Data Structure Project1:Tetris 俄羅斯方塊

NTHUPHYS 106022114 游惟翔

● 前置作業

1. 定義 block

Block 的資訊儲存在 box 的二維陣列中(圖 1.)，而所有的 block 皆可由 4*4 的小 matrix 表示，因此我先將 box 做 0-15 共 16 個 elements 在陣列中(表 1.)。

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

表 1.(左上)陣列意識圖

再透由讀取方塊時的方塊種類判定，將 x.y 加以標示，此處我們先定義 block 內涵 x.y 兩個元素(圖 2.)，而它有四項(因每個 block 由四個小方塊組成)

```
//表示該block
struct block
{int x,y;} a[4];
```

圖 2.(右下)定義 block 的 x.y

```
//block分類內容
int box[19][4] =
{
    8,9,10,13,  // T1
    5,8,9,13,   //T2
    9,12,13,14, //T3
    4,8,9,12,   //T4
    4,8,12,13,  // L1
    8,9,10,12,  //L2
    4,5,9,13,   //L3
    10,12,13,14,//L4
    5,9,12,13,  // J1
    8,12,13,14, //J2
    4,5,8,12,   //J3
    8,9,10,14,  //J4
    9,10,12,13, //S1
    4,8,9,13,   //S2
    8,9,13,14,  //Z1
    5,8,9,12,   //Z2
    0,4,8,12,   // I1
    12,13,14,15,//I2
    8,9,12,13,  //O
};
```

圖 1.(最右)各 block 的資料

2. ifstream

透過 ifstream 輸入來自目標檔案的資料(圖 3.)，用迴圈將每個以空格分開的 element 讀進來做判斷。利用檔案具有的特定格式，新設一整數 count 來計算現在進行到檔案中的哪一個 element，每當進行完該迴圈後 count+1，並讀入下一個 element 的訊息。

```
int main(int argc, char** argv)
{
    //讀入測資
    ifstream ifs(argv[1]);
    string str;
    //將每個測資依照空格分別讀入，由於格式相同可用位置處理
    int count = 0;
    while (ifs >> str)
    {
        count++;
```

圖 3. 以 argv[]讀入測資

3. stringstream

而 ifstream 所取的資料型態都為 string，因此使用 stringstream 將資料轉變成 int 整數型態做使用。

```
stringstream ss;
ss<<str;
ss>>m;
```

圖 4. stringstream 的表示方式

4. 判斷 matrix 大小

讀入的第一 row 資料為 m 個 rows 和 n 個 columns，剛好對應到 count=1,2 讀到的 elements，因此用條件式 if 判斷分別得到代表遊戲進行的行和列。

