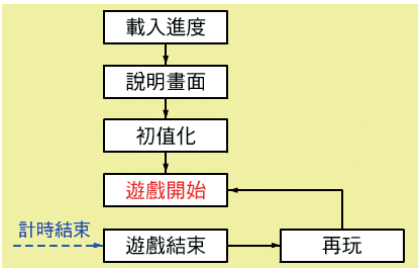
7-6 釣魚囉遊戲

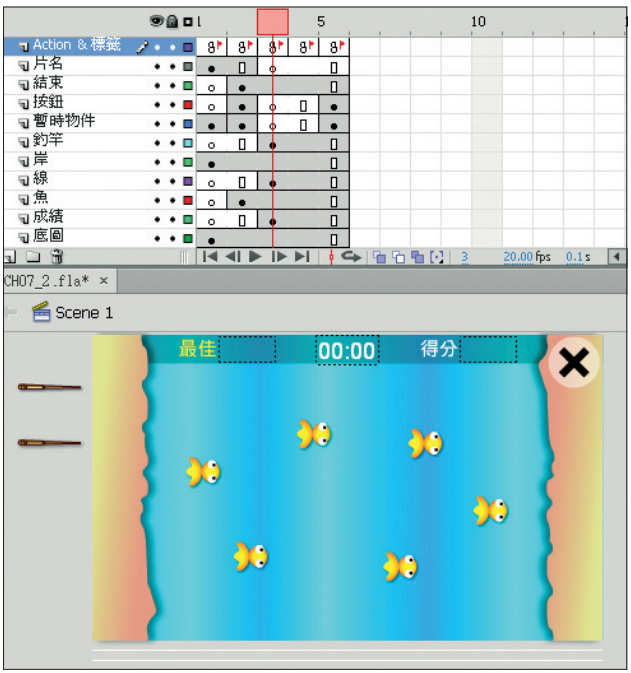
「釣魚囉」遊戲 CH07_2.fla 示範兩點觸控的應用。遊戲玩法很簡單，河裡有不同顏色、大小的魚，玩家看準魚的垂直位置，在左右兩邊河岸使用手指按下，該觸按點立即伸出釣竿與釣線，如果任何魚與水平的釣線接觸，則該魚即被釣中，不同的魚有不同的分數，包括沒有分數的黑色章魚。釣竿最多同時出現 2 支，河岸左右兩邊都可以按下，直到倒數計時結束遊戲結束，統計並記錄最佳得分。

7-6-1 遊戲流程



- 本遊戲重點為多點觸控的應用，遊戲流程非常簡單。
- 「遊戲開始」程序負責互動介面的設定，之後停止播放交給玩家進行遊戲。
- 計時器物件負責倒數計時，遊戲時間結束時自動播放至「遊戲結束」程序，設定再玩互動介面提供玩家選擇。

7-6-2 舞台圖層配置



- 按鈕圖層在「說明畫面」影格置放開始按鈕 start_btn，「再玩」影格置放再玩按鈕 reset_ btn。
- 釣竿圖層置放舞台左邊外面待命的釣竿實體 tool_mc1 與 2，遊戲期間將出現於玩家手指點觸的左右兩河岸上。
- 岸圖層置放舞台左右兩邊河岸實體 btn_mc1 與 2，提供玩家手指點觸的視覺位置，並非觸控事件物件，因為多點觸控不能使用個別物件。
- 線圖層置放舞台下方之外待命的釣線實體 line_mc1 與 2（上圖下方的 2 條白色線），遊戲期間將出現於釣竿的相同垂直位置，顯示魚是否被釣中的視覺效果。釣線實體的寬度大於河面，利用兩邊的河岸實體遮住超過河面的部分，因此應注意釣竿、河岸與釣線 3 個圖層的上下次序。
- 魚圖層置放舞台河面的魚實體 fish_mc1 ～ 6，做為玩家手指點觸釣魚的目標物件。魚實體的影片片段結構類似 4-4 節「小蜜蜂射擊」遊戲的小蜜蜂實體，擁有自己的移動流程與程式碼。
- 成績圖層置放舞台上方成績記錄動態文字，最佳成績 top_txt、計時 timer_txt 與目前得分 score_txt。
- 底圖圖層置放河道靜態底圖。

