探索遊戲的人工智慧：從 tic tac toe 到 AlphaGo

研究者：吳宇倫

指導老師：鄭綺瑩老師

壹、 緒論

一、研究動機

我喜歡下圍棋。圍棋是一種規則簡單，但組合很多的一種棋類運動。棋盤擺設的可能性大約有 361!（階乘） / 161!（階乘）種（不包括「吃子」），地球上的原子數也只有 10^50 個，棋盤擺設的可能性竟然比可觀測宇宙的原子數多！人類當初認為電腦在圍棋的方面應該無法戰勝人類，但是在 2016 年 3 月時，DeepMind 研發出的 AlphaGo 竟然以 4:1 戰勝世界棋王李世石！我精通 Scratch，也會一點 Python，就在想能不能用 Python 和 Scratch 寫出在玩 tic tac toe 時不會輸的 AI，甚至是很會玩西洋棋的 AI。

二、研究目的

1. 了解玩遊戲的 AI 的邏輯和如何製作。
2. 用 Python 做出可以在電腦上玩的棋類遊戲（人 vs 人可以玩即可，例如 tic tac toe、五子棋……）
3. 用 Scratch 做出可以在電腦上玩的棋類遊戲（人 vs 人可以玩即可，例如 tic tac toe、五子棋……）
4. 用 Python 做出會玩桌遊的 AI
   (1) 用 Python 做出做出會玩 tic tac toe 的 AI
   (2) 進階挑戰：用 Python 做出會玩西洋棋的 AI

貳、 文獻探討

一、 Minimax 介紹

Minimax 用於兩人互相較量的遊戲中，會假設對方會選最好的一步，來告訴 AI 要下哪裡。Minimax 是一種演算法，是利用 recursion(遞迴)做出來的，所以跑很慢。為了讓速度更快，所以利用 alpha-beta-pruning，把一些可能性刪掉。

二、 Python

Python 是用 C 語言寫出來的，是一種高階語言。Python 便於閱讀，比較容易學會。Python 和其他程式語言比起來，也比較簡潔，壞處是因為它和電腦比較遠，所以跑得和其他程式語言比起來慢很多。

三、 Scratch

Scratch 是用 Javascript 寫出來的。和其他程式語言不一樣的是，其他程式語言是用打字的方式，而 Scratch 是用拖拉積木的。Scratch 適合沒學過寫程式的人，可以很快的學會，也可以比較容易看懂。但是，Scratch 的限制也很多，有一些事情不能做，也不像 Python 會告訴你哪裡有問題。

