國立交通大學

資訊科學與工程研究所

碩 士 論 文

用於電腦對局遊戲桌機網格的通用遊戲發展之軟體框架

Software Framework for Generic Game Development in CGDG

研究生：劉浩雲

指導教授: 吳毅成 教授

中 華 民 國　一百零一　年 八 月

用於電腦對局遊戲桌機網格的通用遊戲發展之軟體框架

Software Framework for Generic Game Development in CGDG

研 究 生：劉浩雲　　Student：Hao-Yun Liu

指導教授：吳毅成　　Advisor：I-Chen Wu

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

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

Computer Science

August 2012

Hsinchu, Taiwan, Republic of China

中華民國一百零一年八月

用於電腦對局遊戲桌機網格的通用遊戲發展之軟體框架

研究生：劉浩雲　　指導教授：吳毅成

國立交通大學資訊科學與工程研究所

摘要

本篇論文審視了本實驗室過去研發之棋譜編輯器與電腦對局遊戲桌機網格系統，重新設計出一套軟體框架，能提供電腦對局應用的發展者，快速開發用以檢視人工智慧搜尋紀錄的棋譜編輯器、簡單地開發運用電腦對局遊戲桌機網格的演算法、快速地電腦對局遊戲桌機網格的工作端移植到各類作業系統。這個框架能夠使電腦對局應用問題的發展更為容易。

Software Framework for Generic Game Development in CGDG

Student: Hao-Yun Liu Advisor：Yi-Cheng Wu

Institute of Computer Science and Information Engineering

National Chiao Tung University

Abstract

In this thesis, we review the tools and environment developed in our lab, such as the game record editor and computer game desktop grid, and propose an improved software framework for facilitating game development. The software framework supports convenient game record editor development for computer game AI analysis, job-level search algorithm development, and CGDG worker OS portability.

誌謝

首先我要先感謝指導教授吳毅成老師這兩年的殷殷教誨，指引我走向正確的研究方向，並在我遇到困惑時，不厭其煩的與我共同討論。

同時也必須感謝各位口試委員，對我的論文給予寶貴的建議，是我能順利的完成論文。

我也要感謝所上的支援，在升上研究所時，所上提供的獎學金，讓我在研究生生涯的兩年，能夠不用擔心生計，全心的進行研究。

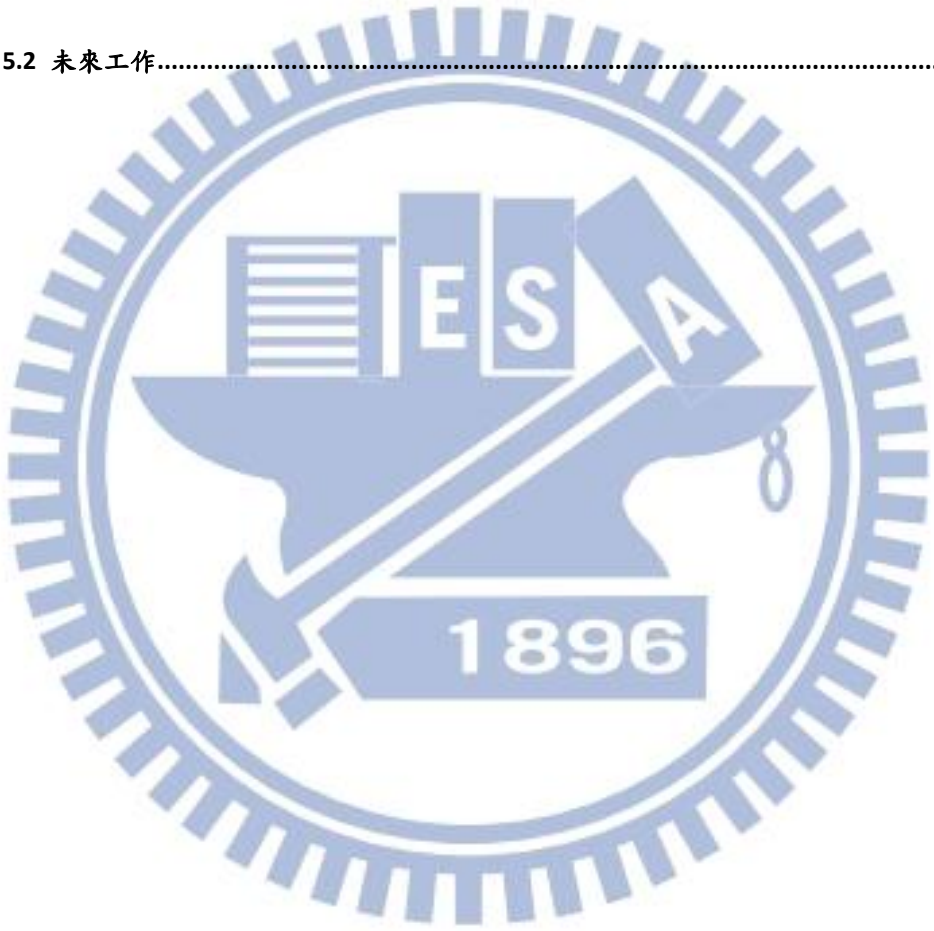
在框架建構時，也非常感謝我的同學與學弟們，像在棋譜編輯器部分，康皓華與廖挺富協助我完成編輯器框架；在電腦對局遊戲桌機網格系統為我提供完善的支援的陳干越，使我能快速地完成工作者測試工作以及吳東穎、張元耀等學弟，因為學弟們在系統維護與工作者部屬等地盡力協助，讓我可以心無旁騖的專心研究。

再來要感謝我們的其他的同學們，左存道、郭青樺、鄭吉閎、胡嘉芸等人，以及學長孫德中、林宏軒、謝志偉與學弟妹，在課業上遇到問題時，與我一起討論切磋，大家互相砥礪加油。

