Bài 1 : Làm quen với cửa sổ Flash- Tự học lập trình Flash

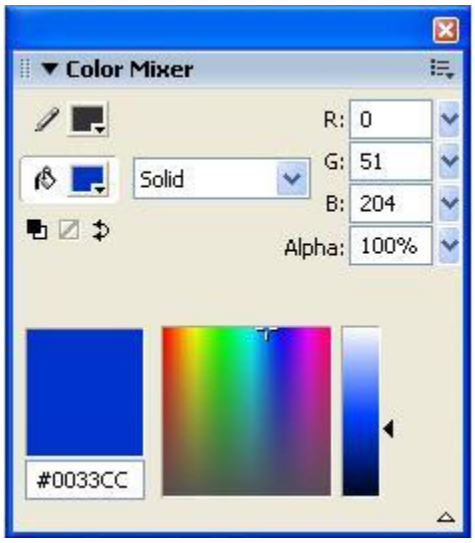
Nếu từng thích thú với những trò chơi trên máy tính, chắc có lúc bạn mơ ước "hành nghề" lập trình trò chơi, nhưng chưa biết bắt đầu từ đâu. Bạn có thể bắt đầu với Flash, công cụ tạo ra trò chơi trên trang Web. Flash giúp dựng nên các hoạt cảnh, trong đó những vật thể, những nhân vật hoạt động theo quy tắc nào đó và có thể có sự thông minh nhất định mà người ta gọi là "trí tuệ nhân tạo" (artificial intelligence). Không chỉ trò chơi, Flash cho phép tạo ra chương trình bất kỳ chạy trên trang Web. Flash là công cụ của người chuyên nghiệp nhưng vẫn "dễ chịu" đối với người "lơ mơ". Chỉ cần chăm chú trong từng bước nhỏ, bạn sẽ đi rất xa.

Trước hết, bạn cần cài đặt phần mềm Flash. Để làm quen với Flash, bạn có thể dùng từ phiên bản Flash 7 trở về sau. Flash 7 còn gọi là Flash MX 2004, chạy tốt trên máy tính của năm… 2004. Máy tính cũ vẫn có thể giúp bạn thu được rất nhiều kỹ năng mới. Những nội dung nâng cao đòi hỏi phải có phiên

bản Flash mới hoặc các công cụ bổ sung (chẳng hạn Flex Builder) sẽ được trình bày sau khi bạn nắm vững phần cơ bản.

Sau khâu cài đặt phần mềm, bạn hãy khởi động Flash. Khi thấy một cửa sổ nhỏ, bạn chọn mục Flash Document ở phần Create New, bắt đầu làm quen với các thứ bên trong cửa sổ Flash.

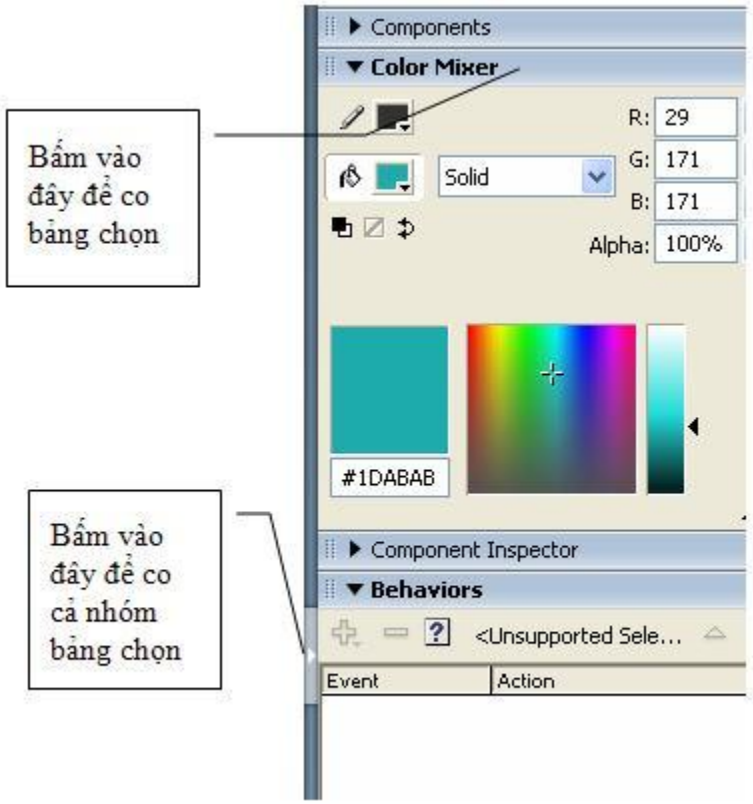
Cửa sổ Flash gồm nhiều bảng chọn (panel), giúp bạn thực hiện những việc khác nhau. Chẳng hạn, hình 1 là bảng chọn Color Mixer, cho phép chọn màu cho hình vẽ. Bạn có thể bấm vào chỗ nào đó trong khung màu để chọn hoặc ghi trị số vào các ô R (Red), G (Green) hoặc B (Blue). Đó là ba trị số xác định màu bất kỳ. Ô Alpha giúp quy định độ trong suốt của màu.



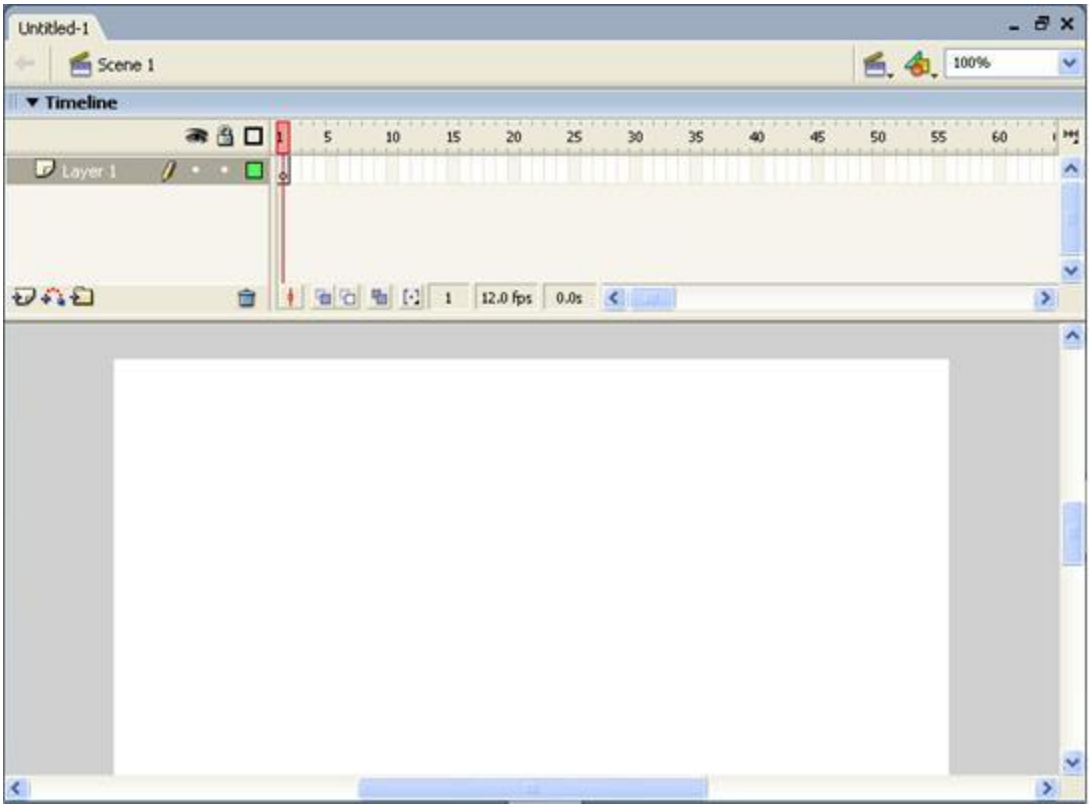
Bạn có thể di chuyển bảng chọn đi đâu cũng được, để thuận tiện cho công việc. Muốn vậy, bạn chỉ cần trỏ vào góc trên, bên trái của bảng chọn , giữ phím trái của chuột, kéo đi. Muốn bảng chọn “đậu” vào chỗ cũ, bạn cũng làm giống vậy: “nắm kéo” góc trên, bên trái của bảng chọn, đưa nó về vị trí cố định bên phải cửa sổ Flash.

Vì có khá nhiều bảng chọn nên khi chưa dùng đến bảng chọn nào đó, bạn nên cho nó “co lại” bằng cách bấm một phát vào thanh tiêu đề của nó. Khi cần dùng bảng chọn, muốn cho nó “bung ra”, bạn lại bấm một phát vào thanh tiêu đề.

Bạn có thể “dẹp” cả nhóm bảng chọn bên hông cửa sổ Flash bằng cách bấm vào dấu mũi tên màu trắng ở “cạnh sườn” của nhóm (hình 2). Muốn cho nhóm bảng chọn đó “bung ra”, bạn bấm vào dấu mũi tên trắng lần nữa.



Bạn thấy ở giữa cửa sổ Flash có một khung trống trơn, màu trắng (hình 3). Đó chính là sân khấu (stage), nơi mà bạn sẽ xây dựng hoạt cảnh, thể hiện các ý tưởng sáng tạo. Ta có thể tạo nên các hình vẽ trên sân khấu, có thể co dãn và di chuyển chúng tùy ý. Người ta gọi chúng là các đối tượng (object).



Bạn để ý đến bảng chọn nằm ở bên trái cửa sổ Flash, thường được gọi là bảng công cụ (tool panel) hay hộp công cụ (tool box), giúp bạn vẽ và sửa hình. Bạn thử vẽ một quả banh nha. Bạn hãy bấm vào hình tròn ở hộp công cụ (hoặc gõ phím chữ O). Đó là biểu tượng của công cụ Oval Tool để vẽ hình ô-van. (hình tròn là trường hợp riêng của hình ô-van thôi). Đưa con trỏ chuột vào sân khấu, bạn thấy nó đổi dạng thành dấu thập. Bạn trỏ vào đâu đó trên sân khấu, giữ phím Shift và kéo chuột (giữ phím trái của chuột khi di chuyển) để vẽ ra hình tròn (bạn giữ phím Shift để vẽ được hình tròn thay vì hình ô-van).

Bạn bấm vào công cụ chọn ở hộp công cụ (hoặc gõ phím chữ V), trỏ vào đâu đó phía trên, bên trái hình tròn vừa vẽ, kéo chuột qua phải, xuống dưới, nhằm “căng” một khung chọn bao quanh hình tròn. Xong, bạn trỏ vào hình tròn, thử kéo nó đến vị trí khác trên sân khấu.

Vùng màu xám bao quanh sân khấu có thể xem là “hậu trường” của sân khấu. Nếu bạn đặt hình tròn vào vùng xám, nó sẽ không xuất hiện khi “trình diễn”. Khi cần cho đối tượng nào đó xuất hiện, bạn có thể kéo nó từ hậu trường vào sân khấu.

Để lưu hoạt cảnh đã tạo ra (dù lúc này quả banh chưa “nhúc nhích” gì hết), bạn bấm vào File ở thanh trình đơn rồi chọn Save trên trình đơn xổ xuống (nói vắn tắt là chọn File > Save). Bạn thấy xuất hiện một hộp thoại (dialog box), giúp bạn chọn thư mục và đặt tên cho tập tin chứa hoạt cảnh, thường gọi là tập tin nguồn của Flash (Flash Document). Tập tin nguồn Flash có tên phân loại là fla.

Bài 2 : Nhân vật và thể hiện trong lập trình Flash

Mỗi khi khởi động Flash, bạn luôn thấy một cửa sổ nhỏ liệt kê một số tập tin fla mà bạn đã mở xem hoặc tạo ra trong những lần trước. Bạn hãy chọn tập tin có hình quả banh do bạn vẽ.

Bạn trỏ vào quả banh, giữ phím trái của chuột, thử kéo quả banh qua vị trí khác. Có lẽ bạn sẽ ngạc nhiên vì ở vị trí cũ vẫn còn một đường tròn, không có màu tô. Đó chính là nét viền của hình tròn được tạo ra bởi công cụ Oval Tool. Bạn có thể quy định màu riêng cho nét viền của hình tròn. Ta chỉ cần quả banh không có nét viền, bạn hãy xóa nét viền bằng cách bấm vào nét viền để chọn và gõ phím Delete.

Để “tô màu” cho quả banh, bạn bấm vào quả banh để chọn, bấm vào ô Fill Color trong bảng Color Mixer (tỏ ý rằng bạn muốn quy định màu tô chứ không phải màu nét) và chọn màu bạn thích.

Quả banh hiện có của bạn chỉ là một hình vẽ đơn thuần. Muốn cho quả banh trở thành một đối tượng có hành vi riêng, có thể điều khiển bằng cách lập trình, trước hết bạn cần chuyển đổi quả banh thành một nhân vật (movie clip). Với quả banh đang ở tình trạng "được chọn", bạn chỉ cần gõ phím F8. Khi thấy hộp thoại Convert to Symbol (hình 1), bạn gõ Ball (quả banh) để đặt tên cho nhân vật sắp được tạo ra. Bạn cũng nên chọn điểm mốc của nhân vật ở giữa cạnh dưới khung bao của hình quả banh bằng cách bấm vào ô giữa, dưới cùng trong chín ô vuông nhỏ của mục Registration. Sau này, khi nói về tọa độ của nhân vật, bạn hiểu đó là tọa độ của điểm mốc.

Sau khi bạn bấm nút OK trong hộp thoại Convert to Symbol, hình quả banh trở thành một nhân vật. Khác với những hình vẽ bình thường trong Flash, nhân vật được lưu trữ trong thư viện (library). Ấn Ctrl+L (hoặc gõ phím F11), bạn thấy bảng Library xuất hiện bên phải cửa sổ Flash, trong đó có tên nhân vật Ball, hiện là nhân vật đầu tiên trong danh sách các thứ được lưu trữ trong thư viện. Bấm vào tên Ball trong bảng Library, bạn thấy "chân dung" của nhân vật Ball hiện ra phía trên danh sách.

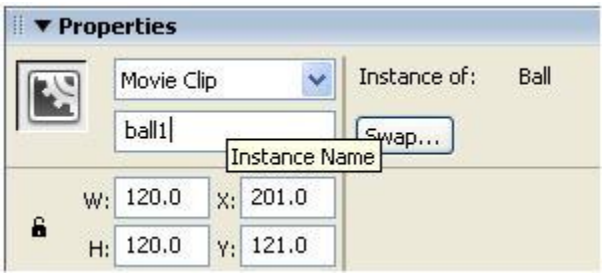
Thư viện còn là nơi lưu trữ nhiều thứ khác, chẳng hạn những âm thanh cần dùng cho chương trình Flash của bạn.

Sau thao tác chuyển đổi hình vẽ quả banh thành nhân vật, quả banh trên sân khấu dường như không có gì thay đổi. Thực ra đã có thay đổi lớn: quả banh

trên sân khấu chỉ là một thể hiện (instance) của nhân vật Ball trong thư viện. Muốn có nhiều quả banh, bạn có thể tạo ra nhiều thể hiện khác của nhân vật Ball. Hoạt cảnh của bạn có thể có rất nhiều quả banh nhưng dung lượng tập tin fla tăng không đáng kể so với trường hợp chỉ có một quả banh. Ngoài ra, nếu bạn chỉnh sửa chi đó cho nhân vật Ball trong thư viện, mọi thể hiện của nhân vật đó đều thay đổi. Cấu trúc nhân vật-thể hiện đem đến những lợi ích tuyệt vời!

Trước mắt, điều nên làm ngay là đặt tên riêng cho thể hiện đầu tiên của nhân vật Ball. Với quả banh trên sân khấu đang ở tình trạng “được chọn”, bạn bấm vào thanh tiêu đề Properties ở cạnh dưới cửa sổ Flash (hoặc ấn Ctrl+F3) để mở bảng Properties (hình 2), nơi trình bày các thuộc tính của thể hiện đang được chọn. Trong bảng Properties, bạn bấm vào ô có dòng chữ <Instance Name> (tên của thể hiện) và gõ tên ball1 (ngụ ý: đây là thể hiện đầu tiên của nhân vật Ball).

Nếu trên sân khấu chỉ có một thể hiện duy nhất của nhân vật, bạn có thể gọi thể hiện đó là “nhân vật”. Nhiều người luôn dùng thuật ngữ “nhân vật” thay cho “thể hiện”. Không sao hết, miễn là bạn hiểu khái niệm.

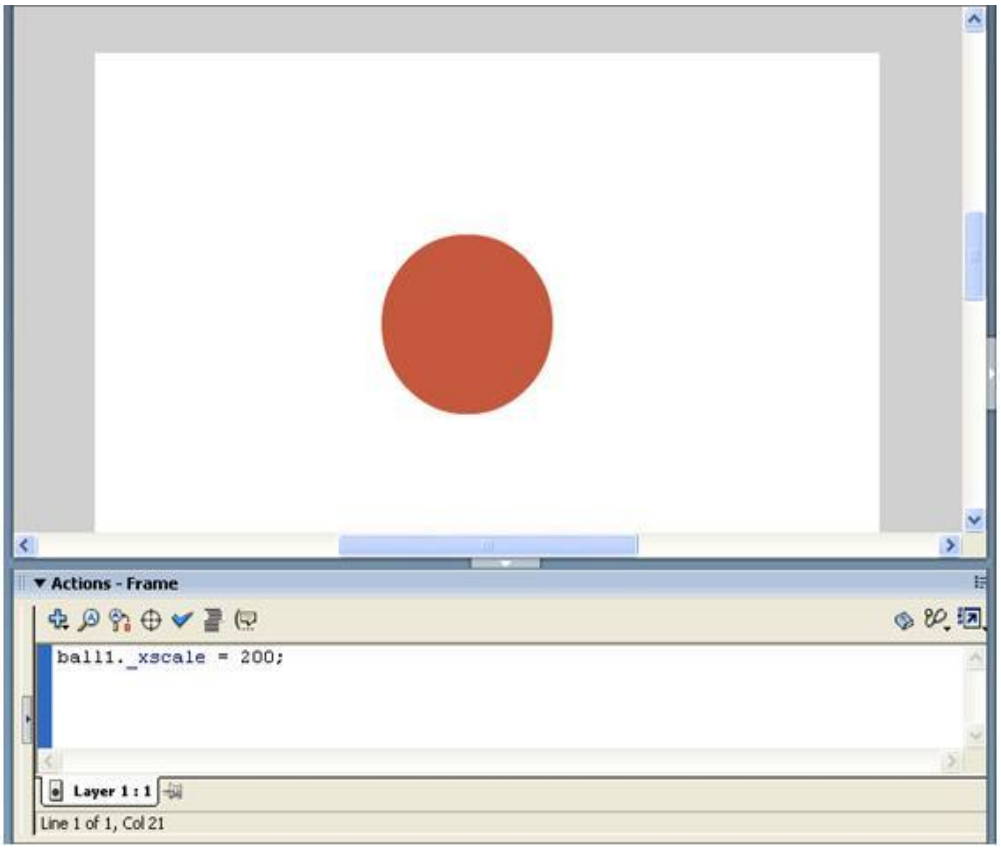


Bấm vào thanh tiêu đề Properties để dẹp bảng Properties, bấm vào đâu đó trên sân khấu để thôi chọn thể hiện ball1, bạn bắt đầu việc lập trình cách bấm vào thanh tiêu đề Actions (hoặc gõ phím F9) để mở bảng Actions (hình 3), nơi

dùng để ghi các câu lệnh ActionScript. ActionScript là tên của ngôn ngữ lập trình dùng để tạo ra kịch bản cho những nhân vật Flash. Bấm vào khung soạn thảo còn trống trơn của bảng Actions, bạn gõ câu lệnh:

?

```
1 ball1._xscale = 200;
```



Câu lệnh vừa nêu làm cho thuộc tính _xscale của nhân vật ball1 có trị số là 200 (gán trị số 200 cho biến _xscale của nhân vật ball1). Điều này nghĩa là nhân vật ball1 được kéo dãn 200% theo phương x (phương ngang).

Để “chạy chương trình” (chương trình hiện chỉ có một câu lệnh), bạn ấn Ctrl+Enter. Cửa sổ Flash lập tức thay đổi, nhiều thứ được giấu đi. Giữa cửa sổ là một khung trắng có hình quả banh bầu dục, cho thấy rõ ràng câu lệnh bạn viết đã được thực hiện “nghiêm túc”: quả banh được kéo dãn theo

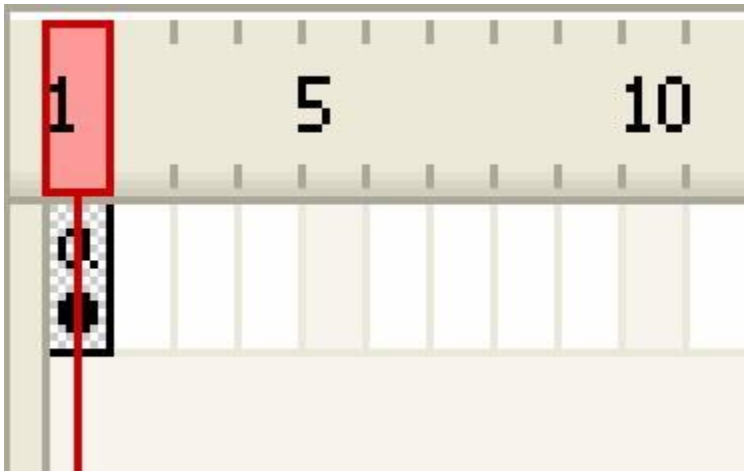
phương ngang. Kích thước theo phương ngang của quả banh bầu dục gấp đôi đường kính của quả banh tròn ban đầu.

Để dẹp khung trắng hiển thị kết quả của chương trình, bạn bấm nút ở góc trên, bên phải khung trắng.

Bài 3 :Khung chốt của hoạt cảnh trong lập trình Flash

Bạn đã viết câu lệnh đầu tiên trong Flash
ball1._xscale = 200;
và chạy thử chương trình chỉ có một câu lệnh ấy. Trong câu lệnh đó, giữa tên nhân vật ball1 và tên biến _xscale có dấu chấm để phân cách. Biến _xscale là biến có sẵn trong mọi nhân vật của Flash. Vế trái của câu lệnh đọc là "biến _xscale của nhân vật ball1".
Toàn bộ câu lệnh đọc là "gán trị số 200 cho biến _xscale của nhân vật ball1". Khi không sợ nhầm lẫn, ta có thể gọi "thể hiện ball1 của nhân vật Ball" là "nhân vật ball1". Bạn chú ý, cần có dấu chấm phẩy (;) để kết thúc câu lệnh.
Xem lại thư mục chứa tập tin fla, bạn thấy có thêm tập tin mới thuộc loại swf. Nếu tập tin chứa quả banh của bạn có tên Ball.fla, tập tin mới được tạo ra có tên Ball.swf (chỉ khác phần phân loại). Trước đây, khi bạn ấn Ctrl+Enter, Flash đã ghi xuống tập tin chương trình swf và chạy chương trình đó. Chính tập tin swf mới là chương trình chạy được, có thể dùng trong trang Web. Việc chuyển đổi tập tin fla thành tập tin swf chạy được gọi là biên dịch (compile). Đối với bạn, swf là một chương trình. Đối với người xem trang Web, swf là một hoạt cảnh (animation) hoặc trò chơi (game).
Phía trên sân khấu, trong bảng Timeline, bạn thấy có một thanh dài với các số 1, 5, 10,… Đó là số thứ tự của các khung hình (frame), gọi tắt là khung. Diễn

biến của chương trình trong Flash theo dòng thời gian giống như một đoạn phim, nên cũng có khái niệm “khung hình”. Mỗi ô ở ngay dưới thanh ghi số thứ tự biểu diễn một khung hình. Dải ô như vậy gọi là thời tuyến (timeline). Trong bảng Timeline, bạn thấy ở số 1 có một ô màu đỏ hồng (hình 1). Người ta gọi đó là đầu đọc (playhead). Đầu đọc đang ở khung 1 cho biết bạn đang xem khung 1 (những gì hiện có trên sân khấu là nội dung của khung 1). Trong ô tương ứng với khung 1 có chữ a be bé, ngụ ý nói rằng có câu lệnh ActionScript được ghi ở khung 1.



Nội dung của khung 1 mặc nhiên được duy trì trong các khung tiếp theo trừ khi bạn chủ động thay đổi. Giả sử bạn muốn ở khung 5, quả banh trở về kích thước bình thường, không bị kéo dãn theo phương ngang nữa.

Muốn vậy, trước hết bạn báo cho Flash biết có sự thay đổi ở khung 5 bằng cách bấm-phải vào khung 5 (ta có thể gọi tắt như vậy thay vì nói rõ “ô tương ứng với khung 5″), chọn mục Convert to Keyframes trong trình đơn vừa hiện ra. Thao tác này chuyển đổi khung thường thành khung then chốt (keyframe), gọi tắt là khung chốt, giúp Flash hiểu rằng phải cẩn thận xem xét lại mọi thứ ở khung 5 vì nó khác với khung trước.

Chính bạn tạo ra sự khác biệt ở khung 5 bằng cách bấm vào khung soạn thảo của bảng Actions (nếu không thấy nó, bạn gõ phím F9) và gõ câu lệnh:

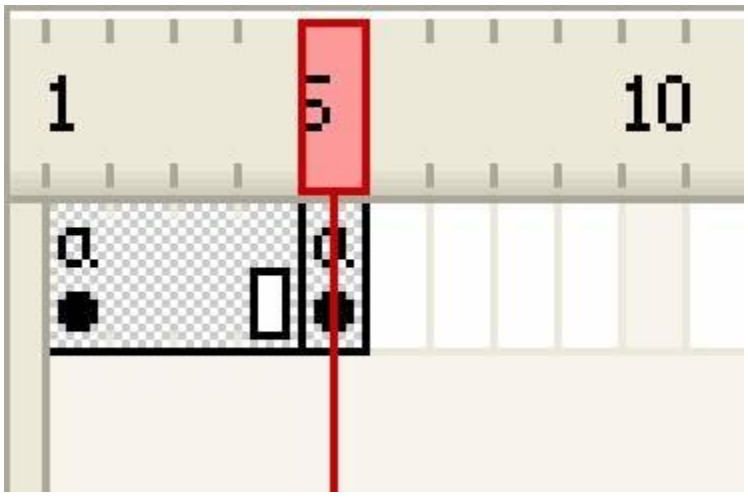
?

```
1 ball1._xscale = 100;
```

Câu lệnh như vậy gán trị 100 cho biến _xscale của nhân vật ball1, quy định độ co dãn theo phương ngang của ball1 là 100%, tức là không co dãn gì nữa. Vào lúc chạy chương trình, khi hiển thị đến khung 5, Flash sẽ thi hành câu lệnh mà bạn vừa viết.

Quan sát thời tuyến, bạn thấy đầu đọc đã nhảy đến khung 5 (hình 2), nghĩa là những gì bạn thấy trên sân khấu thuộc về khung 5. Tại khung 5 có dấu tròn màu đen cho biết đó là khung chốt. Khung 5 cũng có chữ a giống như khung 1 để nói rằng có câu lệnh ActionScript được viết cho khung 5. Nhận ra dấu tròn đen ở khung 1, có lẽ bạn thắc mắc ngay: “Vậy khung 1 cũng là khung chốt?”. Vâng, đúng vậy, khung 1 luôn luôn là khung chốt theo mặc định.

Khung thường ngay trước khung chốt được đánh dấu bằng một dấu chữ nhật màu trắng. Dấu như vậy cho thấy rõ sự kết thúc của một loạt khung giống nhau.



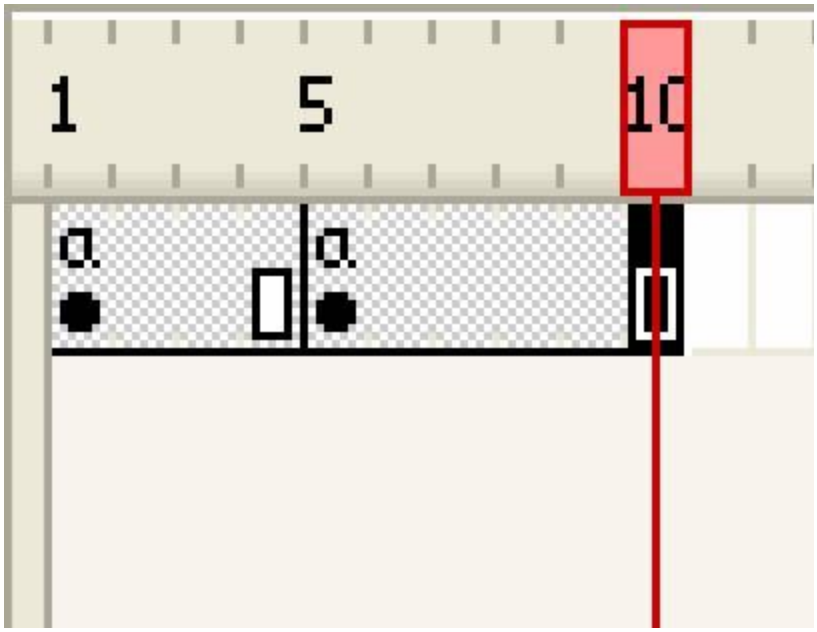
Ấn Ctrl+Enter để biên dịch và chạy chương trình, bạn thấy quả banh “phập phồng” liên tục do câu lệnh ActionScript ở khung 1 và khung 5 được thi hành lặp đi lặp lại. Đóng cửa sổ Ball.swf (cửa sổ trình diễn hoạt cảnh) vừa mở, rồi gõ phím F12 (hoặc chọn File > Publish Preview > Default – HTML), bạn thấy

cửa sổ trình duyệt xuất hiện, trình bày tập tin Ball.html. Quả banh của bạn “diễn trò” thoải mái trong cửa sổ trình duyệt. Đó là nhờ tập tin Ball.html có chứa liên kết trỏ đến tập tin Ball.swf. Điều này giúp bạn hình dung hoạt cảnh của mình trông ra sao khi được đặt trong trang Web.
Xem xong, bạn đóng cửa sổ trình duyệt, trở về cửa sổ Flash, ấn Ctrl+S để lưu lại thành quả của mình trong tập tin Ball.fla.

Bài 4 : Hành vi của nhân vật – Tự học lập trình Flash

Bạn hãy mở lại tập tin fla chứa quả banh. Do hình ảnh quả banh được kéo dãn theo phương ngang hiển thị trong các khung 1-4, trong khi hình ảnh quả banh bình thường chỉ hiển thị trong một khung duy nhất (khung 5) nên bạn thấy không rõ lắm sự “phập phồng” của quả banh khi chạy hoạt cảnh.

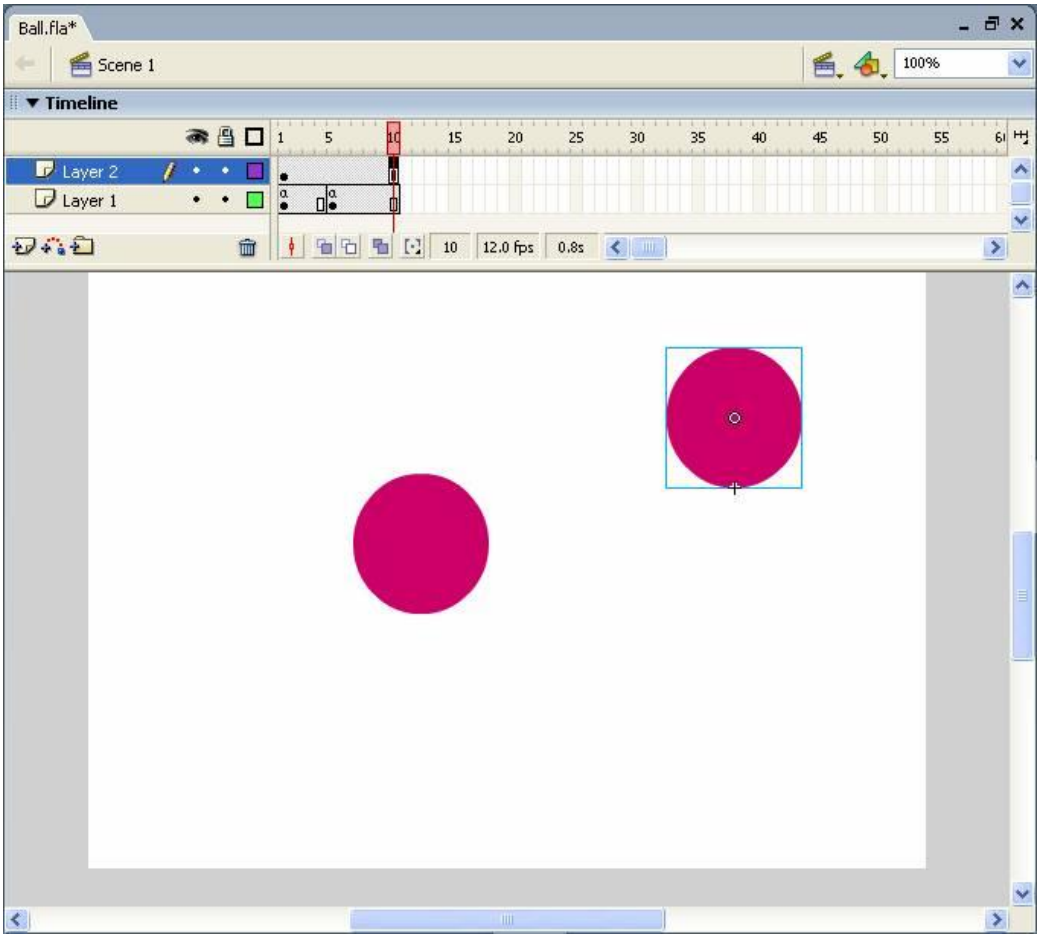
Để mắt có thời gian nhận biết hình ảnh ở khung 5, bạn bấm-phải vào khung 10 trong thời tuyến, chọn mục Insert Frame. Thao tác này tạo thêm các khung 6-10 (hình 1). Đó là các khung thường nằm sau khung 5 (khung chốt) nên có nội dung giống hệt khung 5. Ở khung 10 có dấu chữ nhật màu trắng, đánh dấu sự kết thúc của một loạt khung giống nhau. Hoạt cảnh của bạn giờ đây có tất cả 10 khung. Bạn ấn Ctrl+Enter để xem thử nha.



Đóng cửa sổ trình diễn hoạt cảnh, bạn hãy lấy thêm một quả banh nữa từ thư viện. Cụ thể, bạn gõ phím F11 để mở bảng Library, trỏ vào mục Ball trong bảng đó (hoặc trỏ vào hình quả banh ngay trên mục Ball), kéo nó vào sân khấu. Bạn có ngay thể hiện thứ hai của nhân vật Ball trên sân khấu.

Nhìn vào thời tuyến, bạn thấy các khung 5-10 được tô đen. Đó là cách hiển thị nhắc bạn rằng quả banh thứ hai chỉ hiện diện trong các khung 5-10 giống nhau mà thôi. Thao tác tạo thêm quả banh ở khung 10 không có hiệu lực với loạt khung giống nhau 1-4. Ấn Ctrl+Enter, bạn thấy quả banh thứ hai chớp chớp vì nó không hiện diện trong các khung 1-4.

Để tạo được quả banh thứ hai hiện diện trong mọi khung của hoạt cảnh, không bị “cách trở” bởi sự hiện diện của các khung chốt, bạn cần có một lớp (layer) mới. Bạn ấn Ctrl+Z để hủy bỏ thao tác tạo quả banh thứ hai vừa thực hiện, bấm vào biểu tượng Insert Layer (góc dưới, bên trái bảng Timeline) để tạo thêm lớp mới mang tên Layer 2 nằm bên trên lớp Layer 1 có sẵn. Xong, bạn lại kéo quả banh từ thư viện vào sân khấu (hình 2). Ấn Ctrl+Enter, bạn thấy quả banh thứ hai không chớp chớp như trước nữa vì nó hiện diện trong mọi khung.



Việc giải quyết vấn đề nho nhỏ vừa đặt ra giúp bạn hiểu được cấu trúc lớp của Flash. Hoạt cảnh có thể gồm nhiều lớp chồng lên nhau. Mỗi lớp có những khung chốt riêng biệt.

Quả banh trong lớp Layer 2 hiện thời chưa “nhúc nhích” gì hết. Bạn có thể áp dụng kinh nghiệm từ quả banh thứ nhất cho quả banh thứ hai: đặt tên cho quả banh thứ hai là ball2, tạo khung chốt ở vị trí tùy ý trong thời tuyến của Layer 2 và ghi câu lệnh ActionScript ball2._xscale = … tại các khung chốt.

