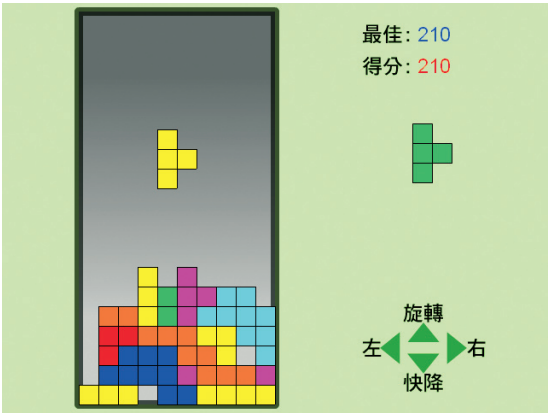
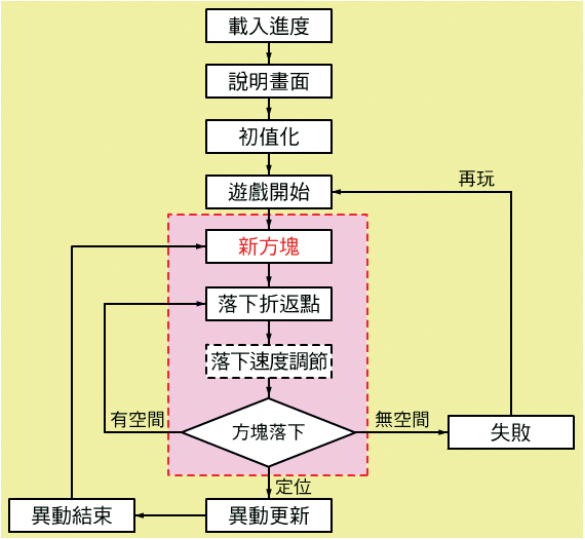
7-2 操作說明

- 執行【俄羅斯方塊 .swf】檔案。按下 Enter 鍵進入遊戲後，隨機的 7 種方塊組合將由舞台上方落下，玩家可使用四個方向鍵控制其移動位置與旋轉角度，直到組合方塊落至碰觸底部或任一堆積的方塊後停止，換由右邊待命組合方塊繼續落下。



- 停止的方塊組合將堆積於底部，如果任一水平列填滿方塊則該列方塊可全部消去並得分，上方堆積的其餘方塊則落下填補。一次消去越多列則得分可越多，隨著得分增加方塊落下速度將逐漸加快。玩家可不斷消去方塊製造置放的空間，直到方塊堆積至頂部則遊戲失敗結束，玩家可按下 Enter 鍵重玩遊戲。