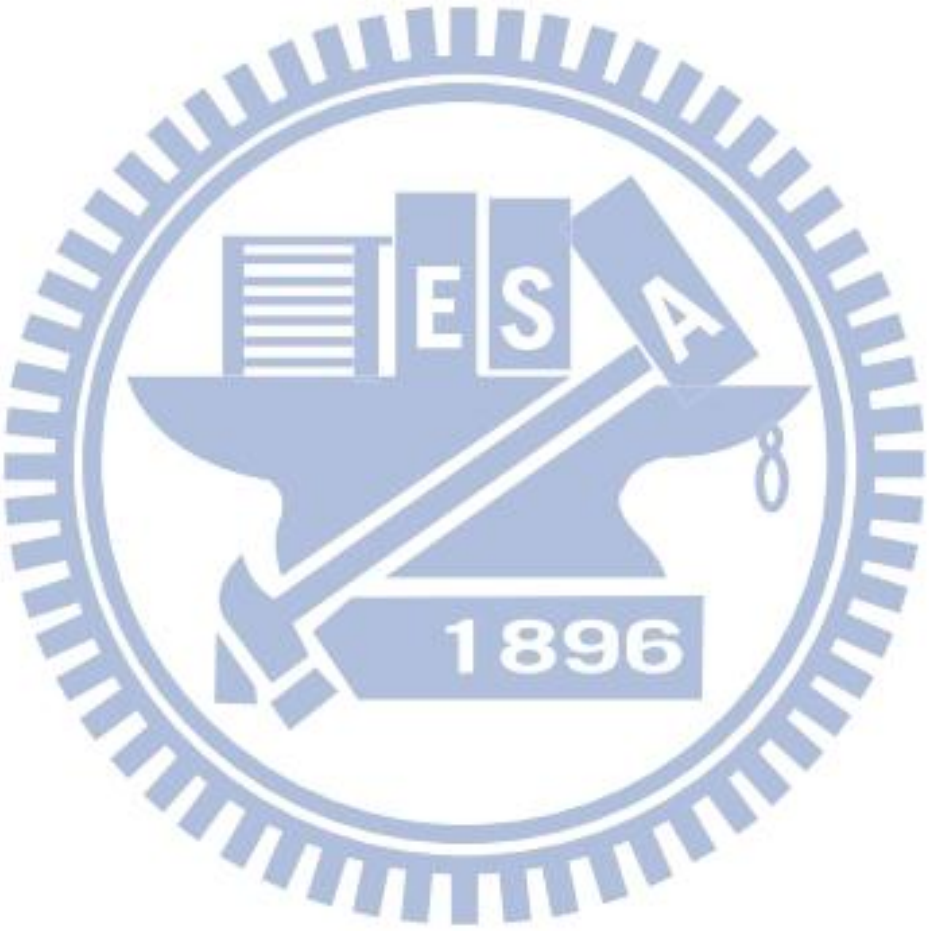
最後要感謝我最摯愛的家人，讓我沒有後顧之憂的完成學業，總是支持著我，謝謝你們。

民國一百零一年八月 於 新竹市交通大學工程三館

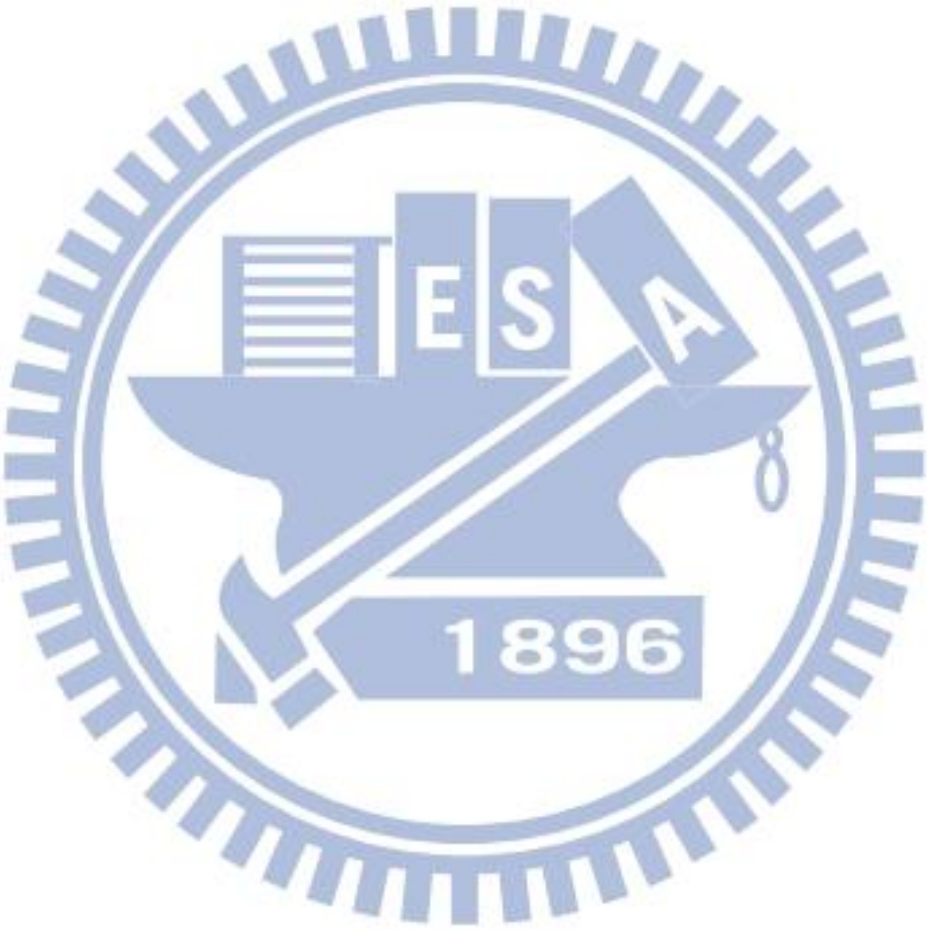
目次



表目錄



圖目錄



第一章　緒論

電腦對局遊戲（Computer Games）經過多年的發展，其人工智慧程式已有逐步追上甚至超越人類的程度。能夠達到這樣的成就，各種輔助工具軟體與環境絕對功不可沒。

另一方面，業界早已開始採用包含 OOP、agile programming[3]、extreme programming[9][11][13][15]、test-driven programming[4][12][16]等等各種開發技術，藉由軟體工程，我們能將學術用軟體也保持在一個容易維護的狀態，如此更能增加學術研究的效率。

在這篇論文中，我們將說明我們是如何重新設計與重構過去開發的輔助工具，並將其框架化，來使它們易於維護且具有延展性。

1.1 研究動機

為了電腦對局遊戲的發展，我們過去開發了一些工具，如棋譜瀏覽器[17]、計算網格[5][7][25]等等用來輔助電腦對局遊戲人工智慧的開發。然而這些工具在沒有有系統的設計下增減功能，而變得難以維護，而且我們發現有越來越多的需求是在原本的結構下難以發展的。

我們過去自主開發的工具包含棋譜編輯器[6]與電腦對局遊戲桌機網格[21]。棋譜編輯器是用來顯示棋譜的輔助工具；電腦對局遊戲桌機網格則是用來提供遠端運算的系統環境。下面小節將逐一介紹它們與重新設計它們的原因。

1.1.1 棋譜編輯器（Game Record Editor）

過去我們為了六子棋[22][23]研究開發了一款編輯器程式，而在我們開始進行圍棋、象棋等的研究後，它也被改成能適合其他棋類。

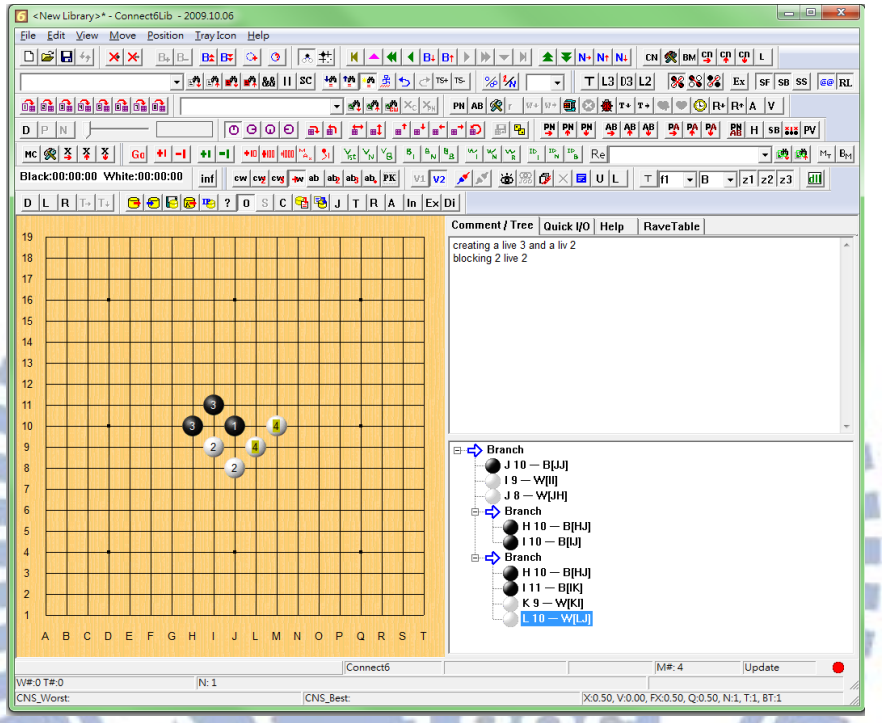


圖 1：棋譜編輯器

在發展其他棋類編輯器的過程中，我們發現在不同棋類間是有相當的共通性，於是我們就希望能將編輯器改寫成能夠快速的改成適合其他棋類的框架，使其他棋類能更容易的套用進來。

1.1.2 工作層級搜尋演算法（Job-Level Search Algorithm）

除了基本的棋譜閱覽功能，編輯器還支援連接到電腦對局桌機網格，並能容易地創造工作、手動分派工作到網格系統內，甚至能夠將上述流

程根據工作層級搜尋演算法（Job-Level Search Algorithm）給自動化。

工作層級搜尋（Job-Level Search）是一種將演算法平行化的方法（Methodology），是由吳教授等人提出[24]，原本用於平行化證明數搜尋（Proof Number Search）[1][2]的方法。後來發現工作層級搜尋（Job-Level Search）的概念是能夠套用在其他很多種演算法上，如蒙地卡羅樹搜尋（Monte-Carlo Tree Search）[8]及 alpha-beta 搜尋[10]等。

有鑑於此，我們也希望各類演算法能快速的改用工作層級的方式，因此在編輯器上，我們也期望提供一個框架使得工作層級搜尋演算法能夠快速的發展。

1.1.3 電腦對局遊戲桌機網格（Computer Game Desktop Grid）

如上所述，電腦對局桌機網格[21]是用來提供遠端運算的平台，其由三者所組成：仲介者（Broker）、工作者（Worker）、使用者（Client）。使用者經由連線到仲介者送出工作，而仲介者再轉送工作給工作者；同樣地，工作者的工作結果也會經由仲介者送回使用者。

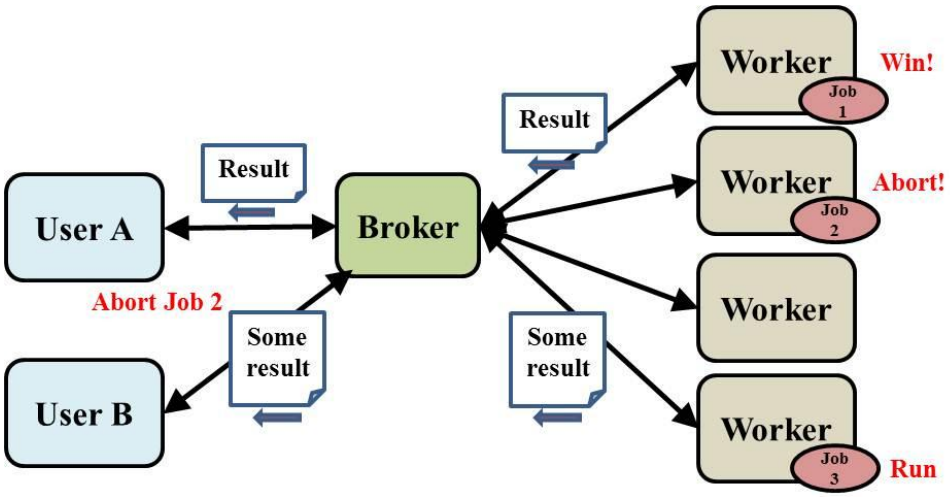


圖 2：電腦對局遊戲桌機網格概念圖

過去已有一個能在 Windows 與 Linux 執行的工作者軟體，但其程式碼是分別為 Windows 與 Linux 維護一份，如此導致工作者難以被轉移到其他作業系統，且也造成維護的困難。

另外，由於在過去，電腦對局桌機網格並沒有錯誤容忍的能力，我們便提出多重仲介者的構想，冀望使這平台能夠提供錯誤容忍、可延展、以及負載平衡的能力，仲介者的修改主要在[14]，此篇論文主要在說明包含工作者與使用者（前面所提的編輯器即是使用者的一例）的終端兩方之修改。我們也在這裡發現使用者與工作者是有著一些相似處。

因此，我們期望工作者能有一個方法來快速地轉移到其他作業系統，並且能輕鬆地維護；另外在工作者與使用者間，也希望能重複利用它們之間相似的模組。

1.2 研究目標

我們希望能提供一整套的軟體框架，使用它，發展者可以：

- 簡單的發展各種棋類的棋譜編輯器。
- 簡單的將工作者移植到各種作業系統上。
- 簡單的快速發展各類工作層級搜尋演算法。

為了能提供上述功能，框架內包含下列元件：

- 用來連接電腦對局桌機網格的函式庫。
- 設計棋譜編輯器用的應用程式介面。
- 支援工作層級搜尋演算法的應用程式介面。
- 用以移植工作者至不同作業系統的應用程式介面。

