A Testbed for Evaluating AI Research Systems in Commercial Games

David W. Aha[1], Matthew Molineaux[1,2] and Philip Moore[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
<*first.last*>@nrl.navy.mil

Abstract

Many AI researchers want to test the utility of their prototypes in complex task environments, such as those defined by commercial gaming simulators. Also, many developers of commercial games need to solve tasks (e.g., game balancing, providing rational agent behaviors) that can be addressed by these systems. However, integrating them with gaming simulators requires substantial effort. We will demonstrate TIELT, a testbed designed to assist with evaluating research prototypes in these task environments.

System Specification

One of our goals in developing TIELT (Testbed for Integrating and Evaluating Learning Techniques) is to provide researchers with the ability to more easily evaluate research systems in comprehensive task environments; we hope that this will facilitate the development of agents that can address knowledge-intensive tasks. This contrasts with popular empirical studies of knowledge-poor machine learning techniques that use only static databases for their experiments. As argued by many researchers (e.g., Laird & van Lent, 2001), commercial games provide excellent environments for this purpose, and our work with TIELT to date has focused on these environments.

TIELT is also intended to provide practitioners with facilities for more easily evaluating the utility of AI research prototypes in their products. Game developers (e.g., Rabin, 2004) are aware of the potential of using certain AI technology in their systems, and wish to conduct such evaluations in an effort-efficient manner.

To fulfill these goals, TIELT has been designed to have the following characteristics:

1. *Integration functionality:* TIELT must support separate, interchangeable descriptions of game environments and agent programs. These descriptions are compatible with a variety of game genres (e.g., RTS, TBS, RPG, team sports, FPS), and a variety of learning techniques, including supervised, unsupervised, and reinforcement; and online and offline algorithms. Agents should be able to learn models of task, players, and environments. During a game, TIELT mediates communications between the environment and agent program.
2. *Evaluation support*: TIELT must input, represent, and execute a variety of empirical studies to test an AI system's utility for a range of tasks. Also, it allows users to run studies that compare the utility of multiple AI systems on multiple tasks.
3. *Use cases*: TIELT must support AI systems that embody either agent programs or agent subprograms (i.e., to study game environment subtasks).
4. *Multi-platform and multi-access*: TIELT must work on all major systems, and provide support for testing slower algorithms.

System Use

A TIELT integration requires a user to develop or reuse four knowledge bases (KBs) and design an experiment (see Figure 1). First, the *Environment Model* (EM) encodes a declarative description of objects in the game state, operators that describe the possible actions, and observable models that describe what an agent can perceive.

Second, the *Simulator Interface Description* (SID) defines how TIELT communicates with a game engine. It includes definitions of message templates, specifies an underlying protocol, and fulfills the contract with an agent program implied by an Environment Model. TIELT supports communication via several standard protocols (e.g., Java method calls, network protocols, console I/O). Separating this communication from the EM gives the agent an implementation-independent view of the game, frees the agent from the game engine's timetable, and allows an EM to be used for multiple games.

Third, the *Agent Program Interface Description* uses message templates similar to the SID's and a dialogue model to define communications with an agent program. Finally, an optional *Performer Description* (PD) is defined when the agent program is intended to address only a subtask within the environment. The PD encodes a hierarchical executable representation of environment tasks, any of which can be taken on by an agent program.

Given these, a procedure can be selected/defined for conducting an empirical evaluation. For example, users can command TIELT to train an agent program for a specified number of gaming sessions, and then test in a "non-learning" mode. Experimental results can be stored in a database for subsequent analysis.

Users define these KBs through the TIELT GUI, which provides a form fill-in interface for each KB, supplemented by the simple TIELT scripting language (TSL). The TIELT GUI has recently been rewritten to be more user-friendly, adopting characteristics of the popular Java Development

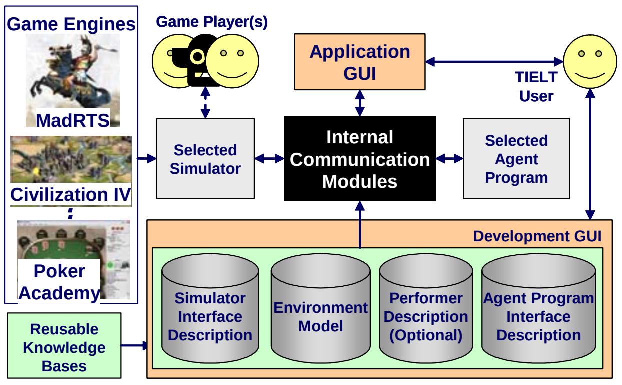


Figure 1: TIELT's Integration Architecture

Environment Eclipse (2006) (Figure 2). By accessing libraries of KBs, agent programs, and game engines that have been integrated with TIELT previously, users can integrate an agent program and compare its performance against selected alternatives on selected tasks.

