EBook Gratuito

APPRENDIMENTO ActionScript 3

Free unaffiliated eBook created from
Stack Overflow contributors.

#actionscrip

t-3

Sommario

Di

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: actionscript-3

It is an unofficial and free ActionScript 3 ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official ActionScript 3.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capitolo 1: Introduzione a ActionScript 3

Osservazioni

ActionScript 3 è il linguaggio di programmazione per gli ambienti di runtime Adobe Flash Player e Adobe AIR. È un linguaggio basato su ECMAScript orientato agli oggetti utilizzato principalmente per lo sviluppo di applicazioni native su dispositivi desktop (Windows / Mac) e mobili (iOS / Android).

Risorse per l'apprendimento di Adobe: http://www.adobe.com/devnet/actionscript/learning.html

Storia e altri dettagli: https://en.wikipedia.org/wiki/ActionScript

Documentazione online su classi e riferimenti:
http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/package-detail.html

Versioni

Esiste una versione singola di Actionscript 3, denominata "ActionScript 3.0"

Versione Flash	Nome in codice	Cambiamenti e miglioramenti	Data di rilascio
Flash Player 9.x	Zaphod	Versione iniziale	2006-06-22
Flash Player 10.0	Astro	ha introdotto il tipo `Vector.<T>` , i filtri shader Adobe Pixel Bender nella classe `flash.filters.ShaderFilter` e il relativo supporto hardware su più CPU.	2008-10-15
Flash Player 10.1	Argo	ha introdotto la classe `flash.events.TouchEvent` per funzionare con dispositivi multitouch e altro supporto dell'hardware dei dispositivi mobili, come l'accelerometro.	2010-06-10
Flash Player 10.2	Speziato	ha introdotto la classe `flash.media.StageVideo` e il framework generale per lavorare con la riproduzione di video di scena in AS3.	2011-02-08
Flash Player 11	Serrano	aggiunge il supporto H.264 allo streaming video su oggetti `NetStream` in entrambe le direzioni. Inoltre aggiunge il supporto SSL / TLS per la connessione Flash con classe `SecureSocket` .	2011-10-04

Versione Flash	Nome in codice	Cambiamenti e miglioramenti	Data di rilascio
Flash Player 11.4	Brannan	ha introdotto la classe `flash.system.Worker` e la capacità di delegare il lavoro asincrono ad altri thread sul client.	2012-08-10
Flash Player 11.8	Harrison	supporto hardware rimosso (compilazione JIT) per i filtri shader Adobe Pixel Bender, riducendo drasticamente le prestazioni di qualsiasi esecuzione del filtro shader PB.	2013/05/09

Examples

Panoramica sull'installazione

ActionScript 3 può essere utilizzato installando Adobe AIR SDK o Apache Flex SDK o come parte del prodotto Animate CC di Adobe *(precedentemente noto come Flash Professional)* .

Adobe Animate CC è una soluzione software professionale che può essere utilizzata per creare progetti AS3 utilizzando strumenti visivi: una volta installati, non sono necessari ulteriori passaggi per iniziare a creare progetti AS3.

AIR SDK e Flex SDK possono essere utilizzati con strumenti da riga di comando o con vari IDE di terze parti.

Oltre ad Adobe Animate CC, ci sono altri quattro IDE popolari in grado di funzionare con AS3. Questi IDE hanno le proprie istruzioni su come iniziare.

- Flash Builder (di Adobe - basato su Eclipse)
- IntelliJ IDEA (By Jetbrains)
- FlashDevelop
- FDT (Eclipse Plugin)

Ciao mondo

Un esempio di classe di documento che stampa "Hello, World" sulla console di debug quando è istanziato.

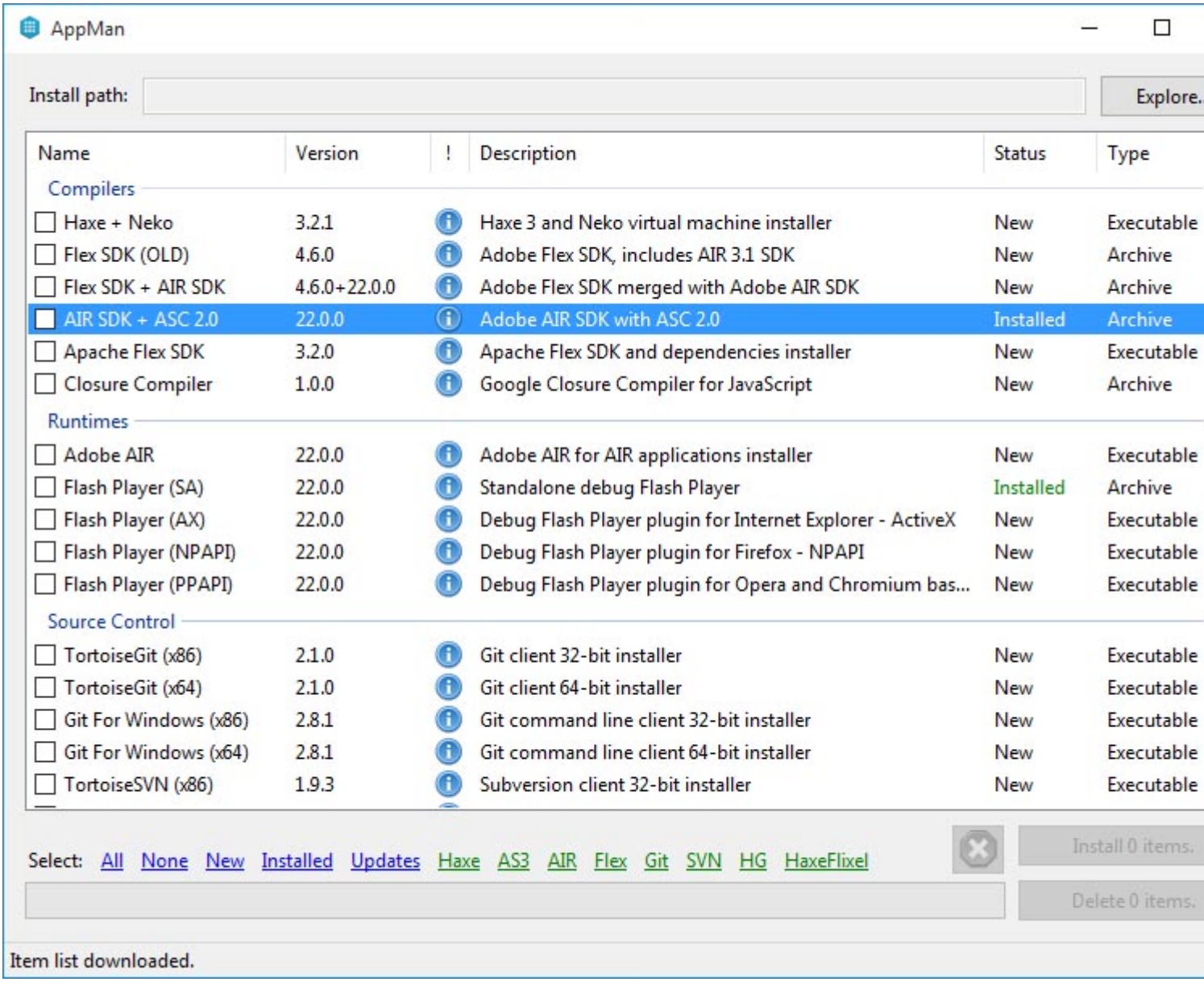
```
import flash.display.Sprite;

public class Main extends Sprite {

    public function Main() {
        super();

        trace("Hello, World");
    }

}
```

Installazione di Flash Develop

FlashDevelop è un IDE open source multipiattaforma creato nel 2005 per gli sviluppatori Flash. Senza costi, è un modo molto popolare per iniziare a sviluppare con AS3.

