2006

Simulation Framework as a Multi-User Environment for a Go*Team game

J. Jagiello
Defence Science and Technology Organisation

M. Eronen
Defence Science and Technology Organisation

N. Tay
Defence Science and Technology Organisation

D. Hart
Australian National University

L. Warne
Defence Science and Technology Organisation

See next page for additional authors

Simulation Framework as a Multi-User Environment for a Go*Team game

Abstract

This paper describes the design and implementation of a computerized team version of the ancient strategy game of Go. The game is called Go*Team, and forms part of a research effort investigating how people and groups coordinate, cooperate and share information, especially in a military network-centric environment. Of particular interest in the research are human or group related factors that may impede or even prevent the successful achievement of such coordination, cooperation and information sharing despite the availability or presence of the technological capability to support it. Accordingly, Go*Team is designed to embed its players in an environment that involves conflict, cooperation and coordination, but also competition, uncertainty, complexity and timely and appropriate decision making In addition to these aspects, the game is designed to be played in a network-centric environment in which players can be required, or choose, to make use of modern communication tools such as email, voice over IP, group support systems, chat rooms and the like to effect the cooperation and coordination they need to successfully play the game. Go*Team is implemented as a multiplayer network computer game by the use of a Simulation Framework designed for setting up distributed simulations. The simulation framework provides a component architecture for implementing game rules, game entities and their sensors. The game is currently being tested in a university environment and more tests are planned within military education facilities. Development work is continuing.

Authors

J. Jagiello, M. Eronen, N. Tay, D. Hart, L. Warne, and Helen M. Hasan

Simulation Framework as a Multi-User Environment for a Go*Team game

[1]Jerzy Jagiello, [1]Marko Eronen, [1]Nicholas Tay, [2]Dennis Hart, [1]Leoni Warne, [3]Helen Hasan

[1]Defence Systems Analysis Division, Defence Science and Technology Organisation, [2]School of Business and Information Management, Australian National University, [3]Information Systems School of Economics and Information Systems Faculty of Commerce, University of Wollongong

Abstract. This paper describes the design and implementation of a computerized team version of the ancient strategy game of Go. The game is called Go*Team, and forms part of a research effort investigating how people and groups coordinate, cooperate and share information, especially in a military network-centric environment. Of particular interest in the research are human or group related factors that may impede or even prevent the successful achievement of such coordination, cooperation and information sharing despite the availability or presence of the technological capability to support it. Accordingly, Go*Team is designed to embed its players in an environment that involves conflict, cooperation and coordination, but also competition, uncertainty, complexity and timely and appropriate decision making In addition to these aspects, the game is designed to be played in a network-centric environment in which players can be required, or choose, to make use of modern communication tools such as email, voice over IP, group support systems, chat rooms and the like to effect the cooperation and coordination they need to successfully play the game. Go*Team is implemented as a multiplayer network computer game by the use of a Simulation Framework designed for setting up distributed simulations. The simulation framework provides a component architecture for implementing game rules, game entities and their sensors. The game is currently being tested in a university environment and more tests are planned within military education facilities. Development work is continuing.

Introduction

The focus of Network Centric Warfare (NCW) research and development has largely been concentrated on issues related to technology and infrastructure. However, there is a growing requirement for analysis of the human aspects of network warfare. The human factors community has concerns about the impact of technology on human performance, and has identified a need for investigation of individual and group behaviours in an NCW context [1]. The NCW context is characterized by conflict, cooperation and competition, information sharing, and timely decision making.

The game of Go itself has nothing to do with NCW per se, but it creates an opportunity for cooperation and coordination between individuals in an environment which exhibits many of the characteristics of the NCW context.

Standard Go

The game of Go is believed to have been originally developed in China between three and four millennia ago. [2,3,4] A Go game consists of black and white stones and a square board with grid lines. The board sizes can vary, though a standard board has 19x19 grid lines. The standard game comprises two players who take turns placing stones onto the grid line intersections of the board. The goal of the game is to occupy

the most territory on the board by players placing stones onto the board staking claim to areas which they intend to occupy. As the game progresses, players have to defend their positions while attempting to gain more territory. Stones cannot be moved once they are placed onto the board. They can, however, be captured, which results in their removal from the board. Stones are captured when they are surrounded by an opponent's stones. While capturing stones is not the object of the game it does provide a useful way of gaining additional territory. The winner of the game is the player who controls a larger proportion of the board when the game ends.

The Go*Team game provides an opportunity to explore how humans function in a simulated NCW environment, the techniques they prefer to use, the techniques that are more successful, and the barriers.

Go*Team: The Team Version of Go

Go*Team is a computerized environment that allows teams of players to play a modified form of Go [5]. Reflecting the aspects of network-centric warfare, this team version of the game is designed to embed its players in an environment that involves:

- conflict (with the other team or teams involved in the game);

- cooperation and coordination, but also competition (with and between the players in one's own team);

- information sharing (through the need to continually share information in order to synthesize and integrate, in a dynamic situation, multiple fragmentary and local perspectives into an overall situational picture);

- timely and appropriate decision making (through the need to balance the time taken for adequate situational analysis and the pressure to avoid being overtaken by events).

The game does not include any built-in communication facilities. It is instead designed to be used as part of a network-centric environment in which the players can choose to make use of modern communication tools such as email, group support systems, chat rooms or the like (although they may, at the experimenter's discretion, also have available the more traditional means such as telephone and face-to-face meetings) to effect the cooperation and coordination they need to successfully play the game.

Variations of Go and Go*Team

Go has been the source of a number of variations, including some involving teams (Rengo) or multiple individual players (e.g. Four colour Go), limited information (Phantom Go), cooperation, parallelism and simultaneous moves. Nevertheless, while some of these variants may encapsulate one or perhaps two of the aspects of NCW environments in which we are interested, none exhibit all that we wish to explore. For these reasons, we have created our own version of Go, called Go*Team. The main features of Go*Team are described below.

Most importantly, in Go*Team the opposing sides now consist of teams of players rather than individuals. This introduces into the game the need to coordinate and

cooperate within groups of individuals. Moreover, each individual player in a team has their own collection of stones, over which they have complete control regarding whether, when and where they are to be placed on the board. This gives each individual in a team autonomy in terms of what happens to their stones. It is possible, for example, that the team, or at least the other members of it, may decide they want a particular player to put a stone in a certain position, but the ultimate decision whether and how much to cooperate with the other team members in the use of "their" resource is up to the player who owns the stone.

