Intelligent Agents in Computer Games

**Michael van Lent, John Laird, Josh Buckman, Joe Hartford,
Steve Houchard, Kurt Steinkraus, Russ Tedrake**

Artificial Intelligence Lab
University of Michigan
1101 Beal Ave.
Ann Arbor, MI 48109
vanlent@umich.edu

As computer games become more complex and consumers demand more sophisticated computer controlled opponents, game developers are required to place a greater emphasis on the artificial intelligence aspects of their games. Our experience developing intelligent air combat agents for DARPA (Laird and Jones 1998, Jones at al. 1999) has suggested a number of areas of AI research that are applicable to computer games. Research in areas such as intelligent agent architectures, knowledge representation, goal-directed behavior and knowledge reusability are all directly relevant to improving the intelligent agents in computer games. The Soar/Games project (van Lent and Laird 1999) at the University of Michigan Artificial Intelligence Lab has developed an interface between Soar (Laird, Newell, and Rosenbloom 1987) and the commercial computer games Quake II and Descent 3. Techniques from each of the research areas mentioned above have been used in developing intelligent opponents in these two games.

The Soar/Games project has a number of goals from both the research and game development perspective. From the research perspective, computer games provide domains for exploring topics such as machine learning, intelligent architectures and interface design. The Soar/Games project has suggested new research problems relating to knowledge representation, agent navigation and human-computer interaction. From a game development perspective, the main goal of the Soar/Games project is to make games more fun by making the agents in games more intelligent. If done correctly, playing with or against these AI agents will more closely capture the challenge of playing online against other people. A flexible AI architecture, such as Soar, will also make the development of intelligent agents for games easier by providing a common inference engine and reusable knowledge base that can be easily applied to many different games.

Quake II and Descent 3, both popular first person perspective action games, include software hooks allowing programmers to write C code that can access the game's internal data structures and agent controls. This has allowed us to extract symbolic information from the games without interpreting the image displayed on the computer screen. A common approach to building intelligent agents in computer games is to use C code and these programming hooks to control agents via a large number of nested if and switch statements. As the agents get more complex, the C code becomes difficult to debug, maintain and improve. A more constrained language that better organizes the conditional statements could be developed but we believe that language would be similar to the Soar architecture. By using the Soar architecture, we are taking advantage of the Soar group's 15 years of research into agent architectures.

Soar serves as the inference engine for the intelligent agent (see figure 1). The job of the inference engine is to apply knowledge to the current situation and decide on internal and external actions. The agent's current situation is represented by data structures representing the states of simulated sensors implemented in the interface and contextual information stored in Soar's internal memory. Soar allows easy decomposition of the agent's actions through a hierarchy of operators. Operators at the higher levels of the hierarchy explicitly represent the agent's goals, while the lower level operators represent sub-steps and atomic actions used to achieve these goals. Representing goals explicitly in internal memory encourages agent developers to create goal directed agents. Soar selects and executes the operators relevant to the current situation that specify external actions, the agent's moves in the game, and internal actions, such as changes to the agent's internal goals. Soar constantly cycles through a perceive, think, act loop, which is called the decision cycle.

1. Perceive: Accept sensor information from the game
2. Think: Select and execute relevant knowledge
3. Act: Execute internal and external actions

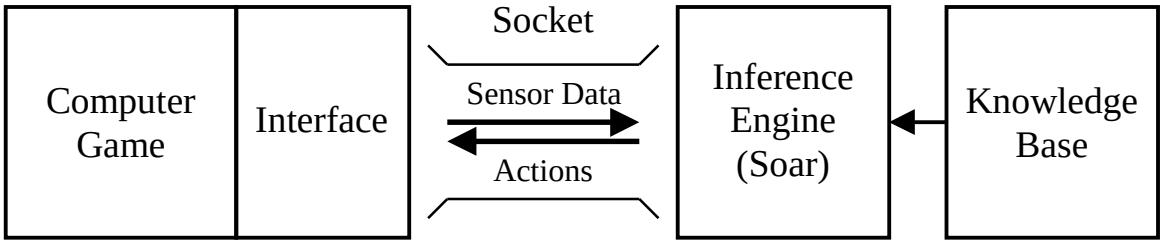

Figure 1: Soar is attached to the computer game through a socket connection to an interface that is compiled into the computer game.

One of the lessons learned, as a result of the DARPA project and the Soar/Games project, is the importance of carefully designing the interface between the inference engine and the simulated environment. The interface extracts the necessary information from the game and encodes it into the format required by Soar. Each game requires a custom interface because the details of the interaction and the content of the knowledge extracted vary from game to game. For example, Descent 3 agent's, flying in a spaceship without gravity, must have the ability to move and rotate in all six degrees of freedom. Quake II agents, running normally with gravity, require only four degrees of freedom. However, basing each interface on a common template allows much of the knowledge developed for one game to be reused in other games.

The Soar/Games project uses the standard Soar knowledge representation of a hierarchy of operators each implemented by multiple production rules. The operators at the top level of the hierarchy represent the agent's general goals or modes of behavior. For example, the top-level operators in a Quake II or Descent 3 agent might include attack, explore, retreat and wander. The lower levels of the hierarchy represent successively more specific representations of the agent's behavior. Sub-operators of the top-level attack operator could include different styles of attacking, such as pop-out-attack or circle-strafe, or steps followed to implement an attack, like select-attack-type and face-enemy. The operators at the bottom of the hierarchy are atomic steps and actions that implement the operators above, such as shoot, move-to-door and stop-moving. The Quake II agent currently under development consists of a five level operator hierarchy containing 57 different operators implemented with more than 400 production rules. Our hope is that many of these rules can be reused in the development of a Descent 3 agent. Because Quake II and Descent 3 are the same genre of games, they share many similarities at the strategic and tactical levels. We hope to take advantage of this by creating a game independent, genre specific knowledge base used by both games.

The game portion of our demonstration consists of six workstations (200MHz or faster Pentium machines), three for the Quake II demonstration and three for Descent 3. For each game one workstation runs the game server and AI system, a second displays the ongoing game from the agent's perspective and audience members can play the game against the AI agent on the third. In addition to understanding how the research has resulted in valuable concepts and how those concepts are used, the audience will also be able to evaluate the effectiveness of the concepts by playing the games. Both games are easily understood, action oriented and visually impressive, which leads to an accessible, exciting demonstration of applied artificial intelligence research.

Acknowledgements

The authors would like to thank Outrage Entertainment Inc. for allowing us to work with Descent 3 while in development and Intel for the donation of machines.

References

Laird, J. E. and Jones, R. M. 1998. Building Advanced Autonomous AI systems for Large Scale Real Time Simulations. In *Proceedings of the 1998 Computer Game Developers' Conference,* 365-378. Long Beach, Calif.: Miller Freeman.

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

Jones, Randolph M., Laird, John E., Nielsen, Paul E., Coulter, Karen J., Kenny, Patrick. and Koss, Frank V. 1999. Automated Intelligent Pilots for Combat Flight Simulation. AI Magazine, 20(1):27-41.

van Lent, M. C. and Laird, J. E. 1999. Developing an Artificial Intelligence Engine. In *Proceedings of the 1999 Game Developers' Conference,* 577-587. San Jose, Calif.