7-6-3 主時間軸程式

▶ 影格 1，「載入進度」

與 2-5 節說明相同，不再贅述。

▶ 影格 2，「說明畫面」

與 7-4 節「說明畫面」影格相同，不再贅述。

▶ 影格 3，「初值化」

設定遊戲初值，取得最佳成績及建立計時器物件。

```
01      var fish_total:int = 6;
02      var topScore:int;
03      var data_so:SharedObject = SharedObject.getLocal("highscore");
04      if ( data_so.data.score != undefined ) {
05          topScore = data_so.data.score;
06      } else {
07          topScore = 0;
08          data_so.data.score = 0;
09      }
10      var myTimer:Timer = new Timer(100);
11      myTimer.addEventListener(TimerEvent.TIMER, myTimerFunc);
12
13      function myTimerFunc(event:Event) {
14          var micro_seconds:int = total_time - (getTimer() - time_base);
15          var secondUsed:int = micro_seconds / 1000;
16          var minutes:int = Math.floor(secondUsed/60);
17          var seconds:int = secondUsed - minutes*60;
18          timer_txt.text=String(minutes+100).substr(1,2)+":"+String(seconds+100).substr(1,2);
19          if ( secondUsed < 1 ) {
20              gotoAndPlay("遊戲結束");
21          }
22      }
```

解說

行 01：設定魚實體總數。

行 02：宣告建立最佳成績 int 類型變數 topScore。

行 03 ～ 09：與第 4 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。

行 10：建立 Timer 計時器類別實體 myTimer，參數設定 100 即每 0.1 秒執行一次。

行 11：在 myTimer 實體註冊計時器事件偵聽處理函數 myTimerFunc()，處理遊戲計時的程序。

行 13 ～ 22：自訂計時器事件偵聽處理函數 myTimerFunc()，負責倒數計時的程序，處理方式與 2-4-2 節相同。如果計時秒數小於 1，遊戲時間結束，將播放至「遊戲結束」影格處理。

▶ 影格 4，「遊戲開始」

每回遊戲的開始，設定遊戲初值與互動介面。

```
01      var score:int = 0;
02      var flag1:int = 1;
03      var total_time:int = 61 * 1000;
04      var time_base:int = getTimer();
05      myTimer.start();
06      updateScore();
07      this.addEventListener(TouchEvent.TOUCH_BEGIN, clickBtn);
08      stop();
09
10      function updateScore() {
11          if ( score > topScore ) {
12              topScore = score;
13              data_so.data.score = topScore;
14              data_so.flush();
15          }
16          score_txt.text = String(score);
17          top_txt.text = String(topScore);
18      }
19      function clickBtn(event:TouchEvent) {
20          var catch_flag:Boolean = false;
21          var fish_no:int;
22          for ( var j:int=1; j<=2; j++ ) {
23              if ( this["btn_mc" + j].hitTestPoint(event.stageX, event.stageY, true) ) {
24                  if ( !this["tool_mc" + j].flag ) {
25                      if ( j == 1 ) {
26                          this["tool_mc" + flag1].rotation = 0;
27                          this["line_mc" + flag1].rotation = 0;
28                          var min:Number = this["line_mc" + flag1].width;
29                          for ( var i:int=1; i<=fish_total; i++ ) {
```

```
30                      var dy:Number = this["fish_mc" + i].y;
31                      if ( !this["fish_mc" + i].catch_flag &&
                            Math.abs(event.stageY-dy)<this["fish_mc" + i].fish_mc.height/2 ) {
32                          if ( this["fish_mc" + i].x < min ) {
33                              min = this["fish_mc" + i].x;
34                              fish_no = i;
35                          }
36                          catch_flag = true;
37                      }
38                  }
39                  this["line_mc" + flag1].x = min;
40              } else if ( j == 2 ) {
41                  this["tool_mc" + flag1].rotation = -180;
42                  this["line_mc" + flag1].rotation = -180;
43                  var max:Number = 0;
44                  for ( i=1; i<=fish_total; i++ ) {
45                      dy = this["fish_mc" + i].y;
46                      if ( !this["fish_mc" + i].catch_flag &&
                            Math.abs(event.stageY-dy)<this["fish_mc" + i].fish_mc.height/2 ) {
47                          if ( this["fish_mc" + i].x > max ) {
48                              max = this["fish_mc" + i].x;
49                              fish_no = i;
50                          }
51                          catch_flag = true;
52                      }
53                  }
54                  this["line_mc" + flag1].x = max;
55              }
56              if ( catch_flag ) {
57                  this["fish_mc" + fish_no].gotoAndPlay("釣中");
58              }
59              this["tool_mc" + flag1].y = event.stageY;
60              this["tool_mc" + flag1].x = event.stageX;
61              this["tool_mc" + flag1].play();
62              this["line_mc" + flag1].y = event.stageY;
63              this["line_mc" + flag1].play();
64              flag1 ++;
```

```
65                  if ( flag1 > 2 ) flag1 = 1;
66              }
67          }
68      }
69  }
```

