavior in a reasonable amount of time; information about what constitutes "reasonable" may be provided to it. At this point, a learning module might consider past successes and failures before selecting a goal from a manually defined set. Its embedding reasoning system would then formulate a plan based on this goal and send the plan's first operator to TIELT.

The reasoning system returns a *TestOutput* message. The Action Translator receives it, then finds the corresponding *TestOutput* message template, whose single parameter has a *MoveSettler* operator plugged into it. Next, the Action Translator checks the Game Interface Description to see what action messages can be triggered by a *MoveSettler* operator. It finds the *Move* message, and the operator is re-composed to *Move(Settler, {1,1})*. This message is sent to the game engine, which receives the *Move* message and moves the settler to the new location {$x_1$, $y_1$} as described.

After ten turns proceeding in much the same way, the reasoning system's performance will be evaluated based on the evaluation methodology, which will define, for example, metrics and experimental variables. Improvement in this scenario can be measured in terms of a higher game score, which is weighted according to city size and its accumulated funds. This score can then be compared to previous or subsequent runs, or the performance of other learning modules and/or reasoning systems.

## 5. Progress and Future Work

We began designing TIELT in December 2003 and now have a nearly complete functional design and a partially implemented prototype. We will repeatedly evaluate TIELT for its ability to assist both learning researchers and (commercial, military) game developers. Thus, some of our metrics will concern ease of use. We plan to show that it could be used to support some previous integrations of learning systems and game engines, a sampling of which are summarized in Table 1. Also, we have begun collaborating with AI researchers who can benefit from applying TIELT and provide us with valuable feedback.

We will take several steps to produce a useful tool for the research community. This will include publicizing TIELT's availability and providing it with knowledge bases for a variety of tested Descriptions (i.e., Game Model, Game Interface, Reasoning Interface, and Task) along with standardized challenge problems. We will explore how it can use sophisticated representations to support capabilities such as qualitative spatial reasoning (Forbus *et al.*, 2001). Also, we will adopt or develop standards for game models, task descriptions, and interfaces for game engines and reasoning systems, which should enhance TIELT's utility. In particular, we plan to specify a standard format for reasoning systems to output their learned behaviors such that they can be inspected by game developers. Finally, to better support high-level strategic decisions, we later plan to investigate using hierarchies of agents, which will allow multiple learning (and other reasoning) systems to easily cooperate in working toward a larger goal than any individual system could handle alone.

However, while assisting the research community is a first goal, our ultimate objective is to impact the process for developing commercial games and military simulators of computer-generated forces. This will require detailed requirements analysis, along with convincing demonstrable progress made by AI researchers while using TIELT.

## Acknowledgements

Thanks to DARPA's Information Processing Technology Office for supporting this research.

## References

Agogino, A., Stanley, K., & Miikkulainen, R. (2000). Online interactive neuro-evolution. *Neural Processing Letters, 11*, 29-37.

Brachman, R. (2002). *Systems that know what they're doing: The new DARPA/IPTO initiative in cognitive systems*. Speech given at the DARPATech 2002 Conference, 31 July 2002.

Buro, M. (2003). Real-time strategy games: A new AI research challenge. *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence* (pp. 1534-1535). Acapulco, Mexico: Morgan Kaufmann.

Chia, C.W., & Williams, K.E. (2003). A modified naïve Bayes approach for autonomous learning in an intelligent CGF. *Proceedings of the Conference on Behavior Representation in Modeling and Simulation*. Scottsdale, AR: SISO.

ESA (2004). Entertainment Software Association. [http://www.theesa.com]

Fasciano, M.J. (1996). *Real-time case-based reasoning in a complex world* (Technical Report TR-96-05). Chicago, Illinois: The University of Chicago, Computer Science Department.

Fogel, L.J., Porto, V.W., & Owen, M. (1996). An intelligently interactive non-rule-based computer generated force. *Proceedings of the Sixth Conference on Computer Generated Forces and Behavioral Representation*. Orlando, FL: University of Central Florida.