參、 研究方法及步驟

一、研究方法

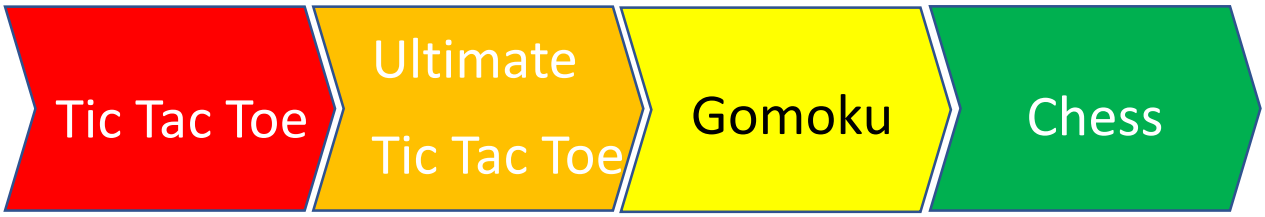
(一) Python 篇

利用 PyGame 做出遊戲畫面，利用 PyCharm 和 Replit 寫程式並測試 AI，並利用 minimax 演算法。

(二) Scratch 篇

利用 Scratch 做出遊戲畫面、寫程式。

二、研究步驟



肆、 製作歷程

我在製作的時候遇到許多問題和 Bug，我有語法上的問題的話，我會先試著自己解決，如果想不到會尋求爸爸的幫助或上網查。

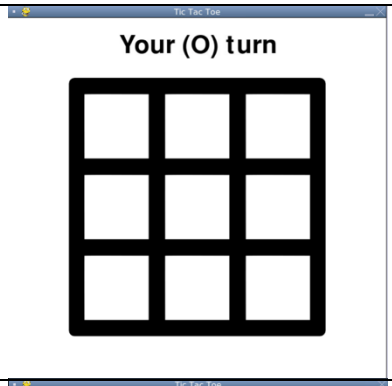
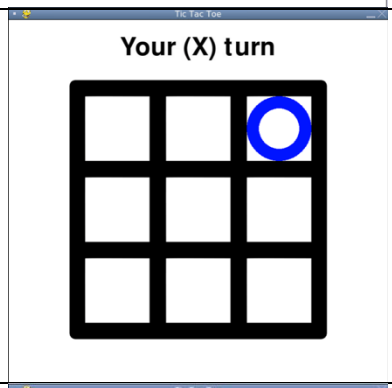
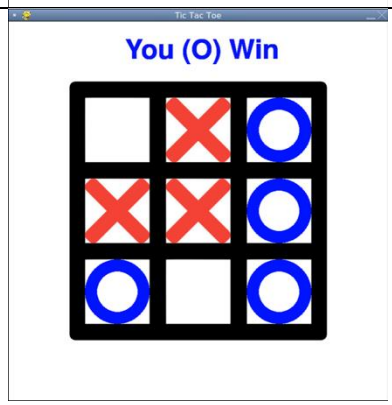
本來想利用 Scratch 製作 AI，但因為後來發現 Scratch 的功能比 Python 少很多，很難製作 AI，所以就放棄了，只有用 Scratch 製作遊戲畫面。

伍、 製作結果

一、Python 遊戲

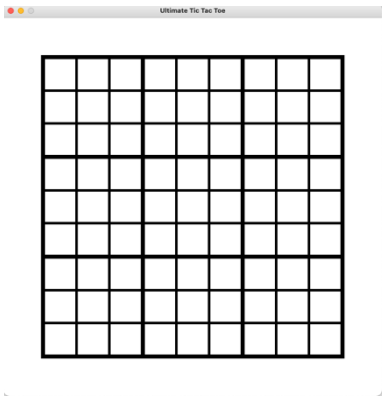
(一) Tic Tac Toe

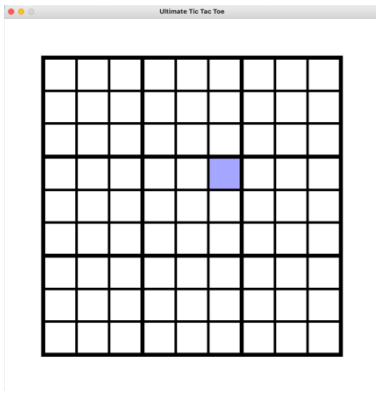
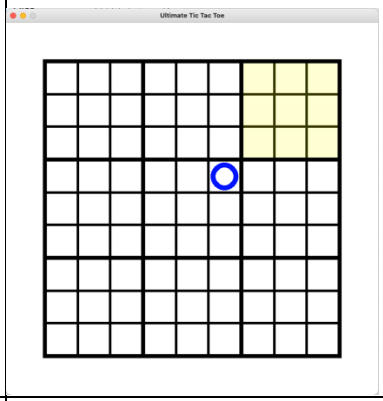
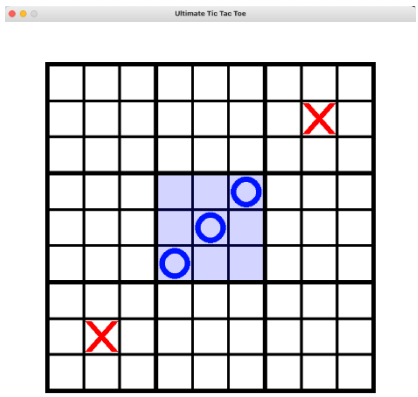
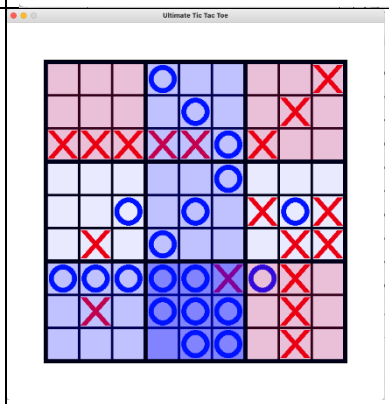
作品介紹

初始畫面 點擊白色方塊來選 O/X 要放的位置 每局 O 和 X 會輪流當第一個下的人。如果是 “Your（O）turn”就代表現在輪到 O，如果是 “Your（X）turn”就代表現在輪到 X。	
點擊以後自動切換輪到的人	
如果 O 贏了，會顯示"You(O) Win";如果 X 贏了，會顯示"You(X) win";如果平手，會顯示" Draw"。	

