Game-engine-based Cooperative Design Agent

Integration of Antagonistic Approach of AI, Symbolic & Behavioural, in Adaptive Virtual World Development

Alpha Wai Keung Lee

Multimedia Innovation Centre, Hong Kong Polytechnic University, China

mcalpha@polyu.edu.hk, www.micn.polyu.edu.hk/~mcalpha

This research focuses on the synergistic approach in coupling symbolic and behavioural AI in cooperative design agent development in an adaptive virtual world for design collaboration, based on the connotation of AI-engined design agents as design collaborators. Recent agent development, such as adaptive hypermedia and adaptive web, cooperation algorithm, Python scripting and SQL database technology, and their influence in the proposed game-engine-based cooperative design agent model are discussed. The emphasis of this research is not only limited to virtual world creation, but also the efficiency in prototyping and integration of industrial standards.

Adaptive Virtual World, Cooperative Design Agent, Artificial Intelligence, Game

Intelligent Interface Agents

Debates on the intelligent interface design development falls mainly into two different traditional and antagonistic categories of artificial intelligence: symbolic, in which hierarchical components predominate, against behavioural, where the meshwork elements are dominant (Maes, 1993).

A symbolic AI program would attempt to create a model of the application as well as a model of the working environment, including a model of an idealized user, and then make these models available in the form of rules or other symbols to the agent. Behavioural AI, on the other hand, gives the agent only the ability to detect patterns of behaviour in the actual user, and to interact with the user in different ways so as to learn not only from his or her actual behaviour, but also from feedback that the user gives it (Maes and Kozierok, 1993). One drawback is that, given that the agent has very little knowledge at the beginning of a relationship with a user, it will be of little assistance until it learns about his or her habits. One solution is to increase the amount of meshwork in the mix and allow agents from different users to interact with each other in a decentralized way (Lashari et al. 1994). Agents may aid one another in coping with novelty. Knowledge gained in one part of the world can be shared, and new knowledge may be generated out of the interactions among agents (De Landa, 2001).

Agents in Game

One stream of recent agent development in game focuses on building computer-controlled character that display human-like intelligence and behaviour. Under the area of AI research, the above character is known as synthetic characters. Frank and Stern (1998), in their work on Virtual Petz, described computer animated animal characters that exhibit human-like behaviour and emotions based on their interaction with the human user and other virtual Petz. Instead of having specific, predefined ways for a human to play the game or achieve game objectives, how a human interacts and behaves in the virtual world can affect synthetic characters and consequently change or create new situations and outcomes.

Laird and Jones (1998) describe and application of their SOAR AI architecture in a large-scale simulated war games. SOAR is a rules-based system which drives the decision making. Its basic knowledge of the world was carefully constructed using information from domain experts. Funge (1998) advocates an expert system approach to game AI by implementing a formal method of knowledge representation and having it separate from the decision making mechanism that controls a synthetic character.

Cooperative Agents

We shall say that several agents are cooperating, or that they are in a cooperation situation, if one of the following two conditions is verified: The addition of a new agent makes it possible to increase the performance levels of the group differentially; The action of the agents serves to avoid or to solve potential or actual conflicts (Ferber 1999). 5 cooperative agent algorithm, namely static, playbook, responsibility, autonomous, and perfect cooperation, are developed during the development of Microsoft Baseball 3D (Rabin, 1998).

Static cooperation comes from the idea that the same agent will always have the same jobs. The disadvantage is that the cooperation has no depth and cannot adapt to difficult situations, and it might need a manager entity to keep AI agents in line. Playbook cooperation comes from the idea that every situation can be looked up in a manual and each agent's job would be listed. Responsibility cooperation comes from the idea that every agent takes some responsibilities for their actions. This algorithm would have all the advantages of the static cooperation algorithm, but also have abandoned jobs taken care by other agents. Autonomous cooperation comes from the traditional AI concept that each agent independently comes to a conclusion about what he needs to do by observing the world around him. The disadvantage is that the consequence of simulating how humans make decisions based on observation is an overwhelmingly complex task.