7-3 遊戲流程

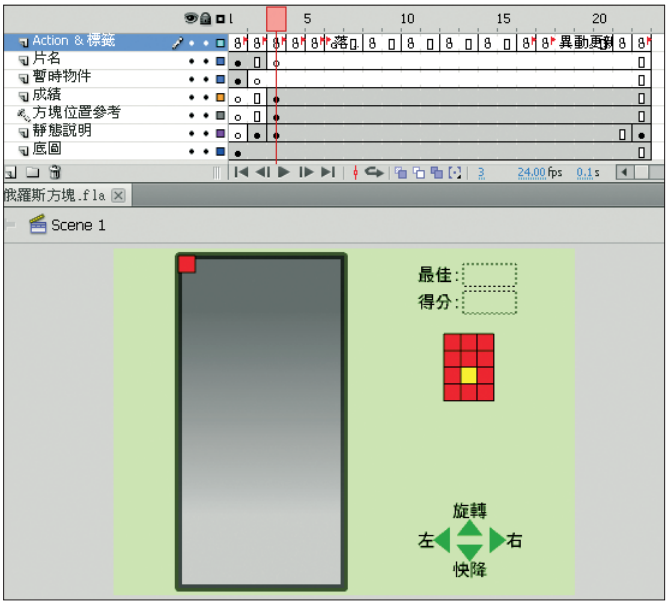


- 經典遊戲不需要複雜的流程，看似持續進行的動態遊戲，其實巧妙的使用回合制。每回落下一組合方塊，由玩家操控直到方塊落至定位（底部或其他方塊之上），進行異動更新程序，再繼續下一組方塊落下回合。
- 方塊組合從落下至定位停止的期間為玩家操控階段，如上圖紅色虛線框範圍，在「新方塊」程序設定玩家操控介面，「異動更新」程序則移除避免干擾方塊異動。
- 「方塊落下」程序除了判斷是否停止落下之外，同時監控無置放空間遊戲失敗的情況。
- 「落下速度調節」程序控制組合方塊落下的速度。本遊戲使用影格數目調節的方法，得分越多時落下過程的間隔時間（影格數）逐漸減少。實際作法為在「落下折返點」與「方塊落下」兩程序之間的影格區間範圍內，設計者可視需要設定不同階段的得分條件，原則為分數越多則越快進入「方塊落下」程序判斷。這種處理方式最簡單，而且不影響遊戲的進行節奏，即前言所謂的「無障礙式關卡設計」。

- 「異動更新」程序處理堆積的方塊有無水平列填滿的情況，有則將該列方塊標示為灰色圖像，於「異動結束」程序再正式消去。兩程序之間有影格時間差，做為方塊消去前的視覺停留。

7-4 舞台圖層配置

舞台場景採用水平方式的 640x480，成績記錄置於右上方，下一回落下方塊組合置於其下，鍵盤操作的說明文字則置放於右下方。左邊矩形範圍為方塊落下與堆積的方塊區域。



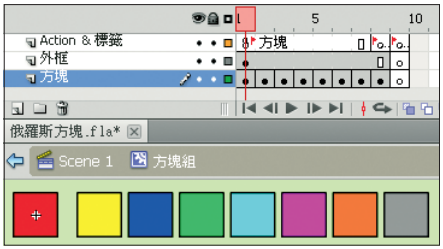
- 本遊戲全程使用鍵盤操作，沒有互動使用的一般按鈕圖層。由於鍵盤的按鍵比滑鼠複雜許多，因此通常必須體貼玩家，應置放使用按鍵的說明文字，靜態說明圖層即擔任這個任務。

- 成績圖層置放舞台上方的最佳成績動態文字 top_txt、得分動態文字 score_txt。

- 方塊位置參考圖層為導引圖層屬性，做為置放方塊區域左上角方塊與右邊方塊組合位置的參考。遊戲使用的全部方塊物件均使用元件庫類別連結動態建立的方式，沒有在舞台上出現。
- 底圖圖層置放黑色漸層靜態底圖，做為遊戲方塊落下與堆積的方塊區域，同時做為設定全部方塊物件原始位置的參考。

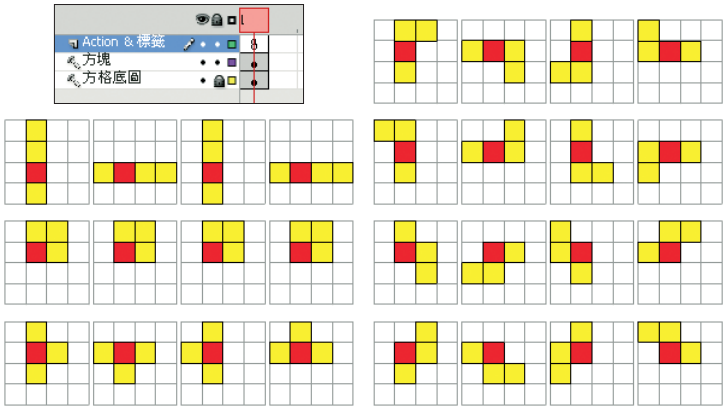
7-5 方塊元件

7-5-1 方塊組元件



本遊戲方塊元件的影格 1 ～ 7 分別置放不同顏色方塊圖像，但是和之前的方塊遊戲做為同色方塊相連結消去的意義不同，這裡它們只做為落下方塊多種顏色的視覺效果，與連結無關。影格 8 置放灰色方塊圖像，標籤為「灰色」，做為水平列方塊整列填滿時即將消去前的顯示用途。影格 9 為沒有圖像的空白關鍵影格，標籤為「無方塊」，做為方塊區域沒有方塊的空格位置使用。