1.3 本文大綱

在第一章我們會介紹此篇論文之研究動機與欲完成的目標，第二章中則簡介過去開發的輔助工具與環境，第三章會說明我們的框架是如何設計的，並在第四章搭配一些簡單的個案分析，說明套用這框架後開發會變得有多容易，最後在第五章下一個結論，並說明未來的研究方向。

第二章　背景

本章節將會說明本篇論文之研究背景，簡單介紹過去發展的輔助工具，並說明在這些工具上存在的問題。在 2.1 節我們會介紹過去發展的編輯器與其提供的功能，而在 2.2 節我們會介紹電腦對局遊戲桌機網格，特別著重在其中的工作者與通訊協定的問題。

2.1 棋譜編輯器（Game Record Editor）

在發展電腦對局遊戲之人工智慧時，棋譜編輯器是很重要的，它能使開發者方便分析人工智慧所產生之棋譜樹與紀錄等等，甚至能夠支援創建開局庫、與電腦對局遊戲桌機網格連結等等附加功能。

過去我們先找到一款開原的五子棋編輯器，將之修改成能給六子棋使用的版本，稱作 Connect6Lib[6]。後來再基於它修改出象棋、圍棋等等版本的編輯器。它包含棋盤顯示、樹顯示、一些輔助的分頁、以及用來提供如棋盤轉向等等功能的各種工具列。

如圖 3，左下是棋盤顯示區，在這區會畫出棋盤與棋子，可以快速地知道盤面狀況如何。在右下的樹顯示區則會畫出棋譜樹的模樣。右中的分頁視窗支援很多方便的功能，像是編輯當時指定的節點的註解等等。最後是上面的大量工具列，支援對電腦對局遊戲桌機網格的操作、資料庫操作等等功能。

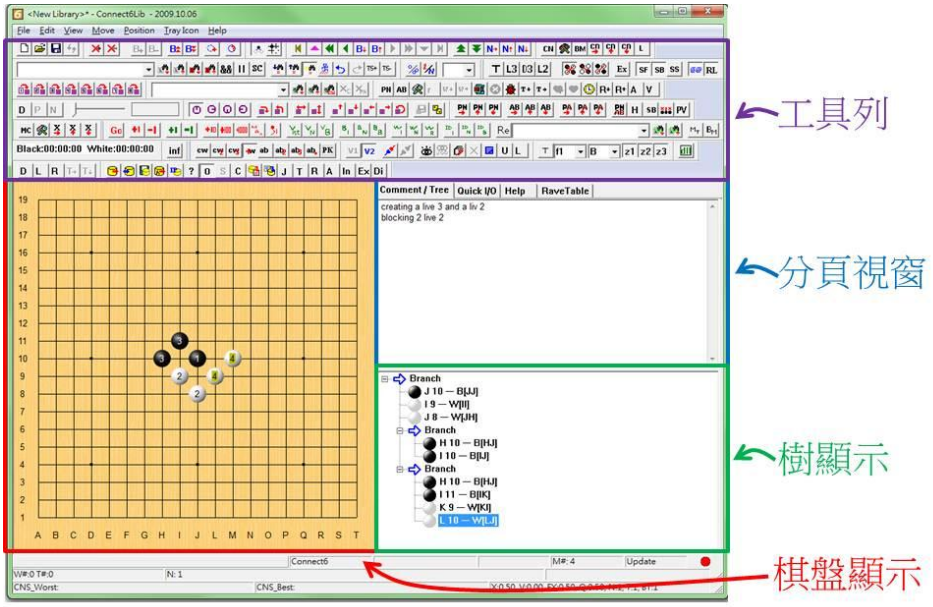


圖 3：Connect6Lib

Connect6Lib 以我們的眼光來看，已經越來越難維護，舉例來說，在棋盤顯示畫棋盤、棋子的函數被分割成六子棋版、圍棋版、象棋版，在這樣的狀況下，想再加入其他棋類會越來越困難。

2.2 電腦對局遊戲桌機網格

電腦對局遊戲桌機網格系統（Computer Game Desktop Grid）是研究室發展的一套專門用來解決電腦對局遊戲領域問題的系統[21]，其為一種計算型的網格系統，同時也是一個志願型計算系統，包含使用者、仲介者、工作者。

在第一章我們已介紹過電腦對局遊戲桌機網格，這邊我們將只著重在工作者與系統的通訊協定對開發者造成的麻煩做說明。

2.2.1 工作者（Worker）

工作者安裝於一般家用桌上型電腦上，在對電腦使用者造成最小限度的干擾下進行高度動態的工作執行以及管理，並即時將結果以串流技術回傳至仲介者處。

實作上，工作者經由 Socket 與仲介者溝通，同時，對工作者創建的應用程式行程則是用標準輸入與標準輸出的重新導向，也就是說，以輸出來說，應用程式端只需經由標準輸出來發出訊息，工作者便會將其經由仲介者送至對應的使用者端。

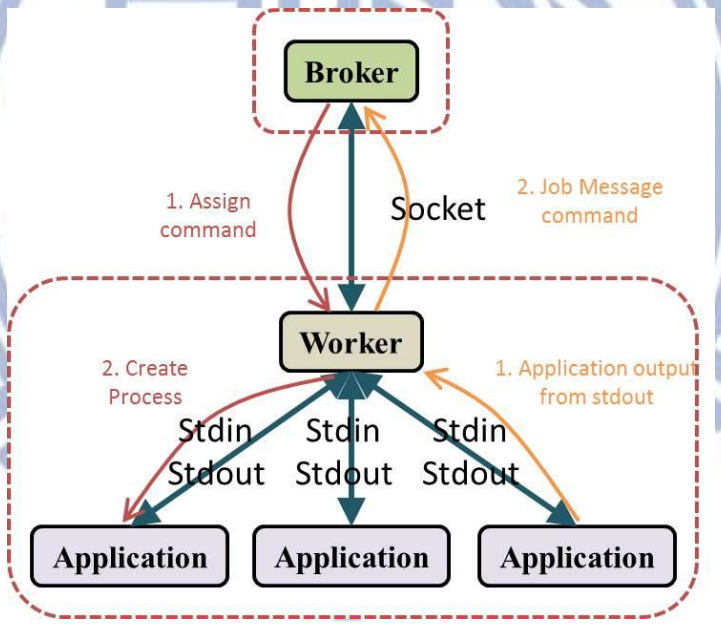


圖 4：工作者概念圖

過去我們已經分別為 Windows 與 Linux 開發了工作者，但是由於開發時的思慮不夠周詳，導致程式碼在不同的作業系統上各維護了一份，造成維護上的麻煩，且若我們需要將工作者移植到其他作業系統上會相當辛苦。

2.2.2 通訊協定與連線機制

以終端開發者的角度而言，電腦對局遊戲桌機網格內有些細節是相當繁瑣的，譬如說通訊協定的內容與連線機制等等。

由於多重仲介者計畫，整個電腦對局遊戲桌機網格有些機制與通訊協定上的修改。以終端要連線到仲介者為例，相對於以前的直接連線，在多重仲介者架構下，需要先問整個系統該連線到哪台仲介者，接著才回到原本單一仲介者狀況。

另外整個通訊協定是基於 XML 設計的，因此對開發者來說，他必須知道各種 Tag 與其內容，瞭解通訊協定的細節對其他組織的開發者也是種負擔。

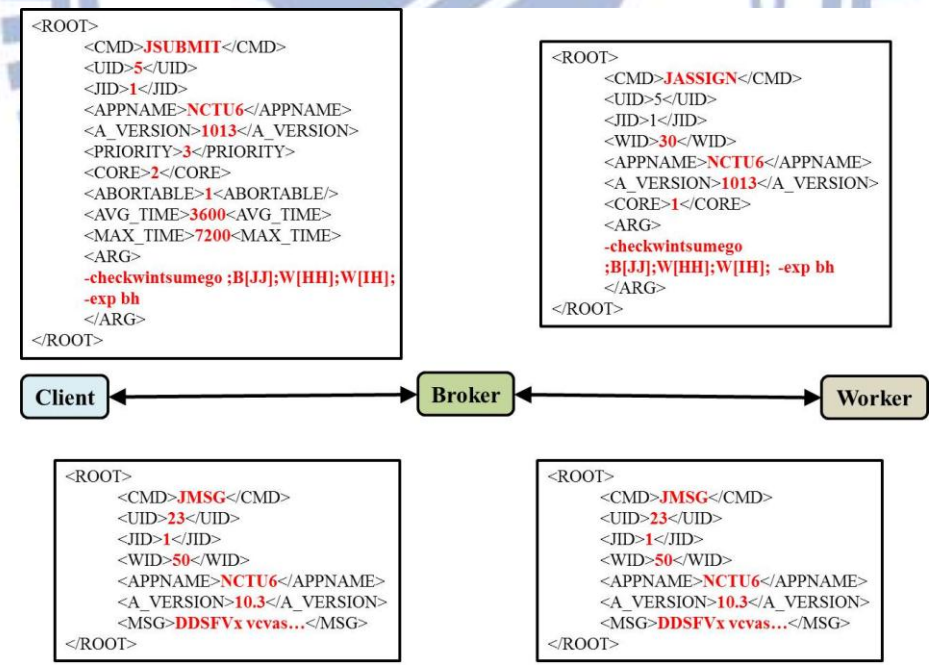


圖 5：CGDG 通訊協定範例

第三章　框架設計

本章節將會說明本篇論文之框架設計，以及各項元件的設計。

3.1 系統架構（Architecture）

本小節將詳細說明系統架構，圖 6 為整個框架內的元件與其層級關係圖，包含棋譜編輯器框架（含工作層級搜尋演算法支援）、電腦對局遊戲桌機網格函式庫（CGDG library or libCGDG）、工作者框架。