解說

行 01：設定得分 int 類型變數 score，初值由 0 開始累計，累加釣中魚的個別分數。

行 02：設定 int 類型變數 flag1，做使用釣竿的編號，初值為 1（第 1 支），使用之後累加 1，因釣竿只有 2 支，編號值大於 2 時再回復 1。

行 03：設定倒數計時的毫秒數，遊戲時間 60 秒，再加 1 秒做為計時器扣除使用，全部秒數乘上 1000 成為毫秒數。

行 04：即將開始計時，因此取得 Flash 影片開始播放迄今的毫秒數 time_base，用來扣除計時開始前的時間。

行 05：myTimer 計時器實體開始播放，即開始計時。

行 06：呼叫 updateScore() 函數，更新舞台上成績記錄。

行 07：在主場景註冊觸控按下事件偵聽處理函數 clickBtn()，處理手指按下任何位置的程序。

行 08：時間軸停止播放，交給玩家進行遊戲。

行 10 ～ 18：自訂函數 updateScore()，與之前遊戲相同，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。

行 19 ～ 69：自訂觸控事件偵聽處理函數 clickBtn()，負責手指按下處理釣魚的程序。本遊戲示範多觸控點的使用，事件物件不宜使用個別實體，必須使用整個場景，因此處理程序比較複雜。程序大致為使用 hitTestPoint() 方法偵測觸控點為左或右岸實體，決定釣竿與釣線出現

的方向，再利用觸控點的 y 座標判斷是否有魚實體的範圍正好在該座標內，則該魚將被釣中。

行 20：設定釣中魚指標 Boolean 類型變數 catch_flag，初值為 false 未釣中魚，處理過程如果觸控點 y 座標在魚實體的範圍內，則更改為 true 釣中魚。

行 21：宣告 int 類型變數 fish_no，將做為釣中魚實體的編號。

行 22 ～ 68：因為河岸有左與右兩邊，實體名稱分別為 btn_mc1 與 btn_mc2，因此使用迴圈處理兩個河岸實體。迴圈值同時做為出現的釣竿與釣線實體的編號。

行 23 ～ 67：使用 hitTestPoint() 方法偵測觸控點與河岸實體是否碰觸，遊戲規則玩家手指必須在河岸碰觸才算有效的釣魚動作，才有需要進一步處理。

行 24 ～ 66：觸控點在河岸實體範圍成立後，必須確認目前編號的釣竿實體指標是否為 false，該指標在待命階段為 false，釣魚出現後更改為 true 表示正在釣魚中，經過一段時間才會恢復 false 狀態（特殊影片片段將詳細說明）。因此必須該指標為 false 待命階段才可以出現釣魚。

行 25 ～ 39：迴圈值 1，即左邊河岸實體被按下的情況，釣竿與釣線實體將以左邊為基準出現。

行 26：設定目前編號釣竿實體的旋轉角度為 0，因為釣竿實體圖像原本即為向左方向，不需要旋轉。

行 27：設定目前編號釣線實體的旋轉角度為 0，釣線實體圖像也是向左方向，不需要旋轉。

行 28：設定 Number 類型變數 min，將做為釣線實體的 x 座標，預設值為釣線實體的寬度，必須視有無釣中魚才能決定最後數據。

行 29 ～ 38：使用迴圈檢視全部魚實體是否有被釣中。

行 30：設定 Number 類型變數 dy，做為迴圈值編號魚實體的 y 座標記錄。

行 31 ～ 37：迴圈值編號魚實體是否被釣中的判斷式。有 2 個情況必須成立，魚實體的 catch_flag 指標為 false，及觸控點 y 座標與上述 dy 值（魚實體的 y 座標）差距的絕對值小於魚實體高度一半。前者若為 true 表示已被釣中，避免重複釣同一條魚；後者為觸控點 y 座標在魚實體高度範圍內即表示被釣中。

行 32 ～ 36：被釣中的判斷式成立則該魚被釣中，如果魚實體的 x 座標小於行 28 的預設值，將該值更改為被釣中魚實體的 x 座標（行 33），由於釣線實體的註冊點在最右邊位置，視覺將可呈現釣線由釣竿到魚的長度，不會穿過魚實體。這是左邊河岸的情況，右邊河岸則不相同。另外將迴圈值記錄為被釣中魚實體的編號（行 34），再將釣中魚指標更改為 true（行 36）。

