Distributed Othello

Pichaya Tandayya, Watchara Tongruang and Alwaris Daragai
Department of Computer Engineering, Faculty of Engineering,
Prince of Songkla University
Hat Yai, Songkhla 90112, Thailand
pichaya@coe.psu.ac.th

Abstract

Othello is an interesting board game for thought. Traditionally, there are 2 players. Each player takes turn to put down their discs on the board. The basic idea is to capture the opponent’s discs and turn them into its colour. The game ends when the board is full or no disc can be put down further. The one that has more discs at the end will be the winner.

Such kind of board games usually involves only two players. In the Internet, there are more board games offered but none of them allow more people involved. It will be more interesting to allow more players to form teams to take turn and make the game distributed.

High Level Architecture (HLA) is an IEEE architecture for distributed interactive simulation that allows each simulation node or a federate to interoperate with one another in a more complex simulation environment called a federation. The Run-Time Infrastructure (RTI) is the infrastructure for connecting and allowing transaction data in the virtual environment using the method of publication and subscription amongst the federates.

This work presents the development of this game by using the HLA to enhance the game to be distributed. The outcome of this work will probably allow new options for board games and more participants.

1 Introduction

Othello is a board game [1]. Traditionally, there are two sides. Each side is represented by discs of a different colour, normally black and white. The board’s dimension is 8x8. At the starting point, there are two discs from both sides located in the middle of the board as shown in the Fig. 1. Usually, the black side starts first and then the white side takes turn. At each turn, each side captures the opponent’s disc and turns them into its colour (Fig. 2.). It continues like this until the game is over when the board is full of discs or no disc can be put down further as shown in Fig. 1. The side that has more discs is the winner.

This game and similar board games involves only two players. Usually, board games offered on the Internet still allow only two players involved. It will be more interesting to make use of the Internet to allow more players to form teams to take turn and make the games distributed. In order to make this kind of board games distributed, the games should allow more players on each side and taking turn amongst players in the team, and have interactions amongst players and team management.

High Level Architecture (HLA) is HLA is a standard approach for building networked computer-based simulation systems developed by the Defense Modeling and Simulation Office (DMSO) [2-5]. The High Level Architecture (HLA) standard facilitates the implementation of distributed interactive simulation which is suitable for developing networked games due to its reusability and interoperability. The HLA
simulation nodes or technically called federates can synchronize with one another via the time management service, and also transfer their object ownerships amongst themselves via the ownership management service [2]. A distributed interactive simulation system using the HLA consists of an HLA Run-time Infrastructure and federates, and is called a federation. Usually, HLA is used in distributed interactive simulation, game applications, and real-time decision support systems.

**Fig. 1.** The starting point and the end of the game.

**Fig. 2.** The black side first moves to capture the white side’s disc and turns to black.

The HLA standards concern the followings.
- Run-time Infrastructure (RTI) that follows basic HLA simulation rules for defining a federate, a federation, and their interaction [6-7]. The HLA RTI shares a similar concept to the CORBA Object Request Broker. As RTI acts as a simulation backbone, federates *must* interact to each other via the RTI.
- Object Model Template (OMT), a data format specification or a consistent method for describing entities used in the simulation system [8].
- An interface specification for interfacing the RTI and the federates [9].

The cooperation between federates and the RTI in a federation is shown in Fig. 3. The local RTI component in a federate consists of an RTI library (libRTI) [6-7]. The RTI node consists of RTIExec and FedExec [6-7]. The RTI library links the federates or simulation nodes with one another through the RTI (RTIExec). FedExec collects federates’ information for forming a federation.
The HLA standard enables each federate to be independent by allowing federates to join and leave the federation without interfering with the running of the simulation. Each federate can issue simulation events independently. By publish-and-subscribe mechanism (Fig. 4.), federates can update and receive only relevant and interested data through the RTI. This mechanism reduces unnecessary information sent around the network. Updates can be either reliable or best-effort.

Fig. 3. The concept of the HLA federation including RTI and federates.

Fig. 4. The Publish-and-Subscribe mechanism.

