Copycat: A cooperative gaming environment for programming adversarial agents by children

Omid Banyasad

Faculty of Computer Science, Dalhousie University, Halifax, Canada

Abstract

This paper proposes a control model and an animated gaming/programming environment for creating programmable agents in an adversarial game environment. The paper proposes the use of Moore machines for modeling the behaviours of agents. The agents are to be trained by using Programming by Demonstration techniques when a trainer and an agent cooperate to play the game in a training mode. We also discuss the importance of the unification of the trainers' and the agent's perception of the game environment while cooperating in the training mode.

1 Introduction

Many adversarial game environments consist of programmed agents that play against human players. Such agents, typically, execute certain strategies to outperform human players and often become predictable after careful observation of their behaviours.

It is expected that a gaming environment in which users can customize and train the agents would be appealing to children. The training of such agents can be achieved through Programming by Demonstration and is the focus of this proposal.

Programming by Demonstration is the use of concrete examples to train an agent in a visual environment. The agent records the actions of the programmer and generalizes them using various generalization techniques to behave in a fashion analogous to the player's actions under circumstances similar to but more general than those of the example.

There has been substantial work done in developing animated and cartoon-like programming environments to increase children's interest in programming, or provide them with simple interactive, yet powerful educational tools [3,4,6,7]. In this work, one of our objectives is in the opposite direction: that is, we aim to develop a gaming/programming environment in which children program agents with a variety of strategies without being aware of the programming process. It is our goal to keep the programming environment and the programming process both simple and transparent to children. We expect to achieve this transparency by generalizing the actions of the trainer when the agent and the trainer cooperate in playing the game in a training mode.

In [1] we proposed a programming environment for autonomous robots based on the subsumption control model [2]. A prototype of our proposed environment was demonstrated at the Visual End User workshop held in conjunction with the 2000 IEEE Symposium on Visual Languages in Seattle. In another work, we have developed a 3D game environment for investigating an optimal level of recursive modeling of opponent agents in an adversarial game environment [5]. It is anticipated that a simplified version of the proposed control model according to [1] and the 3D game environment of [5], combined with a modified version of the user interface of [1] provide the means to develop an interactive game environment for creating custom agents and training them with a variety of strategies.

2 Environment

The game environment will consist of a 3D maze-like arena with tunnels and a variety of randomly placed walls. The goal of the game is to search for and shoot the opponent players. To allow for more interesting strategies, each player and agent will have imperfect knowledge of the game world.

2.1 Sensors

Each agent is to be equipped with at least one sensor. A sensor collects relevant information in its operating range and provides the information to the control model for further processing and decision making. Possible sensors for an agent are as follows.

1. A vision sensor enables an agent to "see" objects. The direction, depth, and field of view of a vision sensor can be set to any values within their valid ranges.
2. A sound sensor enables an agent to "hear" a firing weapon anywhere within its circular range. The direction of sound is to be computed relative to the direction that agent is facing. The range of the sensor can be set to any value within a range of possible values.
3. A bumper sensor allows an agent to "feel" when it bumps into an object, or when another agent bumps into it.

2.2 Actuators

An agent can be equipped with a number of actuators. Following can be a list of possible actuators:

1. A moving motor provides mobility for the agent. A moving motor moves the agent forward or backward.
2. A rotational motor enables an agent to rotate around its centre in clockwise or counterclockwise directions.

2.3 Weapons

An agent can be armed with a selection of weapons. The player can select weapons from a set of choices such as guns, shotguns, missiles or other conventional weapons in adversarial games. Some restrictions on the combination of weapons may apply.

3 Agent Model

An agent is a tuple $\mathcal{A}$ =(**Body**,**Sensors**,**Actuators**,**Weapons**,**Machine**). **Body** contains the geometrical characteristics of the agent such as its shape and size. **Sensors** and **Actuators** are two lists of sensor and actuator devices, respectively. **Weapons** is a list of the agent's weapons. **Machine** is a Moore machine that models the behaviour of the agent. The input to **Machine** is the list of values of the individual sensors in **Sensors**. The output of **Machine** is the list of values of the individual actuators and weapons of the agent.