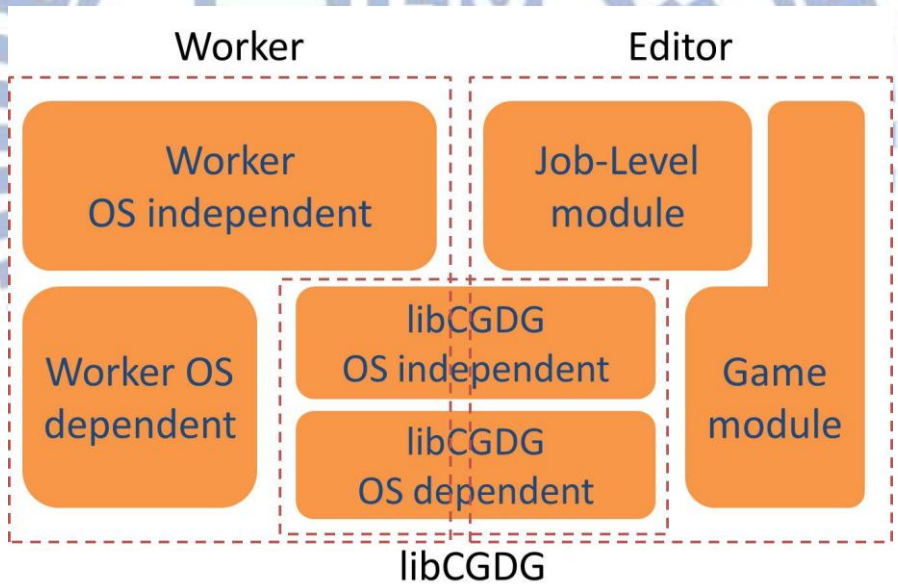


圖 6：系統架構圖

CGDG library 提供一個函式庫來與電腦對局遊戲桌機網格進行連線、溝通、工作交換等等，而棋譜編輯器與工作者框架皆使用 CGDG library。編輯器框架內則分成兩大部分：遊戲模組與工作層級模組，細節將在下

面小節再介紹。工作者內的模組則分成與作業系統相關和與作業系統不相關的兩個種類，同樣的也在後面再詳細介紹。

3.2 棋譜編輯器框架之設計

我們將整個框架分成兩個大模組：

- 遊戲模組。即為基本的棋譜編輯器本體。
- 工作層級模組。掛在遊戲模組上，提供給工作層級演算法開發。

整個棋譜編輯器框架採用的設計哲學為：盡量減少不同遊戲或演算法間的重複程式碼，具體來說，那些重複的程式碼會被我們放到一個基底的類別，各遊戲或演算法的發展者只需繼承此基底類別來發展他們的需求就能快速地完成一個棋類編輯器或工作層級演算法。

3.2.1 遊戲模組（Game Module）

在遊戲模組中，我們採用 MVC 設計模式來開發。

MVC（Model-View-Controller），中文翻譯為模型、視圖、控制器，是一種在軟體工程中使用的軟體架構模式[19][20]，其概念如圖 7。在 MVC 中，主體邏輯與使用者介面被分開，使發展者能獨立地發展、測試、及維護各自的部分。

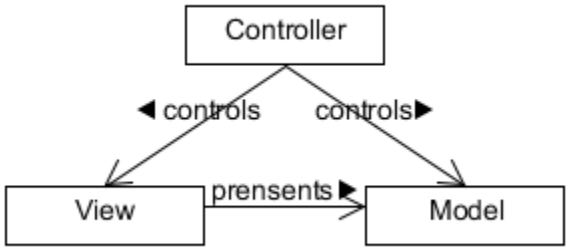


圖 7：MVC 概念圖

模型即為主體邏輯。以棋譜編輯器為例，用來儲存棋譜的資料結構，以及那些用來修改此資料結構的子程序等，就是模型的一部分。

視圖是用來顯示主體邏輯，也就是模型，應該是什麼樣子。同樣以棋譜編輯器為例，當前盤面該如何顯示就是視圖該做的。

控制器負責將使用者介面的控制單元與模型及視圖中的子程序對應起來。舉例來說，當使用者介面中的「存檔」按鈕被按下時，控制器會去呼叫負責存檔的子程序。

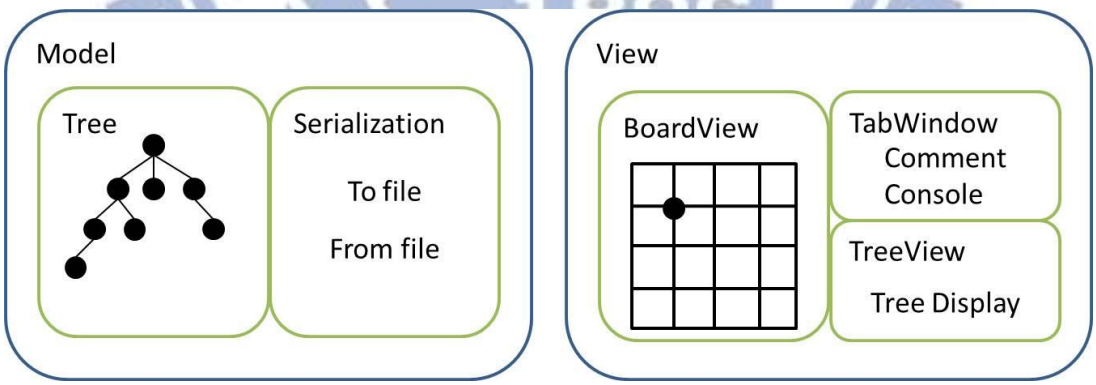


圖 8：遊戲模組結構圖

遊戲模組包含模型與視圖。模型包含一個一般化的樹與序列化功能，而視圖包含三個小視圖，棋盤視圖、樹視圖以及分頁視窗。另外之所以控制器在這邊沒有被提及是因為我們採用 Microsoft Foundation Class

（MFC）做開發，而 MFC 內並沒有一個顯而易見的控制器類別，控制器四散在模型與視圖類別內的訊息映射表內，故我們在此不多述。

3.2.1.1 模型（Model）

模型內有兩個最主要的組件，棋譜樹的本體與序列化功能。如上所述，我們將一些可能不同的地方以多型的方式來留給發展者去定義。而另外有提供一些方便的基本功能，如樹的遍歷等等給發展者使用。

在主體資料模型中，會需要一個資料結構來儲存棋譜，一般的棋譜只有一條路徑，但我們需要能看任何搜尋過程中訪問的點，而能夠處理這種需求的資料結構即為多元樹。我們希望整個樹結構能夠被所有遊戲類型使用，所以我們將樹定義得盡量一般化，並將多元樹轉成二元樹。

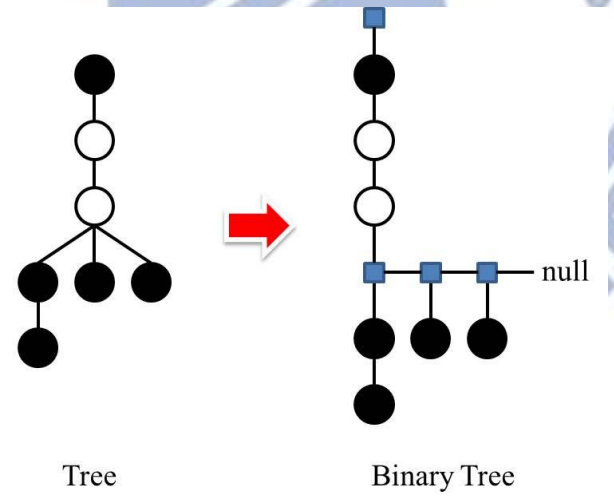


圖 9：棋譜樹概念圖

棋譜樹是由很多的節點所構成，其中節點會有連向其他節點的連結，其結構會在下一段更仔細的描述。另外，我們還提供了一個工具方便發展者來做樹的遍歷，叫做 NodePointer。

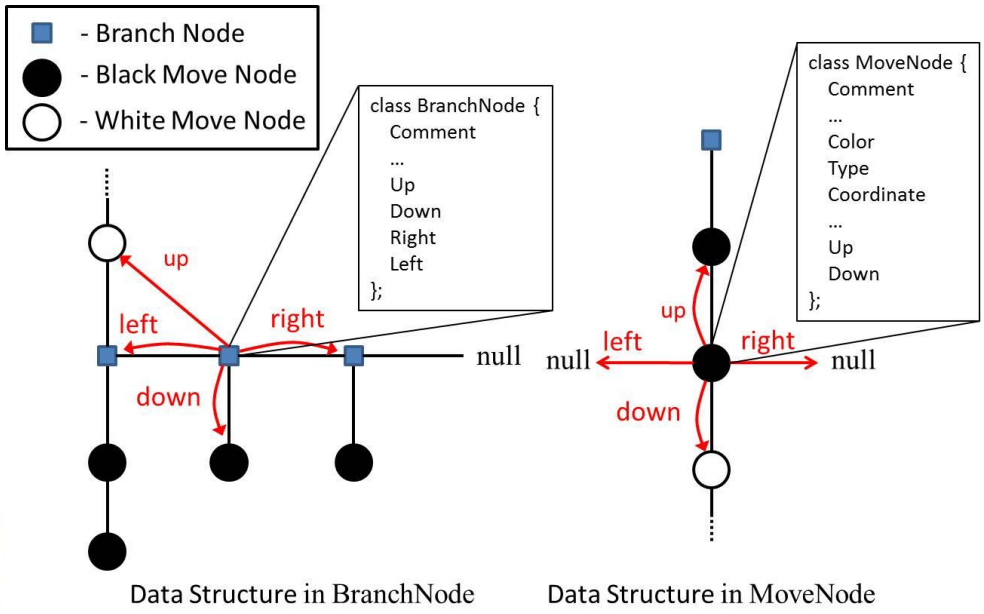


圖 10：節點資料結構圖

在節點的資料結構內，包含了一些基本的資料結構來儲存一個棋步，如座標、顏色、棋子的類型等等，另外還有各個方向的連結用來維持樹的結構。連結有兩類，垂直向的與水平向的，如圖 10。垂直向的連結用來依棋步手順來訪問樹，水平連結則用以拜訪當下的盤面的其他走法。而對棋步節點來說當然只有垂直的連結。此外，我們也限制了該連結的可視範圍，發展者不會碰到這些連結，這樣也能降低發展者開發時的負擔。

