國立交通大學

資訊工程學系

碩士論文

通用棋譜編製系統之研究

The Study of Generic History Authoring Systems

for Chess-Like Games

研 究 生：陳家齊

指導教授：吳毅成　教授

中華民國九十三年六月

通用棋譜編製系統之研究
The Study of Generic History Authoring Systems for Chess-Like Games

研 究 生：陳家齊　　Student：Chia-Chi Chen

指導教授：吳毅成　　Advisor：I-Chen Wu

國 立 交 通 大 學
資 訊 工 程 學 系
碩 士 論 文

A Thesis
Submitted to Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Computer Science and Information Engineering

June 2004

Hsinchu, Taiwan, Republic of China

中華民國九十三年六月

通用棋譜編製系統之研究

研究生：陳家齊　　指導教授：吳毅成

國立交通大學 資訊工程學系

摘要

棋譜編輯系統向來是棋類遊戲愛好者不可或缺的好幫手。市面上各種棋類遊戲專屬的棋譜編輯系統眾多，具有許多共通的功能和特性。如果能夠為棋譜編輯系統設計一個通用的開發平台，那麼就能夠快速建立各種遊戲的棋譜編輯系統。

本篇論文研究市面上的棋譜編輯系統，並根據本實驗室所提出的 POT 通用遊戲模組理論與 GAML 通用棋譜格式，建構出通用棋譜編輯系統開發平台。最後以實例說明如何使用本平台來開發棋譜編輯系統。

The Study of Generic History Authoring Systems for Chess-Like Games

Student: Chia-Chi Chen Advisor: I-Chen Wu

Department of Computer Science and Information Engineering

National Chiao Tung University

ABSTRACT

History authoring systems are always good helpers to chess-game players. There are many history authoring systems for different chess games currently and have many similar functions and properties. If we can develop a generic history authoring system for chess-like games, then we can develop a history authoring system for each game efficiently.

This thesis studies current history authoring systems. According to the general game model, POT, and the general data format, GAML which are both proposed by our lab, we develop a generic history authoring system for chess-like games. We demonstrate how to develop a history authoring system for each game with exact examples at the end.

誌謝

首先要感謝我的指導教授，吳毅成博士，由於他不厭其煩的細心指導，這篇論文才得以順利完成。

此外特別要感謝汪益賢和徐健智兩位學長，在研究的過程中給予我許多寶貴的意見和指導，以及在實驗室的同學朱俊欣、陳智文、林義欽和所有其他伙伴們在研究及生活上的幫助。同時，也要感謝許許多多的同學及朋友，在我的研究期間，給了我相當多的關懷與鼓勵，陪我度過這段最值得回憶的學生生活。

最後，我要感謝我的父母，在我的求學生涯中給了我最大的支持和照顧。謹以此論文，獻給我最摯愛的家人。

目錄

表目錄

圖目錄

第一章　緒論

1.1 研究目標

近年來電腦的快速發展，人們越來越習慣以電腦作為資訊儲存和交流的媒介。存在電腦之中的檔案，不但能夠長久的保存，更能夠透過電子郵件和網際網路與其他人作快速的資訊交流。

在許多棋類遊戲中，正式比賽的棋譜與大盤分析都是相當珍貴的學習教材。若能有一個電腦軟體提供編輯棋譜以及相關資料的功能，進而儲存成棋譜記錄檔案，除了有利於交流與學習之外，棋類愛好者也可以透過相同的介面記錄自己所對弈過的棋局。目前市面上已有許多發展此類功能的軟體，我們一般稱之為「棋譜編輯器」，或「棋譜編輯系統」。

目前市面上的棋譜編輯系統種類繁多，如圍棋、五子棋、象棋等，都擁有許多種棋譜編輯系統。這些針對不同棋類遊戲所設計出來的棋譜編輯系統，大部分擁有共通的功能及特性。另外，像是西洋棋、象棋和日本將棋[1]雖然是三種不同的棋類遊戲，棋子與玩法卻非常相近。因此，如果能夠研究出棋類遊戲共通的模式與棋譜編輯系統共通的功能，進而建構出一套通用棋譜編輯系統開發平台，未來就能夠快速地為各種棋類遊戲建立自己的棋譜編輯系統了。

因此，本篇論文的目標在於研究目前市面上各種棋類遊戲的棋譜編輯系統，找出之間的共通點，並根據本實驗室所提出的 POT [2] 通用遊戲模式理論和通用棋譜格式 GAML[3]，設計一個通用棋譜編輯系統開發平台。透過這個平台來開發的棋譜編輯系統，能夠使得開發時間更短，程式碼更少，同時並保有各個遊戲的特性。

1.2 論文大綱

在本論文第二章的背景說明裡，將會詳細地介紹並比較現有的棋譜編輯系統和棋譜格式。另外，為了使我們的棋譜編輯系統開發平台能夠適用於多種遊戲，我們需要建構一個棋類遊戲的通用模式，因此，在這一章中也會介紹本實驗室先前所提出的 POT[2] 桌上型遊戲模式理論和 GAML[3]通用棋譜格式。

第三章說明我們如何設計一個通用棋譜編輯系統開發平台，包含系統架構以及系統模組設計。第四章是系統實作的部分，在這一章中，以象棋和五子棋為例，說明如何使用本篇論文所提出的通用棋譜編輯系統開發平台來開發棋譜編輯系統。第五章是結論與未來展望。最後，在本篇論文的附錄中，附上使用本平台來開發五子棋棋譜編輯系統的完整實作。

第二章　背景說明

本章介紹通用棋譜編輯系統開發平台的相關背景。在 2.1 節當中，詳細介紹目前市面上圍棋、五子棋與象棋三種棋類遊戲的棋譜編輯系統，並找出其共通的功能與特性。在 2.2 節中介紹棋類遊戲的通用遊戲模式。在 2.3 節中探討棋譜編輯系統所使用的棋譜存檔格式。

2.1 現有棋譜編輯系統之概述

由於棋譜編輯系統對於研究棋藝有著不可或缺的必要性，因此許多的棋類遊戲愛好者，紛紛開發出自己所喜愛的棋類遊戲棋譜編輯系統，就以國內較為熱門的圍棋、象棋、和五子棋而言，圍棋助手[4]、WinMGT[5]、Go2000[6]、MultiGo[7]等是幾個比較熱門的圍棋棋譜編輯系統；五子棋有 Renlib[8]、GoRenjer[9]等；象棋則以象棋演播室[10]和象棋橋[11]為大宗。我們就以圍棋的圍棋助手[4]、五子棋的 Renlib[8]和象棋的象棋演播室[10]做為例子，以介面設計、操作模式和棋譜格式這三方面來深入的瞭解這三種不同棋類遊戲的棋譜編輯系統。

2.1.1 圍棋助手 Go Game Assistant

圍棋助手[4]，是針對圍棋所發展出來的棋譜編輯系統，作者為大陸的胡小奇先生，目前最新版本為 8.30，有簡體中文和繁體中文兩種版本，是目前相當熱門的圍棋棋譜編輯系統。圖 2-1 顯示主要的程式畫面：

圖 2-1、圍棋助手 Go Game Assistant 的程式畫面

以下以介面設計、操作模式和棋譜格式這三方面作介紹：

◆ 介面設計：

整個程式的介面設計是以左半部的棋盤為主，右上角的註解區和右下角的樹狀棋步列表為輔，另有標題列、功能表、工具列和狀態列等。左半部的棋盤用以模擬真實的下棋情境，顯示目前的盤面狀況，同時也能直接在棋盤上下子或標示標記符號。右上角的註解區域用來輸入及顯示棋局和棋步的相關說明。右下角的當前棋局視窗以樹狀結構顯示所有棋步、分支和標記節點，讓使用者可以清楚得知道目前所觀看的棋步，是位於整個棋譜的哪一個部分。

另外，在右下角的區域另可選擇縮圖（棋局的簡略版，分支點之間的棋步以一個節點表示）、圍棋教程（圍棋教學棋譜庫）、研究

心得（圍棋題庫）、經典棋譜（名家實戰譜）與對局面版（操作棋步及打譜相關設定）等選擇。

◆ 操作模式：

操作模式以編輯和瀏覽兩個部分分別作討論。

- 編輯：

使用者在棋盤上可以滑鼠左鍵點選棋盤空點來下子。在下子的過程中，程式會自動控制黑白兩子交替落子。如果想要編輯分支，可以在當前棋局中按滑鼠右鍵選擇「配參考圖模式」，新增的一手就會是分支中的第一步。另外在功能表和工具列上以選單和按鈕的方式讓使用者添加標記符號和棋子（圖 2-2）。

圖 2-2、圍棋助手的添加標記功能

- 瀏覽：

棋譜的瀏覽可透過右下角的樹狀棋譜列表，直接選擇欲觀看的棋步，或者點選工具列表中的按鈕（圖 2-3），依序表示可跳至第一步（局始）、前十步、前一步、後一步、後十步、最後一步（局終）、指定點、讓棋譜自動播放、上一分支點（參考圖）、下一分支點（參考圖）或上一註解、下一註解。另外，在棋盤上也有符號標明下一步的位置（包含主幹和分支部

分），使用者也可以從棋盤直接點選下一步的位置，而進入含有此步的分支中作觀看（圖 2-4）。

圖 2-3、圍棋助手的瀏覽棋步功能按鈕

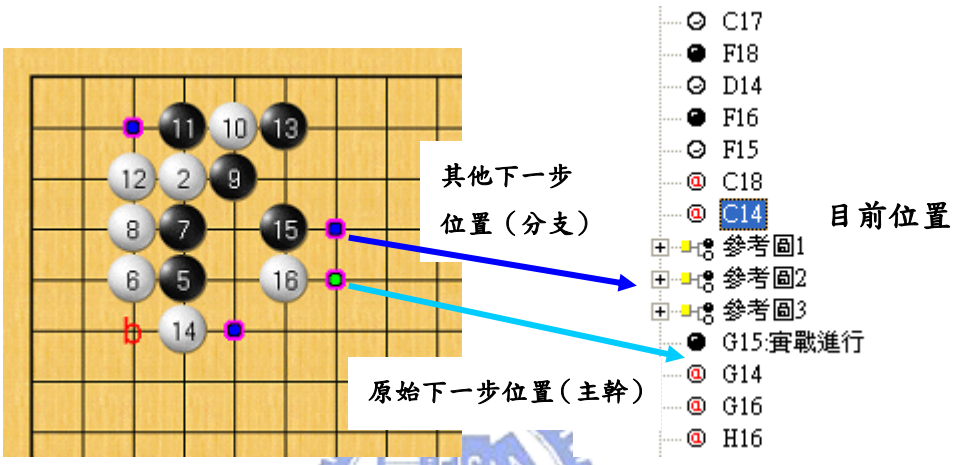


圖 2-4、圍棋助手中標示下一步與分支的提示圖案

◆ 棋譜格式：

目前圍棋助手所支援的棋譜格式主要為自訂的 GOA 和通用的 SGF[12]兩種，另外也能讀取 MGT、NGF、UGF、BDX、GOS、IGO、GO、MAN 和 GIB 等其他圍棋棋譜編輯器統所制訂的棋譜格式檔案。

2.1.2 Renlib

Renlib[8]是針對五子棋所開發的棋譜編輯系統，作者為瑞典的 Frank Arkbo, Uppsala。目前只有支援英文，最新的版本為 3.4.2。程式的主要畫面見圖 2-5：

