LEZIONE 7

ANCORA SULLA CLASSE MOVIECLIP:

LA CLASSE MOVIECLIP

- La classe MovieClip è l'oggetto software che ci consente di gestire gli elementi visuali in Flash da ActionScript.
- Attraverso la classe Movie Clip posso gestire:
  - MovieClip creati runtime utilizzando ActionScript.
  - Simboli di tipo MovieClip memorizzati in libreria e trascinati durante lo sviluppo sullo stage.
  - Simboli di tipo MovieClip memorizzati in libreria e caricati sullo stage in fase runtime.
  - Filmati memorizzati su disco e caricati runtime all'interno del filmato principale.
  - Immagini bitmap (formato jpeg) memorizzate su disco.

CARICAMENTO DI CONTENUTI ESTERNI

CARICAMENTO DI UN FILMATO O DI UN IMMAGINE DA DISCO

- Per caricare dinamicamente un filmato (.swf) o un'immagine jpeg (.jpg) da disco ho a disposizione tre strategie:
  - Caricare il contenuto al posto del filmato principale
  - Caricare il contenuto nel filmato principale su un altro level
  - Utilizzare una MovieClip per caricare il contenuto esterno.
- L'utilizzo di una MovieClip mi dà poi due ulteriori alternative:
  - Utilizzazione del metodo MovieClip.loadMovie
  - Utilizzo dell'oggetto MovieClipLoader.

SOSTITUZIONE DEI CONTENUTI DEL FILMATO PRINCIPALE

- Se semplicemente voglio sostituire il contenuto del mio filmato principale con un altro filmato presente su disco posso usare il comando globale loadMovie() che accetta come parametro l'url del filmato da caricare.
- Devo tener presente che:
  - Tutti i contenuti presenti nello stage
  - Tutti i contenuti presenti nella libreria del filmato
  - Tutte le funzioni e le classi definite nel filmato

  andranno persi.

UTILIZZO DEI LIVELLI

- Se invece voglio aggiungere dei contenuti al filmato principale o sostituirne una parte posso utilizzare i livelli o le movie clip.
- Per utilizzare i livelli userò il comando **loadMovieNum(url, livello)**
- I contenuti verranno caricati nel livello specificato nel parametro
- Il livello è un'oggetto di tipo MovieClip che viene creato automaticamete (se già non esiste) e che si chiama **_level*n***

UTILIZZO DI UNA MOVIECLIP

- Un'alternativa ai livelli (molto più flessibile e da noi preferita) è l'utilizzo di una MovieClip gia presente nel filmato o creata appositamente.
- Basta applicare il metodo **loadMovie()** alla MovieClip.
- Se la clip è vuota, dopo il caricamento il suo contenuto sarà costituito dal filmato o dall'immagine scaricata
- Se nella MovieClip è presente un qualche contenuto, il contenuto sarà sostituito dal contenuto caricato.

UTILIZZO DI MOVIECLIPLOADER

- MovieClipLoader è una classe presente in flash dalla versione 7 in poi molto utile per il controllo del caricamento di contenuti in un filmato.
- Come per tutte le classi prima di tutto dovrò creare un'istanza della per utilizzarla:

```
var myLoader:MovieClipLoader = new MovieClipLoader();
```

- Dopo di che potrò applicare all'istanza creata il metodo loadClip(url, movieClip) che mi consente di caricare un contenuto esterno in una movie clip

```
myLoader.loadClip("unfilmato.swf", my_clip_mc);
```

UTILIZZO DI MOVIECLIPLOADER

- Il vantaggio rispetto all'uso diretto di MovieClip è che l'oggetto mi fornische 5 gestori eventi che mi consentono un pieno controllo del caricameto:
  - **onLoadStart** che mi comunica che il caricamento è iniziato
  - **onLoadProgress** che mi comunica continuamente lo stato del caricamento
  - **onLoadComplete** che mi dice che il caricamento è completato
  - **onLoadInit** che mi comunica che, terminato il caricamento, il filamto è stato inizializzato e quindi i suoi metodi e le sue proprietà sono disponibili
  - **onLoadError** che mi segnala errori

GESTIONE DEGLI EVENTI

COSA È UN EVENTO

- In programmazione un evento è un avvenimento che il mio programma è in grado di registrare e in seguito al quale posso programmare un azione.
- Esempi di eventi:
  - Le interazioni dell'utente con il mouse o la tastiera
  - Il passaggio al frame successivo in un clip filamato
  - Eventi legati al flusso dei dati (inizio caricamento, completamento del caricamento, ecc)
  - Eventi legati alla riproduzione di suono
  - Registrazione di errori
  - Ecc.