Tuy nhiên, đó là chuyện cũ, có lẽ không cần lặp lại. Bạn có thể đặt câu hỏi: “Nếu như có hàng chục quả banh trên sân khấu hoạt động gống nhau, lẽ nào cứ phải lập trình lần lượt cho từng quả banh?”. Trong trường hợp như vậy, bạn nên lập trình cho nhân vật trong thư viện thay vì lập trình cho từng thể hiện của nhân vật trên sân khấu. Nhờ vậy, mọi quả banh khi được đưa từ thư viện vào sân khấu đều tự biết “phập phồng”, không cần “chỉ dẫn” gì nữa.

Để thực hiện ý định vừa nêu, trước hết bạn xóa Layer 1 bằng cách chọn Layer 1 trong cửa sổ Timeline và bấm vào biểu tượng “thùng rác” . Làm như vậy nghĩa là xóa đi quả banh thứ nhất và mọi “diễn xuất” của nó, chỉ để lại quả banh thứ hai “chưa biết diễn xuất”.

Để lập trình cho nhân vật Ball, bạn bấm kép vào hình quả banh trong bảng Library. Sân khấu biến mất, thay vào đó là nhân vật Ball trên nền trắng. Mọi việc bạn sắp làm chỉ tác động vào nhân vật Ball. Tuy nhiên, nếu thấy nhân vật Ball đang ở trong tình trạng “được chọn” (trông mờ mờ), bạn bấm vào nền trắng để “thôi chọn”.

Bạn gõ phím F9 để mở bảng Actions, gõ câu lệnh cho khung 1:

```
1|
_xscale = 200;

1|
```

Khác với trước, trong câu lệnh vừa nêu ta không cần ghi tên của thể hiện nào cả vì ở đây câu lệnh được viết “bên trong” nhân vật Ball, Flash tự động hiểu rằng biến _xscale là thuộc tính của nhân vật Ball. Thời tuyến trong cửa sổ Timeline cũng là của riêng nhân vật Ball.

Như đã từng làm, bạn bấm-phải vào khung 5 trong thời tuyến, chọn Insert Keyframe, gõ câu lệnh trong bảng Actions:

```
1|
_xscale = 100;

1|
```

Bạn bấm-phải vào khung 10, chọn Insert Frame để chèn thêm các khung thường 6-10. Thao tác bạn vừa làm đã quy định xong hành vi của nhân vật. Bạn chọn mục Scene 1 (trên bảng Timeline) để trở về với sân khấu. Đưa vào sân khấu thêm hai quả banh nữa từ thư viện và ấn Ctrl+Enter, bạn thấy cả ba quả banh đều phập phồng như nhau.

Bài 5 : Hành vi của thế hiện – Tự học lập trình Flash

Bạn đã tạo được hoạt cảnh gồm ba quả banh "phập phồng". Đó là ba thể hiện của nhân vật Ball trong thư viện, là nhân vật có hành vi "phập phồng". Tuy nhiên, sẽ có những lúc bạn muốn một thể hiện nào đó có hành vi khác biệt với "đồng loại" của nó. Flash cho phép bạn lập trình cho riêng thể hiện được chọn.

Trong ba quả banh trên sân khấu, bạn hãy bấm chọn một quả banh và gõ phím F9 để mở bảng Actions (hình 1). Bấm vào khung soạn thảo của cửa sổ Actions, bạn gõ nội dung như sau:

```
1 onClipEvent(mouseDown) {
2
3 _y = _y + 20;
4
5 }
```

Những gì bạn ghi trong bảng Actions theo cách như vậy chỉ có hiệu lực đối với thể hiện được chọn. Ở đây không còn là một câu lệnh ActionScript như trước, mà là một đoạn mã ActionScript "bí ẩn" được gọi là hàm (function). Phần đầu tiên onClipEvent là tên hàm. Phần ghi trong cặp dấu ngoặc là đối mục

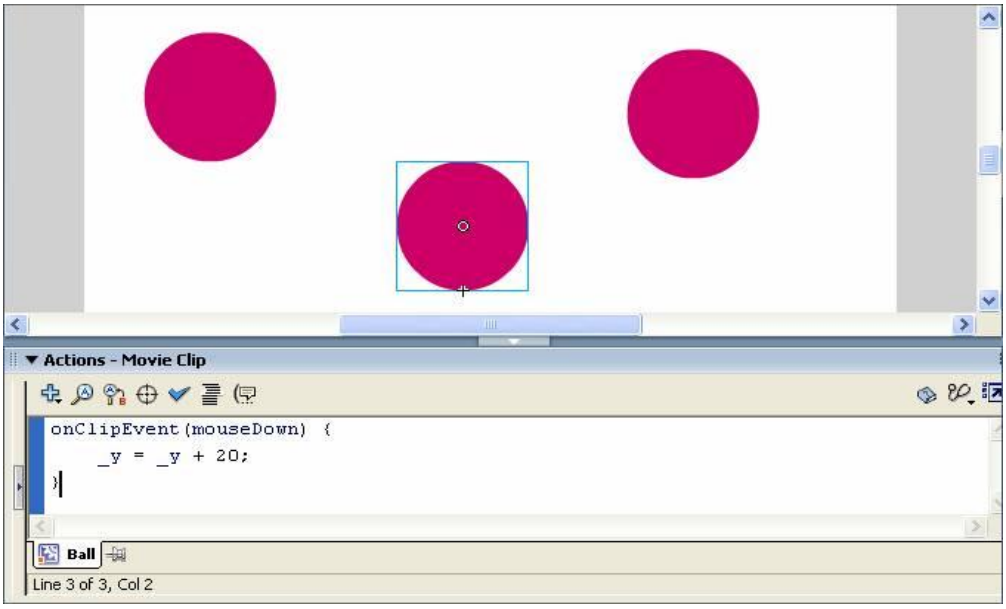
(parameter, argument) của hàm. Đối mục của hàm (ở đây cụ thể là mouseDown) cung cấp thông tin cho hoạt động của hàm. Hoạt động của hàm được diễn đạt bởi thân hàm, là phần nằm trong cặp dấu gộp { }. Phần thân hàm thường được ghi thụt vào trong để dễ phân biệt với tên hàm (điều này không bắt buộc).

Hàm onClipEvent dùng để diễn đạt việc cần làm khi xảy ra một tình huống (event) nào đó đối với thể hiện đang xét. Người ta gọi đó là hàm xử lý tình huống (event handler). Khi lập trình cho thể hiện (quy định hành vi của nó), bạn chỉ được phép viết các hàm xử lý tình huống, không thể viết câu lệnh tùy ý bên ngoài phạm vi của hàm xử lý tình huống.

Với đối mục là mouseDown, hàm onClipEvent trở thành hàm xử lý tình huống "bấm chuột", nghĩa là nó sẽ được thi hành khi ta bấm chuột một phát (bấm phím trái). Phần thân hàm chỉ gồm một câu lệnh _y = _y + 20; nhằm nói rằng "lấy biến _y cộng với 20, rồi lấy kết quả của phép cộng đó gán vào biến _y". Nói cách khác, câu lệnh vừa nêu làm cho trị của biến _y tăng thêm một lượng là 20 điểm ảnh hoặc 20 pi-xôn (pixel).

Biến _y là biến có sẵn bên trong mỗi thể hiện, quy định tung độ của thể hiện trên sân khấu. Dấu gạch dưới trong tên biến _y nhằm nhấn mạnh rằng đây là biến có sẵn, không phải biến do bạn tạo ra.

"Chắc là có cả biến _x bên trong mỗi thể hiện, quy định hoành độ của thể hiện trên sân khấu?". Bạn đoán đúng. Ở đây ta chỉ thay đổi biến _y, không thay đổi biến _x, do đó thể hiện được chọn chỉ dịch chuyển theo phương thẳng đứng.



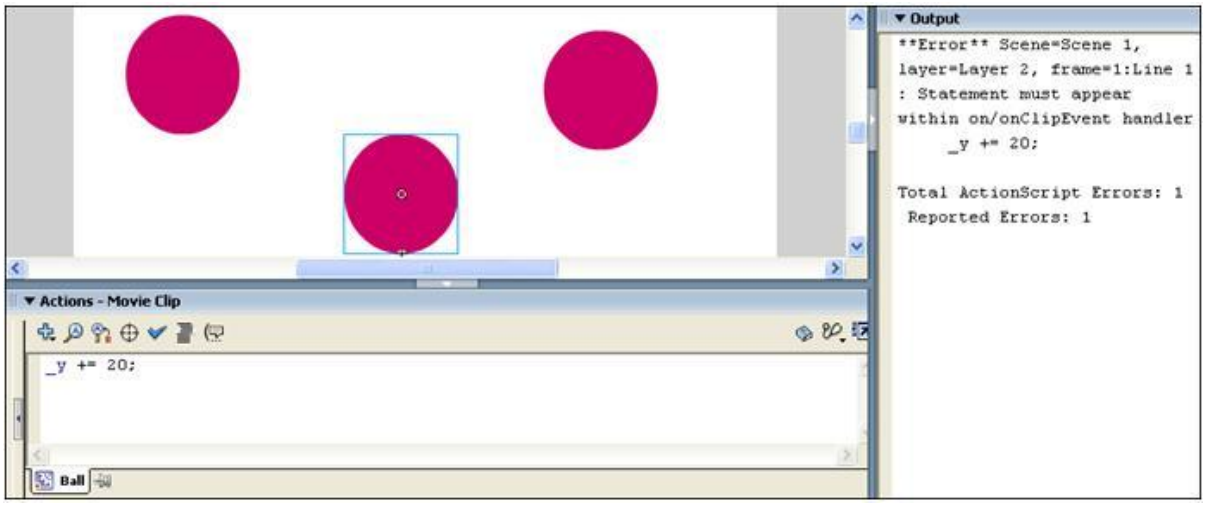
Ấn Ctrl+Enter để chạy chương trình, bạn thấy ba quả banh “phập phồng” như cũ, dường như không có gì mới. Tuy nhiên, khi bạn bấm chuột vào đâu đó trong hoạt cảnh, quả banh vừa được lập trình xê dịch xuống dưới một chút. Cứ mỗi lần bạn bấm chuột, quả banh lại tụt xuống. Điều này không xảy ra với hai quả banh còn lại.

Có lẽ bạn thắc mắc: “Tung độ tăng lên, quả banh phải xê dịch lên trên chứ nhỉ?”. Khác với hệ tọa độ dùng trong toán học phổ thông, hệ tọa độ dùng trong lĩnh vực đồ họa máy tính có trục y hướng xuống dưới (trục x vẫn hướng qua phải). Trong Flash, góc trên, bên trái của sân khấu là gốc của hệ tọa độ. Cách định vị bằng hệ tọa độ như vậy thực ra rất tự nhiên, tựa như khi dò tìm một từ trong văn bản, bạn đọc từ trái qua phải, từ trên xuống dưới.

Để quả banh xê dịch lên trên (ngược chiều trục tung), bạn phải cho tung độ giảm. Bạn có thể sửa câu lệnh trong thân hàm thành _y = _y – 20; (làm cho trị của biến _y giảm đi 20). Thử chạy chương trình, bạn thấy rõ điều đó.

Thay vì viết _y = _y – 20; bạn có thể viết cách khác: y -= 20; . Tương tự, thay vì viết _y = _y + 20; bạn có thể viết: y += 20; . Cũng như dấu gán =, dấu -= và += rất thông dụng khi lập trình. Chúng mô tả thao tác nhất định trên biến nào đó nên được gọi là các tác tử (operator).

“Nếu muốn quả banh tự di chuyển, không cần đợi bấm chuột, chắc là chỉ cần viết câu lệnh _y += 20; thay vì đặt nó vào bên trong hàm xử lý tình huống?”. Khi học lập trình, bạn sẽ có rất nhiều thắc mắc. Những lúc như vậy, bạn cứ mạnh dạn kiểm tra dự đoán của mình. Xóa hàm xử lý tình huống, chỉ ghi câu lệnh đơn giản _y += 20; và ấn Ctrl+Enter, bạn thấy chương trình chạy được, nhưng việc bấm chuột không còn tác dụng như trước. Phía bên phải bỗng xuất hiện bảng Output với những dòng chữ “nhăng nhít” (hình 2). Trong bảng đó, từ đầu tiên Error cho bạn biết chương trình của mình có lỗi. Thông báo lỗi Statement must appear within on/onClipEvent handler nhắc bạn rằng câu lệnh phải được đặt bên trong hàm xử lý tình huống onClipEvent. Đây là quy định bắt buộc khi bạn muốn tạo ra hành vi cho riêng thể hiện được chọn.



Bài 6: Câu lệnh điều kiện – Tự học lập trình Flash

Bạn đã thấy thông báo lỗi xuất hiện ở bảng Output. Khi bạn dừng chương trình (đóng cửa sổ swf), thông báo lỗi vẫn còn lưu lại trong bảng Output ở phía phải cửa sổ Flash để bạn “nghiền ngẫm”.

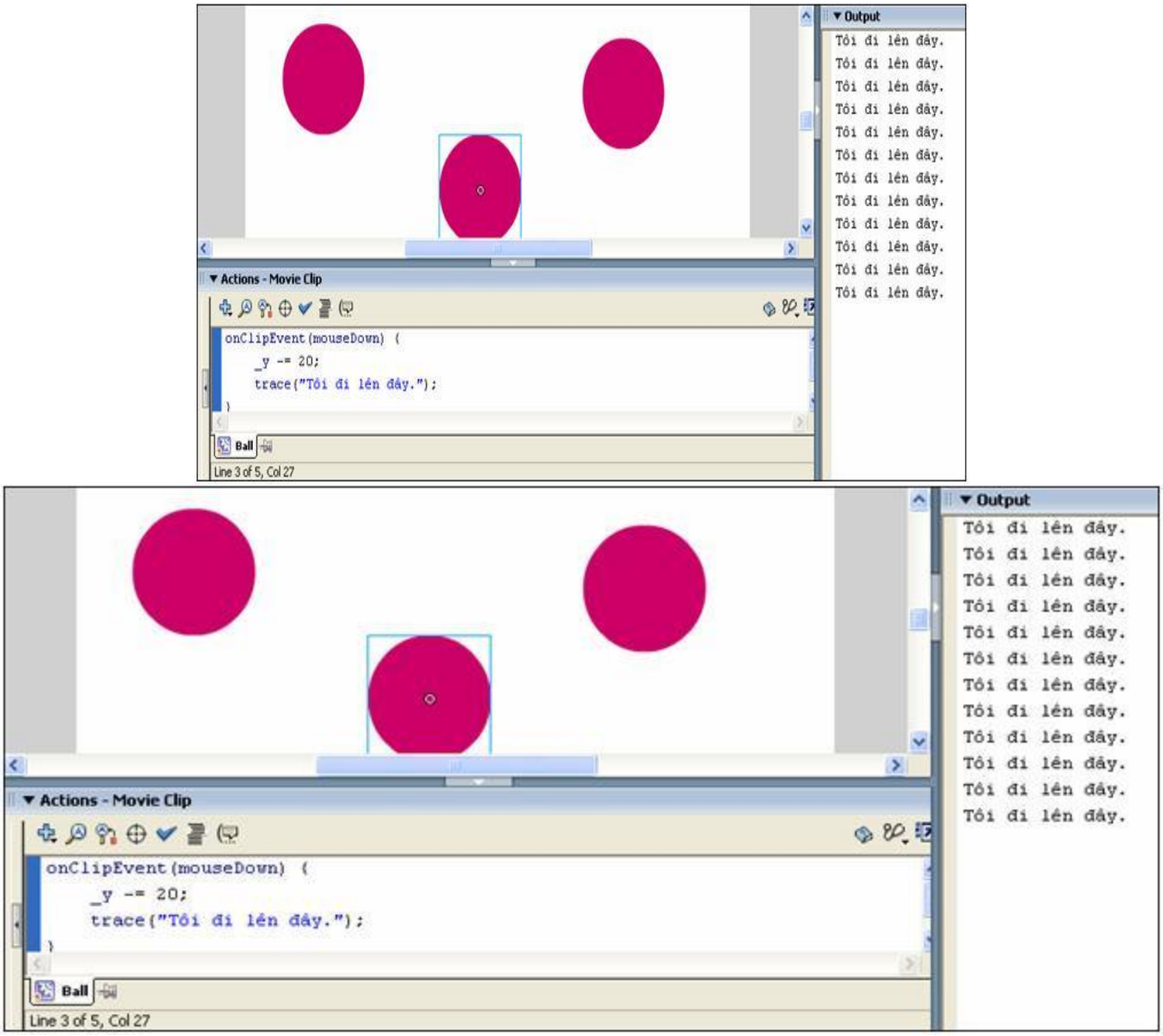
Lỗi như vậy gọi là lỗi lúc biên dịch (compile-time error). Chương trình không chạy được nếu có lỗi lúc biên dịch. Nếu chương trình chạy được nhưng lại tỏ ra “kỳ cục”, không đúng như dự kiến, người ta nói rằng chương trình có lỗi lúc chạy (run-time error). Trong chương trình của mình, bạn có thể chủ động đưa thông báo ra bảng Output. Nhờ

vậy, khi chạy chương trình, bạn dễ dàng theo dõi diễn biến thực tế, dễ dàng phát hiện nguyên nhân gây ra lỗi lúc chạy. Để thử đưa thông báo ra bảng Output, bạn hãy bấm chọn quả banh đang có lỗi biên dịch, gõ phím F9 để mở lại bảng Actions (nếu cần). Trong bảng Actions, bạn xóa câu lệnh sai, gõ đoạn mã mới như sau:

```
onClipEvent(mouseDown) {
_y -= 20;
trace(“Tôi đi lên đây.”);
}
```

Hàm onClipEvent giờ đây có hai câu lệnh. Câu lệnh đầu tiên làm cho quả banh xê dịch lên trên 20 điểm ảnh. Câu lệnh thứ hai là câu lệnh gọi hàm trace, làm cho quả banh biết thông báo về hành động của mình! Thông báo “Tôi đi lên đây” là đối mục của hàm trace. Về mặt cấu trúc, thông báo “Tôi đi lên đây” là một chuỗi ký tự, gọi tắt là chuỗi (string). Dùng bộ gõ UniKey thông dụng, bạn dễ dàng tạo ra ký tự Việt có dấu trong chuỗi. Khi dùng chuỗi, bạn đừng quên mở đầu và kết thúc chuỗi bằng dấu nháy (“).

Ấn Ctrl+Enter để chạy chương trình, bạn thấy mỗi lần bấm chuột, quả banh xê dịch lên trên và thông báo “Tôi đi lên đây” xuất hiện ở bảng Output (hình 1). Bạn hãy bấm chuột liên tiếp nhiều lần để thấy quả banh đi mất tiêu. Đó là khi tung độ _y vượt ra ngoài phạm vi hiển thị của sân khấu.



Để quả banh không đi mất, bạn dừng chương trình, sửa nội dung hàm onClipEvent như sau:

onClipEvent(mouseDown) {

if(_y > 0) {

_y -= 20;

trace(“Tôi đi lên đây.”);

}

}

Những gì bạn vừa viết bên trong hàm onClipEvent là một câu lệnh điều kiện. Câu lệnh ấy ngụ ý: nếu tung độ _y của quả banh lớn hơn 0 thì mới xê dịch quả banh và đưa ra thông báo “Tôi đi lên đây” ở bảng Output. Điều kiện “tung độ _y lớn hơn 0″ phải được viết trong

cặp dấu ngoặc ( ), đặt sau từ if. Những việc cần làm khi điều kiện được thỏa phải được viết bên trong cặp dấu gộp { }, đặt sau cặp dấu ngoặc ( ). Để Flash hiểu được những gì bạn viết, cần tuân thủ nghiêm ngặt các quy tắc vừa nêu.

Chạy lại chương trình và bấm chuột nhiều lần, bạn thấy quả banh rốt cuộc dừng lại, "không thèm" nhúc nhích nữa. Đó là khi điều kiện "tung độ _y lớn hơn 0″ không được thỏa, tức là khi _y nhỏ hơn hoặc bằng 0. Bạn chú ý, tung độ của quả banh là tung độ của điểm mốc. Chính bạn đã quy định điểm mốc nằm ở giữa cạnh dưới khung bao của quả banh.

Nếu muốn quả banh "nói năng" gì đó khi điều kiện "tung độ _y lớn hơn 0″ không được thỏa, chứ không im lìm, bạn viết thêm vào hàm onClipEvent:

```
onClipEvent(mouseDown) {
if(_y > 0) {
_y -= 20;
trace("Tôi đi lên đây.");
}
else {
trace("Tôi không đi nữa.");
}
}
```

Từ else giúp bạn diễn đạt trường hợp ngược với trường hợp được nêu bởi từ if. Việc cần làm khi điều kiện "tung độ _y lớn hơn 0″ không được thỏa phải được ghi bên trong cặp dấu gộp { } theo sau từ else. Nhờ viết như vậy, lúc chạy chương trình, quả banh thông báo "Tôi không đi nữa" nếu tung độ của nó không lớn hơn 0. Lời từ chối đó cho bạn thấy quả banh tuy trơ trơ nhưng vẫn hiểu bạn đang bấm chuột thúc giục nó.

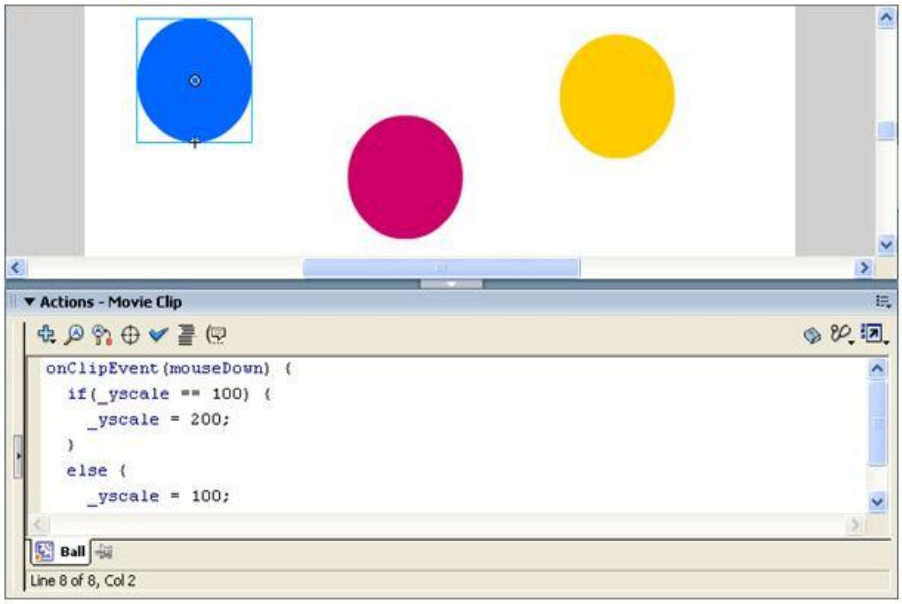
Với hai quả banh còn lại "chưa có cá tính", ta cũng thử lập trình cho chúng có hành vi khác biệt. Trước khi làm như vậy, bạn nên cho hai quả banh còn lại có màu khác cho dễ phân biệt: một xanh, một cam chẳng hạn. Bạn không thể chọn màu tô khác nhau cho các thể hiện khác nhau của một nhân vật nhưng có thể chọn màu nhuộm (tint color) khác nhau cho chúng. Cụ thể, bạn hãy chọn quả banh "chưa có cá tính", ấn Ctrl+F3 để mở cửa sổ Properties tương ứng, chọn Tint trong ô Color, bấm vào ô màu kế bên và chọn

màu trong bảng màu. Muốn màu nhuộm lấn át màu tô, bạn ghi 100% trong ô Tint Amount: . Bằng cách đó, bạn có được ba thể hiện của nhân vật Ball có màu khác nhau (hình 2).

Bạn chọn quả banh màu xanh, gõ đoạn mã như sau trong bảng Actions:

```
onClipEvent(mouseDown) {
if(_yscale == 100) {
_yscale = 200;
}
else {
_yscale = 100;
}
}
```

Để thử đoán nhận hành vi của quả banh màu xanh, bạn cứ chạy chương trình xem sao.



Bài 7 : Hàm xử lý tình huống – Tự học lập trình Flash

Trong hàm xử lý tình huống bấm chuột onClipEvent(mouseDown) của quả banh màu xanh, bạn đã viết câu lệnh điều kiện if(_yscale == 100). Câu lệnh như vậy đặt điều kiện "nếu biến _yscale của quả banh màu xanh có trị bằng 100". Biến _yscale xác định tỉ lệ co dãn theo phương thẳng đứng. Bạn chú ý, dấu "bằng" ở đây (==) khác với dấu "gán" (=) mà bạn từng dùng.

Biểu thức điều kiện _yscale == 100 có thể đúng hoặc sai vào lúc chạy. Người ta còn nói "kiểu cách" hơn một chút: biểu thức _yscale == 100 có thể có trị true hoặc false. Nhờ có hàm onClipEvent(mouseDown) của quả banh màu xanh, mỗi khi bạn bấm chuột vào lúc chạy, quả banh đó chuyển đổi qua lại giữa hai trạng thái: được kéo dãn theo phương thẳng đứng hoặc không.

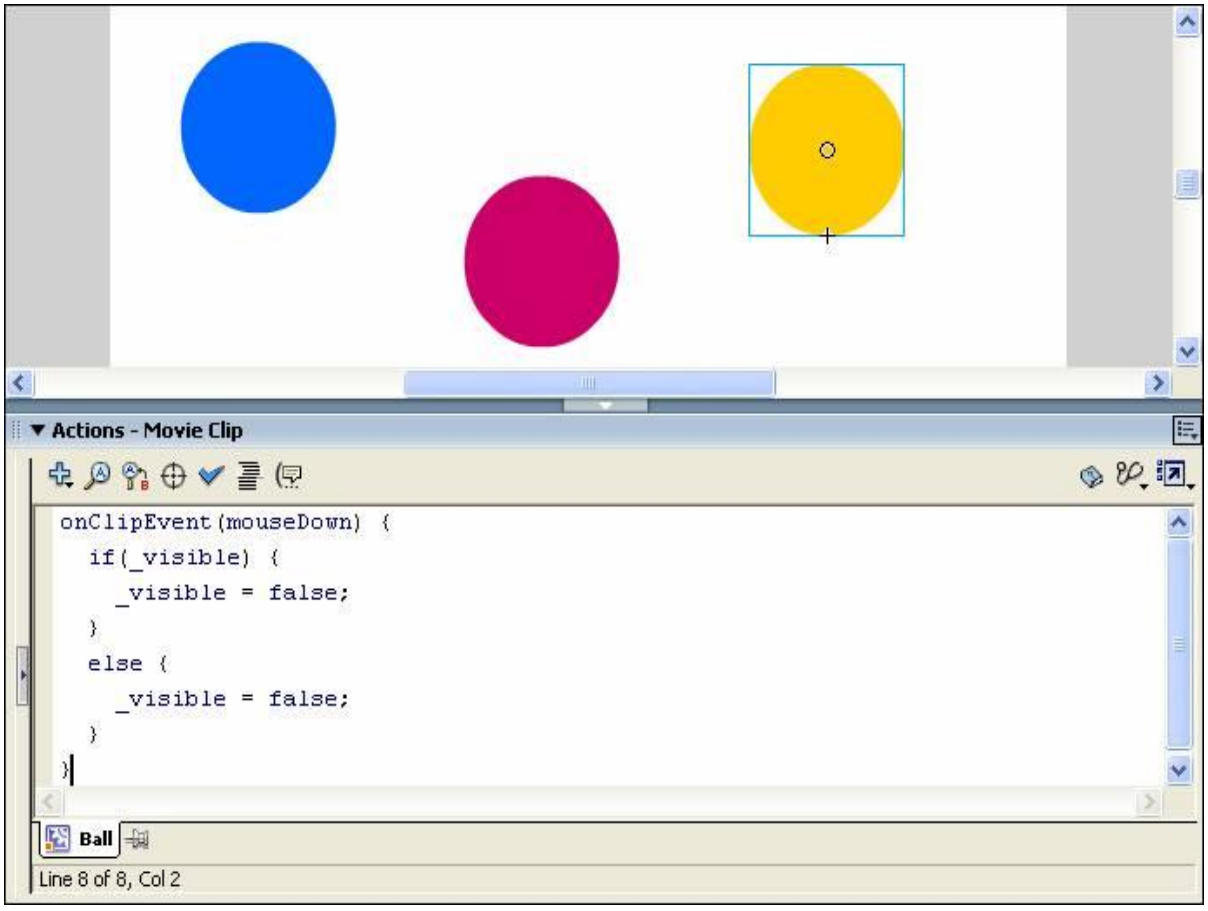
Quả banh màu đỏ và màu xanh đã thể hiện "cá tính" của chúng, ta cũng nên cho quả banh màu cam diễn trò gì đó. Bạn bấm vào quả banh màu cam, gõ phím F9 để mở bảng Actions (hình 1) và gõ đoạn mã như sau:

?

```
1  onClipEvent(mouseDown) {
2
3  if(_visible) {
4
5  _visible = false;
6
7  }
8
9  else{
10
11 _visible = true;
```

```
12
13  }
14
15  }
```

Hàm xử lý tình huống bấm chuột onClipEvent(mouseDown) của quả banh màu cam “nói” rằng: nếu biến _visible của quả banh có trị là true thì gán cho nó trị false và ngược lại. Khi biến _visible của quả banh được gán trị false, quả banh trở nên vô hình! Ấn Ctrl+Enter để chạy chương trình, bạn thấy hàm vừa viết rất “hiệu nghiệm”: với mỗi cú bấm chuột, quả banh màu cam biến đi hoặc hiện ra.



Để diễn đạt điều kiện “nếu biến _visible có trị là true”, bạn có thể viết if(_visible == true). Tuy nhiên, cách viết nêu trên gọn hơn và có ý nghĩa tương đương. Bạn còn có thể viết gọn hơn nữa:

?

```
1 onClipEvent(mouseDown) {
2
3 _visible = !_visible;
4
5 }
```

Trong đó, ta dùng tác tử “không”, biểu diễn bằng dấu chấm than (!). Khi đặt dấu chấm than trước biến _visible, bạn nhận được trị true nếu biến _visible có trị false và ngược lại. Kết quả tác động của tác tử “không” được gán trở lại biến _visible. Nói khác đi, câu lệnh _visible = !_visible; cũng có tác dụng chuyển đổi trị của biến _visible giữa true và false.

Bấm chuột nhiều lần vào lúc chạy chương trình, có lẽ bạn đã hơi hơi mỏi tay và chợt nghĩ: “Phải chi mấy quả banh này tự động diễn trò mà không cần chờ bấm chuột”. Để đạt được “ước mơ” giản dị đó, bạn chỉ cần sửa đổi mục mouseDown thành enterFrame:

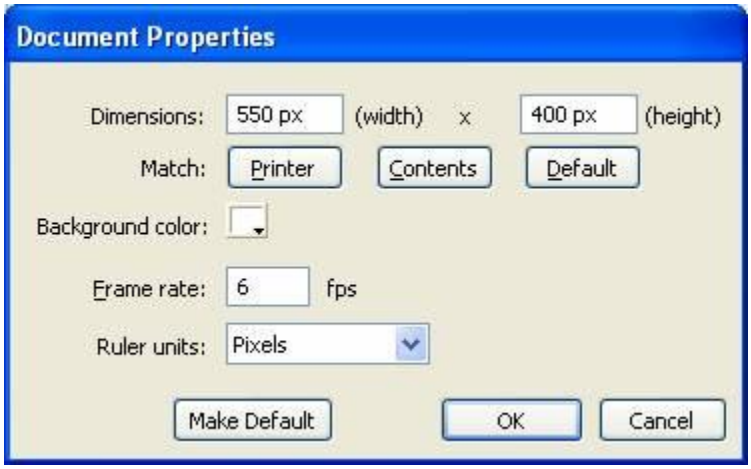
?

```
1 onClipEvent(enterFrame) {
2
3 _visible = !_visible;
4
5 }
```

Đối với hàm xử lý tình huống onClipEvent của quả banh đỏ và quả banh xanh, bạn cũng sửa như vậy. Khi đó, hàm xử lý tình huống “bấm chuột” trở thành hàm xử lý tình huống “chuyển khung”. Bạn nhớ, vào lúc chạy chương trình, Flash không ngừng hiển thị các khung liên tiếp. Mỗi lần khung mới xuất hiện, Flash gọi hàm onClipEvent(enterFrame) của các thể hiện (nếu có). Điều này nghĩa là hàm onClipEvent(enterFrame) của các thể hiện được gọi một cách tự

động và liên tục.

Chạy thử chương trình, bạn thấy khỏe re: cả ba quả banh đều hành động mà không chờ bạn “chỉ đạo”, tạo nên hoạt cảnh vui nhộn. Nếu thấy hoạt cảnh diễn biến quá nhanh, bạn dừng chương trình, giảm tốc độ “chạy khung” bằng cách bấm kép vào ô tốc độ ở cạnh dưới bảng Timeline, bấm kép vào ô Frame rate trong hộp thoại vừa hiện ra (hình 2), gõ 6 và gõ Enter (giảm tốc độ từ 12 khung mỗi giây xuống còn 6 khung mỗi giây).



Trong khi quả banh xanh và cam rất năng nổ, quả banh đỏ khiến bạn hơi phiền lòng vì nó dừng lại khi đụng “trần” và đứng yên luôn. Chắc bạn sẽ nghĩ: “Giá như quả banh đỏ liên tục chạy lên chạy xuống thì vui hơn”. Để thực hiện điều này, bạn chọn quả banh đỏ, sửa đổi trong bảng Actions như sau:

?

```
1  onClipEvent(load) {
2
3  step = 20;
4
5  }
6
7  onClipEvent(enterFrame) {
```

```
8
9  if(_y < 0 || _y > 500) {
10
11 step = -step;
12
13 trace("Tôi đổi chiều đây.");
14
15 }
16
17 _y += step;
18
19 }
```

Như bạn thấy, trong câu lệnh thay đổi tung độ _y của quả banh đỏ, thay vì ghi số cụ thể để xê dịch quả banh từng bước cố định như trước _y += 20; giờ đây ta dùng một biến, gọi là step (bạn tùy ý đặt tên cho biến): _y += step;.

Khi quả banh đụng “trần” hoặc “sàn”, chỉ cần đổi dấu cho trị của biến step, quả banh sẽ đổi chiều chuyển động. Cụ thể, câu lệnh if(_y < 0 || _y > 500) diễn đạt trường hợp đụng “trần” hoặc “sàn”: nếu _y có trị nhỏ hơn 0 hoặc _y có trị lớn hơn 500 (dấu || nghĩa là “hoặc”). Câu lệnh step = -step; dùng để đổi dấu cho trị số của biến step.

Câu lệnh step = 20; đặt bên trong hàm onClipEvent(load) dùng để gán trị ban đầu 20 cho biến step. Hàm onClipEvent(load)được gọi là hàm xử lý tình huống “khởi động” vì nó được gọi khi khởi động hoạt cảnh. Nếu không có hàm như vậy, biến step có trị ban đầu mặc định là 0. Khi đó, cả câu lệnh thay đổi tung độ _y lẫn câu lệnh đổi dấu cho trị số của biến step không có tác dụng gì hết.

Bài 8 : Thuộc tính của thể hiện – Tự học lập trình Flash

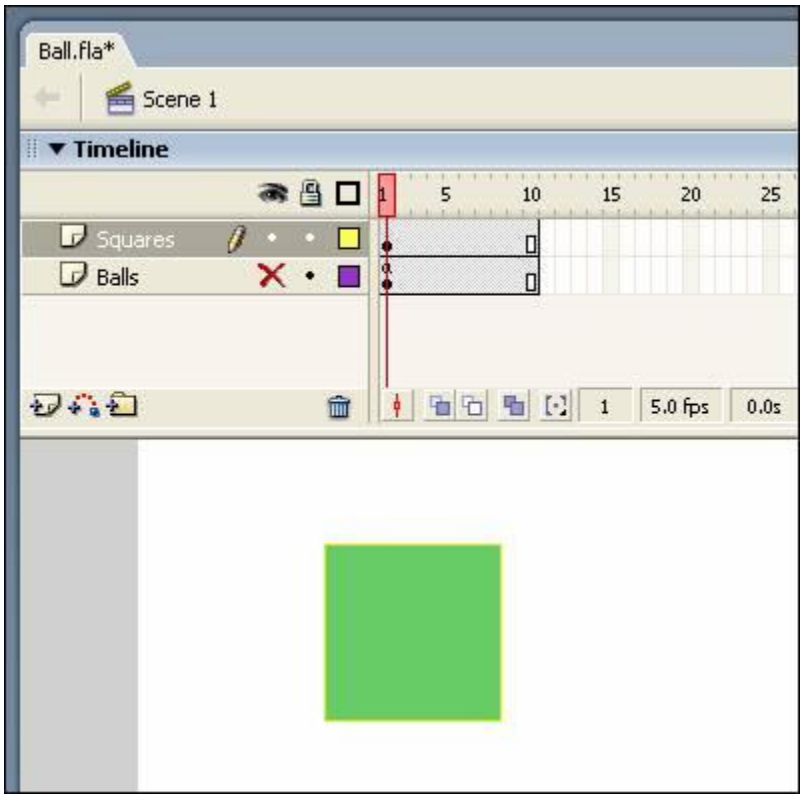
Khi chơi đùa với những quả banh (các thể hiện của nhân vật Ball), bạn đã biết đến các biến có sẵn bên trong từng quả banh, cho biết trạng thái của quả banh: _x, _y, _xscale, _yscale, _visible. Các biến như vậy gọi là các thuộc tính (property) của thể hiện. Không giống như biến do bạn tự tạo ra (như biến step cho quả banh màu đỏ), khi bạn thay đổi một thuộc tính của thể hiện nào, trạng thái của thể hiện đó tự động thay đổi. Chẳng hạn, khi gán trị false cho biến _visible của quả banh, quả banh lập tức biến mất.