（二）Ultimate Tic Tac Toe

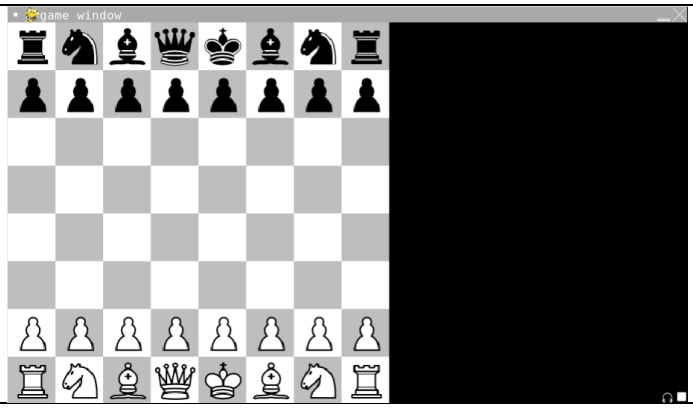
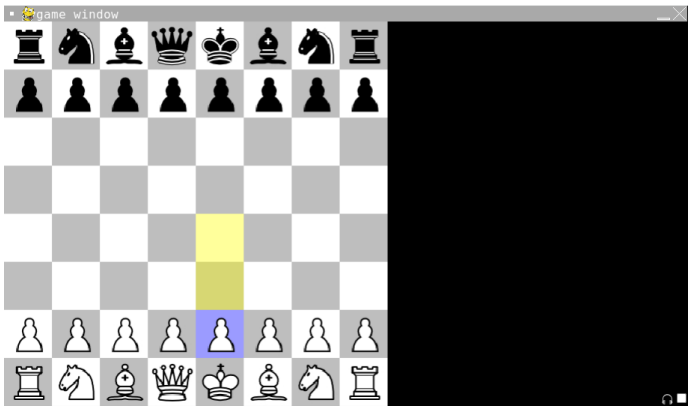
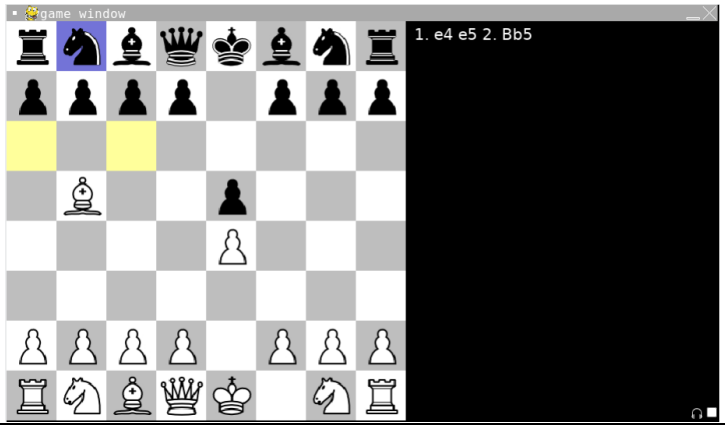
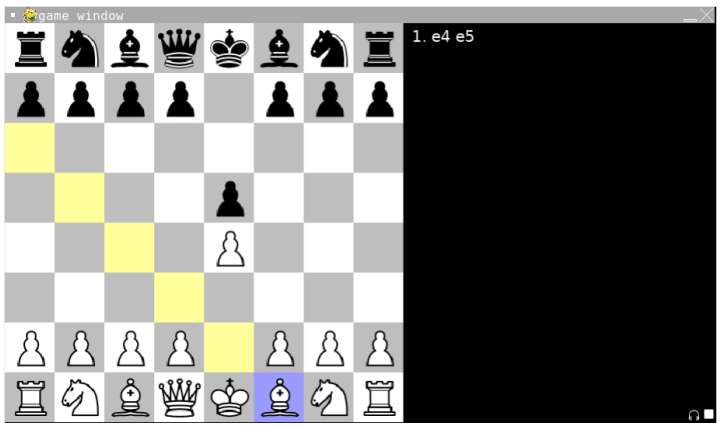
作品介紹