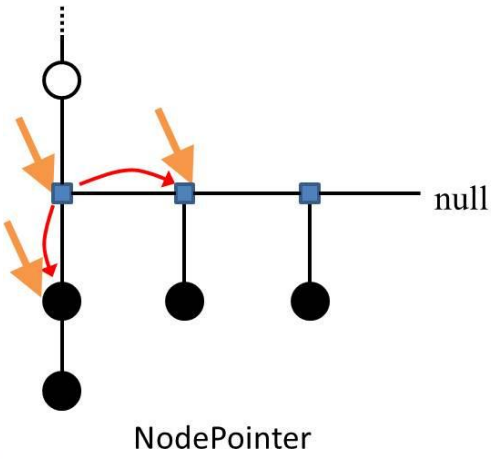


圖 11：NodePointer 概念圖

通常樹與節點的結構對發展者來說是不重要的，所以我們提供一個像 STL 的 Iterator 的機制來讓發展者使用，叫做 NodePointer，如圖 11。使用者可以不用自己改連結，而是利用 NodePointer 來在樹結構上進行移動、賦值、結構改變等等行為。

最後是序列化功能，我們使用 Smart Game Format 來儲存棋譜。SGF 是一種用來儲存棋譜的一般化檔案格式。在序列化時會不一樣的地方是如何解釋一個棋步，因此我們提供能進行序列化功能的基本類別，而將棋步解釋留給發展者。

3.2.1.2 視圖（View）

視圖是用來顯示當前盤面的狀態，它包含三個子視圖元件：

- 棋盤視圖，用來顯示當前盤面。
- 樹視圖，用來顯示樹的樣子。

- 分頁視窗，一個可以方便發展者掛上更多附加功能，像註解視窗的工具。

各種子視圖也被抽象出一個基底類別，若有需要，發展者可以覆寫他們來給予新的定義。

其中棋盤視圖的複雜度是高於樹視圖與標籤視窗的，因為它負責繪出各種棋盤與一些其他的輔助顯示。若是六子棋，則其棋盤視圖要負責繪出圍棋棋盤、棋子，還要顯示盤面上的活四等等。若是象棋則要畫象棋棋盤與棋子。

樹視圖負責顯示棋譜樹的樣子，它用類似檔案瀏覽器顯示資料夾下有什麼檔案的方式，來顯示棋譜樹，讓使用者可以展開或收起一個節點。以六子棋為例，其樹視圖需要在節點顯示棋子位置；象棋則是要顯示棋步描述，如圖 12。

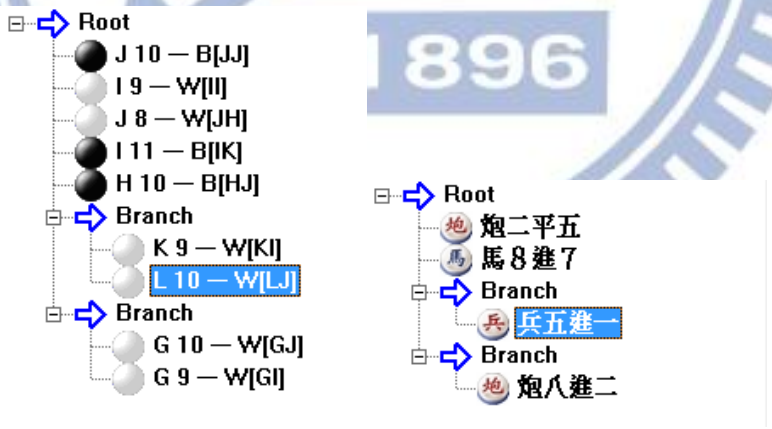


圖 12：樹視圖範例圖

分頁視窗提供了一個能夠簡單的加入其他附加功能的介面。過去想要在編輯器內加入一頁新的功能需要知道很多分頁操作的細節，而現在可以經由分頁視窗來簡單的加入其他顯示。

目前基本的分頁視窗包含有：

- 註解視窗（Comment Window）。顯示節點的註解。
- 主控台視窗（Console Window）。顯示偵錯資訊。
- 快速輸入輸出視窗（QuickIO Window）。快速地合併一串棋步進入樹內。

3.2.2 工作層級模組（Job-Level Module）

工作層級模組提供一個能簡單實作自動化送出工作的應用程式介面。我們會分兩個部分來說明：

- 工作層級演算法：說明如何快速實作工作層級搜尋演算法。
- 遊戲策略：說明如何快速套用演算法至某特定棋類。

3.2.2.1 工作層級演算法（Job-Level Algorithm）

所謂的工作層級是指在演算法運作時，將搜尋節點作為一個工作丟到電腦對局遊戲桌機網格，並等待其結果回來的一種方式，如圖 13（NCTU6是我們的六子棋人工智慧）。由[25]，我們從工作層級證明數搜尋中歸納出整個階段分三類觸發事件：使用者要求開始執行、有空閒工作者、工作結果回傳。

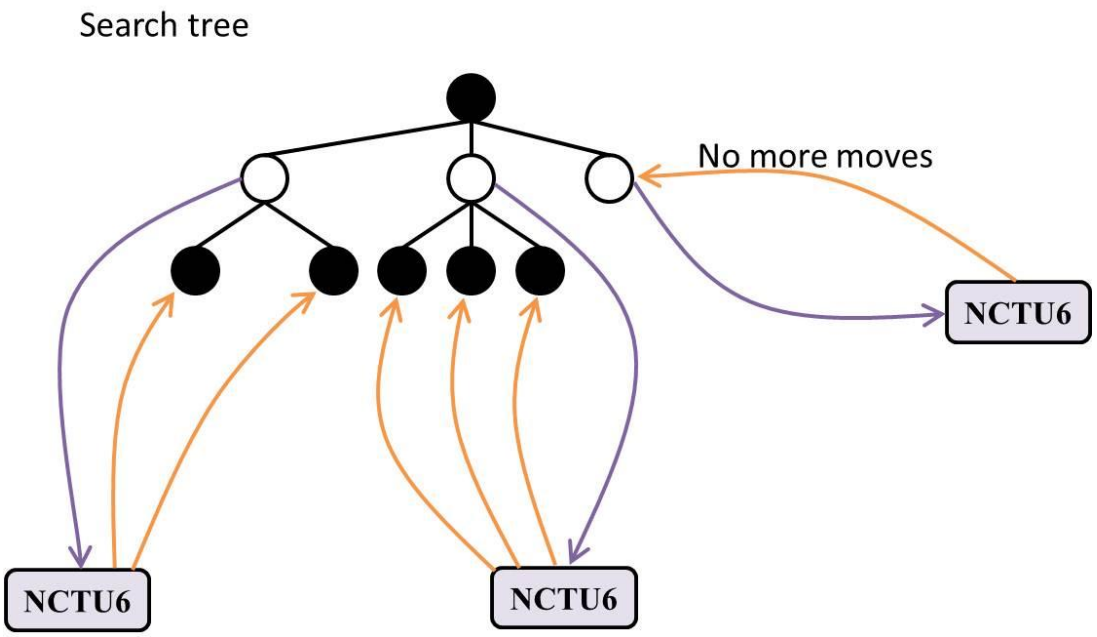


圖 13：工作層級搜尋演算法概念圖

初始化發生於使用者要求執行演算法時，這時需開始做演算法的初始化工作；當有空閒工作者時，這時需決定該選擇哪個節點包裝成工作並送到網格系統中，並進行更新；當結果回傳時，需更新該工作對應的節點並確認是否做收尾工作。

我們採用了 Template Method 設計樣式，將這些共通行為全部定義成八個純虛擬函式，利用覆寫的方式讓各演算法發展者可以清楚地切割演算法的行為。八個函式分別為：

- Initialize：進行初始化。
- Select：選擇要作為工作送出的節點。
- Pre-Update：預先更新樹，避免下次仍會選到同個點。
- Dispatch：包裝工作並送出。

- Parse：將送回的資料放入資料結構中。
- Update：根據送回的資料更新搜尋樹。
- CheckFinish：確認是否完成。
- Finalize：進行收尾的工作。