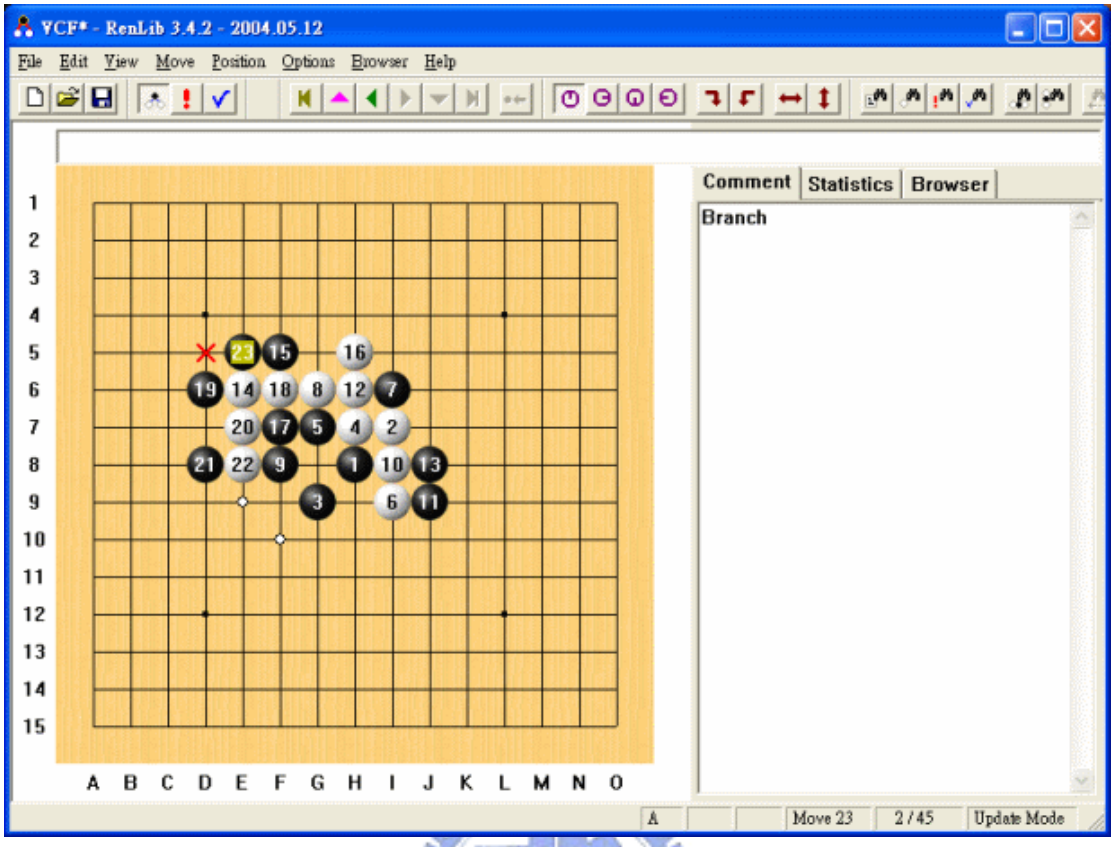


圖 2-5、Renlib 的程式畫面

以下以介面設計、操作模式和棋譜格式這三方面作介紹：

◆ 介面設計：

整個程式的介面設計是以左半部的棋盤為主，右方的註解區為輔，另有標題列、功能表、工具列和狀態列等。左半部的棋盤顯示目前的盤面狀況，同時也能直接在棋盤上下子。右方的註解區域用來輸入及顯示棋局和棋步的相關說明。

Renlib 並沒有設計樹狀結構的棋譜列表，但會在棋盤上以圖案標示所有下一步，如圖 2-6 左圖中棋盤上的白點表示白子下一步位置；右圖中棋盤上的黑點表示黑子下一步位置。

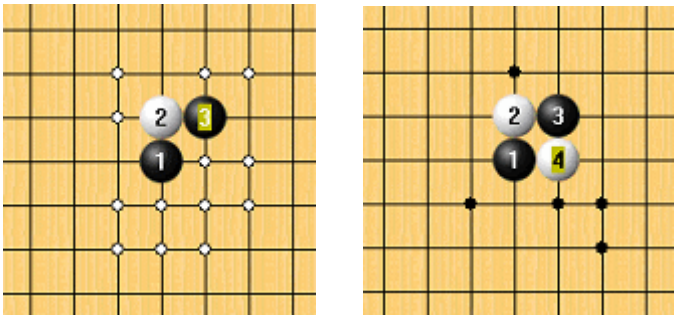


圖 2-6、Renlib 中表示下一步位置的提示圖案

◆ 操作模式：

操作模式以編輯和瀏覽兩個部分分別作討論。

- 編輯：

使用者在棋盤上可以滑鼠左鍵點選棋盤空點來下子。在下子的過程中，程式會自動控制黑白兩子交替落子。另外，可以透過棋盤上點選滑鼠右鍵或按下鍵盤方向鍵「←」回到上一步下子，來編輯不同的走法（在 Renlib 中並無主幹與分支的概念，皆視為可走的下一步）。另目前 Renlib 中並沒有支援標記的部分。

- 瀏覽：

棋譜的瀏覽以在棋盤上點選可走的下一步為主。另外在工具列中亦有按鈕（圖 2-7），依序可供使用者跳至第一步、上一分支點、上一步、下一步、下一分支點、最後一步等。

圖 2-7、Renlib 的瀏覽棋步功能按鈕

◆ 棋譜格式：

目前 Renlib 所支援的棋譜格式主要為自訂的 LIB 和另一五子棋棋譜編輯系統 GoRenjer[9]所制訂的 PDB 兩種，另外也能讀取 WZQ、RENJS、REN 等其他五子棋棋譜編輯系統所制訂的棋譜格式和通用棋譜格式 SGF[12]的檔案。

2.1.3 象棋演播室 XQStudio

象棋演播室[10]是針對中國象棋所開發的棋譜編輯系統，作者為大陸的董世偉先生，目前最新的版本為 1.62 的簡體中文版與 1.6 的繁體中文版。在這裡以 1.6 的繁體中文版作為介紹的範例。圖 2-8 為程式的主要畫面：

圖 2-8、象棋演播室 XQStudio 的程式畫面

以下以介面設計、操作模式和棋譜格式這三方面作介紹：

◆ 介面設計：

整個程式的介面設計是以左半部的棋盤為主，右半部的棋譜列表、註解區、所有走法視窗為輔，另有標題列、功能表、工具列和狀態列等。左半部的棋盤與前兩種系統類似，用來顯示目前的盤面狀況，同時也能直接在棋盤上移動棋子來編輯棋步。

在象棋演播室中的棋譜列表與圍棋助手中的不同，並不是以樹狀的結構來顯示所有的主幹和分支紀錄，而是只有顯示從第一步到某一個分支的最後一步。換句話說，以樹狀結構而言，這裡只有顯示一條從根節點到某一個葉節點的路徑。所有的這一步走法都會顯示在右下角的視窗，使用者可以選擇棋步來進入不同的分支。另外，右上角的註解區域可用來輸入及顯示棋局和棋步的相關說明。

◆ 操作模式：

操作模式以編輯和瀏覽兩個部分分別作討論。

● 編輯：

使用者可以使用滑鼠點選欲移動的棋子，再選擇欲移動的目的地來編輯棋步，或者使用拖拉棋子的方式亦可。程式會控制紅黑方必須交替出手。欲編輯分支（變著）時，需按走法視窗下的  ，再新增一手。另外，在新增文件時，可以同時輸入其他相關資訊（圖 2-9），及使用拖拉棋子的方式來擺設初始棋盤（圖 2-10）。

圖 2-9、象棋演播室的遊戲資訊輸入畫面

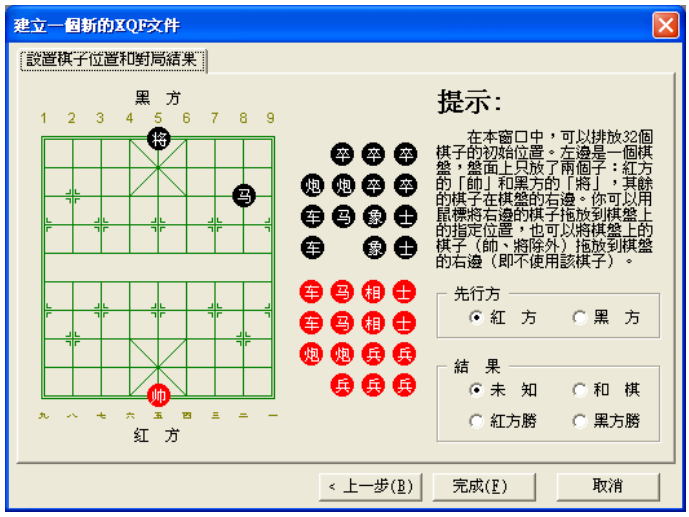


圖 2-10、象棋演播室的擺譜畫面

- 瀏覽：

使用者可以點選棋譜列表觀看棋步或在走法視窗點選變著進入分支。另外使用棋譜列表下的按鈕列（圖 2-11）依序可供使用者跳至第一步、前一步、後一步、最後一步，另外在棋譜上快點兩下可以從這步開始自動播放。

圖 2-11、象棋演播室的瀏覽棋步功能按鈕

◆ 棋譜格式：

目前象棋演播室只有支援自訂的 XQF 格式。

2.1.4 綜合模式

在介紹完三種不同棋類遊戲的棋譜編輯系統之後，我們將這三種不同的系統加以整合，並定義出一個棋譜編輯系統的介面設計與操作模式如下：

◆ 介面設計：

左半部以虛擬棋盤來模擬真實的下棋情境，右上角擺設註解區域來輸入及顯示棋局和棋步的相關說明。右下角以樹狀結構顯示所有棋步、分支和標記節點。

◆ 操作模式：

操作模式以編輯和瀏覽兩個部分分別作討論。

- 編輯：

  要能夠提供使用者編輯遊戲的初始棋盤、棋步、標記、分支、註解及棋局相關資訊等功能：

  1. 編輯初始棋盤：

     使用滑鼠拖曳或點選棋子，使棋子在棋盤內外移動。

  2. 編輯棋步：

     使用滑鼠選點棋盤上的空格、空點或棋子來新增棋步；由按鈕、滑鼠右鍵功能選單或鍵盤 Del 鍵來刪除棋步。

  3. 編輯標記：

     由功能表選單或工具列按鈕選擇欲新增的標記種類後，以

滑鼠點選棋盤上的空格、空點或棋子來新增標記；在棋譜列表選擇欲刪除的標記，由按鈕、滑鼠右鍵功能選單或鍵盤 Del 鍵來刪除。

4. 編輯分支：
   新增的另一棋步自動視為分支或由使用者按下意為新增分支的按鈕後新增棋步。

5. 編輯註解：
   以註解區編輯註解（包含新增、刪除、修改）。

6. 編輯棋局相關資訊：
   在對話框中提供表格讓使用者輸入。

- 瀏覽：
  提供棋譜列表陳列所有棋步、分支與標記，讓使用者可以自由選擇目前所要看的部分。在工具列上也提供按鈕讓使用者可以跳至第一步、上一分支點、上一步、下一步、下一分支點、最後一步等。

經由以上的結果，我們已經瞭解一個棋譜編輯系統普遍的架構與功能。接下來，我們需要一個通用的遊戲模式來描述及定義棋類遊戲的流程和狀態，以便能夠確立通用棋譜編輯系統開發平台的內部模式。同時，我們也需要一個通用的棋譜格式，統一不同遊戲的棋譜紀錄。

2.2 通用遊戲模式概述

為了建立通用棋譜編輯系統開發平台，我們必須要瞭解棋類遊戲的流程，找出共通性並且將流程模式化，如此方能確立平台的通用性，以及所適用的遊戲範圍。本節中介紹本實驗室所提出的通用遊戲模式，及其適用的遊戲範圍。