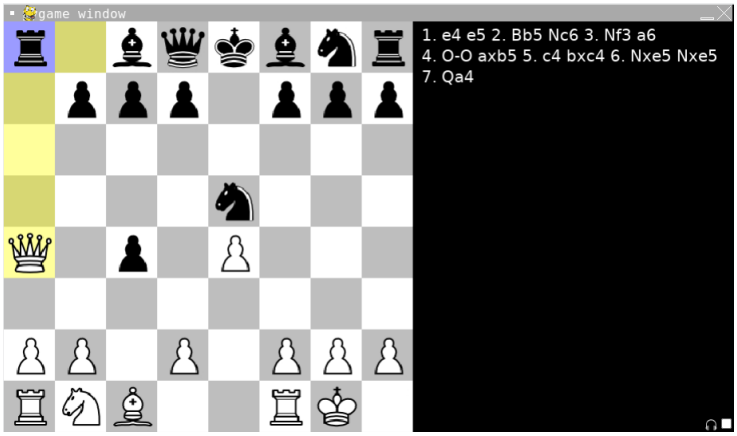
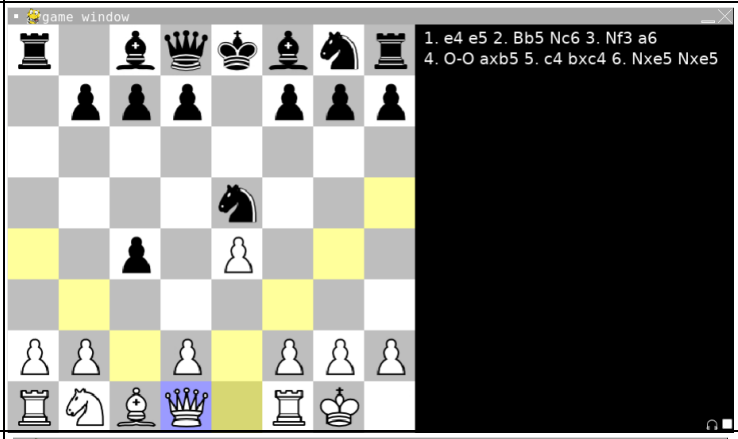
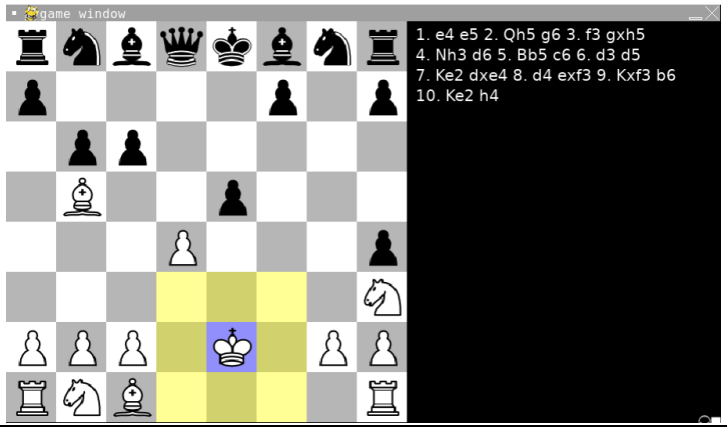
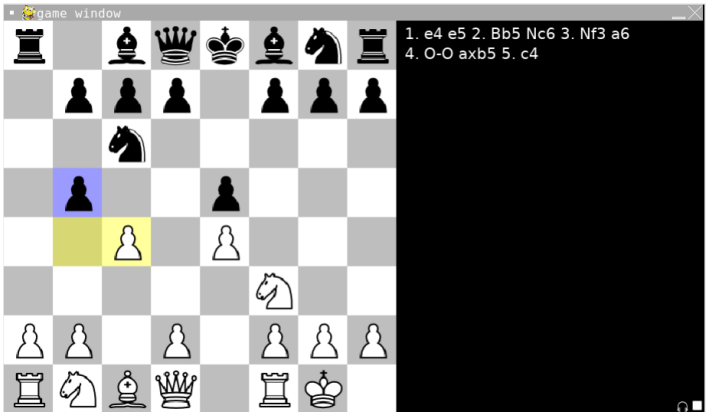
初始畫面，點選白色的方格來選擇 O/X 要放置的位置。	