Per installare FlashDevelop:

1. Scarica il file di installazione ed esegui il programma di installazione
2. Al termine dell'installazione, eseguire FlashDevelop. Al primo avvio dovrebbe apparire la finestra `App Man` che chiede di scegliere quali SDK e strumenti installare.



Se AppMan non si apre automaticamente o se vuoi aggiungere qualcosa in seguito, aprilo selezionando 'Installa software' nel menu 'Strumenti'.

Controlla l'elemento **AIR SDK + ACS 2.0** (nella sezione "Compilatore") e **Flash Player (SA)** nella sezione "Runtimes" (più qualsiasi altra cosa desideri installare). Fai clic sul pulsante Installa.

3. Una volta installato l'SDK, testiamo creando un progetto Hello World. Inizia creando un nuovo progetto (dal menu *Progetto*)

4. Scegli **AIR AS3 Projector** dall'elenco e assegnagli un nome / posizione.

5. Nel pannello del project manager (scegli "Project Manager" dal menu di visualizzazione se non è già visibile), espandi la cartella **src** e apri il file `Main.as`

6. Nel file `Main.as` , ora puoi creare un primo programma di esempio come Hello World

7. Esegui il tuo progetto facendo clic sull'icona di riproduzione o premendo `F5` o `Ctrl+Enter` . Il progetto verrà compilato e al termine dovrebbe apparire una finestra vuota (questa è la tua applicazione). Nella finestra di output di FlashDevelop, dovresti vedere le parole: **Hello World** .

Ora sei pronto per iniziare a sviluppare applicazioni AS3 con FlashDevelop!

Installazione di Apache Flex

da http://flex.apache.org/doc-getstarted.html

1. Scarica il programma di installazione dell'SDK

2. Esegui il programma di installazione dell'SDK. La prima domanda che ti verrà posta è la directory di installazione.

    - su un Mac, utilizzare `/Applications/Adobe Flash Builder 4.7/sdks/4.14.0/`
    - su un PC, utilizzare `C:\Program Files(x86)\Adobe Flash Builder 4.7\sdks\4.14.0`

    Dovrai creare le cartelle 4.14.0. Premere Avanti. Accetta licenze SDK e installa.

Istruzioni specifiche IDE per l'installazione di Apache Flex:

- Flash Builder
- IntelliJ IDEA
- FlashDevelop
- FDT

Costruire progetti Flex o Flash sulla riga di comando usando mxmlc

Il compilatore Flex (`mxmlc`) è una delle parti più importanti di Flex SDK. Puoi modificare il codice AS3 in qualsiasi editor di testo che ti piace. Creare un file di classe principale che si estende da `DisplayObject` .

È possibile attivare build sulla riga di comando come segue:

```
mxmlc -source-path="." -default-size [width in pixels] [height in pixels] -default-frame-rate
[fps] -o "outputPath.swf" "mainClass.as"
```

Se hai bisogno di compilare un progetto Flash (al contrario di Flex) puoi aggiungere un riferimento alla libreria di Flash come segue (devi installare Adobe Animate IDE):

```
mxmlc -source-path="." -library-path+="/Applications/Adobe Animate CC 2015.2/Adobe Animate CC 2015.2.app/Contents/Common/Configuration/ActionScript 3.0/libs" -static-link-runtime-shared-libraries=true -default-size [width in pixels] [height in pixels] -default-frame-rate [fps] -o "outputPath.swf" "mainClass.as"
```

Oppure su Windows:

```
mxmlc -source-path="." -library-path+="C:\Program Files\Adobe\Adobe Animate CC 2015.2\Common\Configuration\ActionScript 3.0\libs" -static-link-runtime-shared-libraries=true -default-size [width in pixels] [height in pixels] -default-frame-rate [fps] -o "outputPath.swf" "mainClass.as"
```

Un esempio "Hello World" visualizzato

```
package {
    import flash.text.TextField;
    import flash.display.Sprite;

    public class TextHello extends Sprite {
        public function TextHello() {
            var tf:TextField = new TextField();
            tf.text = "Hello World!"
            tf.x = 50;
            tf.y = 40;
            addChild(tf);
        }
    }
}
```

Questa classe utilizza la classe `TextField` per visualizzare il testo.

Leggi Introduzione a ActionScript 3 online: https://riptutorial.com/it/actionscript-3/topic/1065/introduzione-a-actionscript-3

Capitolo 2: Caricamento di file esterni

Osservazioni

Vi sono alcuni casi in cui l'applicazione non può manipolare il contenuto delle risorse caricate da una risorsa esterna (ad esempio, trasformare le immagini). Non riesco a ricordare per certo quello che sono, ma sono abbastanza sicuro che sia correlato al caricamento di contenuti interdominio.

Examples

Caricamento di immagini / file SWF esterni con il caricatore

1. Crea un oggetto Loader:

```
var loader:Loader = new Loader();   //import
```

2. Aggiungi listener sul caricatore. Quelli standard sono completi e io / errori di sicurezza

```
loader.contentLoaderInfo.addEventListener(Event.COMPLETE, loadComplete); //when the
loader is done loading, call this function
loader.contentLoaderInfo.addEventListener(IOErrorEvent.IO_ERROR, loadIOError); //if the
file isn't found
loader.contentLoaderInfo.addEventListener(SecurityErrorEvent.SECURITY_ERROR,
loadSecurityError); //if the file isn't allowed to be loaded
```

3. Carica il file desiderato:

```
 loader.load(new URLRequest("image.png"));
```

4. Crea i tuoi gestori di eventi:

```
 function loadComplete(e:Event):void {
     //load complete
     //the loader is actually a display object itself, so you can just add it to the
display list
     addChild(loader)
     //or addChild(loader.content) to add the root content of what was loaded;
 }

 function loadIOError(e:IOErrorEvent):void {
     //the file failed to load,
 }

 function loadSecurityError(e:SecurityError):void {
     //the file wasn't allowed to load
 }
```

Il caricamento con la classe Loader è asincrono. Ciò significa che dopo aver chiamato `loader.load` l'applicazione continuerà a funzionare mentre il file viene caricato. Il

contenuto del caricatore non è disponibile fino a quando il caricatore non invia l'evento `Event.COMPLETE` .