GESTIONE DEGLI EVENTI

- Gestire un evento significa programmare un'azione che viene eseguita ogni volta che l'evento si verifica.
- Gli eventi, in Action Script si gestiscono attraverso le classi.
- Quando un determinato evento si verifica (ad esempio l'utente clicca con il mouse su un pulsante o viene mostrato sullo schermo il frame successivo di un clip filmato) l'evento viene notificato all'istanza della classe bersaglio dell'evento.
- Io posso gestire questa notifica facendo in modo che l'istanza della classe in questione compia un azione ogni volta che l'evento viene notificato.

I GESTORI DI EVENTI

- Il target di un evento è sempre l'istanza di una classe
- In flash ho due tipi di gestori di eventi:
  - Gestione di eventi attraverso metodi specifici della classe
  - Gestione di eventi attraverso oggetti listener (modello listener-broadcaster).
- Alcune classi utilizzano il primo modello altre il secondo

METODI GESTORI DI EVENTI

- Un metodo gestore di un evento è un metodo della classe che viene richiamato quando un'istanza della classe riceve la notifica di un evento.
- La classe MovieClip, ad esempio, definisce un gestore di evento **onPress** che viene richiamato ogni volta che viene premuto il pulsante sinistro del mouse quando il puntatore si trova sulla MovieClip stessa.
- Per default il metodo gestore di eventi non contiene alcuna azione. Per programmare una azione in risposta ad un determinato evento devo assegnare al gestore di eventi una funzione anonima che contiene il codice che verrà eseguito ogni volta che l'evento si verifica.

METODI GESTORI DI EVENTI

- La gestione di un evento, secondo questo approccio, è costituita da tre elementi:
  - l'oggetto al quale si applica l'evento (o che ***riceve la notifica*** dell'evento),
  - il nome del metodo che la classe definisce per gestire l'evento
  - la funzione assegnata al gestore di eventi.
- Ecco la struttura di base di un gestore di eventi:

```
object.eventMethod = function () {
   /* Inserire qui il codice che
   risponde all'evento.*/
}
```

MODELLO LISTENER/BROADCASTER

- Il modello listener/broadcaster per gli eventi, prevede che ci sia un oggetto (detto ***listener***, *che sta in ascolto*) che riceve notifiche di eventi trasmesse da un altro oggetto detto ***broadcaster*** (trasmettitore).
- Le classi che usano questo modello non implementano, quindi, specifici metodi per rispondere agli eventi, ma un metodo generico (***addListener***) che consente di comunicare all'istanza della classe a quali oggetti mandare le notifiche degli eventi.
- L'oggetto listener può ascoltare più eventi, l'oggetto broadcaster puo spedire notifica dello stesso evento a più oggetti.

MODELLO LISTENER/BROADCASTER

- Tipicamente la gestione di un evento, secondo questo approccio, comporta tre passaggi:
  - Creazione dell'oggetto che ascolta gli eventi (o oggetto listener)
  - Assegnazione all'oggetto delle funzioni che gestiscono gli eventi sotto forma di proprietà
  - Registrazione dell'oggetto listener da parte dell'oggetto broacaster utilizzando il metodo ***addListener***.

MODELLO LISTENER/BROADCASTER [1]

- Il primo passo consiste nel creare un oggetto generico (cioè di tipo **Object**) che diventrà il mio listener:

```
myListener = new Object();
```

- Con questo codice ho creato un oggetto di tipo **Object** che ha nome myListener

MODELLO LISTENER/BROADCASTER [2]

- Una proprietà di un oggetto può contenere dati di qualsisasi tipo, una serie di comandi ActionScript (cioè una funzione). Per far diventare il mio oggetto un ***listener*** devo creare una proprietà che abbia il nome dell'evento da gestire e assegnare a quella proprietà il codice da eseguire quando l'evento si verifica:

```
myListener.eventName = function (p1, p2,…) {
   /* Inserire qui il codice che risponde
   all'evento.*/
}
```

- Il codice è praticamente identico a quello usato per i metodi gestori di eventi. La differenza è che invece di applicarlo all'istanza della classe che riceve l'evento lo applichiamo ad un oggetto creato ad hoc.

