一、設計動機

我們希望設計一個老少咸宜，可以鍛鍊思考的遊戲。對於家中長輩來說，這款操作簡單的遊戲能有效防止老年癡呆；對於剛接觸算數的兒童，則可以在遊戲之中訓練簡單加法。

二、需求分析

（一）完成能實現 2048基本操作之遊戲介面。
（二）遊戲介面清楚易操作，並添加音效等增進使用者體驗的功能。
（三）完成能獨立運作的AI，且達成2048成功率80%以上。

三、名詞定義

（一）一般方塊：被置於面板上的物件，為遊戲的操作主體，可根據操作在面板上移動、合併。後續無特別說明的方塊皆屬此類。
（二）障礙方塊：僅出現在困難模式中的特殊方塊，可移動但無法合併。
（三）(真數)數值：一般方塊皆具有的屬性，會顯示在方塊的圖形介面上。數值皆為2的正整數冪次。後續若無特別說明的「數值」皆屬此類。
（四）對數數值：方塊的對數數值 = log(真數數值, 2)，有特別申明的才屬此者。
（五）合法移動：能使版面產生變化的移動操作。
（六）決策：觀察或運算當前盤面所得出的最終操作。

四、遊戲規則

（一）一般規則

１、遊戲使用方向鍵操作，使4乘4面板中的方塊移動。
２、方塊數值皆為2的冪次，數值相同的兩方塊若在移動中碰撞，則兩方塊合併為一方塊，其數值為兩方塊相加(即原方塊數值的兩倍)。
３、每次合法移動過後，程式將在面板上空曠處添加數值為2或4的新的方塊，數值2與4的出現機率分別為90%與10%。
４、若面板不存在任何合法移動(即面板全滿、相鄰方塊數值皆不同)，則遊戲結束。