2.2.1 Play-on-table Game Model

我們實驗室在日前提出桌上型遊戲模式（Play-on-table Game Model，簡稱為 POT）理論[2]，描述桌上型遊戲的流程：玩家圍繞在遊戲桌旁，操縱一些遊戲的實體物件，例如：棋子、撲克牌等等來進行遊戲。遊戲的過程中，有一些物件可能會屬於玩家私有的範圍，只有特定的玩家才能看見，例如在橋牌中，玩家拿在手中的牌。另外有一些物件放置在公開的位置，所有的玩家都看的到。

在 POT 的模式中，所有的遊戲物件在遊戲過程中不會消失，而玩家對遊戲物件的操作可以簡化為「移動」與「翻轉」兩種。遊戲本身的狀態可以遊戲物件的狀態來表示，而遊戲的過程，可視為遊戲物件一連串狀態改變的結果。

符合 POT 遊戲模式的遊戲我們稱之為 POT Game，包含大部分的傳統桌上型遊戲，如：麻將、象棋、圍棋、五子棋和牌類遊戲如：橋牌、大老 2 等等。

由於 POT 模式較為複雜，且包含許多不適用於棋譜記錄的遊戲，因此，我們將重點放在 POT 模式的子集，SPOT（Simplified POT）模式，並於下一小節中作詳盡介紹。

2.2.2 SPOT

SPOT 是 POT 遊戲模式的子集合。在 SPOT 遊戲模式中，所有的遊戲物件放置在公開的位置上。滿足 SPOT 遊戲模式的 SPOT 遊戲可以定義為：G = (A, O, S, M)，A, O, S, M 的定義分別敘述於下：

- A (代表 Area)：
  A 表示遊戲中可放置遊戲物件的區域。在 SPOT 的架構中，遊戲只有一個公開的區域。

- O (代表 Object)：
  O 表示有限個遊戲物件的集合。每個物件的狀態是以 P (Position) 和 F (Face) 來表示；遊戲的狀態為所有遊戲物件狀態的集合。

- S (代表 State)：
  S 表示遊戲狀態的集合。一般而言，至少會包含{(start),(end)}。遊戲由（start）狀態開始，到（end）狀態結束。

- M (代表 Operation)：
  M 表示遊戲中合法動作的集合。一個合法的動作可定義為（s, s')，表示這個動作使得遊戲的狀態由 s 改變成 s'。

根據 SPOT 遊戲模式，我們可以定義出大部分的棋類遊戲流程。因此，依照 SPOT 遊戲模式來設計的系統，就能夠適用於大部分的棋類遊戲。我們的通用棋譜編輯系統開發平台即是遵循 SPOT 遊戲模式來開發，因此能夠達到適用於不只一種棋類遊戲的目的。不過，由於會拿來記棋譜的遊戲，多半以 Perfect information game[13] (如：象棋、五子棋、圍棋等)為主，因此我們的系統在設計時，也是以這類遊戲為主要考量。

2.3 現有棋譜格式概述

目前市面上的棋譜格式主要來自於棋譜編輯系統所使用的檔案格式及一些也支援打譜或存檔功能的棋類線上遊戲網站（如弈天）。在這裡我們將棋譜格式分為兩類，一類稱為專屬棋譜格式，另一類稱為通用棋譜格式。

專屬棋譜格式指的是針對某一個特定的棋類遊戲所訂定的格式，不能用於儲存它種遊戲的棋譜檔案。這類格式最主要的來源為各個棋譜編輯系統的自訂格式，和支援存檔功能的線上遊戲網站自訂格式。

通用棋譜格式顧名思義可以適用於兩種以上的棋類遊戲存檔之用，目前市面上這類的棋譜格式較少，目前有部分棋譜編輯系統有支援。以下就針對這兩種格式分別做詳細的介紹。

2.3.1 專屬棋譜格式

市面上的棋譜編輯系統大多都設計有自己專屬的格式，幾乎是每多一個棋譜編輯系統，就多一個存檔格式。如之前所提到過的圍棋助手[4]的 GOA 格式、Renlib[8]的 LIB 格式、象棋演播室[10]的 XQF 格式等等皆屬此類。

隨著棋譜編輯系統的盛行，各式各樣的棋譜格式也就到了令人眼花撩亂的地步。同樣的一份棋譜記錄，可能會有好幾種版本的格式檔案；一個棋譜編輯系統也可能為了增加市場佔有率，而被迫支援好幾種其他的棋譜格式。

對於使用者而言，棋譜編輯系統所支援的棋譜格式種類限制了他所能瀏覽的棋譜種類。如果使用者有興趣的一份棋譜紀錄的格式，不在目前已安裝好的棋譜編輯系統的支援範圍內，就被迫要安裝可以讀取這種格式的棋譜編輯系統。因此，不論是對於棋譜本身的流通性或者是對使用者的方便性而言，棋譜格式都有整合統一的必要性。

2.3.2 通用棋譜格式

目前市面上最廣為使用的通用棋譜格式，莫過於 SGF[12]。另外，本實驗室也針對桌上型遊戲發展出以 XML[14]為基礎的 GAML[3]語言，在接下來的章節中，讓我們來深入的瞭解這兩種通用棋譜格式的適用範圍和優缺點。

2.3.2.1 SGF

SGF[12]是 Smart Game Format 的縮寫，是一種純文字、樹狀結構的棋譜格式。它只能支援兩人對弈的遊戲，如圍棋、五子棋、Backgammon、Lines of Action、Hex、Amazons、Octi 和 Gess 等。其中最廣為使用 SGF 作為存檔格式的是圍棋，因此不少圍棋棋譜編輯系統都有支援 SGF 格式。

SGF 檔案中，每一個節點（Node）是以分號“；”作為開始，同時也是以分號作為節點間的區隔。一個節點中可能包含一個或多個屬性（Property），每一個屬性的格式為：**屬性名[屬性值]**。例如：

```
;B[ki]C[Black move];W[jk]
```

表示兩個節點，一個是 ;B[ki]C[Black move] 、另一個是 ;W[jk]。第一個節點中包含兩個屬性 B[ki] 與 C[Black move]，分別表示「下一

黑子至（k, i）位置」與「註解：Black move」，另一個節點中的 W[jk] 則表示「下一白子至（j, k）位置」。

SGF 中除了上述提到的可描述棋步和註解的屬性之外，也支援可描述擺譜、分支、標記以及遊戲相關資訊如：日期、雙方棋手姓名等記錄的屬性。

在 SGF 中，是以小括號 () 來表示分支，並依照 Preorder 的順序輸出，第一個分支視為主幹。假設有一個棋譜紀錄可以圖 2-12 表示：

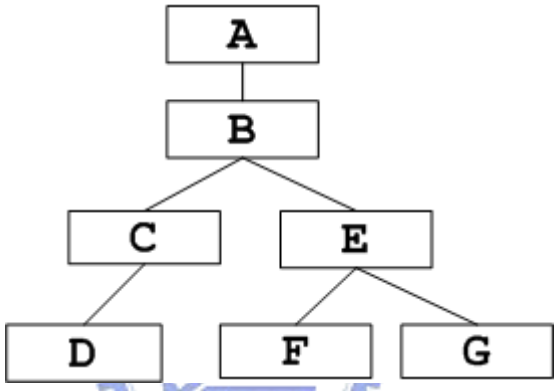


圖 2-12、樹狀結構棋譜範例 - 1

則如之前所述，會先輸出以 C 為首的 Subtree 後再輸出以 E 為首的 Subtree，由於 C 與 E 都是 B 的下一步，因此兩個 Subtree 都會以小括號括住作為標示。下方是這個例子的 SGF 檔案：

```
(;FF[4]
 ;B[aa]C[A];W[bb]C[B]
(;B[cc]C[C];W[dd]C[D])
(;B[ee]C[E](;W[ff]C[F])(;W[gg]C[G]))
```

另外，目前尚有 XML[14]版本的 XGF（XML Game Format）正在發展中，目前還在初稿階段，且只有針對圍棋設計格式。

2.3.2.2 GAML

GAML[3]是本實驗室根據桌上型遊戲模式 POT[2]所設計的 XML 標記語言。它利用 POT 模式中，以遊戲物件狀態的集合來描述遊戲狀態的作法，來記錄遊戲的初始狀態，之後透過記錄使用者對遊戲物件的操作，來記錄遊戲狀態變化的過程。

如圖 2-13 是使用者擺設的象棋初始棋盤，左方則是 GAML 的紀錄。在記錄中，我們可以看到 GAML 利用<PIECE>記錄每一個在盤面上的棋子代碼和位置來記錄初始遊戲的狀態：

```
<INIT>
  <BOARD>
    <PIECE ID="7" POS="B7"/>
    <PIECE ID="14" POS="D1"/>
    <PIECE ID="7" POS="D8"/>
    <PIECE ID="4" POS="E0"/>
    <PIECE ID="3" POS="E2"/>
    <PIECE ID="7" POS="E7"/>
    <PIECE ID="8" POS="E9"/>
    <PIECE ID="1" POS="F0"/>
    <PIECE ID="14" POS="F2"/>
    <PIECE ID="7" POS="F8"/>
    <PIECE ID="11" POS="G9"/>
    <PIECE ID="13" POS="H7"/>
    <PIECE ID="11" POS="H9"/>
    <PIECE ID="14" POS="I6"/>
  </BOARD>
</INIT>
```

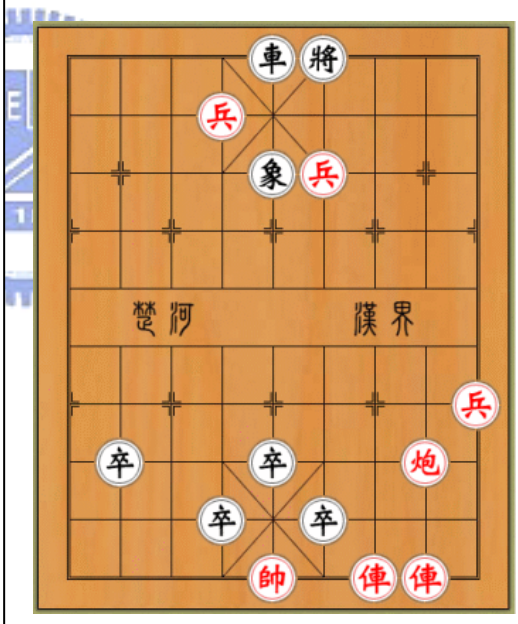


圖 2-13、遊戲初始狀態範例

接下來，以記錄玩家對遊戲物件操作(Operation)的方式來記錄遊戲狀態變化的過程。以上例而言，假使遊戲開始後，紅方玩家將炮向左移動兩步（如圖 2-14）：

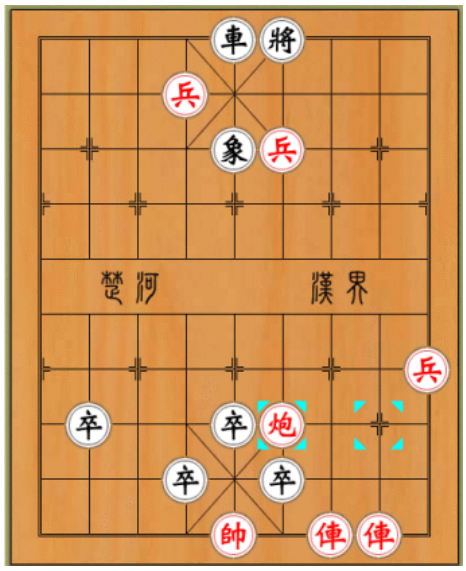


圖 2-14、遊戲物件操作範例

則在 GAML 中的紀錄如下：

```
<PLY>
    <MOVE CMD="MOVE" ID="13" SRC="H7" DES="F7"/>
</PLY>
```

