歡樂拼圖

拼圖遊戲是可以讓您消磨許多時間的，以下就為您介紹如何使用 Flash MX 來製作拼圖遊戲，為了增加遊戲時的困難度，我們讓每一片拼圖都可以旋轉，還可以有統計遊戲的時間。製作原理很簡單，但是過程比較瑣碎，完成後保證會有很大的成就感。

實做練習 \ 本書範例 \ch16\ 歡樂拼圖.fla

先使用 PhotoImpact 和 Fireworks 處理拼圖所需要的圖片，最後再將物件匯入到 Flash MX 來設計拼圖遊戲。

注意事項：

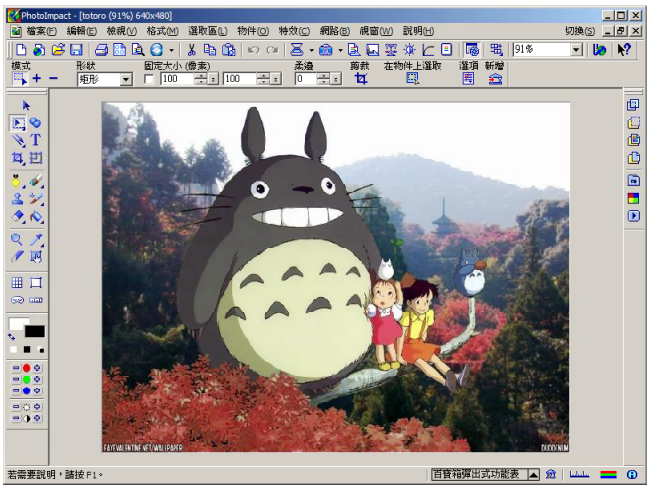
製作本遊戲時，除了 Flash MX 以外，您還需要使用以下的軟體來輔助，才可以完成。

Ulead PhotoImpact- 分割圖片。

Macromedia Fireworks- 將圖片轉換為 SWF 格式。

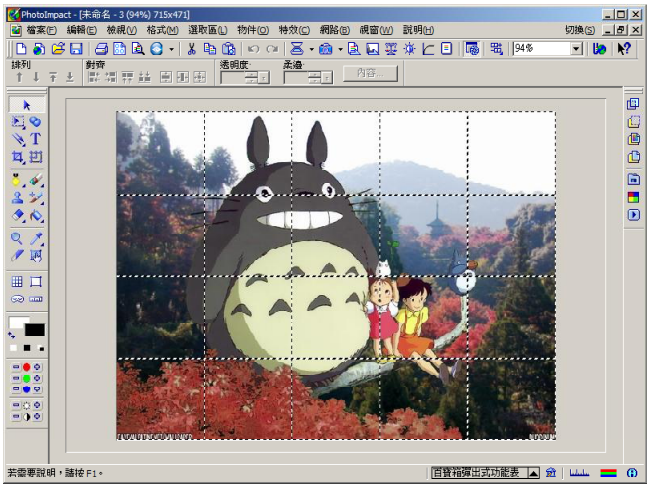
製作拼圖片

1. 使用 PhotoImpact 8.0 中文版開啟要製作拼圖的圖片。

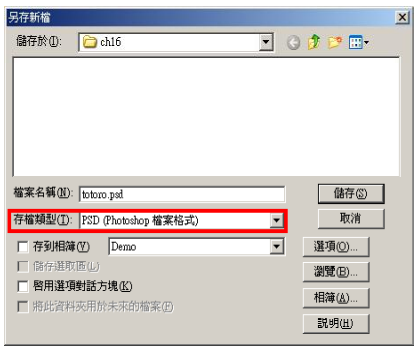


2. 執行「網路＼格線與分割區」指令，參考下圖內容來設定圖片的分割選項。

3. 從畫面上可以看到分割後的結果。不滿意時，可以按「Ctrl+Z」重新執行步驟 2 重新設定分割的內容。



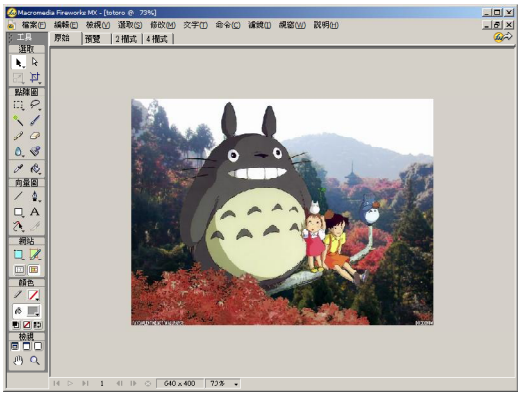
4. 最後執行「檔案＼另存新檔」指令，將檔案存為 PSD 格式。



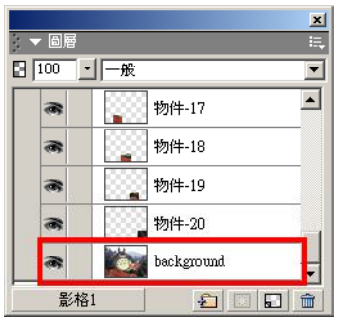
PSD 轉 SWF

在這裡各位一定有疑問，為何不直接將分割好的圖檔匯入到 Flash MX 當中，答案是，經過 Fireworks 處理後，圖片會按照原來在 PhotoImpact 分割的狀態，完整無誤的顯示在 Flash MX 當中。如此可以免除在 Flash MX 當中排列圖片的麻煩。