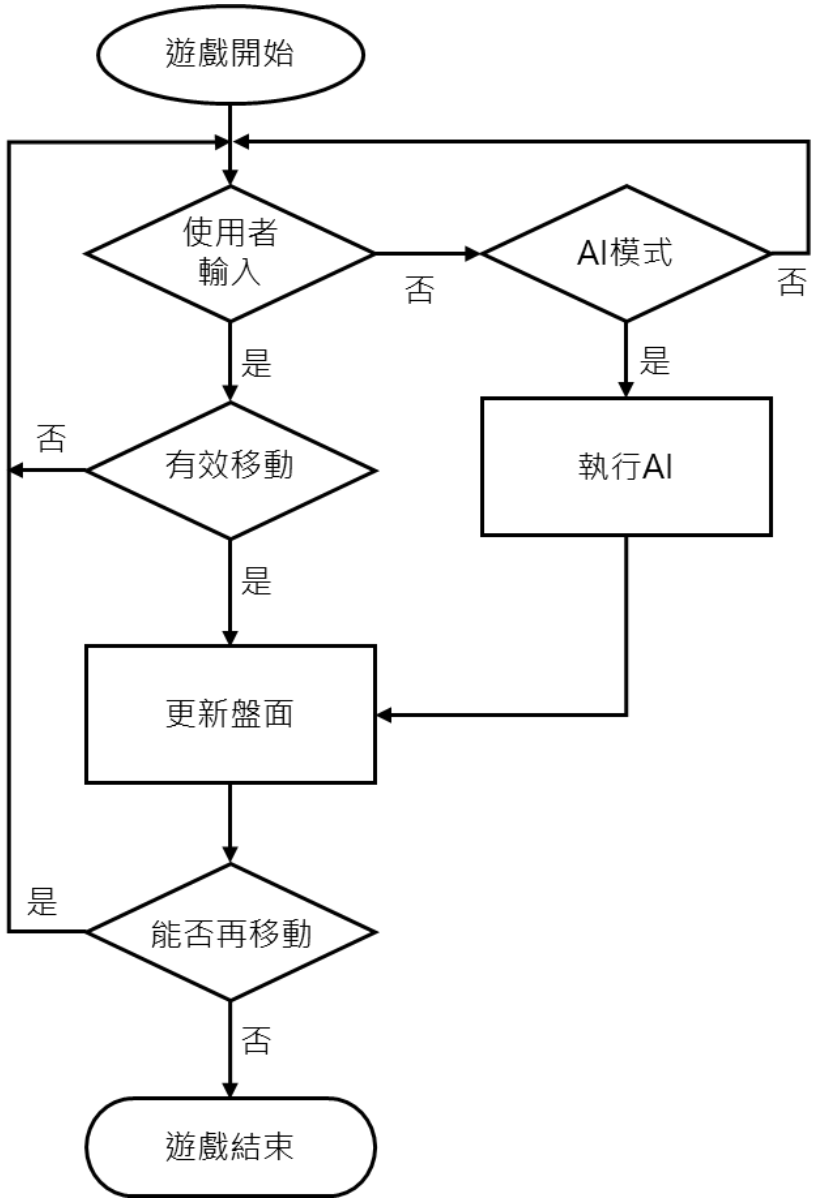
（二）困難模式

遊戲將在開始時添加一塊可移動、無法合併之障礙方塊作為干擾，以增加遊戲難度。

五、計分方式

遊戲初始分數為0，每次合併所產生的新數值會加入分數。例：某次合法操作使兩塊數值為2的方塊合併成一塊數值為4的方塊，則此次操作可獲得4分。

六、程式規劃與流程



七、AI演算法

（一）AI_0

１、運算方式

（１）運算步驟

a.隨機抽樣：對一個盤面先選定一個合法移動方向作為第一步，第二步之後皆隨機移動直到遊戲結束，並記錄結束時的分數。

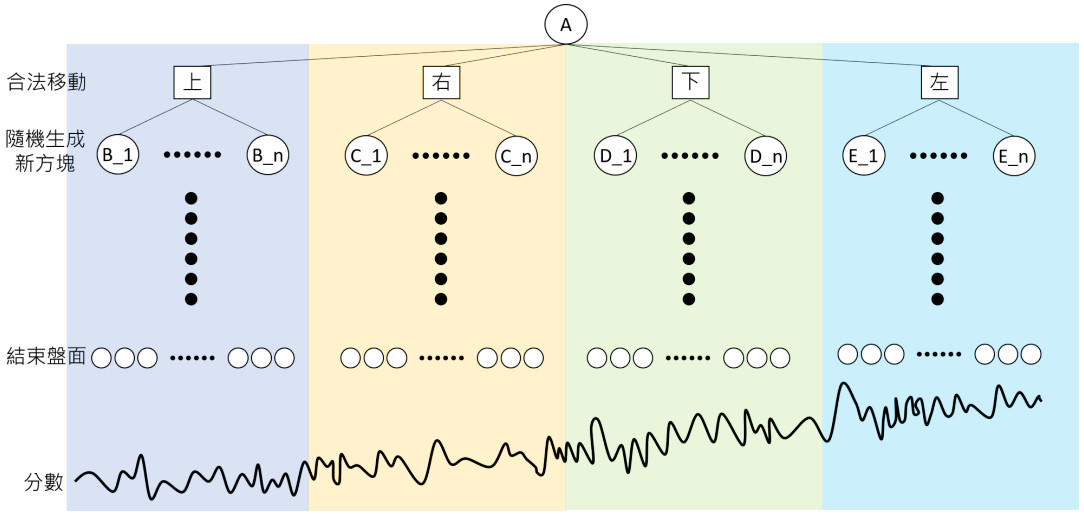
b.統計：對每個合法移動方向都分別做ROUND次上述的隨機抽樣，計算每個方向隨機抽樣的平均分數。(ROUND常數預設為100)

