計算機概論 Final Project

1. 專題簡介

Tap Me Plus One是一款基於手機平台的益智遊戲，旨在提供娛樂和挑戰性的遊戲體驗。遊戲中，玩家需要點擊數字方塊，使其數值增加，並在有限的生命力獲取分數。這個專題旨在開發一個直觀且令人上癮的遊戲，同時也促進玩家在數字計算的能力。挑戰玩家的集中力和反應能力。玩家可以通過不斷練習和提高自己的技巧來達到更高的分數。

1. 遊戲規則與說明
2. 遊戲開始時，玩家將看到一個方塊棋盤，棋盤上有多個帶有數字的方塊。
3. 玩家需要點擊方塊，使其數字增加1。點擊一次方塊，該方塊上的數字將增加1。
4. 當3個以上數字相同的方塊連在一起時，它們將合併為一個數值為原數字加1的方塊。
5. 玩家需要在有限的血量盡可能多地點擊方塊，以獲得更高的分數。
6. 遊戲中有血量條，血量歸零則遊戲結束。
7. 玩家的分數將根據點擊的方塊數量計算，點擊數量越多，得分越高。

示意圖：

說明：玩家需要注意數字方塊的連接和合併，以最佳策略點擊方塊，同時保持血量不斷，以獲得最高的分數。挑戰自己的極限，創造新的紀錄

1. 視窗介紹與功能（附上截圖）

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

1. 程式介紹

我們的程式使用主要根據以下流程切分程式，讀取影片->目錄->遊戲->結束畫面，再根據每個部分開發延伸功能以及附加功能。

* 遊戲流程

遊戲的流程可以簡化為下列流程圖：

|  |
| --- |
|  |

等待點擊是 Event-driven programming 的核心，因為點擊了方塊而產生一連串的程式結果。

其中 BFS function 實作了帶路徑的廣度優先搜尋，當點擊了一個方塊並將其數值加一，並執行 BFS 演算法尋找是否有大於三的連通分量，如果有就可以進行合併，而且因為需要進行合併，我需要知道所有葉節點（非點擊但可合併之方塊）到根節點（點擊的方塊）的最短路徑，並且沿著最短路徑回收方塊，所以我使用廣度優先搜尋，可以一次就得到所有節點的最短路徑，並且根據記錄的表格 Backtracking 得到路徑實作動畫。

|  |  |
| --- | --- |
|  | 點擊 (1,1) 方塊（橘色方塊）並將其加一（由 3 變 4）。並將該點放入 BFS Queue 當中。 |
|  | 接著開始 BFS，將 Queue pop 一個元素出來（橘色方塊），並將橘色方塊周圍也是 4 的方塊加入 Queue 當中。之後右側表格則紀錄路徑。 |
|  | 走到淺橘色方塊，並記錄 R （因為從右邊來，回去要走右邊），接著將其他方塊加入 Queue。 |
|  | 走到淺橘色方塊，並記錄 U。 |
|  | 走訪結束。 |
|  | 開始回朔，得到路徑，以綠色方塊為例。 |
|  | 向上走。 |
|  | 向右走。 |

透過上述表格說明可以知道如何進行 BFS 演算法，之後使用 Backtracking 則可以得到綠色方塊的路徑，接著對所有被走訪過的方塊做相同的事情，並且套用移動的動畫即可完成一次合併。

|  |
| --- |
|  |

合併之後會產生出沒有方塊的空白格，接著我們要進行填充（Padding）。

|  |
| --- |
|  |

搜尋所有方塊底下有多少格空白格（泡泡），該數量則是此方塊應向下移動的數量，即可套用動畫並且移動該方塊。

|  |
| --- |
|  |

經過填充可以保證所有的空白格都會被飄到上面去，即空白格排列必定由上而下而且連續，因此我們就可以直接填充隨機的方塊進入空白格，就完成一次合併。

接著，我撰寫一個檢查式，檢查版面中是否有任何可合併的方塊，因為可合併的方塊不一定是由點擊造成，合併也可能造成更多可合併的方塊，故我要檢測可合併的方塊，並由上述流程合併直到版面無可合併的方塊為止。

* 問題面對

1. 監測動畫播放

因為在檢測可合併方塊時，可能同時出現多個可合併的區塊，但因為撰寫方式無法將 BFS Function 以 Sequential 方式執行，造成若同時呼叫多個 BFS Function 會造成多個區塊同時合併造成錯誤，因此我想要將 BFS Function 以 Sequential 方式執行，最主要訴求就是整個畫面只能同時有一個動畫正在被播放。

為了解決這個問題我設定一個全域變數 flag，在進入動畫時將 flag 設為 true、結束動畫時設為 false，並且開啟一個新的線程（Thread）監控 flag 的變化，當確定 flag 是 false 才執行下一個動畫，即可解決動畫問題。

1. CPU 使用率過高

呈上 1.，我使用新的 Thread 監控，故使用到 while(true) 迴圈，但我沒有設定休息時間導致系統無時無刻都在高效率執行這個迴圈，所以我是用 Thread.sleep(100); 每次監控都讓 Thread 休息一下，就可以讓 CPU 使用率有大幅度下降。

* GitHub Page 與成果

1. 結論