---

importazioni necessarie:

```
import flash.display.Loader;
import flash.events.Event;
import flash.events.IOErrorEvent;
import flash.events.SecurityErrorEvent;
import flash.net.URLRequest;
```

Caricamento di un file di testo con FileStream (solo runtime AIR)

Un semplice esempio su come leggere un file di testo UTF in modo sincrono.

```
import flash.filesystem.File;
import flash.filesystem.FileMode;
import flash.filesystem.FileStream;
```

```
//First, get a reference to the file you want to load
var myFile:File = File.documentsDirectory.resolvePath("lifestory.txt");

//Create a FileStream object
fileStream = new FileStream();

//open the file
fileStream.open(myFile, FileMode.READ);

//read the data and assign it to a local variable
var fileText:String = fileStream.readUTF();

//close the current filestream
fileStream.close();
```

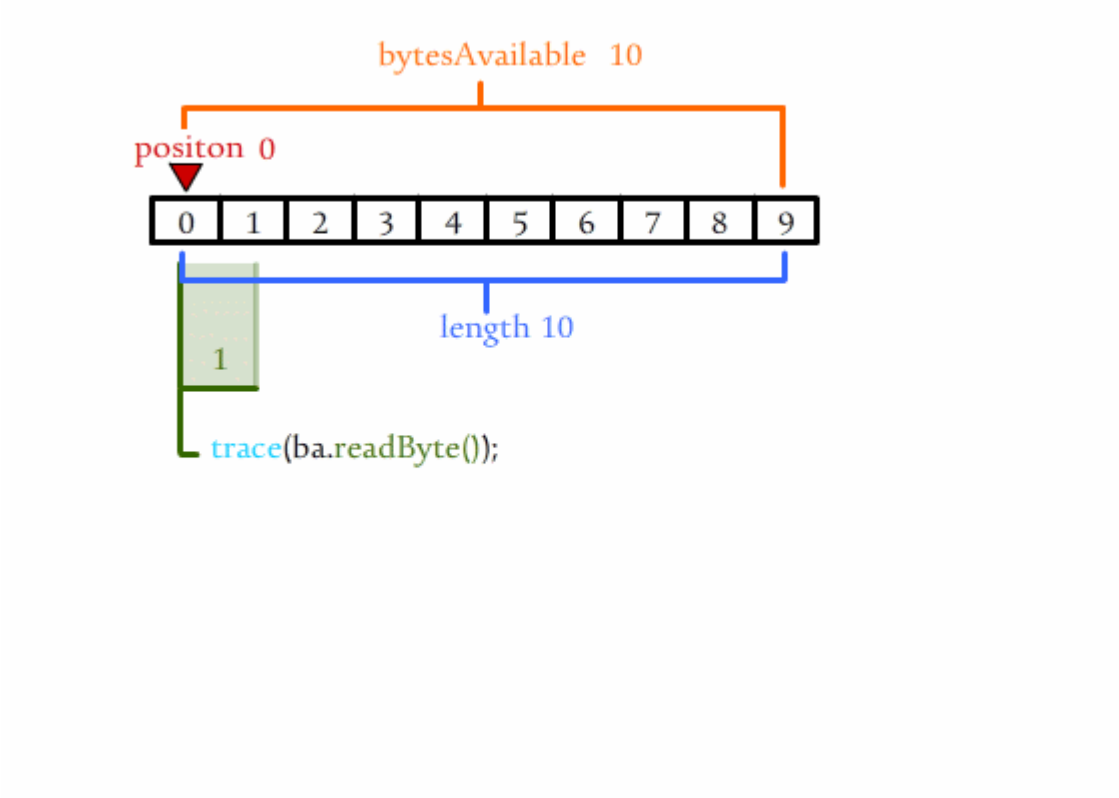
Leggi Caricamento di file esterni online: https://riptutorial.com/it/actionscript-3/topic/1694/caricamento-di-file-esterni

Capitolo 3: Dati binari

Examples

Lettura del modulo ByteArray attraverso l'interfaccia IDataInput.

L'animazione sotto mostra cosa succede quando usi i metodi di interfaccia IDataInput per accedere al modulo dati ByteArray e ad altre classi che implementano questa interfaccia.



Leggi Dati binari online: https://riptutorial.com/it/actionscript-3/topic/9464/dati-binari

Capitolo 4: Disegno di bitmap

Examples

Disegna un oggetto di visualizzazione in dati bitmap

Una funzione di supporto per creare una copia bitmap di un oggetto. Questo può essere usato per convertire oggetti vettoriali, testo o sprite nidificati complessi in una bitmap appiattita.

```
function makeBitmapCopy(displayObj:IBitmapDrawable, transparent:Boolean = false, bgColor:uint
= 0x00000000, smooth:Boolean = true):Bitmap {

    //create an empty bitmap data that matches the width and height of the object you wish to
draw
    var bmd:BitmapData = new BitmapData(displayObj.width, displayObj.height, transparent,
bgColor);

    //draw the object to the bitmap data
    bmd.draw(displayObj, null, null, null, null, smooth);

    //assign that bitmap data to a bitmap object
    var bmp:Bitmap = new Bitmap(bmd, "auto", smooth);

    return bmp;
}
```

Uso:

```
var txt:TextField = new TextField();
txt.text = "Hello There";

var bitmap:Bitmap = makeBitmapCopy(txt, true); //second param true to keep transparency
addChild(bitmap);
```

Disegna un oggetto di visualizzazione con qualsiasi coordinate del punto di registrazione

```
    public function drawDisplayObjectUsingBounds(source:DisplayObject):BitmapData {
        var bitmapData:BitmapData;//declare a BitmapData
        var bounds:Rectangle = source.getBounds(source);//get the source object actual size
        //round bounds to integer pixel values (to aviod 1px stripes left off)
        bounds = new Rectangle(Math.floor(bounds.x), Math.floor(bounds.y),
Math.ceil(bounds.width), Math.ceil(bounds.height));

        //to avoid Invalid BitmapData error which occures if width or height is 0
        //(ArgumentError: Error #2015)
        if((bounds.width>0) && (bounds.height>0)){
            //create a BitmapData
            bitmapData = new BitmapData(bounds.width, bounds.height, true, 0x00000000);

            var matrix:Matrix = new Matrix();//create a transform matrix
            //translate if to fit the upper-left corner of the source
```

```
            matrix.translate(-bounds.x, -bounds.y);
            bitmapData.draw(source, matrix);//draw the source
            return bitmapData;//return the result (exit point)
        }
        //if no result is created - return an empty BitmapData
        return new BitmapData(1, 1, true, 0x00000000);
    }
```

Una nota a `getBounds()` : per `getBounds()` per restituire valori validi l'oggetto deve avere accesso allo stage almeno una volta, altrimenti i valori sono falsi. È possibile aggiungere un codice per assicurarsi che la `source` passata abbia una fase e, in caso contrario, può essere aggiunta allo stage e quindi rimossa nuovamente.

