AI and Human Player Cooperation in RTS Games

Andreas Stiegler
Stuttgart Media University,
Germany
stiegler@hdm-stuttgart.de

Daniel Jack Livingstone
University of the West of Scotland,
Scotland
daniel.livingstone@uws.ac.uk

ABSTRACT

The challenges associated with implementing cooperation between Artificial Intelligence players and human players cover some interesting areas of Artificial Intelligence research. One of the key requirements to efficient cooperation is good communication, to share information on current state and future plans.

Keywords

Artificial Intelligence, Game Development, Semantic Structures, StarCraft

1. Communication

We can consider two forms of communication: unidirectional and bidirectional. Unidirectional communication means that only one party can actively promote ideas or issue orders, while other parties can only follow the instructions or perhaps use a simple response, to confirm or deny, accept or reject. Although even this is strictly speaking bi-directional, we consider the main flow of communication as being unidirectional here, as most importantly only one party can actively propose new plans and trigger new dialogs for communication, while other parties are only reacting on requests. For many game genres, unidirectional communication may feel very natural. In the case of a first person shooter, for example, where the player is often the leader of a special unit or a captain of some kind, issuing orders to their teammates. Such hierarchical structures promote unidirectional communication and, with some creative mission scripting wrapped around it to blur the deficits, can produce a very enjoyable game experience.

Yet, there are also genres where such a unidirectional communication is insufficient. One obvious example are Real Time Strategy games. Here, a player usually embodies a general of some kind, leading an army to victory. Any AI allies, however, are also generals and would be unlikely to simply follow the orders of the player. When two human players would play such a game together, they would probably talk about their strategy and propose ideas. That's a bidirectional communication, where all parties can actively trigger communication, can propose ideas and can respond to the ideas of other players with complex feedback. Following the old motto of "do it right or don't do it at all", we don't see such bidirectional communication with AI players in many games, as there are some difficult challenges involved. We are working on solutions to these problems and this demo will show an early proof of concept.

2. SEMANTIC UNDERSTANDING

Implementing bidirectional communication, where players can issue commands and requests to an AI player and vice versa has to overcome a number of challenges. One such challenge is the huge semantic gap between the AI players and a human player. A human player enters the game with several decades of life experience and probably with a bunch of expectations from other games of the same genre or actual knowledge about this specific game from previous game rounds [1] [2] [3]. The AI, on the other hand, may not even be a single entity, but a cluster of agents and scripts that care for different subsets of the game [4] [5], for example goal identification, economy management or tactical unit handling. During communication, a human player can - and is likely to - link low level tactical and high level strategic aspects together to form semantically rich statements and commands. There may be no single agent on the AI side that qualified to respond, as the AI has limited semantics and perhaps little understanding of the game or its goals and game mechanics at a symbolic level.

3. STARCRAFT 2 TUG OF WAR

Our approach to these problems involves two steps. First: limit the communication on the human side. Second: establish semantic understanding of the game on the AI side. By doing so, we try to bring both the human and the AI players on a similar level of communication. The early proof of concept we would like to demonstrate is a StarCraft 2 [6] "Tug of War" scenario. These "Tug of War" games are simplified versions of the full game, where players only build up an economy and chose units to spawn in waves, without actually commanding the units in battle and therefore skipping most micro management. The key to such "Tug of War" is to use the game mechanics to your advantage, by building the right counter units or unit combinations and thus requiring a strong semantic understanding of the different game elements. These scenarios are also usually played in teams of 2-6 players where communication and cooperation become essential, to adapt your unit composition to the strategies of your team mates. These properties make "Tug of War" games a good test bed to work on cooperation without having to deal with the full spectrum of a RTS AI.

Our scenario is a common "Tug of War" game with two teams of 3 human players. In addition, every human player is grouped with an AI player with which they share research but have individual economy. The "AI Buddy" of a human player also spawns its wave at the same time as the human player, thus emphasizing strong cooperation to maximize the efficiency of the combined army.

Communication with your “AI Buddy” takes place in a dedicated panel, that offer symbolic communication for general strategies like “Focus on military force” or “Focus on research” and different roles a player can fulfill, like “Deal a lot of damage” or “Support me”. While this communication is rather abstract and simplified on the human end, it is covered by a semantic structure on the AI end, enabling it to understand the complex counter mechanics of StarCraft 2. The communication through this panel is also bidirectional, allowing the AI to express what it wants the human player to do and how happy the AI is with the current situation.