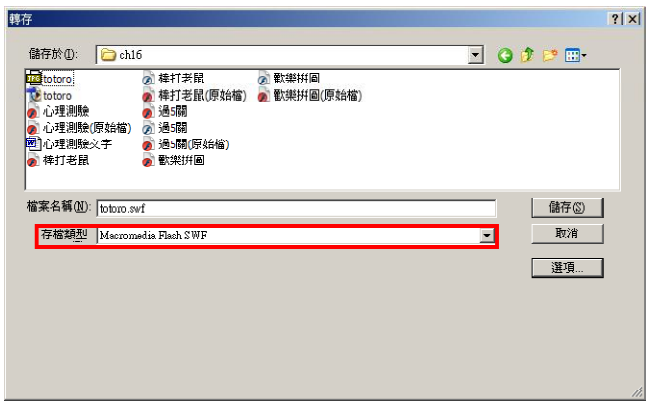
1. 使用 Fireworks MX 開啟剛剛存檔的 PSD 檔案。



2. 從「圖層(F2)」面板中，選取最下層的「background」圖層，按「Delete」鍵將該圖層刪除。



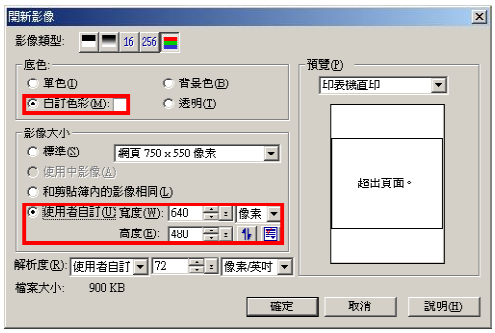
3. 執行「檔案＼轉存」指令，將圖檔轉存為「Macromedia Flash SWF」格式。