MODELLO LISTENER/BROADCASTER [3]

- Infine devo comunicare all'oggetto ***broadcaster*** (l'oggetto principale, quello per il quale devo gestire gli eventi) che gli eventi vanno inviati al mio listener

```
broadcasterObject.addListener(myListener);
```

- Con il metodo ***addListener*** posso registrare più oggetti listener allo stesso broadcaster.
- Con il metodo ***removeListener*** posso, se necessario, togliere un listener dalla lista a cui il broadcaster notifica gli eventi.

LA PAROLA RISERVATA THIS

- Un'altra importante differenza tra il codice scritto per gestire un evento secondo il modello 'metodo gestore di evento' e il modello oggetto brodcaster→oggetto listener è il significato che nei due contesti ha la parola chiave ***this***
- ***this*** indica sempre l'oggetto a cui appartiene la funzione in cui viene usato (o l'istanza della classe a cui è applicato il metodo in cui viene usato)

THIS IN UN METODO GESTORE DI EVENTI

- In un metodo gestore di eventi this rappresenta l'oggetto bersaglio dell'evento:

```
myClip.onEnterFrame = function () {
   this._rotation = this._rotation + 1;
}
```

- Il codice precedente usa l'evento onEnterFrame per far ruotare il Clip Filmato myClip
- La proprietà onEnterFrame è una proprietà di myClip quindi in questo caso this indica myClip

THIS NEL MODELLO LISTENER => BROADCASTER

- In un listener this rappresenta l'oggetto listener stesso:

```
myLoader = new MovieClipLoader();
myListener = new Object();
myListener.onLoadInit = function
(myMc:MovieClip) {
   myMc._alpha = 100;
}
myLoader.addListener(myListener);
theClip._alpha = 0;
myLoader.loadClip("pippo.swf",theClip);
```

THIS NEL MODELLO LISTENER => BROADCASTER

- Il codice della slide precedente carica una clip utilizzando l'oggetto ***MovieClipLoader.*** Quando si verifica l'evento **loadInit** la clip viene visualizzata impostanto la sua proprietà **_alpha** a 100 (assenza di trasparenza).
- In questo caso non si potrebbe ulitizzare ***this*** per modificare la proprietà della clip. ***this*** infatti indica sempre l'oggetto a cui appartiene la funzione in cui viene usato, il questo caso l'oggetto listener.
- Per semplificarci le cose la notifica dell'evento ci procura come parametro il riferimento alla MovieClip caricata. Per intervenire sulla clip si utilizzerà, quindi, questo parametro.

CLASSI CHE DEFINISCONO METODI GESTORI DI EVENTI:

- Button
- ContextMenu, ContextMenuItem
- LoadVars
- LocalConnection
- MovieClip
- SharedObject
- Sound
- TextField
- XML, XMLSocket.

CLASSI CHE USANO IL MODELLLO LISTENER:

- Key
- Mouse
- MovieClipLoader
- Selection
- Stage

PER APPROFONDIRE

fl8_learning_as2.pdf

Pagina 311 e seguenti

UNA SUPERFICIE SU CUI DISEGNARE

I METODI PER DISEGNARE

- È possibile disegnare linee e riempimenti sullo stage con Action Script, utilizzando i metodi di disegno della classe MovieClip.
- L'oggetto MovieClip a cui si applicano i metodi è la superficie su cui si disegna. La definizione e la modifica delle proprietà della MovieClip (colorazione, rotazione, posizione, trasparenza, scala, ecc.) influiranno sull'aspetto dei disegni.
- La superfice della MovieClip è trasparente. Disegnale su una MovieClip è come disegnare su un lucido.
- I metodi di disegna che la classe MovieClip definisce sono i seguenti:
  - ***beginFill()***
  - ***beginGradientFill()***
  - ***clear()***
  - ***curveTo()***
  - ***endFill()***
  - ***lineTo()***
  - ***lineStyle()***
  - ***moveTo()***

DISEGNARE UN LINEA

- Questo codice disegna una linea

```
this.createEmptyMovieClip("line_mc", 10);
line_mc.lineStyle(1, 0x000000, 100);
line_mc.moveTo(0, 0);
line_mc.lineTo(200, 100);
line_mc._x = 100;
line_mc._y = 100;
```

- Viene creata la MovieClip
- Vengono impostate le caratteristiche della linea
- Il penello vien spostato alla posizione 0,0
- Viene disegnata la linea
- Viene spostata la movie clip

