System Demonstration for Generic Game Development Framework

Hao-Yun Liu, I-Chen Wu, Hao-Hua Kang, Ting-Fu Liao

Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan

{hyliou,icwu,kangbb,tfliao}@java.csie.nctu.edu.tw

Abstract

In this demonstration, we show a software framework for generic game development, including game record editing and job-level (JL) computing. For the former, the framework supports the display and editing of game positions and the browsing of the game position tree. Currently, we have developed game record editors for Connect6, Go, Chinese Chess, Mahjong, etc. In this demonstration, we show how easily an editor for the game Tic-Tac-Toe is built. For the latter, the framework supports job submission to a volunteer computing system, named Computer Game Desktop Grid developed by our team, to help solve or analyze game positions. Currently, we support JL proof number search, JL alpha-beta search, and JL Monte-Carlo tree search. In this demonstration, we show how easily JL-PNS is designed.

Keyword: Software framework, game record editor, CGDG, Job-level search, volunteer computing.

1. Introduction

To facilitate research in the computer game area, this demonstration paper demonstrates our software framework [7] for generic game development, including game record editing and job-level computing. For the former, the framework supports the display and editing of game positions and the browsing of the game position tree. For the latter, the framework supports job submission to a volunteer computing system, named *Computer Game Desktop Grid* [10] (*CGDG*) developed by our team, to help solve or analyze game positions.

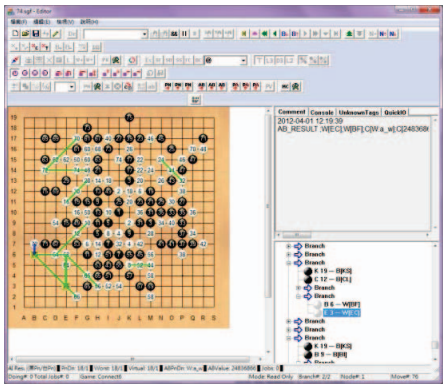


Figure 1. Connect6 Editor.

For the former, we have currently developed game record editors for Connect6 [11][12], Go, Chinese Chess, Mahjong, etc. The editor for Connect6 is illustrated in Figure 1. For the latter, we have developed a lot of job-level (JL) search algorithms such as JL proof number search (JL-PNS) [1][2][13][14], JL Monte-Carlo Tree Search (JL-MCTS) [3][6], etc. to solve many game problems, such as the openings of Connect6, and the openings of Chinese Chess.

Most importantly, both game record editing and job-level computing are orthogonal in the aspect of software development. Game developers can design a game editor for their game by extending the base modules which are extracted from the code may replicate; while JL search algorithm developers can easily implement their own JL search algorithms in a similar way. When completing the design of the game record editor for a new game, say Chinese Chess, all the designed JL algorithms can be easily applied to the new game. On the other hand, when a new JL algorithm is designed, it can be applied to all the different games.

In this demonstration, we illustrate the framework by designing an editor for the game Tic-Tac-Toe and JL-PNS. Section 2 reviews the software framework [7], Section 3 demonstrates the Tic-Tac-Toe editor and JL-PNS, and Section 4 lists the links to the demonstration system.

2. System Description

In this section, the software framework [7] is reviewed. This software framework includes two major modules: game record editing module and job-level module. The game record editing module support game developers to construct a basic game record editor for their game, and the job-level module provides an algorithm template to quickly implement JL search algorithms. Both are summarily reviewed in the following two subsections, while the details are described in [9].

2.1 Game Editing Module

As shown in Figure 2 (below), the game editing module uses MVC design pattern [8][9] and includes two components: model and view corresponding to the same name in MVC. The controller is not encapsulated into a class because we use MFC to

978-0-7695-4919-4/12 $26.00 © 2012 IEEE
DOI 10.1109/TAAI.2012.55

construct our software framework which has its own mechanism to map UI events to functions.

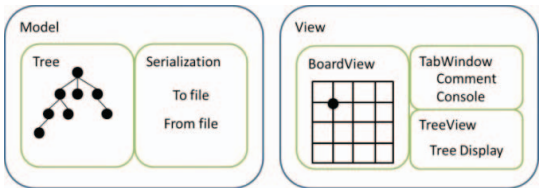


Figure 2. Game Editing Module.

The model has two main parts, tree structure and move interpretation, and some minor functionalities; the view includes three sub-views, named `BoardView`, `TreeView`, and `TabWindow`, which present a position in their own ways.