Để tìm hiểu thêm các thuộc tính khác của thể hiện, bạn nên tạo nhân vật mới. Lần này ta vẽ hình vuông để "thay đổi không khí". Trước hết, bạn bấm kép vào tên lớp Layer 2 trong bảng Timeline, gõ Balls, rồi gõ Enter. Làm như vậy để đổi tên lớp thành Balls, ngụ ý rằng lớp đang xét là lớp dành cho các quả banh.

Bạn hãy tạo lớp mới dành cho các hình vuông. Bạn bấm nút "tạo lớp mới" (góc dưới, bên trái bảng Timeline), bấm kép vào tên lớp mới Layer 3 và gõ Squares. Để tập trung chú ý vào các hình vuông sắp tạo ra, bạn cho các quả banh biến mất bằng cách bấm vào dấu chấm của hàng biểu thị lớp Balls trong bảng Timeline, ở cột có hình con mắt . Các quả banh chỉ được giấu đi để đỡ vướng víu cho bạn khi làm việc, chúng vẫn "góp mặt" bình thường khi bạn chạy chương trình.

Bạn vẽ hình vuông bằng cách chọn công cụ Rectangle ở hộp công cụ, trỏ vào đâu đó trên sân khấu, giữ phím Shift và giữ phím trái của chuột, kéo chuột qua

phải, xuống dưới. Nhờ bạn giữ phím Shift, hình khung được vẽ ra là hình vuông. Bạn nên tô màu cho hình vuông khác với các quả banh, màu xanh lá chẳng hạn (hình 1).



Bạn bấm vào công cụ chọn , bấm vào hình vuông xanh lá để chọn và gõ phím F8 để chuyển hình được chọn thành nhân vật. Hộp thoại Convert to Symbol hiện ra. Bạn gõ Square để đặt tên cho nhân vật mới (hình 2). Ở phần Registration trong hộp thoại, bạn thấy điểm mốc nằm ở tâm nhân vật theo mặc định. Không cần thay đổi điểm mốc, bạn gõ Enter. Hình vuông vừa vẽ tạo nên nhân vật Square. Hình vuông trên sân khấu trở thành thể hiện của nhân vật Square.

Hình vuông xanh lá đang ở tình trạng “được chọn”. Bạn gõ phím F9 để mở bảng Actions, viết hai hàm xử lý tình huống cho hình vuông được chọn như sau:

?

```
1  onClipEvent(load) {
2
3  step = 10;
4
5  }
6
7  onClipEvent(enterFrame) {
8
9  _rotation += step;
10
11 }
```

Trong đó, hàm xử lý tình huống khởi động tạo ra biến step chứa trị số 10, hàm xử lý tình huống chuyển khung cộng biến step vào thuộc tính _rotation của hình vuông. Nhờ vậy, mỗi lần chuyển khung, góc quay của hình vuông lại tăng thêm 10. Bạn chú ý, biến step ta dùng ở đây không phải là biến step đã được dùng cho quả banh màu đỏ. Hai biến step tuy cùng tên nhưng có phạm vi (scope) khác nhau, không có “dây mơ rễ má” gì hết.

Ấn Ctrl+Enter để chạy chương trình, bạn thấy hình vuông quay đều do góc quay của nó “tà tà” tăng lên mỗi lần chuyển khung. “Nếu cứ để hình vuông quay hoài hoài, lẽ nào góc quay sẽ tăng đến… vô cùng?”. Bạn yên tâm, thực ra góc quay của hình vuông chỉ có thể nhận các trị số từ -180 đến 180 (tính bằng độ). Flash tự động điều chỉnh trị được gán cho biến _rotation để có trị

thích hợp trong khoảng đó.

Để kiểm tra, bạn có thể ghi câu lệnh theo dõi trace(“Góc quay: ” + _rotation); sau câu lệnh _rotation += step; trong hàm OnClipEvent(enterFrame). Bằng cách dùng hàm trace như vậy, khi chạy chương trình, bạn sẽ thấy dòng thông báo đại loại như thế này: Góc quay: 120. Đó là nhờ chuỗi “Góc quay: ” được ghép với trị số của biến _rotation bằng dấu cộng.

Nếu không muốn có quá nhiều thông báo ở bảng Output khi chạy chương trình, bạn thêm dấu // trước câu lệnh gọi hàm trace:

//trace(“Góc quay: ” + _rotation);

Flash hiểu rằng những gì được ghi sau dấu // trên một dòng là phần chú thích, không cần xét đến khi biên dịch. Do vậy, đặt dấu // trước một câu lệnh giúp bạn tạm thời vô hiệu hóa câu lệnh đó. Khi muốn khôi phục hiệu lực của câu lệnh, bạn chỉ cần xóa dấu // trước câu lệnh.

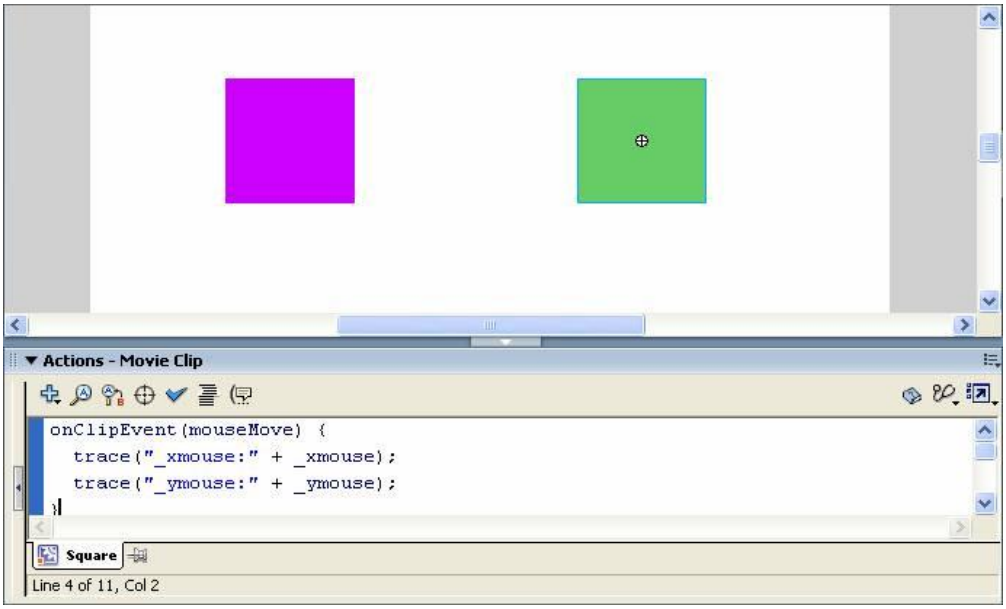
Bạn hãy nhuộm màu tím cho hình vuông hiện có, gõ phím F11 để mở bảng Library và kéo hình vuông xanh lá từ thư viện vào sân khấu, tạo thêm một thể hiện nữa của nhân vật Square (hình 3). Trong bảng Actions, bạn viết hàm xử lý tình huống như sau cho hình vuông xanh lá:

?

```
1  onClipEvent(mouseMove) {
2
3  trace("_xmouse: " + _xmouse);
4
5  trace("_ymouse: " + _ymouse);
6
7  trace("_alpha: " + _alpha);
8
```

```
9  _alpha -= 1;
10
11 }
```

Bạn chạy chương trình để thử đoán nhận ý nghĩa của những thuộc tính mà bạn chưa biết. Bạn sẽ có “đáp án” vào kỳ sau.

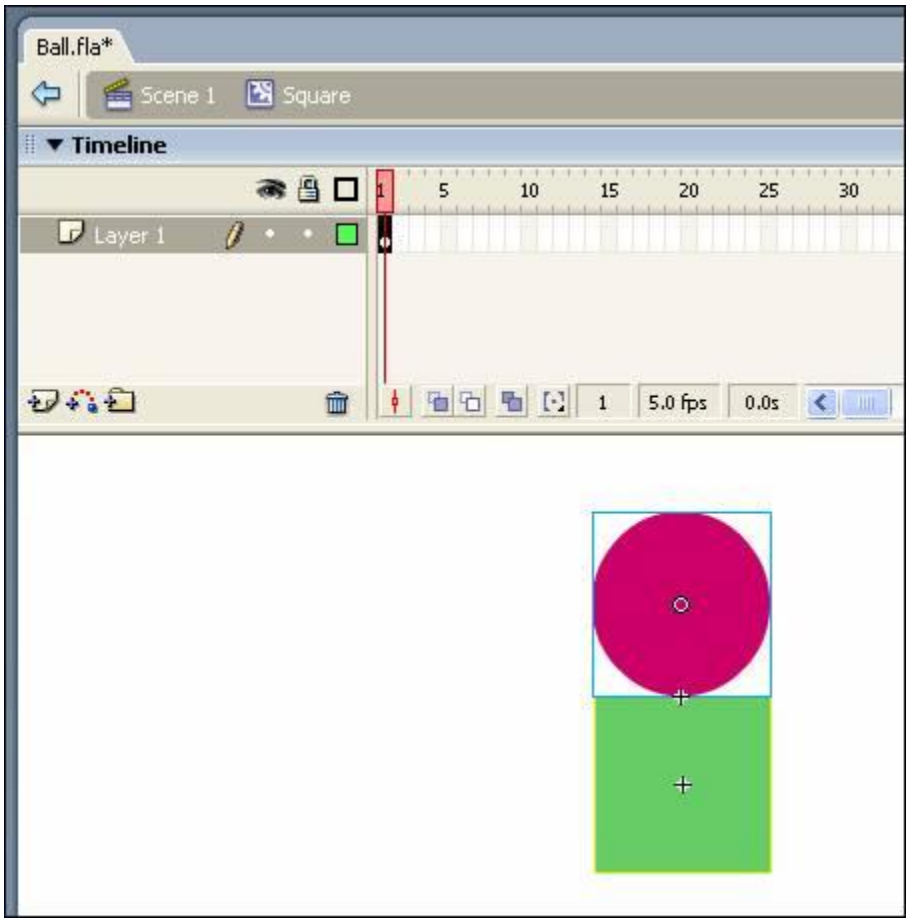


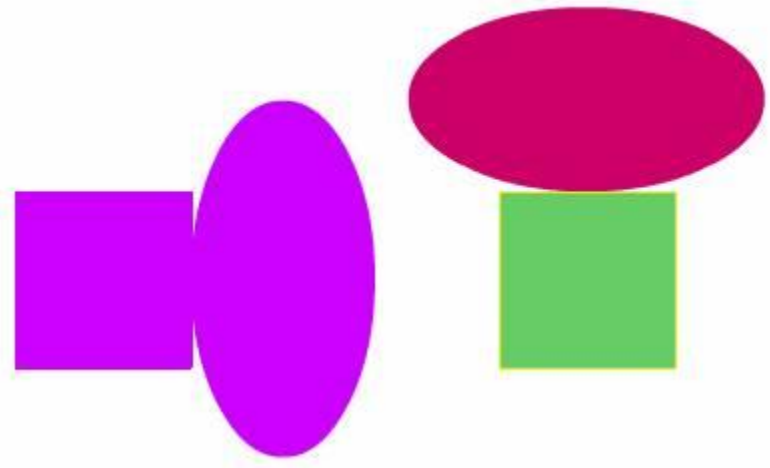
Bài 9 : Nhân vật phức hợp – Tự học lập trình Flash

Bạn hãy chỉnh sửa nhân vật Square theo cách đơn giản: chọn nhân vật Ball trong danh sách, kéo quả banh trong khung phía trên danh sách, đặt ngay trên hình vuông trong khung lớn ở giữa (hình 1). Làm như vậy, nhân vật Square có thêm một bộ phận mới là nhân vật Ball. Nhân vật Square trở thành nhân vật phức hợp, chứa đựng bên trong nó một nhân vật khác.

Bấm vào mục Scene 1 phía trên bảng Timeline để thoát khỏi chế độ chỉnh sửa nhân vật Square, trở về với sân khấu, bạn thấy hai thể hiện của nhân vật Square giờ đây đều bao gồm một hình vuông và một hình tròn. Lúc chạy chương trình, hai thể hiện của nhân vật Square vẫn có hành vi như trước, chỉ

khác ở chỗ có thêm quả banh liên tục “phập phồng” (hình 2). Nhân vật Ball trở thành một bộ phận của nhân vật Square nhưng không hề quên đi “bản năng” của nó.





Bài 10 : Hàm có sẵn trong Flash – Tự học lập trình Flash

Có lẽ bạn đang nóng lòng muốn lập trình trò chơi gì đó thú vị với Flash. Tuy vậy, bạn cần kiên trì tìm hiểu những khái niệm lập trình cơ bản. Khi có nền tảng vững vàng, bạn mới có thể tự do sáng tạo. Thực ra vẫn có nhiều điều thú vị trong kiến thức cơ bản.

Ngoài hàm trace() mà bạn từng dùng vài lần để theo dõi diễn biến của chương trình, còn khá nhiều hàm có sẵn (built-in function) khác trong Flash, làm đủ thứ việc. Gọi là "hàm có sẵn" để phân biệt với hàm do bạn tự tạo ra, tự đặt tên (user-defined function). Bạn cần biết cách dùng hàm có sẵn trước khi thử tạo ra hàm có chức năng như ý.

Bạn hãy khởi động Flash và chọn Flash Document để tạo tập tin mới. Bạn gõ phím F9 để mở bảng Actions. Nói cho rõ, dòng tiêu đề của bảng là Actions – Frame, ngụ ý rằng những câu lệnh bạn viết trong bảng có hiệu lực đối với khung hiện hành, sẽ được thi hành khi Flash hiển thị khung đó vào lúc chạy. Khác với trường hợp lập trình cho nhân vật hoặc thể hiện, ở đây bạn không viết hàm xử lý tình huống. Bạn gõ hai câu lệnh như sau:

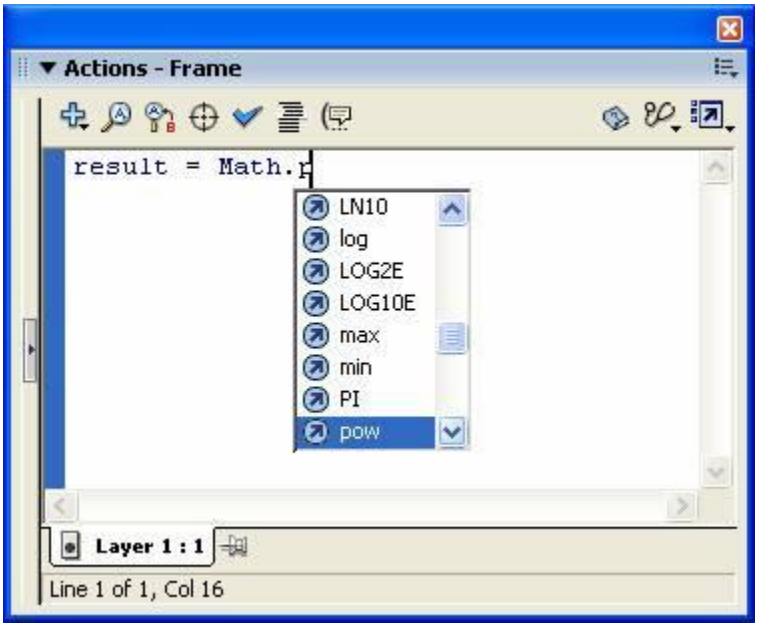
?

```
1 result = Math.pow(2, 3);
2
3 trace("2 lũy thừa 3: " + result);
```

Câu lệnh thứ nhất tính "2 lũy thừa 3″ và gán kết quả cho biến mang tên result (để tạo ra một biến, bạn thoải mái viết tên biến tùy ý và gán trị cho nó). Câu lệnh thứ hai trình bày trị của biến result ở bảng Output, giúp bạn biết kết quả tính có đúng hay không.

Lúc viết câu lệnh thứ nhất, sau khi bạn gõ Math và dấu chấm, Flash nhanh nhẩu đưa ra danh sách các hàm khả dĩ (hình 1). Bạn gõ thêm chữ p, Flash đoán rằng đó là hàm lũy thừa pow. Thay vì gõ tiếp ow, bạn có thể gõ Enter để chấp thuận cho Flash ghi hàm pow thay mình. Tiếp theo, Flash nhiệt tình đưa ra dòng hướng dẫn nhắc bạn rằng đối mục thứ nhất của hàm pow là cơ số và đối mục thứ hai là số mũ.

Viết xong hai câu lệnh, bạn ấn Ctrl+Enter để chạy chương trình. Khi hiển thị khung số 1 (ngoài khung số 1, bạn chưa có khung nào khác), Flash thi hành hai câu lệnh được ghi ở khung đó. Kết quả xuất hiện ở bảng Output cho thấy Flash “tính toán như thần”: 2 lũy thừa 3: 8.



Có lẽ bạn thấy hơi lạ: “Tại sao phải viết tên hàm là Math.pow, thay vì chỉ viết pow cho gọn?”. Viết Math.pow nghĩa là gọi hàm pow của lớp Math (“lớp toán học”). Những hàm có sẵn của Flash được đặt trong nhiều lớp có tên khác nhau để phân loại. Các hàm trong một lớp phục vụ cho một lĩnh vực nhất định. Bạn cần phân biệt khái niệm lớp (class) khá trừu tượng ở đây với lớp (layer) ở bảng thời tuyến. Trong công việc thực tế, bạn dùng “tầng tầng lớp lớp” ở bảng thời tuyến là để tách biệt những hình vẽ và nhân vật trên sân khấu cho khỏi

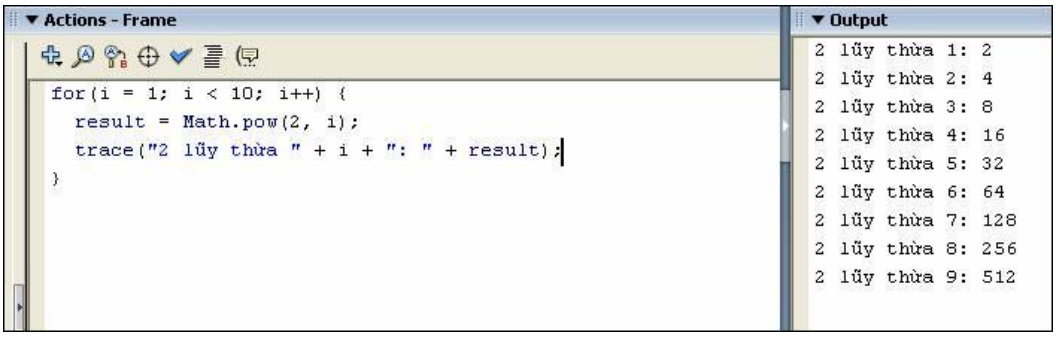
rối.

Bạn có thể yêu cầu Flash liên tiếp tính lũy thừa với cơ số là 2 và số mũ là 1, 2,…, 9 bằng cách viết lại đoạn mã ActionScript cho khung 1 như sau:

?

```
1 for(i = 1; i < 10; i++) {
2
3 result = Math.pow(2, i);
4
5 trace("2 lũy thừa " + i + ": " + result);
6
7 }
```

Chạy chương trình, bạn thu được ngay kết quả trong bảng Output (hình 2). Bạn thấy rõ câu lệnh result = Math.pow(2, i); được thi hành lặp đi lặp lại 9 lần. Người ta gọi đó là một vòng lặp (loop). Trong lần thi hành đầu tiên, biến i được gán trị 1. Mỗi lần lặp lại, trị của biến i tăng thêm 1 so với trước.



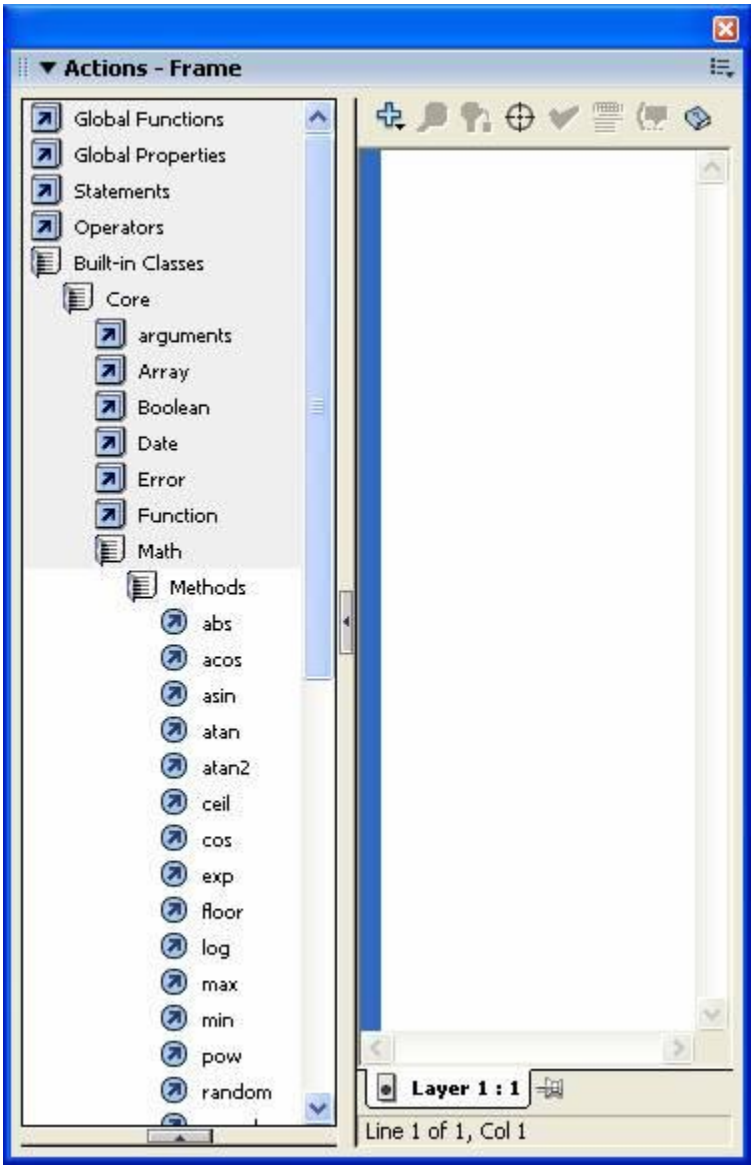
Hai câu lệnh tính lũy thừa và xuất kết quả được thi hành lặp đi lặp lại là nhờ được ghi bên trong cặp dấu gộp { } sau dòng lệnh for(i = 1; i < 10; i++). Trong cặp dấu ngoặc ( ) sau từ for, ta diễn đạt việc cần làm khi bắt đầu vòng lặp (gán trị 1 cho biến i) và việc cần làm sau mỗi lần lặp (tăng trị của i thêm 1 và xem i có còn bé hơn 10 hay không). Khi viết i < 10, bạn đã đưa ra điều kiện duy trì vòng lặp. Nếu điều kiện đó không thỏa, vòng lặp kết thúc.

Với biến i chứa trị là số nguyên, để tăng trị của i thêm 1, bạn viết i++. “Sao không dùng một dấu cộng thôi nhỉ?”. Vâng, một dấu cộng thì “tự nhiên” hơn nhưng người ta phải dùng tác tử “cộng cộng” ++ để phân biệt với tác tử “cộng” + thực hiện phép cộng thông thường.
Nói cho đúng, tác tử “cộng” + trong ngôn ngữ ActionScript không hẳn là “phép cộng thông thường”. Khi viết đối mục của hàm trace(), ta đã dùng tác tử “cộng” để ghép chuỗi với trị nguyên. Chắc bạn thấy rằng viết như vậy cũng… “tự nhiên” thôi.

Bài 11 : Các hàm tự tạo trong Flash – Tự học lập trình Flash

Sau khi dùng thử hàm Math.pow (hàm pow của lớp Math), có lẽ bạn muốn biết những hàm có sẵn khác. Để tra cứu các hàm có sẵn, bạn trỏ vào cạnh trái khung soạn thảo trong bảng Actions (trong cửa sổ Flash, nếu bảng Actions chưa mở, bạn gõ phím F9), sao cho con trỏ chuột chuyển thành dạng “mũi tên hai đầu”, rồi kéo chuột qua phải. Bạn thấy lộ ra một cửa sổ “bí mật”, chứa đựng các “tàng thư” quan trọng đối với việc lập trình Flash (hình 1).
Thư mục đầu tiên Global Functions giúp bạn tìm hiểu những hàm toàn cục, tức những hàm cần dùng thường xuyên, không nằm trong lớp nào cả. Thư mục tiếp theo Global Properties liệt kê các biến toàn cục, tức những biến mà bạn có thể dùng bất cứ lúc nào khi viết chương trình. Biến toàn cục được dùng chung cho mọi nhân vật và thể hiện.

Thư mục Built-in Classes giúp bạn tìm hiểu các lớp có sẵn. Bấm vào thư mục đó, bạn thấy các thư mục con, trong đó thư mục Core liệt kê các lớp cốt lõi. Thử bấm vào thư mục Core, bấm vào thư mục Math, bấm vào thư mục Methods, bạn thấy rõ các hàm thuộc lớp Math. Trỏ vào từng hàm, bạn thấy hiện ra dòng giải thích ngắn gọn về công dụng của hàm.



Sau này, khi đã lập trình “quen tay”, bạn sẽ thường xuyên mở cửa sổ tra cứu để xem lại cách dùng các hàm có sẵn. Hiện thời, bạn chỉ cần biết sơ lược như vậy về chỗ tra cứu. Bạn hãy bấm nút có dấu tam giác ở cạnh phải cửa sổ tra cứu để dẹp nó đi.

Ngoài việc dùng các hàm có sẵn, bạn sẽ phải tạo ra hàm phù hợp với nhu cầu

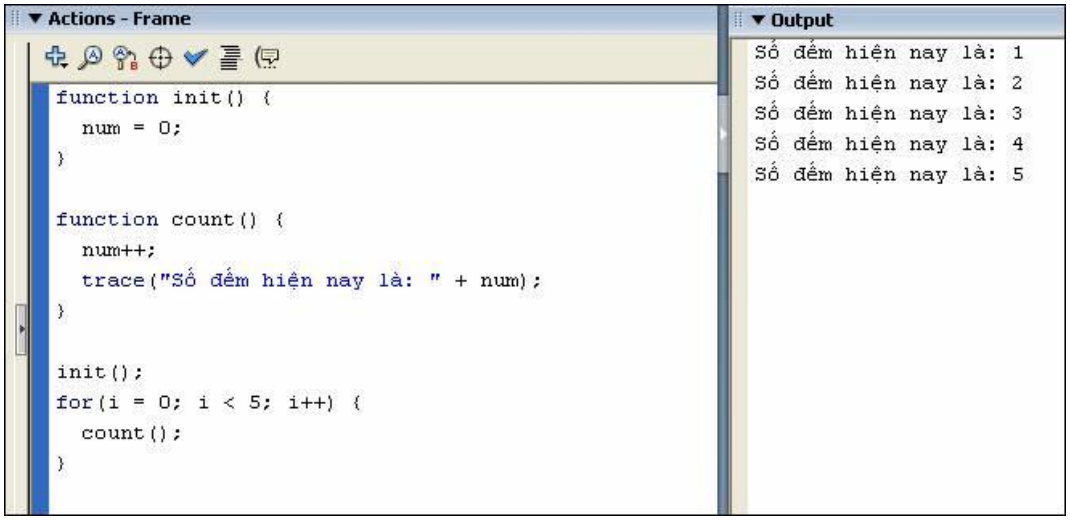
của mình. Người ta gọi đó là hàm tự tạo (user-defined function) nhằm phân biệt với hàm có sẵn.

Để thử tạo ra hàm, bạn xóa sạch nội dung hiện có trong bảng Actions (nếu còn) và gõ đoạn mã như sau:

?

```
1  function init() {
2
3  num = 0;
4
5  }
6
7  function count() {
8
9  num++;
10
11 trace("Số đếm hiện nay là: " + num);
12
13 }
```

Viết như vậy nghĩa là bạn định nghĩa hai hàm init() và count(). Định nghĩa hàm bắt đầu bằng từ function, nhằm làm cho Flash hiểu rằng “Đây là hàm mới đấy nhé!”. Sau từ function là tên hàm do bạn tùy ý chọn và cặp dấu ngoặc (). Tiếp theo, phần được ghi giữa cặp dấu gộp {} là thân hàm (function body), diễn đạt những việc mà hàm thực hiện.



Hàm init() gán cho biến num trị ban đầu là 0. Hàm count() làm cho trị của biến num tăng thêm 1 và thể hiện trị của biến num trong bảng Output. Bạn có thể hình dung hàm count() dùng để đếm thứ gì đó. Bạn gọi hàm count() để "đếm tới". Khi nào cần "đếm lại từ đầu", bạn gọi hàm init() trước, sau đó gọi hàm count().

Để thử dùng hai hàm mới toanh của mình, bạn ghi thêm như sau bên dưới hai định nghĩa hàm:

?

```
1 init();
2
3 for(i = 0; i < 5; i++) {
4
5 count();
6
7 }
```

Viết như vậy nghĩa là gọi hàm init(), rồi gọi hàm count() năm lần. Thay vì viết năm lần câu lênh count(); ta dùng một vòng lặp for theo cách thức mà bạn đã biết. Ấn Ctrl+Enter để chạy chương trình, bạn thấy dòng thông báo về số đếm

hiện ra năm lần đúng như dự định.

Định nghĩa hàm còn có thể được viết theo cách khác như sau:

?

```
1  init = function() {
2
3  num = 0;
4
5  }
6
7  count = function() {
8
9  num++;
10
11 trace("Số đếm hiện nay là: " + num);
12
13 }
14 Thay vì viết vòng lặp for, bạn có thể dùng vòng
15 lặp while:
16
17 while(num < 5) {
18
19 count();
20
   }
```

Sửa lại chương trình theo cách viết mới và chạy thử, bạn thấy kết quả trong bảng Output hoàn toàn giống như trước. Quan sát cách viết vòng lặp while,

bạn hiểu ngay: vòng lặp này chỉ được duy trì khi điều kiện num < 5 được thoả. Sau khi biến num nhận trị số 5, điều kiện num < 5 không được thỏa nữa, vòng lặp kết thúc ngay.

Khi lập trình, nếu bạn thấy mình đang viết đi viết lại một nhóm câu lệnh nào đó, cần nghĩ ngay đến việc viết một hàm chứa đựng nhóm câu lệnh đó để dùng cho tiện, không nên để chương trình dài lê thê với nhiều đoạn tương tự. Sau khi định nghĩa hàm, chỉ cần một câu lệnh gọi hàm đó, bạn có thể giải quyết nhanh gọn nhiều việc.

Bài 12 : Trò chơi "rượt bắt" – Tự học lập trình Flash

Với những điều đã biết về Flash, bạn có thể bắt tay vào việc thực hiện trò chơi "rượt bắt" đơn giản: một con vật gì đó rượt theo bạn, nói cho đúng là rượt theo con trỏ chuột do bạn điều khiển. Có thể hình dung con trỏ chuột đang di chuyển… dưới nước và một con cá "hung dữ" đang đuổi theo.

Trước hết, bạn mở cửa sổ Flash, bắt đầu với tập tin mới trống trơn. Bạn cần vẽ một hình tròn tượng trưng cho con cá (tưởng tượng đó là… cá nóc). Khi chương trình đã chạy tốt, ta sẽ vẽ con cá một cách tỉ mỉ. Để vẽ hình tròn, bạn đã biết rằng cần dùng công cụ Oval Tool và giữ phím Shift khi vẽ trên sân khấu. Xong, bạn bấm vào công cụ chọn Selection Tool , căng khung chọn bao quanh hình tròn vừa vẽ, chuẩn bị chọn màu tô (fill color) và màu nét (stroke color).

Ở hộp công cụ phía trái cửa sổ Flash, bạn chú ý phần Colors có hai ô màu. Bạn bấm vào ô màu trên, chọn màu nét trong bảng màu vừa hiện ra. Tương tự, bạn bấm vào ô màu dưới, chọn màu tô. Bạn nên chọn màu nét đen và

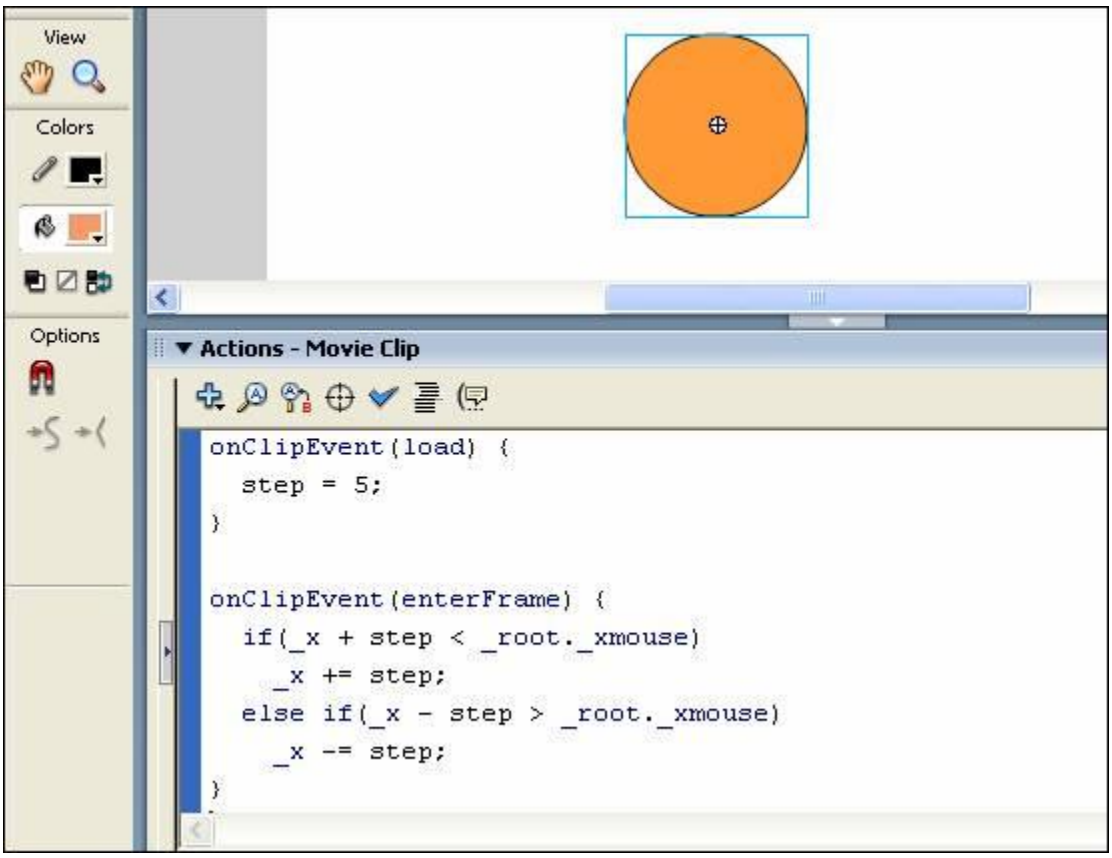
màu tô sáng (màu cam chẳng hạn) cho dễ thấy “con cá” của mình.
Để hình tròn trở thành nhân vật, bạn gõ phím F8. Khi thấy hộp thoại Convert to Symbol, bạn gõ Fish để đặt tên cho nhân vật và gõ Enter. Từ đây, hình tròn đang hiện diện trên sân khấu là một thể hiện của nhân vật Fish.
Để lập trình cho con cá, bạn gõ phím F9 để mở cửa sổ Actions – Movie Clip (hình 1). Vì thể hiện của nhân vật Fish đang ở tình trạng “được chọn”, những gì bạn sắp ghi vào cửa sổ Actions – Movie Clip sẽ quy định hành vi của thể hiện. Bạn đã biết rằng khi muốn diễn đạt hành vi của thể hiện, ta phải viết các hàm xử lý tình huống. Bạn viết hai hàm xử lý tình huống như sau để làm cho con cá biết đuổi theo con trỏ chuột:

?

```
1  onClipEvent(load) {
2
3  step = 5;
4
5  }
6
7  onClipEvent(enterFrame) {
8
9  if(_x + step < _root._xmouse)
10
11 _x += step;
12
13 else if(_x - step > _root._xmouse)
14
15 _x -= step;
```

16

17 }



Trong hàm onClipEvent(load), ta tạo ra biến step và gán trị cụ thể cho nó. Biến step xác định bước dịch chuyển của cá. Trị của biến step càng lớn, cá dịch chuyển càng nhanh.