7-5-2 方塊結構 1 ～ 7 元件



前言提到遊戲設計原理同 LED 跑馬燈方式，指的是遊戲落下的方塊組合由 4 個方塊組成，如上圖共有 7 種組合方式。其中紅色方塊為組合方式的中心方塊，玩家每按一次向上鍵，方塊組合將以該方塊為中心順時鐘方向旋轉 90 度，結果每一種方塊組合將產生 4 種不同方式的組合（其中田字型亦視為 4 種）。注意紅色方塊的垂直位置，除了一字型為 3（由上向下）的位置外，其餘皆為 2，方塊組合將由方塊區域的最高處落下，亦即這個位置（方塊垂直編號），因此程式碼將分為一字型與其他類型兩種情況。

實務上並不需要準備 7x4 個元件做為落下方塊實體，而是將 7 種組合方式分別利用一個方塊結構元件做為“描述”方塊組合的物件，雖然該元件有方塊圖層置放方塊、方格底圖圖層置放位置參考方格（左上圖），但兩者皆為導引圖層，僅做為參考方塊的位置使用，真正應用的是「Action & 標籤」圖層影格內的程式碼，設定 3（方塊）x4（方向）共 12 個變數，做為 3 個非中心位置的黃色方塊與紅色方塊（中心位置）的方塊編號相對差距數，如以下程式碼以一字型方塊組合為例：

```
01      var p11:String = "+0-2";
02      var p12:String = "+0-1";
03      var p13:String = "+0+1";
04
```

```
05      var p21:String = "-1+0";
06      var p22:String = "+1+0";
07      var p23:String = "+2+0";
08
09      var p31:String = "+0-2";
10      var p32:String = "+0-1";
11      var p33:String = "+0+1";
12
13      var p41:String = "-1+0";
14      var p42:String = "+1+0";
15      var p43:String = "+2+0";
```

變數名稱由 3 字元組成，p 為命名開頭字母；接下來第 1 個數字代表旋轉狀態，1 ～ 4 分別表示旋轉角度 0、90、180、270；第 2 個數字代表黃色方塊 1 ～ 3。變數值宣告為字串類態，由 4 個字元組成，前 2 字元代表水平方向與紅色方塊的方塊編號差距，紅色方塊的左方為負、右方為正；後 2 字元代表垂直方向與紅色方塊的方塊編號差距，紅色方塊的上方為負、下方為正。方塊初落下時旋轉狀態初值為 1，即旋轉角度為 0，變數名稱第 1 個數字為 1；當玩家旋轉一次後變成 90 度，即變數名稱第 1 個數字變更為 2。所以利用 12 個變數值即可產生舞台上看似方塊組合落下、旋轉角度的變化，實際上是由預先建立方塊區域的方塊顏色值數據的改變所產生的，與 LED 動態跑馬燈的運作方式完全相同。

7-6 主場景程式

► 影格 1，「載入進度」

和第 2 章遊戲「載入進度」影格相同，不再贅述。

► 影格 2，「說明畫面」

主場景時間軸在此停止播放，置放遊戲規則及註冊鍵盤事件偵聽處理函數提供玩家互動。

```
01    stage.addEventListener(KeyboardEvent.KEY_DOWN, startKey);
02    stop();
03
04    function startKey(event:KeyboardEvent) {
05        var codeK:int = event.keyCode;
06        if ( codeK == 13 ) {
07            play();
08            stage.removeEventListener(KeyboardEvent.KEY_DOWN, startKey);
09        }
10    }
```