A player can create a new agent by selecting a body, a number of sensors, actuators, and desired weapons from their corresponding sets . The player can also set various parameters of selected sensors, actuators and weapons. These parameters may include range, depth, speed, and the amount of ammunition available to a particular weapon. A newly created agent has no behaviour; however, an agent can be trained to behave in a certain way through demonstration. The training process will be cooperative: that is, under different conditions, either the trainer or the agent is in control of the agent's actuators and weapons.

4 Training

The game can be run in two different modes; *play* and *training*. In play mode, the player fights against his or her opponent(s) as in any other gaming environment.

Once a new agent is created, it can be trained. In the training mode, the trainer cooperates with an untrained agent. There are two major differences between play and training modes. In training mode, when the agent encounters an unknown condition in the state of the game (by reading its sensor values) the system pauses the execution of the game. At this point, the player can take control of the agent's actions. Once the player defines a new action for the agent, the game is resumed. The agent monitors the state of the game. If there is a previously defined action for the current state of the sensor values, the agent takes control back and acts accordingly. Otherwise, the game is suspended again, and control given back to the trainer. During this process, the system constructs the agent's control Moore machine according to the trainer's actions and expands it as the game proceeds. In short, in the training mode the agent is being switched between automated and manual mode. In the manual mode, the player teaches the agent how to respond to the changes in its sensor values. In the automated mode, the agent executes its training in response to the changes in its sensor values.

The second major difference between the play mode and the training mode is that in the training mode the information collected by the agent's sensors must be presented to the user in a way that prevents any inconsistencies between the human player's perception of the state of the game and that of the agent. For example, when an agent is equipped with a vision sensor and the sensor sees an opponent, the only information that the sensor provides to the control model is the existence of an opponent in its field of view. However, the trainer can actually see and estimate the distance and the direction of the opponent. Such information available to the trainer and not to the agent may lead the trainer to force the agent to make premature decisions. This may also occur in the opposite direction. For instance, a sound sensor can tell the control model the occurrence of gunfire in the agent's vicinity as well as the direction of its source. If the hardware system is not equipped with a high-end sound system or the game environment does not support stereo sound, the direction of the sound would not be perceivable by the human player. Hence, in the training mode, it is essential to prevent such inconsistencies between the player's perception of the state of the game and those of the agent.

5 Concluding remarks

We propose a control model and the basics of a user interface for programming agents in an adversarial game environment. The proposed system provides a randomly generated arena and a set of body, sensor, actuator and weapon components for creating custom agents. The behaviours of agents are defined by players. The process of programming an agent is performed by demonstration when the trainer cooperates in playing the game with an agent.

There are two important aspects of this proposal. First the programming process of an agent is cooperative, with control switching back and forth between the trainer and the agent. Second, the training is performed from the agent's view point which in turn requires the unification of the player's and the agent's perception of the game world in the training mode.

6 References

[1] O. Banyasad, *A Visual Programming Environment for Autonomous Robots*, MCompSci Thesis, Dalhousie University (2000). ftp://borg.cs.dal.ca/pcox/banyasad.pdf

[2] R. A. Brooks (1986). A Robust Layered Control System For a Mobile Robot. *IEEE Journal of Robotics and Automation*, RA-2, pp. 14-23

[3] J. Gindling, A. Ionnidou, J. Loh, O. Lokkebo, A. Repenning(1995). LEGOSheets: A Rule-Based Programming, Simulation and Manipulation Environment for the LEGO Programmable Brick. *Proceedings of the 11th IEEE Symposium on Visual Languages*, pp. 172-179

[4] K. Kahn, Seeing systolic computations in a video game world, *Proc. 1996 IEEE Symposium on Visual Languages*, IEEE Computer Society Press, Los Alamitos, CA (1996), pp 95-101.

[5] J. MacInnes, O. Banyasad, and A. Upal, Watching You, Watching Me, *Proceedings of the 14th Canadian Conference on Artificial Intelligence*, Ottawa (June 2001), pp 361-364.

[6] A. Repenning. Bending the Rules: Steps Toward Semantically Enriched Graphical Rewrite Rules. *Proceedings of 1995 IEEE Symposium on Visual Languages*, pp. 226-234

[7] D.C. Smith, A. Cypher and J. Spohrer, KidSim: Programming Agents Without a Programming Language. *Communications of the ACM*, vol. 37, (1994) pp. 54-68.