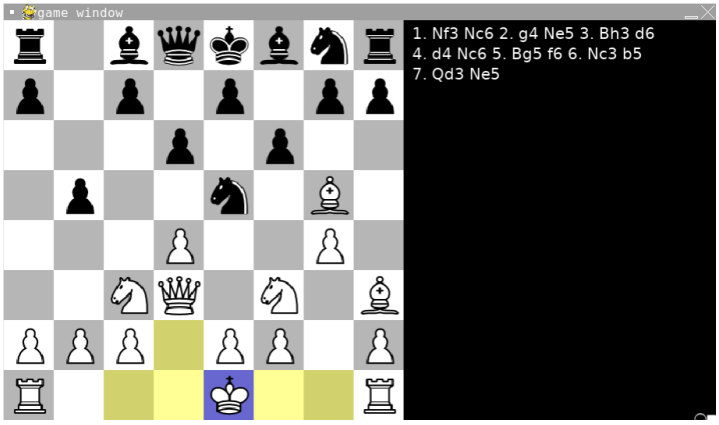
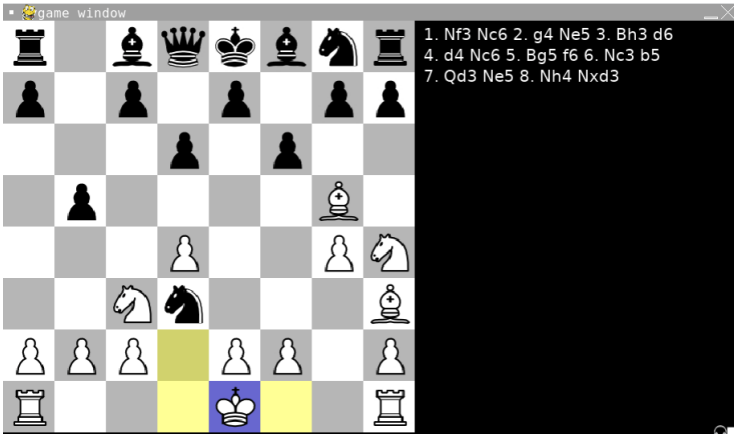
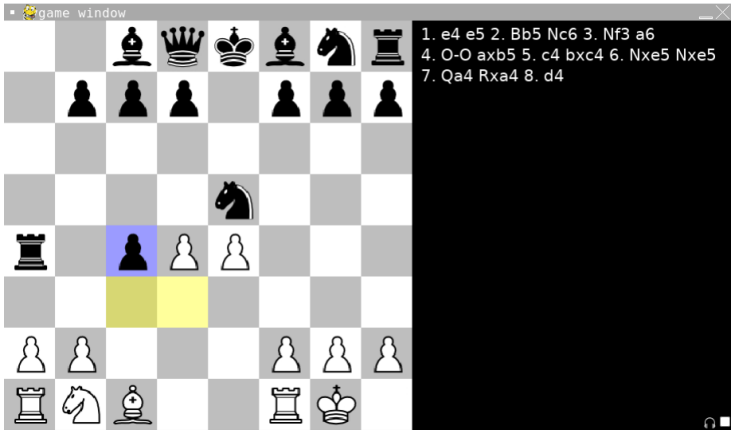
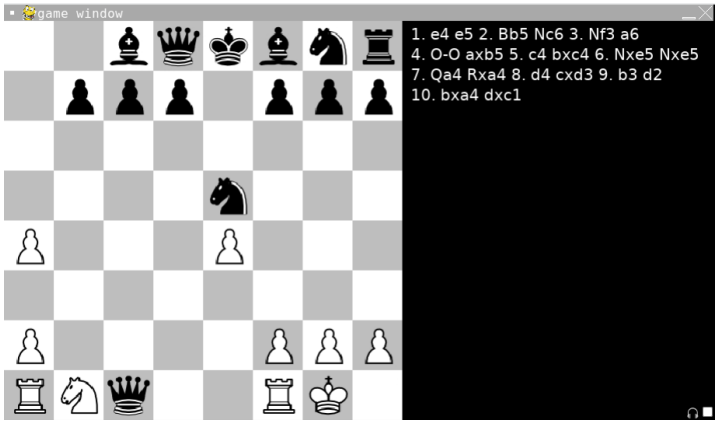
滑鼠移到白色放個上方，如果那格可以放置 O/X，他會顯示淡淡的藍色/紅色。如果那格不能放置 O/X，就不會有顏色。	
黃色方格代表可以放置 O/X 的範圍，如果沒有黃色方格就代表每一個地方都可以放置。	
如果 O/X 在其中一個小九宮格中贏了，那個九宮格就會變成淡淡的藍色/紅色。	
如果 O/X 在大九宮格中贏了，整個棋盤會變成超級淡的藍色/紅色。	