c.完成決策：將隨機抽樣平均分數最高的移動方向作為此盤面的決策結果。

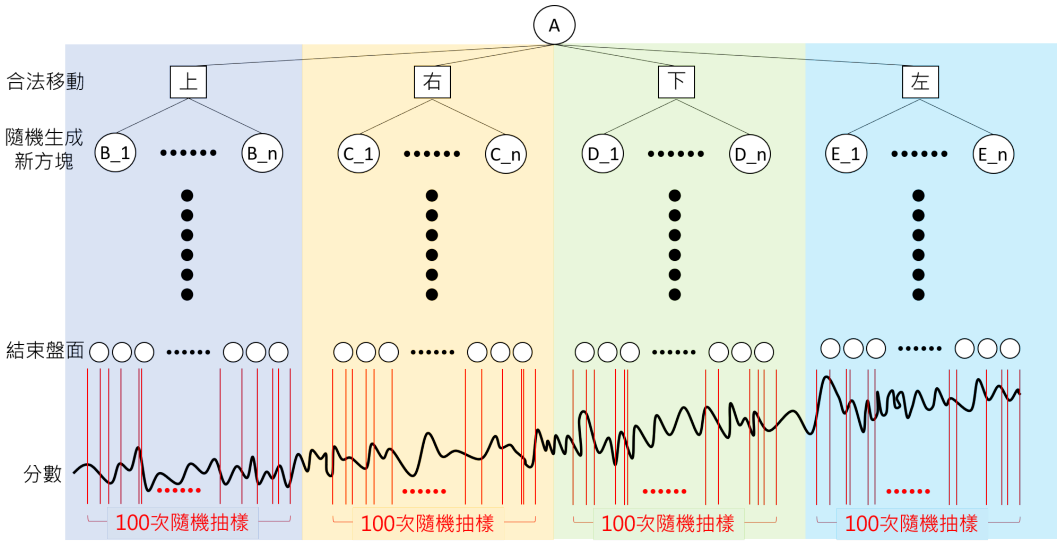
（２）舉例

假設對某盤面A而言，上、右、下、左四個移動方向皆為合法移動。且第一步向上，之後隨機走到遊戲結束100次的平均分數為1000；第一步向右，之後隨機走到遊戲結束100次的平均分數為2000；第一步向下，之後隨機走到遊戲結束100次的平均分數為3000；第一步向左，之後隨機走到遊戲結束100次的平均分數為4000。則盤面A的決策結果為向左移動，因為第一步向左的平均分數最高。

2、算法原理



由於當前決策(圖中上、右、下、左方框)會影響後續每個盤面的發展，所以當前決策的好壞會影響結束盤面的分數高低。因此可用結束盤面的分數高低，反推較好的移動方向做為當前決策結果。以圖中所繪之分數高低分布為例，若知道四個方向對應的結束盤面分數高低，就能推測出當前最佳決策應為向左。



然而此遊戲的可能盤面數量會隨著移動步數指數上升，即使用電腦也無法在短時間內算完所有可能。故在此使用隨機抽樣的方法，用少量的樣本，大致決定最終分數的好壞，也就能反推出最有可能的最佳決策。以圖中所繪之分數高低分布為例，對當前決策的四個方向各做100次隨機抽樣，向左移動得出的平均分數普遍高於其餘三者，向左移動即為最佳決策。

根據參考資料，此算法可算是廣義的蒙地卡羅演算法，若以蒙地卡羅的角度切入，更能顯示此算法的合理性。

3、算法測試

測試以intel i5-8300H CPU、8G RAM之windows 10電腦，python 3.6.8 版在cmd中執行程式(由於測試並非由labview啟動python程式，以下時間數據皆未計算labview-python間呼叫與傳遞的耗時)，完整進行500場遊戲。(剩餘三種算法的測試軟硬體規格、環境、方式皆與此相同，後續不再贅述)

AI_0測試-基本數據表

最大方塊 數值區間	遊戲平均 分數(分)	遊戲平均 步數(步)	遊戲平均 耗時(sec)	每步平均 耗時(sec)
512 - 4096	35308.784	1782.338	439.287	0.246

AI_0測試-最大方塊分布表

最大方塊數值	512	1024	2048	4096	總計
次數(次)	10	83	313	94	500
百分比(%)	2	16.6	62.6	18.8	100

4、優缺分析

（１）優點：邏輯直觀、簡單，程式碼實踐容易。

（２）缺點：計算耗時高，且運算資源分配不合理，資源運用效率低。

下方左圖盤面簡單，決策精度對遊戲生死的影響較小；右圖盤面複雜，決策精度對遊戲生死影響較大。然而左邊盤面遊戲進度較右邊慢，玩到遊戲結束所需的時間比右邊多，即左側運算一步消耗的資源比右側高。所以這個算法會造成「消耗過量資源在低重要性盤面，使用過少資源在高重要性盤面」的現象。

（３）改善方法：

每次隨機抽樣的數量改以時間限制，替代原先的固定次數，此即為AI_1的運算方式。

（二）AI_1

１、運算方式

（１）運算步驟

a.隨機抽樣：對一個盤面先選定一個合法移動方向作為第一步，第二步之後皆隨機移動直到遊戲結束，並記錄結束時的分數。(與AI_0相同)

b.統計：對每個合法移動方向都分別做上述的隨機抽樣 [TIME_LIMIT/合法移動方向個數] 秒，計算每個方向隨機抽樣的平均分數。(TIME_LIMIT常數預設為AI_0每步平均耗時0.246秒)

c.完成決策：將隨機抽樣平均分數最高的移動方向作為此盤面的決策結果。(與AI_0相同)