Another important part of Go*Team is that individual players in a team have only a local view of the overall Go*Team "world" in which they are embedded. This view consists of a board showing the positions of their own stones plus any stones of the opposing team that are closer to their own stones than those of any other player on their team. This modification is to introduce the problem of information sharing and integration into the game. Since each player has only a local and partial picture of what is going on, it is necessary that they share what they can see with the other members in order to develop an integrated overall picture of the state of the board – and even if they can accurately achieve this in the time available, then they have to decide not only what is the best next move, but also who should make it.

Each player has the ability to place various types of "markers" on their local view of the Go*Team board. They can use these markers to record where they know, or think they know, stones belonging to the other members of their own team as well as those of the opposition are. However, at the same time as this gives the individual players a tool for helping them build an integrated picture of the overall state of their Go*Team "world", it also introduces the possibility that errors will be made, and therefore for there to be disinformation, incomplete and conflicting information circulating between the players in a team. This will only add to the uncertainty they face as well as the complexity of the decision-making processes in which they are engaged when deciding their next move(s).

Unlike standard Go, in which the players take turns to place their stones, teams playing Go*Team no longer have to take turns; a team's next turn can be taken after a "relaxation time" (specified via the server, and probably something of the order of a few minutes) regardless of whether or not the opposing team has done anything in the interim. This is to introduce the problems of tempo, uncertainty and potential decision paralysis into the game. While a team may take as long as they like over and above the specified relaxation time to analyze their situation before making a move, they thereby increase the risk that the other team will gain an advantage by making further moves while they are still working out what to do next – the decision paralysis syndrome. Conversely, if they take insufficient time to do their analysis then they risk making inappropriate, worthless or even disadvantageous moves as a result. Moreover, this ability to make moves independently of whether or not the other team has done so removes, or at least significantly alters, the relevance of the "classic" situations and their related moves that players of standard Go learn and use. This introduces the requirement for flexibility, imagination and innovation into working out how best to deal with the situations with which the team finds itself confronted on the board.

The overall winner of the Go*Team game is the individual player whose team wins,

and who has a greater proportion of his or her own collection of stones remaining on the board than any other player in their team. This modification introduces an element of competition as well as cooperation between the players in a team in that, while a player cannot win unless their team also wins (so they have to share information, coordinate and cooperate with the other team members in order to achieve this at least), they also have the motivation to work to their own advantage, and against their fellow team members, as far as is possible within the bounds of that overall cooperation necessary to ensure that their own team wins. And, in order to do this, they, as individuals, will need to try to keep tabs on how the other team members are doing relative to themselves so they, as individual players, face not only the potential problem of information processing overload in trying to work this out but also how to influence team decisions so they work as much in their individual favour as possible.

While not dictated by the nature of the game itself, the intention is that Go*Team players will operate in a distributed way, using largely electronic means to effect their information sharing, integration, decision making and coordination needs. These electronic means operate outside of, though in parallel with, the running of the Go*Team game. That is, it is envisaged that team members (who will normally be physically distributed) will have access to standard electronic communication technologies such as an email system (e.g. Microsoft Outlook), chat room software (e.g. WebChat), video-conferencing, or group support systems like NetMeeting, Lotus Notes or similar.

Finally, the Go*Team design also includes the capability to have more than two teams of players, who may form and dissolve alliances, and also for teams to play on multiple boards concurrently. This provides the researchers with the potential to add to not only the complexity of the situations with which the players are faced, but also to introduce issues of priority and command organization into the game. These aspects are, however, still in the early stages of investigation and development.
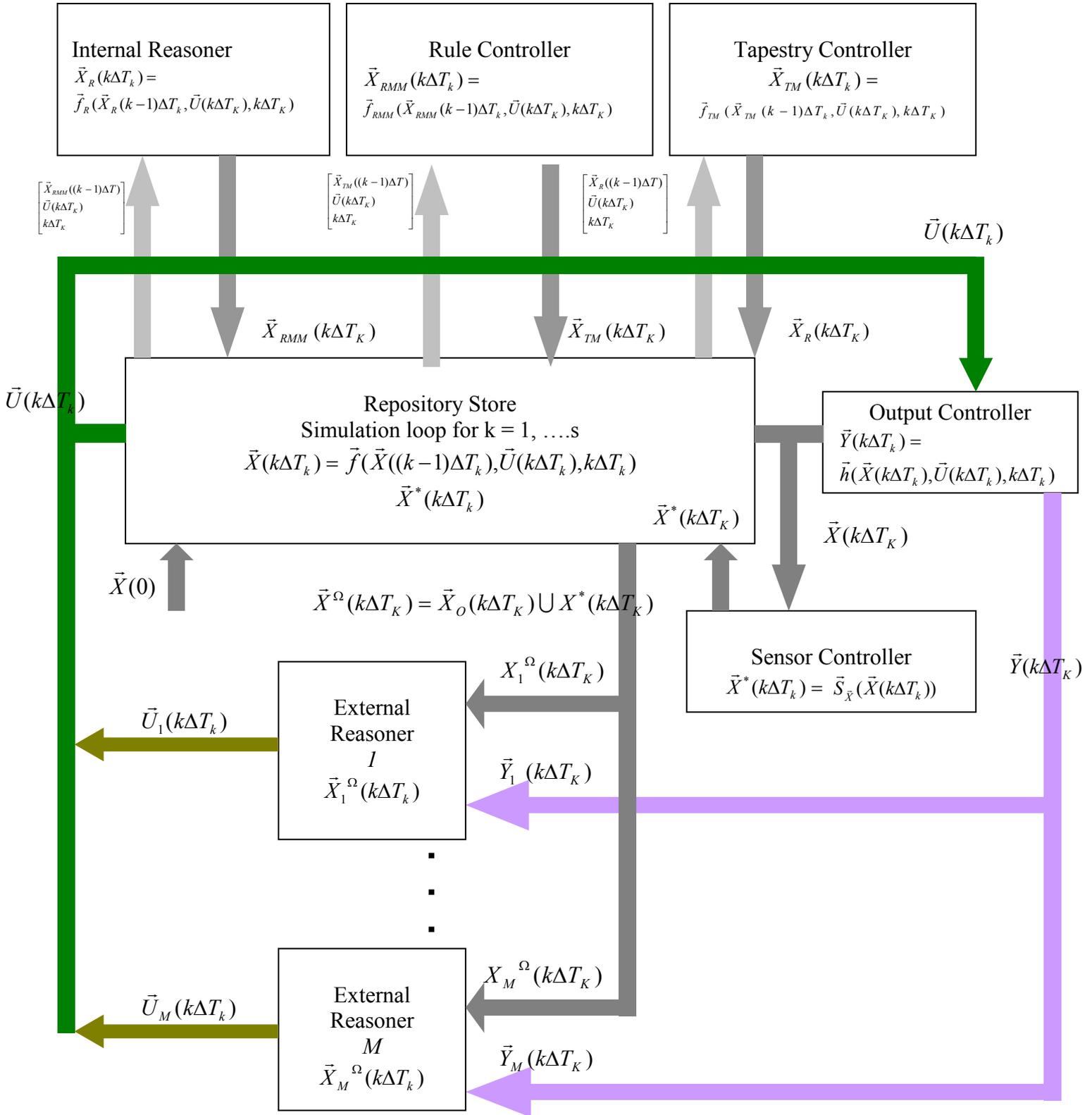
Go*Team Design

