TIELT: A Testbed for Gaming Environments

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

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

Abstract

Many AI researchers want to test the utility of their systems in complex task environments defined by (e.g., real-time strategy) gaming simulators and/or simulators of computer-generated forces. Also, many developers of commercial and military gaming simulators seek behaviors that can be supported by these systems. However, these integrations require great effort. We will demonstrate the late Alpha version of TIELT, a testbed designed to fill these needs.

Motivation for TIELT

Several researchers have suggested using complex gaming simulators for AI research (e.g., Laird & van Lent, 2001). However, integrating AI systems with such simulators is effort intensive. Our goal with the Testbed for Integrating and Evaluating Learning Techniques (TIELT) is to reduce this effort, particularly for machine learning systems (Aha & Molineaux, 2004). With this vision in mind, and our goal to streamline the process of integrating learning-embedded decision systems with gaming simulators, we have the following specification for TIELT:

1. *Integration*: TIELT should input a description of the game's model and state, a description of the inputs required by the decision system, and a performance task. During a game, it should interpret a sequence of game states, translate them for input to the decision system, interpret the system's response (e.g., actions for controlling a game engine agent), and translate it for display or as input to the game engine. It should also support empirical studies on the decision system's utility for learning a given performance task(s).
2. *Learning*: TIELT should support investigations for learning at least 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 an action.
   c. *Game model*: TIELT could also be used to model aspects of the game's environment or its agents' behaviors (e.g., for prescribing response actions).
3. *Learning methods*: TIELT should work with supervised, unsupervised, analytic, and reinforcement learning methods. It should support online and offline training by providing systems with a stream of game state descriptions or recorded sessions for later training.
4. *Game engines*: TIELT should facilitate access to strategy (real-time and turn-based), role-playing, team sports, and first-person shooter games. These include games whose learning tasks have large hypothesis spaces and that can benefit from learned strategies.
5. *Reuse*: TIELT should permit researchers to easily study a decision 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 <game engine, decision system> integration to a new performance task, only that task requires specification; the game description and interface descriptions may remain unchanged.
6. *Availability to a wide range of platforms and programs*: TIELT should be available for use on all major platforms, and provide support for slower algorithms (e.g., for use with real-time game engines).

The TIELT System

TIELT is currently in a late Alpha stage of development. It is freely available (with an "as-is" license), and can be used on all systems supporting the Java Virtual Machine. It is downloadable from http://nrlsat.ittid.com, where we also provide documentation (e.g., User's Manual and Tutorial), solicit requests for additional functionality, and maintain a bug tracking facility.

A TIELT integration requires developing five knowledge bases (KBs) (see Figure 1):

Game Model: This encodes a declarative game description of objects in a game state, operators that describe a player's interaction with the game, game events that can occur, and state transition rules that govern how the game is played. Separating this information from the Game Interface Model permits viewing a game in different ways, frees a decision system from a game engine's timetable, and allows a single Game Model to be used for a category of games.

Game Interface Model: This defines how TIELT communicates with a game engine. It includes

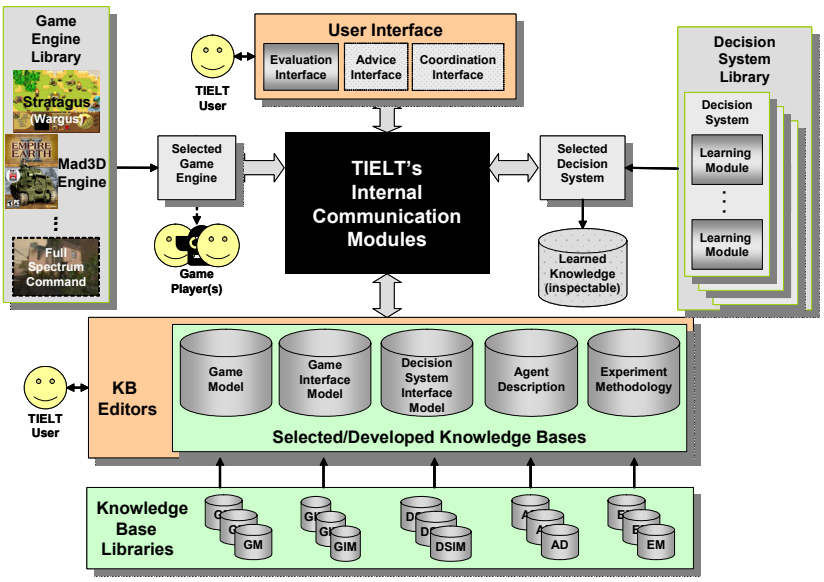

Figure 1: TIELT's Architecture

definitions of message templates that describe the parameters of a message and its connection to a Game Model. TIELT uses action messages from the decision system to affect the game, and percept messages from the game engine to inform the decision system. TIELT supports message communication via several protocols (e.g., imported Java classes, socket-based TCP/IP network connections, console I/O).

Decision System Interface Model: This uses message templates similar to those in the Game Interface Model and a dialogue model to define communication with the AI decision system.

Agent Description: This encodes an executable and hierarchical representation of simulator tasks that allows users to quickly identify the tasks to be performed by the decision system.

Evaluation Methodology: This allows researchers to define how to conduct an empirical evaluation. For example, users can command TIELT to train a decision system for a specified number of gaming sessions, and then test in a "non-learning" mode. Experimental results can be stored in databases supporting JDBC or ODBC standards for subsequent analysis.

Users define these KBs through TIELT's editors using its GUI and the simple TIELT Scripting Language. By accessing libraries of KBs, decision systems, and game engines that have been integrated with TIELT, users can integrate a decision system and compare its performance against selected alternatives on selected tasks.

TIELT has been downloaded by over 50 people and integrated with a small number of games, including Full Spectrum Command/Research (which is used for Army training purposes at Ft. Benning), the Mad3D engine (which is used in the popular Empire Earth II™), and Wargus, a Warcraft II™ clone implemented using the freeware Stratagus engine. These simulators are real-time strategy (RTS) games. TIELT has also been integrated with a few decision systems, including Soar and CaT, a case-based reasoner that learns to defeat Wargus opponents, as reported in TIELT's first published research investigation (Aha *et al*., 2005). This experience provided us with valuable feedback on requirements for successful use of TIELT in learning investigations (e.g., concerning game engine API, Game Model content, and evaluation methodologies). We are integrating TIELT with the General Game Playing (GGP) Testbed (http://games.stanford.edu) for use in the AAAI'05 GGP competition (Genesereth & Love, 2005), and, with Mad Doc Software, are developing challenge problems concerning TIELT integrations for use in workshops to be held at IJCAI'05 and ICCBR'05.

Demonstration

Our demonstration will show how TIELT can be used to define the five KBs and execute experiments involving the application of CaT to the RTS game Wargus.

References

Aha, D.W., & Molineaux, M. (2004). Integrating learning in interactive gaming simulators. In D. Fu & J. Orkin (Eds.) *Challenges in Game AI: Papers of the AAAI'04 Workshop* (Technical Report WS-04-04). San José, CA: AAAI Press.

Aha, D.W., Molineaux, M., & Ponsen, M. (2005). Learning to win: Case-based plan selection in a real-time strategy game. To appear in *Proceedings of the Sixth International Conference on CBR*. Chicago, IL: Springer.

Genesereth, M., & Love, N. (2005). General game playing: Overview of the AAAI competition. To appear in *AI Magazine.*

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