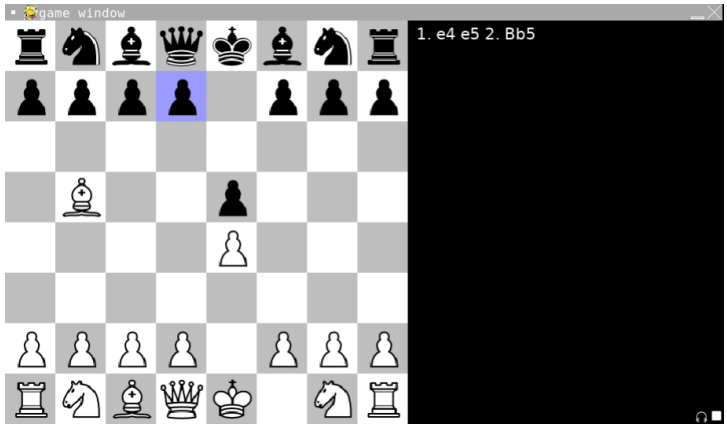
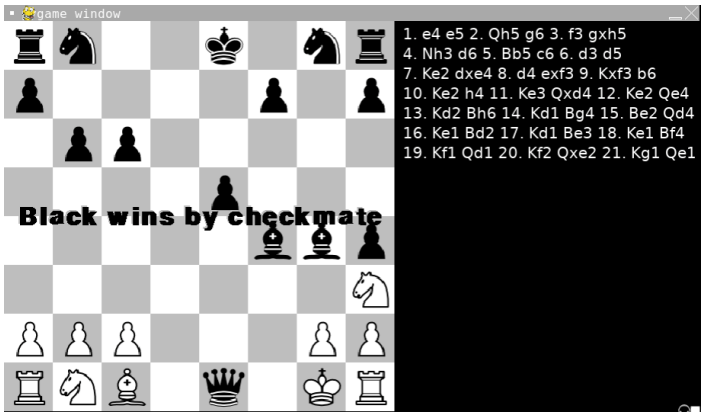
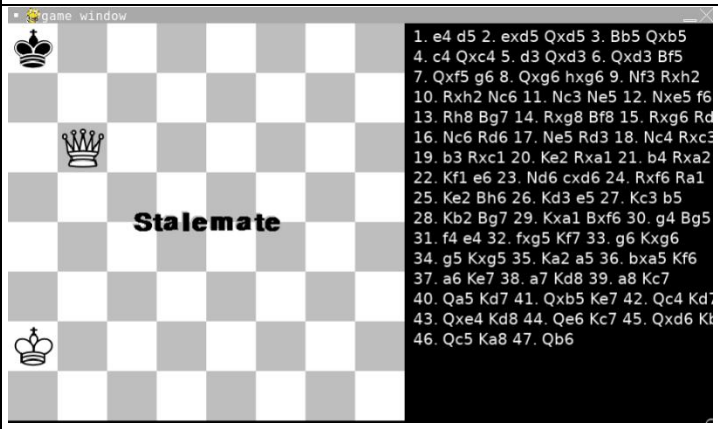
（三）西洋棋

作品介紹

初始畫面，點選棋子再點選要到的格子就可以下棋。白棋先，接下來輪流下棋。右方紀錄下棋的紀錄。按鍵盤上的 z 可以復原。	
點選棋子，會把你點選的方塊塗成藍色，會把可以去的地方塗成黃色，點選其他地方可以取消選取。右圖顯示兵可以去的地方。	
右圖顯示騎士可以去的地方。	
右圖顯示主教可以去的地方。	

右圖顯示城堡可以去的地方。	
右圖顯示皇后可以去的地方。	
右圖顯示國王可以去的地方。	
右圖顯示兵的斜吃。	

右圖顯示兩邊的國王入堡。	
右圖顯示被將軍不能入堡，移動過也不能入堡（無法展示）。	
右圖顯示「吃過路兵」。	
右圖顯示「兵的升變」（c1 的兵自動升級為皇后）。	

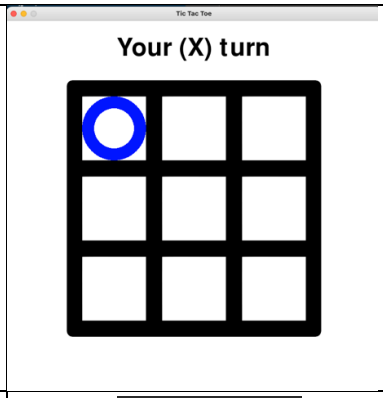
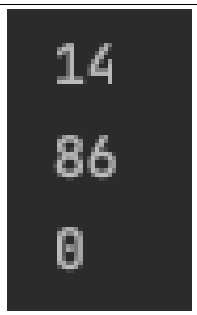
如果該棋移動會造成國王被將軍，則該棋不能移動。	
右圖顯示將死。	
右圖顯示和局。	

二、Python AI

（一）Tic Tac Toe

1. 作品介紹（部分與前面的 Tic Tac Toe 相同）

設定模式（human vs human/human vs random/human vs medium/human vs computer/random vs computer/medium vs computer）	`mode = 'human vs human'`

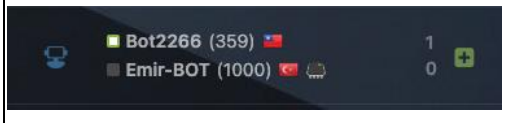
computer 會自動下出最好的一步（用 minimax 演算法） medium 也會下出最好的一步（非自行判斷），但是有一種情況會輸。 random 會下出隨機的一步	
AI 互打模式（random vs computer/medium vs computer）第一個數字代表平手的次數，第二個代表 computer 贏的次數，第三個代表 computer 輸的次數	

2. AI 互打紀錄

Smart Toe 1.0 vs random : 64 勝 33 敗 3 和
Smart Toe 1.0 vs medium 1.0 : 18 勝 82 敗 0 和
Smart Toe 1.0 vs medium 2.0 : 2 勝 98 敗 0 和
Smart Toe 2.0 vs medium 2.0 : 23 勝 977 敗 0 和
Smart Toe 2.0 vs random : 724 勝 256 敗 3 和
Smart Toe 3.0 vs medium 2.0 : 53 勝 0 敗 47 和
Smart Toe 3.0 vs random : 85 勝 0 敗 15 和
Smart Toe 1.0 ~ 3.0 的勝率逐漸提高