Hàm onClipEvent(enterFrame) được gọi mỗi khi Flash hiển thị một khung. Hiện thời, bạn chỉ có một khung nhưng vì Flash hiển thị khung đó lặp đi lặp lại vào lúc chạy, hàm onClipEvent(enterFrame) vẫn sẽ được gọi liên tục.

Trong hàm onClipEvent(enterFrame), để biết cá có nên đi tới hay không (có nên gia tăng hoành độ _x của cá hay không), ta lấy hoành độ _x của cá cộng với step và so sánh kết quả với hoành độ của con trỏ chuột (tức _root._xmouse). Nếu kết quả của phép cộng ấy vẫn còn nhỏ hơn hoành độ của con trỏ chuột (điều kiện _x + step < _root._xmouse được thỏa), ta cho hoành độ _x của cá tăng thêm một bước: x += step; . Nếu kết quả lớn hơn

hoành độ của con trỏ chuột, rõ ràng không nên đi tới và cần tiếp tục xét xem có nên đi lui hay không.

Để biết có nên cho cá đi lui hay không, ta lấy hoành độ _x trừ đi step và so sánh kết quả của phép trừ đó với hoành độ của con trỏ chuột. Nếu kết quả vẫn còn lớn hơn hoành độ của con trỏ chuột, nên cho cá đi lui bằng cách giảm hoành độ _x một bước: x -= step;. Ngoài các trường hợp đã xét, hoành độ _x sẽ được giữ nguyên, tức là cá đứng yên.

Có lẽ bạn hơi ngờ ngợ về cách viết hàm onClipEvent(enterFrame). Dường như phải dùng các cặp dấu gộp {} như sau mới đúng cú pháp của câu lệnh điều kiện:

?

```
1  onClipEvent(enterFrame) {
2
3  if(_x + step < _root._xmouse) {
4
5  _x += step;
6
7  }
8
9  else if(_x - step > _root._xmouse) {
10
11 _x -= step;
12
13 }
14
15 }
```

Thực ra, nếu phần diễn đạt việc cần làm khi điều kiện được thỏa chỉ có một câu lệnh, ngôn ngữ ActionScript cho phép ta viết câu lệnh ấy theo sau điều kiện, không cần đặt trong cặp dấu gộp.

Thử chạy chương trình, bạn sẽ thấy không có lỗi biên dịch. Bạn đưa con trỏ chuột qua trái (hoặc phải), cá cũng bơi qua trái (hoặc phải). Cá chưa biết bơi lên, bơi xuống để "bắt dính" con trỏ chuột vì ta chưa "huấn luyện" cho cá làm điều đó. Để cá biết rằng nên bơi lên hoặc xuống khi nào, trong hàm onClipEvent(enterFrame), bạn cần xét đến tung độ _y của cá và tung độ của con trỏ chuột _root._ymouse theo cách thức tương tự như đã làm đối với hoành độ. Trở lại với chương trình, bạn viết thêm vào hàm onClipEvent(enterFrame) như sau:

?

```
1  onClipEvent(enterFrame) {
2
3  if(_x + step < _root._xmouse)
4
5  _x += step;
6
7  else if(_x - step > _root._xmouse)
8
9  _x -= step;
10
11 if(_y + step < _root._ymouse)
12
13 _y += step;
14
```

```
15 else if(_y - step > _root._ymouse)
16
17 _y -= step;
18
19 }
```

Thử chạy chương trình, bạn thấy rõ cá trở nên “khôn lanh” hơn hẳn.

Bài 13 : Hàm kiểm tra va chạm – Tự học lập trình Flash

Trong trò chơi “rượt bắt” đơn giản đã thực hiện, bạn huấn luyện con cá của mình cách thức rượt đuổi mục tiêu di động là con trỏ chuột. Với hàm onClipEvent(enterFrame) đã viết, con cá mới chỉ “rượt”, chứ chưa “bắt”. Con cá cần nhận biết thời điểm nó chạm vào được mục tiêu để còn “la toáng” lên.

Thực ra ta chỉ cần bổ sung một chút nữa thôi. Trường hợp cá bắt kịp con trỏ chuột có thể xem là trường hợp cá đứng yên, không phải di chuyển theo phương ngang cũng như theo phương dọc.

Bạn hãy bấm chọn con cá trên sân khấu, mở bảng Actions (gõ phím F9) và viết thêm vào hàm onClipEvent(enterFrame) để có nội dung như sau:

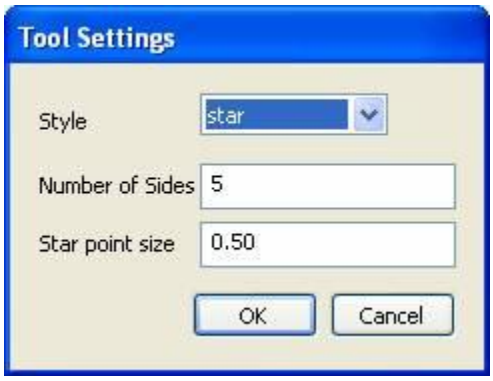
```
onClipEvent(enterFrame) {
caught = false;
if(_x + step < _root._xmouse)
_x += step;
else if(_x – step > _root._xmouse)
_x -= step;
```

```
else caught = true;
if(_y + step < _root._ymouse)
_y += step;
else if(_y – step > _root._ymouse)
_y -= step;
else if(caught == true)
trace(“Bắt được rồi nhé!”);
}
```

Trong hàm onClipEvent(enterFrame), trước hết ta tạo biến caught có trị ban đầu là false. Khi biết chắc cá không cần di chuyển theo phương ngang, ta gán trị true cho biến caught để “làm hiệu”. Tiếp theo, nếu biết chắc cá không cần di chuyển theo phương dọc, ta xét xem biến caught có trị là true hay không. Khi đó, nếu caught có trị là true, nghĩa là xảy ra tình trạng cá không cần di chuyển theo phương ngang cũng như theo phương dọc, câu lệnh trace(“Bắt được rồi nhé!”); thông báo hả hê rằng con trỏ chuột đã bị bắt.

Chạy thử chương trình và… giả vờ chậm chạp để cá bắt kịp con trỏ chuột, bạn thấy rõ những chi tiết bổ sung trong hàm onClipEvent(enterFrame) có hiệu lực ra sao. Nhưng một khi người chơi phải giả vờ thua, cuộc chơi quá dễ, mất ý nghĩa. Ta cần làm cho người chơi bận rộn hơn, căng thẳng hơn, bằng cách bày thêm luật chơi như sau: trong khi bị cá đuổi, người chơi phải tóm một con sao biển. Mỗi lần bấm trúng sao biển, người chơi được thêm một điểm, sao biển xuất hiện ngẫu nhiên ở vị trí khác và cá di chuyển nhanh hơn. Để có sao biển, bạn tạm vẽ hình ngôi sao đơn giản. Bạn trỏ vào công cụ vẽ hình khung Rectangle Tool , giữ phím trái của chuột chút xíu, chọn công cụ PolyStar Tool (công cụ vẽ đa giác hoặc hình sao). Ấn Ctrl+F3, bạn thấy bảng Properties mở ra, trình bày những quy định liên quan đến hoạt động của công

cụ mà bạn đang “cầm trong tay”. Bạn bấm nút Options để mở hộp thoại Tool Settings (hình 1). Trong đó, bạn chọn mục star ở ô Style và bấm OK. Trong bảng Properties, bạn có thể chọn trước màu nét và màu tô cho hình sao sắp vẽ.



Trỏ vào đâu đó trên sân khấu, bạn giữ phím trái của chuột, kéo chuột qua phải, xuống dưới, để căng ra một hình ngôi sao năm cánh.

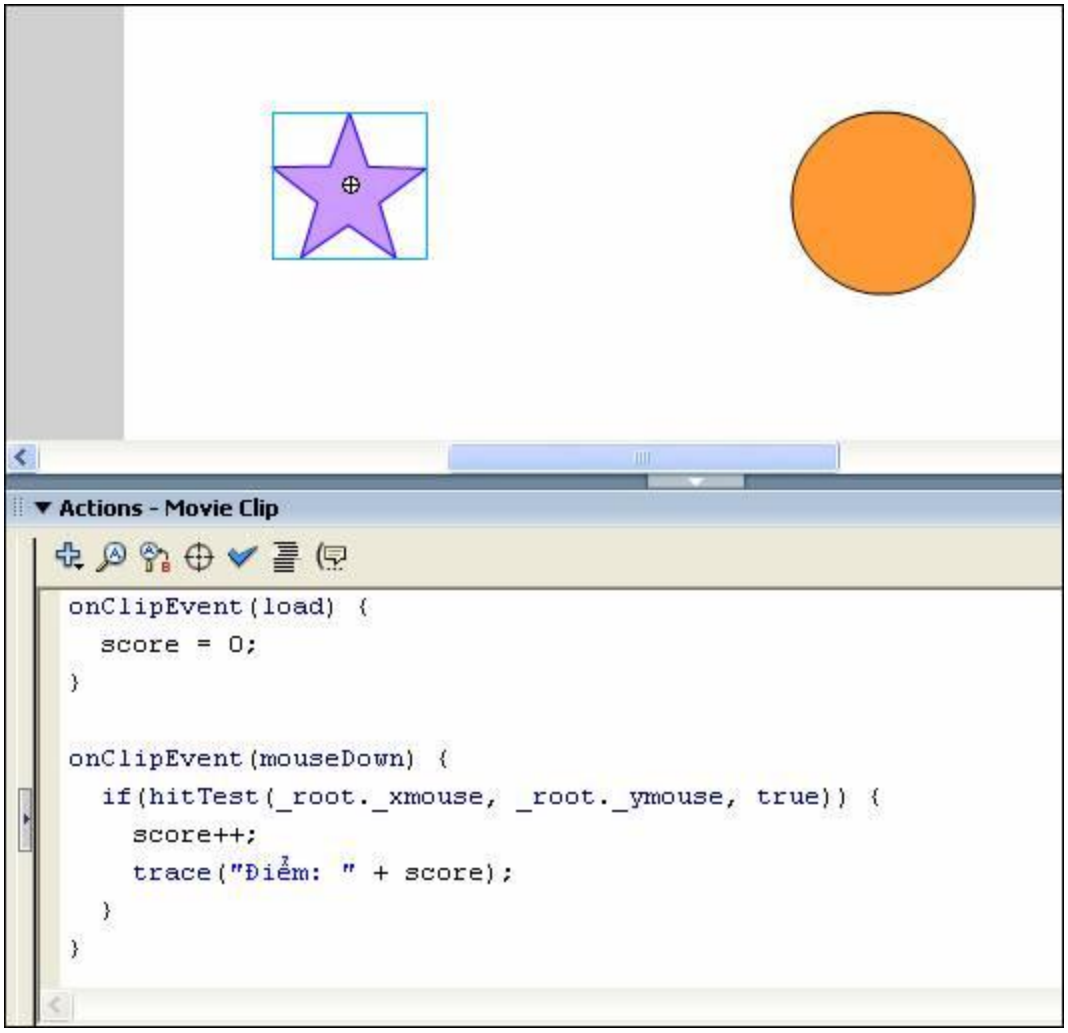
Gõ phím V để chuyển qua công cụ chọn Selection Tool, ban căng khung chọn bao quanh hình sao. Bạn gõ phím F8 để mở hộp thoại Convert to Symbol, gõ tên Star và gõ Enter. Thao tác như vậy tạo ra nhân vật “sao biển” mang tên Star. Hình sao trên sân khấu trở thành một thể hiện của nhân vật Star.

Ta cần lập trình để sao biển nhận biết con trỏ chuột có chạm vào nó hay không mỗi khi người chơi bấm chuột. Mỗi khi bị bấm trúng, sao biển có nhiệm vụ cộng điểm cho người chơi. Gõ phím F9 để mở bảng Actions tương ứng với thể hiện được chọn (hình 2), bạn gõ đoạn mã như sau:

?

```
1  onClipEvent(load) {
2
3  score = 0;
4
5  }
```

```
6
7  onClipEvent(mouseDown) {
8
9  if(hitTest(_root._xmouse, _root._ymouse, true))
10 {
11
12 score++;
13
14 trace("Điểm: " + score);
15
16 }
17
   }
```



Hàm onClipEvent(load) của sao biển tạo ra biến score để ghi điểm cho người chơi, có trị ban đầu là 0. Hàm onClipEvent(mouseDown), chắc bạn vẫn nhớ, diễn đạt những việc cần làm khi người chơi bấm chuột. Để biết vị trí của con trỏ chuột có nằm trong hình sao hay không, ta dùng hàm hitTest() có sẵn trong mọi nhân vật Flash. Hàm hitTest() cho kết quả là trị true hoặc trị false tùy theo con trỏ chuột có chạm vào thể hiện đang xét hay không.

Hai đối mục đầu tiên của hàm hitTest() là hoành độ và tung độ của con trỏ chuột (_root._xmouse và _root._ymouse). Nếu đối mục thứ ba là false, hàm hitTest() sẽ kiểm tra xem con trỏ chuột có nằm trong khung bao chữ nhật (bounding box) của thể hiện hay không. Vì ta ghi đối mục thứ ba của hàm hitTest() là true, hàm hitTest() kiểm tra kỹ càng hơn, chỉ trả về cho ta trị true nếu con trỏ chuột thực sự chạm vào hình sao.

Quan sát câu lệnh điều kiện trong hàm onClipEvent(mouseDown), bạn hiểu ngay: nếu con trỏ chuột chạm vào sao biển, trị của biến score được tăng thêm một và thông báo xuất hiện ở bảng Output cho người chơi biết họ đã đạt được bao nhiêu điểm.

Thử chạy chương trình, bạn thấy tuy sao biển chưa di chuyển được như dự định nhưng trò chơi đã trở nên thú vị hơn.

Bài 14 : Hàm tính trị ngẫu nhiên – Tự học lập trình Flash

Trong trò chơi đang thực hiện, ta dự định cho sao biển di chuyển ngẫu nhiên mỗi khi được bấm trúng (xem như người chơi nhặt được sao biển và một sao biển khác xuất hiện tại vị trí bất kỳ). Muốn vậy, trong hàm onClipEvent(mouseDown) của sao biển, bạn gán trị ngẫu nhiên cho hoành độ _x và tung độ _y của sao biển. Nhưng trước tiên bạn cần làm quen với hàm tính trị ngẫu nhiên random() trong lớp Math.

Bạn bấm chọn sao biển, mở bảng Actions – MovieClip để xem lại chương trình của sao biển và ghi thêm câu lệnh hiển thị trị ngẫu nhiên do hàm Math.random() cung cấp:

?

```
onClipEvent(mouseDown) {
1
2
if(hitTest(_root._xmouse, _root._ymouse, true))
3
{
4
5
trace("Trị ngẫu nhiên: " + Math.random());
6
```

```
7  score++;
8
9  trace("Điểm: " + score);
10
11 }
12
13 }
```

Thử chạy chương trình, bạn thấy mỗi khi bấm trúng sao biển, sao biển thông báo một trị ngẫu nhiên ở bảng Output. Bấm sao biển nhiều lần, bạn sẽ nhận ra trị ngẫu nhiên trả về bởi hàm Math.random() luôn nhỏ hơn 1. Nói chính xác, hàm Math.random() cho ta trị ngẫu nhiên lớn hơn hoặc bằng 0 và nhỏ hơn 1. Muốn thu được trị ngẫu nhiên trong khoảng tùy ý, ta phải "phóng lớn" trị trả về của hàm Math.random().

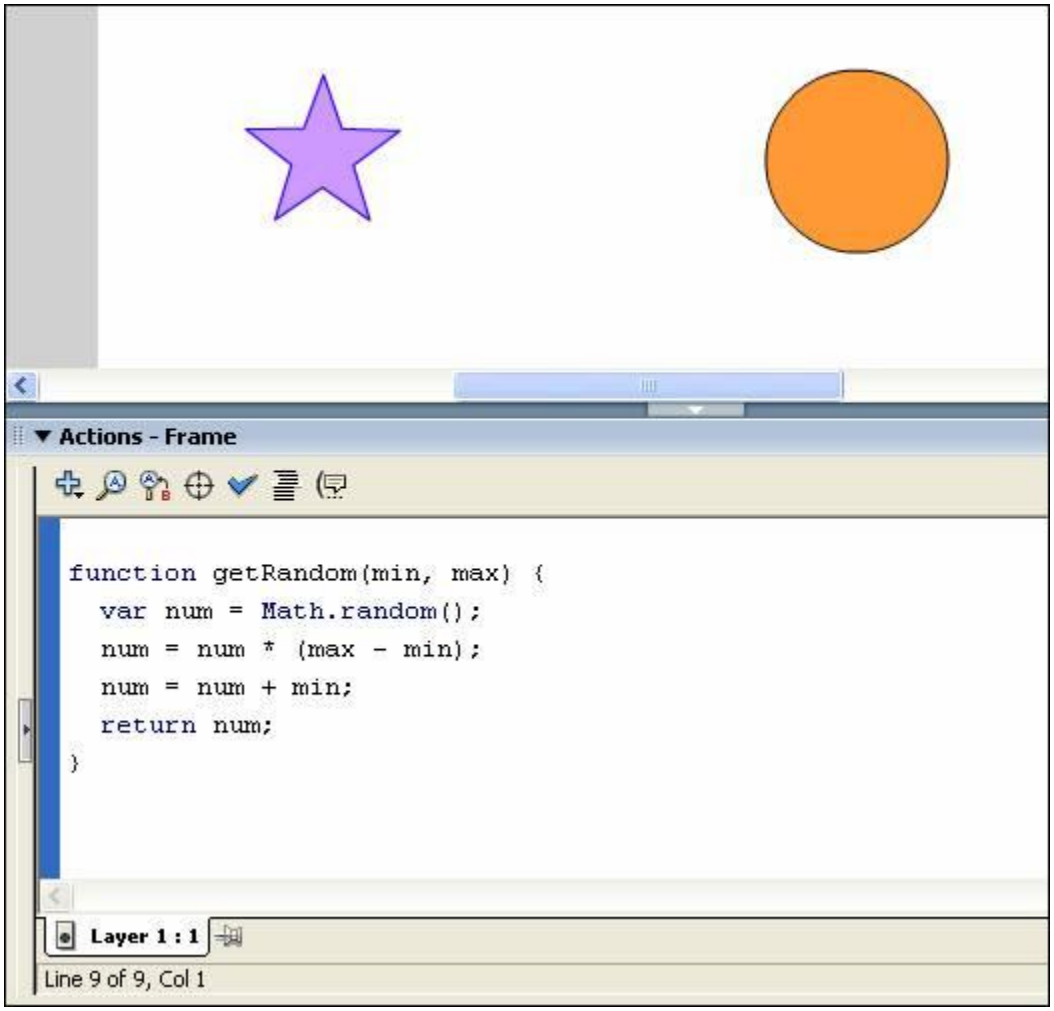
Để cho rõ ràng, ta viết một hàm mới để tính trị ngẫu nhiên trong khoảng tùy ý và dùng hàm đó trong hàm onClipEvent(mouseDown) của sao biển. Chương trình của sao biển chỉ chấp nhận các hàm xử lý tình huống nên bạn phải viết hàm cần thiết ở chỗ khác.

Bấm vào chỗ trống trên sân khấu, bạn thấy chương trình của sao biển biến mất. Bảng Actions – Frame trước mắt bạn dùng để lập trình cho khung đầu tiên (hình 1). Bạn viết hàm tính trị ngẫu nhiên "ngon lành" hơn, gọi là getRandom(min, max), như sau:

?

```
1  function getRandom(min, max) {
2
3  var num = Math.random();
4
```

```
5  num = num * (max - min);
6
7  num = num + min;
8
9  return num;
10
11 }
```



Hàm getRandom(min, max) cho trị ngẫu nhiên trong khoảng từ min đến max bằng cách nhân trị trả về của hàm Math.random() với khoảng cách giữa max và min, tức max – min, rồi cộng kết quả đó với min. Câu lệnh return num; làm cho hàm getRandom(min, max) trả về kết quả tính toán khi được gọi. Đây là lần đầu tiên bạn viết một hàm có trả về một trị số.

Ngoài ra, chắc bạn chú ý đến từ var trong câu lệnh var num = Math.random();. Câu lệnh này tạo ra biến (variable) mang tên num để chứa trị trả về bởi hàm Math.random(). Viết từ var khi tạo ra biến num, ta ngụ ý rằng biến num là biến được tạo ra tạm thời trong hàm getRandom(). Biến như vây gọi là biến cục bộ, được xóa khỏi bộ nhớ máy khi hàm getRandom() kết thúc công việc.
Bạn bấm vào sao biển để trở về với chương trình của sao biển trong bảng Actions -Movie Clip . Trong hàm onClipEvent(mouseDown), bạn gọi hàm getRandom() vừa viết để thử tính trị ngẫu nhiên từ 0 đến 400:

```
?
   onClipEvent(mouseDown) {
1
2  if(hitTest(_root._xmouse, _root._ymouse, true))
3  {
4
5  trace("Trị ngẫu nhiên: " + _root.getRandom(0,
6  400));
7
8  score++;
9
10 trace("Điểm: " + score);
11
12 }
13
   }
```

Có lẽ bạn thắc mắc: “Vì sao phải viết _root.getRandom(0, 400), thay vì viết đơn giản getRandom(0, 400)?”. Nếu bạn không dùng tham chiếu _root, khi

xem xét chương trình của sao biển, Flash không biết rằng phải đọc hàm getRandom() ở khung 1 của thời tuyến chính, khác với thời tuyến được dùng bên trong thể hiện của nhân vật sao biển.

Thử chạy chương trình và bấm nhiều lần vào sao biển, bạn thấy rõ hàm getRandom() của ta cho trị ngẫu nhiên nhỏ hơn 400.

Để sao biển di chuyển khắp sân khấu, hoành độ _x của sao biển phải có trị bất kỳ nhỏ hơn chiều rộng sân khấu và tung độ _y phải có trị bất kỳ nhỏ hơn chiều cao sân khấu. Bấm vào chỗ trống trên sân khấu để thôi chọn sao biển, bạn thấy lại chương trình ở thời tuyến chính. Bạn chỉnh sửa và bổ sung để có nội dung như sau:

?

```
1  function getRandomX() {
2
3  return getRandom(0, 550);
4
5  }
6
7  function getRandomY() {
8
9  return getRandom(0, 400);
10
11 }
12
13 function getRandom(min, max) {
14
15 return Math.random()*(max - min) + min;
```

```
16
17 }
```

Như bạn thấy, ta định nghĩa thêm hai hàm mới: getRandomX() và getRandomY().

Hàm getRandomX() dùng để tạo ra trị ngẫu nhiên lớn hơn hoặc bằng 0 và nhỏ hơn 550. Trong đó, 550 là chiều rộng mặc định của sân khấu. Hàm getRandomX() thích hợp cho việc tạo ra hoành độ ngẫu nhiên bao quát chiều rộng sân khấu. Hàm getRandomX() không làm gì nhiều, chỉ dựa hoàn toàn vào tính toán của hàm getRandom(min, max). Chiều cao mặc định của sân khấu là 400, do vậy bạn hiểu ngay hàm getRandomY() được viết ra để dùng vào việc gì.

Ngoài ra, vì bạn đã hiểu cách tính toán của hàm getRandom(min, max), ta viết lại nội dung của hàm đó ở dạng gọn hơn, có ý nghĩa tương đương.

Chuyển qua chương trình của sao biển (bấm vào sao biển), bạn chỉnh sửa như sau:

?

```
1  onClipEvent(mouseDown) {
2
3  if(hitTest(_root._xmouse, _root._ymouse, true))
4  {
5
6  _x = _root.getRandomX();
7
8  _y = _root.getRandomY();
9
10 score++;
```

```
11
12 trace("Điểm: " + score);
13
14 }
15
   }
```

Thử chạy chương trình, bạn thấy lần này sao biển thực sự di chuyển ngẫu nhiên mỗi khi được bấm trúng.

Bài 15 :Liên lạc giữa các thể hiện – Tự học lập trình Flash

Bạn đã làm cho sao biển xê dịch ngẫu nhiên và tăng thêm 1 điểm cho người chơi mỗi khi sao biển được bấm trúng. Khi đó, theo dự định, ta còn phải làm cho cá chuyển động nhanh hơn. Điểm của người chơi càng cao, cá bơi càng nhanh. Có như vậy, trò chơi mới hào hứng! Chắc chắn cá sẽ bắt được con trỏ chuột vào lúc nào đó. Khi cá bắt được con trỏ chuột, trò chơi cần trở lại từ đầu: điểm của người chơi được gán lại trị số 0. Điểm cao nhất đạt được sẽ thể hiện "đẳng cấp" của mỗi người chơi.

Với mục tiêu như vậy, sao biển cần liên lạc với cá để yêu cầu tăng tốc. Ngược lại, cá phải yêu cầu sao biển cho người chơi điểm 0 khi cá bắt được con trỏ chuột. Để cá và sao biển có thể "nói chuyện" với nhau, trước hết bạn phải đặt tên cho chúng. Cá là thể hiện của nhân vật Fish, còn sao biển là thể hiện của

nhân vật Star, cả hai thể hiện này đều chưa có tên riêng.
Bạn hãy bấm vào cá và ấn Ctrl+F3 để mở bảng Properties. Bạn bấm vào ô có dòng chữ <Instance Name>, gõ fish và gõ Enter (đặt tên cho cá là fish).
Bạn bấm chọn sao biển. Bảng Properties thay đổi, tương ứng với sao biển. Bạn bấm vào ô có dòng chữ <Instance Name>, gõ star và gõ Enter (đặt tên ngắn gọn cho sao biển là star). Xong, bạn bấm vào thanh tiêu đề của bảng Properties để tạm dẹp nó đi.
Bạn mở bảng Actions (trình bày chương trình của sao biển), viết thêm một câu lệnh trong hàm onClipEvent(mouseDown) như sau:
?

```
1  onClipEvent(mouseDown) {
2
3  if(hitTest(_root._xmouse, _root._ymouse, true))
4  {
5
6  _x = _root.getRandomX();
7
8  _y = _root.getRandomY();
9
10 score++;
11
```

```
12 trace("Điểm: " + score);
13
14 _root.fish.step++;
15
16 }
17
   }
```

Câu lệnh mà bạn vừa viết tăng thêm 1 cho trị số của biến step bên trong cá. Để diễn đạt biến step của cá trong chương trình của sao biển, bạn phải ghi _root.fish.step, chứ không thể ghi đơn giản fish.step. Tham chiếu _root trỏ đến sân khấu, nơi lưu giữ “tên tuổi” của các “diễn viên”.

Bạn cần điều chỉnh thêm chút xíu trước khi chạy thử chương trình: bấm-phải vào cá, trỏ vào mục Arrange trên trình đơn vừa hiện ra, chọn mục Bring to Front. Nhờ vậy, cá được đặt phía trước sao biển, gây “khó dễ” cho bạn nhiều hơn.

Thử chạy chương trình, bạn thấy câu lệnh mới có hiệu lực rõ ràng: khi điểm của bạn càng tăng (bắt được càng nhiều sao biển), cá càng hoảng hốt, ra sức bảo vệ sao biển. Khi cá bắt được con trỏ chuột, người chơi không hề hấn gì. Điều này không công bằng.

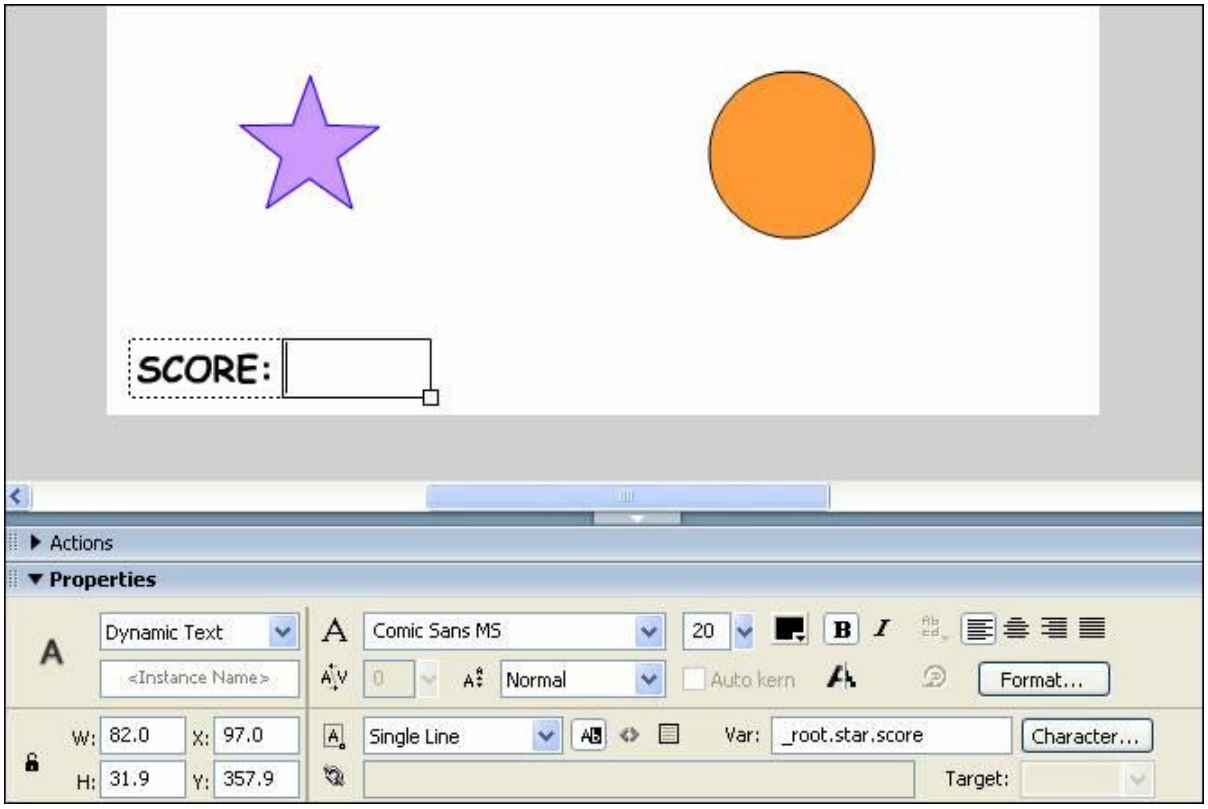
Bạn hãy đóng cửa sổ chương trình và nhìn vào bảng Actions (lúc này đang trình bày chương trình của cá). Bạn chú ý trường hợp “bắt được con trỏ chuột” diễn đạt ở phần cuối của hàm onClipEvent(enterFrame). Bạn chỉnh sửa như sau:

?

```
1  onClipEvent(enterFrame) {
2
3  ...
4
5  else if(caught == true) {
6
7  trace("Bắt được rồi nhé!");
8
9  _root.star.score = 0;
10
11 step = 5;
12
13 }
14
15 }
```

Câu lệnh _root.star.score = 0; gán trị số 0 cho biến score bên trong sao biển, buộc người chơi trở lại mức xuất phát. Để diễn đạt biến score của sao biển trong chương trình của cá, bạn phải viết _root.star.score, chứ không thể viết

đơn giản star.score.
Câu lệnh step = 5; làm cho cá “bình tĩnh” trở lại, bơi chậm như lúc đầu sau khi bắt được con trỏ chuột.
Chạy lại chương trình, vừa chơi vừa liếc nhìn bảng Output, bạn sẽ thấy sự công bằng của trò chơi được thiết lập: khi cá bắt kịp con trỏ chuột, “điểm tích lũy” của bạn mất sạch!
Khi trò chơi của bạn chạy trong trang web, người chơi không thấy bảng Output. Do vậy, bạn cần hiển thị điểm ngay trên sân khấu.
Bạn bấm vào công cụ Text Tool (hoặc gõ phím T), căng một khung nhỏ ở góc dưới, bên trái sân khấu và gõ SCORE:. Bạn mở lại bảng Properties bằng cách bấm vào thanh tiêu đề của bảng đó. Bấm vào ô Text Type trong bảng Properties, bạn chọn Static Text (thay cho Dynamic Text), quy định rằng khung chữ của bạn có nội dung cố định, không thay đổi. Người ta gọi đó là khung chữ tĩnh.
Nếu cần thay đổi phông chữ và cỡ chữ, bạn chọn dòng chữ SCORE: (kéo chuột ngang qua dòng chữ), chọn phông chữ trong ô Font, bấm kép vào ô Font Size và gõ trị số tùy ý (20 chẳng hạn).
Bạn căng một khung khác bên phải dòng chữ SCORE: để tạo khung chữ mới (hình 1). Trong bảng Properties (lúc này tương ứng với khung chữ mới), bạn chọn phông chữ và cỡ chữ giống như dòng chữ SCORE. Bấm vào ô Var, bạn gõ _root.star.score và gõ Enter. Thao tác như vậy tạo ra khung chữ động trình bày trị số của biến _root.star.score (nội dung của khung chữ thay đổi theo trị số của biến _root.star.score).
Chạy thử chương trình, bạn thấy khung chữ động thể hiện đúng điểm số của mình (so với thông báo ở bảng Output).



Theo echip.com

Bài 16 : Vẽ sao biển – Tự học lập trình Flash

Trong trò chơi "bắt sao biển" đang thực hiện, mỗi khi cá bắt kịp con trỏ chuột, cá yêu cầu sao biển cho điểm của người chơi trở về trị số 0. Tuy nhiên, vào lúc ấy người chơi không kịp chú ý rằng mình đã đạt được bao nhiêu điểm. Đối với người chơi, điều quan trọng là ghi nhớ điểm cao nhất mà họ đạt được sau nhiều lần chơi. Do vậy, bên trong sao biển, ta nên tạo thêm một biến để ghi nhớ điểm cao nhất đạt được.

Cụ thể, bạn bấm vào sao biển trên sân khấu, mở bảng Actions và viết thêm vào chương trình của sao biển như sau:

?

```
onClipEvent(load) {

score = 0;

highscore = 0;

}

onClipEvent(mouseDown) {

if(hitTest(_root._xmouse, _root._ymouse, true))
{

_x = _root.getRandomX();

_y = _root.getRandomY();

score++;

```

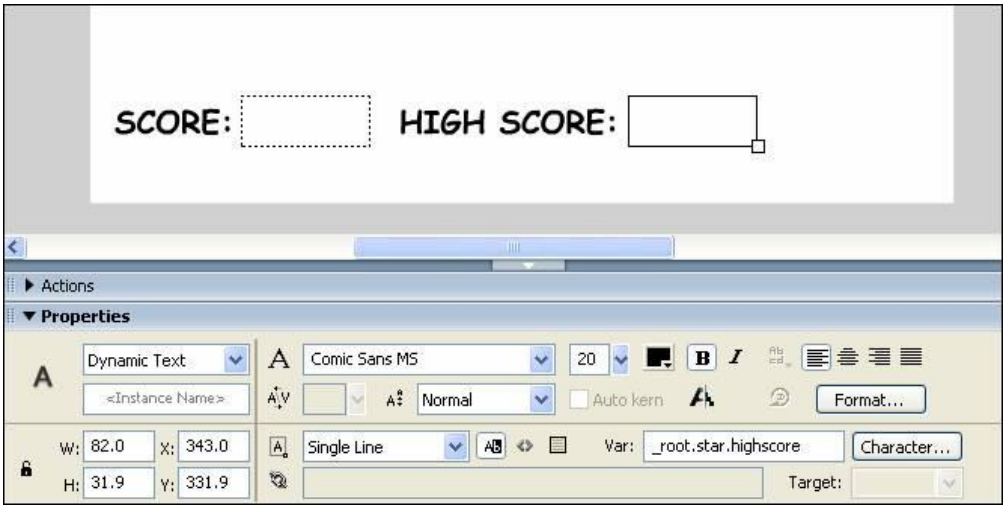
```
19 if(score > highscore)
20
21 highscore = score;
22
23 trace("Điểm: " + score);
24
25 _root.fish.step++;
26
27 }
28
29 }
```

Trong hàm onClipEvent(load), ta tạo ra biến mới highscore với trị ban đầu là 0. Sau mỗi lần tăng điểm cho người chơi, ta đều xét xem điểm mới có cao hơn trị số của biến highscore hay không. Nếu cao hơn, ta gán điểm mới cho biến highscore. Khi biến score trở về trị số 0, biến highscore vẫn lưu giữ điểm cao nhất của người chơi.