在 GAML 中，是以<PLY>作為 Operation 記錄的單位。<PLY>中包含玩家在這一手中對於遊戲物件的所有操作，例如：

```
<PLY>
    <MOVE CMD="REMOVE" ID="5" SRC="H0">
    <MOVE CMD="MOVE" ID="13" SRC="H7" DES="H0"/>
    <COMMENT>紅炮吃黑馬</COMMENT>
</PLY>
```

便是記錄象棋中，紅方的炮吃去黑方的馬的一手。

另外，在 GAML 中，有主要棋步與分支棋步的區別。分支的棋步會以<BRANCH>標籤包住，與主要棋步為兄弟的關係，並且會出現在主要棋步之前。如以圖 2-15 為例：

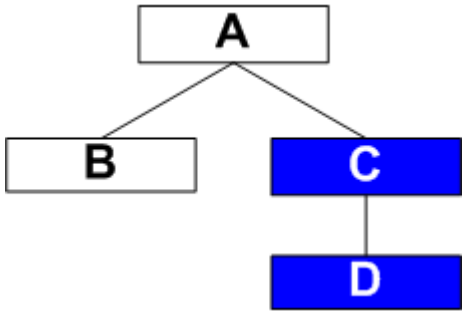


圖 2-15、樹狀結構棋譜範例 - 2

在此例中，B 與 C 同為 A 的下一手，但 B 為主要棋步，C -> D 為分支。GAML 以<BRANCH>包住 C 與 D 兩步表示分支：

```
<PLY>
     <MOVE CMD="PLACE" ID="0" DES="A0"/>
     <COMMENT> A </COMMENT>
</PLY>
<BRANCH>
     <PLY>
         <MOVE CMD="PLACE" ID="1" DES="C0"/>
          <COMMENT> C </COMMENT>
     </PLY>
     <PLY>
         <MOVE CMD="PLACE" ID="0" DES="D0"/>
         <COMMENT> D </COMMENT>
     </PLY>
</BRANCH>
<PLY>
     <MOVE CMD="PLACE" ID="1" DES="B0"/>
     <COMMENT> B </COMMENT>
</PLY>
```

另外，在 GAML 中也支援紀錄標記以及遊戲相關資訊的標籤。

2.3.2.3 綜合比較

我們已經瞭解 SGF[12]與 GAML[3]這兩種不同的格式，都能夠用來記錄數種遊戲的棋譜，我們在這裡做一個簡單的整理：

	GAML	SGF
支援的遊戲種類	所有 POT Games	兩人對弈的遊戲
記錄格式	一連串的標籤所組成的 XML tree	一連串的 Node 所組成的樹狀結構
可記錄的資訊	棋步、分支、標記、註解、遊戲資訊等	棋步、分支、標記、註解、遊戲資訊等
棋譜檔案	標籤名稱為完整單字，可直接閱讀，棋譜架構一目了然，檔案稍大	屬性名稱精簡，直接閱讀較不容易，檔案小

表 2-1 、GAML 與 SGF 的比較表

最後讓我們分別使用 SGF 與 GAML 這兩種存檔格式，來記錄圖 2-12 的樹狀棋譜。

使用 SGF 格式的棋譜檔案如下：

```
(;FF[4]
 ;B[aa]C[A];W[bb]C[B]
(;B[cc]C[C];W[dd]C[D])
(;B[ee]C[E](;W[ff]C[F])(;W[gg]C[G]))
```

使用 GAML 格式的棋譜檔案如下：

```
<GAML>
   <PLY> <MOVE CMD="PLACE" ID="0" DES="A0"/>
         <COMMENT> A </COMMENT>
   </PLY>
   <PLY> <MOVE CMD="PLACE" ID="1" DES="B1"/>
         <COMMENT> B </COMMENT>
   </PLY>
   <BRANCH>
      <PLY> <MOVE CMD="PLACE" ID="0" DES="E4"/>
            <COMMENT> E </COMMENT>
      </PLY>
      <BRANCH>
            <PLY> <MOVE CMD="PLACE" ID="0" DES="G6"/>
                  <COMMENT> G </COMMENT>
            </PLY>
      </BRANCH>
      <PLY> <MOVE CMD="PLACE" ID="0" DES="F5"/>
            <COMMENT> F </COMMENT>
      </PLY>
   </BRANCH>
   <PLY> <MOVE CMD="PLACE" ID="0" DES="C2"/>
         <COMMENT> C </COMMENT>
   </PLY>
   <PLY> <MOVE CMD="PLACE" ID="1" DES="D3"/>
         <COMMENT> D </COMMENT>
   </PLY>
</GAML>
```

第三章　通用棋譜編輯系統

在這一章中，主要介紹我們通用棋譜編輯系統開發平台的設計部分。首先在 3.1 節中介紹的是系統架構，講述我們如何處理不同格式的棋譜檔案，接下來在 3.2 節講述系統模組，我們如何切割系統的各個部分，以及各部分的功能。為了說明方便，以下敘述中的「平台」即為通用棋譜編輯系統開發平台的簡稱。

3.1 系統架構

整個通用棋譜編輯系統開發平台是建構在 J2SE[15]之上（如圖 3-1），提供完整棋譜編輯系統相關功能 API 來供各個遊戲使用。凡符合我們內部所遵循的通用遊戲模式的遊戲，都能夠透過我們的平台來開發棋譜編輯系統。遊戲設計者必須負責遊戲相關部分的設定和實作（即 Game-specific 的部分），有關這部分的內容，將在下一章中以實例說明。

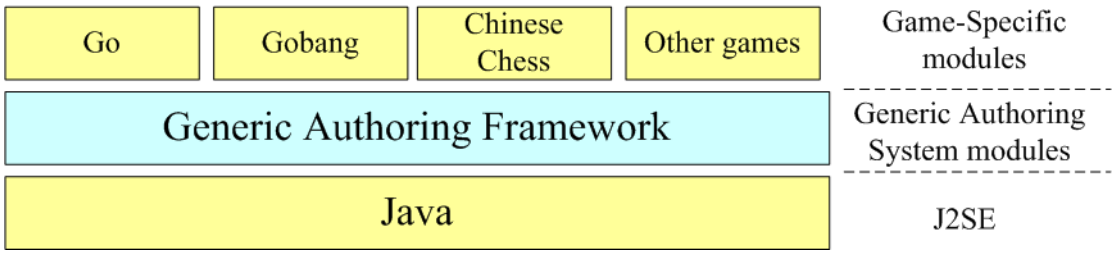


圖 3-1、遊戲的階層架構圖

通用棋譜編輯系統開發平台的系統架構如圖 3-2：

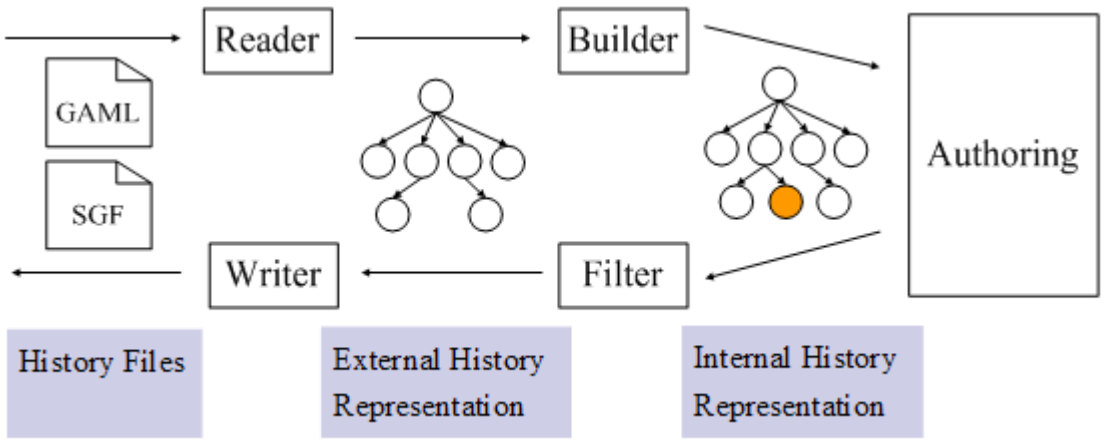


圖 3-2、通用棋譜編輯系統開發平台的系統架構圖

各種格式的棋譜檔案透過 Reader 讀入並轉換成 GAML[3]架構的 DOM[16] Tree，之後再透過 Builder 機制使得這棵 DOM Tree 的資訊能夠完整對應於內部的遊戲模式，最後進入編輯階段。在編輯階段中，使用者可以對這些資料作任意的修改與瀏覽。輸出時，先透過 Filter 將不必要寫在棋譜中的資訊濾除，最後透過 Writer 將棋譜資料依指定的格式輸出。以下針對各個部分做詳細的介紹：

3.1.1 Reader 和 Writer

在系統中，Reader 負責棋譜檔案的讀取與分析，不同格式的棋譜檔案使用不同的 Reader 來處理。系統目前支援 GAML 與 SGF[12]這兩種格式的 Reader，日後如需支援其他種棋譜格式，需要自行繼承 Reader 類別（如圖 3-3）。

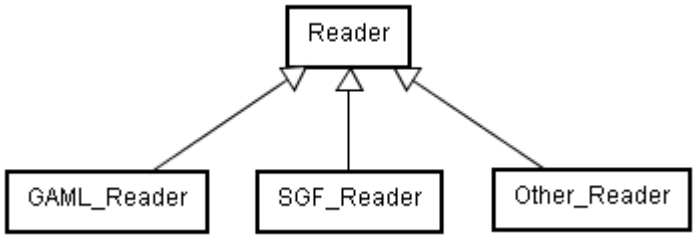


圖 3-3、Reader

不同的格式的棋譜檔案在透過 Reader 之後，都會轉換成相同 GAML 架構的 DOM Tree，我們稱之為外部紀錄表述（External History Representation）。

與 Reader 功能相對應的 Writer 負責將 DOM Tree 輸出成為指定的棋譜格式檔案，目前系統亦支援有 GAML 與 SGF 這兩種格式的 Writer，同樣地，日後如需支援其他種棋譜格式，需要自行繼承 Writer 類別（如圖 3-4）。

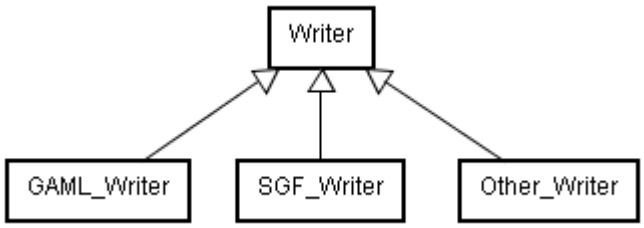


圖 3-4、Writer

3.1.2 外部紀錄表述

如上一小節所述，外部紀錄表述為 GAML 架構的 DOM Tree，完全對應於原始的棋譜檔案內容。一般而言，在將檔案讀入使其成為內部資料結構後，理應可以直接使用這些資料，不需額外的處理。但是，因為棋譜中記錄的資訊，並不完全遵循我們平台內部的遊戲模式架構，因此我們必須再透過下一小節中將會詳述的 Builder 機制將資料再做處理。最明顯的例子是棋類遊戲中吃子的部分，例如圖 3-5 中兵向前一步將卒吃掉的一手：

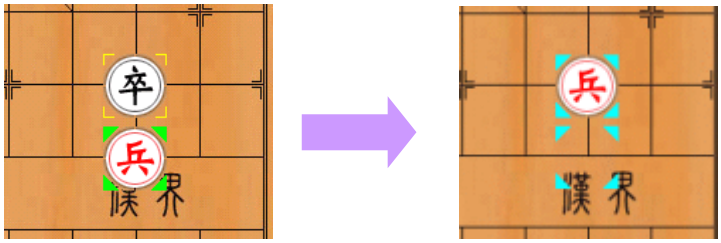


圖 3-5、象棋中「兵吃卒」示意圖

在棋譜當中，對於這一手只會記錄「兵前進一步」這個動作，但在我們的內部 SPOT 模式中，這一手包含兩個動作（見圖 3-6），一是「卒」從棋盤上移開，另一動作是「兵」在棋盤上前進一步：