4. Implementation

The AI Implementation uses a static semantic net storing the game mechanics relations, like weapon types, unit categories or tech-tree requirements. The AI players query the semantic net in order to understand the symbolic commands of the human players. This allows the AI players to interpret the strategy “Focus on military force” in respect to the current situation, for example the army composition of allied players or the last observed hostile waves.

The semantic structures and their query language are written in a Lisp-Like domain specific language, tailored to the requirements of the StarCraft 2 engine and are compiled into the actual StarCraft 2 in-Game scripting representation for integration. This allows maintaining and visualizing the semantic net externally, while still employing performance optimizations like caching during run-time.

An example would be a team where one player is supposed to build a strong early-game force, buying time for their teammate to construct necessary requirements for stronger late-game units. One of the important challenges for both human and AI players is now answering the question which units to build. The semantic structure contains most gameplay elements and their relations, of which *“requires”* and *“counters”* are the relevant ones for this example. The following figure shows a simplified excerpt of a semantic structure covering two units with their requirements. The *“counters”* relation expresses that the *Siege Tank* is a good choice fielding against *Space Marines*.

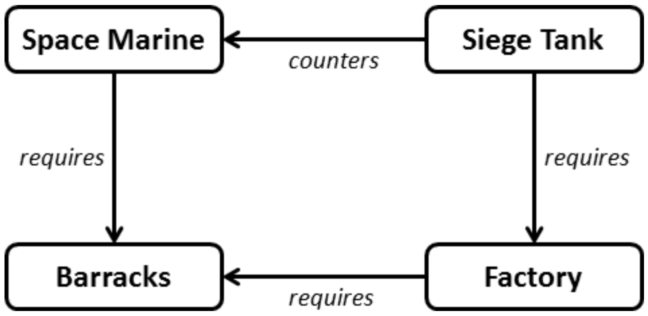

Figure 1: Simplified Semantic Structure

In reality, these relations are obviously more complex, as a *“counters”* relation contains many parameters regarding its strength, requirements like unit positioning or modifiers depending on tech-tree progress, resource income and other contexts. As unit costs different amounts of three distinct resources, these will also have to be accounted for. A *Siege Tank* is a strong counter against *Space Marines*, but it also costs a lot more.

Based on that simplified structure, however, an AI player could identify that building *Siege Tanks* is a good choice against the enemies *Space Marine* army. Walking down the *“requires”* relations, it further identifies that it would need a *Factory* it does not yet command. As building such a requirement is an investment of resources, communication with its allies is used to identify the threshold of estimated gain of producing a target unit in respect to the resource costs of building the requirements first. The early-game partner in the game would have a higher threshold, only advancing in the tech-tree if a high-tech unit is really a significantly strong counter of the enemy force, while the teching player would have a smaller threshold, investing more resources in building up requirements for stronger counters instead of building early game forces.

5. GOAL AND OUTLOOK

The goal of this prototype is to see how well bi-directional communication with an AI player can be implemented in an RTS using rather simple methods on both the human and the AI end.

The very simple symbolic communication on the human-side of communication could be enhanced significantly, for example by adding spatial information, using mouse gestures or deploying natural language processing to enable text or voice chat. The semantic net on the AI side could also be expanded. A dynamic semantic net looks promising, allowing adjusting the relations between entities (unit types) at runtime, for example to cover certain unit formations where certain counter mechanics suddenly behave differently.

This proof of concept we search for the technical minimum that needs to be implemented to achieve enjoyable cooperation with an AI agent, intentionally skipping some more advanced techniques. From there, we want to expand these concepts to a full RTS and benchmark the different AI-implementation and human-usability methods.

REFERENCES

[1] A. Drachen and J. H. Smith, "Player talk - the functions of communication in multplayer role-playing games," Computers in Entertainment - SPECIAL ISSUE: Media Arts (Part II), vol. 6, no. 4, 2008.

[2] G. Wadley, M. Gibbs and P. Benda, "Speaking in character: using voice-over-IP to communicate within MMORPGs," in IE '07 Proceedings of the 4th Australasian conference on Interactive entertainment, 2007.

[3] L. Dabbish, R. Kraut and J. Patton, "Communication and commitment in an online game team," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2012.

[4] M. Ramsey and S. Rabin, "Designing a Multi-Tiered AI Framework," in AI Game Programming Wisdom 2, Hingham, MA, Charles River Media, Inc., 2004.

[5] M. van Lent and J. Laird, "Developing an Artificial Intelligence Engine," in Proceedings of the Game Developers Conference, San Jose, CA, 1999, pp. 577-588.

[6] Blizzard Entertainment, StarCraft II: Wings of Liberty, 2010.