DISEGNARE UN TRIANGOLO

- Questo codice disegna una triangolo

```
this.createEmptyMovieClip("tri_mc", 1);
with (tri_mc) {
   lineStyle(5, 0xFF00FF, 100);
   moveTo(200, 200);
   lineTo(300, 300);
   lineTo(100, 300);
   lineTo(200, 200);
}
```

- Invece di ripetere tri_mc posso usare il costrutto with

DISEGNARE UN TRIANGOLO

- Scrivere una funzione che disegna un rettangolo

```
this.createEmptyMovieClip("rectangle_mc", 10);
rectangle_mc._x = 100;
rectangle_mc._y = 100;
drawRectangle(rectangle_mc, 240, 180, 0x99FF00, 100);
function drawRectangle(target_mc:MovieClip, boxWidth:Number,
                       boxHeight:Number, fillColor:Number,
                       fillAlpha:Number):Void {
        with (target_mc) {
                beginFill(fillColor, fillAlpha);
                moveTo(0, 0);
                lineTo(boxWidth, 0);
                lineTo(boxWidth, boxHeight);
                lineTo(0, boxHeight);
                lineTo(0, 0);
                endFill();
        }
}
```

PER APPROFONDIRE

fl8_learning_as2.pdf

Pagina 590 e seguenti

CREIAMO UNA CLASSE

FASI DELLA CREAZIONE DI UNA CLASSE

- Le fasi di realizzazione di una classe possono essere così riassunte:
  - ***Progettazione****:* comprende il lavoro da svolgere prima della scrittura del codice. Si tratta di definire il funzionamento del nuovo oggetto che la classe mette disposizione, e quindi stabilire quali proprietà e metodi saranno necessari.
  - ***Scrittura del codice****.* Si passerà quindi alla fase di scrittura del codice. Dovremo realizzare un file di classe secondo le specifiche che abbiamo visto nel capitolo precedente, preoccupandoci soprattutto della corretta sintassi di quanto scriviamo.
  - ***Test e correzione degli errori***. Dovremo poi creare un file FLA di prova in cui testare il corretto funzionamento della classe e correggere gli eventuali errori
  - ***Inserimento in una o più applicazioni***. Una volta testato il funzionamento della classe passeremo al suo utilizzo in alcune applicazioni flash.

PROGETTAZIONE

- Le fasi di progettazione è probabilmente la più importante riduce possibilità di errore, necessità di aggiustamenti. In altri termini ci fa risparmiare tempo.
- Il fase progettuale dobbiamo:
  - Definire l'obbiettivo del nuovo tipo di oggetto che andiamo a realizzare
  - Definire le funzionalità che ci permettono di conseguire il nostro obbiettivo
  - Definire le proprietà che ci servono per modificare il comportamento dell'oggetto
  - Definire i metodi che consentiranno al nostro oggetto di svolgere le suo funzioni

DEFINIZIONE DELL'OBIETTIVO

- Il nostro obbiettivo è creare un semplice oggetto riutilizzabile che carichi un contenuto esterno e lo mostri sullo stage.
- L'oggetto dovrà mostrare una barra che segnali all'utente il progresso di caricamento del contenuto che voglio caricare
- Chiamerò la classe PBarLoader

FUNZIONALITÀ

- Per il nostro PBarLoader definiamo alcune funzionalità base:
  - Dovrò poter definire in che **posizione** collocare limagine o il filmato caricati.
  - La barra che segnala il progresso di caricamento dovrà essere **collocata al centro** dell'area di caricamento.
  - Dovrò poter definire il **colore** della barra che segnala il progresso del filmato

PROPRIETÀ

- Le proprietà sono variabili che definisco all'interno della classe e che contengono le informazioni e i riferimenti agli oggetti utili al funzionamento della classe.
- Le proprietà possono essere pubbliche o private:
  - Le proprietà pubbliche sono quelle utilizzando le quali personalizzo l'istanza delle classe
  - Le proprietà private sono quelle che uso esclusivamente nel codice scritto nella definizione di classe.
- Inoltre per ogni proprieta dovrò stabilire il tipo e definire un valore di default

PROPRIETÀ

- Dovrò creare un MovieClip per caricare il contenuto esterno e una seconda MovieClip su cui disegnare la barra di caricamento.
- Una MovieClip deve essere creata all'interno di una MovieClip che la contiene.
- Ecco quindi le tre prime proprietà che definiamo