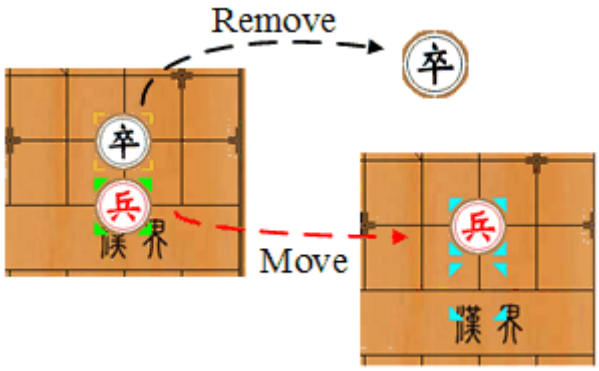


圖 3-6、SPOT 模式中「兵吃卒」示意圖

在一般的棋譜當中，並不需要記錄吃子的部分，因為這部分的資訊可以透過棋規判斷得知；然而如果不將吃子的動作記錄在內部資料中，那麼內部平台在處理的時候，就會依據我們所遵循的遊戲模式而認為那個棋子仍然在棋盤上，而導致嚴重的錯誤。

3.1.3 Builder 和 Filter

如上節所述，我們需要 Builder 與 Filter 的機制來處理棋譜記錄與內部記錄的差異：透過 Builder 將棋譜中遺失的部分，重新建立回來；透過 Filter 將不需要記錄在棋譜中的部分濾除。很顯然的，平台本身並沒有辦法為所有遊戲決定有哪些資訊是屬於棋譜中不需要記錄但是內部資料需要的部分，因此，Builder 和 Filter 的機制必須交由各個遊戲自行處理。有關於這部分的實作方式，請參見下一章中的實例。

3.1.4 內部資料表述

外部資料表述透過 Builder 重新建立出完整記錄後，這一份完整

對應於平台內部遊戲模式的資料，仍然是一棵 GAML 架構的 DOM Tree，我們稱之為「內部資料表述（Internal history representation）」。內部資料表述與外部資料表述之間的差異，便在於 Builder 重新建立出來的資訊。

在平台內部除了有內部資料表述記錄完整棋譜資訊外，我們還需要一個虛擬棋盤來記錄目前盤面的狀態。虛擬棋盤以二維陣列來記錄盤面上的棋子和位置，由於各個遊戲所使用的棋盤不同，棋子的種類和個數也不一樣，因此當遊戲透過平台來開發自己的棋譜編輯系統時，平台會要求遊戲提供這方面的資訊，詳細的方式我們會在系統實作部分說明。

3.2 系統模組

瞭解了系統架構之後，接下來看到的是系統模組的介紹。我們將整個平台切割成兩個部分來設計：核心資料模組（Core Module）與使用者介面模組（GUI Module）。核心資料模組負責棋譜資料的處理，包含所有資料的編輯以及內部資料狀態的記錄等部分；使用者介面模組負責處理介面設計、介面操作與畫面繪圖等相關部分。

在實際操作棋譜編輯系統時，使用者透過使用者介面模組對資料進行操作，在透過使用者介面模組發出改變資料的指令後，使用者介面模組會將使用者的指令傳達至核心資料模組，由核心資料模組執行指令並通知使用者介面模組更新資料的顯示。以下讓我們更詳細的瞭解核心資料模組與使用者介面模組各自負責的部分。

3.2.1 核心資料模組 Core Module

核心資料模組主要分成六個類別：Editor、History、Element、Board、Reader 和 Writer。其中 Reader 和 Writer 對應於上一節系統架

構中的 Reader 和 Writer，分別負責棋譜的讀入與輸出。以下就其他類別分別作介紹：

- Editor：

  負責管理整個核心資料模組。所有使用者透過使用者介面模組所發出的命令（如編輯或瀏覽相關功能等），都會透過 Editor，並分派給特定的類別處理。

- History：

  負責處理整個棋譜記錄，即對應於上節中所介紹的內部資料表述。所有的編輯功能，都是交由 History 來完成。另外，在 History 中亦有變數記錄目前所在的棋步位置，以便作畫面的更新與資料的對應。

- Element：

  Element 類別 對應於 GAML 中的標籤。每一個 Element 紀錄標籤名、屬性、屬性值和所包含的子標籤，同時也以指標鏈結前後 Element 及父 Element。History 類別內所記錄的就是以許多個 Element 物件所建立出來的樹狀結構，使用者對於棋譜記錄的編輯與瀏覽就是對這棵樹的操作。

- Board：

  負責處理上節中所介紹的虛擬棋盤，主要負責記錄目前盤面的狀態。當使用者在瀏覽棋步時，隨著前一步、後一步的移動，Board 必須負責回復盤面和執行這一手的動作，以便與外界的棋盤作完全的對應。

以下是簡單的 Class Diagram 說明核心資料模組中各類別之間的關係：

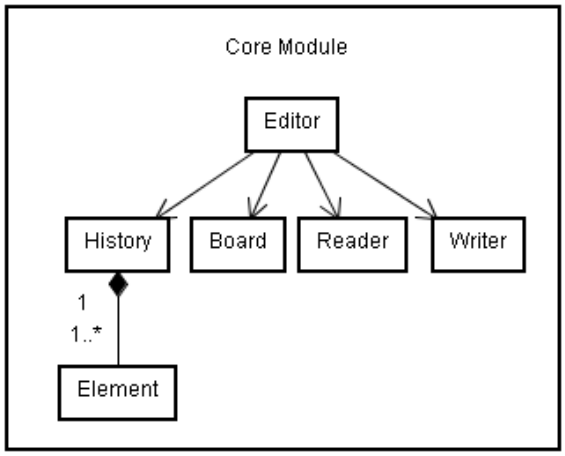


圖 3-7、Core Module Class Diagram

3.2.2 使用者介面模組 GUI Module

在介紹完核心資料模組之後，接下來要看到的就是有關使用者介面設計的部分。為了讓使用者能夠方便的編輯及瀏覽棋譜，我們也為使用者設計人性化的操作介面（如圖 3-8）：

圖 3-8、透過平台開發的象棋棋譜編輯系統畫面

我們將整個圖形使用者介面（GUI）分做四個部分來設計：

- Main Frame：

  Main Frame 掌管整個視窗控制，包含程式視窗、標題列、功能表、工具列及其他所有 Panel。所有使用者操作資料的動作，例如：編輯時移動棋子或瀏覽上一步等，都會透過 Main Frame 將命令傳達至內部資料模組以執行命令。

- Board Panel：

  Board Panel 掌管左側圖像化的虛擬棋盤，包含使用者對棋盤的操作，以及棋盤、棋子的繪製等相關繪圖功能。透過這個棋盤，使用者可以瞭解目前的盤面狀況，同時在編輯時，可以直接點選棋盤上的空格、空點或棋子來編輯棋步或標記等。之前我們曾經提到不同的遊戲使用的棋盤和棋子並不相同，平台在設計 Board Panel 時也遇到同樣的問題，甚至還包括使用不同的標記與一些讓使用者操作更方便的提示符號的繪製等等，我們將在系統實作那一章節討論我們如何解決這樣的問題。

- Comment Panel：

  Comment Panel 管理註解區的輸入及顯示。

- History Panel：

  在這裡以樹狀結構呈現棋譜資料，呈現的方式是參考「圍棋助手[4]」的作法，在棋譜中直接顯示分支，讓使用者可以自行點選棋譜中的棋步做跳躍式瀏覽。

圖 3-9 是簡單的 Class Diagram 說明使用者介面模組中各類別之間的關係：

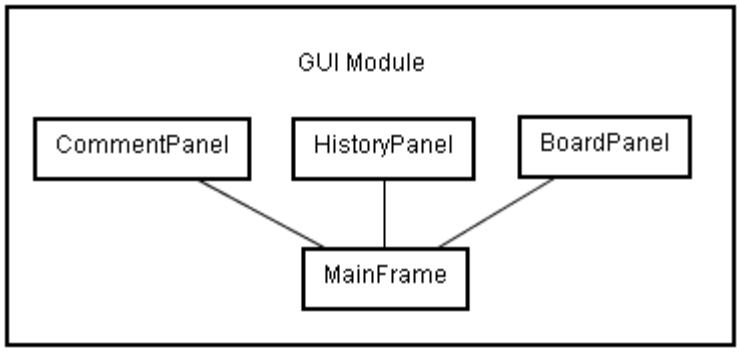


圖 3-9、GUI Module Class Diagram

3.2.3 Class Diagram

最後是整個平台的 Class Diagram，包含前兩小節的核心資料模組與使用者介面模組：

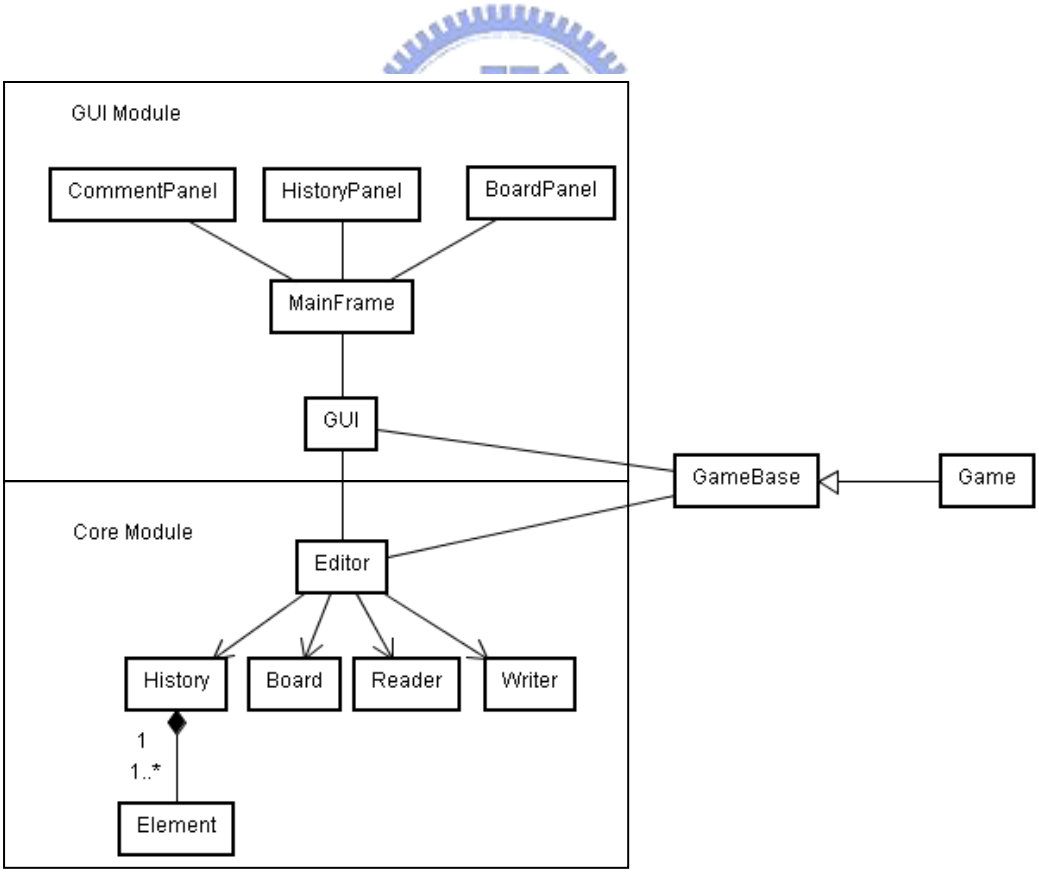


圖 3-10、通用棋譜編輯系統開發平台 Class Diagram

GUI 類別負責核心資料模組與使用者介面模組間的溝通，另外，我們將所有提供給遊戲使用、覆寫的函式，集中在 GameBase 這個類別裡。欲使用我們通用棋譜編輯系統開發平台來開發的遊戲，可以使用繼承 GameBase 類別的方式來實作。實作的方法及需要覆寫的函式我們將在下一章中仔細的介紹。