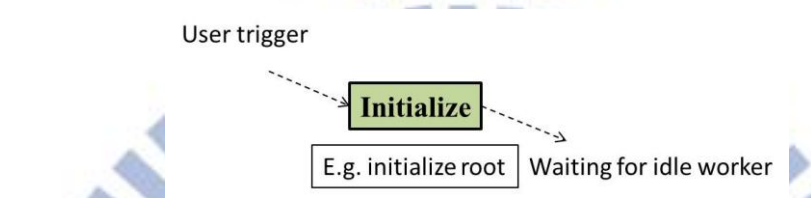


圖 14：初始化階段

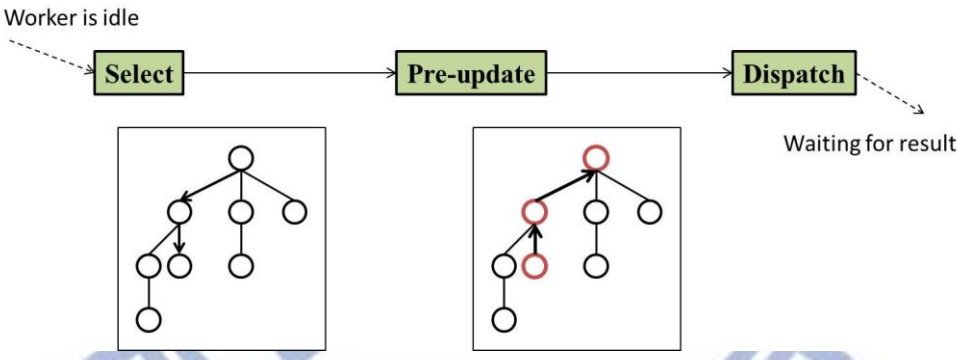


圖 15：空閒工作者階段

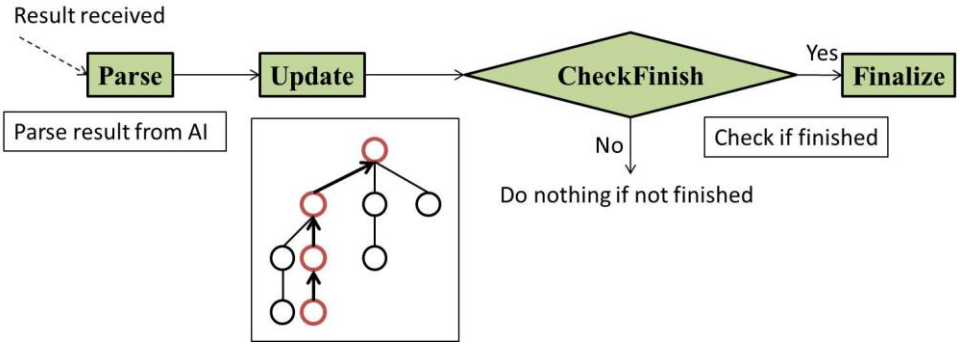


圖 16：工作結果回傳階段

3.2.2.2 遊戲策略（Game Policy）

當我們執行一個Job-Level演算法時，會有些資料屬於演算法層級的。舉例來說，此節點下的哪個子節點是應該下一個被訪問的；但同時也會有一些是只跟遊戲人工智慧有關的，像是特徵數量或盤面的估計值等等。要我們將這些全部都設計在一個結構內是不可能的。所以我們將這些資料分成兩類：遊戲的資料與演算法的資料。

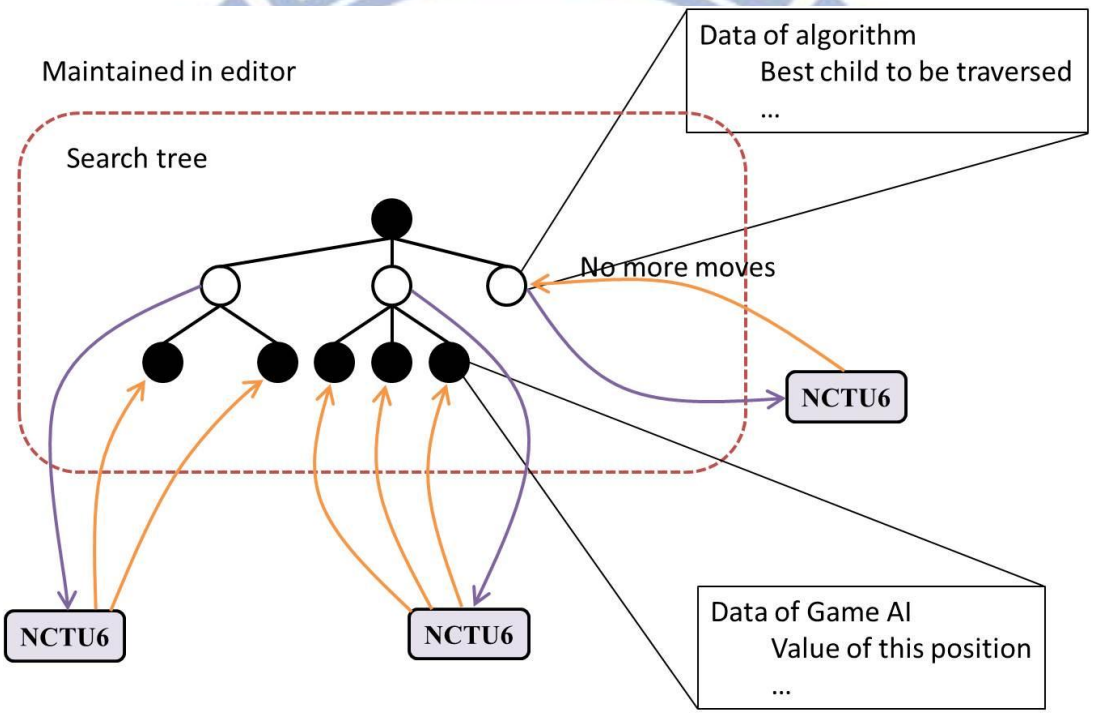


圖 17：資料觀點

而要讓演算法能適用各棋類，它們必須知道該棋類人工智慧的資料；以演算法的角度來說，知道演算法的資料很自然，但因為演算法與遊戲應該分開的，所以演算法不應該知道遊戲人工智慧的資料。若我們照之前的作法用繼承解決這問題的話，演算法數量會有爆炸性的成長。

因此，我們將人工智慧資料的處理對每一演算法，內部會有一個稱作遊戲策略（Game Policy）的物件來處理關於遊戲的資料。它會負責處理關於遊戲的資料，並作為一個代理人來進行一些關於遊戲的決定。如果有需要，遊戲策略也能依需求，來取出屬於遊戲人工智慧的資料。如此一來，演算法就能從遊戲內獨立出來。

3.3 電腦對局遊戲桌機網格函式庫之設計

如前面所述，工作者與使用者等電腦對局遊戲桌機網格終端是有能共用的部分，如圖 18。另外對終端發展者來說，處理像是通訊協定等細節是很惱人的。為了讓發展者不用處理那些細節，我們將連線與通訊等等工作抽成一個函式庫叫做 libCGDG，寄望能簡化開發流程並將連接上電腦對局遊戲桌機網格的工作與應用程式分開。

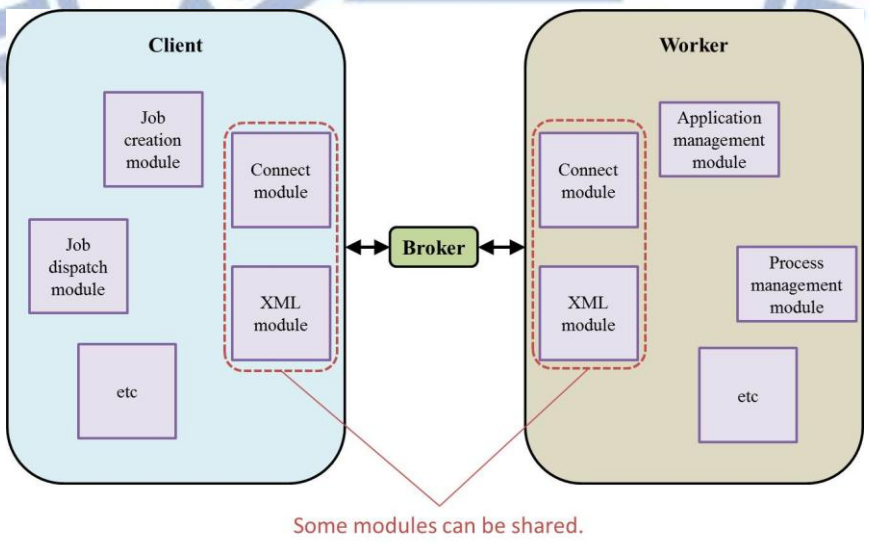


圖 18：終端模組可重複利用性

libCGDG 分成與作業系統相關與無關兩部分，如圖 19，使它能容易

地與一些其他的像 MFC、X-Window 等視窗程式框架結合，並支援發展者開發他們自己的應用。libCGDG 使用事件驅動模型來設計，如此可以避免讓使用者還要處理並行性（Concurrency）的問題。

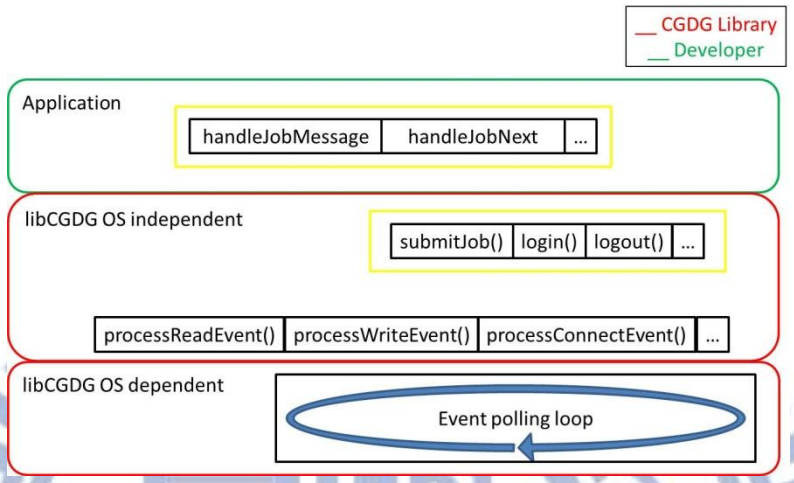


圖 19：libCGDG 階層圖

與作業系統無關的部分，libCGDG 提供了兩個介面，一個是用來送指令給 Broker，另一個是用來接收從 Broker 送來的訊息。發展者需實作接收用的介面來收訊息並進行像送出工作等等發展者所希望的操作。