框線

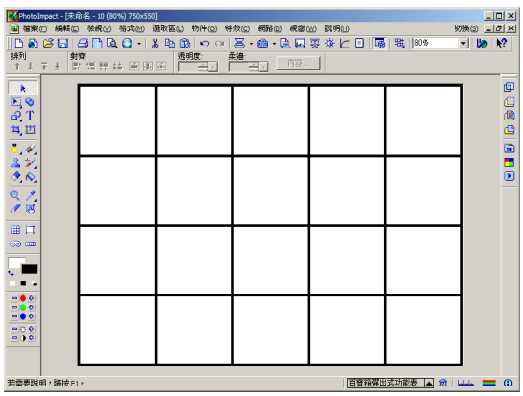
製作拼圖的框線，方便在遊戲中作為參考的基準。

1. 在 PhotoImpact 當中，建立一個新的圖檔，圖檔的大小需要和拼圖大小一樣。



2. 執行「網路＼格線與分割區」指令，分割的方式需要和拼圖片一樣，其餘參考下圖的說明。

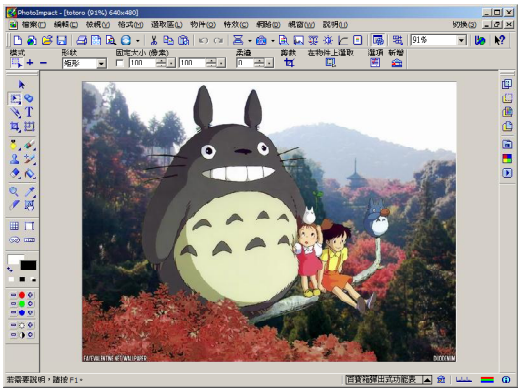
3. 下圖就是分割後的結果，最後將圖檔存為 JPEG 格式。



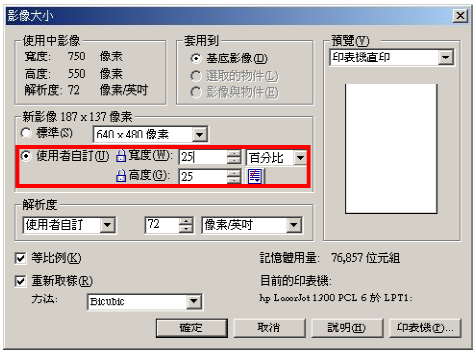
縮圖

製作拼圖的縮圖，方便在遊戲中作為參考的基準。

1. 使用 PhotoImpact 8.0 中文版開啟要製作拼圖的圖片。



2. 按「Ctrl+G」鍵，來重新調整圖片大小。如下圖將圖片縮小為原來的 25% 。



3. 最後執行「檔案＼另存新檔」指令，將圖片存為 JPEG 格式。

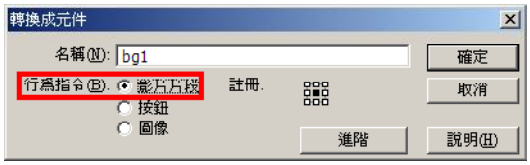
動畫元件製作

元件名稱：bg1 、bg2..bg20

元件類型：影片片段

元件說明：匯入 Fireworks 所轉存的 SWF 檔案。雖然在畫面上看到一整張完整的圖片，事實上選取拼圖時，您會發現每一片拼圖都是獨立的物件。

1. 按「F8」鍵，參考下圖將圖片轉換為動畫元件，來作為拼圖的背景元件。



bg1 puzzle1a	bg2 puzzle2a	bg3 puzzle3a	bg4 puzzle4a	bg5 puzzle5a
bg6 puzzle6a	bg7 puzzle7a	bg8 puzzle8a	bg9 puzzle9a	bg10 puzzle10a
bg11 puzzle11a	bg12 puzzle12a	bg13 puzzle13a	bg14 puzzle14a	bg15 puzzle15a
bg16 puzzle16a	bg17 puzzle17a	bg18 puzzle18a	bg19 puzzle19a	bg20 puzzle20a

藍色：元件名稱紅色：元件實體名稱

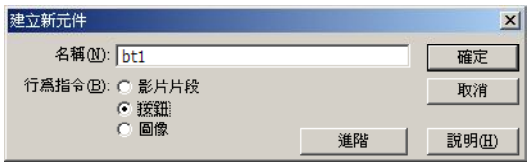
2. 請選取每一片拼圖，參考上圖在屬性面板中設定替每個元件設定實體名稱。

元件名稱： bt1 、 bt2..bt20

元件類型： 按鈕

元件說明： 新增按鈕元件，將背景元件引入，使元件具有互動的功能。

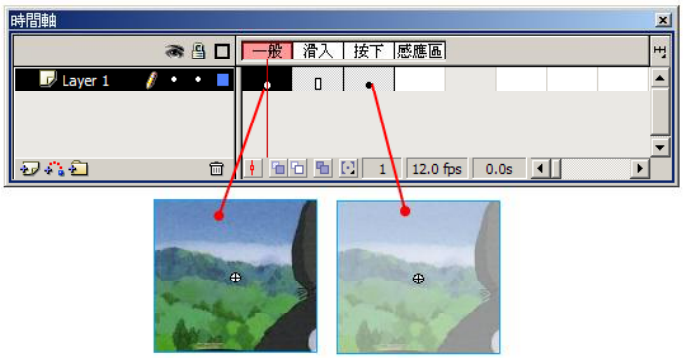
1. 新增按鈕元件「bt1」，從元件庫當中將「bg1」元件引入「一般」影格。



bt1 bg1	bt2 bg2	bt3 bg3	bt4 bg4	bt5 bg5
bt6 bg6	bt7 bg7	bt8 bg8	bt9 bg9	bt10 bg10
bt11 bg11	bt12 bg12	bt13 bg13	bt14 bg14	bt15 bg15
bt16 bg16	bt17 bg17	bt18 bg18	bt19 bg19	bt20 bg20

藍色：元件名稱 紅色：引入的元件

2. 選取「按下」影格，按「F6」鍵新增關鍵影格，選取影格中的元件，在屬性面板中設定 Alpha 值為 50% 。如此可以製作出按下拼圖圖片時，呈現透明的效果，以方便和其它的拼圖分別。



3. 重複步驟 1-2 的動作，依序完成其它的按鈕元件。

元件名稱：puzzle1、puzzle2..puzzle20

元件類型：影片片段

元件說明：說明如下：

1. 新增影片片段元件，將按鈕元件引入。元件對應的按鈕請參考下圖的說明。

puzzle1 bt1	puzzle2 bt2	puzzle3 bt3	puzzle4 bt4	puzzle5 bt5
puzzle6 bt6	puzzle7 bt7	puzzle8 bt8	puzzle9 bt9	puzzle10 bt10
puzzle11 bt11	puzzle12 bt12	puzzle13 bt13	puzzle14 bt14	puzzle15 bt15
puzzle16 bt16	puzzle17 bt17	puzzle18 bt18	puzzle19 bt19	puzzle20 bt20

藍色：元件名稱 紅色：引入的元件

2. 選取按鈕元件，加入以下的動作：

```
on (press) {
startDrag("", true);
```

滑鼠按下時，開啟拖曳所有的元件。也就是讓按下滑鼠時就可以拖曳拼圖圖片。

```
}
on (release) {
stopDrag();
```

滑鼠放開後，停止拖曳元件。也就是讓放開滑鼠時就停止對拼圖圖片的拖曳。

```
if ("/"+_name+"a" eq _droptarget and _rotation == 0) {
```

檢查拖曳的拼圖圖片的實體名稱部分字串是否和「背景圖層」的元件相同及旋轉角度是否為 0 。例如：puzzle（拼圖片）就對應到 puzzle1a（背景），來確定拼圖的位置是否正確，如果為真，就接著進行以下的動作。

```
snd = new Sound();
snd.attachSound("puzzle_sound");
snd.start();
```