Go*Team is implemented as a multiplayer network computer game. This task was made much simpler by use of a Simulation Framework designed for setting up distributed simulations of real-time robotic applications. The simulation framework provides a component architecture for implementing game rules, game entities and their sensors. It also provides time step scheduling and networking.

Distributed simulators are tools that are often used to examine the conditions of complex systems. For an effective simulation, models must be found that accurately represent the states and transitions of the complex system. Simulation models of complex systems can be implemented as custom applications dedicated to a particular domain with dedicated GUIs and support tools. Alternatively, the same effect can be achieved by building a generic simulation framework and then implementing the simulation model as a specific instance of that framework. The purpose of a simulation framework is to provide basic infrastructure for development of simulation experiments [6].

Simulation Framework Discrete Model

The following diagram represents a conceptual model of the simulation framework.

Internal Reasoner

$$\vec{X}_R(k\Delta T_k) = \vec{f}_R(\vec{X}_R(k-1)\Delta T_k, \vec{U}(k\Delta T_K), k\Delta T_K)$$

Rule Controller

$$\vec{X}_{RMM}(k\Delta T_k) = \vec{f}_{RMM}(\vec{X}_{RMM}(k-1)\Delta T_k, \vec{U}(k\Delta T_K), k\Delta T_K)$$

Tapestry Controller

$$\vec{X}_{TM}(k\Delta T_k) = \vec{f}_{TM}(\vec{X}_{TM}(k-1)\Delta T_k, \vec{U}(k\Delta T_K), k\Delta T_K)$$

$$\begin{bmatrix} \vec{X}_{RMM}((k-1)\Delta T) \\ \vec{U}(k\Delta T_K) \\ k\Delta T_K \end{bmatrix}$$

$$\begin{bmatrix} \vec{X}_{TM}((k-1)\Delta T) \\ \vec{U}(k\Delta T_K) \\ k\Delta T_K \end{bmatrix}$$

$$\begin{bmatrix} \vec{X}_R((k-1)\Delta T) \\ \vec{U}(k\Delta T_K) \\ k\Delta T_K \end{bmatrix}$$

$$\vec{U}(k\Delta T_k)$$

$$\vec{X}_{RMM}(k\Delta T_K) \qquad \vec{X}_{TM}(k\Delta T_K) \qquad \vec{X}_R(k\Delta T_K)$$

$$\vec{U}(k\Delta T_k)$$

Repository Store
Simulation loop for k = 1, ….s

$$\vec{X}(k\Delta T_k) = \vec{f}(\vec{X}((k-1)\Delta T_k), \vec{U}(k\Delta T_k), k\Delta T_k)$$

$$\vec{X}^*(k\Delta T_k)$$

$$\vec{X}^*(k\Delta T_K)$$

Output Controller

$$\vec{Y}(k\Delta T_k) = \vec{h}(\vec{X}(k\Delta T_k), \vec{U}(k\Delta T_k), k\Delta T_k)$$

$$\vec{X}(k\Delta T_K)$$

$$\vec{X}(0)$$

$$\vec{X}^{\Omega}(k\Delta T_K) = \vec{X}_O(k\Delta T_K) \bigcup X^*(k\Delta T_K)$$

$$X_1^{\Omega}(k\Delta T_K)$$

Sensor Controller

$$\vec{X}^*(k\Delta T_k) = \vec{S}_{\vec{X}}(\vec{X}(k\Delta T_k))$$

$$\vec{Y}(k\Delta T_K)$$

External Reasoner 1

$$\vec{X}_1^{\Omega}(k\Delta T_k)$$

$$\vec{U}_1(k\Delta T_k)$$

$$\vec{Y}_1(k\Delta T_K)$$

$$X_M^{\Omega}(k\Delta T_K)$$

External Reasoner M

$$\vec{X}_M^{\Omega}(k\Delta T_k)$$

$$\vec{U}_M(k\Delta T_k)$$

$$\vec{Y}_M(k\Delta T_K)$$

where:

> *rule controller* – manipulates the $\vec{X}_{RMM}(k\Delta T_k)$ states
>
> *tapestry controller* - manipulates the $\vec{X}_{TM}(k\Delta T_k)$ states
>
> *internal reasoner* - manipulates the $\vec{X}_R(k\Delta T_k)$ states
>
> *sensor controller* – converts the "true" state $\vec{X}(k\Delta T_k)$ into the perceived state $\vec{X}^*(k\Delta T_k)$ of the simulator
>
> *external reasoner 1 .....i* – stores its perceive state and generates the input vector $\vec{U}_i(k\Delta T_k)$ in order to propose the change in the state of the simulator. These inputs can be rejected or accepted by the simulator.