5. 判斷程式終結

當讀入的 str 讀取到 End 時，跳出迴圈。

● 讀取方塊

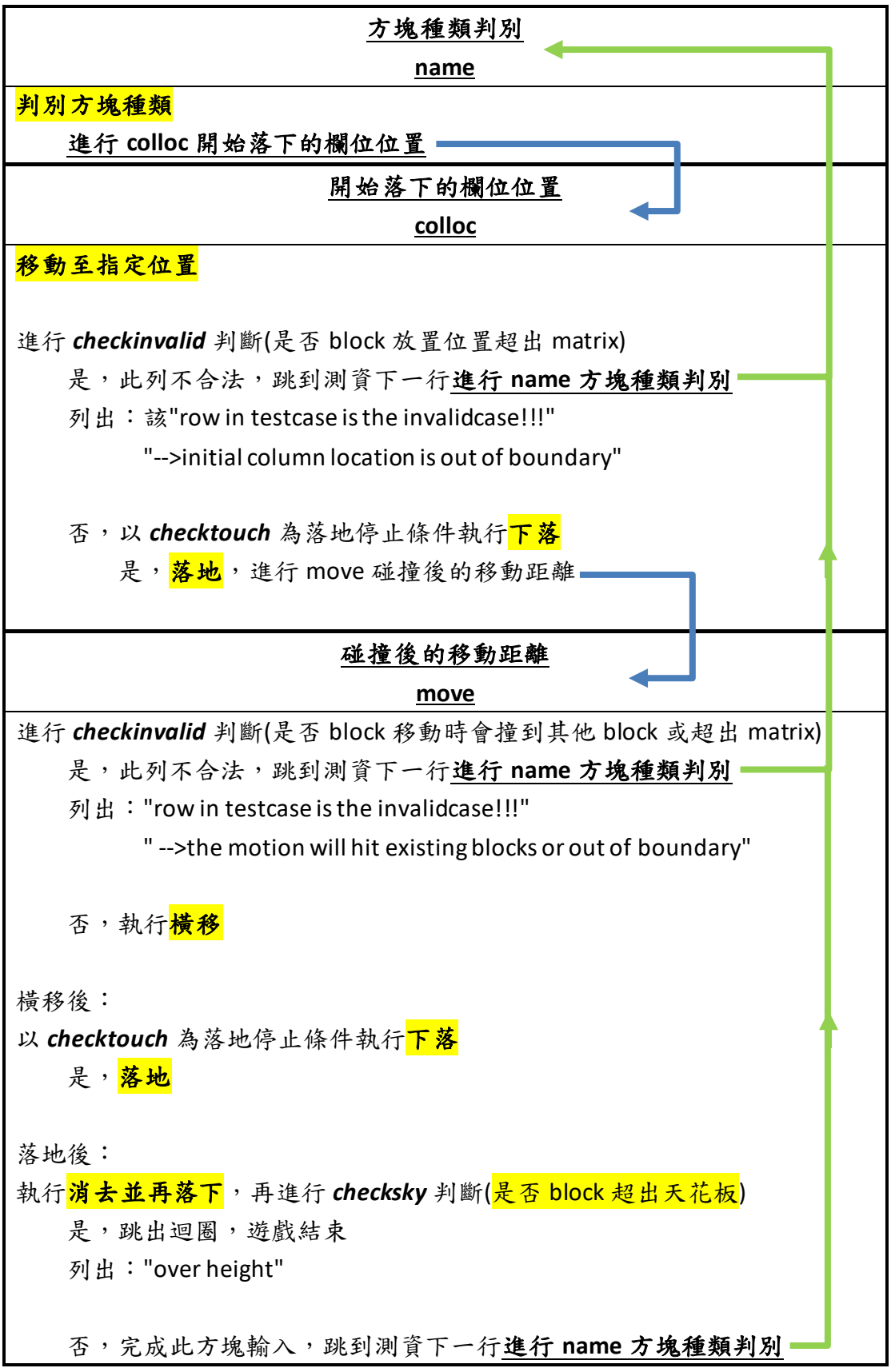

表 2. 讀取方塊流程圖

進行讀取方塊

而接下來從測資讀取到的資訊都為每一個方塊 block 所表示的訊息：
方塊種類(name)、開始落下的欄位位置(colloc)、碰撞後的移動距離(move)
而觀察 count 變數，當資料為方塊種類時，count 會是 3 的倍數；落下的欄位位置是除 3 後餘 1；移動距離的 count 是除 3 後餘 2。因此我們用以上的特性，用條件式 if 對各個讀到的 element 做分類。

1. 方塊種類(name)：(count%3)==0

block 的資訊儲存在 box 的二維陣列中，而所有的 block 皆可表示在 4*4 的小 matrix 上。所以我將 box 陣列，依照%4 轉換成水平軸 x:向右為正，/4 轉換為垂直軸 y:向下為正的 4*4 二維座標(表 3.)，並將 x.y 座標輸入 block 內(圖 5.)。

(0,0)	(1,0)	(2,0)	(3,0)
(0,1)	(1,1)	(2,1)	(3,1)
(0,2)	(1,2)	(2,2)	(3,2)
(0,3)	(1,3)	(2,3)	(3,3)

表 3. 小 matrix 意識圖

```
//方塊的種類
if ((count%3)==0)
{
    name=str;
    if (name == "T1")
    {
        for (int i=0;i<4;i++)
        {
            //a[i].x block表示 最左為0 col
            //a[i].y block表示 最上為0 row
            //reference point 為(a[i].x+colloc,a[i].y+4) 也就是(0,4)
            //map到matrix上時，row不變，天花板下降4
            a[i].x = box[0][i]%4;
            a[i].y = box[0][i]/4;
        }
    }
```

