2017 DSD Final Report

資工19陳品媛 104062121

主題 Tic-tac-toc + Data I/O

Tic-tac-toc

- 單機版的OOXX，player可與AI對戰，AI設計成不會輸

Data I/O

- Input
  - 玩家回答遊戲進行流程
  - 玩家要mark的位置
- Output
  - 每一手之後的遊戲棋盤狀態
  - 在每一回合結束後，會依據遊戲最後輸贏結果，透過三種不同的image（win，lose，tie），輸出他們的ascii art

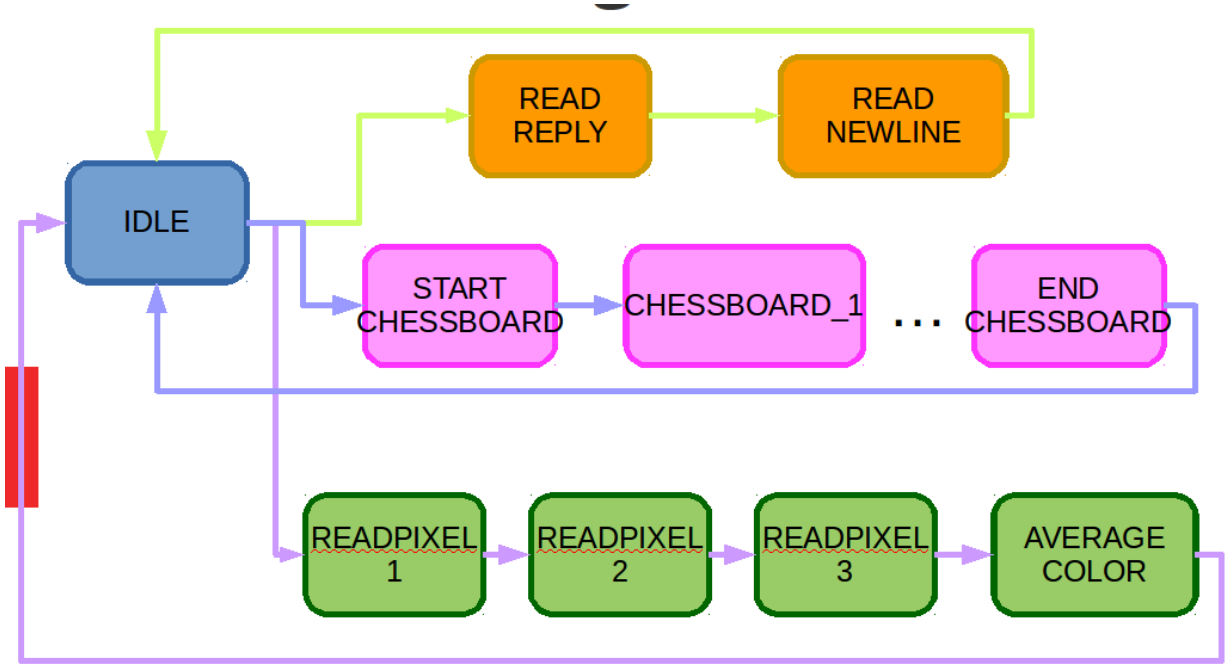
架構細節

軟體

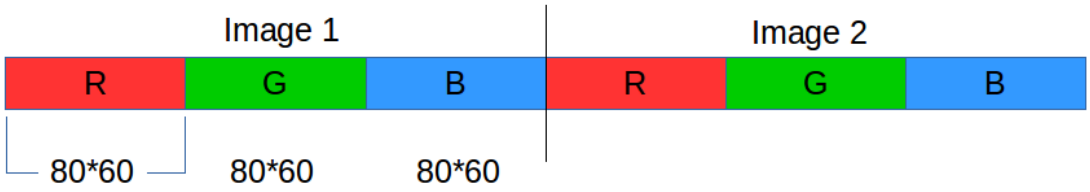
- 控制遊戲主要流程
  1. 開始遊戲
  2. AI與玩家之間輪流下棋
  3. 印棋盤狀況
  4. Gameover，印ascii art
  5. 是否開啟下一局
- Error handler
  1. Player回答是否落在範圍之中
  2. Player mark是否已有做標記
- AI演算法
  1. Minimax (link)
  2. Alpha-Beta-Pruning (link)
  3. 加速AI先手第一棋
     - 由於在所有的格子的勝率是一樣的，所以我讓他隨機下(tick( ))，不再讓他跑minimax，因為一開始的時候，樹是最大的，會花很多時間