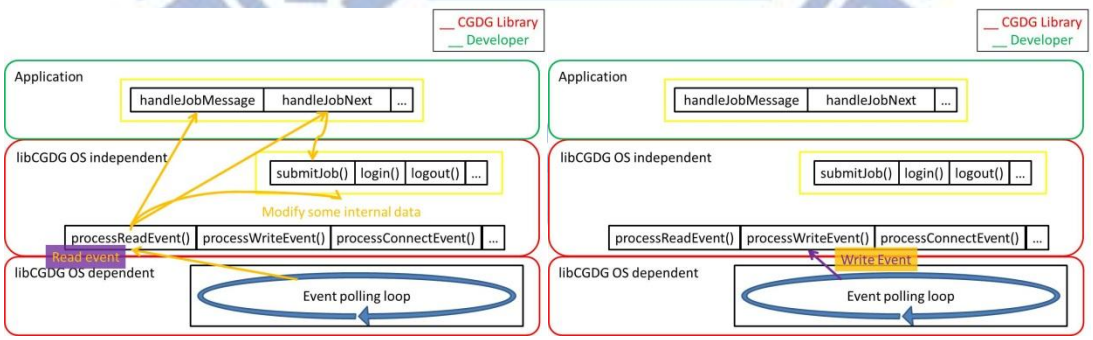


圖 20：libCGDG 程序流

在與作業系統相關的部分，若我們想讓 libCGDG 能在所有平台上執行，會遇到一些相依於平台的限制，包含如何使用 Socket、如何讓 Socket

做輸入輸出等等。我們將 Socket 行為包裝成一個介面，並對不同作業系統實作 Socket 與事件輪詢迴圈。當有 Socket 事件發生時，會去呼叫各事件的處理函數，從而向上通知發展者，如圖 20。圖 21 是其資料流。

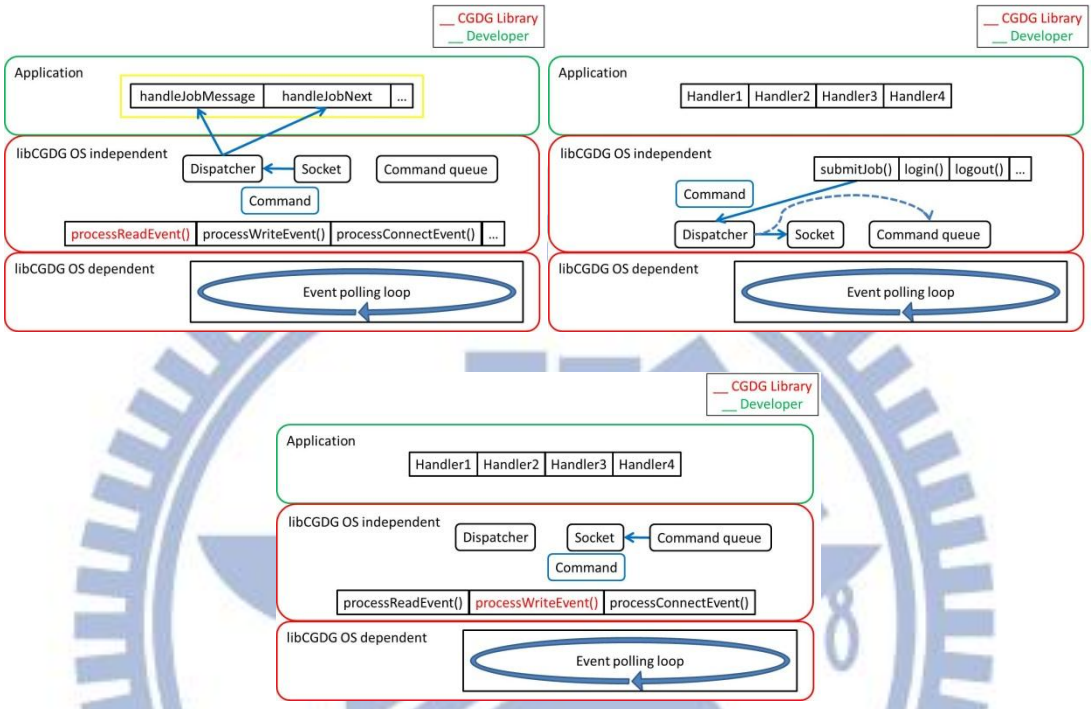


圖 21：libCGDG 資料流

3.4 工作者框架之設計

工作者實際上須提供的功能都跟過去的工作者程式並無二致，故我們只是重構過去的程式，使其能容易地維護。在工作者框架內，我們採用單執行緒事件導向模型，使工作者實作人只需要定義當特定事件發生時該進行對應的行為即可。內部模組分成兩大類：

- 與作業系統無關之模組（圖 22 上面的紅圈）。
- 與作業系統相關之模組（圖 22 下面的紅圈）。

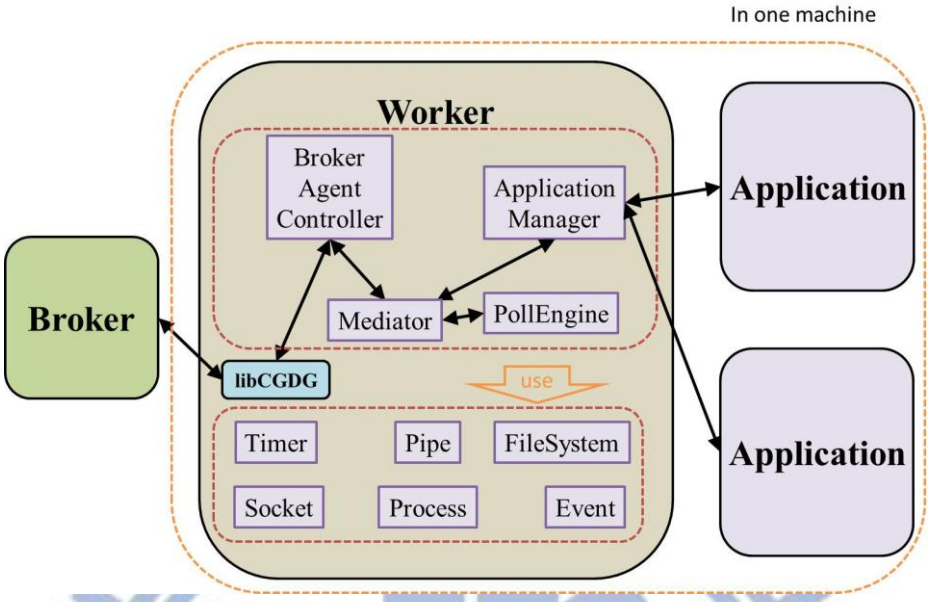


圖 22：工作者模組圖

與作業系統無關之模組只會有一份程式碼複本;與作業系統相關之模組則在不同作業系統都會維護一份。若有需要將工作者搬到其他作業系統時，只需將與作業系統相關之模組抽換掉即可。下面會再詳述兩類模組內所包含的類別。

3.4.1 與作業系統無關的模組（OS Independent）

與作業系統無關的模組負責進行較高階的操作,比如說應用程式行程管理、與仲介者的溝通、應用程式下載等等，共有以下類別：

- BrokerAgent
- BrokerAgentController
- ApplicationManager

- PollEngine
- Pollable
- Mediator

BrokerAgent、BrokerAgentContrller、ApplicationManager 是最高階的類別，他們負責處理各事件發生時該做的事情。BrokerAgent 負責扮演與仲介者溝通的橋樑，由它負責接受仲介者送來的指令，且也經由它向仲介者發指令。而由於電腦對局遊戲桌機網格使用多仲介者，故需要 BrokerAgentController 來做 BrokerAgent 物件的管理。ApplicationManager 負責應用程式的下載、執行應用程式、應用程式行程的管理。

Pollable 與 PollEngine 將事件輪詢迴圈其本身給包裝起來，對工作者的發展者來說，他只需要去跟 PollEngine 登錄有興趣的事件與其對應處理函式即可。Pollable 則提供各類可輪詢的物件或事件一個詢問介面，PollEngine 經由此介面去詢問是否有事件發生。

Mediator 實作了同名的設計樣式，用以降低類別間耦合度，在類別增加且互相又有關聯性的時候，程式會變得難以維護，使用 Mediator 可以使耦合度降低，使程式碼容易維護。

3.4.2 與作業系統有關的模組（OS dependent）

這部分的類別，為了效能上的考量，我們不使用繼承來處理不同作業系統的模組抽換，而是使用 C++的模板。我們在編譯時決定實際使用的類別，經由模板進行具現化，如此可以減少使用虛擬函式造成的執行期

負擔。

與作業系統相關的模組內共有以下類別：

- Process
- Pipe
- Socket
- Timer
- Event
- FileSystem
- SystemInformation
- SharedMemory

他們各自定義了在不同作業系統上對應的功能該如何使用。Process 負責行程的創建、移除、親和性與優先權的設定等等。Pipe 用來重新導向應用程式行程的標準輸入與標準輸出。Socket 提供與網際網路的連接能力。Timer 提供計數器的能力。Event 包裝了事件本身該怎麼被詢問。FileSystem 提供檔案系統的存取能力。SystemInformation 讓我們可以拿出像中央處理器能力或記憶體大小等等系統資訊。SharedMemory 提供存取共用記憶體的方法。

第四章　個案研究

在本章節中，我們會用井字遊戲示範該如何快速地實作編輯器，另外用另一個範例，六子棋版的工作層級證明數搜尋，來示範如何快速地實作工作層級搜尋演算法，並快速地套用在任意棋類遊戲上。

4.1 井字遊戲之編輯器實作範例（Editor for Tic-Tac-Toe）

要實作出一個編輯器，開發者需要繼承 BaseEditorDoc、BaseSgfParser、BaseBoardView、BaseTreeView、BaseTabWindow 等等類別，並覆寫他們純虛擬函式（若有），詳細的 UML 類別圖請見圖 23 與圖 24。以井字遊戲為例，假設不需要額外的分頁視窗，則實際要實作的類別有：

- TicTacToeEditorDoc：繼承 BaseEditorDoc，此為模型本身。
- TicTacToeSgfParser：繼承 BaseSgfParser，從檔案讀出棋譜的功能。
- TicTacToeSgfSerializer：繼承 BaseSgfSerializer，寫入檔案的功能。
- TicTacToeBoardView：繼承 BaseBoardView，畫出棋盤、棋子。
- TicTacToeTreeView：繼承 BaseTreeView，顯示棋步敘述。