The simulation framework is modelled as a non-liner discrete system where the state of the simulation is represented as follows[7,8,9]:

$$\vec{X}(k\Delta T_k) = \vec{f}(\vec{X}((k-1)\Delta T_k),\vec{U}(k\Delta T_k),k\Delta T_k)$$

where:

> $\vec{X}$ - state vector of the simulator
>
> $\vec{X}(0)$ - initial state of the state vector for k = 0
>
> k = 1, …. s – represents number of simulation steps
>
> ΔT_k - simulation time step
>
> $\vec{f} = \vec{f}_{TM} \circ \vec{f}_{RMM} \circ \vec{f}_R((k-1)\Delta T_K,\vec{U}(k\Delta T_K - \tau),k\Delta T_K)$ - non-linear multi dimensional state transition function where:
>
> $\vec{f}_{RMM}((k-1)\Delta T_K,\vec{U}(k\Delta T_K - \tau),k\Delta T_K)$ - rule manager state transition function
>
> $\vec{f}_{TM}((k-1)\Delta T_K,\vec{U}(k\Delta T_K - \tau),k\Delta T_K)$ - tapestry state transition function
>
> $\vec{f}_R((k-1)\Delta T_K,\vec{U}(k\Delta T_K - \tau),k\Delta T_K)$ - internal reasoner state transition function
>
> $\vec{U}(k\Delta T_k) = : :_{i=1}^{M} U_i(k\Delta T_k)$ - concatenation of M reasoner vectors
>
> $\vec{Y}(k\Delta T_k) = \vec{h}(\vec{X}(k\Delta T_k),\vec{U}(k\Delta T_k),k\Delta T_k)$ - output vector where:
>
> $\vec{h}$ - non-linear multi dimensional output function
>
> $\vec{X}^{\Omega}(k\Delta T_K) = \vec{X}_O(k\Delta T_K) \cup X^*(k\Delta T_K)$ - state of the system available for all external reasoners where:
>
> $\vec{X}_O(k\Delta T_K) \subseteq \vec{X}(k\Delta T_k)$ state available for all external reasoners
>
> $\vec{X}^*(k\Delta T_K) = \vec{S}_{\vec{X}}(\vec{X}(k\Delta T_k))$ state seen by all sensors
>
> $X_1^{\Omega}(k\Delta T_K) \ldots X_M^{\Omega}(k\Delta T_K) \subseteq \vec{X}^*(k\Delta T_k)$ state of the system available for each individual external reasoners

The state vector $\vec{X}(k\Delta T_k)$ is controlled by the rule, internal reasoner, tapestry and external reasoners as defined below.

$$\vec{X}_{RMM}(k\Delta T_k) \cup \vec{X}_{TM}(k\Delta T_k) \cup \vec{X}_R(k\Delta T_k) \subseteq \vec{X}(k\Delta T_k)$$

where:

$\vec{X}_{RMM}(k\Delta T_k),\vec{X}_{TM}(k\Delta T_k),\vec{X}_R(k\Delta T_k) \subseteq \vec{X}(k\Delta T_k)$ - subsets of the state vector

$\vec{R}(k\Delta T_k) = \vec{X}(k\Delta T_k) / (\vec{X}_{RMM}(k\Delta T_k) \cup \vec{X}_{TM}(k\Delta T_k) \cup \vec{X}_R(k\Delta T_k))$ - residual of the state vector

The following equation is true at any stage of the simulation process

$$\forall_{i=1,n} x_i \in \vec{X}_{RMM}(k\Delta T_k) \vee \vec{X}_{TM}(k\Delta T_k) \vee \vec{X}_R(k\Delta T_k) \Rightarrow x_i(k\Delta T_k) \in \vec{X}(k\Delta T_k)$$

Only in the case when $\vec{R}(k\Delta T_k) = \varnothing$ then

$$\forall_{i=1,n} x_i \in \vec{X}_{RMM}(k\Delta T_k) \vee \vec{X}_{TM}(k\Delta T_k) \vee \vec{X}_R(k\Delta T_k) \Leftrightarrow x_i(k\Delta T_k) \in \vec{X}(k\Delta T_k)$$

The order of applying state transition functions is very important. It has a profound impact on the calculated state and has to be controlled dynamically based on circumstances using sophisticated logic expressions. Logic can be applied to the group of functions as well between the groups. By default the general rule will apply.

$$\vec{f}_R \succ \vec{f}_{RMM} \succ \vec{f}_{TM}$$

The true state $\vec{X}(k\Delta T_k)$ of the system is filtered out by the sensor controller and the perceived state $\vec{X}^{\Omega}(k\Delta T_K) = \vec{X}_O(k\Delta T_K) \cup X^*(k\Delta T_K)$ is stored and maintained by the external reasoners.

$\vec{X}_O(k\Delta T_K) \subseteq X(k\Delta T_K)$ - observable state of the system

$\vec{X}^*(k\Delta T_k) = \vec{S}_{\vec{X}}(\vec{X}(k\Delta T_k))$ - perceived subset of the system state

$$\vec{S}_{\vec{X}}^{-1}(\vec{X}^*(k\Delta T_k)) \subseteq \vec{X}(k\Delta T_k)$$

$$\forall_{i=1,M} \vec{X}_i^{\Omega}(k\Delta T_K) \subseteq \vec{X}^{\Omega}(k\Delta T_K)$$

$\vec{X}_i^{\Omega}(k\Delta T_K)$ - state of the system available for i external controller

The calculation time step ΔT_c is the amount of time that takes to computer every simulation step as follows:

$$\Delta T_C = \Delta T_A + \tau$$

ΔT_A - allocated time for submission of controls represented by the $\vec{U}$ vector

τ - variable time that defines the necessary time to calculate and update the state of the simulator which is a function of dynamically calculated state transition functions $\tau(\vec{f}_{TM}, \vec{f}_{RMM}, \vec{f}_R)$. The simulation time step has no impact on τ under condition that the numerical accuracy of simulation is excluded from our considerations.

$$\tau = \max\left(\Delta T_{S_{Min}}, \Delta T_{SR}\right)$$

$\Delta T_{S_{Min}}$ - min allocated time to complete the state change of the simulator