播放 puzzle_sound 音效。此音效在使用前必須在元件庫當中的音效檔案上，指定 Link_id ，才可以使用動作來控制音效放。

```
setProperty(_droptarget, _alpha, "100");
```

將「背景」圖層上的對應元件的 Alpha 值調為 100 ，配合下一個指令，製作出拼圖顯示在正確位置的效果。

```
setProperty("", _x, "2000");
```

將「拼圖」圖層上的元件放置到場景以外，所以 _x 的數值，必須大於場景的寬度。

```
_root.finish = _root.finish+1;
```

每完成一張拼圖時 finish 的數值就會加 1 。

```
if (_root.finish == 20) {
```

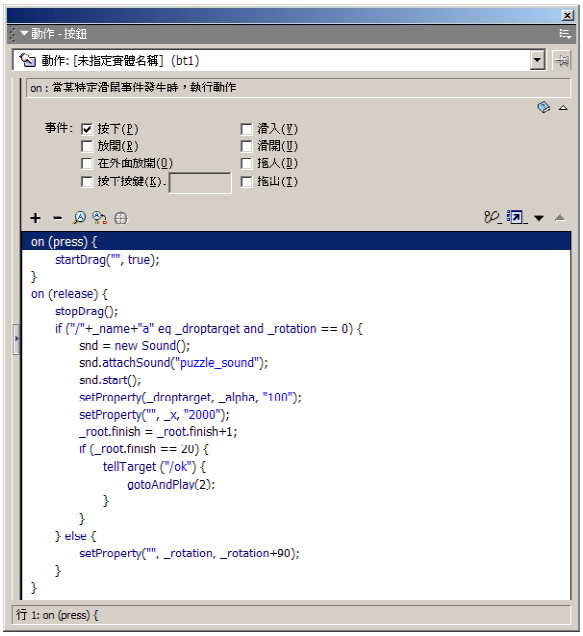
如果 finish 等於 20 時，也就是拼圖完成。 finish 的數值和拼圖的片數有關。

```
tellTarget ("/ok") {
gotoAndPlay(2);
```

播放完成的動畫。

```
            }
        }
    } else {
        setProperty("", _rotation, _rotation+90);
    }
}
```

如果拼圖的位置錯誤，就將拼圖旋轉 90 度。



3. 重複步驟 1-2 的動作，依序完成其它的元件。

元件名稱： 開始、離開

元件類型： 按鈕

元件說明： 製作如下圖的按鈕元件。

元件名稱： 完成

元件類型： 圖像

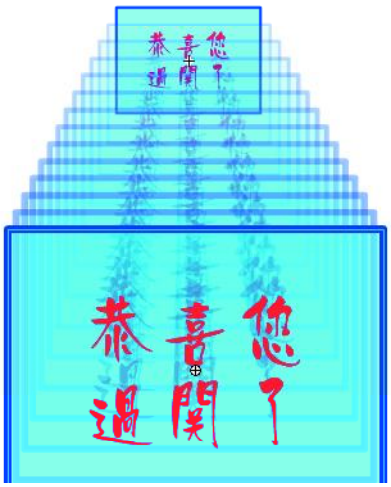
元件說明： 製作如下圖的元件。

元件名稱： 完成動畫

元件類型： 影片片段

元件說明： 說明如下。

1. 選取第 2 個影格按「F6」鍵新增關鍵影格，將「完成」元件引入，製作一段動畫。



2. 選取第 1 個影格，在影格中加入以下的動作：

```
stop();
```

並設定影格標籤為「ok_start」。

3. 選取第 2 個影格，在影格中指定音效。

4. 選取最後一個影格，在影格中加入以下的動作：

```
stop();
```

元件名稱： 參考圖

元件類型： 圖像

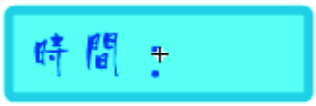
元件說明： 將縮圖圖片引入，製作以下的內容。

元件名稱： 計時器

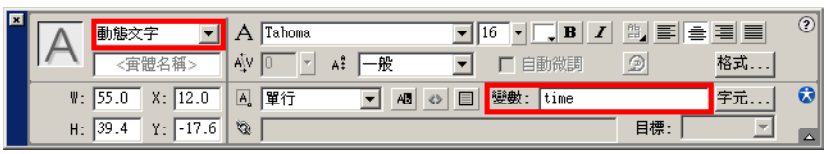
元件類型： 影片片段

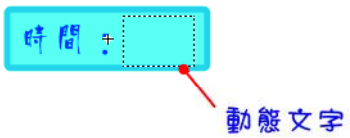
元件說明： 說明如下。

1. 製作如下圖的內容，選取第 13 影格按「F5」鍵新增影格。



2. 使用文字工具，設定動態文字欄位，定義變數名稱為「time」，用來顯示拼圖所使用的時間。





3. 選取第 1 個影格，在影格中加入以下的動作：

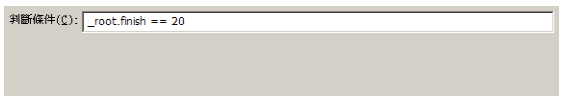
```
time = 0;
```

將 time 變數的值設為 0

4. 選取第 2 個影格，在影格中加入以下的動作：

```
if (_root.finish == 20) {
    stop();
}
```

如果 finish == 20，表示拼圖已經完成，停止計時器的運作。
finish == ?，數值是根據拼圖的片數來決定。



5. 選取第 13 影格，在影格中加入以下的動作：

```
time = time+1;
gotoAndPlay(2);
```

設定 time = time+1，回到第 2 個影格繼續。
從 2-13 影格共包含 12 影格，由於 fps 為 12 影格，所以動畫每循環一次，就等於是一秒的時間，這就是最簡單的計時器了。