In the model, for the tree structure, we use left-child right-sibling binary tree with two kinds of node: move node and branch node, each contains information about a move and a branch respectively. Both kinds of nodes include links to maintain the tree structure. To make tree access easier, the links are hidden from developers, and a tree modification helper, named `NodePointer`, is provided. It implements visitor and iterator design pattern and simplifies tree traversal from developers.

For move interpretation, game records are saved into a file and loaded from a file in Smart Game Format (SGF), and some abstract serialization classes are supported to interpret files for different games.

In the view, a board view is used to draw the board which presents the current game position. To facilitate game developerment, we encapsulate the board view into an abstract class which provides utilities such as line drawing, text drawing, and symbol drawing, and one function, named `drawBoard`, remains unimplemented so that game developers can define how to draw a board.

A tree view is to show what the game position tree looks like. If game developers have to customize it, they can override abstract functions of the base classe, named `BaseTreeView`. A tab window supports some functionalities such as displaying comment, SGF tags of position, and debugging console in tabs, and provides a convenient interface to allow developers to add new tab sub-windows on it.

In this module, the following base classes are supported for game editor developers.

- `BaseEditorDoc`: The model class of our framework
- `BaseSgfParser`: The class for move interpretation, which implements SGF structure parsing so that developers can implement a game parser by writing only few lines of code about move parsing.
- `BaseSgfSerializer`: The class, similar to `BaseSgfParser`, that implements SGF structure serializing for developers.
- `BaseBoardView`: the class that supports most APIs for drawing a game board.
- `BaseTreeView`: the class that supports most APIs for maintaining the tree view window.
- `BaseTabWindow`: the class that displays a tab and the client tab page windows.

2.2 Job-Level Module

Job-level module offers a template of JL search algorithm which uses a design pattern named *template method*, and provides the accessibility to CGDG. JL developers can quickly implement their JL search algorithm by extending this template. According to [7], the template includes eight functions grouped into three event handlers, initialization event handler, idle worker event handler, and returning job result event handler.

The initialization event is triggered when a user demands to start. One function supporting this is `initialize`.

Worker idling event is triggered when an idle worker is available. The functions supporting the handling are `select`, `preupdate`, and `dispatch`.

Returning job message event is triggered when a job result is returned. The functions supporting the handling are `parse`, `update`, `checkfinish`, and `finalize`.

We provide a template class, named `BaseJobLevelAlgorithm`, to implement a JL algorithm. The above eight functions are abstracted in the class for JL algorithm developers to easily implement their JL search algorithm.

On the other hand, JL algorithm should be adaptive. That is, how to decide something according to games should be able to be customized. We introduce an interface for game policy to easily apply a JL algorithm to a game. Once a JL algorithm developer designed a JL algorithm with the corresponding policy, a game developer can easily adapt the JL algorithm by providing an implementation of that policy. It makes development of game and JL algorithm orthogonal.

3. System Demonstration

We illustrate the framework by designing the Tic-Tac-Toe editor and JL-PNS.

To design game record editors, game developers should extend the six base classes, described in Subsection 2.1: BaseEditorDoc, BaseSgfParser, BaseSgfSerializer, BaseBoardView, BaseTreeView, and BaseTabWindow, and override those pure virtual

functions, if any. For the Tic-Tac-Toe editor, we customize (redefine) five of the six classes, excluding `BaseTabWindow` whose customization is not needed.

Table 1. Functions to be overridden in Tic-Tac-Toe Editor

`TicTacToeEditorDoc::addOnePiece(…)`
`TicTacToeSgfParser::parseMove(…)`
`TicTacToeSgfSerializer::serializeMove(…)`
`TicTacToeBoardView::drawBoard(…)`
`TicTacToeTreeView::getText(…)`

In these classes, the methods to be overridden or redefined are listed in Table 1. The method `addOnePiece` is required since each game may have different way to add a piece, such as Chinese Chess, Connect6. The methods `parseMove` and `serializeMove` are required for input and output. The methods `drawBoard` and `getText` are required for displaying the board.