行 39：將釣線實體的 x 座標設定為上述處理後的 min 值。

行 40 ～ 54：迴圈值 2，即右邊河岸實體被按下的情況，釣竿與釣線實體將以右邊為基準出現。

行 41 ～ 42：從右邊河岸出現，因此釣竿與釣線實體應旋轉 -180 度，即水平翻轉，將註冊點更改至水平方向的另一邊。

行 43：設定 Number 類型變數 max，將做為釣線實體的 x 座標，預設值為 0，這是以右邊為基準出現的預設值，必須視有無釣中魚決定最後數據。

行 44 ～ 53：使用迴圈檢視全部魚實體是否有被釣中，與上述左邊河岸情況類似，只有魚實體的 x 座標與預設值的比較不同。

行 54：將釣線實體的 x 座標設定為上述處理後的 max 值。

行 56 ～ 58：左右兩邊處理後，如果釣中魚指標為 true，將記錄編號之魚實體播放至「釣中」影格，顯示該魚被釣中的視覺效果。

行 59 ～ 60：分別將觸控點的 x 與 y 座標設定為目前編號釣竿實體的 x 與 y 座標，使其在該位置出現。

行 61：告知目前編號釣竿實體開始播放，圖像將顯示持續一段時間之後再恢復消失的狀態。

行 62：將觸控點的 y 座標設定為目前編號釣線實體的 y 座標，使其在該垂直位置出現。

行 63：告知目前編號釣線實體開始播放，圖像將顯示持續一段時間之後再恢復消失的狀態。

行 64：釣竿編號遞增 1，下回應使用另一編號的釣竿與釣線實體。

行 65：如果釣竿編號大於 2，令其等於 1，因為只有兩個觸控點可使用。

▶ 影格 5，「遊戲結束」

計時結束自動播放至此處，遊戲結束。

```
01      myTimer.stop();
02      updateScore();
03      this.removeEventListener(TouchEvent.TOUCH_BEGIN, clickBtn);
04      reset_mc.addEventListener(TouchEvent.TOUCH_BEGIN, gameReset);
05      stop();
06
07      function gameReset(event:TouchEvent) {
08          this.gotoAndPlay("遊戲開始");
09      }
```

解說

行 01：計時器實體 myTimer 停止播放，不再計時。

行 02：呼叫 updateScore() 函數，檢視並更新最佳成績記錄。

行 03：移除註冊之觸控按下事件偵聽處理函數 clickBtn()。

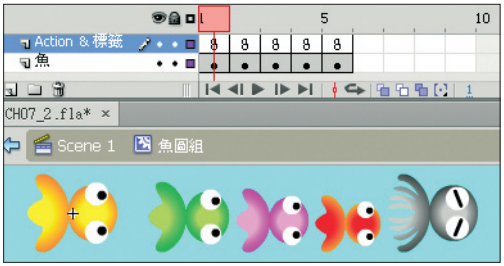
行 04：在再玩實體 reset_btn 註冊觸控按下事件偵聽處理函數 gameReset()，使其被按下後具有再玩遊戲的功能。

行 05：時間軸停止播放，等待玩家的觸控事件。

行 07 ～ 09：自訂觸控事件偵聽處理函數 gameReset()，處理 reset_btn 實體被按下後的程序，時間軸返回「遊戲開始」影格再玩遊戲。

7-6-4 特殊影片片段

魚圖組影片片段



置放於游走的魚元件的不同分數魚圖像的實體 fish_mc。每一影格置放一個魚的圖像，Action & 標籤圖層置放該魚的得分設定。

► 影格 1

```
01    var score:int = 1;
02    stop();
```

解說

行 01：設定得分數值，int 類型變數。其他影格指定數值即可，不可重複宣告類型。

行 02：停止播放，其他影格不需此行。

游走的魚影片片段

擁有獨立的時間軸影格程序，與 4-4 節飛舞的蜜蜂影片片段類似，流程大致分為三階段，並設定 Boolean 類型變數 catch_flag 做為是否可被釣中的指標。第一階段為隨機等待迴圈，魚的圖像由完全透明逐漸出現，製造消失後再次出現的隨機間隔時間。第二階段設定移動的隨機新位置，並在影格迴圈內移動至該位置，如果沒有被釣中則再產生新位置重複此階段移動，只有此階段可以被釣中，因此 catch_flag 為 false。第二階段為被釣中後逐漸消失的動畫過程。與小蜜蜂相同，程式碼有魚移動時旋轉角度的設計，因此魚的圖像務必比照上圖，頭的方向向右。

魚圖層置放魚實體 fish_mc，被釣中後建立逐漸透明消失的補間動畫效果；得分圖層置放被釣中後魚的分數動態文字 score_txt；釣中標示圖層置放被釣中後製造視覺效果的圖形。