Forbus, K., Mahoney, J., & Dill, K. (2001). How qualitative spatial reasoning can improve strategy game AIs. In J. Laird & M. van Lent (Eds.) *AI and Interactive Entertainment: Papers from the AAAI Spring Symposium* (Technical Report SS-01-02). Stanford, CA: AAAI Press.

Geisler, B. (2002). *An empirical study of machine learning algorithms applied to modeling player behavior in a "first person shooter" video game*. Master's thesis, Department of Computer Sciences, University of Wisconsin, Madison.

Goodman, M. (1993). Projective visualization: Acting from experience. *Proceedings of the Eleventh National Conference on Artificial Intelligence* (pp. 54-59). Washington, DC: AAAI Press.

Guestrin, C., Koller, D., Gearhart, C., & Kanodia, N. (2003). Generalizing plans to new environments in relational MDPs. *Proceedings of the Eighteenth International Joint Conference in Artificial Intelligence* (pp. 1003-1010). Acapulco, Mexico: Morgan Kaufmann.

Hieb, M. R., G. Tecuci, Pullen, J.M., Ceranowicz, A., & Hille, D. (1995). A methodology and tool for constructing adaptive command agents for computer generated forces. *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*. Orlando, FL: University of Central Florida.

Houk, P.A. (2004). *A strategic game playing agent for FreeCiv*. Master's thesis (draft), Department of Computer Science, Northwestern University, Evanston, IL.

Kaminka, G.A., Veloso, M.M., Schaffer, S., Sollitto, C., Adobbati, R., Marshall, A.N., Scholer, A., & Tejada, S. (2002). GameBots: A flexible test bed for multiagent team research. *Communications of the ACM, 45*(1), 43-45.

Laird, J.E. (2001). It knows what you are going to do: Adding anticipation to a Quakebot. *Proceedings of the Fifth International Conference on Autonomous Agents* (pp. 385-392). Montreal, Quebec, Canada: ACM Press.

Laird, J.E., & van Lent, M. (2001). Interactive computer games: Human-level AI's killer application. *AI Magazine, 22*(2), 15-25.

Petty, M.D. (2001). Do we really want computer generated forces that learn? *Proceedings of the Tenth Conference on Computer Generated Forces and Behavioral Representation*. Norfolk, VA: SISO.

Ponsen, M. (2004). *Online and offline learning in computer games*. Master's thesis (draft), Department of Computer Science, University of Maastricht, Maastricht, The Netherlands.

RoboCup (2004). The RoboCup Soccer Simulator. [http://sserver.sourceforge.net/]

Spronck, P., Sprinkhuizen-Kuyper, I., & Postma, E. (2004). Online adaptation of game opponent AI with dynamic scripting. *International Journal of Intelligent Games & Simulation, 3*(1), 45-53.

Sweetser, P., & Dennis, S. (2003). Facilitating learning in a real time strategy computer game. In R. Nakatsu & J. Hoshino (Eds.) *Entertainment Computing: Technologies and Applications*. Boston, MA: Kluwer.

UCI (2004). The UCI Machine Learning Repository. [http://www.ics.uci.edu/~mlearn/MLRepository.html]

van Lent, M., & Laird, J.E. (1998). Learning hierarchical performance knowledge by observation. *Proceedings of the Sixteenth Int. Conference on Machine Learning* (pp. 229-238). Bled, Slovenia: Morgan Kaufmann.

Woodcock, S. (2002). AI roundtable moderator's report. *2002 Game Developer's Conference*. [http://www.gameai.com/cgdc02notes.html]

Zyda, M., Hiles, J., Mayberry, A., Wardynski, C., Capps, M., Osborn, B., Shilling, R., Robaszewski, M., & Davis, M. (2003). Entertainment R&D for defense. *IEEE Computer Graphics and Applications*, Jan./Feb., 28-36.