圖 5. 指派 x.y 資訊到以 a 為名的 block 型態 class

2. 開始落下的欄位位置(colloc)：(count%3)==1

確認完 block 的種類後，即可以從測資當中跳下一個資料-開始落下的欄位，並且將值加到原本第 1 步**判斷方塊種類**時設的小 matrix 上，因此這個小 matrix 平移到正確的欄位(圖 6.)，block 初始的落下位置確定。

```
//colloc
else if ((count%3)==1)
{
    stringstream ss;
    ss<<str;
    ss>>colloc;

    //依照測資第二格說的欄位選定地方下落
    for (int i=0;i<4;i++) a[i].x+=colloc;
```

圖 6. 移動 block 到指定欄位

隨後進行這個移動是否合法，有時部分的方塊會移動過度，導致超過左右邊界，因此在此使用 ***invalidcase*** 進行檢查(圖 7.)，合法後開始進行落下(圖 8.)直到第一次撞到地面(以 ***checktouch*** 進行判斷)。

```
//判斷是否是invalidcase
if (! checkinvalid(m,n,move))
```

圖 7. 判斷是否擺錯 block

```
while(checktouch(m,n))
{
    //執行下落
    for (int i=0;i<4;i++)
        a[i].y+=1;
```

圖 8. 開始落下直到撞地面

3. **碰撞後的移動距離(move)：(count%3)==2**

第一次觸地後，從測資讀取橫移資料，開始進行橫移(圖 9.)，並且在此以 ***invalidcase*** 檢查是否合法(圖 7.)，在橫移過程中沒有碰觸到既有的方塊，一切通過以後，在進行下落直到觸地停止(以 ***checktouch*** 進行判斷)。

```
//執行橫移
for (int i=0;i<4;i++)
    a[i].x+=move;
```

圖 9. 橫移

在進行消去前，先將該 block 儲存到大的 matrix 上，再執行消去。

透過 count 計算 matrix 每一列已經擁有的值，因此當 count 和欄位總數 n 的值相符時，則可以表示該行已完全被填滿，可以進行消去，並將上消去之後的上層資料下拉做取代(再落下)。

最後，以 ***checksky*** 判斷是否超出天花板，若是都通過，則跳回下一個測資繼續進行 block 的判別……直到遊戲結束。

```
//儲存現在的field上的狀態
for (int i=0;i<4;i++)
    field[a[i].y][a[i].x-1]=1;

int k=m+3;
//消去變為0
for (int i=m+3;i>=0;i--)
{
    int count=0;
    for (int j=0;j<n;j++)
    {
        //檢查整排是否都有值
        if (field[i][j]) count++;
        field[k][j]=field[i][j];
    }
    if (count<n) k--;
}
```

圖 10. 儲存 block 並進行消去

● 判斷式

1. ***checkinvalid*** 判斷

block 放置位置超出 matrix/**落下的欄位位置(colloc)**

移動時會撞到其他 block 或超出 matrix/**碰撞後的移動距離(move)**

3 2

Z2 1 1

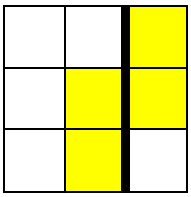

表 3.解釋落下位置設定錯誤的狀況，code 為圖 11.else if (a[i].x>n 的)

```
//invalid case 判斷(中間撞到方塊或移動超出邊界)
bool checkinvalid(int m,int n,int move)
{
    for (int i=0;i<4;i++)
        if (move<0)
        {
            for (int j=0;j>=move;j--)
            {
                //中間撞到方塊
                if(field[a[i].y][a[i].x-1+j]) return 0;
                //超出左右兩邊邊界
                else if (a[i].x+move<0) return 0;
            }
        }
        else if (move>0)…
        else if (a[i].x>n) return 0;
    return 1;
};
```

圖 11. ***Checkinvalid*** 程式碼

2. ***checktouch*** 判斷

著地判斷/**落下的欄位位置(colloc)**

/碰撞後的移動距離(move)