Bạn hãy đóng bảng Actions. Việc cần làm tiếp theo là hiển thị điểm cao nhất của người chơi. Với công cụ chọn trong tay, bạn căng khung chọn bao quanh hai khung chữ hiện có (một khung chữ tĩnh có dòng chữ SCORE và một khung chữ động hiển thị trị số của biến score) và ấn Ctrl+C để sao chép. Bạn ấn Ctrl+V để dán bản sao vào sân khấu, bấm kép vào khung chữ SCORE,

sửa nội dung thành HIGHSCORE. Bấm vào khung chữ động kế bên, mở bảng Properties, bạn sửa nội dung trong ô Var thành _root.star.highscore và gõ Enter (hình 1). Nhờ vậy, khung chữ động mới sẽ hiển thị trị số của biến highscore.

Bạn nên chạy chương trình ngay để biết chắc khung chữ động mới hoạt động tốt như mong muốn. Bạn có thể rủ rê người khác tranh tài cùng bạn, xem ai đạt điểm highscore cao hơn trong năm phút chẳng hạn.



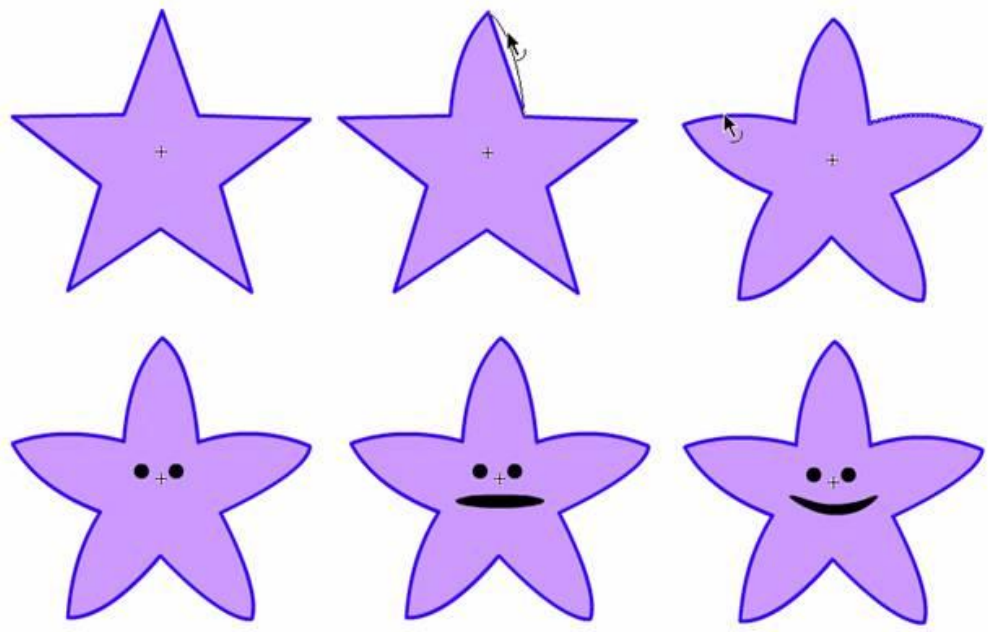
Khi trò chơi đã chạy tốt, bạn có thể bước qua giai đoạn chăm chút “ngoại hình”. Ta bắt đầu với việc “tút lại vẻ đẹp giai” của sao biển. Bạn bấm kép vào sao biển để chuyển qua chế độ chỉnh sửa nhân vật sao biển. Muốn lấy tầm nhìn gần cho tiện việc “tân trang”, bạn chọn công cụ Zoom Tool (hoặc gõ phím Z) và bấm vào sao biển vài lần. Nếu lỡ nhìn quá gần, bạn chọn Reduce ở hộp công cụ và bấm vào sao biển để “lui ra xa”.

Bạn “cầm lấy” công cụ chọn , bấm vào đâu đó bên ngoài sao biển để “thôi chọn” sao biển. Đưa con trỏ chuột vào một cạnh của sao biển, bạn thấy con trỏ chuột đổi dạng , tỏ ý sẵn sàng giúp bạn chỉnh sửa đường nét của sao biển. Bạn nắm kéo cạnh sao biển để uốn cong tùy ý (hình 2). Thật tuyệt! Cứ thế, bạn uốn cong mọi cạnh của hình sao khô cứng ban đầu để có được đường nét mềm mại.

Khi đã hài lòng về hình dạng sao biển, bạn bấm kép vào dòng chữ Layer 1 ở bảng Timeline , gõ tên Thân . Thao tác đổi tên lớp như vậy giúp bạn nhớ rằng lớp đang xét chứa thân của sao biển. Bạn bấm vào dấu chấm bên dưới hình ổ khóa để lớp Thân được “khóa cứng”, không thể chỉnh sửa.

Bạn tạo lớp mới Layer 2 bằng cách bấm vào biểu tượng Insert Layer , rồi đổi tên Layer 2 thành Mắt, ngụ ý rằng lớp mới sẽ được dùng để chứa… mắt của sao biển.

Vẽ mắt rất dễ, bạn chọn công cụ Oval Tool , bấm vào ô Fill Color , chọn màu tô là màu đen và vẽ ra một hình tròn nhỏ. Bạn bấm vào công cụ chọn , căng khung chọn bao quanh hình tròn đen vừa vẽ, ấn Ctrl+C để sao chép, rồi ấn Ctrl+V để dán bản sao vào lớp Mắt. Bạn lần lượt kéo hai hình tròn đen vào giữa sao biển, tạo nên cặp mắt.



Bạn khóa lớp Mắt, tạo lớp mới mang tên Miệng. Bạn vẽ một hình ô-van dẹt bên dưới cặp mắt của sao biển, dùng công cụ chọn để chỉnh dạng ô-van, tạo nên “miệng cười xinh” (hình 2).

Xong xuôi, bạn bấm vào mục Scene 1 để thoát khỏi chế độ chỉnh sửa nhân

vật sao biển, trở về với sân khấu. Bạn đừng quên ấn Ctrl+S để lưu lại “thành quả” của mình

Bài 17 : Vẽ cá nóc – Tự học lập trình Flash

Mở lại trò chơi “bắt sao biển” trong Flash, bạn bấm kép vào hình tròn mà ta đã gọi là… cá nóc để chuyển qua chế độ chỉnh sửa nhân vật Fish. Bạn gõ phím Z để lấy công cụ Zoom Tool , căng khung chọn bao quanh hình tròn để có được tầm nhìn gần, tập trung vào hình tròn.

Bạn đổi tên lớp Layer 1 thành Thân: bấm kép vào Layer 1 ở bảng thời tuyến, gõ Thân . Tên lớp như vậy ngụ ý rằng lớp đang xét chỉ dùng để chứa phần thân cá. Nhờ đặt mỗi bộ phận của hình vẽ vào lớp riêng, bạn dễ dàng chỉnh sửa hình vẽ sau này.

Bạn tạo lớp mới Layer 2 bằng cách bấm vào biểu tượng Insert Layer , đổi tên Layer 2 thành Mắt. Để vẽ mắt, bạn chọn công cụ Oval Tool . Trước khi vẽ, bạn nhớ chọn màu tô thích hợp ở ô Fill Color .

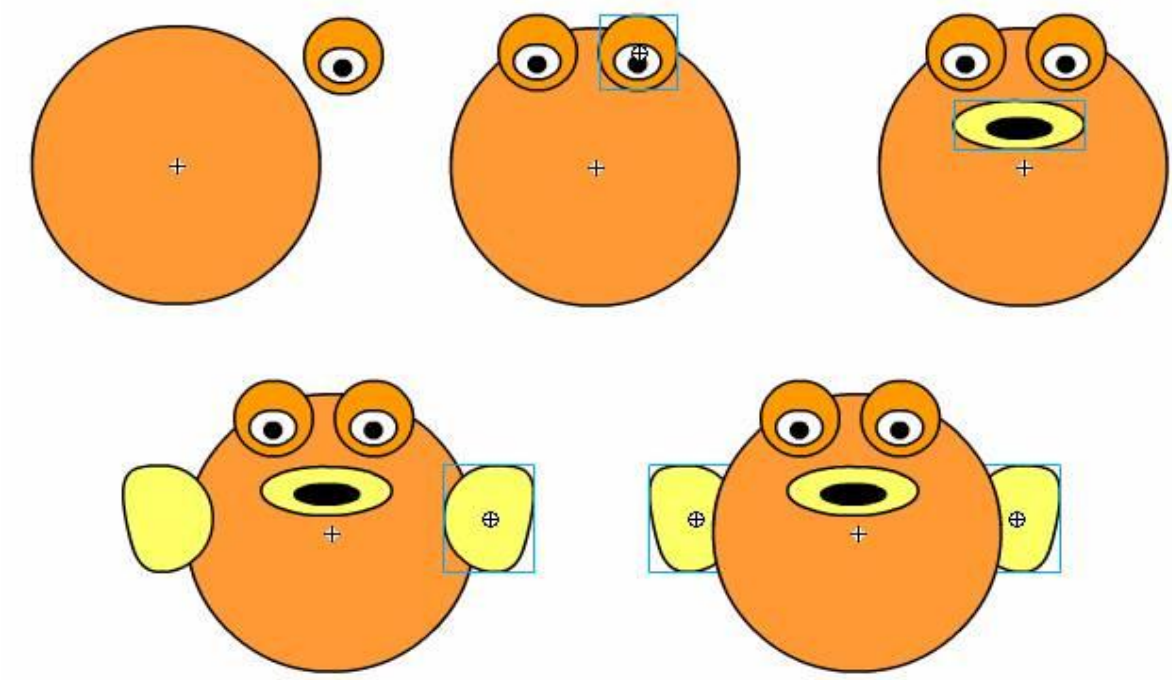
Ở bên ngoài thân cá, bạn lần lượt vẽ ba hình ô-van chồng nhau để tạo nên một con mắt . Dự trù sau này sẽ làm cho con mắt “nhúc nhích”, ta chuyển con mắt vừa vẽ thành nhân vật hẳn hòi. Cụ thể, bạn lấy công cụ chọn , căng khung chọn bao quanh con mắt, gõ phím F8 để mở hộp thoại Convert to Symbol, gõ tên Eye và gõ Enter. Khi đặt tên cho nhân vật trong Flash, bạn nên dùng tiếng Việt không dấu hoặc tiếng Anh cho tiện việc lập trình.

Nếu cần co dãn con mắt để có kích cỡ như ý, bạn bấm vào công cụ Free Transform Tool và bấm vào con mắt. Bạn kéo một trong các dấu vuông màu đen vừa xuất hiện quanh con mắt cho đến khi đạt được kết quả mong muốn.

Trỏ vào con mắt sao cho con trỏ chuột đổi dạng thành “mũi tên bốn đầu” , bạn kéo hình đó vào phần trên thân cá (hình 1), đến gần đỉnh hình tròn. Bạn ấn Ctrl+C để sao chép con mắt, ấn Ctrl+V để dán con mắt thứ hai vào lớp Mắt và kéo con mắt thứ hai đến vị trí thích hợp, ngang với con mắt thứ nhất.

Bạn tạo lớp mới, đặt tên là Miệng và vẽ hai hình ô-van dẹt chồng nhau để làm miệng cá. Bạn nên vẽ bên ngoài thân cá cho dễ chỉnh sửa. Để hai hình ô-van vừa vẽ dính liền nhau thành một nhóm (group), thuận tiện cho việc di chuyển, bạn lấy công cụ chọn , căng khung chọn bao quanh hai hình ô-van và ấn Ctrl+G. Xong, bạn kéo hai hình ô-van đó vào thân cá, đặt ngay dưới cặp mắt, tạo thành miệng cá.

Bạn tạo lớp mới, đặt tên là Vây, vẽ hình ô-van và dùng công cụ chọn để làm méo hình ô-van đó ở phía trái, tạo thành vây cá . Ta cũng dự trù làm cho vây cá ve vẩy nên cần chuyển vây cá thành nhân vật. Bạn căng khung chọn bao quanh vây cá vừa vẽ, gõ phím F8, gõ tên Fin và gõ Enter. Vây cá trở thành nhân vật mang tên Fin.



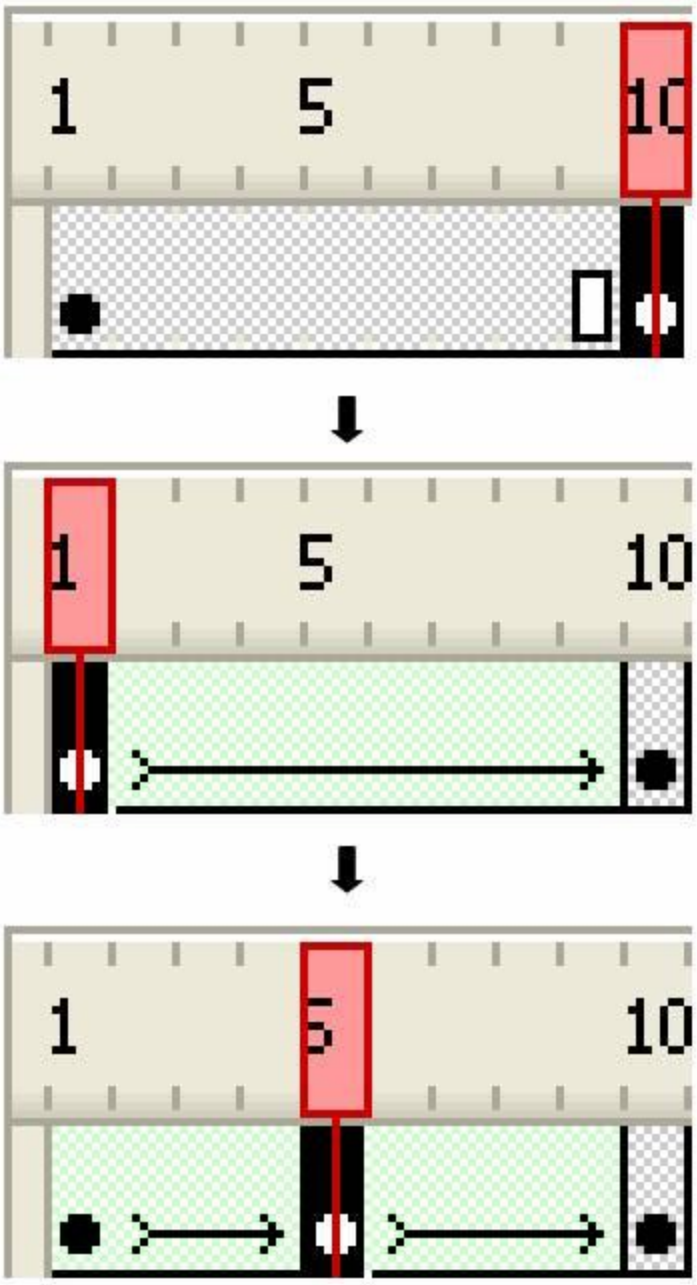
Bạn bấm kép vào vây cá để chuyển qua chế độ chỉnh sửa nhân vật Fin. Trong bảng thời tuyến của vây cá, bạn bấm vào khung 10, gõ phím F6. Thao tác như vậy làm cho thời tuyến của vây cá gồm 10 khung (thay vì chỉ có một khung như trước). Khung 10 trở thành khung chốt và có nội dung giống hệt khung 1 (hình 2).

Bạn bấm vào khung 1, ấn Ctrl+F3 để mở bảng Properties. Trong ô Tween, bạn chọn mục Shape. Thao tác này nói với Flash rằng bạn muốn tạo ra những hình trung gian cho sự biến đổi hình dạng giữa hai khung chốt. Dấu mũi tên chạy từ khung 1 đến khung 10 xuất hiện, chứng tỏ Flash đã hiểu ý bạn.

Vì vây cá ở khung 1 và khung 10 giống hệt nhau nên việc tạo ra các hình trung gian chưa thực sự có ý nghĩa. Bạn bấm vào khung 5, gõ phím F6. Khung 5 cũng trở thành khung chốt. Bạn chọn công cụ co dãn , kéo dấu vuông ở giữa cạnh trái vây cá qua phải để co hẹp vây cá . Vây cá ở khung 5 có dạng khác với vây cá ở khung 1 và khung 10. Phép biến hình của Flash trở nên có ý nghĩa: vây cá sẽ co hẹp dần từ khung 1 đến khung 5 và nở rộng dần từ khung 6 đến khung 10 (trở lại hình dạng cũ).

Bạn bấm vào mục Fish phía trên bảng thời tuyến để trở về với nhân vật Fish, sao chép vây cá hiện có để có vây thứ hai. Bạn chọn Modify > Transform > Flip Horizontal để “lật ngang” vây thứ hai. Bạn đặt hai vây ở hai bên thân cá, kéo lớp Vây xuống dưới lớp Thân để vây nằm sau thân (hình 1).

Bạn ấn Ctrl+Enter, xem thử cá hoạt động ra sao nha.



Theo echip.com

Bài 18 : Diễn hoạt bộ phận – Tự học lập trình Flash

Khi vẽ cá nóc, bạn đã diễn hoạt (animate) cho vây của nó, khiến vây cá ve vẩy thật sinh động. Ta hãy thực hiện một việc khó hơn: diễn hoạt mắt cá.

Nói cụ thể, bạn sẽ làm cho cá biết “liếc ngang liếc dọc”: tròng đen của mắt luôn dịch chuyển bên trong tròng trắng, hướng về phía con trỏ chuột. Nhờ vậy, người chơi có cảm giác cá nóc của ta thực sự khôn lanh. Diễn hoạt cho một bộ phận của hình vẽ, làm cho nó hết “cứng đơ” là việc rất thường xuyên trong công đoạn tinh chỉnh trò chơi.

Sau khi mở lại trò chơi “bắt sao biển” trong cửa sổ Flash, bạn gõ phím F11 (hoặc ấn Ctrl+L) để mở bảng Library. Trong danh sách nhân vật, bạn chọn nhân vật Eye. Khung hình phía trên danh sách lập tức hiển thị mắt cá. Bấm kép vào hình mắt cá trong bảng Library, bạn chuyển qua chế độ chỉnh sửa nhân vật Eye. Sân khấu biến mất, chỉ còn hình mắt cá trên nền trắng trống trải.

Hiện thời, hai mắt của cá là hai thể hiện của nhân vật Eye. Do vậy, khi bạn chỉnh sửa nhân vật Eye, cả hai mắt cá đều thay đổi giống nhau.

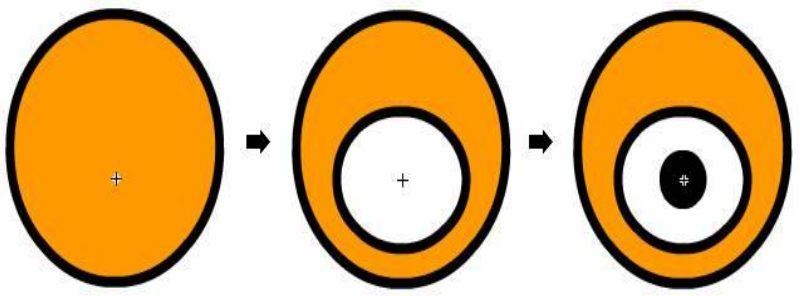
Trước khi chỉnh sửa mắt cá, bạn nên lấy tầm nhìn gần thích hợp. Cách “chỉnh sửa” dễ dàng nhất là xóa hình vẽ tạm hiện có (ấn Ctrl+A để chọn tất cả và gõ phím Delete), rồi vẽ lại một cách cẩn trọng. Mỗi phần của hình cần được đặt trong một lớp riêng.

Bạn vẽ một hình tròn màu cam, di chuyển hình tròn sao cho điểm mốc của nhân vật Eye nằm hơi thấp so với tâm hình tròn (hình 1). Khi chọn hình tròn để di chuyển, bạn nhớ chọn cả đường biên hình tròn (bấm vào hình tròn, giữ phím Shift và bấm vào biên hình tròn). Nếu có gì chưa ưng ý, bạn chỉ cần ấn ngay Ctrl+Z để hủy bỏ thao tác vừa thực hiện.

Bạn tạo lớp mới Layer 2 và vẽ một hình ô-van màu trắng. Bạn chú ý di chuyển hình ô-van sao cho tâm của nó trùng với điểm mốc của nhân vật Eye (hình 1).

Bạn tạo lớp mới Layer 3, vẽ một hình tròn màu đen và cũng di chuyển hình

tròn sao cho tâm của nó trùng với điểm mốc (hình 1). Đó là tròng đen của mắt cá.



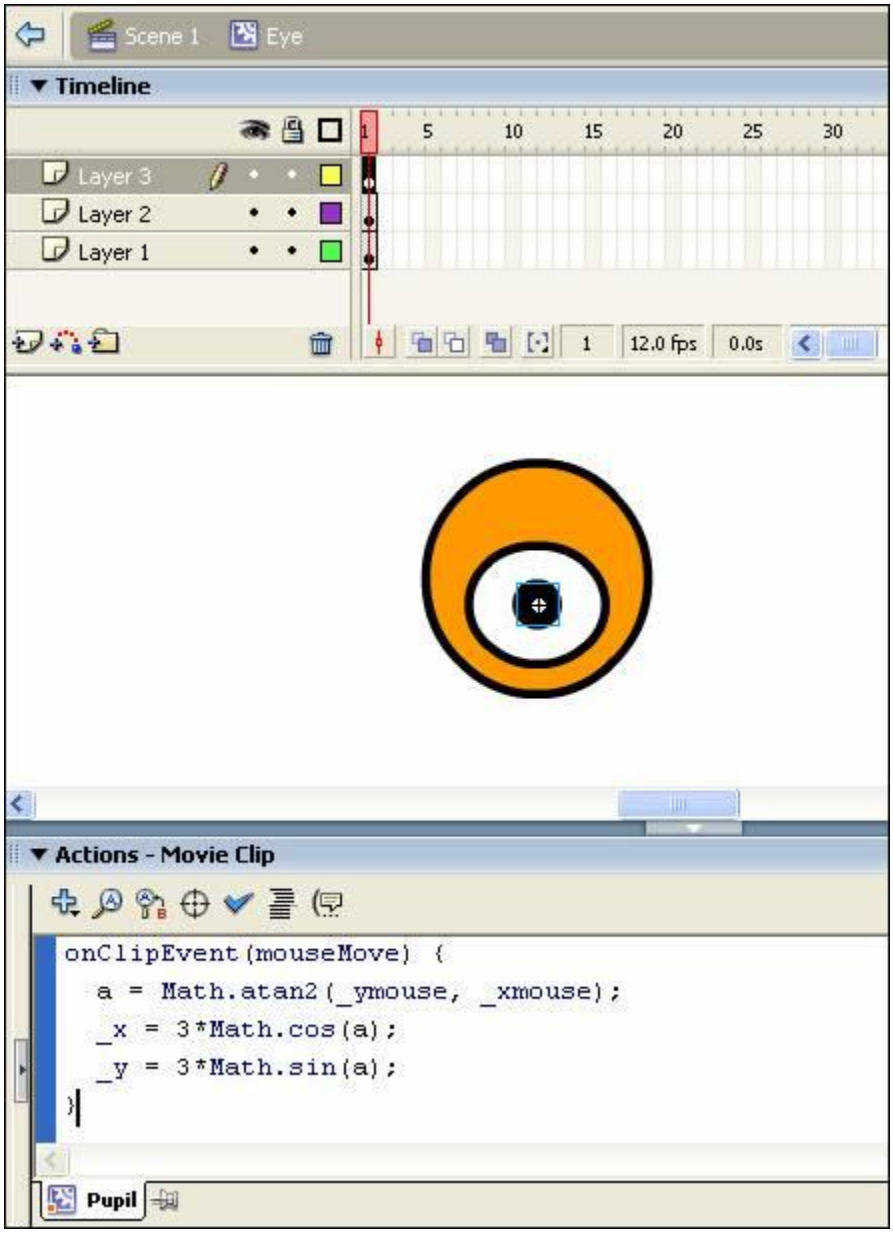
Để có thể lập trình cho tròng đen, ta cần chuyển tròng đen thành nhân vật riêng biệt. Muốn vậy, trước hết bạn chọn tròng đen và đường biên của nó (để khỏi "vướng víu" đường biên của tròng đen, bạn có thể chọn đường biên đó, gõ phím Delete để xóa, rồi dùng công cụ co dãn để điều chỉnh kích cỡ tròng đen sao cho phù hợp). Bạn gõ phím F8 để mở hộp thoại Convert to Symbol, gõ tên Pupil (tròng đen) và gõ Enter. Tròng đen trở thành một thể hiện của nhân vật Pupil.

Với tròng đen đang ở trong tình trạng "được chọn", bạn gõ phím F9 để mở bảng Actions – Movie Clip (hình 2), viết hàm xử lý tình huống "di chuyển chuột" như sau:

?

```
  onClipEvent(mouseMove) {
1
2
  a = Math.atan2(_ymouse, _xmouse);
3
4
  _x = 3*Math.cos(a);
5
```

```
6  _y = 3*Math.sin(a);

7

8  }

9

;
```



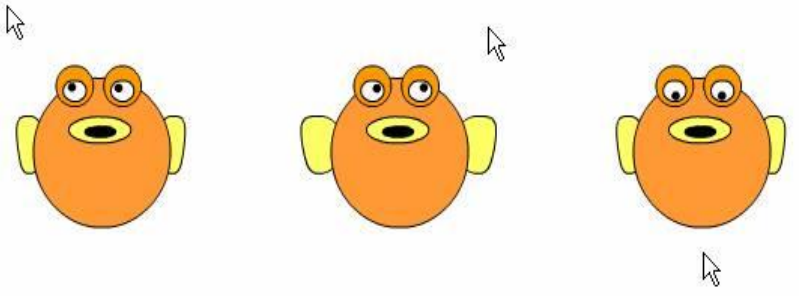
Hàm onClipEvent(mouseMove) được gọi mỗi khi người chơi xê dịch con chuột. Bên trong hàm, ta tính góc nghiêng của tia nối điểm mốc với con trỏ

chuột (góc giữa tia đó và phương ngang) bằng cách dùng hàm atan2() của lớp Math. Bạn chú ý, ta đang ở "bên trong" nhân vật Eye, do đó _xmouse và _ymouse là hoành độ và tung độ của con trỏ chuột so với điểm mốc của nhân vật Eye.

Khi có góc nghiêng cần thiết, ta tính được tọa độ của tròng đen phù hợp với góc nghiêng đó. Lấy độ dịch chuyển của tròng đen so với điểm mốc là 3 pi-xôn, bạn nhân độ dịch chuyển với cosin/sin của góc nghiêng để có hoành độ/tung độ của tròng đen.

Ấn Ctrl+Enter để chạy thử trò chơi, khi "vờn vờn" con trỏ chuột quanh cá nóc, bạn thấy cá luôn đảo mắt dõi theo con trỏ chuột (hình 3).

Bạn có thể chỉnh sửa độ dịch chuyển của tròng đen cho phù hợp với kích cỡ cụ thể của mắt cá do bạn vẽ, nhưng đừng để tròng đen vượt ra ngoài tròng trắng!



Theo echip.com

Bài 19: Hình nền và nhạc nền – Tự học lập trình Flash

Trò chơi Flash đầu tiên của bạn đã gần hoàn chỉnh, bạn có thể thêm vào hình nền và nhạc nền cho "sướng mắt đã tai". Do bạn chưa thật quen với các công cụ vẽ của Flash để có thể tự vẽ hình nền, bạn nên tìm hình

ảnh trên mạng. Chẳng hạn, nếu gõ các từ chốt underwater images trong ô tìm kiếm Google, bạn sẽ tìm được nhiều hình chụp dưới nước.

Giả sử bạn tìm thấy một hình nào đó “coi được”. Sau khi mở trò chơi “bắt sao biển” trong cửa sổ Flash, bạn cần tạo một lớp mới để chứa hình nền. Cụ thể, trong bảng Timeline, bạn bấm Insert Layer để tạo nên lớp mới Layer 2, trỏ vào Layer 2, kéo nó xuống dưới Layer 1. Bạn có thể đổi tên Layer 2 thành Background hoặc Nền để nêu rõ ý nghĩa của lớp mới.

Bạn ấn Ctrl+R (hoặc chọn File > Import > Import to Stage). Trong hộp thoại Import vừa hiện ra, bạn tìm đến thư mục chứa hình cần thiết và bấm kép vào hình đó. Hình được chọn lập tức được đưa vào thư viện và thể hiện trên sân khấu ở lớp nền.

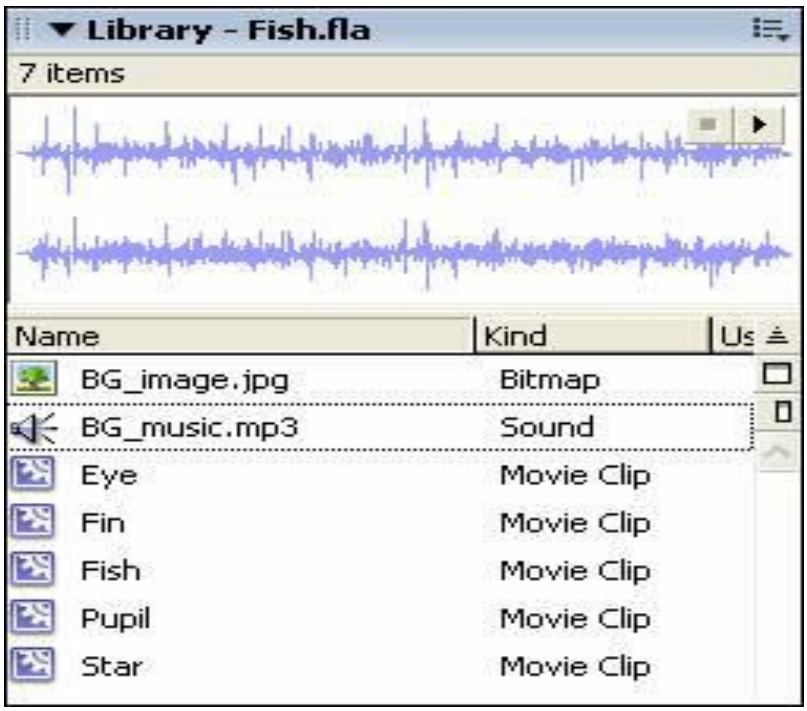
Bạn bấm vào hình nền và ấn Ctrl+F3 để mở bảng Properties, quan sát các thuộc tính của hình nền. Vì sân khấu của ta có kích thước mặc định 550 x 400 (rộng 500 điểm ảnh, cao 400 điểm ảnh), bạn cần điều chỉnh hình nền để có kích thước giống như vậy: sửa trị số trong ô W thành 550 và sửa trị số trong ô H thành 400 (hình 1). Hai ô X và Y đều có trị số mặc định là 0, nghĩa là góc trên, bên trái của hình nền trùng với góc trên, bên trái của sân khấu. Trong phần lớn trường hợp, đó chính là điều ta mong muốn, bạn không cần sửa thêm gì nữa

Để có nhạc nền, bạn cũng có thể tìm trên mạng với các từ chốt background music mp3. Bạn nên chọn đoạn nhạc ngắn, vui vẻ, có thể nghe lại nhiều lần mà không chán. Đoạn nhạc dài không thích hợp với trò chơi Flash vì sẽ làm cho tập tin SWF cuối cùng mà bạn thu được “phình” lên. Mọi thứ đưa lên mạng càng nhỏ gọn càng tốt.

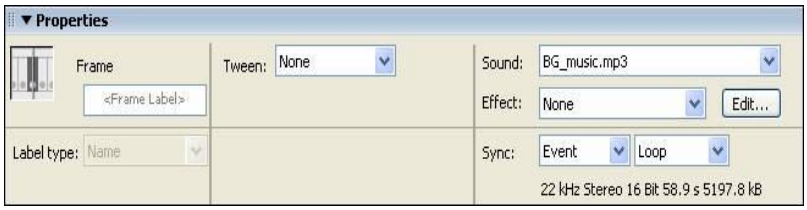
Sau khi có được tập tin MP3 chứa nhạc nền đặt trong thư mục nào đó, bạn trở lại cửa sổ Flash, ấn Ctrl+R để mở hộp thoại Import, tìm chọn tập tin MP3 cần thiết. Đoạn nhạc MP3 đó cũng được đưa vào thư viện.

Để thấy được các thứ đang có trong thư viện, bạn mở bảng Library bằng cách ấn Ctrl+L hoặc gõ phím F11. Trong danh sách của bảng Library, khi bạn chọn đoạn nhạc MP3, tín hiệu âm thanh được thể hiện trong khung phía trên (hình 2). Bạn sẽ thấy hai dòng âm thanh trong khung đó nếu đoạn nhạc đang xét

thuộc loại stereo. Nếu cần nghe lại đoạn nhạc để kiểm tra chất lượng, bạn bấm nút Play . Muốn thôi nghe, bạn bấm nút Stop .



Khác với trường hợp hình nền đã xét, đoạn nhạc của bạn không được đưa vào sân khấu một cách tự động. Bạn phải chủ động quy định nhạc nền bằng cách bấm vào khung 1 (cũng là khung duy nhất) của lớp nền, bấm vào ô Sound trong bảng Properties, chọn đoạn nhạc cần thiết (hình 3). Theo mặc định, đoạn nhạc được chọn như vậy chỉ phát ra một lần. Để đoạn nhạc được lặp lại hoài hoài suốt trò chơi, bạn bấm vào ô Sound Loop, chọn Loop thay cho Repeat. Vậy là đủ, bạn hãy ấn Ctrl+Enter để chạy trò chơi, nghe thử nhạc nền.



Nếu quan sát thư mục chứa tập tin FLA và SWF của trò chơi, bạn sẽ ngạc nhiên vì tập tin SWF (được biên dịch từ tập tin FLA) rất bé so với tập tin FLA. Tập tin SWF tuy chứa mọi thứ cần thiết của trò chơi (chương trình, hình vẽ,

hình nền, âm thanh) nhưng thường nhỏ hơn nhiều so với tập tin MP3 đã dùng nhờ Flash nén âm thanh rất tốt. Bạn có thể đưa nhiều đoạn nhạc vào thư viện để dễ lựa chọn nhưng chỉ đoạn nhạc mà bạn thực sự dùng trong trò chơi mới được lưu trữ trong tập tin SWF.

Bài 20: Câu lệnh tạo thể hiện – Tự học lập trình Flash

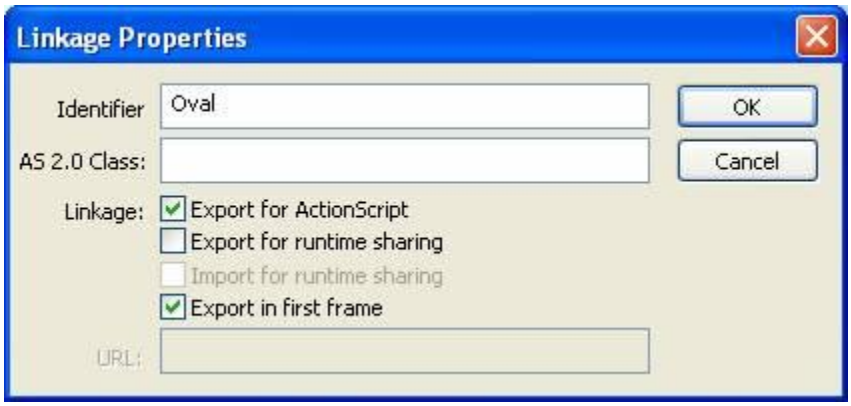
Bạn đã biết cách tạo ra thể hiện của nhân vật: trỏ vào tên nhân vật trong thư viện, kéo vào sân khấu.

Ngoài cách thức thủ công như vậy, ta còn có thể tạo ra thể hiện của nhân vật bằng câu lệnh ActionScript, nghĩa là tạo ra thể hiện vào lúc chạy chương trình. Đó là việc thường làm khi lập trình Flash vì ta không luôn luôn biết trước cần có bao nhiêu thể hiện trên sân khấu. Số lượng thể hiện trên sân khấu có thể thay đổi. Mỗi thể hiện có thể xuất hiện và biến mất tùy lúc. Thử hình dung diễn biến của trò chơi Space Invaders quen thuộc, bạn thấy ngay: việc tạo thể hiện một cách linh hoạt vào lúc chạy là nhu cầu bức thiết.

Để thử tạo thể hiện bằng câu lệnh, bạn hãy mở tập tin Flash mới, vẽ một hình ô-van (có màu tô tùy ý nhưng đừng quá đậm), căng khung chọn bao quanh hình vừa vẽ, gõ phím F8, gõ tên Oval và gõ Enter. Bạn biết quá rõ: thao tác vừa thực hiện tạo ra một nhân vật mang tên Oval. Hình ô-van trên sân khấu trở thành một thể hiện của nhân vật Oval. Gõ phím F11 để mở bảng Library, bạn thấy rõ nhân vật Oval được lưu trữ ở đó.

Bạn xóa thể hiện của nhân vật Oval trên sân khấu (bấm vào hình ô-van, gõ phím Delete). Ta sẽ tạo ra thể hiện của nhân vật Oval trên sân khấu trống trơn bằng câu lệnh thích hợp.