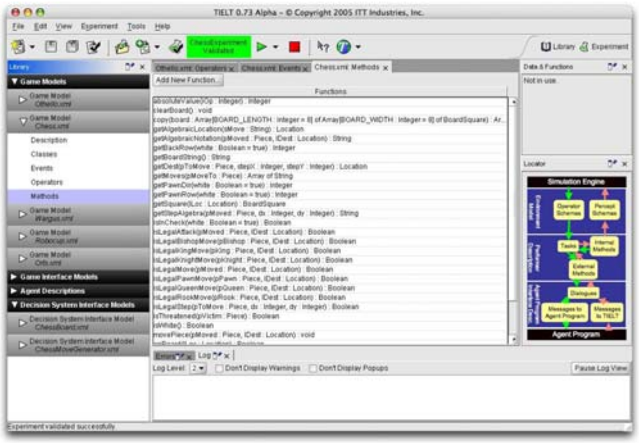
System Status and Planned Tasks

TIELT is freely available, documented, comes with an "as-is" license, and is supported via our www site (TIELT, 2006). It has been integrated with over a dozen gaming simulators, including classic board games (e.g., chess), Unreal Tournament™, a clone of Warcraft II™ named Wargus, and MadRTS – Mad Doc Software's adaptation of the engine that underlies Empire Earth II™.

Several agent programs have also been integrated with TIELT, including the SOAR cognitive architecture (Laird *et al.*, 1987) and CAT, a case-based planner that we designed for learning to defeat Wargus opponents (Aha *et al.*, 2005).

TIELT has been downloaded by 124 people from 68 organizations, including 18 commercial organizations. Its most significant use has been through the DARPA/IPTO Transfer Learning program, in which TIELT supports evaluations of state-of-the-art transfer learning technology. We are also supporting several academic and industry personnel on focused projects involving TIELT. As we release new challenge problems, we hope to attract more of the research community to TIELT. Additionally, TIELT will be used in the proposed reinforcement learning competition and benchmarking event to be held at the 2006 International Conference on Machine Learning.

While TIELT has begun to demonstrate utility in a few research investigations, it has not been adopted for serious use by many commercial game developers. However, based on work they performed during our collaboration, Mad Doc Software has plans to develop products for future release. We plan to continue making TIELT available for use by game developers, with the goal of providing a suite of agent programs that can be quickly and easily tested in a game as it is being developed. We expect that, if a technique proves useful, then the developers would reimplement and incorporate the agent's functionality in their product.



```
absoluteValue(iOp : Integer) : Integer
clearBoard() : void
copy(board : Array[BOARD_LENGTH : Integer = 8] of Array[BOARD_WIDTH : Integer = 8] of BoardSquare) : Ar...
getAlgebraicLocation(sMove : String) : Location
getAlgebraicNotation(pMoved : Piece, lDest : Location) : String
getBackRow(white : Boolean = true) : Integer
getBoardString() : String
getDest(pToMove : Piece, stepX : Integer, stepY : Integer) : Location
getMoves(pMoveTo : Piece) : Array of String
getPawnDir(white : Boolean = true) : Integer
getPawnRow(white : Boolean = true) : Integer
getSquare(lLoc : Location) : BoardSquare
getStepAlgebra(pMoved : Piece, dx : Integer, dy : Integer) : String
isInCheck(white : Boolean = true) : Boolean
isLegalAttack(pMoved : Piece, lDest : Location) : Boolean
isLegalBishopMove(pBishop : Piece, lDest : Location) : Boolean
isLegalKingMove(pKing : Piece, lDest : Location) : Boolean
isLegalKnightMove(pKnight : Piece, lDest : Location) : Boolean
isLegalMove(pMoved : Piece, lDest : Location) : Boolean
isLegalPawnMove(pPawn : Piece, lDest : Location) : Boolean
isLegalQueenMove(pQueen : Piece, lDest : Location) : Boolean
isLegalRookMove(pRook : Piece, lDest : Location) : Boolean
isLegalStep(pToMove : Piece, dx : Integer, dy : Integer) : Boolean
isThreatened(pVictim : Piece) : Boolean
isWhite() : Boolean
movePiece(pMoved : Piece, lDest : Location) : void
```

Figure 2: A Snapshot of TIELT's New GUI

Upcoming extensions to TIELT include speed enhancements from compiling TSL scripts to Java, a system for packaging and installing agents and environments, and the addition of wizards to help with common tasks.

Demonstration

Our demonstration shows how TIELT can be used to integrate a gaming simulator and an agent program, and to define and perform experiments on them. This will involve defining TIELT's four knowledge bases and an experiment procedure. A version of this demonstration can be found at http://nrlsat.ittid.com/ISD05MolineauxM.html.

Acknowledgements

TIELT is being developed through sponsorship of the DARPA/IPTO Transfer Learning program. We gratefully acknowledge DARPA's support for this project. The views, opinions, and/or findings contained in this paper are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

References

Aha, D.W., Molineaux, M., & Ponsen, M. (2005). Learning to win: Case-based plan selection in a real-time strategy game. *Proceedings of the Sixth International Conference on CBR* (pp. 5-20). Chicago, IL: Springer.

Eclipse (2006). [http://www.eclipse.org]

Laird, J.E., Newell, A., & Rosenbloom, P.S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence* **33**(3), 1-64.

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

Rabin, S. (Ed.) (2004). *AI game programming wisdom 2.* Hingham, MA: Charles River Media.

TIELT (2006). Testbed for integrating and evaluating learning techniques. [http://nrlsat.ittid.com]