Rabin (1998) suggests the requirements for a perfect cooperation are as follows: The best agent for the job always gets it; there is no duplication of jobs only one agent should have; higher priority jobs are always taken care of; lower priority jobs are sometimes sacrificed; there is a minimal amount of look-up data, which reduces data entry and the chance of errors or typos; simple

Cooperative Design Agent Model

In multi-agent systems, there is a continuum between the pure reactive (behavioural) agent which reacts only to stimuli, and the entirely cognitive (symbolic) agent which has a symbolic model of the world which it updates continually and on the basis of which it plans all its actions. Current interest lies, on the one hand, in trying to construct cognitive agents based in reactive organisations, and on the other hand, in creating agents which have both cognitive and reactive capacities at the same time (Ferber 1999).

Reynolds (2000) suggests the use of autonomous agents based on steering controllers under the direction of a simple mental model which mediates between several conflicting behavioural goals. In his model, agents respond in real time to the user's intention, as well as to each other and their environment. The implementation is based on a simple physical model for the agents, a reactive behavioural model composed of multiple behavioural states, steering behaviours.

The author has developed a prototype of cooperative design agent model in adaptive virtual world (Fig.1) based on the above notions of integration of symbolic and behavioural AI, perfect cooperation algorithm, and reactive behavioural model composed of multiple behavioural states and steering behaviours. Virtual world evolves as a result of the shift of agent model of one or more cooperative design agents to adapt to a given situation: user requirement, problem to solve, agent knowledge accumulated. During the interaction, non-participating agents become idle, waiting for interaction in another situation. Depending on the result of the shift of the model, agents cooperate to solve the problem either by letting the one sole best agent for the action to be taken, or by further interaction amongst agents to find the best agent for the purpose and scarify the lower priority jobs.

Research Method

One difficulty of virtual world development lies in team member communication and limited prototype testing due to the computing and coding intensive nature of the platform.

Virtools Dev, Behaviour Server and Multi-user Pack

Virtools Dev, Virtools Behaviour Server and Virtools Multi-user Pack are used to deploy and publish interactive virtual world in a distributed environment via the WWW. During the system construction, multiple team members can work simultaneously, with modifications instantly taken into account. Enabled with ODBC and SQL connectivity, it allows prototyping so that we can build interactive and realistic 3D environments while testing and evaluating the virtual environment from the very start to avoid delay.

Lua and Python Scripting

C++ programming is burdened with many restrictions and inconveniences, 1). Manual memory management. 2). Link phase: C++ modules are linked together so that function addresses don't need to be resolved at run-time. 3). Lack of introspection (Dawson, 1998). The emphasis in C++ is run-time performance (Stroustrup 1994). C++ is static, scripting languages are dynamic. This means that C++ runs fast, but scripting languages let you code faster. Python is chosen because it has more extension modules than Lua, more documentation, and being stackless, it is a promising way to create micro-threads for object AI.

References

Dawson, B.: 2002, *Game scripting in Python*, Proceedings of the Game Developers Conference, CD-ROM,

De Landa, M.: 2001, *Meshworks, hierarchies, and interfaces*, In The Virtual Dimension - Architecture, Representation, and Crash Culture, ed. John Beckman, Princeton Architectural Press.

Frank, A. and Stern, A.: 1998, *Multiple Character Interaction between Believable Characters*, Proceedings of the Game Developers Conference

Laird, J.E. and Jones, R.M.: 1998, *Building advanced autonomous AI systems for large scale real time simulations*, Proceedings of the Game Developers Conference, 365-377, San Jose: Gama Network.

Lashari, Y., Metral, M. and Maes, P.: 1994, *Collaborative interface agents*, Proceedings of 12th National Conference on AI, AAAI Press, 444-449.

Maes, P.: 1993, *Behavior-based artificial intelligence, from animals to animats*, MIT Press, (2) 3.

Maes, P. and Kozierok, R.: 1993, *Learning interface agents*, Proceedings of AAAI '93 Conference

Rabin, S.: 1998, *Making the play: team cooperation in Microsoft baseball 3D*, Proceedings of the Game Developers Conference, 543-557. San Jose: Gama Network.