Trước khi làm điều đó, bạn bấm-phải vào tên Oval trong thư viện, chọn Linkage trong trình đơn vừa hiện ra. Trong hộp thoại Linkage Properties, bạn chọn mục Export for ActionScript (hình 1) và bấm OK. Thao tác như vậy nhằm khai báo với Flash rằng: “Khi tạo ra tập tin SWF, nhớ ghi vào đó nhân vật mang tên Oval”. Nếu bạn không khai báo như vậy, Flash sẽ không ghi nhân vật Oval vào tập tin SWF với lý do đơn giản: nhân vật Oval không hiện diện trên sân khấu. Một khi trong tập tin SWF không có nhân vật Oval, chương trình được ghi trong SWF không thể tạo ra thể hiện của nhân vật Oval vào lúc chạy.



Có lẽ bạn hơi ngạc nhiên vì chuyện khai báo nêu trên: “Tại sao Flash không tự động ghi mọi nhân vật trong thư viện vào tập tin SWF, dù có hiện diện trên sân khấu hoặc không?”. Nghĩ kỹ một chút, bạn thấy ngay sự “khó chịu” của Flash mang đến lợi ích cho bạn. Bạn có thể lưu trữ rất nhiều thứ trong thư viện nhưng tập tin sản phẩm SWF luôn nhỏ gọn, chỉ chứa đựng những thứ thực sự được dùng. Trong trường hợp đang xét, tuy nhân vật Oval không có trong sân khấu nhưng ta cần nó vào lúc chạy, do vậy phải “nói trước” để Flash hiểu rõ ý định của ta.

Bạn gõ phím F9 để mở bảng Actions – Frame, nơi dùng để viết chương trình cho khung đầu tiên của thời tuyến chính. Ta dùng thuật ngữ thời tuyến chính

(main timeline) để phân biệt với thời tuyến bên trong mỗi thể hiện. Bạn gõ đoạn mã như sau trong bảng Actions – Frame:

?

```
1 attachMovie("Oval", "oval1", 1);
2
3 attachMovie("Oval", "oval2", 2);
4
5 attachMovie("Oval", "oval3", 3);
```

;

Ta dùng ba câu lệnh như trên để tạo ra ba thể hiện của nhân vật Oval có tên là oval1, oval2 và oval3. Đối mục đầu tiên của hàm attachMovie() là tên nhân vật. Đối mục thứ hai là tên của thể hiện. Đối mục thứ ba cho biết thể hiện nằm ở "độ sâu" nào trên sân khấu. Với cách viết như trên, thể hiện oval1 nằm sâu nhất trên sân khấu. Thể hiện oval2 nằm trên oval1 và thể hiện oval3 nằm trên oval2.

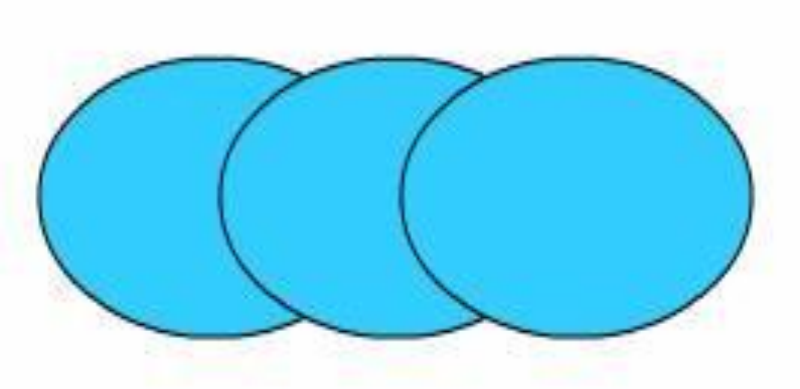
Ấn Ctrl+Enter để chạy chương trình, bạn thấy trên sân khấu chỉ có một hình ô-van. Đó chính là thể hiện oval3 nằm trên cùng, chồng khít lên hai thể hiện oval2 và oval1. Theo mặc định, hàm attachMovie() tạo ra thể hiện tại điểm gốc (0, 0) trên sân khấu. Muốn thấy rõ cả ba thể hiện, bạn cần xê dịch chúng đôi chút. Trở lại với chương trình đang viết, bạn gõ thêm đoạn mã như sau:

?

```
1 oval1._x += 100; oval1._y += 100;
```

```
2

3 oval2._x += 160; oval2._y += 100;

4

5 oval3._x += 220; oval3._y += 100;

;
```

Đoạn mã vừa thêm quy định vị trí cho các thể hiện bằng cách thay đổi trị của biến _x và _y bên trong từng thể hiện. Câu lệnh oval1._x += 100; làm cho hoành độ _x của thể hiện oval1 tăng thêm 100 (xê dịch qua phải 100 điểm ảnh). Câu lệnh oval1._y += 100; làm cho tung độ _y của thể hiện oval1 tăng thêm 100 (xê dịch xuống dưới 100 điểm ảnh). Các câu lệnh tiếp theo làm cho hoành độ _x của thể hiện oval2 và oval3 tăng nhiều hơn oval1. Thử chạy chương trình, bạn có kết quả như hình 2.



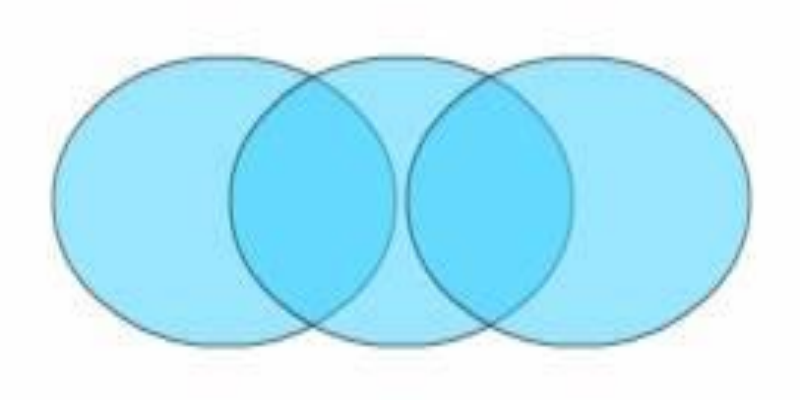
Với các thể hiện được tạo ra vào lúc chạy, bạn có thể thay đổi các thuộc tính của chúng một cách bình thường. Chẳng hạn, bạn có thể thay đổi biến _alpha trong từng thể hiện:

?

```
1 oval1._alpha = 50;
```

```
2

3 oval2._alpha = 50;

4

5 oval3._alpha = 50;

;
```

Đoạn mã viết thêm như trên làm cho các thể hiện trở nên nửa trong suốt (hình 3).



Bài 21 : Sao chép thể hiện – Tự học lập trình Flash

Bạn đã thử nghiệm các câu lệnh tạo thể hiện. Từ sân khấu trống trơn, các thể hiện được tạo ra vào lúc chạy chương trình.

Trong chương trình, bạn có thể thay đổi các thuộc tính của thể hiện như vậy một cách bình thường. Bạn hãy mở lại tập tin FLA chứa các câu lệnh thử nghiệm. Ta sẽ tiếp tục tìm hiểu việc xóa bỏ thể hiện và sao chép thể hiện vào lúc chạy.

Trong bảng Actions – Frame chứa chương trình ứng với khung 1, bạn gõ thêm một câu lệnh sau đoạn mã đã có:

?

```
1  attachMovie("Oval", "oval1", 1);
2
3  attachMovie("Oval", "oval2", 2);
4
5  attachMovie("Oval", "oval3", 3);
6
7  ...
8
9  oval1._alpha = 50;
10
11 oval2._alpha = 50;
12
13 oval3._alpha = 50;
14
15 oval3.removeMovieClip();
```

;

Câu lệnh vừa thêm dùng để xóa bỏ thể hiện mang tên oval3. Nói đúng hơn, ta gọi hàm removeMovieClip() của oval3 để yêu cầu nó “tự hủy”. Ấn Ctrl+Enter

để thử chạy chương trình, bạn thấy quả thực thể hiện oval3 mất đi, chỉ còn hai thể hiện oval1 và oval2.

Vào lúc chạy chương trình, ta còn có thể tạo ra bản sao của thể hiện nào đó. Bạn thử viết thêm câu lệnh như sau:

?

```
1 oval1.duplicateMovieClip("oval3", 3);
```

;

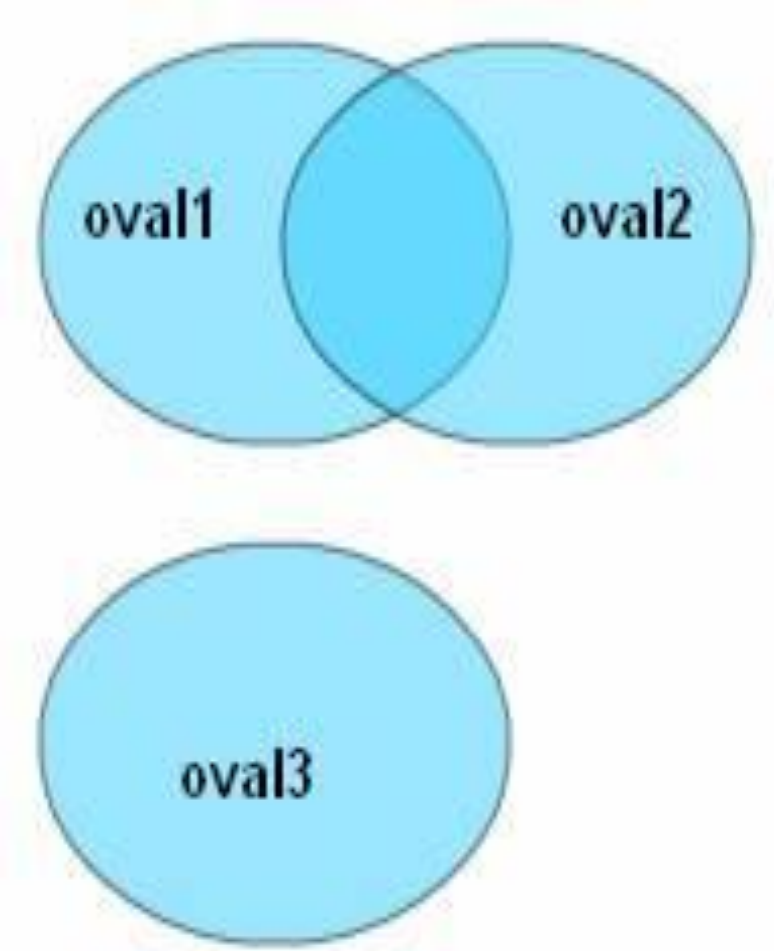
Câu lệnh vừa viết yêu cầu oval1 tạo ra bản sao của chính nó. Bản sao được đặt tên là oval3 và có độ sâu là 3 (nằm trên oval1 và oval2). Thử chạy chương trình, bạn khó nhận ra sự hiện diện của bản sao vì bản sao oval3 chồng khít lên oval1. Để thấy rõ bản sao oval3, bạn viết thêm câu lệnh như sau để dịch chuyển nó xuống dưới:

?

```
1 oval3._y += 100;
```

;

Bạn thu được kết quả như hình 1 (tên của các thể hiện được ghi vào hình để bạn dễ phân biệt). Bản sao oval3 hoàn toàn giống oval1, cũng nửa trong suốt.



Có thể bạn đang ngờ ngợ: “Lẽ ra nên đặt tên cho bản sao là oval4 vì tên oval3 đã được dùng trước đó?”. Đúng là ta đã dùng tên oval3 cho một thể hiện khác, được tạo ra trước. Tuy nhiên, thể hiện đó đã bị xóa bỏ nên bạn có thể dùng lại tên oval3 mà không gây ra xung đột.

Nhân tiện, xin… nói nhỏ với bạn rằng nếu ta tạo ra một thể hiện có cùng độ sâu với thể hiện nào đó đã có, thể hiện mới sẽ thay thế thể hiện đã có ở cùng độ sâu. Cho dù bạn không dùng câu lệnh oval3.removeMovieClip(); để xóa bỏ thể hiện oval3, câu lệnh oval1.duplicateMovieClip(“oval3″, 3); tự động xóa bỏ thể hiện oval3 có trước ở độ sâu là 3 để tạo ra bản sao của oval1 ở độ sâu đó. Nếu không tin, bạn thử vô hiệu hóa câu lệnh oval3.removeMovieClip(); (ghi dấu // ở đầu câu lệnh) và chạy lại chương trình để thấy kết quả vẫn như hình 1.

Với thể hiện được tạo ra vào lúc chạy, bạn còn có thể làm cho nó thay hình đổi dạng. Bạn viết thêm hai câu lệnh như sau:

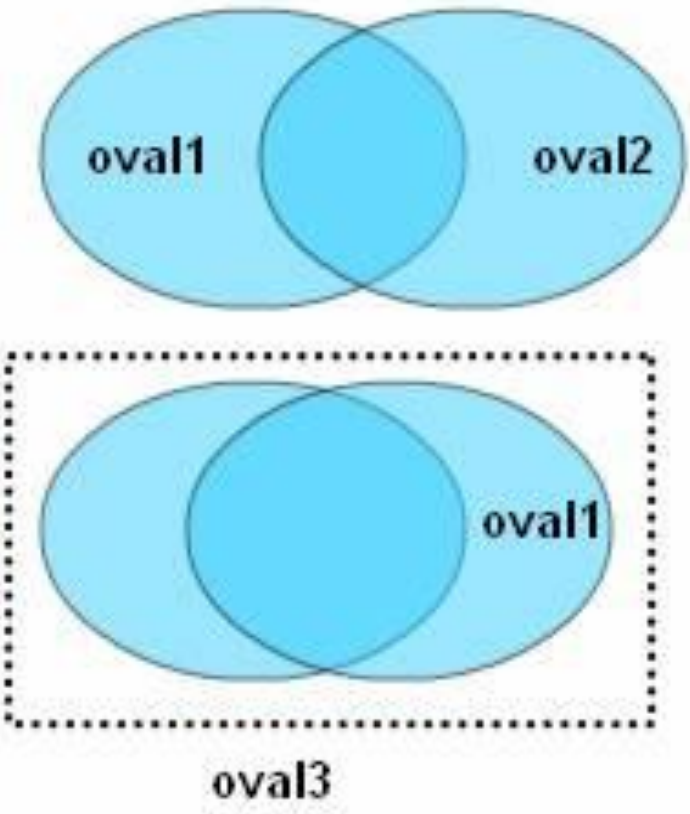
?

```
1  ...
2
3  //oval3.removeMovieClip();
4
5  oval1.duplicateMovieClip("oval3", 3);
6
7  oval3._y += 100;
8
9  oval3.attachMovie("Oval", "oval1", 1);
10
11 oval3.oval1._x += 40;
```

;

Câu lệnh oval3.attachMovie(“Oval”, “oval1″, 1); nhằm tạo ra thể hiện mang tên oval1 của nhân vật Oval bên trong thể hiện oval3 với độ sâu là 1. Bạn chú ý, độ sâu 1 của thể hiện oval1 vừa thêm là độ sâu bên trong oval3.

Câu lệnh tiếp theo oval3.oval1._x += 40; dịch chuyển thể hiện vừa thêm qua phải cho dễ phân biệt. Thử chạy chương trình, bạn thấy kết quả như hình 2.



Thể hiện oval3 giờ đây trở thành một thể hiện phức hợp bao gồm hai hình ô-van, trong đó hình ô-van vừa thêm có tên riêng là oval1. Nói rõ hơn, oval3 không còn là thể hiện của nhân vật Oval, mà là một phức hợp chứa đựng hai thể hiện của nhân vật Oval.

Thử viết thêm câu lệnh để co chiều rộng của oval3 theo phương ngang (chỉ còn 50%):

?

```
1 ...
2
3 oval3.attachMovie("Oval", "oval1", 1);
4
5 oval3.oval1._x += 40;
```

```
6

7 oval3._xscale = 50;

;
```

bạn thấy cả hai hình ô-van của oval3 đều bị ép như nhau.

Chắc bạn thắc mắc… kinh khủng: “Sao lại dùng tên oval1? Tên oval1 dùng cho thể hiện đầu tiên rồi mà!”. Bạn yên tâm, thể hiện oval1 vừa tạo ra nằm bên trong oval3. Nhìn từ bên ngoài oval3, thể hiện vừa thêm có tên là oval3.oval1. Tên oval1 được dùng trong hai phạm vi (scope) khác nhau nên ở đây cũng không có xung đột trong việc đặt tên.

Bài 22 : Nhân vật trống rỗng – Tự học lập trình Flash

Bạn hãy mở lại tập tin FLA chứa đoạn mã thử nghiệm việc tạo các thể hiện vào lúc chạy chương trình. Ta đã viết “nhăng nhít” ở khung 1 của thời tuyến chính, do đó bạn cần bấm vào khung 1 trong bảng Timeline, gõ F9 để mở bảng Actions – Frame.

Các câu lệnh đã viết tạo ra các thể hiện khác nhau của nhân vật Oval. Bạn đã tạo ra nhân vật Oval từ trước bằng cách vẽ hình ô-van và chuyển hình đó thành nhân vật.

Bạn có thể tạo ra bản thân nhân vật vào lúc chạy, không cần tạo trước nhân vật như đã làm. Đó là loại nhân vật trống rỗng (empty movie clip) chưa có hình thù gì cả vào lúc được tạo ra. Sau khi tạo ra nhân vật như vậy, bạn có thể dùng các câu lệnh thích hợp để vẽ vời “chân dung” cho nó. Bạn cũng có thể đưa vào nhân vật trống rỗng các thể hiện của nhân vật khác, tạo thành nhân

vật phức hợp. Trong thực tế, người ta thường tạo ra nhân vật trống rỗng và đưa vào đó các hình ảnh được tải xuống từ máy chủ (khi việc tải xuống hoàn tất). Điều này được thực hiện vào lúc chương trình đang chạy, không đòi hỏi các hình ảnh được tải xuống ngay khi khởi động, nhờ vậy rút ngắn đáng kể thời gian khởi động của chương trình.

Bạn ấn Ctrl+A và gõ phím Delete để xóa mọi câu lệnh đã viết. Bạn viết đoạn mã mới như sau để thử tạo nhân vật trống rỗng:

?

```
createEmptyMovieClip("myOvals", 1);

myOvals._x = 100;

myOvals._y = 100;

for(i = 1; i < 6; i++) {

myOvals.attachMovie("Oval", "oval" + i, i);

}
```

Câu lệnh createEmptyMovieClip(“myOvals”, 1); tạo ra một thể hiện của nhân vật trống rỗng ở thời tuyến chính, có tên là myOvals và được đặt ở độ sâu là

1. Hai câu lệnh tiếp theo quy định hoành độ _x và tung độ _y của thể hiện đó (dù nó chưa "mặt mũi"). Vòng lặp for có nhiệm vụ lặp lại năm lần việc tạo thể hiện của nhân vật Oval bên trong myOvals. Nhớ lại cách dùng hàm attachMovie(), bạn hiểu ngay: các thể hiện của nhân vật Oval có tên là oval1,…, oval5 và được đặt ở các độ sâu từ 1 đến 5 (các trị khả dĩ của biến i) bên trong myOvals..

Để phân biệt dễ dàng các thể hiện oval1,…, oval 5 bên trong myOvals, ta cần làm cho chúng có vị trí khác nhau. Bạn viết thêm như sau bên dưới vòng lặp for:

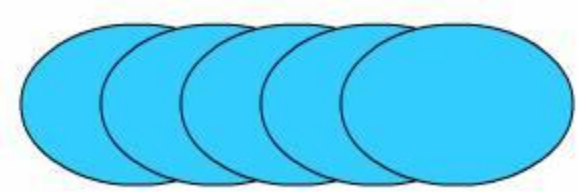
?

```
1 myOvals.oval1._x = 40;
2
3 myOvals.oval2._x = 80;
4
5 myOvals.oval3._x = 120;
6
7 myOvals.oval4._x = 160;
8
9 myOvals.oval5._x = 200;
```

Năm câu lệnh vừa nêu xê dịch các thể hiện qua phải. Các thể hiện có hoành độ _x khác nhau. Bạn chú ý, biểu thức myOvals.oval1._x dùng để diễn tả hoành độ _x của thể hiện oval1 bên trong thể hiện myOvals. Nếu bạn viết đơn

giản oval1._x, Flash sẽ không thực hiện được yêu cầu của bạn vì ở thời tuyến chính không có thể hiện nào tên là oval1.

Thử chạy chương trình, bạn thấy kết quả như hình 1.



Thay vì dùng đến năm câu lệnh để thay đổi hoành độ _x của năm thể hiện oval1,…,oval5, bạn viết lại theo cách khác, lịch lãm hơn, chỉ dùng một câu lệnh trong vòng lặp for:

?

```
...
for(i = 1; i < 6; i++) {
myOvals.attachMovie("Oval", "oval" + i, i);
eval("myOvals.oval" + i)._x += i*40;
}
/*
```

```
11
12 myOvals.oval1._x = 40;
13
14 myOvals.oval2._x = 80;
15
16 myOvals.oval3._x = 120;
17
18 myOvals.oval4._x = 160;
19
20 myOvals.oval5._x = 200;
21
22 */
23
```

Câu lệnh vừa thêm trong vòng lặp dùng hàm eval(), một hàm mà bạn chưa từng biết. Hàm eval() chuyển đổi một chuỗi thành tham chiếu (reference). Chẳng hạn, khi trị số của i là 1, câu lệnh vừa thêm tương đương với câu lệnh eval("myOvals.oval" + 1)._x += 40; hoặc myOvals.oval1._x += 40;. Nhờ có hàm eval(), Flash hiểu được rằng chuỗi "myOvals.oval" + 1 dùng để trỏ đến thể hiện oval1 bên trong thể hiện myOvals: myOvals.oval1.
Nhờ hàm eval(), bạn có thể tạo ra biến mới với tên tùy ý vào lúc chạy. Điều này là một thế mạnh của Flash, giúp bạn đưa ra những quyết định linh hoạt

vào lúc chạy chương trình, chứ không phải vào lúc viết chương trình.
Bạn hãy viết tiếp một câu lệnh nữa ở cuối chương trình để thay đổi thuộc tính _alpha của thể hiện myOvals:
?

```
1 myOvals._alpha = 50;
```

Chạy lại chương trình, bạn thấy kết quả như hình 2. Việc thay đổi thuộc tính _alpha của myOvals tác động lên cả năm thể hiện oval1,…,oval5 của nhân vật Oval bên trong myOvals, làm cho chúng đều trở nên nửa trong suốt (thuộc tính _alpha của các thể hiện oval1,…,oval5 đều trở thành 50). Rõ ràng, việc gộp các thể hiện riêng lẻ vào một thể hiện duy nhất giúp bạn thay đổi đồng loạt thuộc tính của chúng

Bài 23 : Gán hành vi vào thể hiện – Tự học lập trình Flash

Bạn đã biết rằng hành vi của một thể hiện bao gồm các hàm xử lý tình huống như onClipEvent(mouseDown), onClipEvent(load), onClipEvent(enterFrame),… Thông thường, bạn kéo nhân vật từ bảng Library vào sân khấu để tạo ra thể hiện của nhân vật. Bạn quy định hành vi của thể hiện bằng cách bấm chọn thể hiện và viết các hàm xử lý tình huống cho thể hiện được chọn trong bảng Actions – Movie Clip.
Đối với thể hiện được tạo ra vào lúc chạy chương trình, cách quy định hành vi của thể hiện có khác. Trước khi chạy, thể hiện không có trên sân khấu để bạn bấm chọn. Do vậy, bạn chỉ có thể gán hàm xử lý tình huống vào thể hiện trong lúc chạy.

Bạn hãy mở lại tập tin FLA dùng để thử nghiệm việc tạo các thể hiện vào lúc chạy, mở bảng Actions – Frame, xóa hết đoạn mã hiện có ứng với khung 1 và viết đoạn mã thử nghiệm mới như sau:

?

```
1  attachMovie("Oval", "oval1", 1);
2
3  oval1._x = 200;
4
5  oval1._y = 100;
6
7  oval1.onMouseDown = function() {
8
9  this._xscale += 5;
10
11 }
```

Ba câu lệnh đầu tiên trong đoạn mã vừa viết rất quen thuộc với bạn, chúng dùng để tạo ra thể hiện oval1 từ nhân vật Oval và quy định tọa độ cho thể hiện đó. Câu lệnh cuối cùng quy định hàm xử lý tình huống onMouseDown của thể hiện oval1. Đó là hàm được gọi khi người dùng nhấn phím chuột. Như bạn thấy, “quy định hàm xử lý tình huống onMouseDown của thể hiện oval1″ ở đây nghĩa là viết ra một định nghĩa hàm cụ thể và gán định nghĩa dó cho hàm

onMouseDown của oval1. Trước phép gán như vậy, hàm onMouseDown của oval1 chưa có nội dung gì cả. Sau phép gán, hàm onMouseDown của oval1 có nhiệm vụ cộng thêm 5 cho thuộc tính _xscale của oval1.

Thử chạy chương trình và bấm chuột nhiều lần vào đâu đó trong sân khấu, bạn thấy hình ô-van của ta phình ra theo phương ngang do thuộc tính _xscale của nó tăng dần mỗi lần bấm chuột.

Bạn để ý, ta đã viết this._xscale để chỉ rõ thuộc tính _xscale của thể hiện đang xét. Nếu bạn chỉ viết _xscale, Flash sẽ hiểu đó là thuộc tính _xscale của sân khấu vì ta đang viết chương trình cho khung 1 của thời tuyến chính. Khi đó, mỗi lần bấm chuột, sân khấu (cùng nội dung của nó) phình ra từ điểm mốc là góc trên, bên trái (gốc tọa độ của sân khấu).

Bạn hãy viết thêm vài câu lệnh như sau:

?

```
1 attachMovie("Oval", "oval2", 2);
2
3 oval2._x = 200;
4
5 oval2._y = 150;
6
7 oval2.onMouseDown = oval1.onMouseDown;
```

Đoạn mã vừa thêm tạo ra một thể hiện nữa của nhân vật Oval, đặt tên là oval2, nằm trên oval1 (độ cao là 2). Câu lệnh cuối gán hàm onMouseDown của oval1 cho hàm onMouseDown của oval2. Chạy lại chương trình và bấm

chuột nhiều lần, bạn thấy cả hai hình ô-van đều phình ra như nhau. Điều này nghĩa là khi muốn hai thể hiện có hành vi như nhau, bạn không cần lặp lại định nghĩa hàm xử lý tình huống cho thể hiện thứ hai, chỉ cần cho hai thể hiện dùng chung một hàm xử lý tình huống.

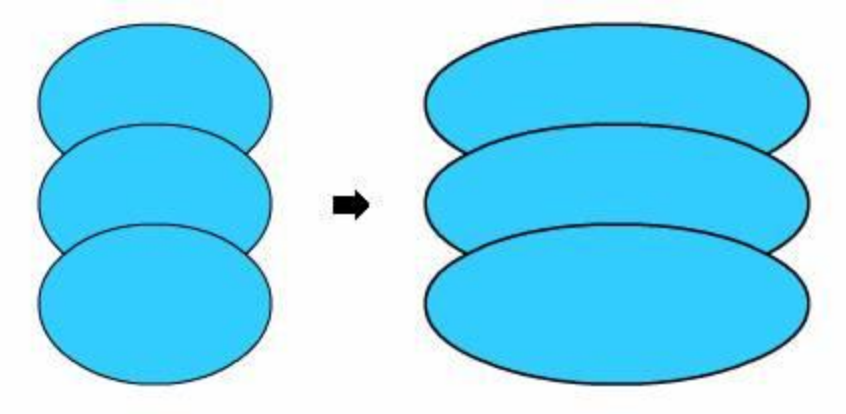
Chú ý rằng khi sao chép một thể hiện, bạn thu được thể hiện mới với các thuộc tính giống hệt bản gốc nhưng hành vi của thể hiện không được sao chép. Bạn có thể kiểm tra ngay điều đó bằng cách viết thêm:

?

```
1 oval1.duplicateMovieClip("oval3", 3);
2
3 oval3._y += 100;
4
5 oval3.onMouseDown = oval1.onMouseDown;
```

trong đó, ta gọi hàm duplicateMovieClip của oval1 để tạo ra một bản sao của oval1, lấy tên là oval3, có độ cao là 3 (nghĩa là nằm trên oval2). Câu lệnh cuối cho phép oval3 dùng chung hàm onMouseDown với oval1.

Thử chạy chương trình, bạn thấy cả ba hình ô-van (oval1, oval2, oval3) đều phình ra như nhau khi bấm chuột (hình 1). Trở lại với đoạn mã đang xét, bạn ghi thêm dấu // trước câu lệnh cuối (để vô hiệu hóa câu lệnh đó). Chạy lại lần nữa và bấm chuột nhiều lần, bạn thấy oval3 không hề thay đổi (hình 2). Điều này chứng tỏ việc sao chép một thể hiện không bao gồm hành vi của thể hiện.



Trong bảng Actions – Frame, bạn hãy bấm nút Replace để mở hộp thoại Replace. Trong ô Find what, bạn gõ onMouseDown. Trong ô Replace with, bạn gõ onMouseUp (hình 3). Bấm Replace All, bạn thấy tất cả tên hàm onMouseDown trong chương trình của ta được thay thế bằng onMouseUp. Thử chạy chương trình, bạn thấy các hình ô-van chưa phình ra khi nhấn phím chuột, chỉ phình ra khi bạn buông phím chuột.

Theo cách tương tự, bạn thử lần lượt khảo sát hai hàm xử lý tình huống onPress và onRelease. Hai hàm này tương tự hai hàm onMouseDown và onMouseUp nhưng có điểm khác biệt quan trọng: chỉ được gọi khi bạn bấm trúng thể hiện. Hai hàm onPress và onRelease rất cần thiết cho bạn sau này.

Bài 24 : Tạo nút bấm – Tự học lập trình Flash

Khi thử gán hàm xử lý tình huống onPress cho một thể hiện được tạo ra lúc chạy chương trình, có lẽ bạn thấy hàm onPress thú vị hơn hàm onMouseDown.

Hàm onPress chỉ được gọi khi người dùng bấm trúng thể hiện. Chắc rằng điều này khiến bạn nghĩ ngay đến việc tạo ra một nút bấm trong chương trình của mình. Quả thực, ta có thể vẽ hình bất kỳ để làm nút bấm. Muốn chương trình làm gì đó khi người dùng bấm nút, chỉ cần định nghĩa hàm onPress thích hợp. Bạn hãy mở tập tin FLA mới trong cửa sổ Flash để thực hiện ý định nêu trên. Trước hết, cần tạo một nhân vật đóng vai nút bấm. Bạn ấn Ctrl+F8 (hoặc chọn Insert > New Symbol). Khi hộp thoại Create New Symbol hiện ra, bạn gõ Button để đặt tên cho nhân vật mới. Bấm nút Advanced, bạn thấy hộp thoại được mở rộng, bày ra những quy định bổ sung (hình 1). Trong phần Linkage, bạn bật ô duyệt Export for ActionScript. Chắc bạn còn nhớ, nhờ quy định này, Flash sẽ đưa nhân vật đang xét vào tập tin SWF dù cho nhân vật không hiện diện trên sân khấu lúc biên dịch.

Tuy rằng ta định tạo ra nút bấm, bạn đừng chọn Button ở phần Behavior vì lựa chọn đó nhằm tạo ra nút bấm theo cách cũ, tương thích với phiên bản trước của Flash. Bạn cứ giữ nguyên lựa chọn Movie clip.

Bấm OK để đóng hộp thoại Create New Symbol, bạn bắt đầu việc vẽ hình để làm nút bấm. Bạn để ý dấu thập nhỏ trên nền trắng trơn, đó là điểm mốc của nhân vật. Bạn có thể xem đó là góc trên, bên trái của nút bấm. Thử hình dung ta cần làm một “nút khởi động” gì đó màu xanh lá, bạn bấm vào ô Fill Color ở hộp công cụ và chọn màu xanh lá trên bảng màu . Bạn gõ phím R (hoặc chọn Rectangle Tool), trỏ vào dấu thập, giữ phím trái của chuột, căng ra một hình khung. Hình khung vừa vẽ tự động được tô màu xanh đã chọn.

Ta hãy ghi chữ Start màu đen lên hình khung xanh lá. Bạn bấm vào Fill Color, chọn màu đen, Để ghi chữ, bạn gõ phím T (hoặc chọn Text Tool), trỏ vào đâu đó gần hình khung xanh lá, căng ra một khung chữ và gõ Start. Bạn ấn Ctrl+F3 để mở bảng Properties. Trong bảng Properties, bạn chọn Static text (khung chữ tĩnh) trong ô Text Type , điều chỉnh cỡ chữ ở ô Font Size .

Bạn bấm vào đâu đó bên ngoài khung chữ để tỏ ý kết thúc việc ghi chữ, gõ phím V (hoặc chọn Selection Tool) để lấy công cụ chọn, kéo dòng chữ Start,

đặt lên hình khung xanh lá sao cho cân xứng. Bạn có thể gõ các phím mũi tên nếu cần tinh chỉnh vị trí của dòng chữ Start.

Bạn bấm vào Scene1 để kết thúc việc tạo hình cho nhân vật Button. Muốn biết chắc nhân vật Button đã được tạo ra trong thư viện hay chưa, bạn gõ phím F11 để mở xem bảng Library.

Bạn gõ phím F9 để mở bảng Actions – Frame và viết chương trình (cho khung 1) như sau:

?

```
1  attachMovie("Button", "button1", 1);
2
3  button1._x = 200;
4
5  button1._y = 200;
6
7  button1.onPress = function() {
8
9  trace("Bạn đã bấm nút.");
10
11 }
```

Đoạn mã nêu trên nhằm tạo ra thể hiện button1 của nhân vật Button, quy định vị trí của button1 và định nghĩa hàm onPress của button1. Hàm onPress chỉ làm một việc đơn giản: hiển thị thông báo trong bảng Output.

Ấn Ctrl+Enter để chạy thử, bạn thấy nút bấm của ta hoạt động đúng như dự định: mỗi lần bấm nút đó, thông báo Bạn đã bấm nút xuất hiện trong bảng Output.

Có lẽ bạn từng thấy các loại nút bấm thay đổi hình dạng hoặc màu sắc khi

được trỏ vào hoặc khi được bấm.

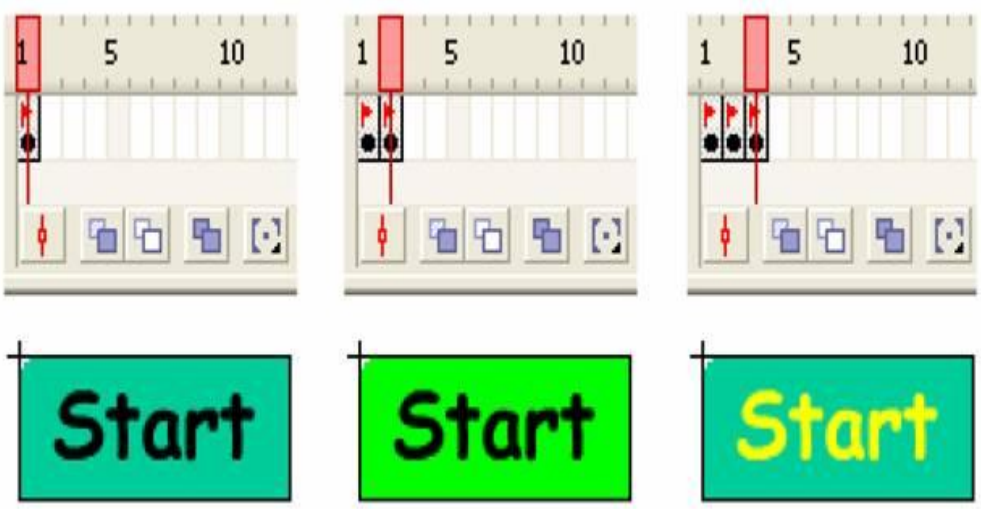
Ta cũng làm được điều đó. Bạn hãy bấm kép vào nút bấm trong bảng Library để bắt đầu chỉnh sửa.

Nếu nút bấm đang ở tình trạng “được chọn”, bạn bấm bên ngoài nút để “thôi chọn”. Bạn bấm vào khung 1 trong thời tuyến của nút bấm. Trong bảng Actions – Frame, bạn ghi câu lệnh stop(); cho khung 1. Xong, bạn bấm vào ô có dòng chữ <Frame Label> trong bảng Properties và gõ _up để đặt tên cho khung 1.

Bạn bấm-phải vào khung 2 ở bảng Timeline, chọn Insert Keyframe nhằm tạo ra khung chốt thứ hai. Bạn ghi câu lệnh stop(); và đặt tên _over cho khung 2. Bạn bấm vào hình khung xanh lá, chọn màu xanh sáng hơn (hình 2) ở ô Fill Color .