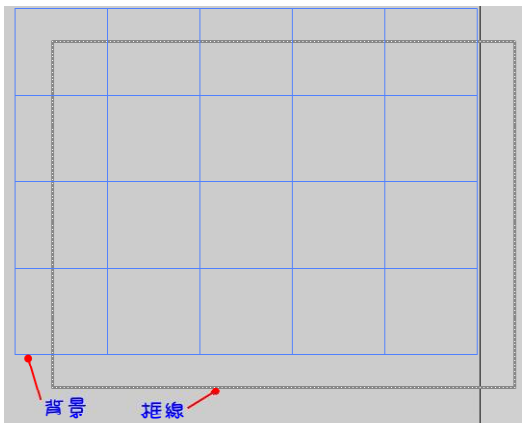
安排場景動畫

步驟 1

將圖層命名為「背景」，選取場景中所有的元件，在屬性面板中，將元件的 Alpha 值設為 0 ，選取第 2 個影格按「F5」新增影格，將動畫延續到第 2 個影格。完成後請將圖層鎖定。

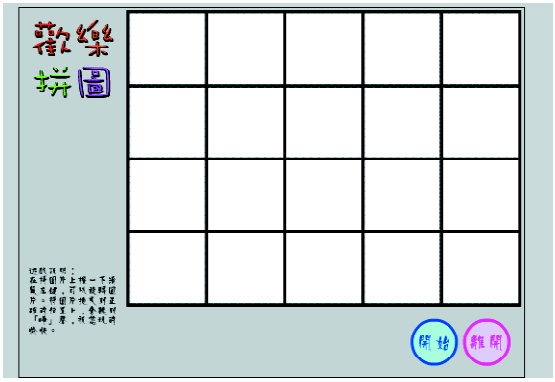
步驟 2

新增「框線」圖層，將圖層移動到「背景」圖層下方，從元件庫當中引入「框線」點陣圖，使用外框檢視的方式，將「框線」和「背景」圖層的位置對齊。



步驟 3

新增「按鈕」圖層，從元件庫當中引入「開始」和「離開」元件，分別在按鈕上加入以下的動作：



「開始」按鈕的動作：

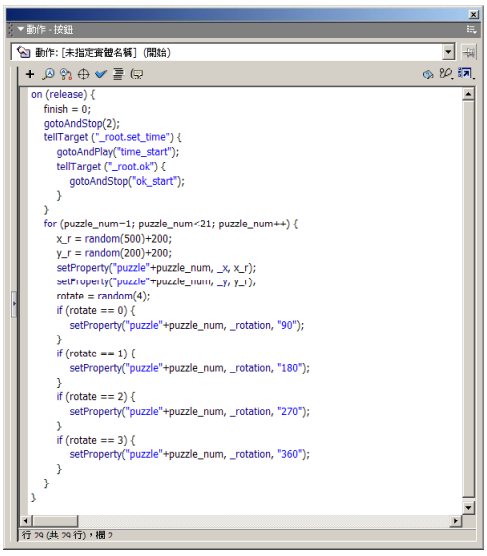
```
on (release) {
滑鼠按下時。
finish = 0;
設定變數 finish=0 ， finish 用來計算拼圖完成的片數。
gotoAndStop(2);
跳到第 2 個影格並停止，第 2 影格計時器才會開始動作。
tellTarget ("_root.set_time") {
gotoAndPlay("time_start");
計時器開始動作。
tellTarget ("_root.ok") {
gotoAndStop("ok_start");
        }
 }
因為 ok 元件的 start 影格是空白的，如果拼圖完成時，就可以清除完成的顯示畫面。
for (puzzle_num=1; puzzle_num<21; puzzle_num++) {
x_r = random(500)+200;
y_r = random(200)+200;
setProperty("puzzle"+puzzle_num, _x, x_r);
```

```
setProperty("puzzle"+puzzle_num, _y, y_r);
```