2 The Design Approach of the Distributed Game

The design approach of a distributed game using HLA is quite different and unique due to the HLA services supported by the RTI. In this paper, we discuss the design that concerns the following the HLA RTI services: Data Declaration Management, Data Distribution Management, Ownership Management and Time Management.

HLA is object-based. Objects in HLA federations include attributes and method calls. Attributes correspond to instance variables. The attributes are updated on an object. To receive updates of an attribute, a federate must declare its interest. Federates subscribe to classes of objects. They are notified when another federate sends updates of objects of that class. Objects in an HLA federation are created and destroyed dynamically over the life of the simulation. The RTI informs a federate when relevant instances are created by other federates by the object registration and discovery services.

An HLA object could have all attributes owned by a single federate, or have attributes owned by different federates. The ownership of an attribute can be transferred...
over the course of the simulation. However, each of those attributes has to be owned by only one federate at a time.

Apart from the object attribute publication, the HLA also provides publication of interactions for handling events with no states. An interaction is an event in time that has no continuing state. Federates subscribe to classes of interactions and are notified when another federate sends an interaction of that class. An interaction can also have parameters.

2.1 Federation Design and its Objects

The distributed board game in this paper is designed to have two basic federates including Manager and Participant. The objects sent around the federation are mainly objects of class Board. Manager create boards, matches players and controls databases for teams and players. A participant can first be passive as Observer that receives only update of board information and then change a role to Player to be actively playing in games. The class Board has three major methods including Create, Destroy and Update. Only Manager can create and destroy a board. The Board objects are first created by Manager and the ownership of the board will be transferred to the first player and then between the players of the two teams through method Update until the game ends. The board owner at each time is responsible for the update of the game which means the update of the discs on the board. The ownership will return to Manager that will destroy the board when the game is over. The life of the object of class Board is shown in Fig. 5.

![Fig. 5. Life of an object Board.](image)

<table>
<thead>
<tr>
<th>Federate/ Object</th>
<th>Board</th>
<th>Team</th>
<th>Intention</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manager</td>
<td>Publish/Sub</td>
<td>Publish</td>
<td>Subscribe</td>
</tr>
<tr>
<td>Participant</td>
<td>Publish/Sub</td>
<td>Subscribe</td>
<td>Publish</td>
</tr>
</tbody>
</table>

Table 1. Federates and their object publication

The other objects sent around in the federation are objects of class Team and Intention. The Team objects are created by and belong to Manager only. The Team object contains a list of Participants joining in the team and helps in controlling switching sides. Manager updates the list after a participant shows an intention to join in. A Participant send an Intention object to inform Manager for interest for joining in a team. Table 1. shows the list of federates and their related objects that they involve in object publication and subscription. Fig. 6. shows the relationship of federates and objects in the federation for the Distributed Othello game.

2.2 Time Management
The aim of the HLA is to reuse simulations, to get multiple simulations to work together, and to work with multiple time-management schemes. The HLA RTI provides clock time utilities for the distributed interactive simulation. HLA has a clock management scheme that supports associated times of events for a wide range of problems including real-time simulation, conservative simulation, discrete event simulation, and a simulation with no time management. Federates have boolean flags \textit{time-regulating} and \textit{time-constrained}. Federates with different time managements can interoperate with one another using the time Boolean flags as shown in Fig. 7. Likewise, participants in the distributed board game can play different roles in the game. For example, a viewer participant who does not play a game stays synchronized and gets updates but generate no events. The players are synchronized and generate game events.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{Fig.6.png}
\caption{Relationship of federates and objects in Distributed Othello game.}
\end{figure}

\section{Conclusion}
This paper presents the approach for the development of a distributed board game by using the HLA to enhance the game. Using the HLA RTI services, the design aspect is much different to that of sequential applications. Data Declaration Management concerns declaring classes and object attribute publication and reflection. Data Distribution Management helps in filtering and distributing relevant or interested data in the system. Ownership Management enables the ownership transfer of object attributes.
Time Management allows integrating different time-scheme simulation applications into the system. The paper also proposes new options for board games and will therefore lead to more participants of such games on the Internet.

**Fig. 7.** Federates with different time managements can interoperate in the same HLA federation using time-constrained and time-regulating Boolean flags.

**Acknowledgments**
This work is partly supported by the Thai National Grid Project.

**References**