Animazione di un foglio sprite

Un foglio sprite per definizione è una bitmap che contiene una certa animazione. I vecchi giochi usano il foglio sprite di tipo griglia, cioè ogni frame occupa una regione uguale, ei frame sono allineati dai bordi per formare un rettangolo, probabilmente con alcuni spazi non occupati. Successivamente, al fine di ridurre al minimo la dimensione della bitmap, i fogli sprite iniziano ad essere "impacchettati" rimuovendo spazi bianchi extra attorno al rettangolo che contiene ciascun fotogramma, ma ogni fotogramma è ancora un rettangolo per semplificare le operazioni di copia.

Per animare un foglio sprite, è possibile utilizzare due tecniche. Innanzitutto, è possibile utilizzare `BitmapData.copyPixels()` per copiare una determinata area del proprio foglio sprite su una `Bitmap` visualizzata, producendo un personaggio animato. Questo approccio è migliore se si utilizza una singola `Bitmap` visualizzata che ospita l'intera immagine.

```
var spriteSheet:BitmapData;
var frames:Vector.<Rectangle>; // regions of spriteSheet that represent frames
function displayFrameAt(frame:int,buffer:BitmapData,position:Point):void {
    buffer.copyPixels(spriteSheet,frames[frame],position,null,null,true);
}
```

La seconda tecnica può essere utilizzata se nella lista di visualizzazione sono presenti molti `Sprite` o `Bitmap` e condividono lo stesso foglio sprite. Qui, il buffer di destinazione non è più un singolo oggetto, ma ogni oggetto ha il proprio buffer, quindi la strategia corretta è di manipolare la proprietà `bitmapData` delle bitmap. Prima di farlo, tuttavia, il foglio sprite dovrebbe essere diviso in riquadri singoli.

```
public class Stuff extends Bitmap {
    static var spriteSheet:Vector.<BitmapData>;
    function displayFrame(frame:int) {
        this.bitmapData=spriteSheet[frame];
    }
    // ...
}
```

Leggi Disegno di bitmap online: https://riptutorial.com/it/actionscript-3/topic/2814/disegno-di-bitmap

Capitolo 5: Generazione di valore casuale

Examples

Numero casuale compreso tra 0 e 1

```
Math.random();
```

produce un numero casuale uniformemente distribuito tra 0 (incluso) e 1 (esclusivo)

Esempio di output:

- ,22282187035307288
- ,3948539895936847
- ,9987191134132445

Numero casuale tra valori minimi e massimi

```
function randomMinMax(min:Number, max:Number):Number {
    return (min + (Math.random() * Math.abs(max - min)));
}
```

Questa funzione viene chiamata passando un intervallo di valori minimi e massimi.

Esempio:

```
randomMinMax(1, 10);
```

Risultati di esempio:

- 1,661770915146917
- 2,5521070677787066
- 9,436270965728909

Angolo casuale, in gradi

```
function randomAngle():Number {
    return (Math.random() * 360);
}
```

Risultati di esempio:

- 31,554428357630968
- 230,4078639484942
- 312,7964010089636

Valore casuale da una matrice

Supponendo che abbiamo un array `myArray` :

```
var value:* = myArray[int(Math.random() * myArray.length)];
```

Nota che usiamo `int` per `2.4539543` il risultato di `Math.random()` a un int perché valori come `2.4539543` non sarebbero un indice di array valido.

Punto casuale all'interno di un cerchio

Prima definisci il raggio del cerchio e il suo centro:

```
var radius:Number = 100;
var center:Point = new Point(35, 70);
```

Quindi genera un angolo casuale in *radianti* dal centro:

```
var angle:Number = Math.random() * Math.PI * 2;
```

Quindi genera un raggio efficace del punto restituito, quindi sarà all'interno del `radius` dato. Un semplice `Math.random()*radius` non funzionerà, perché con questa distribuzione i punti prodotti finiranno nel cerchio interno di mezzo raggio metà del tempo, ma il quadrato di quel cerchio è un quarto dell'originale. Per creare una corretta distribuzione, la funzione dovrebbe essere così:

```
var rad:Number=(Math.random()+Math.random())*radius; // yes, two separate calls to random
if (rad>radius) { rad=2*radius-rad; }
```

Questa funzione produce un valore che ha la sua funzione di probabilità che aumenta linearmente da 0 a zero al massimo al `radius` . Succede perché una somma di valori casuali ha una funzione di densità di probabilità uguale alla convoluzione di tutte le funzioni di densità individuale dei valori casuali. Questa è una matematica estesa per una persona di grado medio, ma una GIF gentile viene presentata per tracciare un grafico della funzione di convoluzione di due funzioni di densità di distribuzione uniformi spiegate come " segnali di casella ". L'operatore `if` ripiega la funzione risultante sul suo massimo, lasciando solo un grafico a forma di dente di sega.

Questa funzione è selezionata perché il quadrato di una striscia circolare situata tra `radius=r` e `radius=r+dr` aumenta linearmente con `r` crescente e costante `dr` molto piccolo in modo che `dr*dr<<r` . Pertanto, la quantità di punti generati vicino al centro è minore della quantità di punti generati sul bordo del cerchio dello stesso margine in cui il raggio dell'area centrale è inferiore al raggio dell'intero cerchio. Quindi, in generale, i punti sono equamente distribuiti in tutto il cerchio.

Ora, prendi la tua posizione casuale:

```
var result:Point = new Point(
    center.x + Math.cos(angle) * rad,
    center.y + Math.sin(angle) * rad
);
```

Per ottenere un punto casuale sul cerchio (sul bordo del cerchio di un dato raggio), utilizzare il `radius` anziché il `rad` .

PS: l'esempio è stato sovraccaricato dalla spiegazione della matematica.

Angolo casuale, in radianti

```
function randomAngleRadians():Number
{
    return Math.random() * Math.PI * 2;
}
```

Risultati di esempio:

- 5,490068569213088
- 3,1984284719180205
- 4,581117863808207

Determinazione del successo di un'operazione "percentuale di possibilità"

Se è necessario eseguire il roll su `true` o `false` in una situazione di "x% chance", utilizzare:

```
function roll(chance:Number):Boolean {
    return Math.random() >= chance;
}
```

Usato come:

```
var success:Boolean = roll(0.5); // True 50% of the time.
var again:Boolean = roll(0.25); // True 25% of the time.
```

Crea un colore casuale

Per ottenere *un* colore casuale:

```
function randomColor():uint
{
    return Math.random() * 0xFFFFFF;
}
```

Se hai bisogno di più controllo sui canali rosso, verde e blu:

```
var r:uint = Math.random() * 0xFF;
var g:uint = Math.random() * 0xFF;
var b:uint = Math.random() * 0xFF;

var color:uint = r << 16 | g << 8 | b;
```

Qui è possibile specificare un intervallo per `r` , `g` e `b` (questo esempio è 0-255).