解說

行 01：因為採用鍵盤操作不提供按鈕，所以在 stage 註冊鍵盤事件偵聽處理函數 startKey()。請注意使用鍵盤操作的遊戲在測試階段時，應將測試視窗內控制選項的“停用鍵盤快速鍵”項目鉤選起來，避免按鍵與 Flash 預設的快速鍵衝突造成錯誤動作。

行 02：主場景時間軸暫停，等待玩家按下 <Enter> 鍵。

行 04 ～ 10：自訂按鍵事件偵聽處理函數 startKey()。使用 keyCode 屬性取得按鍵碼值（行 05），如果該值等於 13，即 <Enter> 鍵的按鍵碼值，主場景時間軸開始播放（行 07），並移除此函數（行 08）。

► 影格 3，「初值化」

在此除了設定一般遊戲初值及全部方塊實體建立之外，7 種方塊結構實體也在此連結建立。

```
01    const x_num:int = 10;
02    const y_num:int = 20;
03    const blockTypes:int = 7;
04    const blockColors:int = 7;
05    const blockWidth:Number = 23;
06    var type_1:type_mc1 = new type_mc1();
07    var type_2:type_mc2 = new type_mc2();
```

```
08      var type_3:type_mc3 = new type_mc3();
09      var type_4:type_mc4 = new type_mc4();
10      var type_5:type_mc5 = new type_mc5();
11      var type_6:type_mc6 = new type_mc6();
12      var type_7:type_mc7 = new type_mc7();
13      for ( var i:int=1; i<=x_num; i++ ) {
14          for ( var j:int=1; j<=y_num; j++ ) {
15              this["block" + i + "_" + j] = new block_mc();
16              this["block" + i + "_" + j].x = (i-1) * blockWidth + 101.5;   //(101.5,21.5)
17              this["block" + i + "_" + j].y = (j-1) * blockWidth + 21.5;
18              this["block" + i + "_" + j].bx = i;
19              this["block" + i + "_" + j].by = j;
20              this["block" + i + "_" + j].gotoAndStop("無方塊");
21              this.addChild(this["block" + i + "_" + j]);
22          }
23      }
24      for ( i=1; i<=3; i++ ) {
25          for ( j=1; j<=4; j++ ) {
26              this["next" + i + "_" + j] = new block_mc();
27              this["next" + i + "_" + j].x = i * blockWidth + 447;   //(470,130)
28              this["next" + i + "_" + j].y = j * blockWidth + 107;
29              this["next" + i + "_" + j].gotoAndStop("無方塊");
30              this.addChild(this["next" + i + "_" + j]);
31          }
32      }
33      var topScore:int;
34      var data_so:SharedObject = SharedObject.getLocal("highscore");
35      if ( data_so.data.score != undefined ) {
36          topScore = data_so.data.score;
37      } else {
38          topScore = 0;
39          data_so.data.score = 0;
40      }
```