第四章　系統實作

在這一章我們以象棋和五子棋做為例子，介紹如何使用我們的通用棋譜編輯系統開發平台來實作出一套棋譜編輯系統。

承上一章文末中所提到，欲使用我們通用棋譜編輯系統開發平台來開發棋譜編輯系統的遊戲，必須繼承 GameBase 這個類別並覆寫部分函式，另外我們還使用一個 Property 檔來設定遊戲繪圖與需要動態調整的選項。使用 Property 檔而不將設定寫在程式碼中，保留了日後更換圖檔或是調整繪圖位置的彈性，同時也簡化程式碼的寫作。

接下來分別在 4.1 節和和 4.2 節中介紹核心資料與使用者介面部分的實作方式。4.3 節當中則介紹遊戲相關資訊與擺譜功能這兩部分的實作。

4.1 核心資料

在這一節當中，我們將實作的目標鎖定在遊戲資料上，包含平台如何處理不同遊戲的棋盤、棋子資料，還有之前所提到過的遊戲必須自行處理的 Builder 與 Filter 功能的實作。

◆ 棋盤：

由於同一種遊戲可能會使用不同大小的棋盤，例如：圍棋有 19*19、13*13 及 9*9 三種棋盤大小，因此我們將這部分的設定放在 Property 檔中，以便能夠動態調整如表 4-1：

Property	屬性說明	屬性值 （以象棋為例）
GAME_NAME	遊戲名稱	Cchess
EDITER_BOARD_WIDTH	棋盤寬	9
EDITER_BOARD_HEIGHT	棋盤高	10

表 4-1 、棋盤 Property 設定項目

◆ 棋子：

要求遊戲覆寫 getPiece_IDs() 與 getPiece_count()兩個函式來設定所使用的棋子代碼及相對應的棋子個數。以下是象棋在這部分的實作：

- getPiece_IDs()：

```
private static int piece_id[] = {
    // 黑方
    CchessPiece.BKING,      CchessPiece.BROYALIST,
    CchessPiece.BMINISTER,  CchessPiece.BCHARIOT,
    CchessPiece.BKNIGHT,    CchessPiece.BGUN,
    CchessPiece.BPAWN,
    // 紅方
    CchessPiece.RKING,      CchessPiece.RROYALIST,
    CchessPiece.RMINISTER,  CchessPiece.RCHARIOT,
     CchessPiece.RKNIGHT,   CchessPiece.RGUN,
    CchessPiece.RPAWN
};
public int[] getPiece_IDs(){
    return piece_id;
}
```

- getPiece_count()：

```
private static int piece_count[] =
{1,2,2,2,2,2,5,1,2,2,2,2,2,5};
public int[] getPiece_count(){
    return piece_count;
}
```

◆ Builder：

要求遊戲覆寫 buildFullPly()函式。平台在讀入資料後，將棋步資料透過此函式傳給遊戲，由遊戲建立完整棋步資訊後回傳平台。以下是象棋在這部分的實作：

```
public PlyNode buildFullPly(PlyNode originPly){
    PlyNode newPly = new PlyNode();
    for (int i = 0; i < originPly.getMoveCount(); i++){
        MoveNode move = originPly.getMoveNodeAt(i);
        newPly.addMoveNode(move);
        if (move.getMoveType() == MoveNode.MOVE){
            Point dest = move.getDestPosition();
            int tar_id = super.getPieceID(dest.x,dest.y);
            if (tar_id != Piece.EMPTY){
                MoveNode remove =
                       MoveNode.newRemovePieceMoveNode
                       (dest.x,dest.y,tar_id,true);
                newPly.addMoveNode(remove);
            }
        }
    }
    return newPly;
}
```

◆ Filter：

遊戲在透過平台提供的函式建構棋步時，可透過參數指定這個標籤要不要輸出，來達到過濾某些標籤的目的。例：

```
public static MoveNode newPlacePieceMoveNode (int destX,int destY,
                                              int pieceID,boolean hidden)
public static MoveNode newMovePieceMoveNode(int srcX,int srcY,int destX,
                                            int destY,int pieceID,boolean hidden)
public static MoveNode newRemovePieceMoveNode(int srcX,int srcY,
                                              int pieceID,boolean hidden)
```

4.2 使用者介面

在上一章當中我們已經介紹整個平台的圖形使用者介面主要分成 Main Frame、Board Panel、Comment Panel 和 History Panel 等部分，其中與遊戲最直接相關的就是管理虛擬圖像棋盤的 Board Panel 與顯示棋譜的 History Panel，因此接下來我們會針對這兩部分的實作詳細介紹。

4.2.1 Board Panel

之前我們提到 Board Panel 掌管左側圖像化的虛擬棋盤，包含使用者對棋盤的操作，以及棋盤、棋子的繪製等相關繪圖功能。在這裡我們將遊戲對這部分的實作方式分成三個部份：Board Customization、Drawing Customization 與 Operation Customization 分別在以下三個小節中作介紹。

4.2.1.1 Board Customization

這一章節的重點在於棋盤與棋子的繪製問題。不同的遊戲所使用的棋盤大小不同，棋盤格的大小與棋子的大小也各異。我們在平台內部是以虛擬座標(以棋盤格為單位)來記錄棋子位置，顯示時卻是使用盤面座標(以像素為單位)來為棋子定位，因此，如何將棋子的虛擬座標轉換為正確的盤面座標，是一個非常重要的課題。以下就以象棋為範例，讓我們來看看關於這部分要設定的 property 項目：

Property	屬性說明	屬性值 （以象棋為例）
EDITER_BOARD_PANEL_WIDTH	Board Panel 寬度	430
EDITER_BOARD_PANEL_HEIGHT	Board Panel 高度	550
EDITER_GRID_SIZE_X	棋盤格 X 軸長度	45
EDITER_GRID_SIZE_Y	棋盤格 Y 軸長度	51
EDITER_IMG_BOARD	棋盤圖檔	board.jpg
EDITER_IMG_PIECE	棋子圖檔	Pieces.gif
EDITER_CUT_PIECE	棋子圖檔中的 棋子個數	14
EDITER_BOARD_OFFSET_X	棋盤邊線到 Board Panel 邊線的距離	0
EDITER_BOARD_OFFSET_Y		0
EDITER_GRID_OFFSET_X	棋盤格到棋盤邊線的距離	10
EDITER_GRID_OFFSET_Y		11
EDITER_PIECE_OFFSET_X	左上角的棋子中心點到棋盤格線的距離	0
EDITER_PIECE_OFFSET_Y		0

表 4-2 、Board Customization Property 設定項目

在這部分的設定項目有關距離的設定是以像素做為單位。EDITER_BOARD_PANEL_WIDTH 和 EDITER_BOARD_PANEL_HEIGHT 設定整個 Board Panel 的大小，一般而言應該會比棋盤略大一些。EDITER_GRID_SIZE_X 和 EDITER_GRID_SIZE_Y 設定棋盤格大小，用來作為繪製棋子時，X 軸與 Y 軸的座標單位長。另外，透過 EDITER_IMG_BOARD 和 EDITER_IMG_PIECE 分別指定棋盤圖檔與棋子圖檔路徑，以這個例子而言，象棋將兩個圖都放在遊戲的根目錄中。

EDITER_CUT_PIECE 指的是棋子圖檔中的棋子個數，透過這個屬性值的設定，平台就能等比例切割出棋子圖案，例如象棋的棋子圖檔（圖 4-1），這是由 14 個棋子所組成的檔案，因此 EDITER_CUT_PIECE = 14。

圖 4-1、象棋的棋子圖檔範例

最後六個屬性是調整棋子擺放在棋盤上的位置。Board Offset 調整棋盤邊線到 Board Panel 邊線的距離；Grid Offset 調整棋盤格到棋盤邊線的距離；Piece Offset 調整左上角的棋子中心點到棋盤格線的距離，見圖 4-2：

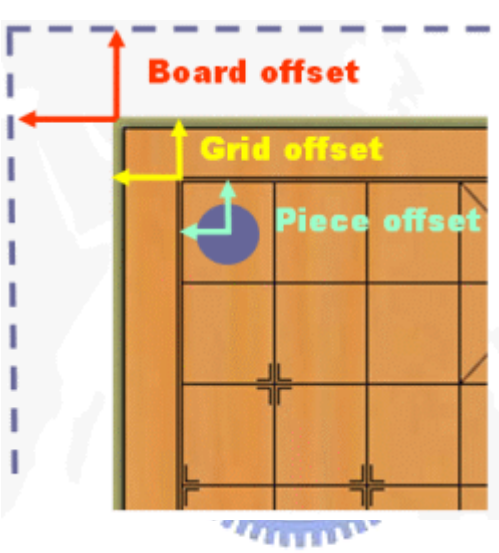


圖 4-2、Board Customization 設定說明

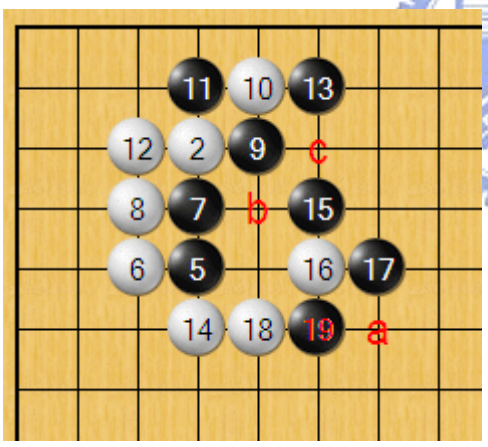
透過以上的 property 設定，平台就能夠將各個遊戲的棋子虛擬座標轉成正確的盤面座標。如以象棋為例，根據以上的屬性值設定，虛擬座標（1, 2）的棋子，在盤面上的 X 座標 = EDITER_BOARD_OFFSET_X + EDITER_GRID_OFFSET_X + EDITER_PIECE_OFFSET_X + 1 * EDITER_GRID_SIZE_X = 0+10+0+1*45 = 55；盤面上的 Y 座標 = EDITER_BOARD_OFFSET_Y + EDITER_GRID_OFFSET_Y+ EDITER_PIECE_OFFSET_Y + 2 * EDITER_GRID_SIZE_Y = 0+11+0+2*51 = 113。因此，原先的虛擬座標（1, 2）就能順利的轉換為棋盤上的位置（55, 113），而能將棋子正確的繪製在棋盤上。

4.2.1.2 Drawing Customization

棋盤上除了需要繪製棋盤和棋子之外，為了使用者操作和瀏覽的方便，也會在棋盤上繪製一些提示符號或是圖案來輔助說明。由於各遊戲對這部分的需求不一，在這裡我們同樣是使用 Property 檔來處理。我們將這類的繪圖功能，分成標記、下子順序及提示圖案等三個部分來討論。

4.2.1.2.1 標記

有時為了註解說明方便起見，我們會需要在棋盤格或棋子上用一個符號表示目前所說明的位置，一般我們將這類符號稱做「標記」。例如圖 4-3 中的 a b c 三個字母即為標記的一種。