ΔT_{SR} - actual time to complete the state change of the simulator

Go*Team Discrete Model

The Go*Team discrete model is an instance of the simulation framework model described above. The state, inputs, and outputs represent the Go*Team game specification.
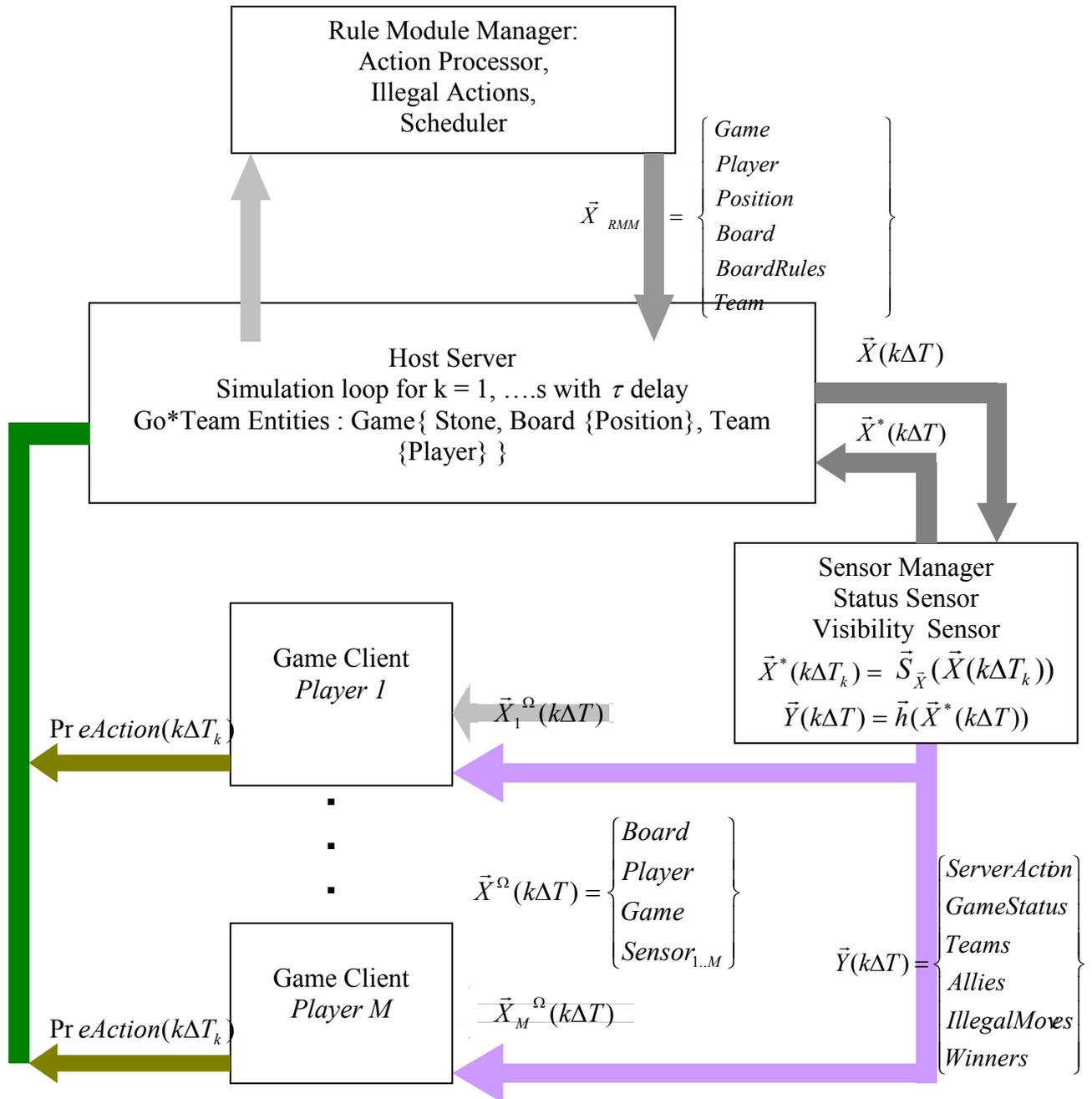

*Figure 1. Go*Team discrete model*

Conceptually the Game Host accepts proposed actions from clients and processes them using the game rules to determine a new game state. A perception of the game state is reported back to the clients.
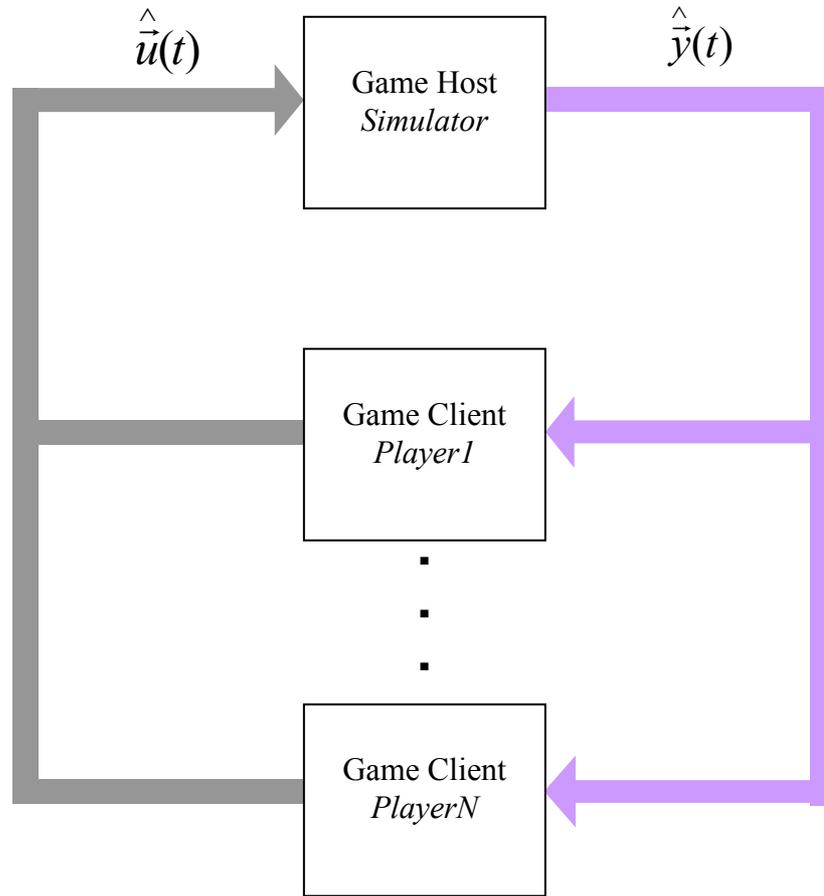
$$\hat{\vec{u}}(t) \qquad\qquad\qquad \hat{\vec{y}}(t)$$