Tương tự, bạn tạo thêm khung chốt thứ ba, ghi câu lệnh stop(); và đặt tên cho khung 3 là _down. Bạn chọn màu xanh lá đậm cho hình khung và chọn màu vàng cho dòng chữ Start (hình 2).

Ấn Ctrl+Enter để chạy thử, bạn thấy nút Start sáng lên khi được trỏ vào. Bấm nút Start, bạn thấy nó “chớp” một phát, trông thật… điệu nghệ.



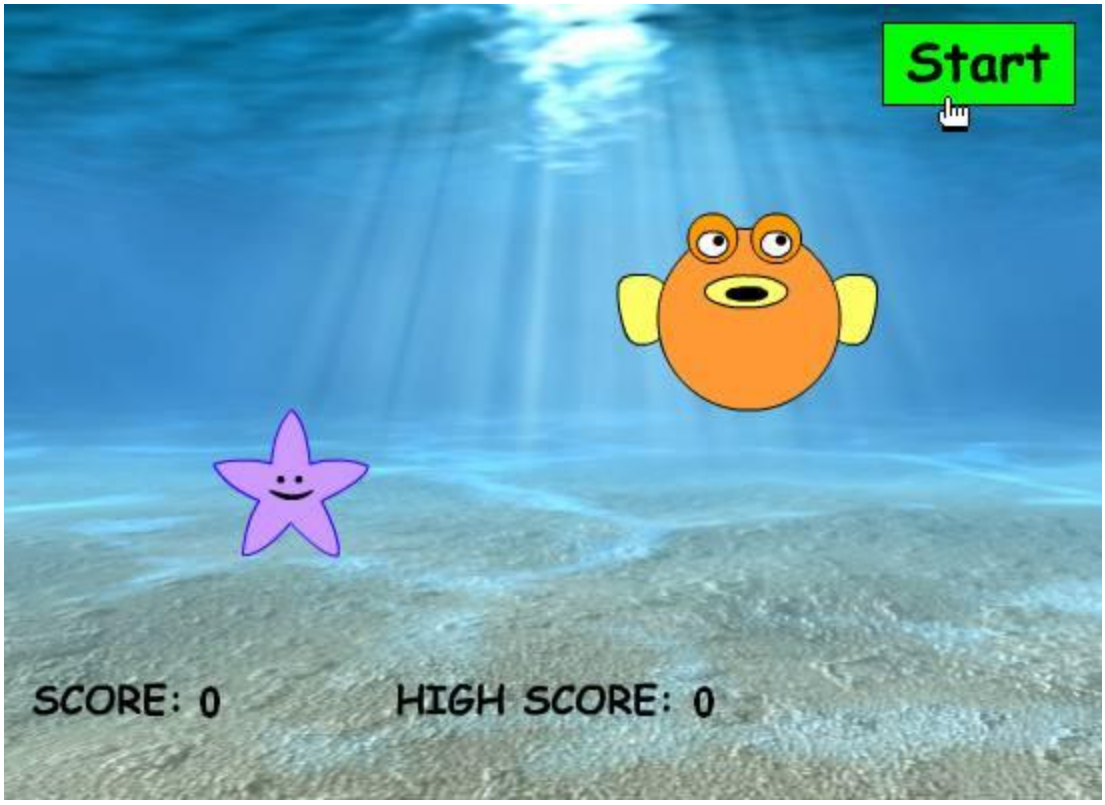
Bài 25 : Nút bấm khởi động – Tự học lập trình Flash

Bạn đã thấy cách thức tạo nút bấm rất đơn giản: chỉ cần định nghĩa hàm xử lý tình huống onPress của nhân vật được dùng làm nút bấm. Muốn cho nút bấm có vẻ "chuyên nghiệp" hơn, bạn đã tạo ra ba khung chốt liên tiếp trong thời tuyến của nút bấm, đặt tên là _up, _over, _down.

Đó là các tên bắt buộc theo quy ước của Flash, bạn đừng đặt tên khác đi (chú ý đừng viết thiếu dấu gạch chân _). Quan sát thời tuyến của nút bấm, bạn thấy các khung đã được đặt tên đều có gắn cờ hiệu màu đỏ để phân biệt với khung không có tên.

Hình ảnh ở khung _up chính là diện mạo của nút bấm ở trạng thái bình thường. Khi bạn trỏ vào nút bấm, Flash hiển thị hình ảnh ở khung _over của nút bấm. Khi bạn bấm nút, Flash chuyển qua khung _down của nút bấm. Ở mỗi khung _up, _over và _down, ta đều phải viết câu lệnh stop(); để ra lệnh ngừng, ngăn không cho hiển thị khung tiếp theo. Nếu không làm như vậy, các khung _up, _over và _down sẽ tự động được hiển thị liên tiếp, lặp đi lặp lại vào lúc chạy, khiến bạn thấy nút bấm nhấp nháy liên tục.

Ta hãy dùng nút bấm Start hiện có làm nút khởi động trong trò chơi "bắt sao biển" trước đây. Nhờ nút khởi động Start, người chơi có thể chủ động bắt đầu trò chơi khi họ sẵn sàng (hình 1). Khi người chơi bị cá nóc bắt được, trò chơi cần kết thúc với trạng thái tĩnh. Muốn chơi tiếp, người chơi lại bấm nút Start. Nhờ vậy, người chơi kịp "hoàn hồn" để xem điểm số mà mình đạt được.



Nếu đang mở tập tin chứa nút bấm Start, bạn hãy đóng tập tin đó. Bạn mở lại tập tin FLA của trò chơi "bắt sao biển" và ấn Ctrl+Shift+O (hoặc chọn File > Import > Open External Library). Trong hộp thoại Open as Library vừa hiện ra, bạn tìm tập tin FLA chứa nút bấm Start và bấm kép vào nó. Thao tác như vậy mở ra cửa sổ Library của tập tin đã chọn, trong đó có nút bấm Start cần dùng. Trong thời tuyến chính, bạn chọn khung 1 của lớp Layer 1 và mở bảng Actions – Frame (gõ phím F9). Bạn viết thêm như sau vào bên dưới đoạn mã hiện có:

?

```
1  play = false;
2
3  attachMovie("Button", "button1", 1);
4
```

```
5  button1._x = 440;
6
7  button1._y = 10;
8
9  button1.onPress = function() {
10
11 play = true;
12
13 star.score = 0;
14
15 }
```

Để tạo ra nút bấm Start (nhân vật Button) trên sân khấu, bạn dùng hàm attachMovie đã biết và quy định tọa độ thích hợp cho nút bấm. Biến play đóng vai trò “cờ hiệu”, quy định trạng thái của trò chơi. Lúc đầu, ta gán trị false cho biến play. Trong hàm onPress của nút Start (hàm được gọi khi bấm nút Start), ta gán trị true cho biến play và cho điểm số trở lại trị 0. Điểm số được lưu giữ bởi biến star.score, tức biến score bên trong sao biển có tên là star.
Tiếp theo, bạn cần chỉnh sửa hành vi của sao biển và cá nóc sao cho chúng “án binh bất động” khi biến play có trị là false. Bạn bấm vào đâu đó trên sân khấu, rồi bấm vào sao biển, quan sát chương trình của nó trong bảng Actions – Movie Clip và viết thêm vào hàm onClipEvent(mouseDown) như sau:

?

```
...

onClipEvent(mouseDown) {

if(!_root.play)

return;

if(hitTest(_root._xmouse, _root._ymouse, true))
{

...
```

Khi biến play ở thời tuyến chính (biến _root.play) có trị là false, biểu thức !_root.play có trị là true, hàm onClipEvent(mouseDown) kết thúc ngay, không làm gì cả. Nghĩa là khi biến play có trị là false, sao biển sẽ không “nhúc nhích” chi hết dù bạn bấm trúng nó.

Bạn bấm vào cá nóc để chuyển qua chương trình của nó và chỉnh sửa trong hàm onClipEvent(enterFrame) như sau:

?

```
...

onClipEvent(enterFrame) {

if(!_root.play)

return;

caught = false;

if(_x + step < _root._xmouse)

...

else if(caught == true) {

//trace("Bắt được rồi nhé!");

```

```
19 //_root.star.score = 0;
20
21 _root.play = false;
22
23 step = 5;
24
25 }
26
27 }
```

Tương tự như trường hợp sao biển, câu lệnh điều kiện if vừa thêm nhằm kiểm tra biến play ở thời tuyến chính: nếu biến play có trị là false, hàm onClipEvent(enterFrame) kết thúc ngay. Khi biến play có trị là true, cá nóc mới rượt theo con trỏ chuột. Khi cá nóc bắt được con trỏ chuột, ta gán trị false cho biến play để trò chơi chuyển qua trạng thái tĩnh. Ngoài ra, câu lệnh _root.star.score = 0; được vô hiệu hóa để biến star.score giữ nguyên điểm số mà người chơi vừa đạt được, giúp họ đọc điểm dễ dàng hơn. Biến star.score chỉ trở lại trị 0 khi người chơi bấm nút Start để chơi lần nữa.
Bạn ấn Ctrl+Enter để chạy chương trình, xem thử những sửa đổi vừa thực hiện có hiệu lực ra sao nha.

Bài 26 : Thao tác trên dãy – Tự học lập trình Flash

Trong trò chơi "bắt sao biển" đã thực hiện, bạn chỉ có hai nhân vật (sao biển và cá nóc) và bạn gọi tên chúng (star và fish) khi cần "liên hệ công tác".

Thử hình dung trò chơi nào đó có nhiều thể hiện của nhiều nhân vật khác nhau, không ngừng sinh ra và mất đi, bạn thấy ngay rằng cách thức quản lý "thủ công" các thể hiện của trò chơi "bắt sao biển" sẽ không phù hợp. Trong trường hợp như vậy, số lượng thể hiện biến động trong lúc chơi, bạn không thể biết trước vào lúc lập trình.

Để chuẩn bị cho việc lập trình các trò chơi phức tạp hơn, bạn cần biết đến khái niệm dãy (array). Nhờ đưa các thể hiện vào một dãy, bạn không cần đặt tên riêng cho từng thể hiện. Lúc đó, các thể hiện "xếp hàng" trong bộ nhớ máy tính, mỗi thể hiện được xác định bởi tên của dãy kèm theo một chỉ số (index), cho biết vị trí của nó trong dãy. Xin nói ngay, xếp hàng trong bộ nhớ không có nghĩa là xếp hàng trên sân khấu. Bạn đừng nhầm nha.

Bạn hãy mở tập tin FLA mới, gõ phím F9 để mở bảng Actions – Frame ứng với khung 1 và gõ câu lệnh tạo ra một dãy như sau:

?

```
1 arr = new Array();
```

Với câu lệnh như vậy, ta tạo ra một dãy có tên là arr. Người ta cũng nói rằng arr là một biến trỏ đến một dãy. Nói cho… phức tạp, câu lệnh vừa nêu tạo ra một đối tượng (object) mới thuộc lớp (class) Array và arr là biến trỏ đến đối tượng mới đó.

Bạn ghi tiếp hai câu lệnh như sau để đặt trị số 0 vào vị trí thứ nhất của dãy và hiển thị phần tử thứ nhất đó ở bảng Output:

?

```
1 arr = new Array();
2
3 arr[0] = 0;
4
5 trace(arr[0]);
```

Bạn để ý, khi gọi tên một phần tử của dãy, ta ghi tên dãy, kèm theo chỉ số của phần tử đặt trong cặp dấu ngoặc vuông. Chỉ số của phần tử đầu tiên của dãy là 0, chứ không phải 1. Thử chạy chương trình, bạn thấy phần tử thứ nhất của dãy arr (trị số 0) xuất hiện trong bảng Output.
Bạn viết thêm một câu lệnh nữa để đặt trị số 10 vào vị trí thứ hai của dãy (ứng với chỉ số 1):
?

```
1 arr = new Array();
2
3 arr[0] = 0;
4
5 arr[1] = 10;
6
7 trace(arr);
```

Ngoài ra, bạn sửa câu lệnh trace(arr[0]); thành trace(arr); để in ra bảng Output toàn bộ dãy arr, thay vì chỉ in phần tử đầu tiên. Với câu lệnh trace(arr), Flash

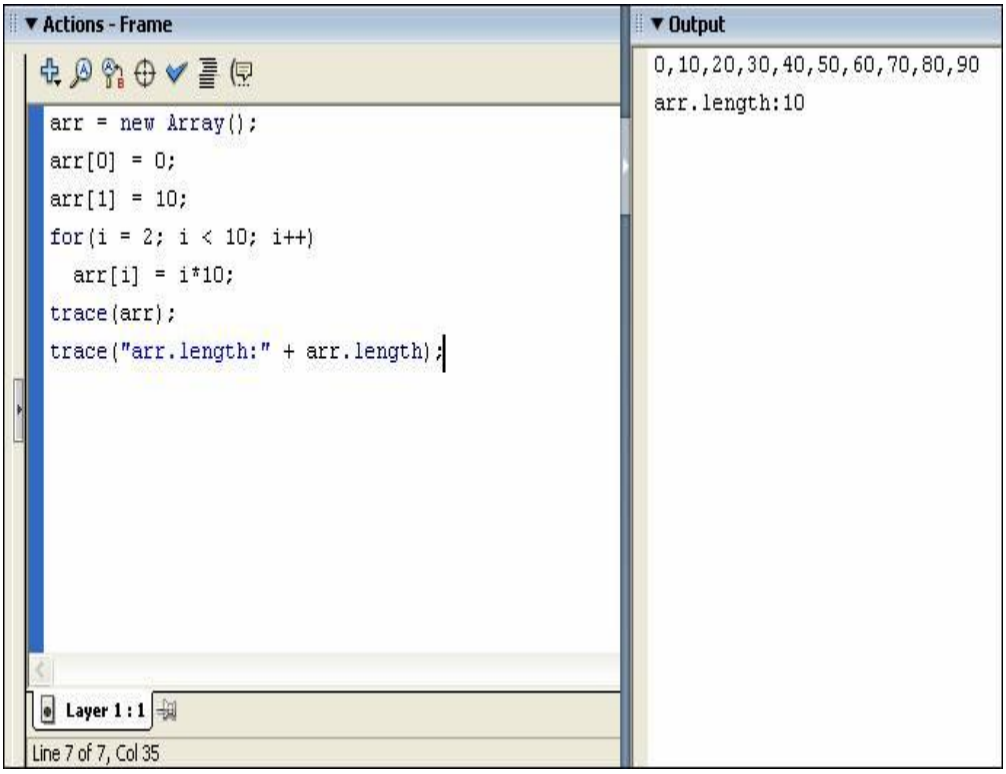
tự biết rằng phải ghi tuần tự từng phần tử trong dãy và phân cách bằng dấu phẩy. Trong trường hợp đang xét, dãy chỉ có hai phần tử, bạn thấy kết quả ở bảng Output như sau: 0,10.

Giả sử bạn cần đặt trị số 20 vào vị trí thứ ba, trị số 30 vào vị trí thứ tư,… Bạn dùng vòng lặp như sau cho nhanh:

?

```
1  arr = new Array();
2
3  arr[0] = 0;
4
5  arr[1] = 10;
6
7  for(i = 2; i < 10; i++)
8
9  arr[i] = i*10;
10
11 trace(arr);
12
13 trace("arr.length:" + arr.length);
```

Câu lệnh cuối trong đoạn mã nêu trên nhằm in ra bảng Output chiều dài của dãy, tức số vị trí có trong dãy. Chiều dài của dãy được cho bởi biến length trong đối tượng arr, viết là arr.length. Biến length được tạo ra trong đối tượng arr và được cập nhật một cách tự động. Biến length là thuộc tính (property) của dãy, chứ không phải phần tử của dãy. Mỗi lần bạn thêm phần tử vào dãy, dãy dài ra thêm và trị số của biến thuộc tính length của dãy tự động tăng lên.



Thử chạy chương trình, bạn có kết quả ở bảng Output như hình 1. Chiều dài của dãy là 10, được chiếm bởi mười trị số (hình 2). Mười vị trí của dãy ứng với các chỉ số từ 0 đến 9.

0	1	2	3	4	5	6	7	8	9
0	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**

Khi đặt các trị số vào dãy, bạn không nhất thiết phải theo thứ tự. Bạn thử ghi thêm hai câu lệnh như sau:

?

```
1  arr = new Array();
2
3  arr[0] = 0;
4
5  arr[1] = 10;
6
7  for(i = 2; i < 10; i++)
8
9  arr[i] = i*10;
10
11 arr[14] = 200;
12
13 arr[12] = 100;
14
15 trace(arr);
16
17 trace("arr.length:" + arr.length);
```

Hai câu lệnh vừa thêm “ngang nhiên” ghi trị số 200 vào vị trí ứng với chỉ số 14, “bất chợt” ghi trị số 100 vào vị trí ứng với chỉ số 12. Thử chạy chương trình, bạn thấy kết quả như sau:

?

```
0,10,20,30,40,50,60,70,80,90,undefined,undefined,100,unde
fined,200

arr.length:15
```

Bạn thấy rõ dãy arr hiện có 15 vị trí, tăng lên so với trước để đáp ứng nhu cầu được diễn đạt bởi hai câu lệnh mới. Khi trình bày nội dung của dãy, Flash tự động ghi undefined (chưa được xác định) ở các vị trí còn bỏ trống. Trong trường hợp đang xét, các vị trí bỏ trống ứng với các chỉ số 10, 11, 13.
Nhờ hai câu lệnh “ngẫu hứng”, bạn thấy rõ chiều dài của dãy là số vị trí có trong dãy, chứ không phải số phần tử nằm trong dãy.

Bài 27 : Hành vi của dãy – Tự học lập trình Flash

Bạn đã làm quen với dãy và biết cách đặt các trị số vào dãy, tạo thành các phần tử của dãy. Khi đặt một trị số vào dãy, bạn chỉ ra vị trí của nó trong dãy. Dãy tự động dài ra theo nhu cầu của bạn.
Thao tác trên dãy có lẽ khiến bạn hình dung rằng dãy chẳng qua chỉ là các “ô chứa” xếp thành hàng trong bộ nhớ máy tính. Điều đó đúng nhưng… chưa đủ. Dãy là một đối tượng có khả năng quản lý các “ô chứa”. Dãy có những hành vi (hàm) nhất định. Bạn có thể yêu cầu dãy thực hiện khá nhiều việc “ly kỳ”.

Bạn hãy mở lại tập tin FLA có đoạn mã thử nghiệm dãy ở khung 1. Bạn xóa đoạn mã hiện có, viết đoạn mã khác như sau:

?

```
1  arr = new Array();
2
3  for(i = 0; i < 5; i++) {
4
5  trace("Đưa vào: " + i);
6
7  arr.push(i);
8
9  trace(arr);
10
11 }
```

Đoạn mã trên tạo một dãy mang tên arr, rồi dùng một vòng lặp for để lần lượt "đẩy" các trị số từ 0 đến 4 vào dãy. Ta đẩy một trị số vào dãy bằng cách trao trị số đó cho hàm push của dãy. Nói khác đi, ta yêu cầu dãy "nhét" trị số mới vào trong "lòng". Trong trường hợp này, ta không quan tâm đến chỉ số của phần tử mới trong dãy, chỉ cần nhớ luật lệ xếp hàng đơn giản: phần tử vào trước thì đứng trước, phần tử vào sau cùng thì đứng sau cùng.
Trong mỗi lần lặp, câu lệnh trace(arr); in ra bảng Output toàn bộ nội dung của

dãy. Nhờ vậy, khi chạy thử chương trình, bạn có thể theo dõi diễn biến bên trong dãy, thấy được dãy từ từ dài ra như thế nào:

Đưa vào: 0

0

Đưa vào: 1

0,1

Đưa vào: 2

0,1,2

Đưa vào: 3

0,1,2,3

Đưa vào: 4

0,1,2,3,4

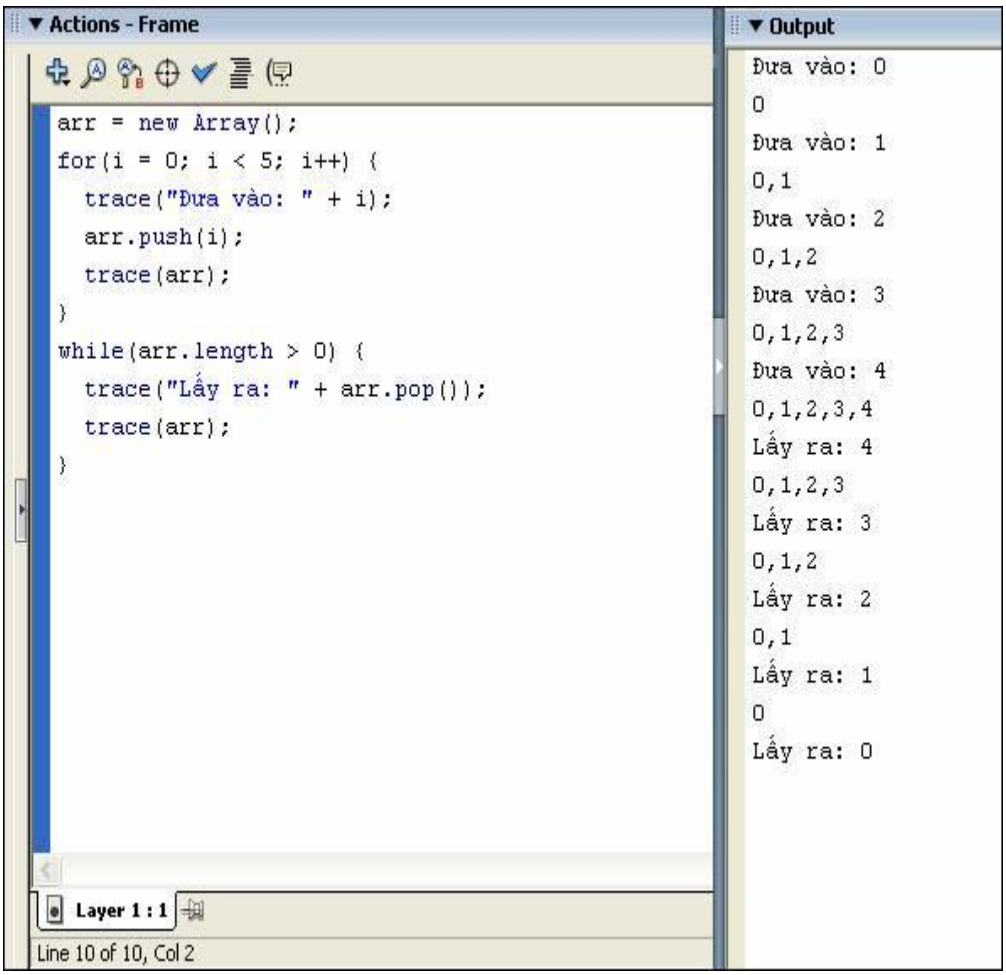
Dãy có một hàm khác gọi là pop, có thể xem là hàm ngược của push. Khi bạn gọi hàm pop của dãy, phần tử đứng sau cùng "bật" ra khỏi dãy cho bạn "hứng". Nghĩa là hàm pop trả về phần tử sau cùng của dãy, đồng thời loại nó ra khỏi dãy, làm cho chiều dài của dãy giảm đi một đơn vị. Bạn viết thêm đoạn mã như sau:

?

```
1 while(arr.length > 0) {
2
3 trace("Lấy ra: " + arr.pop());
4
5 trace(arr);
```

```
6
7  }
```

trong đó, ta dùng vòng lặp while để gọi hàm pop của dãy arr nhiều lần, chừng nào chiều dài của dãy vẫn còn lớn hơn 0. Trong mỗi lần lặp, ta cho in ra nội dung của dãy để thấy được dãy từ từ ngắn lại ra sao (hình 1).



Trong vòng lặp while vừa viết, ở câu lệnh trace(“Lấy ra: ” + arr.pop()); bạn hãy sửa arr.pop() thành arr.shift() và ấn Ctrl+Enter để chạy lại chương trình. Bạn sẽ thấy hàm shift khác với hàm pop ở chỗ: làm cho phần tử đầu tiên, chứ không phải phần tử sau cùng, bật ra khỏi dãy. Khi phần tử đầu tiên bị loại khỏi dãy, phần tử thứ hai tự động dịch lên, trở thành phần tử đầu tiên, phần tử thứ ba nối đuôi, trở thành phần tử thứ hai,… Kết quả là mọi phần tử trong dãy đều

xê dịch. Tên gọi shift (xê dịch) xuất phát từ đó.
Tiếp tục “chơi đùa” với dãy, bạn viết thêm vòng lặp for như sau:

?

```
1 for(i = 0; i < 5; i++) {
2
3 trace("Đưa vào: " + i);
4
5 arr.unshift(i);
6
7 trace(arr);
8
9 }
```

Nghĩa là sau khi lấy mọi phần tử ra khỏi dãy, ta lại đưa các trị số từ 0 đến 4 vào dãy, nhưng lần này ta gọi hàm unshift của dãy, chứ không gọi hàm push. Quan sát kết quả ở bảng Output, bạn hiểu ngay: hàm unshift luôn luôn đưa phần tử mới vào vị trí đầu tiên của dãy, chứ không phải vị trí sau cùng. Phần tử mới chen vào vị trí đầu tiên, đẩy các phần tử có sẵn ra sau. Bạn thu được dãy trị số ngược: 4,3,2,1,0.
Hàm unshift và hàm shift tạo ra nhiều xê dịch trong dãy nên tốn thời gian hơn hàm push và hàm pop. Tuy vậy, vẫn có những trường hợp mà việc thêm phần tử mới vào dãy ở ngay vị trí đầu tiên là nhu cầu thực sự.

Dãy còn có khả năng tự sắp thứ tự nữa. Bạn hãy viết thêm ba câu lệnh như sau:

?

```
1 trace("Sắp thứ tự:");

2

3 arr.sort();

4

5 trace(arr);
```

Để sắp thứ tự trong dãy, bạn chỉ cần gọi hàm sort của dãy. Trước khi gọi hàm sort, nội dung của dãy là 4,3,2,1,0. Sau khi gọi hàm sort, nội dung của dãy có thứ tự "nhỏ trước, lớn sau": 0,1,2,3,4.
Đối với dãy chứa các chuỗi, hàm sort của dãy giúp bạn có được thứ tự vần (alphabet order) trong dãy. Bạn thử hình dung: ta cho người dùng chương trình nhập tên các bài hát. Chỉ cần lưu những tên bài hát trong dãy và gọi hàm sort của dãy, bạn có ngay danh sách bài hát được sắp thứ tự theo vần. Bạn nên xóa hết đoạn mã hiện có và tự viết đoạn mã thử nghiệm khả năng sắp thứ tự các chuỗi theo vần của hàm sort của dãy.

Bài 28 : Vui đùa với dãy – Tự học lập trình Flash

Bạn đã biết đến khái niệm dãy (array). Ta hãy tiếp tục "quậy" với dãy thêm chút nữa, chuẩn bị cho việc thực hiện các trò chơi có dùng đến

dãy. Điều này cần thiết giống như bạn vui đùa với bóng trước khi đá bóng thực sự.

Bạn hãy mở tập tin mới trong Flash và viết đoạn mã như sau trong bảng Actions – Frame:

?

```
1 arr = ["mãng cầu", "dừa", "đu đủ", "xoài"];
2
3 trace(arr);
4
5 subarr = arr.slice(0,1); trace(subarr);
6
7 subarr = arr.slice(2,3); trace(subarr);
8
9 subarr = arr.slice(2); trace(subarr);
```

Đoạn mã vừa nêu giúp bạn biết một cách mới để tạo ra dãy với các phần tử sắp sẵn và làm quen với hàm slice của dãy. Phần tử đầu tiên là “mãng cầu”. Các phần tử kế tiếp là “dừa”, “đu đủ”, “xoài” (tức là… “cầu vừa đủ xài”). Sau khi tạo ra dãy arr, ta dùng câu lệnh trace(arr); để in dãy arr ra bảng Output, xem thử dãy arr có chứa các thứ mà ta đặt vào hay chưa.

Hàm slice giúp bạn cắt lấy một phần của dãy, tạo ra dãy mới. Dãy bị cắt thực ra vẫn còn nguyên, không bị mất tí tẹo nào. “Dãy con” thu được chứa đựng một số phần tử của “dãy gốc”. Nói rõ hơn, dãy con dùng chung một số phần

tử với dãy gốc.

Khi viết arr.slice(0, 1), bạn “xắn” vào dãy arr, “trước mặt” phần tử thứ nhất ở vị trí 0 và “trước mặt” phần tử thứ hai ở vị trí 1. Lát cắt được lấy ra chỉ chứa phần tử thứ nhất, tức “mãng cầu”. Ta đặt tên cho dãy con thu được từ hàm slice là subarr và dùng hàm trace kiểm tra ngay nội dung của dãy con subarr.

Khi viết arr.slice(2, 3), ta có lát cắt từ vị trí 2 (phần tử “đu đủ”) đến vị trí 3 (phần tử “xoài”). Dãy con thu được chỉ gồm phần tử “đu đủ”, không có phần tử “xoài”. Nói chung, bạn cần nhớ rằng phần tử ứng với đối mục thứ nhất của hàm slice có mặt trong kết quả của hàm slice nhưng phần tử ứng với đối mục thứ hai thì không.

Khi viết arr.slice(2), bạn gọi hàm slice nhưng chỉ cung cấp một đối mục. Flash tự hiểu rằng bạn muốn cắt từ vị trí 2 (phần tử “đu đủ”) đến hết dãy arr. Dãy con thu được gồm có “đu đủ” và “xoài”.

Chạy thử chương trình và nhìn vào bảng Output, bạn thấy rõ nội dung của dãy gốc và các dãy con do hàm slice tạo ra:

mãng cầu,dừa,đu đủ,xoài

mãng cầu

đu đủ

đu đủ,xoài

Bạn hãy viết thêm các câu lệnh gọi hàm slice “ly kỳ” hơn:

?

```
1 subarr = arr.slice(); trace(subarr);

2

3 subarr = arr.slice(-2); trace(subarr);
```

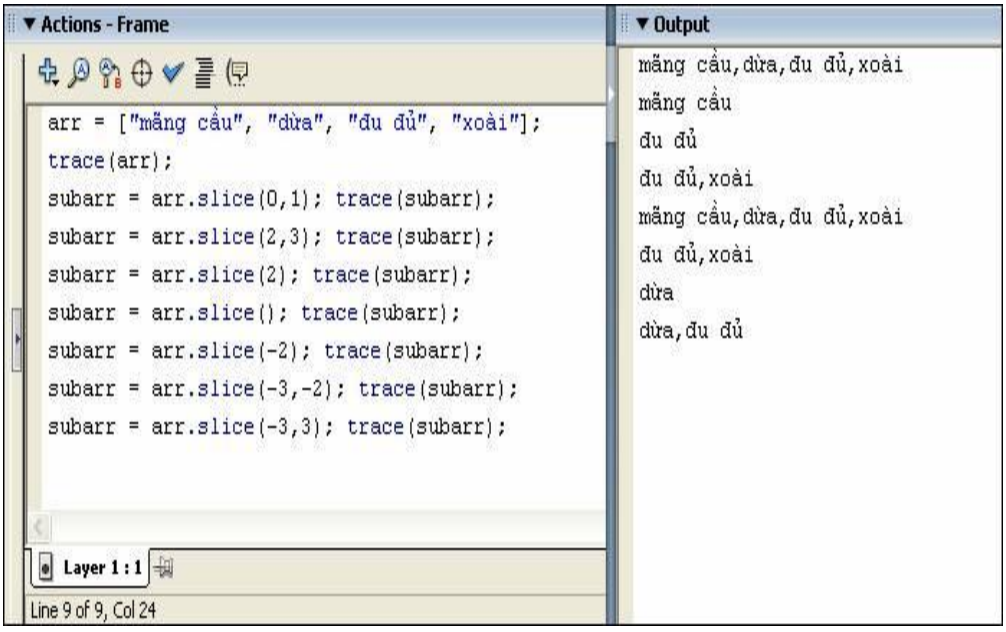
```
4
5 subarr = arr.slice(-3,-2); trace(subarr);
6
7 subarr = arr.slice(-3,3); trace(subarr);
```

Khi bạn gọi hàm slice của dãy arr mà không cung cấp đối mục nào, Flash hoan hỉ cho bạn toàn bộ dãy arr. Nếu đối mục của hàm slice là chỉ số âm, Flash không hề bối rối và tự hiểu rằng đó là chỉ số tính từ đuôi dãy, chứ không phải tính từ đầu dãy như bình thường. Vị trí cuối dãy có chỉ số là -1, vị trí áp cuối có chỉ số là -2,…

Khi viết arr.slice(-2), bạn thu được dãy con của arr, từ với vị trí áp cuối (“đu đủ”) đến hết dãy arr. Khi viết arr.slice(-3, -2), bạn thu được dãy con từ vị trí -3 (“dừa”) đến vị trí -2 (“đu đủ”). Dãy con như vậy chỉ có “dừa”, không có “đu đủ”.

Bạn có thể dùng đồng thời chỉ số âm và chỉ số dương khi gọi hàm slice. Khi bạn viết arr.slice(-3, 3), Flash dư sức hiểu rằng dãy con được lấy từ vị trí -3 (“dừa”) đến vị trí 3 (“xoài”).

Nhìn vào kết quả của chương trình (hình 1), bạn có thể kiểm tra xem Flash “suy nghĩ” có giống mình hay không.



Cần nhắc lại rằng hàm slice không làm “sứt mẻ” dãy arr. Ở cuối đoạn mã đã có, bạn có thể thử ghép dãy arr với dãy con subarr bằng một hàm có tên là concat và in ra kết quả: trace(arr.concat(subarr));. Nhờ hàm concat của arr, bạn thu được… “dãy gộc” dài hơn arr, bao gồm các phần tử của arr và subarr. Bạn thử ngay xem sao.

Nếu muốn “xắn” vào dãy arr và làm dãy arr mất đi lát cắt, bạn phải dùng hàm khác, gọi là splice. Bạn hãy xóa đoạn mã hiện có và viết đoạn mã mới như sau:

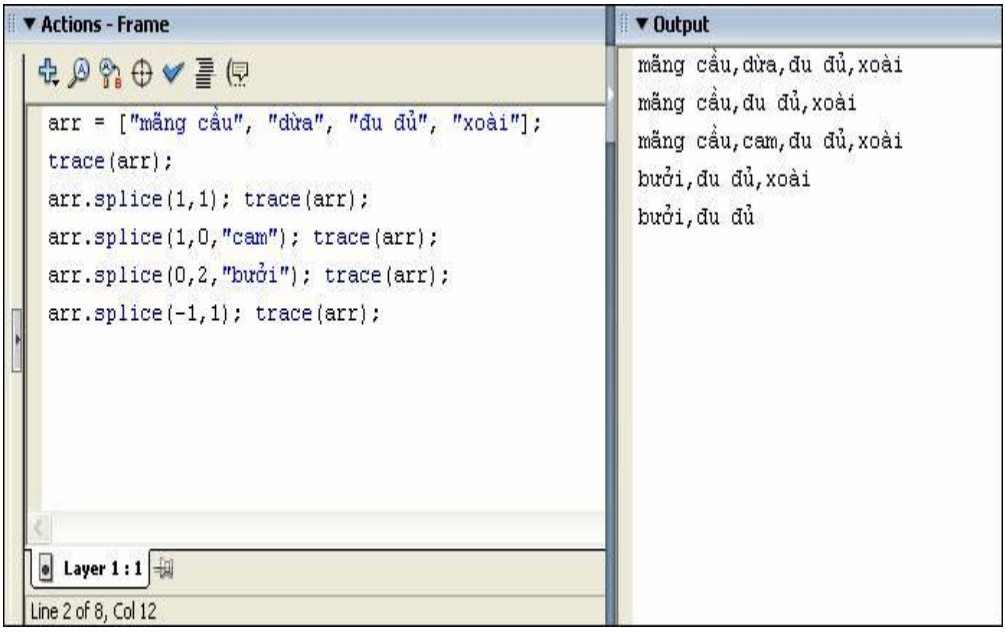
?

```
1  arr = ["mãng cầu", "dừa", "đu đủ", "xoài"];</p>
2  trace(arr);
3
4  arr.splice(1,1); trace(arr);
5
```

```
6  arr.splice(1,0,"cam"); trace(arr);
7
8  arr.splice(0,2,"bưởi"); trace(arr);
9
10 arr.splice(-1,1); trace(arr);
```

Trong đoạn mã nêu trên, ta gọi hàm splice của arr, rồi in ra ngay nội dung của arr để thấy rõ rằng dãy arr bị "xà xẻo" ra sao. Đối mục thứ nhất của hàm splice là vị trí cắt thứ nhất. Nhưng khác với hàm slice, đối mục thứ hai của hàm splice cho biết phải lấy bao nhiêu phần tử từ vị trí cắt. Nếu bạn ghi đối mục thứ hai là 0 thì Flash không cắt gì hết!

