An Approach to Directing Intelligent Agents in Real-time

Paul Scerri and **Nancy Reed** and **Anders Törne**
Real-time Systems Laboratory
Department of Computer and Information Science
Linköpings Universitet, S-581 83 Linköping, Sweden
pausc,nanre,andto@ida.liu.se

Abstract

We describe an approach to directing agents in real-time that is similar to Blumberg and Galyean (1995). The approach is being applied to create automated pilots for a beyond visual range flight simulator. In this domain it is necessary that a human controller can control an agent at all levels of abstraction. We believe a multi-agent action selection mechanism provides a natural way of achieving this goal.

Introduction

We describe an ongoing project to build intelligent agents that can be directed in real-time. The agents are designed for a real-time simulation environment where a human controller must be able to take control of particular aspects of an agents behavior while the agent continues to act. Two challenging aspects of the problem are that the agent will be expected to continue on with other tasks as well as fulfilling the operators request and that the human operator should be able to make requests at all levels of abstraction.

The agents implement simulated pilots for a beyond visual range simulator called TACSI, developed by Saab Missions and Systems. The TACSI simulator is used to test and evaluate systems and to train pilots. Human pilots may interact with simulated pilots in the environment. In this domain it is often important for the human tester to be able to direct the agents during the execution of a scenario. Sometimes the directions of the tester need to be low level and should override other current tasks of the agent, for example, to change heading five degrees. At other times the directions will be at a high level of abstraction and should be smoothly integrated with the other actions of the agent, for example to attack a particular ground station.

Agent System Design

We are currently implementing a multi-agent system for action selection in the simulated pilot. The agents are arranged hierarchically with agents at the top of the hierarchy responsible for more abstract tasks, and agents lower in the hierarchy responsible for negotiating values for outputs of the simulated pilots effectors. Each agent can be controlled internally by any type of decision-making mechanism. A current prototype supports only state-machines for this purpose.

In the new system, higher level agents *contract* lower level agents to fulfill specific parts of their tasks. The hierarchy need not have a single root, i.e. there may be a number of separate hierarchies whose bottom agents negotiate with each other to decide on the output of the effectors. Each hierarchy will be responsible for one aspect of the agents overall behavior, for example there may be a hierarchy for carrying out a bombing raid and another for maintaining the safety of the aircraft. Coordination between the hierarchies is facilitated *only* by negotiations at the bottom level.

The negotiation is framed as a constraint satisfaction problem and solved in a distributed multi-agent manner via a mechanism as described by Ghedira (1994). The agents use a priority system whereby agents with higher priority have more weight in negotiations over effector output. In particular, if agents have incompatible needs, higher priority agents have their needs addressed first. The priority of an agent is a function of three factors: the priority of the agent that contracted this agent; the relevance of the agent in the light of the current environment; and the intrinsic importance of the agent (specified by the designer). For example an agent contracted by a high priority agent but irrelevant in the current situation will have a low overall priority, conversely an agent with high intrinsic importance such as one responsible for some aspect of the aircrafts safety and relevant in the current situation will have a relatively high priority.

Discussion

An underlying multi-agent representation seems to allow a natural method for supporting directability at all levels of abstraction. The idea is that the human controlling the simulated pilot can create new agents at run-time and they will negotiate with other agents in the multi-agent controller to produce an output that is a combination of "normal" agent behavior and user control. To direct a simulated pilot to do a new high-level task, the user dynamically creates a new high-level agent. To direct a pilot to do a low-level task a new

low-level agent is dynamically created. By appropriately setting the priority of the newly created agents, the user's direction can override some or all of the other aspects of the agents behavior. For example, if the aircraft should be forced to crash into the ground, a new low-level high-priority agent should be created that will override other agents, i.e. those for avoiding the ground, in negotiations about the effector output. In contrast, a task such as attacking a particular ground station would require the creation of a high level, low priority agent that will not override agents such as those concerned with the safety of the aircraft.