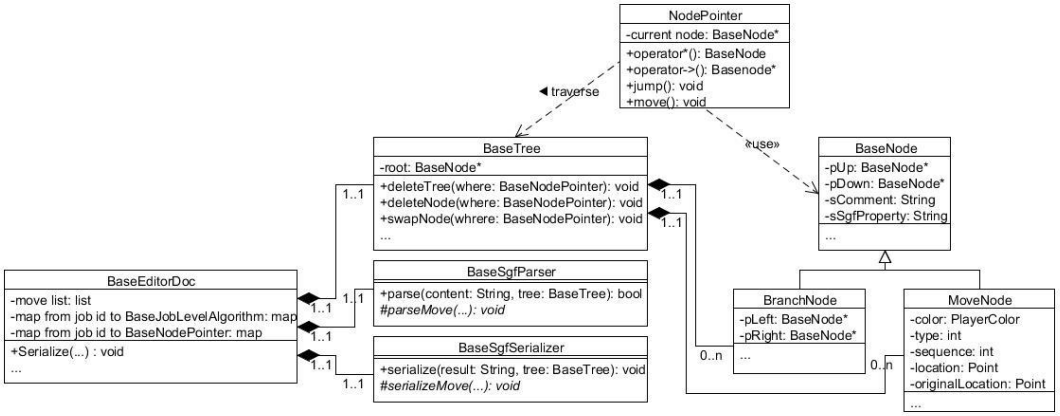


圖 23：遊戲模組模型之類別示意圖

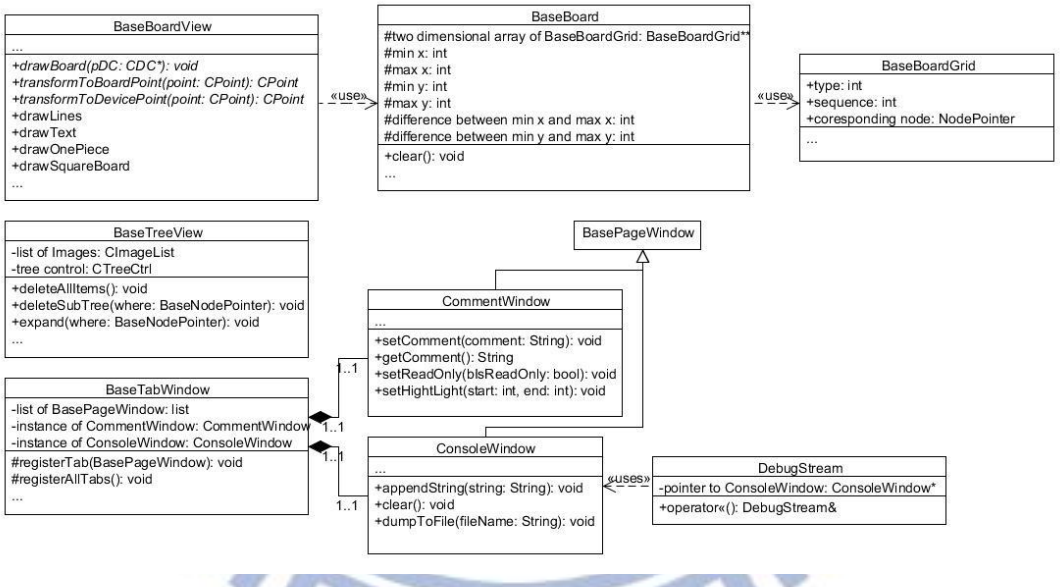


圖 24：遊戲模組視圖之類別示意圖

最後實際要覆寫或需要定義的函式，如在棋譜樹中增加一個棋步節點，或是畫出棋盤等等，我們放在表 1。

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

表 1：井字遊戲編輯器之需覆寫函式表

4.2 六子棋用工作層級證明數搜尋演算法之實作範例（Job-Level PNS for Connect6）

在這小節，我們會用六子棋版的工作層級證明數搜尋作為例子，分別說明：

- 如何實作一個工作層級演算法。
- 如何將已有的演算法套用到特定棋類。

4.2.1 工作層級證明數搜尋演算法本身之實作

要實作一個工作層級搜尋演算法，發展者需要繼承 BaseJobLevelAlgorithm 來定義他的演算法，以這裡的例子即為 JobLevelProofNumberSearch，並覆寫它內部的八個虛擬函式，如表 2。

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

表 2：工作層級證明數搜尋演算法之需覆寫函數

另外需要定義一個 PNS 的遊戲策略之介面 PnsPolicy，讓各棋類發展者能經由此介面定義該棋類的策略，比如說，如何經由人工智慧送來的資訊選下一個的點來送工作。

4.2.2 六子棋遊戲策略之實作

最後，當六子棋的發展者想要套用 JobLevelProofNumberSearch 時，他需要實作 PnsPolicy 來定義他的遊戲策略 Connect6PnsPolicy，再覆寫所有純虛擬函式即可。實際要覆寫的函式由於有些繁雜，在此就不列出。

4.3 比較

The original Connect6Lib and JL-PNS	Game record editing module	Job-level module	JL-PNS	Connect6	Go	Chinese Chess	Mahjong	Tic-Tac-Toe
86688	27854	6658	1178	8215	8843	3535	2192	836

表 3：棋譜編輯器之行數統計

在最後我們做了一個統計，如表 3，實際完成的井字遊戲編輯器程式碼僅有 836 行，相較棋譜編輯器的遊戲模組，其程式碼高達 27854 行，是非常大的差距；而過去發展的 Connect6Lib，其程式碼混雜，在粗略的估計後，純編輯器部分約占四萬行。

另一部分，工作層級證明數搜尋演算法共 1178 行，而棋譜編輯器的工作層級模組共 6658 行，但這還不包括 libCGDG 有 6904 行；另外 Connect6Lib 關於工作層級的部分，其程式碼在粗略估計後，大約占兩萬行，所以框架化的確帶給我們不少好處。

第五章　結論

本章節將會說明本篇論文之結論，並列出可加強之部分，提出未來工作方向以供繼續發展。

5.1 研究結論

Δ Ongoing
O Finish

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

圖 25：編輯器目前已實作之遊戲與演算法

本篇論文主要貢獻在於設計了一個整合了電腦對局遊戲桌機且能容易的發展各種棋類研究的軟體框架，此軟體框架具備下列能力：

1. 定義了一個能快速開發任何棋類編輯器的應用程式介面：發展者在繼承各種基底類別，並覆寫需要修改的虛擬函式就能快速地完成各種棋類編輯器。現在已經有六子棋、五子棋、圍棋、象棋的編輯器正在使用中，參見圖 25。

2. 定義了一個能快速發展各類工作層級演算法，並能快速地套用在各棋類的應用程式介面：我們定義了工作層級搜尋演算法的骨架，發展者只需覆寫該八個純虛擬函式就能完成實作一種工作層級搜尋演算法。現在已有證明數搜尋、蒙地卡羅樹搜尋、人工智慧競賽等演算法正在運作中，參見圖 25。

3. 提供一個用來連線到電腦對局遊戲桌機網格的函式庫：我們在編輯器與工作者皆使用這個函式庫，增加程式碼的再利用程度。

4. 定義了一個能快速移植工作者到各作業系統的應用程式介面：藉由它，我們可以很快速的發展各作業系統上的工作者，增加能支援的平台。目前已有 Windows 和 Linux 版本正在使用中。

我們相信這樣的一個軟體框架，能使各種輔助工具容易維護且容易的拓展到各種方向，對電腦對局遊戲領域發展有極大的幫助。

5.2 未來工作

本軟體框架提供了研究與發展者一個良好的環境，但這只是個開始。棋譜編輯器端，要能支援非完美資訊遊戲（Imperfect Information Game）甚至推廣到一般化的搜尋。在工作者端，希望能做更細節的控制，如記憶體控管、硬碟容量管理等等。如此使此環境更臻成熟，解決更多電腦對局，甚至是一般化的搜尋議題。

參考文獻

[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] Beck, Kent; et al. (2001). "Manifesto for Agile Software Development" . Agile Alliance. Retrieved 14 June 2010.

[4] Beck, K. Test-Driven Development by Example, Addison Wesley - Vaseem, 2003

[5] BOINC website. avalible at http://boinc.berkeley.edu/.

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

[7] Condor website. available at http://www.cs.wisc.edu/condor/.

[8] 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.

[9] "Design Patterns and Refactoring", University of Pennsylvania, 2003, lecture slide at http://www.cis.upenn.edu/~matuszek/cit591-2003/Lectures/49-design-patterns.ppt.

[10] Edwards, D.J. and Hart, T.P., The Alpha-Beta Heuristic (AIM-030), Massachusetts Institute of Technology.

[11] "Extreme Programming", USFCA.edu, lecture paper at http://www.cs.usfca.edu/~parrt/course/601/lectures/xp.html.

[12] Feathers, M. Working Effectively with Legacy Code, Prentice Hall, 2004

[13] "Human Centred Technology Workshop 2005", 2005, PDF webpage: ftp://ftp.informatics.sussex.ac.uk/pub/reports/csrp/csrp585.pdf .

[14] Kan-Yueh Chen, The Development of the Multi-Broker Desktop Grid for Computer Games, National Chiao-Tung University, master thesis

[15] Lee Copeland (December 2001). "Extreme Programming" . Computerworld. Retrieved January 11, 2011.

[16] Newkirk, JW and Vorontsov, AA. Test-Driven Development in Microsoft .NET, Microsoft Press, 2004.

[17] Renlib, Renju – A Renju Editor, available at http://www.renju.se/renlib/.

[18] Smarr, L., Catlett, C., Metacomputing, Communications of the ACM, v.35 n.6, p.44-52, June 1992.

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

[20] Trygve Reenskaug, Models-Views-Controllers, 1979

[21] 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.

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

[23] 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.

[24] 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, presented at the Int. Conf. Comput. Games Kanazawa, Japan, 2010.

[25] 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).

[26] XtremWeb website. avalible at http://www.xtremweb.net/.