Game Host
Simulator

Game Client
Player1

Game Client
PlayerN

Figure 2. Game Host and Game Client interaction

$$y(k\Delta T) = \begin{bmatrix} \phi \vee \mathrm{y}_i(k\Delta T), i = 1..6 \\ where \\ y_i \in \begin{cases} ServerAction(k\Delta T), GameStatus(k\Delta T), Teams(k\Delta T) \\ Allies(k\Delta T), IllegalMoves(k\Delta T), Winners(k\Delta T) \end{cases} \end{bmatrix}$$

$$\vec{u}(k\Delta T) = \left[\phi \vee \Pr eAction(k\Delta T) \right] \text{ - input to simulation}$$

The Game is defined by initial set of parameters such as the number of teams, number of players for each team, number of boards and board sizes.
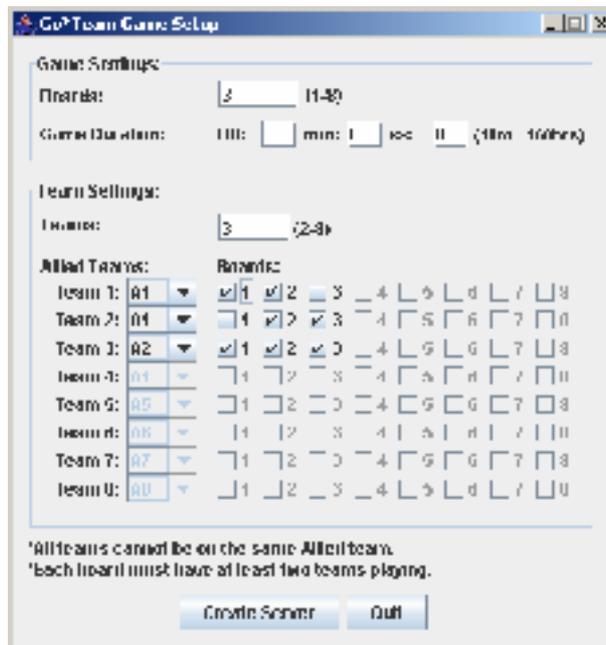
Figure 1. Initial Game configuration

The pace of the game is controlled by the scheduling scheme. Teams can take a turn whenever they want (independent), or they can be forced to take turns (turn based) or to wait certain amounts of time before being allowed to make their next move (pacing, forced delay).
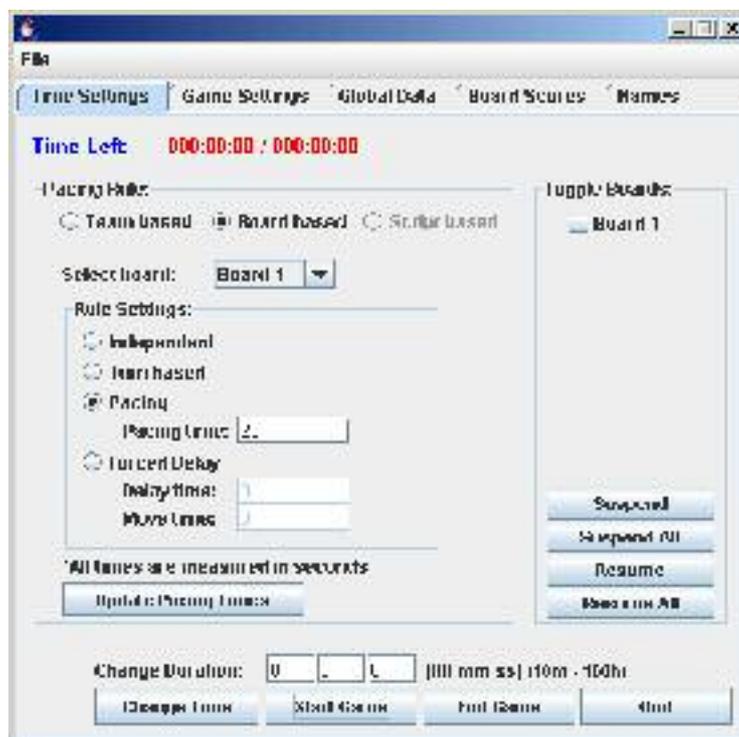

Figure 2. Time Settings pane

Teams can form alliances to work together to capture stones. Stones captured by an alliance are distributed according to prisoner ownership rules.
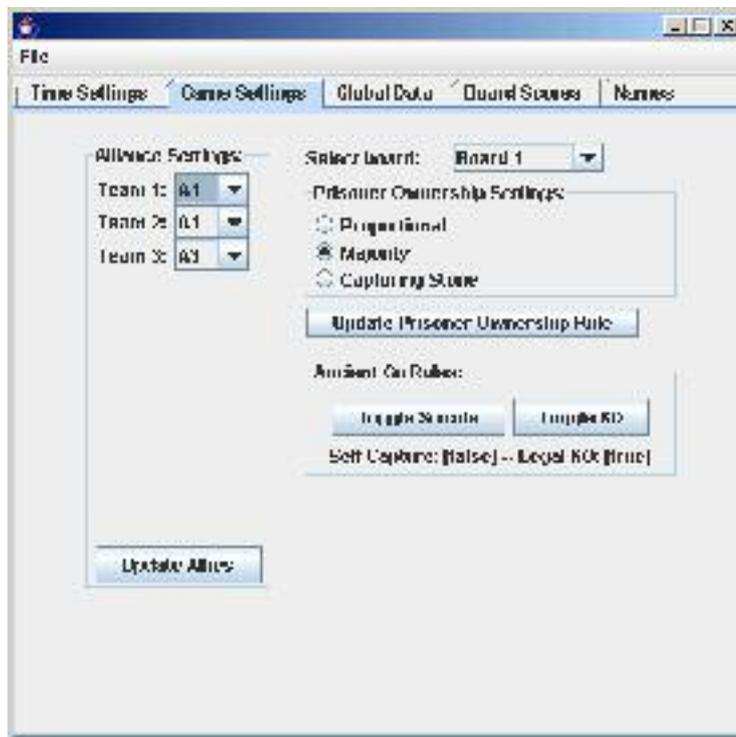
Figure 3. Game Settings pane