Passa in modo casuale attraverso l'alfabeto

```
var alphabet:Vector.<String> = new <String>[ "A", "B", "C", "D", "E", "F", "G",
                                             "H", "I", "J", "K", "L", "M", "N",
                                             "O", "P", "Q", "R", "S", "T", "U",
                                             "V", "W", "X", "Y", "Z" ];

while (alphabet.length > 0)
{
    var letter:String = alphabet.splice(int(Math.random() *
                                            alphabet.length), 1)[0];
    trace(letter);
}
```

Esempio di output:

V, M, F, E, D, U, S, L, X, K, Q, H, A, I, W, N, P, Y, J, C, T, O, R, G, B, Z

Randomize An Array

```
var alphabet:Array = [ "A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N",
"O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z" ];

for (var i:int=alphabet.length-1;i>0;i--) {
    var j:int=Math.floor(Math.random()*(i+1));
    var swap=alphabet[j];
    alphabet[j]=alphabet[i];
    alphabet[i]=swap;
}
trace(alphabet);
```

Esempio di output

B, Z, D, R, U, N, O, M, I, L, C, J, P, H, W, S, D, E, K, T, F, V, X, Y, G, UN

Questo metodo è noto come array shauffle Fisher-Yates .

Leggi Generazione di valore casuale online: https://riptutorial.com/it/actionscript-3/topic/1441/generazione-di-valore-casuale

Capitolo 6: Informazioni su "Errore 1009: impossibile accedere a una proprietà oa un metodo di riferimento a un oggetto nullo"

introduzione

Un errore 1009 è un errore generale che si verifica quando si tenta di ricevere un valore da una variabile o una proprietà con valore `null` . Gli esempi forniti illustrano vari casi in cui si verifica questo errore, insieme ad alcune raccomandazioni su come mitigare l'errore.

Osservazioni

Il temuto e spesso chiesto "Errore 1009: impossibile accedere a una proprietà o metodo di un riferimento a oggetto nullo" è un segnale che alcuni dati appaiono nulli, ma viene tentato di essere usato come oggetto popolato. Esistono molti tipi di problemi che possono causare questo comportamento e ognuno deve essere verificato rispetto al codice in cui si è verificato l'errore.

Examples

Stage non disponibile

A volte gli sviluppatori scrivono un codice che desidera accedere allo `stage` o allo `stage` Flash per aggiungere ascoltatori. Può funzionare per la prima volta, quindi all'improvviso non funziona e genera l'errore 1009. Il codice in questione può essere anche sulla timeline, poiché è la prima iniziativa ad aggiungere codice e molti tutorial ancora esistenti strato di codice temporale per posizionare il codice.

```
public class Main extends MovieClip {
    public function Main() {
        stage.addEventListener(Event.ENTER_FRAME,update); // here
```

Il motivo per cui questo codice non funziona è semplice: un oggetto di visualizzazione viene prima istanziato, quindi aggiunto all'elenco di visualizzazione e mentre è fuori dall'elenco di visualizzazione, lo `stage` è nullo.

Peggio ancora se il codice come questo:

```
stage.addEventListener(Event.ENTER_FRAME,update); // here
```

è posto sulla timeline. Può anche funzionare per un po 'di tempo, mentre l'oggetto `Main` viene schiaffeggiato sullo stage tramite la GUI. Quindi, il loro SWF viene caricato da un altro SWF e, all'improvviso, il codice si interrompe. Ciò accade perché i frame del `Main` sono costruiti in modo

diverso quando il file SWF viene caricato direttamente dal player e quando il caricamento viene elaborato in modo asincrono. La soluzione è usare il listener `Event.ADDED_TO_STAGE` e mettere tutto il codice che indirizzi stage in esso, e mettere il listener stesso in un file AS invece della timeline.

Typecast non valido

```
function listener(e:Event):void {
    var m:MovieClip=e.target as MovieClip;
    m.x++;
}
```

Se tale listener è collegato a un oggetto che non è un discendente `MovieClip` (ad esempio, uno `Sprite`), il typecast fallirà e qualsiasi successiva operazione con il suo risultato genererà l'errore 1009.

Oggetto non istintivo

```
var a:Object;
trace(a); // null
trace(a.b); // Error 1009
```

Qui viene dichiarato un riferimento a un oggetto, ma non viene mai assegnato un valore, sia esso con `new` o l'assegnazione di un valore non nullo. Richiedere le sue proprietà o il metodo genera un errore 1009.

Espressione a più livelli

```
x=anObject.aProperty.anotherProperty.getSomething().data;
```

Qui, qualsiasi oggetto prima del punto può risultare null, e l'utilizzo di metodi che restituiscono oggetti complessi aumenta solo la complicazione di eseguire il debug dell'errore nullo. Caso peggiore se il metodo è soggetto a guasti estranei, ad esempio il recupero dei dati attraverso la rete.

Risultato della funzione non elaborata

```
s=this.getChildByName("garbage");
if (s.parent==this) {...}
```

`getChildByName()` è una delle molte funzioni che può restituire null se si è verificato un errore durante l'elaborazione del suo input. Pertanto, se si riceve un oggetto da qualsiasi funzione che può eventualmente restituire null, verificare prima null. Qui, una proprietà viene interrogata istantaneamente senza prima verificare se `s` è nullo, questo genererà l'errore 1009.

Ascoltatore di eventi dimenticato

```
addEventListener(Event.ENTER_FRAME,moveChild);
```

```
function moveChild(e:Event):void {
    childMC.x++;
    if (childMC.x>1000) {
        gotoAndStop(2);
    }
}
```

Questo esempio sposterà il `childMC` (aggiunto a `Main` in fase di progettazione) ma getterà istantaneamente un 1009 non appena `gotoAndStop()` viene invocato, se quel `childMC` non esiste sul frame 2. Il motivo principale per questo è che ogni volta che un playhead passa un fotogramma chiave (un fotogramma che non eredita il set di oggetti del fotogramma precedente), utilizzando `gotoAndStop()` , `gotoAndPlay()` con il fotogramma di destinazione separato dal fotogramma corrente da un fotogramma chiave o dalla riproduzione normale se il file SWF è un animazione, i contenuti del frame corrente vengono **distrutti** e i nuovi contenuti vengono creati utilizzando i dati memorizzati dalla GUI. Quindi, se il nuovo frame non ha un figlio chiamato `childMC` , la richiesta di proprietà restituirà null e verrà lanciato 1009.

Lo stesso principio si applica se aggiungi due listener di eventi, ma ne rimuovi solo uno o aggiungi un listener a un oggetto, ma prova a rimuoverlo da un altro. La chiamata `removeEventListener` non ti avviserà se l'oggetto non ha un rispettivo listener di eventi collegato, quindi leggi il codice che aggiunge e rimuove attentamente i listener di eventi.