解說

行 01：設定水平 x 方向的方塊總數 x_num。

行 02：設定垂直 y 方向的方塊總數 y_num。

行 03：設定方塊組合數 blockTypes。

行 04：設定方塊顏色總數 blockColors。

行 05：設定方塊的寬度 blockWidth，做為計算方塊座標用途。

行 06 ～ 12：分別宣告建立 7 種方塊組合實體，做為落下方塊組合的計算處理。

行 13 ～ 23：依慣例使用迴圈方式宣告建立舞台上使用的全部方塊實體、方塊編號等。全部方塊實體初始狀態均顯示無方塊的空格位置待命（行 20）。

行 24 ～ 32：使用迴圈宣告建立舞台右邊下次落下方塊組合使用的方塊實體，全部實體初始狀態同樣顯示無方塊的空格位置待命（行 29）。

行 33：宣告建立最佳成績 int 類型變數 topScore。

行 34 ～ 40：與第 2 章遊戲建立或取得最佳成績記錄的方法相同，建立記錄初值或取出最佳成績記錄。

► 影格 4，「遊戲開始」

每回遊戲（新遊戲或重玩）的開始點，在此完成遊戲前準備工作，得分歸零、全部方塊顏色值歸零等，並設定下次落下方塊組合。

```
01    var score:int = 0;
02    var bkTypeNext, bkColorNext:int;
03    for ( i=1; i<=x_num; i++ ) {
04        for ( j=1; j<=y_num; j++ ) {
05            this["block" + i + "_" + j].bc = 0;
06            this["block" + i + "_" + j].gotoAndStop("無方塊");
07        }
08    }
09    nextBlocks();
10    updateScore();
```

```
11
12      function nextBlocks() {
13          bkTypeNext = Math.floor(Math.random()*blockTypes) + 1;
14          bkColorNext = Math.floor(Math.random()*blockColors) + 1;
15          for ( i=1; i<=3; i++ ) {
16              for ( j=1; j<=4; j++ ) {
17                  this["next" + i + "_" + j].gotoAndStop("無方塊");
18              }
19          }
20          this["next" + 2 + "_" + 3].gotoAndStop(bkColorNext);   //中心方塊位置編號(2,3)
21          for ( i=1; i<=3; i++ ) {
22              var xx:int = 2 + Number(this["type_" + bkTypeNext]["p1"+i].substr(0,2));
23              var yy:int = 3 + Number(this["type_" + bkTypeNext]["p1"+i].substr(2,2));
24              this["next" + xx + "_" + yy].gotoAndStop(bkColorNext);
25          }
26      }
27      function updateScore() {
28          if ( score > topScore ) {
29              topScore = score;
30              data_so.data.score = topScore;
31              data_so.flush();
32          }
33          top_txt.text = String(topScore);
34          score_txt.text = String(score);
35      }
```

解說

行 01：設定玩家遊戲得分，初值從 0 開始計算。

行 02：宣告設定下次落下方塊組合使用的 int 類型變數，分別為方塊組合類型與顏色。

行 03 ～ 08：使用迴圈將全部方塊實體恢復初始狀態，即顏色值歸零，並顯示空格位置。

行 09：呼叫 nextBlocks() 函數設定待命落下的方塊組合。

行 10：呼叫 updateScore() 函數更新舞台上成績記錄。

行 12 ～ 26：自定函數 nextBlocks()，負責設定待命的落下方塊組合。「遊戲開始」程序呼叫執行做為遊戲第 1 次落下的方塊組合，之後只在「新方塊」程序呼叫執行，方塊組合落下之後執行設定下次的方塊組合。

行 13：根據組合類型總數的隨機值產生方塊組合的類型。

行 14：根據顏色總數的隨機值產生方塊組合的顏色值。

行 15 ～ 19：使用迴圈將右邊方塊組合實體清除成為空格避免殘留前次圖像。

行 20：根據顏色值設定右邊方塊組合中心方塊實體的顏色圖像，中心方塊實體的方塊編號為固定值，水平 2 與垂直 3。

行 21 ～ 25：使用迴圈根據方塊結構實體提供的變數值（行 22、23）處理其它 3 個方塊實體的顏色圖像顯示（行 24）。

行 27 ～ 35：自訂函數 updateScore()，依慣例負責更新舞台上成績動態文字欄位內容，並處理最佳成績的比較與存檔。

► 影格 5，「新方塊」

設定即將落下的待命方塊組合的初值，並設定下回方塊組合，同時設定互動介面提供玩家操控遊戲。

```
01      var bkType, bkColor, bkMode, bkModeNew:int;
02      var bk_x, bk_y:int;
03      var checkFlag:Boolean;
04      bkType = bkTypeNext;
05      bkColor = bkColorNext;
06      bkMode = 1;
07      var x0:int = 5;         //中心方塊位置編號(x0,y0)=(5,2)
08      var y0:int = 2;
09      if ( bkType == 1 ) y0 = 3;          //中心方塊位置編號(5,3) for type1
10      bk_x = x0;
```