Players can join a game and declare which team they are playing on.
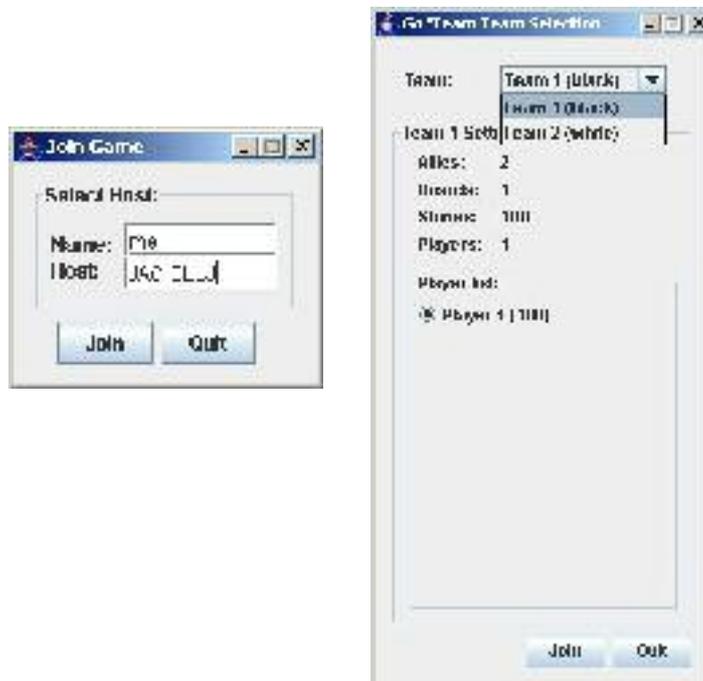

Figure 4. Player me joining the game hosted by the jagiellj server

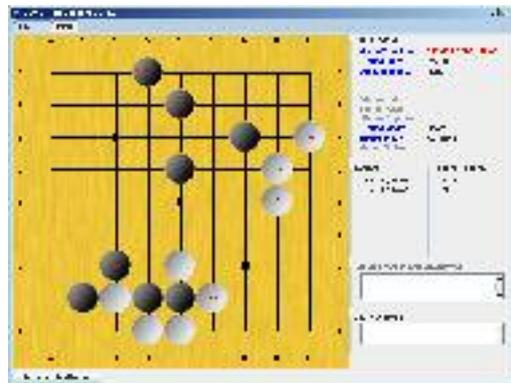The host has a complete view of all the stones placed.

Figure 5. Host game board GUI

Each player has their own view of the board in which they only see their own stones and the stones of opposition players that are closest to them.
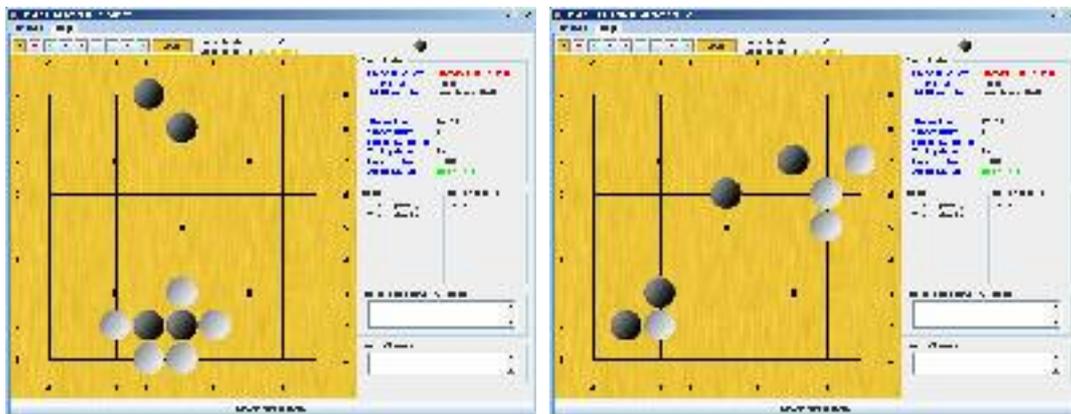

Figure 6. Client game board GUIs for different players on the same team

As described above, players may use markers to improve their situational awareness in the game.
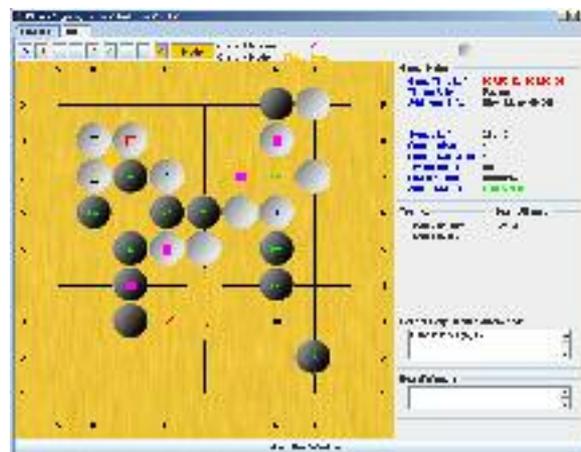