Nota anche: Usando gli oggetti `Timer` , la chiamata a `setInterval()` e `setTimeout()` crea anche listener di eventi, e anche questi dovrebbero essere cancellati correttamente.

Riferimento invalidato a un oggetto basato su frame

A volte `gotoAndStop()` viene chiamato nel mezzo del codice che fa riferimento ad alcune proprietà basate su frame. Ma, **subito dopo la modifica del frame,** tutti i collegamenti alle proprietà esistenti sul frame corrente vengono invalidati, quindi qualsiasi elaborazione che li coinvolge deve essere immediatamente interrotta.

Esistono due scenari generali di tale elaborazione: Primo, un ciclo non termina dopo la chiamata a `gotoAndStop()` , come qui:

```
for each (bullet in bullets) {
    if (player.hitTestObject(bullet)) gotoAndStop("gameOver");
}
```

Qui, un errore 1009 significa che il `player` MC è stato distrutto durante l'elaborazione della chiamata `gotoAndStop()` , ma il ciclo continua, e fa riferimento al link now-null per ottenere `hitTestObject()` da. Se la condizione dovesse dire `if (bullet.hitTestObject(player))` invece, l'errore sarebbe # 2007 "Il parametro hitTestObject non deve essere nullo". La soluzione è inserire un'istruzione `return` dopo aver chiamato `gotoAndStop()` .

Il secondo caso riguarda più listener di eventi sullo stesso evento. Come questo:

```
stage.addEventListener(Event.ENTER_FRAME,func1);
stage.addEventListener(Event.ENTER_FRAME,func2);
```

```
function func1(e:Event):void {
    if (condition()) {
        gotoAndStop(2);
    }
}
```

Qui, se `condition()` è true, il primo listener eseguirà `gotoAndStop()` , ma il secondo listener verrebbe comunque eseguito e, se questo fa riferimento a oggetti sul frame, verrà generato un errore 1009. La soluzione è evitare più ascoltatori su un singolo evento, in un singolo oggetto, è meglio avere un ascoltatore che gestisca tutte le situazioni su quell'evento e che possa terminare correttamente se è necessario un cambio di frame.

Leggi Informazioni su "Errore 1009: impossibile accedere a una proprietà oa un metodo di riferimento a un oggetto nullo" online: https://riptutorial.com/it/actionscript-3/topic/2098/informazioni-su--errore-1009--impossibile-accedere-a-una-proprieta-oa-un-metodo-di-riferimento-a-un-oggetto-nullo-

Capitolo 7: Invio e ricezione di dati dai server

Examples

Fare una richiesta da Flash

Le classi `URLRequest` e `URLLoader` per rendere richieste da Flash a risorse esterne. `URLRequest` definisce le informazioni sulla richiesta, ad esempio il corpo della richiesta e il tipo di metodo di richiesta, e l' `URLLoader` riferimento a questa per eseguire la richiesta effettiva e fornire un mezzo per ricevere una notifica quando viene ricevuta una risposta dalla risorsa.

Esempio:

```
var request:URLRequest = new URLRequest('http://stackoverflow.com');
var loader:URLLoader = new URLLoader();

loader.addEventListener(Event.COMPLETE, responseReceived);
loader.load(request);

function responseReceived(event:Event):void {
    trace(event.target.data); // or loader.data if you have reference to it in
                              // this scope.
}
```

Aggiunta di variabili alla tua richiesta

La classe `URLVariables` consente di definire i dati da inviare insieme a un `URLRequest` .

Esempio:

```
var variables:URLVariables = new URLVariables();

variables.prop = "hello";
variables.anotherProp = 10;

var request:URLRequest = new URLRequest('http://someservice.com');
request.data = variables;
```

È possibile inviare la richiesta tramite `URLLoader` o aprire l'URL della richiesta con le variabili allegate nella query querys utilizzando `navigateToURL` .

Modifica del metodo HTTP (GET, POST, PUT, ecc.)

La classe `URLRequestMethod` contiene costanti per i vari tipi di richieste che è possibile creare. Queste costanti devono essere allocate alla proprietà del `method` `URLRequest` :

```
var request:URLRequest = new URLRequest('http://someservice.com');
request.method = URLRequestMethod.POST;
```

Si noti che solo `GET` e `POST` sono disponibili al di fuori del runtime AIR.

I miei dati di risposta sono sempre nulli, cosa significa "asincrono"?

Quando Flash inoltra una richiesta di dati da un'origine esterna, quell'operazione è *asincrona* . La spiegazione più semplice di ciò che significa è che i dati vengono caricati "in background" e attivano il gestore eventi che si assegna a `Event.COMPLETE` quando viene ricevuto. Questo può accadere in qualsiasi momento della vita della tua applicazione.

I tuoi dati **NON** saranno disponibili immediatamente dopo aver chiamato `load()` sul tuo `URLLoader` . È **necessario** collegare un listener di eventi per `Event.COMPLETE` e interagire con la risposta c'è.

```
var request:URLRequest = new URLRequest('http://someservice.com');
var loader:URLLoader = new URLLoader();

loader.addEventListener(Event.COMPLETE, responseReceived);
loader.load(request);

trace(loader.data); // Will be null.

function responseReceived(event:Event):void {
    trace(loader.data); // Will be populated with the server response.
}

trace(loader.data); // Will still be null.
```

Non puoi aggirare questo con qualche piccolo trucco come usare `setTimeout` o simili:

```
setTimeout(function() {
    trace(loader.data); // Will be null if the data hasn't finished loading
                        // after 1000ms (which you can't guarantee).
}, 1000);
```

Richieste interdominio

Flash non caricherà i dati da un dominio diverso da quello su cui è in esecuzione l'applicazione, a meno che tale dominio non abbia una politica di dominio XML nella radice del dominio (ad esempio `http://somedomain.com/crossdomain.xml`) o da qualche parte può targetizzare con `Security.loadPolicyFile()` . Il file crossdomain.xml è dove puoi specificare domini che sono in grado di chiedere dati al tuo server da un'applicazione Flash.

Esempio del crossdomain.xml *più permissivo* :

```
<?xml version="1.0" ?>
<cross-domain-policy>
  <allow-access-from domain="*"/>
  <allow-http-request-headers-from domain="*" headers="*"/>
</cross-domain-policy>
```

Nota che questo esempio **non deve essere usato negli ambienti di produzione** , usa un'istanza più restrittiva.