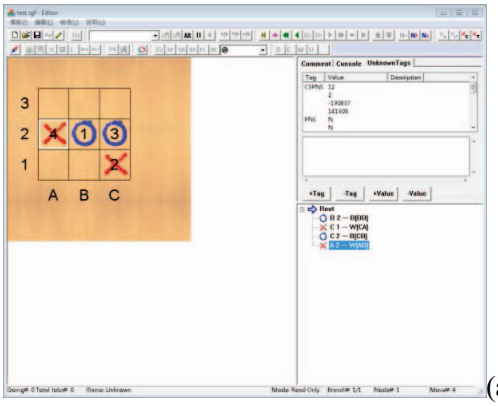
To develop a JL algorithm, JL algorithm developers should inherit `BaseJobLevelAlgorithm`, and override those eight abstract functions (described in Subsection 2.2) to define the algorithm, i.e. JL-PNS here. The eight functions should be overridden are listed in Table 2 (below).

Table 2. Functions to be overridden in Job-Level PNS

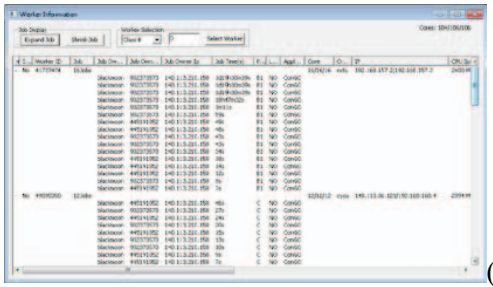
`JobLevelProofNumberSearch::initialize(…)`
`JobLevelProofNumberSearch::select(…)`
`JobLevelProofNumberSearch::preupdate(…)`
`JobLevelProofNumberSearch::dispatch(…)`
`JobLevelProofNumberSearch::parse(…)`
`JobLevelProofNumberSearch::update(…)`
`JobLevelProofNumberSearch::checkFinish(…)`
`JobLevelProofNumberSearch::finalize(…)`

To apply job-level PNS to Tic-Tac-Toe, game developers should implement a game policy TicTacToePnsPolicy extending PnsPolicy. Similarly, to apply job-level PNS to Connect6, game developers should implement a game policy Connect6PnsPolicy extending PnsPolicy.

After the above development, we list the functionalities supported by this framework as follows:



(a)


(b)

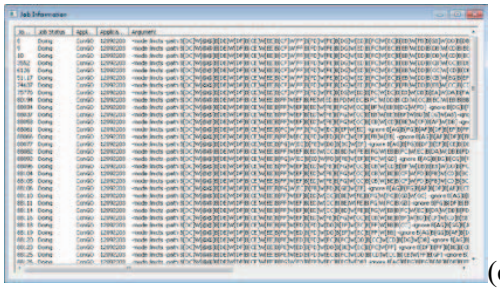

(c)

Figure 3. Some displays of the demonstration.

1. Game editing related functionalities:
   a. Display and edit game positions, including rotation, comments and tags of positions. (Figure 1 and Figure 3(a))
   b. Browse and edit the position tree.
   c. Access files.
2. Job-level related functionalities:
   a. Access CGDG, our volunteer computing system.
   b. Display the status of workers. (Figure 3 (b))
   c. Set the worker filtering. For example, if we only implement a job, a game-playing program, on Windows, the workers running on Unix should be filtered out.
   d. Submit jobs to CGDG.
   e. Support operations for JL-PNS.
   f. Support operations for JL program competition.
   g. Display current jobs in CGDG, as shown in Figure 3 (c).
3. Other functionalities:
   a. Support database accesses for openings.
   b. Support console display for debugging.

The current status of this framework is listed in Table 3 (below). Circle denotes finished projects and triangle denotes ongoing projects. The line counts of each game and the base module are listed in Table 4 (below). The table shows that this framework saves a lot of efforts in developing games and JL algorithms. This framework have helped us to win many tournaments [5][15][16][17][18] and helped the researches in [13][14].

Table 3. Current Status.

Status	Pure Algorithm	Connect6	Go	Chinese Chess	Mahjong	Tic-Tac-Toe
Pure Editor		O	O	O	O	O
JL-PNS	O	O				
JL-MCTS	O	Δ	O			
JL-SSS*	Δ			Δ		
AI Competition	O	O		O		

Table 4. Line Counts.

The original Connect6Lib and JL-PNS	Game record editing module	Job-level module	Connect6	Go	Chinese Chess	Mahjong	Tic-Tac-Toe
86688	27854	6658	8215	8843	3535	2192	836

4. Download Link

A demonstration version is available at http://connect6.csie.nctu.edu.tw/Editor.zip. It includes a game record editor for Tic-Tac-Toe that is a Windows executable binary, source files of classes for Tic-Tac-Toe developers, and source files of JL-PNS algorithm for JL developers.