將 20 片圖片以亂數的方式排列，如此在每次進行遊戲的時候，都會有不同的開始。因為我們使用的拼圖片數為 20 片，所以設定 puzzle_num<21。

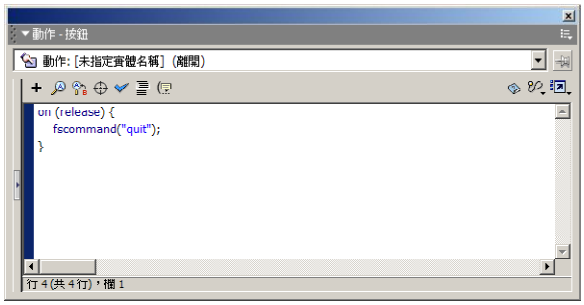
```
rotate = random(4);
if (rotate == 0) {
setProperty("puzzle"+puzzle_num, _rotation, "90");
}
if (rotate == 1) {
setProperty("puzzle"+puzzle_num, _rotation, "180");
        }
if (rotate == 2) {
setProperty("puzzle"+puzzle_num, _rotation, "270");
        }
if (rotate == 3) {
setProperty("puzzle"+puzzle_num, _rotation, "360");
        }
  }
}
```

亂數旋轉拼圖，「0」表示旋轉 90 度、「1」表示旋轉 180 度、「2」表示旋轉 270 度、「3」表示旋轉 360 度（不旋轉）。



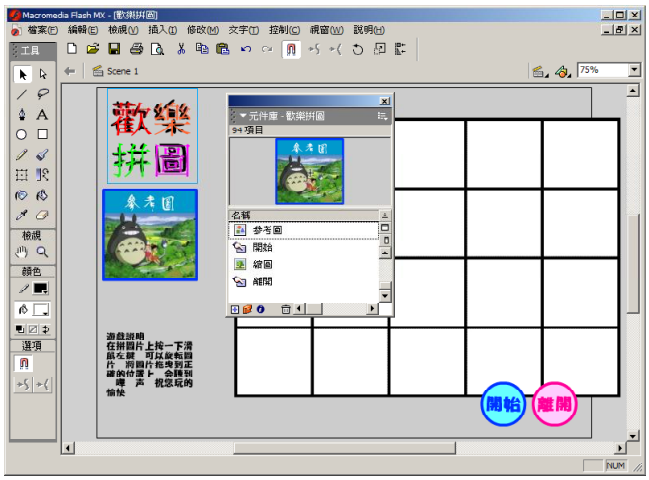
「離開」按鈕的動作：

```
on (release) {
滑鼠按下時
fscommand("quit");
}
離開遊戲並關閉視窗。
```



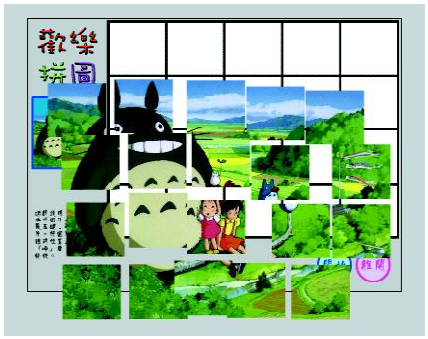
步驟 4

新增「縮圖」圖層，從元件庫當中引入「參考圖」元件。



步驟 5

新增「拼圖」圖層，從元件庫當中引入 puzzle1-puzzle20 元件，隨意的放置在場景當中。選取每一個元件，在屬性面板中設定實體名稱，在這裡我們將實體名稱和元件名稱設成一樣。

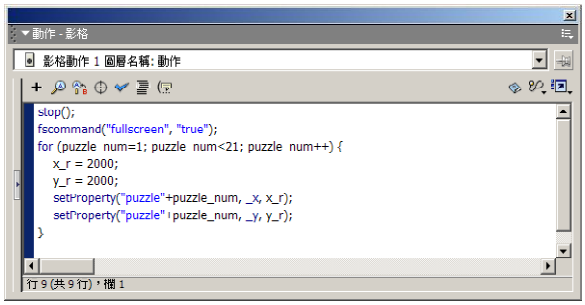


步驟 6

新增「動作」圖層，在第 1 個影格中加入以下的動作：

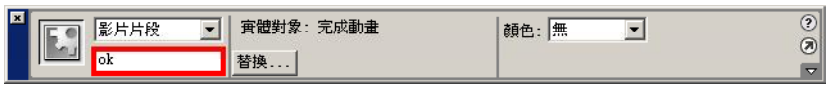
```
stop();
停止。
fscommand("fullscreen", "true");
使用全螢幕來進行遊戲。
for (puzzle_num=1; puzzle_num<21; puzzle_num++) {
x_r = 2000;
y_r = 2000;
setProperty("puzzle"+puzzle_num, _x, x_r);
setProperty("puzzle"+puzzle_num, _y, y_r);
}
```