Ad esempio, un crossdomain.xml specifico più restrittivo sarà simile a questo:

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM "http://www.macromedia.com/xml/dtds/cross-domain-
policy.dtd">
<cross-domain-policy>
    <site-control permitted-cross-domain-policies="master-only" />

    <allow-access-from domain="*.domain.com" to-ports="80,843,8011" />
    <allow-access-from domain="123.123.123.123" to-ports="80,843,8011" />
</cross-domain-policy>
```

risorse:

- La specifica del file di criteri crossdomain .

Leggi Invio e ricezione di dati dai server online: https://riptutorial.com/it/actionscript-3/topic/1893/invio-e-ricezione-di-dati-dai-server

Capitolo 8: Lavorare con data e ora

Examples

Ante meridiem (AM) o Post meridiem (PM) per 12 ore

```
function meridiem(d:Date):String {
    return (d.hours > 11) ? "pm" : "am";
}
```

Numero di giorni nel mese specificato

```
/**
 * @param year   Full year as int (ex: 2000).
 * @param month  Month as int, zero-based (ex: 0=January, 11=December).
 */
function daysInMonth(year:int, month:int):int {
    return (new Date(year, ++month, 0)).date;
}
```

Se l'anno specificato è bisestile

```
function isLeapYear(year:int):Boolean {
    return daysInMonth(year, 1) == 29;
}
```

Se l'ora legale è attualmente osservata

```
function isDaylightSavings(d:Date):Boolean {
    var months:uint = 12;
    var offset:uint = d.timezoneOffset;
    var offsetCheck:Number;

    while (months--) {
        offsetCheck = (new Date(d.getFullYear(), months, 1)).timezoneOffset;

        if (offsetCheck != offset)
            return (offsetCheck > offset);
    }

    return false;
}
```

Data di inizio di oggi, a mezzanotte

```
function today(date:Date = null):Date {
    if (date == null)
        date = new Date();
```

```
    return new Date(date.fullYear, date.month, date.date, 0, 0, 0, 0);
}
```

Leggi Lavorare con data e ora online: https://riptutorial.com/it/actionscript-3/topic/1926/lavorare-con-data-e-ora

Capitolo 9: Lavorare con gli eventi

Osservazioni

Gli eventi sono pezzi di dati che un programma può creare, scambiare e reagire. Il flusso di eventi asincroni viene inviato nell'elenco di visualizzazione da parte di Flash Engine come reazione a eventi esterni, come i movimenti del mouse o un altro frame visualizzato. Ogni altro flusso di eventi e l'elaborazione di tutti gli eventi sono sincroni, quindi se un pezzo di codice ha generato un evento, tutte le reazioni su di esso vengono elaborate prima dell'esecuzione della riga successiva, anche se ci sono più ascoltatori di un evento, tutti avrebbe funzionato prima che potesse essere elaborato il prossimo evento.

Ci sono molti eventi importanti associati alla programmazione Flash. `Event.ENTER_FRAME` viene generato prima che Flash disegna un altro frame, segnali l'intero elenco di visualizzazione per prepararsi a essere disegnato e può essere utilizzato come timer sincrono. `MouseEvent.CLICK` e i suoi fratelli possono essere utilizzati per ricevere input del mouse dall'utente e `TouchEvent.TOUCH_TAP` è un analogo per i touch screen. `KeyboardEvent.KEY_DOWN` e `KEY_UP` forniscono mezzi per ricevere input dell'utente dalla tastiera, tuttavia, il loro utilizzo nel reparto mobile è quasi impossibile a causa di dispositivi che non dispongono di tastiera fisica. Infine, `Event.ADDED_TO_STAGE` viene inviato una volta che un oggetto di visualizzazione riceve l'accesso allo stage ed è incluso nell'elenco di visualizzazione globale che riceve l'intera serie di eventi che possono saltare su e giù nell'elenco di visualizzazione.

La maggior parte degli eventi in Flash sono specifici dei componenti. Se stai progettando il tuo componente che userà eventi Flash, usa una classe discendente `flash.events.Event` e le sue proprietà `String` statiche per creare il set di eventi del componente.

Examples

Eventi personalizzati con dati di eventi

```
package
{
    import flash.events.Event;

    public class CustomEvent extends Event
    {
        public static const START:String = "START";
        public static const STOP:String  = "STOP";

        public var data:*;

        public function CustomEvent(type:String, data:*,
                                    bubbles:Boolean=false, cancelable:Boolean=false)
        {
            super(type, bubbles, cancelable);

            if (data)
```

```
                this.data = data;
        }
    }
}
```

Per inviare un evento personalizzato:

```
var dataObject:Object = {name: "Example Data"};

dispatchEvent(new CustomEvent(CustomEvent.START, dataObject))
```

Per ascoltare eventi personalizzati:

```
addEventListener(CustomEvent.STOP, stopHandler);

function stopHandler(event:CustomEvent):void
{
    var dataObject:* = event.data;
}
```

Mancata gestione degli eventi dall'elenco di visualizzazione

```
package {
import flash.events.EventDispatcher;

public class AbstractDispatcher extends EventDispatcher {

    public function AbstractDispatcher(target:IEventDispatcher = null) {
        super(target);
    }

}
}
```

Per inviare un evento su un'istanza:

```
var dispatcher:AbstractDispatcher = new AbstractDispatcher();
dispatcher.dispatchEvent(new Event(Event.CHANGE));
```

Per ascoltare gli eventi su un'istanza:

```
var dispatcher:AbstractDispatcher = new AbstractDispatcher();
dispatcher.addEventListener(Event.CHANGE, changeHandler);

function changeHandler(event:Event):void
{
}
```

Gestione degli eventi di base

Flash invia **`Events`** per la maggior parte dei suoi oggetti. Uno degli eventi più elementari è **`ENTER_FRAME`** , che viene inviato (al framerate del file SWF) su ogni oggetto della lista di

visualizzazione.

```
import flash.display.Sprite;
import flash.events.Event;

var s:Sprite = new Sprite();
s.addEventListener(Event.ENTER_FRAME, onEnterFrame);

function onEnterFrame(e:Event)
{
    trace("I am called on every frame !");
}
```

Questa funzione sarà chiamata in modo asincrono su ogni frame. Ciò significa che la funzione che si assegna come `onEnterFrame` gestore di eventi viene elaborato prima di qualsiasi altro codice ActionScript che è collegato ai fotogrammi interessati.

Aggiungi i tuoi eventi

Puoi creare i tuoi eventi e inviarli, estendendo la classe `Event` .

```
import flash.events.Event;

class MyEvent extends Event
{
    var data: String;

    static public var MY_EVENT_TYPE = "my_event_my_event_code";

    public function MyEvent(type: String, data: String)
    {
        this.data = data;
    }

    override public function clone():Event
    {
        return new MyEvent(type, data);
    }
}
```

È quindi possibile inviarlo e ascoltarlo, utilizzando un `EventDispatcher` . Si noti che la maggior parte degli oggetti flash sono dispatcher di eventi.