註解：

吳清源[17]：
黑 19 是搶先手的下法。
此棋如要忍耐，則在 a 長，白 b，黑 c，白拿到先手。

圖 4-3、標記使用範例

由於不同的遊戲所需要的標記種類可能會不一樣，因此現在就讓我們來看看如何透過 property 的設定來解決這樣的問題：

Property	屬性說明	屬性值 （以五子棋為例）
EDITER_MARK_SYMBOL	標記符號 （以｜作間隔）	○|●|△|▲|□|■
EDITER_IMG_MARK	標記圖檔	marks.gif
EDITER_CUT_MARK	標記圖檔的標記個數	6
EDITER_MARK_MENU_TOTAL	功能表選單個數	3
EDITER_MARK_MENU_0	選單名稱	添加圓形標記
EDITER_MARK_MENU_1	選單名稱	添加三角形標記
EDITER_MARK_MENU_2	選單名稱	添加正方形標記
EDITER_MARK_BUTTON_0	工具列按鈕文字	○
EDITER_MARK_BUTTON_1	工具列按鈕文字	△
EDITER_MARK_BUTTON_2	工具列按鈕文字	□

表 4-3 、標記相關 Property 設定項目

對應於標記圖檔中的順序，在 EDITER_MARK_SYMBOL 中為每一個標記設定一個代表符號，用來在棋譜列表中表示標記。EDITER_IMG_MARK 設定標記圖檔的路徑，同時，與棋子圖檔的設計相同，需要透過 EDITER_CUT_MARK 設定內含的標記個數。另外，由於標記功能需要由功能表或工具列提供功能選單或按鈕方能使用，而各個遊戲的標記項目又不相同，因此在這裡使用 EDITER_MARK_MENU_TOTAL 指定標記項目個數，以及以 EDITER_MARK_MENU_x 和 EDITER_MARK_BUTTON_x（x 為從 0 開始的數字），依序指定在功能選單中的文字與工具列按鈕上的文字。以上例而言，呈現的結果如圖 4-4：

圖 4-4、標記 Property 設定結果範例

當使用者按下其中一個功能表選單時，其相對應的工具列按鈕也會呈現選取狀態，反之亦然，此時使用者在棋盤上可以點選欲新增標記的位置，當使用者點選時，平台會告知遊戲目前是第幾個功能表選單被點選，遊戲可以透過這個值來決定目前是添加哪一個標記。

4.2.1.2.2 下子順序

某些棋類遊戲如五子棋、圍棋等，需要在棋子上標明下子的順序（又稱手順），如圖 4-5 中黑白兩子上的數字：

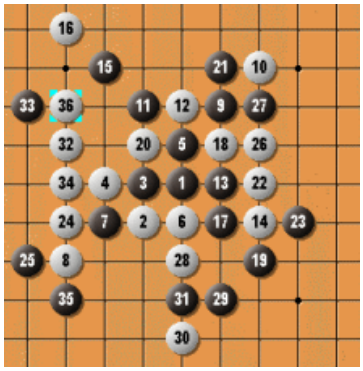


圖 4-5、手順使用範例

平台內部提供函式讓遊戲可以取得手順數字，但是手順數字的顯示顏色往往隨著棋子的顏色不同而跟著改變，同時顯示的位置和大小也會需要做調整，因此我們設計相關的 Property 如下：

Property	屬性說明	屬性值 （以五子棋為例）
EDITER_PIECE_PLY_COLOR	數字顏色	FFFFFF|000000
EDITER_PIECE_PLY_FONT_SIZE	數字大小	14
EDITER_PIECE_PLY_OFFSETX_0	數字為 0~9 時的顯示位置調整	10
EDITER_PIECE_PLY_OFFSETY_0		17
EDITER_PIECE_PLY_OFFSETX_1	數字為 10~99 時的顯示位置調整	7
EDITER_PIECE_PLY_OFFSETY_1		17
EDITER_PIECE_PLY_OFFSETX_2	數字為 100~999 時的顯示位置調整	4
EDITER_PIECE_PLY_OFFSETY_2		17

表 4-4 、手順相關 Property 設定項目

我們透過 EDITER_PIECE_PLY_COLOR 屬性來設定手順的數字顏色，使用十六進位數字依序指定 00~FF 的紅、綠、藍色彩值。顏色的順序對應於棋子的順序，同代碼的棋子會使用同樣的顏色。EDITER_PIECE_PLY_FONT_SIZE 設定顯示的數字大小，單位為 pt，其他屬性則用來調整數字顯示顯示的位置。

4.2.1.2.3 提示圖案

除了以上提到的部分，有的時候為了便利使用者的操作與瀏覽，會需要在盤面上繪製一些提示圖案，例如圖 4-6 中在「卒」的外層細框表示目前游標所在，而「兵」的外層圖案表示目前選取的子。

圖 4-6、提示圖案使用範例

對於這類的繪圖，我們使用下列 Property 設定：

Property	屬性說明	屬性值 （以象棋為例）
EDITER_IMG_HIGH_LIGHT	提示圖案圖檔	highLight.gif
EDITER_CUT_HIGH_LIGHT	圖檔中的圖案個數	3
EDITER_IMG_INDEX_CURSOR	游標移動圖案索引值	0
EDITER_IMG_INDEX_CLICK	點選棋子圖案索引值	1
EDITER_IMG_INDEX_SRC	標示棋子來源位置索引值	2
EDITER_IMG_INDEX_DES	標示棋子目的地位置索引值	2

表 4-5 、提示圖案相關 Property 設定項目

由於平台會使用到滑鼠游標移動、點選棋子以及標示棋子來源和目的地的提示圖案，因此要求遊戲必須提供這四種圖檔，並分別指定索引值。如果遊戲還需要繪製其他的提示符號，應該將所有會使用到的圖檔都放置在 EDITER_IMG_HIGH_LIGHT 中，平台中亦提供函式 setHighLight(int x,int y,int index)來讓遊戲繪製指定的提示圖案。

4.2.1.3 Operation Customization

Board Panel 除了以圖像化的方式呈現出棋盤和棋子外，為了更逼近真實的下棋情境，也提供可以滑鼠直接點選棋盤的方式來模擬真實的下子動作。但由於每種遊戲的玩法不同，同樣點選棋盤的動作所代表的意義會因遊戲而異。比如說：以象棋而言，會以點選棋子表示選取，再點選空點表示要將選取的棋子移動到此；而圍棋或五子棋則是以點選空點來表示放子。

因此，我們平台在這裡的作法是將使用者所點選棋盤位置和所使用的滑鼠鍵傳達給遊戲，由遊戲來決定使用者要做的動作以及一手當中會包含多少動作。為了要達到這樣的要求，遊戲必須覆寫 mouseClick() 函式，以下是象棋的例子：

```
public void mouseClick(int x,int y, int mouse_key){
    if (mouse_key == Game.MOUSE_KEY_RIGHT){
            src = null; //右鍵取消
            return;
    }
    if (super.getMode() == Game.PlyMode)
        ply(x,y);
    else if (super.getMode() == Game.MarkMode)
        mark(x,y);
}

public void ply(int x, int y){
    Piece tar = getPiece(x,y);
    if (tar == null){
        tar = new Piece(Piece.EMPTY,x,y);
    }
    if ((src == null)){ //如果沒有點選過
        if (tar.getID() != Piece.EMPTY)
            src = tar;
        return;
    }
    else if (tar.isEqual(src)){
        src = null; //又點選同一子，表示 deselect
        return;
    }
    if (checkLegalPly(src.x,src.y,tar.x,tar.y)){
        PlyNode ply = buildFullPly(src.x,src.y,tar.x,tar.y);
        super.addPly(ply,PlyNode.BRANCH);
    }
    src = null;
    clearHighLight();
}
```

從程式中我們不難發現，遊戲必須自己紀錄使用者所點選的棋子或位置，並將這些資訊自行包裝成一個棋步之後，傳給平台，達成新增棋步的功能。

4.2.2 History Panel

在 History Panel 我們以樹狀結構呈現棋譜資料，由於我們是以 GAML 架構記錄棋譜，而 GAML 又是通用棋譜格式，因此遊戲不需要自己負責棋譜記錄的處理。但是在棋譜列表中，遊戲對於棋步的描述方式會有不同。例如圖 4-7 是象棋與五子棋的例子：

象棋　　五子棋

圖 4-7、象棋與五子棋棋譜列表說明文字顯示範例

這部分的實作方式，一樣是讓遊戲覆寫函式，以下是五子棋在這部分的實作：

```
public String getStepString(PlyNode ply,int plyNo){
    MoveNode place = ply.getMoveNodeAt(0);
    for (int i = 0; i < ply.getMoveCount(); i++){
        place = ply.getMoveNodeAt(i);
        if (place.getMoveType() == MoveNode.PLACE)
            break;
    }
  if (place.getPieceID() == BLACK)
    return "●  " + POS.charAt(place.getDestPosition_X())+" "
               + place.getDestPosition_Y();
    return "○  "+POS.charAt(place.getDestPosition_X())+" "
           + place.getDestPosition_Y();
}
```

4.3 其他功能實作

我們除了設計棋盤，讓使用者可以透過棋盤的操作，來編輯棋步、標記外，我們還需要額外的介面讓使用者可以輸入遊戲的相關資訊，和擺設初始棋盤。因此，我們在接下來的兩小節中，介紹這兩部分的實作方式。

4.3.1 遊戲相關資訊

遊戲除了要記錄棋譜之外，有的時候需要紀錄其他相關資訊，尤其是實戰譜會需要記錄對弈棋手姓名、比賽時間、使用規則等等。在我們平台的設計上，由於每個遊戲所需要記錄的相關資訊不同，如圍棋需要記錄貼目，象棋需要記錄讓子項目，如：讓馬等，因此在這裡我們讓遊戲自己去設計整個遊戲相關資訊設定對話框，再傳給平台顯示。如圖 4-8 是象棋在這部分設計的一個範例：

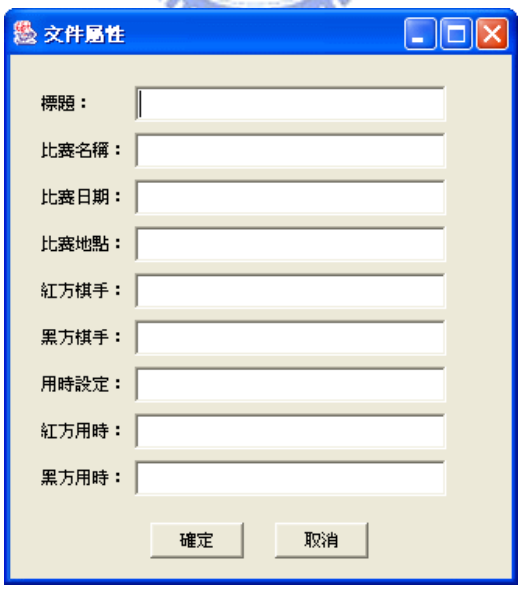


圖 4-8、遊戲資訊輸入對話框範例

4.3.2 擺譜

棋譜記錄在大部分的情況中，都是從一樣的初始盤面開始一直記錄到最後一手，但是有些時候，例如一些教學譜，會希望能夠自由擺設初始棋盤的狀態，而不使用預設的棋盤。因此，在不少的棋譜編輯系統中，也會提供讓使用者自己擺設初始棋盤的功能，也就是我們所說的「擺譜」。

擺譜功能不外乎棋子在棋盤與棋罐間的移動與棋子在棋盤上的移動，因此，擺譜功能依然遵循著我們的遊戲模組理論。不過，對於每個遊戲而言，並不是每個棋子都可以任意放置在棋盤上任意的位置，比如說：在象棋中，象不能過河或是在圍棋中有禁著點的限制，因此，遊戲有必要做擺譜完畢的檢查。在這裡我們的作法是由遊戲覆寫 checkLegalBoard()函式，在擺譜完成後由系統呼叫此函式，交由遊戲來作盤面的檢查，並回傳檢查的結果。如果是不合法的棋子擺設，則顯示警告視窗，並讓使用者重新調整棋子位置；如果是合法的擺設，那麼目前的棋子位置就會是此局初始的棋盤狀態。