在遊戲開始以前，將所有的拼圖圖片移到場景外，所以在這裡的 x_r 和 y_r 的數值要大於場景的尺寸。



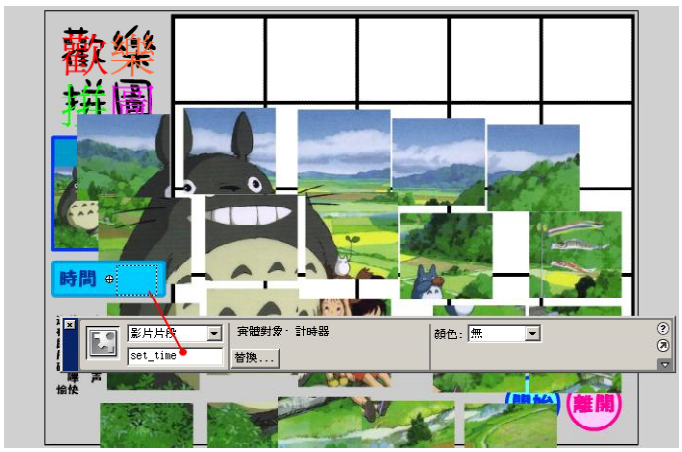
步驟 7

新增「過關」圖層，將「完成動畫」元件引入場景，由於此元件的第 1 個影格是沒有內容，所以只會顯示空心圓圈，請在屬性面板中將實體名稱設為「ok」。



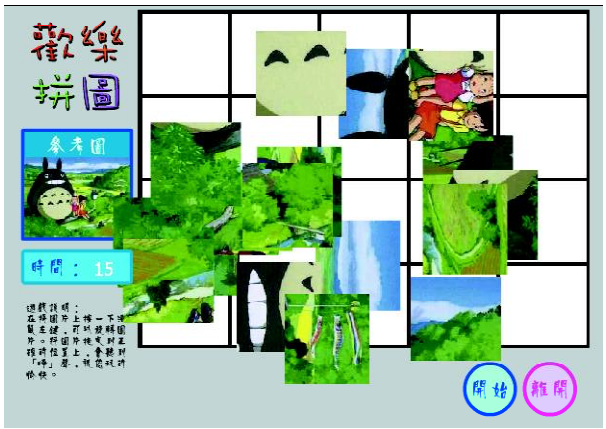
步驟 8

新增「計時器」圖層，選取第 2 個影格按「F7」新增空白關鍵影格，將「計時器」元件引入，在屬性面板中將實體名稱設為「set_time」。



步驟 9

完成了，最後按「Ctrl+Enter」鍵來測試一下吧。



棒打老鼠

相信許多人一定玩過打老鼠的遊戲，當老鼠從洞裡冒出來的那一刻，您就會拿著 500 公斤的大榔頭，猛力一敲，就把老鼠打昏了。遊戲雖然單純，但可是會訓練您的反應能力呢？

實做練習 \ 本書範例 \ch16\ 棒打老鼠.fla

遊戲開始時，您總共有 30 秒鐘的時間，用大棒槌將地洞中冒出來的老鼠打暈，打中一隻，就可以得到 50 分，看看誰的手腳快，在時間內，打中最多隻的老鼠。

動畫元件製作

元件名稱：老鼠
元件類型：圖像
元件說明：繪製如右圖的老鼠。

元件名稱：老鼠被打
元件類型：圖像
元件說明：複製「老鼠」元件，修改成老鼠被扁的樣子。

元件名稱：隱形按鈕
元件類型：按鈕
元件說明：直接選取「感應區」影格，按「F7」新增空白關鍵影格，在編輯區當中繪製圓形區域，並填上顏色，作為「隱形按鈕」的感應區。

元件名稱： 槌子

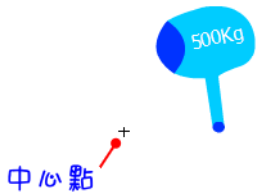
元件類型： 影片片段

元件說明： 製作槌子動畫，說明如下。

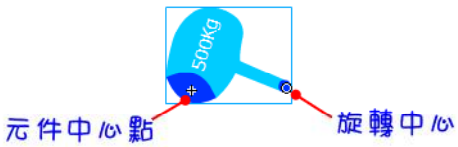
1. 在第 1 個影格中繪製槌子，並將物件群組（調整元件的中心點，以方便在下一個影格，讓元件作旋轉的動作）。在第 1 個影格當中，加入以下的動作指令。

stop ();

停止在第 1 個影格。



2. 選取第 2 個影格，按「F6」新增關鍵影格，將槌子旋轉，變成打下的樣子。選取第 6 個影格按「F5」新增影格，延長槌子打下的時間。



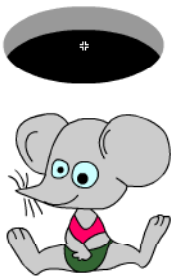
元件名稱： 老鼠移動

元件類型： 影片片段

元件說明： 參考以下的說明，製作出老鼠從地洞中出現的動畫。

1. 在「黑洞」圖層中，繪製橢圓形填上黑色，作為老鼠的洞穴，選取第 24 影格按「F5」新增影格。

2. 新增「老鼠」圖層，引入「老鼠」元件，選取第 5 個影格，按「F6」新增關鍵影格，在第 1-5 影格製作「老鼠」元件由下往上移動到洞口的動畫。

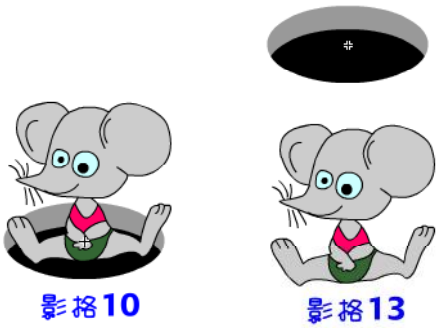


影格1

影格5

3. 選取第 9 影格，按「F5」新增影格，延長「老鼠」停留在洞口的時間。

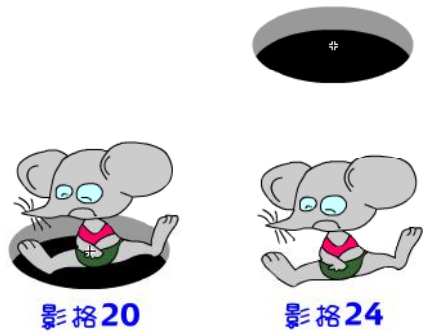
4. 分別選取第 10 和 13 影格，按「F6」新增關鍵影格，製作「老鼠」元件跑回洞內的動畫。