硬體

- State diagram



- Read player reply
  - sub state
    1. READREPLY
    2. READNEWLINE
  - 由於遊戲的設計，每次的輸入可能會是0~9的數字，但會有Enter鍵，因此在吃完換行符號，才做反應，所以需要兩個state
- Check game status
  - sub state
    - STARTCHESSBOARD
    - CHESSBOARD_(1~5)：從software讀所需要的data, e.g. 棋盤狀態$\times 9$, 棋子數目$\times 1$，共十個數字從軟體傳到硬體
    - ENDCHESSBOARD：檢查八條連線，看有無勝負，並回傳status
  - 在下第一步棋時，由於棋盤可能局面很多種，導致minimax在遞迴的部份會花很多時間，所以我將**minimax**確認遊戲是否結束的部份抽出來給硬體做
- Ascii art
  - sub state
    - READPIXEL(1~3)：分別依序讀R, G, B
    - AVERAGECOLOR：求其平均值，並回傳適當的character，讓software印出
  - 原理
    - 有10個character做表示, e.g. $, #, @...
    - 將RGB三個channel做平均，依據平均過後的值除以(256/10)，看其落在哪個區間，選擇用哪個character表示
  - Python前處理
    - image全部resize到80×60
    - From jpg to binary file：利用python skimage讀圖檔，並以row major的方式寫到binary file內



- image to ascii art成果

測試與分析

Tic-tac-toc

由於minimax是用遞迴的把局面跑過一次，這棵樹會長得很廣，會有9!種局面。

若是AI的部份全部交由軟體處理，在現在軟體的環境中，CPU跑得很慢，所以會導致AI在前面幾手時，計算時間很久，第一手約需要3分鐘左右。因此我將其中的一部分拉出來給硬體實現，也就是Check game status，他需要檢查八條可能的線，而且他會在每次遞迴中就要做一次，判斷是否有結果產生，所以像這個繁瑣的事情交由硬體做加速。

實驗結果

在玩家下完一步後，接著換AI下一手的實驗結果

- software 5804
- hardware 343

Ascii art

- 由於圖片是80*60，所以在讀取pixel，與計算平均值時給硬體做，會達到加速的目的

困難與解決方法

困難

我一開始先實作了軟體版本的AI，一開始我只用了Minimax，在自己電腦的環境可以看到即時性，但是我在picorv32的環境跑時，應該花了有5分鐘以上，所以時間上的加速是我首要問題。

解決方法

我因為在網路上尋找minimax的過程中，有看到關於alpha-beta pruning的優化，所以之後加入了進來，但是時間還是要兩至三分鐘。

之後在詢問助教的過程中，助教提到將部份的拿進硬體做加速，所以我想到將check game status的部份拿給硬體，加速到20s左右。

心得

我原本只是想要做Data I/O，做image的加速，想說再搭配個小遊戲做輔助開頭，完成我這次的final。

意外的是，cpu的環境真的太慢，所以必須將AI部份的運算挪到硬體，想如何拉出那一塊給硬體是件有趣的事，而且之後明顯看到他的時間加速，真的很有成就感。

另外之前沒有接觸過tic-tac-toe AI的演算法，所以算是除了學習到軟體硬體的切割與加速以外，學到的一個有趣的演算法。