4.4 實作流程

最後，讓我們將整個遊戲需要實作的部分做一個整理。欲使用我們的平台來開發棋譜編輯系統的遊戲必須要負責的部分如下：

1. 繼承 GameBase 類別。
2. 覆寫必要的函式如下：

getPiece_IDs() （詳見 4.1 節）
getPiece_count() （詳見 4.1 節）
buildFullPly() （詳見 4.1 節）

mouseClick ( )（詳見 4.2.1.3 節）
getStepString ( )（詳見 4.2.2 節）
checkLegalBoard()（詳見 4.3.2 節）
getSettingPanel()（詳見 4.3.1 節）

3. 設定 Property 項目，包括：

棋盤資料相關項目設定（詳見 4.1 節）
棋盤繪製相關項目設定（詳見 4.2.1.1 節）
標記相關項目設定（詳見 4.2.1.2.1 節）
手順相關項目設定（詳見 4.2.1.2.2 節）
提示圖案相關項目設定（詳見 4.2.1.2.3 節）

4. 準備相關圖檔，包括：
   棋盤、棋子、標記與提示圖案圖檔。

在完成以上的步驟之後，就可以看到一個專屬於這個遊戲並具備有編輯遊戲的初始棋盤、棋步、標記、分支、註解及棋局相關資訊等功能的棋譜編輯系統了。

第五章　結論與未來展望

本篇論文研究市面上的棋譜編輯系統，並根據本實驗室所提出的 POT 通用遊戲模組理論與 GAML 通用棋譜格式，開發出通用棋譜編輯系統開發平台。

目前我們的通用棋譜編輯系統開發平台，能夠支援編輯遊戲的初始棋盤、棋步、標記、分支、註解及棋局相關資訊等功能。在瀏覽方面，我們提供完整的棋譜列表，讓使用者自由點選欲觀看的棋步。另外在工具列中也提供按鈕，讓使用者可以跳至第一步、上一分支點、上一步、下一步、下一分支點及最後一步。

透過我們的平台所開發的棋譜編輯系統，能夠享有快速開發、程式碼更少、維護更容易的優點。同時透過平台提供的機制，能讓遊戲保有本身的特性，與附加自己獨特的功能。

在未來的發展上，可以考慮支援更多種類的棋譜格式，或是擴充如棋盤的翻轉，棋盤棋子圖檔的替換等圖形使用者介面功能。另外，可以設計更多的附加機制或是遊戲自由設定項目，讓遊戲在開發時能夠保有更多的彈性與擴充空間，同時又能享受到透過平台開發的好處。

參考文獻

[1] Brian Burns, The Encyclopedia of Games, MetroBooks, October 2000.

[2] 徐健智, “A General Development Platform for Play-on-table Game Over Internet”, 交通大學資訊工程系, 碩士論文, June 1999.

[3] I-Chen Wu, “Platform for Artificial Intelligence Game Tournament”, invited talk at International Computer Symposium, December 2002, Hualien, Taiwan, R.O.C.

[4] 胡小奇, “圍棋助手”, available from http://www.go-assistant.com/english/, 2004.

[5] 黃暉, “WinMGT”, available from http://waterfire.us/go/winmgt.htm, 2003.

[6] 天元科技, “Go2000”, available from http://sanppy.bizland.com/.

[7] “MultiGo”, available from http://www.ruijiang.com/multigo/chs/, 2004.

[8] Frank Arkbo, Uppsala , “Renju Library program”, available from http://www.renju.nu/renlib/, 2004.

[9] “GoRenjer”, available from http://www.gonnta.ne.jp/~itty/GoRenjer-en.html, 2002.

[10] 董世偉, “象棋演播室”, available from http://www.qipaile.net/xqstudio/index.htm, 2002.

[11] “象棋橋”, available from http://www.ccbridge.net/, 2001.

[12] Arno Hollosi, “Smart Game Format”, available from http://www.red-bean.com/sgf/, 2001.

[13] Judea Pearl, Heuristics/Intelligent Search Strategies for Computer Problem Solving, Addison-Wesley, 1984.

[14] W3C, “Extensible Markup Language”, available from http://www.w3c.org/XML/.

[15] Sun Microsystems, “Java 2 SDK”, available from http://java.sun.com/j2se/1.4.2/index.jsp, 2003.

[16] W3C, “Document Object Model”, available from http://www.w3.org/DOM/.

[17] 吳清源, 吳仁譯, 吳清源名局細解(4), 台灣世界文物出版社, 1986.

[18] Martin Fowler, Kendall Scott, UML distilled :a brief guide to the standard object modeling language, 2nd Edition, Addison-Wesley, December 2001.

[19] Robert Eckstein, Marc Loy, Dave Wood, Java Swing, 2nd Edition, O’REILLY, February 2003.

[20] David Flanagan, Java Examples In A Nutshell, 2nd Edition, O’REILLY, May 2002.

附錄一

以下是使用我們的通用棋譜編輯系統開發平台，來開發五子棋棋譜編輯系統的完整程式碼和 Property 檔案內容。

◆ 程式部分：

```
package Five;
import javax.swing.*;
import Gamlib.lib.*;
import Gamlib.GameBase;
import java.util.*;

public class Five extends GameBase{
    public static final int BLACK = 0;
    public static final int WHITE = 1;
    private static int[] piece_id = {BLACK,WHITE};
    private static int[] piece_count = {300,300};
    Vector nextPlys;

    public void init(){
        nextPlys = null;
    }
    //設定棋子代碼
    public int[] getPiece_IDs(){
        return piece_id;
    }
    //設定棋子數量
    public int[] getPiece_count(){
        return piece_count;
    }
    //設定遊戲資訊輸入對話框
    public JPanel getSettingPanel(){
        return new FiveSettingPanel(this);
    }
```

```
//檢查擺譜後的棋盤
public boolean checkLegalBoard() {
    return true;
}

//repaint 時需要更新的資料寫在這裡
public void refresh(){
    clearHighLight();
    nextPlys = super.getNextPlys();

    for (int i = 0; i < nextPlys.size(); i++){
        PlyNode ply = (PlyNode)nextPlys.elementAt(i);
        MoveNode move = ply.getMoveNodeAt(0);
        for (int j = 0; j < ply.getMoveCount(); j++){
            move = ply.getMoveNodeAt(j);
            if (move.getMoveType() == MoveNode.PLACE)
                break;
        }
        int x = move.getDestPosition_X();
        int y = move.getDestPosition_Y();
        super.setHighLight(x,y,1);
    }
}
//設定棋譜列表中的棋步文字
public String getStepString(PlyNode ply){
    MoveNode place = ply.getMoveNodeAt(0);
    for (int i = 0; i < ply.getMoveCount(); i++){
        place = ply.getMoveNodeAt(i);
        if (place.getMoveType() == MoveNode.PLACE)
            break;
    }
    if (place.getPieceID() == BLACK)
    return "●  " + POS.charAt(place.getDestPosition_X())+" "
           + place.getDestPosition_Y();
    return "○  "+POS.charAt(place.getDestPosition_X())+" "
           + place.getDestPosition_Y();
}
```

```
public void mouseClick(int x,int y, int mouse_key) {
        if (mouse_key == Game.MOUSE_KEY_RIGHT){
            return;   //使用滑鼠右鍵無作用
        }

        //瀏覽模式
        if ((getMode() == Game.ViewMode)&& (nextPlys != null)){
            for (int i = 0; i < nextPlys.size(); i++){
                PlyNode ply = (PlyNode)nextPlys.elementAt(i);
                MoveNode move = ply.getMoveNodeAt(0);
                for (int j = 0; j < ply.getMoveCount(); j++){
                    move = ply.getMoveNodeAt(j);
                    if (move.getMoveType() == MoveNode.PLACE)
                        break;
                }
                if ((x == move.getDestPosition_X())
                     && (y == move.getDestPosition_Y()))
                    super.gotoPlyNode(ply);
            }
        }

        //下子模式
        else if (super.getMode() == Game.PlyMode){
            //已經有子的位置不能點選
            if (getPieceID(x,y) != Piece.EMPTY)
                return;
            PlyNode ply = new PlyNode();
            if ((getCurrentPlyNo()%2) == 0){
                ply.addMoveNode(MoveNode.newPlacePieceMoveNode
                    (x,y,BLACK,false));
            }
            else{
                ply.addMoveNode(MoveNode.newPlacePieceMoveNode
                    (x,y,WHITE,false));
            }
            this.addPly(ply,PlyNode.LEGAL);
        }
```

```
        //標記模式
        else if (super.getMode() == Game.MarkMode){
            int tar_id = getPieceID(x,y);
            int index = getCurrentMarkMenuIndex();
            if (tar_id == BLACK)
                index = 2*index;
            else
                index = 2*index + 1;
            super.addMark(x,y,index);
        }
    }
}
    //重建棋步
    public PlyNode buildFullPly(PlyNode originPly) {
        return originPly;
}
```

◆ Property 部分：

```
#遊戲名稱
GAME_NAME = Five

#棋盤大小
EDITER_BOARD_WIDTH = 15
EDITER_BOARD_HEIGHT = 15

#Board Panel 大小
EDITER_BOARD_PANEL_WIDTH = 500
EDITER_BOARD_PANEL_HEIGHT = 530

#棋子位置調整與棋盤格大小調整
EDITER_BOARD_OFFSET_X = 20
EDITER_BOARD_OFFSET_Y = 20
EDITER_PIECE_OFFSET_X = 0
EDITER_PIECE_OFFSET_Y = 0
EDITER_GRID_OFFSET_X = 8
EDITER_GRID_OFFSET_Y = 7
EDITER_GRID_SIZE_X = 30
EDITER_GRID_SIZE_Y = 30
```

```
#棋盤與棋子圖檔
EDITER_IMG_BOARD = table.gif
EDITER_IMG_PIECE = pieces.gif
EDITER_CUT_PIECE = 2

#標記相關設定
EDITER_MARK_SYMBOL = ○|●|△|▲|□|■
EDITER_IMG_MARK = marks.gif
EDITER_CUT_MARK = 6
EDITER_MARK_MENU_TOTAL = 3
EDITER_MARK_MENU_0 = 添加圓形標記
EDITER_MARK_MENU_1 = 添加三角形標記
EDITER_MARK_MENU_2 = 添加正方形標記
EDITER_MARK_BUTTON_0 = ○
EDITER_MARK_BUTTON_1 = △
EDITER_MARK_BUTTON_2 = □

#手順使用的顏色
EDITER_PIECE_PLY_COLOR = FFFFFF|000000

#手順數字的大小
EDITER_PIECE_PLY_FONT_SIZE = 14

#手順數字的位置
EDITER_PIECE_PLY_OFFSETX_0 = 10
EDITER_PIECE_PLY_OFFSETY_0 = 17
EDITER_PIECE_PLY_OFFSETX_1 = 7
EDITER_PIECE_PLY_OFFSETY_1 = 17
EDITER_PIECE_PLY_OFFSETX_2 = 4
EDITER_PIECE_PLY_OFFSETY_2 = 17

#提示圖案相關設定
EDITER_IMG_HIGH_LIGHT = highLight.gif
EDITER_CUT_HIGH_LIGHT = 3
EDITER_IMG_INDEX_CURSOR = 0
EDITER_IMG_INDEX_CLICK = 1
EDITER_IMG_INDEX_SRC = 2
EDITER_IMG_INDEX_DES = 2
```