（２）舉例

假設對某盤面A而言，僅上、下、左 3個移動方向為合法移動。則每個方向的運算時限為 AI_0每步平均耗時 / 合法移動方向次數 = 0.246/3 = 0.082 (秒)。則程式會不斷執行向上的抽樣(第一步向上，之後隨機走到遊戲結束)直到向上的總時間超過0.082秒；之後不斷執行向下的抽樣(第一步向下，之後隨機走到遊戲結束)直到向下的總時間超過0.082秒；最後不斷執行向左的抽樣(第一步向左，之後隨機走到遊戲結束)直到向左的總時間超過0.082秒。最後分別算出每個方向的平均結束分數，分數最高的方向即為決策決果。

２、算法原理

（１）抽樣方式

AI_1的抽樣方式與AI_0相同，所以不再贅述其原理。

（２）抽樣數量

AI_1的抽樣數量是浮動的。再度以AI_0優缺分析的例子說明。下方左邊盤面重要性較低，但單次抽樣耗時較高；右邊盤面重要性較高，但單次抽樣耗時較低。

根據大數原理，抽樣數越多，越能貼近原始資料的情況，也就能做出越精準的決策。若以時間作為抽樣數量的限制，則單次抽樣耗時高的盤面抽樣數量較低，決策精度較低，恰適用於低重要性的盤面；單次抽樣耗時較低的盤面，抽樣數量較高，決策精度較高，恰適用於高重要性的盤面。用這個方法能使運算資源平均分配，改善AI_0資源不均的問題。

3、算法測試

AI_1測試-基本數據表

最大方塊 數值區間	遊戲平均 分數(分)	遊戲平均 步數(步)	遊戲平均 耗時(sec)	每步平均 耗時(sec)
512 - 4096	43374.712	2128.132	545.720	0.256

AI_1測試-最大方塊分布表

最大方塊數值	512	1024	2048	4096	總計
次數(次)	3	41	285	171	500
百分比(%)	0.6	8.2	57	34.2	100

4、優缺分析

（１）優點：邏輯直觀、簡單，程式碼實踐容易，且改善AI_0資源分配不均的問題。

（２）缺點：

a.計算耗時略比AI_0高，並未解決AI_0計算耗時高的問題。

b.此算法的結果與硬體算力有關，算力越弱，單次抽樣耗時越高，則在固定時間內的抽樣數量越少，決策結果較差。

（３）改善方法：更換演算法。

（三）AI_2

１、運算方式

由於AI_0、AI_1都有無法突破的時間瓶頸，AI_2將改用minimax演算法，以下為其步驟。

運算步驟：

a.盤面評估函數：對不同盤面，我們用以下特徵為每個盤面評分。

(a) 平滑度：所有鄰近方塊的對數數值差的絕對值的總和(鄰近方塊係指兩個不受其他方塊分隔的同行或同列方塊)。此項目會使數值接近的方塊靠近，增加合成機會。

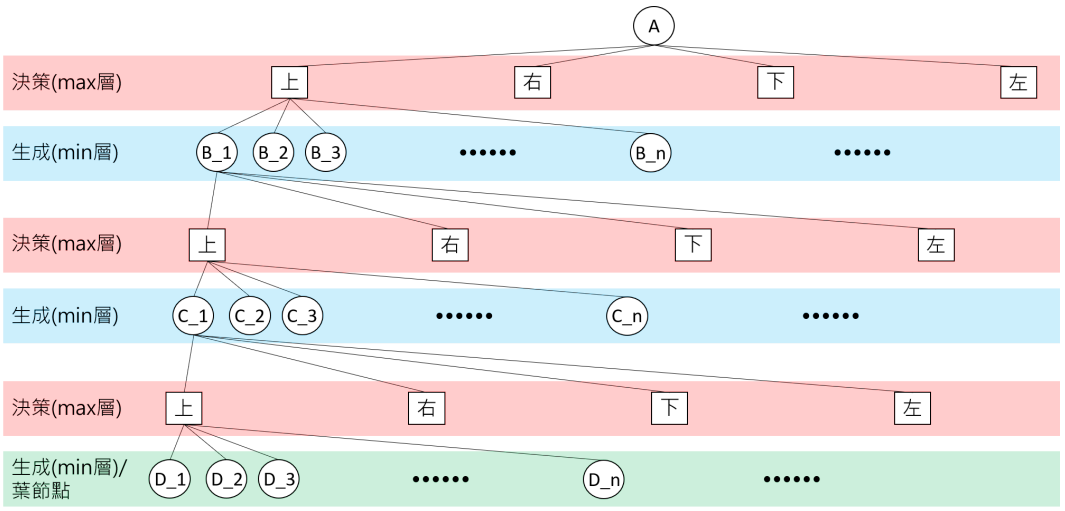
(b)單調性：四方向鄰近方塊的對數數值差的最大值，此項目會使盤面同行(列)方塊傾向嚴格遞增(減)，使盤面整齊、有序的排列。