（二）西洋棋

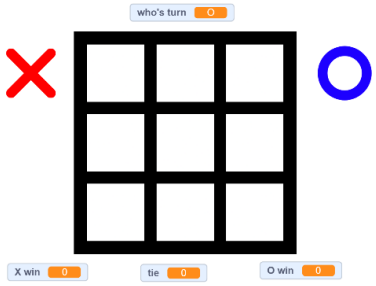
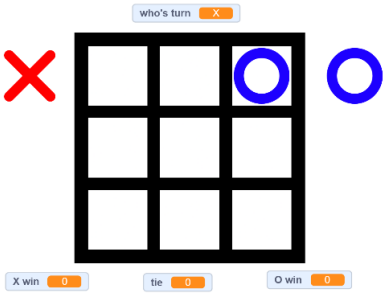
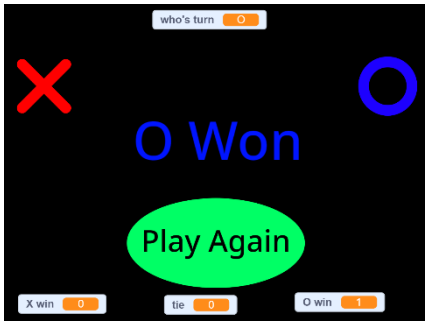
作品介紹（部分與前面的 Chess 相同）

如果白棋（playerOne）設為 True，代表人要當白棋，如果是 False 就代表是電腦要當白棋。黑棋（playerTwo）也一樣。	
設定深度（越大電腦越聰明，但執行速度也越慢）	
戰績（沒錯，這是 AI 下的）我的 AI 是 Bot2266（我的帳號）戰勝 Elo 1000 的機器	

三、Scratch 遊戲

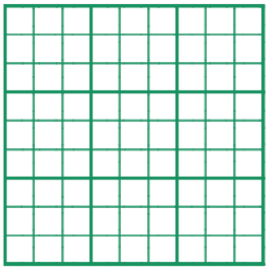
（一）Tic Tac Toe

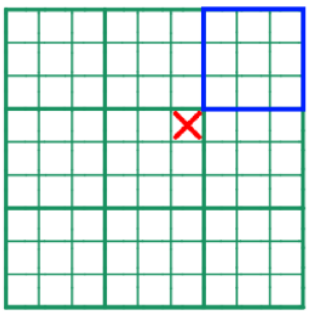
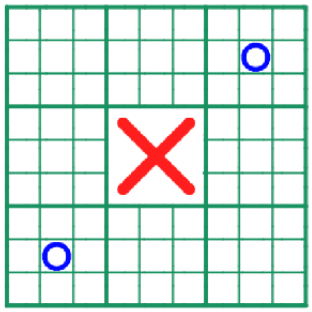
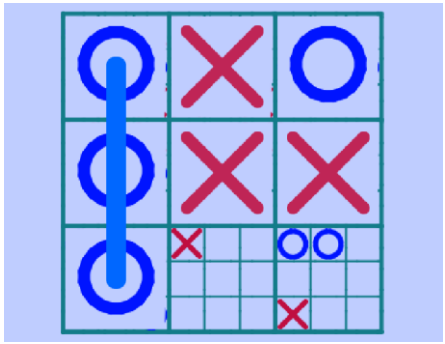
作品介紹

初始畫面 點擊白色方塊來選 O/X 要放的位置 每局 O 和 X 會輪流當第一個下的人。	
點擊以後自動切換"who's turn"變數	
如果 O 贏了，會顯示"O won"，並把"O win"的變數增加 1;如果 X 贏了，會顯示"X won"，並把"X win"的變數增加 1;如果平手，會顯示" It's a tie "，並把"tie"的變數增加 1	