由於我的 matrix 是從 colloc 後，由 box 分配成的小 matrix 拓展而成(可以想成在 matrix 上還有一個 4*n 的區域)，因此判斷式 if 中的列高從 m 改為 m+3。

```
//check著陸/卡到方塊
bool checktouch(int m,int n)
{
    for (int i=0;i<4;i++)
        //著陸
        //因為從col開始算 多3row
        if (a[i].y==m+3) return 0;
        //卡到底下的方塊
        else if (field[(a[i].y)+1][a[i].x-1]) return 0;
    return 1;
};
```

圖 12. ***checktouch*** 程式碼

3. ***checksky*** 判斷

天花板判斷/**碰撞後的移動距離(move)**

整個讀取方塊流程迴圈的最後一步，和 2. ***Checktouch*** 的理由一樣，因為 matrix 式拓展而成的，因此判斷時只需要看最上面拓展的部分是否在程式全部執行完以後還有值。

```
//check 是否超過天花板
bool checksky(int n)
{
    for (int i=0;i<4;i++)
        {
        for (int j=0;j<n;j++)
            if (field[i][j]==1) return 0;
        }
    return 1;
};
```

圖 13. ***checksky*** 程式碼

● 輸出結果

最後由 ofstream 指令，生成新的檔案：“106022114_proj1.final”，將 field 內的最終資料以迴圈的方式寫入檔案中，並在每個 element 之間以空格做區分。

而和判斷式中 ***checktouch***、***checksky*** 當中提及的理由一樣，因為 field 是將測資提供的 matrix 大小+生成 block 用的 4*n 大小的 matrix，因此在此需要進入檔案的 elements 從第四 row 開始取。(圖 14.)

```
ofstream fout("106022114_proj1.final");

//輸出結果至檔案
for  (int i=4;i<m+4;i++)
{
    for (int j=0;j<n;j++)
    {
        if (j==n-1)
        {
            fout<<field[i][j];
        }
        else
            fout<<field[i][j]<<" ";
    }
    fout<<endl;
}
```

圖 14. Ofstream 輸出檔案

● TestCase 設計

希望透過落地以後的左右移動，將”NTHU”四個英文字母呈現出來，所以這段 testcase 沒有削去的列，都是在觸地以後開始進行移動。

而執行的結果雖沒有預期的明顯字樣，已有雛型，而 testcase 內也有著大量需要移動的 block，可以增加程式執行時移動的次數。

106022114_proj1.final - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 1 0 0 0 1 1 0 0 1 1 1 1 0 1 1 1 0 0 1 1 1 0 0 1 0 0 0 0 0 1 1 1 1
1 1 1 0 0 1 1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 0 0 1 1 1 0 0 1 1 0 0 1 1
1 1 1 0 0 1 1 1 0 0 0 0 1 1 0 0 0 0 1 1 1 1 1 1 1 1 0 0 1 1 0 0 1 1
1 1 1 1 0 1 1 0 0 0 0 0 1 1 0 0 0 0 1 1 1 1 1 1 1 1 0 0 1 1 0 0 1 1
1 1 0 1 1 1 1 0 0 0 0 0 1 1 0 0 0 0 1 1 1 0 0 1 1 1 0 0 1 1 0 0 1 1
1 1 0 1 1 1 1 0 0 0 0 0 1 1 0 0 0 0 1 1 1 0 0 1 1 1 0 0 1 1 0 0 1 1
1 1 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1 1 1 1
```

106022114_proj1.data

檔案(F) 編輯(E) 格式(O)

```
10 34
0 1 0
I1 1 0
T4 1 1
T2 1 4
I1 6 1
L2 1 5
I1 6 7
I1 14 0
L2 1 13
O 15 5
O 20 4
I2 30 0
I1 31 3
I1 34 -5
I1 34 -4
I1 32 1
T4 1 3
T4 30 -1
T2 33 0
J1 24 0
L1 21 -1
I2 19 2
J4 32 -21
I2 20 1
O 20 4
O 20 0
O 33 -27
L2 14 -4
J4 18 -3
S2 9 -7
I1 34 -8
I1 34 -15
I1 26 0
I1 19 0
End
```