(c)空格數：面版上沒有方塊佔據的位置數量。此項目會使盤面避免堆積方塊，減少死局的機會。

(d)最大方塊的對數數值：使盤面傾向合成出更大的方塊。

b.minimax演算法：以dfs向下模擬3步驟(每步驟包含決策的max層與生成新塊的min層，總計6層)的所有可能情況。在葉節點計算盤面評估函數後往上回傳，之後在決策節點(max層)取評估值高的回傳給生成新塊節點(min層)，在生成新塊節點取評估值低的回傳給決策節點，如此往復直到回傳到第一層決策節點。當前盤面(根節點)根據第一層決策節點的評估值，取最大者作為決策結果。

c.alpha-beta剪枝：dfs過程中透過alpha、beta紀錄上層評估值的上下界，在確認某節點已無法得出更好的解後，就不需要再繼續此分支計算，節省時間。



2、算法原理

minimax常被用於棋類零和遊戲，基本邏輯是在假設己方會最大化自己的利益，而對手也會最大程度的造成我方損失，如此模擬當前局面後數手的結果，得出最有機會活命/勝利的決策。

根據參考資料，2048遊戲中的移動、增加隨機方塊等事件都有許多可能性，且可能的狀況會隨著步數疊加，指數增長，與棋類遊戲狀況相同。此外，2048玩家的任務是保持版面的整齊排列，以增加合成大塊的機會；而電腦增加隨機方塊則會使版面趨向混亂，降低玩家勝利機會。因此，電腦就像是玩家的對手一般，而添加隨機方塊就如同對手的反擊行為，玩家與電腦之間具有對弈關係。以上2048的性質，都是2048適用minimax演算法的理由。

3、算法測試

AI_2測試-基本數據表

最大方塊 數值區間	遊戲平均 分數(分)	遊戲平均 步數(步)	遊戲平均 耗時(sec)	每步平均 耗時(sec)
512 - 8192	43528.912	2135.708	91.908	0.043

AI_2測試-最大方塊分布表

最大方塊數值	512	1024	2048	4096	8192	總計
次數(次)	3	62	256	174	5	500
百分比(%)	0.6	12.4	51.2	34.8	1	100

4、優缺分析

（1）優點：三步效率極高，且結果與AI_1相差不多。

（2）缺點：理論上，minimax增加運算步數應能提升其表現，然而步數提升至四步之後運算量指數上升，若盤面不易剪枝會極為耗時。

（3）改善方法：運算步數隨每步運算時間浮動調整。

（四）AI_3

1、運算方式

在AI_2的架構下，若前一步的運算時間在0.05秒以下，則下一步的運算步數會提升為四步；前一步的運算時間在0.5秒以上，則運算步數降為三步。

2、算法測試

AI_3測試-基本數據表

最大方塊 數值區間	遊戲平均 分數(分)	遊戲平均 步數(步)	遊戲平均 耗時(sec)	每步平均 耗時(sec)
512 - 8192	45827.736	2234.916	409.908	0.183

AI_3測試-最大方塊分布表

最大方塊數值	512	1024	2048	4096	8192	總計
次數(次)	5	45	247	200	3	500
百分比(%)	1	9	49.4	40	0.6	100

3、優缺分析

（1）優點：AI_3平均分數比AI_2高，且最大方塊達到2048的比例較高。

（2）缺點：運算步數浮動，每步運算耗時也有較大差異，決策時快時慢。

八、參考資料

（一）stackoverflow
https://stackoverflow.com/questions/22342854/what-is-the-optimal-algorithm-for-the-game-2048

（二）AI_0、AI_1參考：https://github.com/ronzil/2048-AI

（三）AI_2、AI_3參考：https://github.com/ovolve/2048-AI