（二）Ultimate Tic Tac Toe

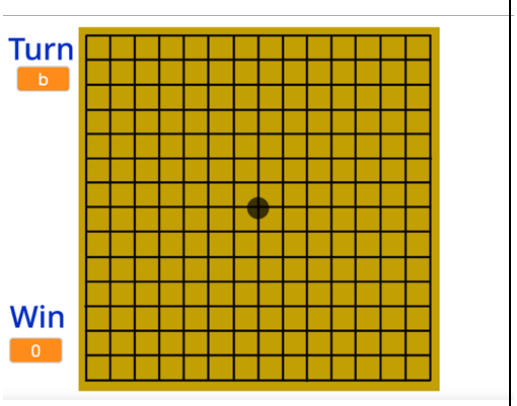
作品介紹

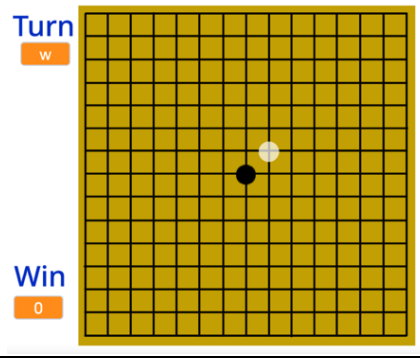
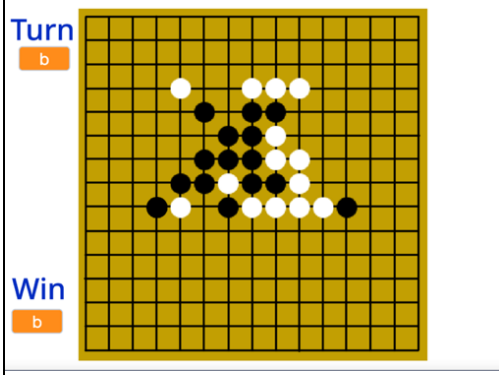
初始畫面，點選白色的方格來決定 O/X 要放的位置。	

方框表示 O/X 一定要放在方框中。(藍色代表現在輪到 O，紅色代表現在輪到 X)	
O/X 在其中一個九宮格中勝利會在那一格畫上一個大 O/X，沒有方框代表每個空方格都可以放置 O/X。	
如果 O/X 在大九宮格中勝利，會將整個畫面設為藍色/紅色，如果平手會變成灰色。	

（三）五子棋

作品介紹

初始畫面，如果接近其中一個下棋的點（交界點）就會有淺淺的棋子出現在哪個點上。	

如果確定要下棋，就點選你要下棋的點，Turn 會切換成另一邊。	
如果其中一方贏了，Win 就會設為贏得一方，並停止程式。	

四、作品彙整

1. https://replit.com/@JamesWu13/Tic-Tac-Toe-AI-alpha-beta#main.py
2. https://replit.com/@JamesWu13/Ultimate-Tic-Tac-Toe#main.py
3. https://replit.com/@JamesWu13/Chess#Chess/ChessMain.py
4. https://scratch.mit.edu/projects/562342230/
5. https://scratch.mit.edu/projects/564905886/
6. https://scratch.mit.edu/projects/659252993/

陸、 心得與建議

在製作的時候雖然遇到很多挫折，絞盡腦汁都想不出來，但爸爸一來看就知道問題在哪，雖然會覺得不應該浪費那些時間來思考，直接請他幫忙就好了，但其實過程才是最重要的一個步驟，在過程中的思考和解決才能讓我有最大的進步。我很開心我能一直堅持到現在，最後做出來也覺得真開心。建議學弟妹可以利用你的興趣當作獨立研究的題目。興趣就是持續的關鍵，只要你很喜歡，就可以一直做下去。這次很幸運的，雖然停課，但因為我是做和寫程式相關的研究，所以不會沒辦法做實驗。我在製作獨立研究的時候學到原來 Python 有 Pygame 這種有趣、好玩的套件，還有原來 Python 也可以在 Replit 這個網站線上編輯，而且也可以在 PyCharm 這個好用的軟體寫程式。我不會後悔做這份題目，甚至希望以後如果有時間可以繼續製作跟 AI 有關的研究。製作這種題目讓我得到很大的成就感，祝學弟妹能在製作獨立研究的時候一帆風順。

柒、參考資料

1. Ultimate tic-tac-toe。摘自 https://en.m.wikipedia.org/wiki/Ultimate_tic-tac-toe
2. Pygame Tutorial for Beginners - Python Game Development Course。摘自 https://www.youtube.com/watch?v=FfWpgLFMI7w
3. Tic-tac-toe-minimax。摘自 https://github.com/Cledersonbc/tic-tac-toe-minimax
4. Minimax Algorithm in Game Theory | Set 3 (Tic-Tac-Toe AI – Finding optimal move)。摘自 https://www.google.com/amp/s/www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-3-tic-tac-toe-ai-finding-optimal-move/amp/
5. Minimax Algorithm in Game Theory | Set 4 (Alpha-Beta Pruning) 。摘自 https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/
6. History of Python。摘自 https://en.wikipedia.org/wiki/History_of_Python
7. Scratch (programming language)。摘自 https://en.wikipedia.org/wiki/Scratch_(programming_language)#History
8. Creating a Chess Engine in Python。摘自 https://www.youtube.com/watch?v=EnYui0e73Rs&list=PLBwF487qi8MGU81nDGaeNE1EnNEPYWKY_