5. 選取第 14 影格，按「F7」新增空白關鍵影格，從圖庫當中將「老鼠被打」元件引入，放置在洞口上方。

6. 選取第 16 影格，按「F5」新增影格，延長「老鼠被打」元件在洞口的時間。

7. 分別選取第 20 和 24 影格，按「F6」新增關鍵影格，製作「老鼠被打」元件回到洞內的動畫。

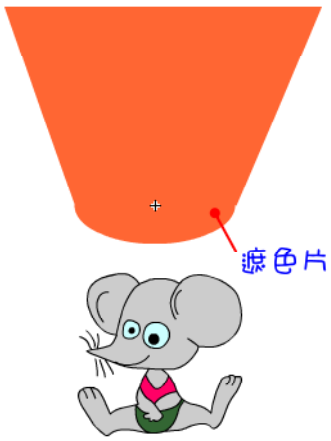


8. 選取第 1 個影格，在屬性面板中設定標籤名稱為「play」。選取第 14 個影格，設定影格的標籤名稱為「hit」。

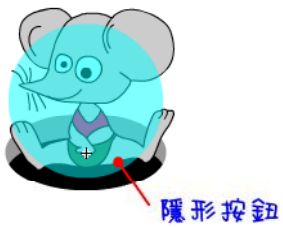
9. 選取第 13 個影格，在影格當中，加入以下的動作指令。
gotoAndStop (1);
播放「老鼠」出現在洞口的動畫後，回到第 1 個影格並停止。

10. 選取第 24 個影格，在影格當中，加入以下的動作指令。
gotoAndStop (1);
播放「老鼠被打」跑回洞裡的動畫後，回到第 1 個影格並停止。

11. 新增「遮色片」圖層，在洞口上方老鼠出現的範圍，繪製封閉區域並填上顏色。完成之後，選取「遮色片」圖層，按滑鼠右鍵，執行「遮色片」指令，將圖層設為遮色片。如此可以讓老鼠出現在「遮色片」圖層，所填上顏色的區域中。



12. 新增「隱形按鈕」圖層，選取第 6 個影格，按「F7」新增空白關鍵影格，將「隱形按鈕」元件引入，放置在洞口，也就是可以打到老鼠的位置。接著在第 10 個影格，按「F5」新增影格，延長打到老鼠的時間。最後在第 11 個影格，按「F7」新增空白關鍵影格，清除「隱形按鈕」元件。所以在第 6-10 影格之間，就是您可以打到老鼠的時間。



13. 選取「隱形按鈕」元件，在按鈕上加入以下的動作指令:

on (press, rollOver) {

當滑鼠按下或經過按鈕時，執行下面的指令。

/:score = Number(/:score)+50;

執行到這裡表示老鼠已被打中，分數自動加上 50 分。

tellTarget ("/hammer") {

gotoAndPlay (2);

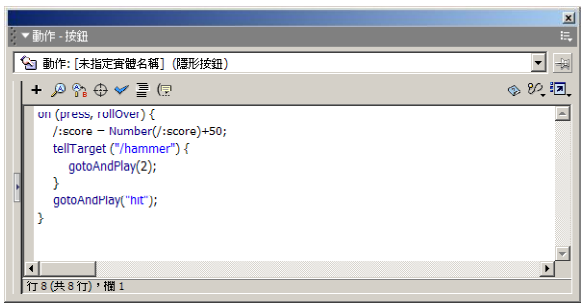
}

播放「hammer」的第 2 個影格。也就是槌子打下去的動作。

gotoAndpPlay ("hit");

}

跳到「hit」影格。播放老鼠被打到後的動畫。

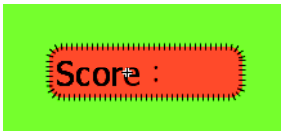


元件名稱：計分板

元件類型：圖像

元件說明：說明如下。

1. 製作計分板。



2. 使用文字工具新增「動態文字」。設定變數名稱為「score」。

元件名稱：計時器

元件類型：影片片段

元件說明：計時器用來控制遊戲進行的時間，時間到遊戲停止。說明如下。

1. 在「背景」圖層繪製計時器的外觀，選取第 13 個影格，按「F5」新增影格。這是因為電影預設播放速度是 12fps，當「計時器」影片片段播放 1 次，約花掉 1 秒鐘的時間（第 1 影格不算）。