A complementary way of directing a simulated pilot is to kill (or lower the priority of) one of the agents in the simulated pilots controlling system. This would mean that the simulated pilot would no longer attempt to achieve a specific task. For example the user could kill the agent responsible for attacking a particular ground station allowing the simulated pilot to continue with other parts of its mission. Notice also that agents can be killed at different levels of abstraction.

The appealing aspect of this approach to directability is that the action selection system does not need to distinguish between user directions and built in "intelligence". Because direction and normal action selection mechanisms are so closely coupled there should be a smooth integration of the overall output of the agent. The priority system ensures that only parts of a simulated pilots overall behavior are overridden.

Blumberg and Galyean's (1995) system has similar properties, however it uses different mechanisms for control at each different layer. Hayes-Roth et al(1998) describes a system where children can direct *animated puppets.* The system proposed here allows far more control over pilots than a child has over the animated puppet, although with much higher complexity.

Future Work

Although the approach we discuss allows human users to direct an agent, the same mechanism can be applied to allow agents to direct other agents. As with human direction, the directions given can be at varying levels of abstraction and carry varying weights. This would be achieved by allowing simulated pilots to create or destroy agents that negotiated within hierarchies of other simulated pilots. If a simulated pilot creates a high level, low priority agent for another simulated pilot, the effect will be to have the second simulated pilot perform the task if, and only if, it fits with other current tasks. Alternatively if a simulated pilot creates a high level, high priority agent for another agent, it will effectively be an "order" the second agent must perform. This approach could be used to create agents with adjustable autonomy. Cremer (1995) describes a system using Hierarchical Communicating State Machines (HCSMs) which also allows control of agents by other agents, however in that case a direction can only be taken as an order.

This approach to directability gives the tester a high degree of control, however they need tools to visualize the details of the multi-agent structure in order to know which new agents need be created or existing agents destroyed. Even with an effective visualization tool it would be difficult to control a large number of simulated pilots by creating and destroying single agents. Future work could extend the approach by automating the process of determining which agent to create or destroy. An obvious example would be to create a mapping between human pilot terminology and underlying agents so that a high level request for a particular action resulted in the appropriate agent being created or killed and hence the simulated pilot taking the appropriate action. For example a verbal command "Move in tight!" may result in an existing team formation agent being killed and a new agent, designed for tight formations, created instead. The Tac-Air-Soar system (Jones (1998)) uses the Soar computation model and allows directability at an abstract level via verbal commands.

At a more abstract level, one might map between actions and agents for an entire team of simulated pilots. For example, a team leader might want to change from a "routine flying" formation to an "attack" formation when close enough to a target on the ground. Then one or two pilots could be directed to perform the attack. By creating agents for each team member to accomplish its part of the formation or maneuver, we aim to extend this approach to include team-level directability.

Acknowledgements

This work is supported by Saab AB, Missions and Systems and by NUTEK grants IK1P-97-09677 and IK1P-98-06280.

References

Blumberg, B. M., and Galyean, T. A. 1995. Multi-level direction of autonomous creatures for real-time virtual environments. In *Siggraph '95 Proceedings*, 295–304. New York: ACM Press.

Cremer, J.; Kearney, J.; and Papelis, Y. 1995. HCSM: A framework for behavior and scenario control in virtual environments. *ACM Transactions on Modeling and Computer Simulation* 242–267.

Ghedira, K. 1994. A distributed approach to partial constraint satisfaction problems. In *6th European Workshop on Modelling Autonomous Agents in a Multiagent World, MAAMAW'94*. Springer Verlag Lecture Notes in Artificial Intelligence. 106–122.

Hayes-Roth, B.; Brownston, L.; and van Gent, R. 1998. Multiagent collaboration in directed improvization. In *Readings in Agents*. San Francisco: Morgan Kaufmann. 141–147.

Jones, R.; Laird, J.; and Nielsen, P. 1998. Automated intelligent pilots for combat flight simulation. In *Proceedings of the Tenth National Conference on Innovative Applications of Artificial Intelligence (IAAI-98)*, 1047–1054. Menlo Park, California: AAAI Press.