Figure 7. Client game board GUI with markers

In order to improve the player's perception of the game, a new interface is being developed to simulate immersion in a 3D environment.
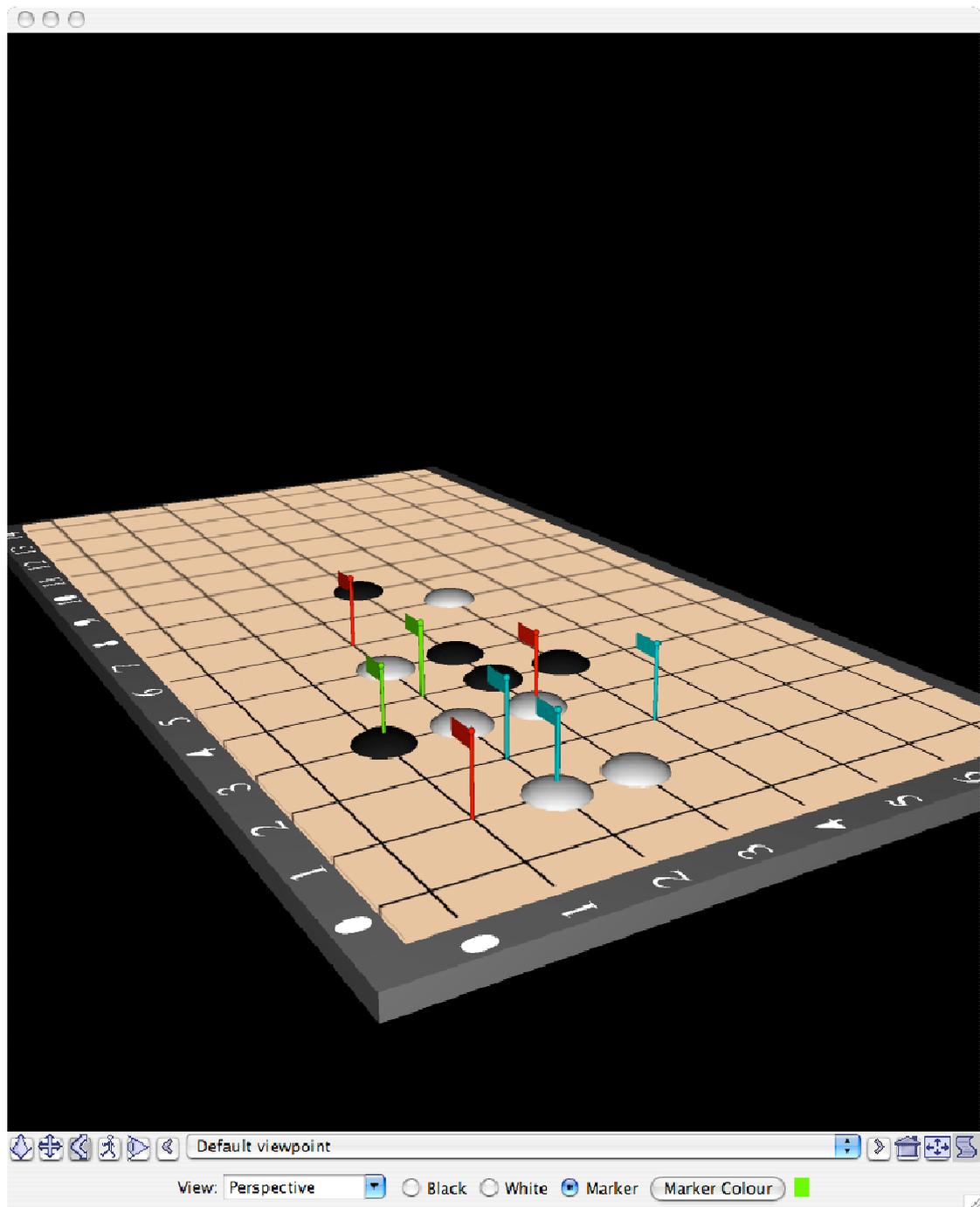

Figure 8. 3D Client game board GUI

Go*Team Sessions Design

To date several sessions have taken place in a Usability Laboratory set up in a converted 6 room cottage. The configuration for the Go*Team session uses the computer set up in one central room for usability testing as the server. This provides several options for recording, principally screen and audio capture by the Camtasia program. Six other computers are set up as clients in other rooms, with two or three computers in one room where the White Team members cannot see each other's screens but can communicate verbally. Members of the Black Team are in a different room and can only communicate via CHAT. The screen on the server shows both the Server view of the Go*Team board in play and the Black team Chat. It also captures the audio of the White Team's communication via microphone.

Before and after each session players are asked questions pertaining to the variable listed above. After sessions all players are also allowed to talk about their experience and this discussion is recorded. Where the same players participate in more than one session their learning is also observed as to their performance as well as their ability to cooperate and communicate.

To date, analysis has been based on qualitative techniques, Q-method to conduct factor analysis on statements collected from participants, and content analysis on participant comments using Leximancer. Future versions of Go*Team will include built-in metrics to assess the issues under investigation.

Conclusion

The Go*Team game has been designed as a research vehicle for investigating collaboration and cooperation between team members in a competitive and dynamic environment. The Go*Team computerized game environment exhibits many of the features of an NCW environment with its inherent uncertainties, ambiguities and complexities, information sharing, integration and overload issues, tempo, communication technologies, and requirement for cooperation and coordination as well as the inevitable competition that seems to occur between different individuals and groups in such situations. However, these features are often mirrored in many other types of organizations. Consequently, there is also a growing interest in the defence community in the possible use of the Go*Team game for educational purposes.

One further goal of the development is to create an environment in which people can play against artificial intelligence agents. This requirement is dictated by economical and organisational circumstances, whereby it is typically very difficult to arrange many teams of people to play against each other. Teams of players could play against agents, or agents could be included in teams of people. This idea is depicted in the game logo.

Anticipated short term developments include addition of tools for analysis and recording/replaying of games. These will be used to derive strategies on how to teach collaboration, cooperation and information sharing in non-hierarchical team based environments.

References

1. Warne, L., Ali, I., Bopping, D., Hart, D. and Pascoe, C. (2004) "The Future Warrior", *Proceedings of the 9th International Command and Control Research and Technology Symposium (ICCRTS)*, September 14-16.

2. Burmeister, J. M. (2000) "Studies in Human and Computer Go: Assessing the Game of Go as a Research Domain for Cognitive Science". PhD thesis, University of Queensland

3. AGA – American Go Association (2005) "Official AGA Rules of Go", http://www.usgo.org/resources/downloads/completerules.pdf accessed 17 Feb, 2006.

4. Müller, M. "Review: Computer Go 1984-2000", University of Alberta 2000

5. Hart D., Eronen M., Jagiello J., Warne L. (2006) "Go*Team: A Team Version of Go for Investigating Cooperation, Coordination and Information Sharing in Network-Centric Warfare Environments", SimTect Conference 2006 submission

6. Gottlieb, E., Harrigan, R., McDonald, M., Oppel, F., Xavier, P. "The Umbra Simulation Framework", Sandia Report SAND2001-1533 June 2001

7. Luenberger, D. G. "Introduction to Dynamic Systems, Theory, Models & Application" by John Wiley & Sons, Inc 1979

8. Franklin, G. F., Powell, J. D., Workman, M., "Digital Control of Dynamic Systems" Third Edition by Addison Wesley Longman, Inc 1998

9. Bender, C. M., Orszag, S. A. "Advanced Mathematical Methods for Scientists and Engineers" Springer-Verlag New York, Inc 1999