Nome	Descrizione	Visibilità	Tipo	Default
image_mc	MovieClip che verrà usata per caricare il contenuto esterno	private	MovieClip	
progress_mc	MovieClip su cui sarà disegnata la barra di caricamento	private	MovieClip	
parent_mc	MovieClip che contiene image_mc	private	MovieClip	_root

PROPRIETÀ

- Per caricare il contenuto esterno utilizzerò l'oggetto MovieClipLoader.
- Creerò l'oggetto alla creazione del mio **PBarLoader** e memorizzero il suo riferimento in una proprietà. Creerò anche le proprietà onLoadInit, onLoadProgress e onLoadError che conterranno il codice che verrà eseguito in risposta agli eventi dello stesso nome notificati da myLoader:

Nome	Descrizione	Visibilità	Tipo	Default
myLoader	Oggetto MovieClipLoader che utilizzerò per il caricamento dei dati esterni	private	MovieClipLoader	
onLoadInit	Risposta all'evento onLoadInit	private	Function	
onLoadProgress	Risposta all'evento onLoadProgress	private	Function	
onLoadError	Risposta all'evento onLoadError	private	Function	

PROPRIETÀ

- Dovrò comunicare al mio oggetto la posizione, le dimensioni del contenuto da caricare in modo da poter collocare la barra di caricamento al centro.
- Per semplificare per il momento stabilisco che queste proprietà potranno essere impostate solo alla creazione le definisco, quindi, private.

Nome	Descrizione	Visibilità	Tipo	Default
x	Posizione x del contenuto caricato	private	Number	0
y	Posizione y del contenuto caricato	private	Number	0
width	Larghezza del contenuto caricato	private	Number	Stage
height	Altezza del contenuto caricato	private	Number	Stage

PROPRIETÀ

- Infine definirò le proprietà che mi consentono di personalizzare la barra di caricamento: dimensioni e colore.

Nome	Descrizione	Visibilità	Tipo	Default
bgcolor	Colore di sfondo della barra	public	Number	**0xCCCCCC**
color	Colore della barra	public	Number	**0x000000**
bar_width	Larghezza della barra	public	Number	200
bar_height	Altezza della barra	public	Number	5

I METODI

- I metodi sono funzioni che definisco all'interno della classe e che svolgono le operazioni necessarie al funzionamento della classe.
- I metodi, come le proprietà, possono essere pubblici o privati:
  - I metodi pubblici sono quelli che utilizzo per far svolgere agli oggetti istanze della classe la loro funzione
  - I metodi privati sono quelli usati esclusivamente da altri metodi della classe.

I METODI

- In PBarLoader definirò quattro metodi
  - Il constructor della classe che che creerà gli oggetti necessari e inizializzerà le proprietà della classe
  - Il metodo pubblico loadClip (il comando che caricherà il contenuto esterno)
  - Il due metodi privati initBar e writeBar che rispettivamente inizializzaranno e aggiorneranno la barra di caricamento.

I METODI: CONSTRUCTOR

- In constructor è sempre un metodo pubblico che ha lo stesso nome della classe. Potrà avere da uno a cinque parametri:
  - ***PBarLoader(Contenitore:MovieClip, xpos:Number, ypos:Number, largehzza:Number, altezza:Number)***.
- Il primo parametro è obbligatorio, gli altri mi consentono di inizializzare delle proprietà per cui ho previsto in alternativa dei valori di default. Azioni del metodo:
  - Inizializzazione della proprietà **parent_mc**.
  - Creazione della MovieClip ***image_mc***
  - Creazione della MovieClip ***progress_mc***
  - Creazione dell'oggetto MovieClipLoader ***myLoader***
  - Inizializzazione delle proprietà ***x***, ***y***, ***width*** e ***height***.
  - Registrazione del PBarLoader creato in myLoader affinchè questo notifichi gli eventi legati al caricamento

I METODI: CARICAMENTO

- Per caricare il contenuto esterno construirò un metodo con un unico parametro il nome del file da caricare:
  - ***loadClip(url: String)***.
- Il metodo:
  - Inizializzerà la barra di caricamento.
  - Utilizzerà myLoader per caricare il file

I METODI: LA BARRA

- Disegna lo sfondo della barra secondo il colore e le dimensioni definite e la posiziona al centro :
  - ***initBar()***.
- Disegna la barra nella dimensione che rappresenta lo stato del caricamento (il parametro rapresnta il rapporto tra byte scaricati e bytes totali:
  - ***writeBar(caricamento:Number)***;