Acknowledgement

The authors would like to thank the National Science Council of the Republic of China (Taiwan) for financial support of this research under contract numbers NSC 95-2221-E-009-122-MY2 and NSC 97-2221-E-009-126-MY3.

Reference

[1] Allis, L.V., Searching for solutions in games and artificial intelligence, Ph.D. Thesis, University of Limburg, Maastricht, The Netherlands, 1994.

[2] Allis, L.V., Meulen, M. van der, and Herik, H. J. van den, Proof-number search, Artificial Intelligence, Vol. 66(1), pp. 91-124, 1994.

[3] Browne, C., Powley, E., Whitehouse, D., Lucas, S., Cowling, P., Rohlfshagen, P.,Tavener, S., Perez, D., Samothrakis, S., and Colton, S., A Survey of Monte Carlo Tree Search Methods, the IEEE Transactions on Computational Intelligence and AI in Games, Vol. 4(1), Forthcoming, 2012.

[4] Chen, C.-P., Wu, I.-C., and Chan, Y.-C., Connect6Lib – A Connect6 Editor, available at http://www.connect6.org/Connect6Lib_Manual.htm, 2009.

[5] Chou, C.-W., Yen, S.-J., and Wu, I.-C., "TAAI 2011 Computer Go Tournaments," ICGA Journal, vol. 34, no.4, 2011, pp. 251-252.

[6] Coulom, R., Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. Proceedings of the 5th International Conference on Computer and Games (eds. H. J. van den Herik, P. Ciancarini, and H. J. Donkers), Vol. 4630/2007 of Lecture Notes in Computer Science, pp. 72–83, Springer, Turin, Italy, 2006.

[7] Liu, H.-Y., Wu, I.-C., Kang, H.-H., and Liao, T.-F., "Software Framework for Generic Game Development in CGDG", International Computer Symposium (ICS2012), Hualien, Taiwan, December 2012.

[8] Trygve Reenskaug, Models-Views-Controllers, 1979.

[9] Trygve Reenskaug, THING-MODEL-VIEW-EDITOR an Example from a planningsystem, 1979.

[10] Wu, I.-C., Chen, C.-P., Lin, P.-H., Huang, K.-C., Chen, L.-P., Sun, D.-J., Chan, Y.-C., and Tsou, H.-Y., A Volunteer-computing-based grid environment for Connect6 applications, in IEEE Int. Conf. Comput. Sci. Eng., Vancouver, BC, Canada, Aug. 29–31, 2009, pp. 110–117.

[11] Wu, I.-C., Huang, D.-Y., and Chang, H.-C., Connect6. ICGA Journal, Vol. 28(4), pp. 234-242, 2006.

[12] Wu, I.-C. and Huang, D.-Y., A New Family of k-in-a-row Games. The 11th Advances in Computer Games Conference (ACG'11), pp. 180-194, Taipei, Taiwan, 2005.

[13] Wu, I.-C., Lin, H.-H., Lin, P.-H., Sun, D.-J., Chan, Y.-C., and Chen, B.-T., "Job-Level Proof-Number Search for Connect6", The International Conference on Computers and Games (CG 2010), Kanazawa, Japan, September 2010.

[14] Wu, I.-C., Lin, H.-H., Sun, D.-J., Kao, K.-Y., Lin, P.-H., Chan, Y.-C., and Chen, B.-T., "Job-Level Proof-Number Search for Connect6", submitted to IEEE Transactions on Computational Intelligence and AI in Games (IEEE TCIAIG).

[15] Wu, I.-C., and Lin, P.-H., NCTU6-Lite Wins Connect6 Tournament, ICGA Journal, Vol. 31(4), 2008.

[16] Wu, I.-C., Lin, Y.-S., Tsai, H.-T., and Lin, P.-H., "The Man-Machine Connect6 Championship 2011", ICGA Journal (SCI), vol. 34, no. 2, June 2011.

[17] Yang, J.-K., Su, T.C., and Wu, I.-C., "TCGA 2012 Computer Game Tournament Report," submitted to ICGA Journal, 2012.

[18] Yen, S.-J., Su, T.-C., and Wu, I.-C., "The TCGA 2011 Computer-Games Tournament", ICGA Journal (SCI), vol. 34, no. 2, June 2011.