Hàm splice còn có thể có đối mục thứ ba, thứ tư,… để giúp bạn liệt kê các phần tử mà bạn muốn đưa vào dãy, thay thế cho lát cắt. Thử chạy chương trình và nghiền ngẫm kết quả (hình 2), bạn sẽ hiểu rõ ý nghĩa của từng câu lệnh.



Bài 29 : Dãy nhiều chiều – Tự học lập trình Flash

Bạn đã thấy rằng ta có thể đặt các số hoặc các chuỗi nào đó vào dãy. Phần tử của dãy có thể là mọi thứ. Nếu mỗi phần tử của dãy lại là một dãy khác, bạn có dãy hai chiều (2D array). Để làm quen với dãy hai chiều, bạn hãy mở tập tin Flash mới và gõ đoạn mã như sau trong bảng Actions – Frame (ứng với khung 1):

?

```
1 arr = new Array();
2
3 for(i = 0; i < 3; i++) {
4
5 arr[i] = ["mãng cầu", "dừa", "đu đủ", "xoài"];
6
7 }
```

Trong đoạn mã trên, ta tạo ra một dãy mang tên arr, rồi dùng vòng lặp for để tạo ra từng phần tử của dãy arr. Nhìn kỹ, bạn thấy rằng với cách dùng vòng lặp như vậy, dãy arr sẽ có ba phần tử (chiều dài của dãy là 3) và mỗi phần tử lại là một dãy khác, chứa 4 phần tử (“mãng cầu”, “dừa”, “đu đủ”, “xoài”).
Để in ra bảng Output từng phần tử của dãy arr, bạn viết thêm đoạn mã sau:

?

```
1 for(i = 0; i < arr.length; i++) {
2
3 for(j = 0; j < arr[i].length; j++)
4
5 trace(arr[i][j]);
6
7 }
```

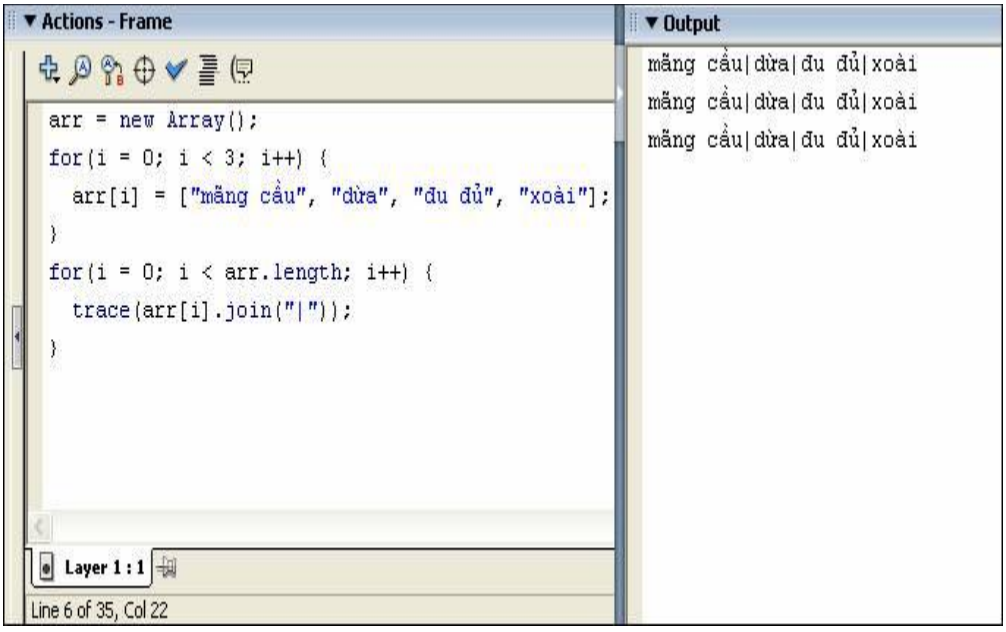
Trong đó, ta dùng hai vòng lặp for. Vòng lặp for bên ngoài for(i = 0; i < arr.length; i++) giúp bạn xem xét từng phần tử arr[i] của dãy arr. Bạn chú ý: arr.length là chiều dài của dãy arr. Trong trường hợp đang xét, arr.length có trị là 3. Tuy nhiên, bạn nên viết arr.length, đừng viết 3. Nhờ vậy, khi muốn sửa đổi chiều dài của dãy, chẳng hạn sửa 3 thành 5, bạn chỉ cần sửa ở vòng lặp for đầu tiên for(i = 0; i < 3; i++), và không cần sửa thêm ở chỗ nào khác.
Vì mỗi phần tử arr[i] lại là một dãy, vòng lặp for bên trong for(j = 0; j < arr[i].length; j++) giúp bạn in ra từng phần tử của dãy đó. Để chỉ phần tử thứ j của dãy arr[i], bạn viết một cách tự nhiên: arr[i][j].
Ấn Ctrl+Enter để chạy chương trình, bạn thấy tên bốn loại trái cây được lặp lại ba lần. Để mỗi "dãy con" arr[i] được in ra trên cùng một hàng, cho dễ phân biệt với "dãy cha" arr, bạn sửa lại đoạn mã in dãy arr như sau:

?

```
1 for(i = 0; i < arr.length; i++) {
2
```

```
3 trace(arr[i].join("|"));

4

5 }
```

Thay vì in từng phần tử của dãy arr[i], ta gọi hàm join của dãy arr[i] để nối mọi phần tử của dãy arr[i] thành một chuỗi duy nhất. Dấu vạch đứng được dùng làm “mối nối” giữa hai phần tử. Nhờ có hàm join, bạn thu được kết quả như hình 1, cho thấy rõ ràng dãy arr có ba phần tử và mỗi phần tử lại là một dãy. Kết quả in giúp bạn hình dung dãy hai chiều như một bảng, trong đó chỉ số i của “dãy cha” là chỉ số hàng và chỉ số j của “dãy con” là chỉ số cột.



Trong ví dụ vừa xét, các dãy con giống hệt nhau. Để thấy rằng các dãy con có thể khác nhau, bạn xóa đoạn mã hiện có, viết đoạn mã thử nghiệm khác như sau:

?

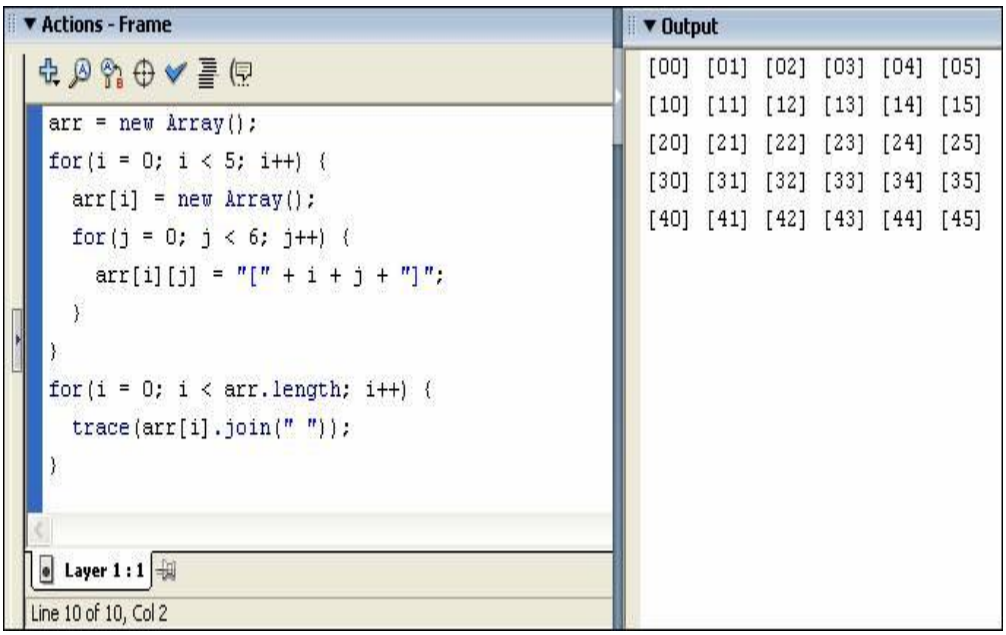
```
arr = new Array();
1
```

```
2  for(i = 0; i < 5; i++) {
3
4  arr[i] = new Array();
5
6  for(j = 0; j < 6; j++) {
7
8  arr[i][j] = "[" + i + j + "]";
9
10 }
11
12 }
13
14 for(i = 0; i < arr.length; i++) {
15
16 trace(arr[i].join(" ");
17
18 }
19
```

Nhìn vào đoạn mã vừa viết, bạn hiểu ngay: dãy arr có năm phần tử và mỗi phần tử arr[i] lại là một dãy có sáu phần tử. Mỗi phần tử trong dãy con arr[i] có dạng “[" + i + j + "]“, nghĩa là gồm hai chỉ số hàng và cột ghép lại, đặt trong cặp dấu ngoặc vuông.

Để in từng phần tử arr[i], bạn viết tương tự như ví dụ trước: gọi hàm join của dãy arr[i] để nối các phần tử của dãy thành một chuỗi duy nhất. Lần này ta làm khác một chút: dùng ký tự trắng ” ” làm mối nối. Thử chạy chương trình, bạn có kết quả như hình 2, cho thấy rõ dãy hai chiều của ta là một bảng gồm 5 hàng, 6 cột.

Có một chuyện nhỏ nhưng cũng đáng để ý: nếu bạn viết câu lệnh: arr[i][j] = i + j; để tạo phần tử của bảng, Flash sẽ lấy i cộng với j và cho kết quả là một số, chứ không phải một chuỗi. Cách viết “[" + i + j + "]” giúp Flash hiểu rằng phải chuyển i và j thành chuỗi để ghép với dấu ngoặc mở và dấu ngoặc đóng.



Trong ví dụ vừa xét, các dãy con arr[i] có chiều dài như nhau (6). Thực ra, các dãy con hoàn toàn có thể có chiều dài khác nhau.

Nếu mỗi phần tử của dãy con arr[i][j] lại là một dãy, bạn sẽ có dãy ba chiều (3D array). Để diễn đạt một phần tử của “dãy cháu” arr[i][j], bạn phải dùng ba

chỉ số, chẳng hạn: arr[i][j][k].

Cứ thế, bạn có thể có dãy nhiều chiều hơn nữa (khiếp!). Bạn yên tâm, dãy hai chiều đủ sức đáp ứng phần lớn nhu cầu thực tế.

Bài 30 : Chỉ mục của dãy – Tự học lập trình Flash

Nếu đã vui đùa với dãy qua các bài trước và có kinh nghiệm nhất định với các ngôn ngữ lập trình khác, bạn nhận ra ngay khái niệm dãy trong Flash (nói cụ thể hơn, trong ngôn ngữ ActionScript) có nhiều nét thú vị, khác lạ.

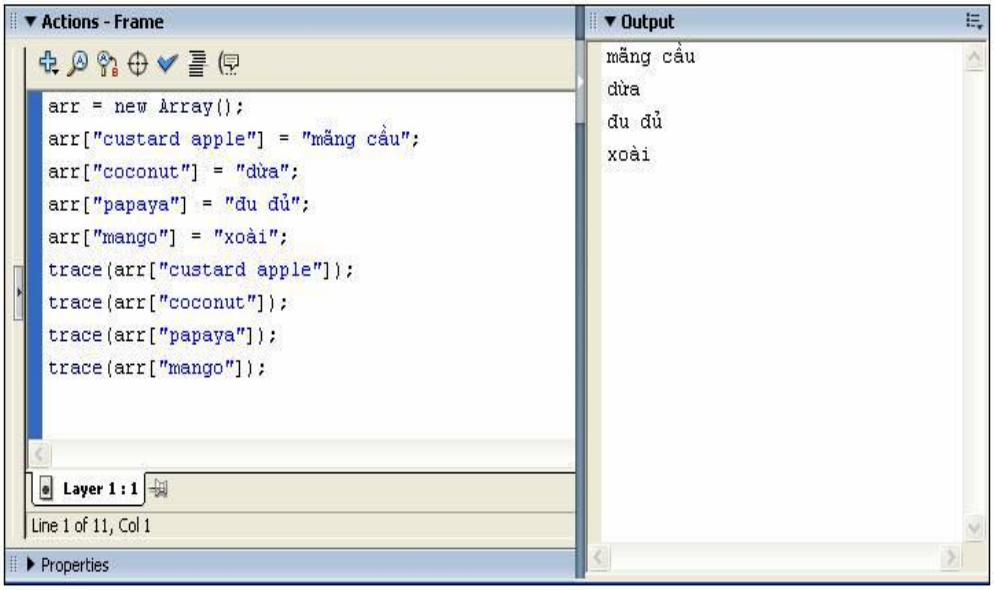
Ta hãy tìm hiểu thêm một nét khác lạ nữa: bên cạnh cách thức truy xuất phần tử trong dãy bằng chỉ số, bạn có thể truy xuất phần tử trong dãy bằng một chuỗi được gán cho phần tử. Cụ thể, bạn hãy xóa nội dung hiện có trong bảng Actions – Frame của cửa sổ Flash và gõ đoạn mã thử nghiệm mới như sau:

?

```
1  arr = new Array();
2
3  arr["custard apple"] = "mãng cầu";
4
5  arr["coconut"] = "dừa";
6
7  arr["papaya"] = "đu đủ";
```

```
8
9  arr["mango"] = "xoài";
10
11 trace(arr["custard apple"]);
12
13 trace(arr["coconut"]);
14
15 trace(arr["papaya"]);
16
17 trace(arr["mango"]);
```

Trong đoạn mã vừa nêu, bạn tạo dãy mới và lần lượt tạo bốn phần tử của dãy: “mãng cầu”, “dừa”, “đu đủ”, “xoài”. Bốn phần tử đó được đặt tương ứng với bốn chuỗi: “custard apple” (mãng cầu), “coconut” (dừa), “papaya” (đu đủ), “mango” (xoài). Các chuỗi tương ứng được đặt trong cặp dấu ngoặc vuông, có vai trò tương tự chỉ số của phần tử mà bạn đã quen thuộc. Bốn câu lệnh cuối trong đoạn mã trên in ra các phần tử của dãy trong bảng Output, nhằm giúp bạn thấy cách dùng các chuỗi tương ứng hoàn toàn giống cách dùng chỉ số (hình 1).



Ta có thể gọi chuỗi tương ứng với phần tử của dãy là chỉ mục của phần tử (thuật ngữ chỉ số không thích hợp trong trường hợp này). Nếu đã tạo ra phần tử trong dãy bằng chỉ mục tương ứng, bạn phải truy xuất phần tử đó bằng chỉ mục, chứ không thể dùng chỉ số. Bạn hiểu ngay điều này nếu thử truy xuất các phần tử của dãy hiện có bằng chỉ số:

?

```
1  arr = new Array();
2
3  arr["custard apple"] = "mãng cầu";
4
5  arr["coconut"] = "dừa";
6
7  arr["papaya"] = "đu đủ";
8
```

```
9  arr["mango"] = "xoài";
10
11 trace(arr["custard apple"]);
12
13 trace(arr["coconut"]);
14
15 trace(arr["papaya"]);
16
17 trace(arr["mango"]);
18
19 trace("------");
20
21 trace(arr[0]);
22
23 trace(arr[1]);
24
25 trace(arr[2]);
26
```

```
27 trace(arr[3]);
```

Với các câu lệnh viết thêm, bạn thu được kết quả như sau ở bảng Output, cho thấy rõ không thể dùng chỉ số thay cho chỉ mục:

mãng cầu

dừa

đu đủ

xoài

——

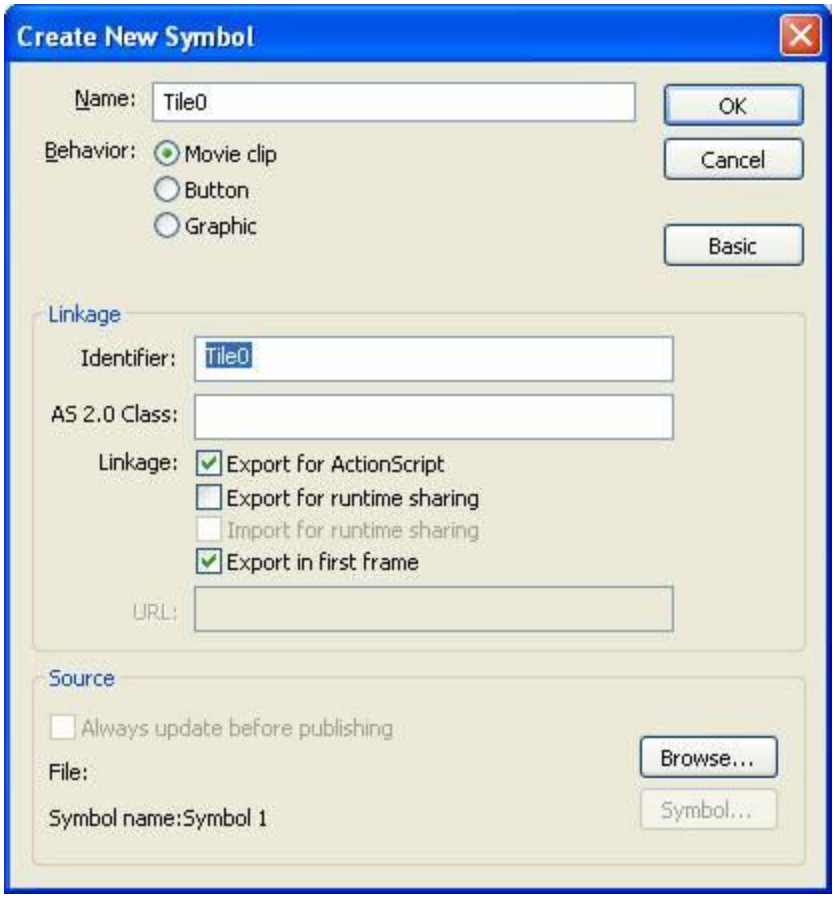
undefined

undefined

undefined

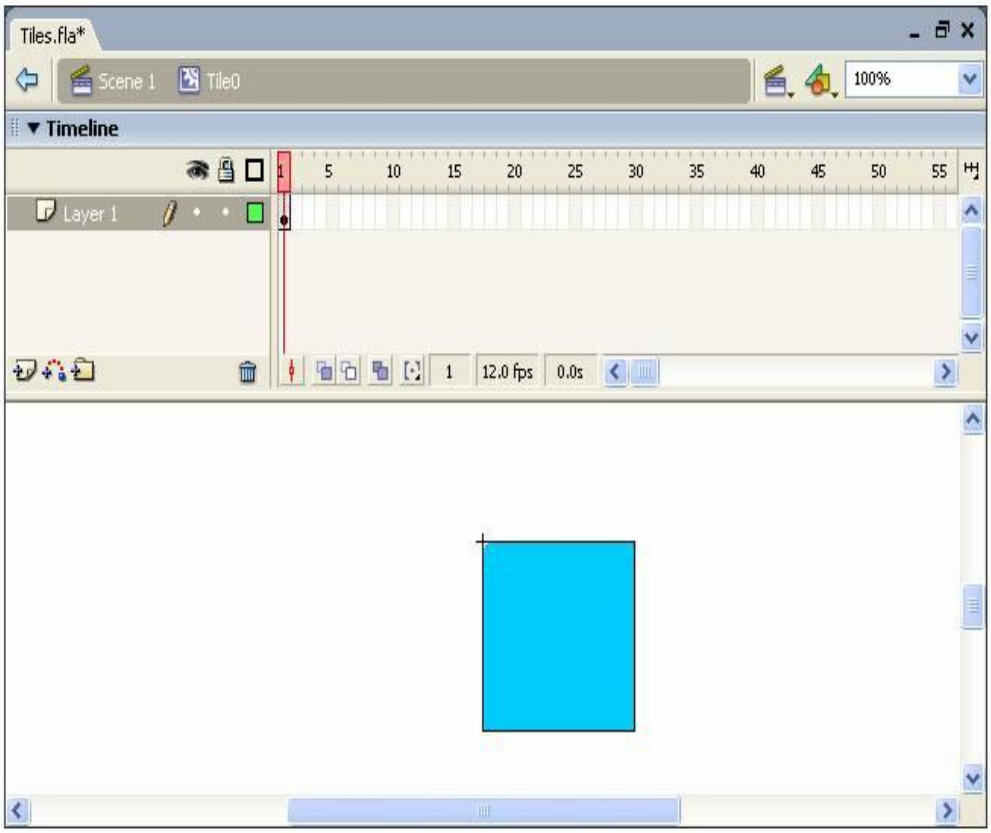
undefined

Việc sử dụng chỉ mục để truy xuất phần tử của dãy thường xảy ra đối với dãy của các thể hiện. Để khảo sát dãy của các thể hiện, bạn hãy mở tập tin Flash mới và lần lượt tạo ra bốn thể hiện là các hình vuông có màu khác nhau. Cụ thể, bạn ấn Ctrl+F8 để mở hộp thoại Create New Symbol (hình 2), gõ tên Tile0, chọn Export for ActionScript (điều này cần thiết vì ta sẽ tạo ra thể hiện của nhân vật Tile0 bằng câu lệnh, chứ không phải bằng cách kéo nhân vật từ bảng Library vào sân khấu).

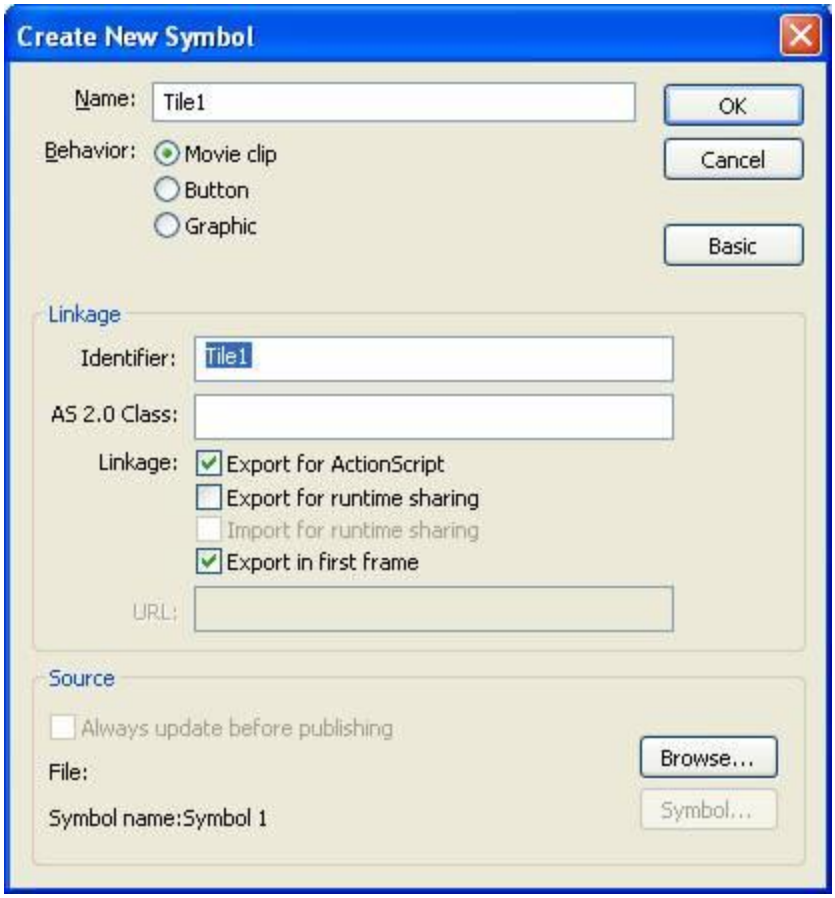


Bấm OK để đóng hộp thoại Create New Symbol, bạn chọn màu tô ở ô Fill Color , chọn công cụ vẽ hình khung Rectangle Tool , trỏ vào dấu thập (điểm mốc) trong miền vẽ, giữ phím Shift, kéo chuột qua phải, xuống dưới để có được hình vuông. Nhờ vậy, điểm mốc của nhân vật Tile0 là góc trên, bên trái của hình vuông (hình 3). Bạn ấn Ctrl+F3 để mở bảng Properties, gõ 100 trong ô W (chiều rộng) và gõ 100 trong ô H (chiều cao) để hình vuông có kích thước 100×100. Để dẹp bảng Properties, bạn lại ấn Ctrl+F3.

Bạn chọn mục Scene1 để trở về với sân khấu, kết thúc việc tạo hình cho nhân vật Tile0.



Để có nhân vật Tile1 là hình vuông giống hệt như Tile0, chỉ khác màu tô, bạn có thể sao chép Tile0 cho nhanh. Bạn gõ phím F11 để mở bảng Library (nếu bảng Library chưa được mở), bấm phải vào mục Tile0 trong bảng Library, chọn Duplicate. Hộp thoại Duplicate vừa hiện ra (hình 4), bạn gõ Tile1 để đặt tên cho bản sao, chọn Export for ActionScript và bấm OK.



Muốn đổi màu tô cho nhân vật Tile1, bạn bấm kép vào biểu tượng Tile1 trong bảng Library (hoặc bấm-phải vào mục Tile1 và chọn Edit) để chuyển qua chế độ chỉnh sửa nhân vật Tile1, chọn màu tô khác ở ô Fill Color , chọn công cụ tô và bấm vào hình vuông Tile1 trong miền vẽ. Bạn chọn mục Scene1 để kết thúc việc chỉnh sửa nhân vật Tile1, trở về với sân khấu.

Theo cách tương tự, bạn tạo ra hai hình vuông nữa (nhân vật Tile2 và Tile3) với màu tô khác.

Bạn mở bảng Actions – Frame (ứng với khung 1), gõ đoạn mã như sau để tạo ra bốn thể hiện của bốn nhân vật hiện có (Tile0, Tile1, Tile2, Tile3) và xếp chúng thành hàng ngang:

?

```
1 for(i = 0; i < 4; i++) {
```

```
2
3 attachMovie("Tile" + i, "tile" + i, i);
4
5 this["tile" + i]._x = 120 * i;
6
7 this["tile" + i]._y = 20;
8
9 }
```

Ấn Ctrl+Enter để chạy thử, bạn có được kết quả như hình 5.

Đoạn mã vừa viết tạo ra các thể hiện mang tên tile0, tile1, tile2, tile3 từ các nhân vật Tile0, Tile1, Tile2, Tile3. Ta quy định vị trí của các thể hiện bằng cách gán trị số thích hợp cho thuộc tính _x và _y của từng thể hiện. Hai câu lệnh trong vòng lặp nhằm quy định vị trí cho các thể hiện tương đương với các câu lệnh sau:

?

```
this["tile0"]._x = 0;

this["tile0"]._y = 20;

this["tile1"]._x = 120;

this["tile1"]._y = 20;

this["tile2"]._x = 240;

this["tile2"]._y = 20;

this["tile3"]._x = 360;

this["tile3"]._y = 20;
```

Như bạn thấy, ta truy xuất thể hiện tile0 bằng cách viết this["tile0"]. Từ chốt this ở đây chỉ một dãy được tạo sẵn, chứa các thể hiện được tạo ra ở thời tuyến chính. Trong dãy đó, chỉ mục của từng thể hiện chính là tên của thể hiện. Cách truy xuất thể hiện nêu trên thích hợp cho vòng lặp, rất thuận tiện khi bạn phải điều khiển khá nhiều thể hiện trên sân khấu.

Bài 31 : Trò chơi “lật hình” – Tự học lập trình Flash

Kiến thức về lập trình Flash mà bạn đã tích lũy cho phép nghĩ đến trò chơi phức tạp hơn trước.

Bạn đã hiểu biết về dãy, vì vậy chỉ cần đặt tên thích hợp cho những thể hiện, bạn dễ dàng điều khiển nhiều thể hiện trên sân khấu. Ta có thể xem xét trò chơi “lật hình” quen thuộc, trong đó người chơi bấm chuột để lật hai hình liên tiếp, nếu hai hình đó giống nhau, chúng không bị úp xuống trở lại. Trò chơi kết thúc khi mọi hình đều được lật lên. Thành tích của người chơi được thể hiện bởi số lần bấm chuột (số lần bấm chuột càng ít càng tốt) hoặc bởi thời gian hoàn thành trò chơi.

Ta hãy bắt đầu từ việc đơn giản: sắp xếp các hình thành hàng và cột. Bạn thử hình dung ta có 16 hình, xếp thành 4 hàng, 4 cột, trong đó có 8 cặp hình giống nhau. Trong bài trước, bạn đã tạo được 4 hình vuông khác nhau (có màu tô khác nhau). Ta cần có thêm 4 hình vuông khác nữa trong thư viện Library để có đủ 8 hình khác nhau. Trước mắt, ta tạo ra các hình vuông khác nhau bằng cách tô màu khác nhau. Sau này, khi chương trình chạy tốt, bạn vẽ thêm hình vui mắt gì đó vào mỗi hình vuông trong Library hoặc lấy hình từ mạng.

Bạn hãy mở lại tập tin FLA đã tạo ra trong bài trước, gõ phím F11 để mở bảng Library. Trong bảng Library, bạn bấm-phải vào mục Tile0, chọn Duplicate. Hộp thoại Duplicate hiện ra, bạn gõ tên Tile4, rồi chọn Export for ActionScript và bấm OK. Thao tác này sao chép nhân vật Tile0, tạo ra nhân vật mới Tile4.

Để nhân vật mới Tile4 có màu tô khác với Tile0, bạn lại bấm-phải vào mục Tile0 trong bảng Library, chọn Edit để chuyển qua chế độ chỉnh sửa nhân vật.

Bạn chọn màu tô khác ở ô Fill Color , chọn công cụ tô và bấm vào hình vuông Tile4 trong miền vẽ.

Để có nhân vật mới Tile5 có màu tô khác, bạn lặp lại thao tác như trên để sao chép nhân vật Tile0 thành nhân vật Tile5 và chọn màu tô cho Tile5 khác với những màu tô đã dùng.

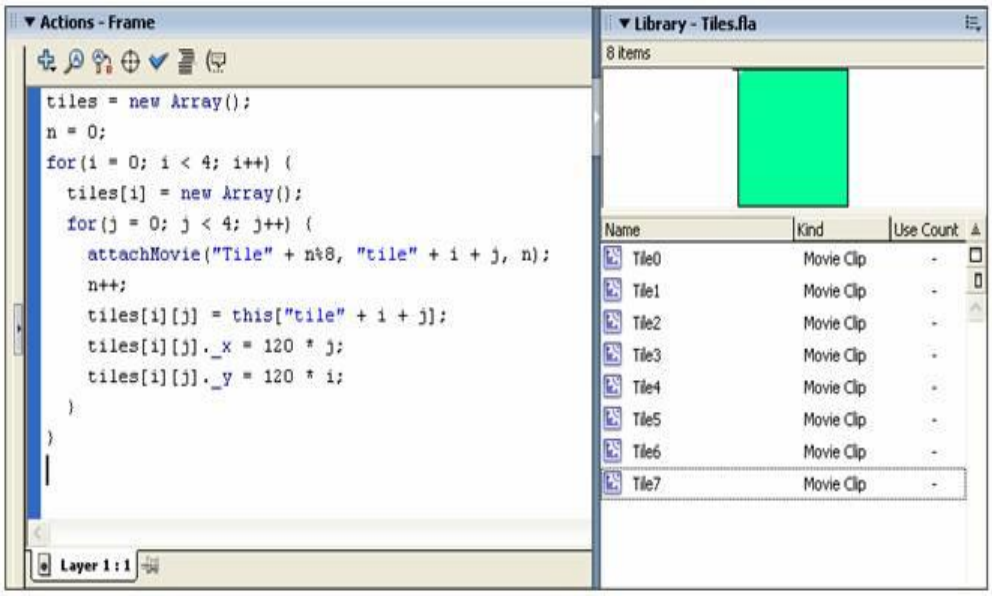
Cứ thế, bạn tạo thêm nhân vật Tile6 và Tile7 để có được cả thảy 8 nhân vật hình vuông (từ Tile0 đến Tile7) với màu tô khác nhau. Gõ phím F9 để mở bảng Actions – Frame, bạn xóa đoạn mã đã viết từ bài trước, viết đoạn mã khác như sau:

?

```
1  tiles = new Array();
2
3  n = 0;
4
5  for(i = 0; i < 4; i++) {
6
7  tiles[i] = new Array();
8
9  for(j = 0; j < 4; j++) {
10
11 attachMovie("Tile" + n%8, "tile" + i + j, n);
```

```
12
13 n++;
14
15 tiles[i][j] = this["tile" + i + j];
16
17 tiles[i][j]._x = 120 * j;
18
19 tiles[i][j]._y = 120 * i;
20
21 }
22
23 }
```

Trong đoạn mã vừa viết, ta dùng dãy hai chiều tiles để “quản lý” 16 thể hiện được tạo ra từ 8 nhân vật trong thư viện (hình 1). Dãy hai chiều phù hợp với cách sắp xếp thành hàng và cột của các thể hiện trên sân khấu. Việc đưa các thể hiện vào dãy được thực hiện bởi hai vòng lặp for.



Vòng lặp ngoài có chỉ số i chạy từ 0 đến 3. Trong mỗi lần lặp theo chỉ số i, ta tạo ra một phần tử của dãy tiles: tiles[i] = new Array();. Nhờ vậy, dãy tiles có 4 phần tử, mỗi phần tử lại là một dãy khác, tạm gọi là dãy con.

Vòng lặp trong có chỉ số j chạy từ 0 đến 3, có nhiệm vụ đưa 4 phần tử vào dãy con vừa tạo ra (mỗi dãy con có 4 phần tử). Trong mỗi lần lặp theo chỉ số j, ta lại tạo ra một thể hiện từ nhân vật trong Library.

Bạn chú ý cách đặt tên cho thể hiện. Khi biến i có trị số là 0, biến j có trị số là 0, thể hiện được tạo ra có tên là “tile” + i + j, tức là tile00. Tương tự, khi i là 0, j là 1, thể hiện có tên là tile01,…

Do câu lệnh tiles[i][j] = this["tile" + i + j]; thể hiện tile00 được đưa vào dãy tiles, trở thành phần tử tiles[0][0]. Tương tự, thể hiện tiles01 trở thành phần tử tiles[0][1] trong dãy tiles,…

Hai câu lệnh quy định vị trí cho các thể hiện:

?

```
1 tiles[i][j]._x = 120 * j;

2
```

```
3 tiles[i][j]._y = 120 * i;
```

tương đương với nhiều câu lệnh như sau:

?

```
   tiles[0][0]._x = 0;
1

2
   tiles[0][0]._y = 0;
3

4
   tiles[0][1]._x = 120;
5

6
   tiles[0][1]._y = 0;
7

8
   tiles[0][2]._x = 240;
9

10
   tiles[0][2]._y = 0;
11

12
   ...
13

14
   tiles[1][0]._x = 0;
15
```

```
16 tiles[1][0]._y = 120;
17
18 tiles[1][1]._x = 120;
19
20 tiles[1][1]._y = 120;
21
22 tiles[1][2]._x = 240;
23
24 tiles[1][2]._y = 120;
25
26 ...
27
```

Với cách sắp xếp như vậy, bạn thấy rõ các phần tử trong dãy tiles có chỉ số i giống nhau được xếp trên cùng một hàng. Các phần tử có chỉ số j giống nhau thuộc cùng một cột. Vì vậy, ta gọi i là chỉ số hàng, j là chỉ số cột.
Trong câu lệnh tạo thể hiện
attachMovie(“Tile” + n%8, “tile” + i + j, n);
bạn chú ý biểu thức n % 8, trong đó n có trị số ban đầu là 0 và tăng một đơn vị trong mỗi lần lặp do câu lệnh n++; (sau câu lệnh tạo thể hiện). Ký hiệu % chỉ tác tử modulo. Biểu thức n % 8 cho ta số dư trong phép chia của n cho 8. Nếu n có trị số nhỏ hơn 8, tức là các trị số từ 0 đến 7, biểu thức n % 8 cũng

cho trị số từ 0 đến 7. Nếu n có trị số từ 8 trở lên, chẳng hạn các trị số từ 8 đến 15, biểu thức n % 8 vẫn cho các trị số tương ứng từ 0 đến 7. Chẳng hạn, khi n là 8, biểu thức n % 8 cho trị số 0 (số dư của phép chia 8 cho 8). Khi n là 9, biểu thức n % 8 cho trị số 1 (số dư của phép chia 9 cho 8).

Nói chung biểu thức n % 8 luôn luôn cho trị số từ 0 đến 7, không bao giờ cho trị số vượt ra ngoài phạm vi đó. Nhờ vậy, biểu thức “Tile” + n%8 luôn cho kết quả trong phạm vi từ Tile0 đến Tile7, dù biến n tăng đều do vòng lặp.

Ấn Ctrl+Enter để chạy thử chương trình, bạn thu được kết quả như hình 2.