```
import flash.events.EventDispatcher;

var d = new EventDispatcher();
d.addEventListener(MyEvent.MY_EVENT_TYPE, onType);

function onType(e: MyEvent)
{
    trace("I have a string: "+e.data);
}

d.dispatchEvent(new MyEvent(MyEvent.MY_EVENT_TYPE, "Hello events!"));
```

Nota che il metodo `clone` è richiesto se vuoi ridispatchare il tuo evento.

Struttura di eventi mouse semplice

Attraverso l'uso di `event types` di `event types` è possibile ridurre facilmente il bloat del codice che si verifica spesso quando si definiscono eventi per molti oggetti sul palco filtrando gli eventi in 1 funzione anziché definire molte funzioni di gestione degli eventi.

Immagina di avere 10 oggetti sul palco chiamati `object1` , `object2` ... `object10`

Potresti fare quanto segue:

```
var i: int = 1;
while(getChildByName("object"+i) != null){
    var obj = getChildByName("object"+i)
    obj.addEventListener(MouseEvent.CLICK, ObjectMouseEventHandler);
    obj.addEventListener(MouseEvent.MOUSE_OVER, ObjectMouseEventHandler);
    obj.addEventListener(MouseEvent.MOUSE_OUT, ObjectMouseEventHandler);
    obj.alpha = 0.75;
    i++;
}

function ObjectMouseEventHandler(evt:Event)
{
    if(evt.type == "click")
    {
        trace(evt.currentTarget + " has been clicked");
    }
    else
    {
        evt.currentTarget.alpha = evt.type == "mouseOver" ? 1 : 0.75;
    }
}
```

I vantaggi di questo metodo includono:

1. Non è necessario specificare la quantità di oggetti su cui applicare gli eventi.
2. Non ha bisogno di sapere in modo specifico quale oggetto è stato interagito con ancora applica ancora funzionalità.
3. Applicando facilmente eventi in blocco.

Leggi Lavorare con gli eventi online: https://riptutorial.com/it/actionscript-3/topic/1925/lavorare-con-gli-eventi

Capitolo 10: Lavorare con gli oggetti di visualizzazione

Sintassi

1. `addChild(child)` - aggiunge un nuovo elemento all'albero figlio di questo oggetto come elemento più in alto.
2. `addChildAt(child, index)` - aggiunge un nuovo elemento all'albero figlio di questo oggetto in una posizione specificata. L'elemento in basso ha indice 0.
3. `getChildAt(index)` - restituisce un child con un determinato indice.
4. `getChildIndex(child)` restituisce l'indice di un figlio *diretto* di questo oggetto. Altrimenti viene generata un'eccezione.
5. `removeChild(child)` - rimuove il figlio diretto specificato dall'albero figlio di questo oggetto. Genera un'eccezione se il genitore del figlio fornito non è uguale a `this` .
6. `removeChildAt(index)` - rimuove un figlio selezionato per indice invece di riferimento. Genera un'eccezione se l'albero figlio non è così largo.
7. `removeChildren(beginIndex:int = 0, endIndex:int = 0x7fffffff))` : aggiunto in Flash Player 11, rimuove un sottoinsieme di elementi `removeChildren(beginIndex:int = 0, endIndex:int = 0x7fffffff))` per intervallo di indice o tutti i child se viene chiamato senza parametri.
8. `setChildIndex(child,index)` - cambia l'indice del bambino nel nuovo valore, spostando tutti i bambini in mezzo per occupare il punto rilasciato.
9. `swapChildren(child1,child2)` - scambia le posizioni dei due bambini nell'elenco di visualizzazione, senza influenzare le posizioni di altri bambini.
10. `swapChildrenAt(index1,index2)` - scambia i bambini che si trovano nei loro indici.

Osservazioni

L'elenco di visualizzazione è in realtà un albero e viene visualizzato con il primo algoritmo di profondità. Qualsiasi oggetto elencato in precedenza verrà visualizzato in precedenza e potrebbe essere oscurato da oggetti elencati in seguito. Tutte le tecniche che possono essere utilizzate contro un albero possono essere applicate a lavorare con l'elenco di visualizzazione.

Examples

Introduzione alla lista di visualizzazione

In AS3, le risorse di visualizzazione non sono visibili finché non vengono aggiunte all'elenco di visualizzazione.

Il runtime AIR / Flash ha una struttura di visualizzazione gerarchica (relazione figlio genitore in cui i bambini possono avere figli propri), con lo `stage` come genitore di livello superiore.

Per aggiungere qualcosa all'elenco di visualizzazione, si usi `addChild` o `addChildAt` . Ecco un

esempio di base per disegnare un cerchio e aggiungerlo all'elenco di visualizzazione:

```
var myCircle:Shape = new Shape();

myCircle.graphics.beginFill(0xFF0000); //red
myCircle.graphics.drawCircle(25, 25, 50);
myCircle.graphics.endFill();

this.addChild(myCircle); //add the circle as a child of `this`
```

Per vedere l'oggetto nell'esempio sopra, anche `this` (il contesto del codice) deve essere presente nell'elenco di visualizzazione, così come i genitori che potrebbe avere. In AS3, il `stage` è il più alto dei genitori.

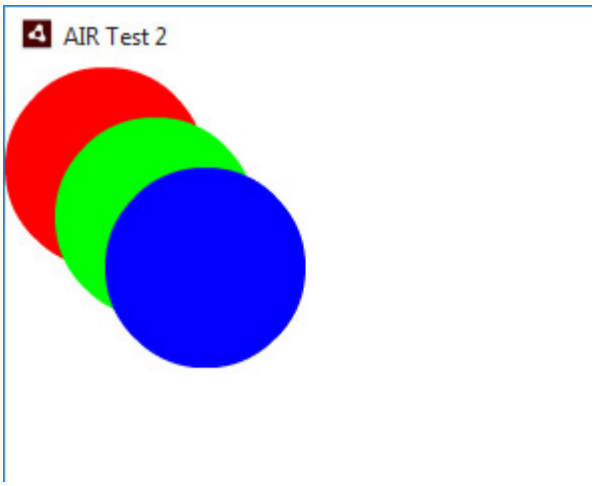
> Gli oggetti di visualizzazione possono avere solo un genitore. Quindi, se un bambino ha già un genitore e lo aggiungi a un altro oggetto, verrà rimosso dal precedente genitore.

Ordine Z / Stratificazione

Supponiamo che tu abbia replicato il codice dell'esempio precedente in modo da avere 3 cerchi:

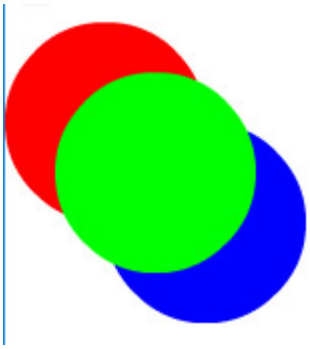
```
var redCircle:Shape = new Shape();
redCircle.graphics.beginFill(0xFF0000); //red
redCircle.graphics.drawCircle(50, 50, 50); //graphics.endFill is not required

var greenCircle:Shape = new Shape();
greenCircle.graphics.beginFill(0x00FF00); //green
greenCircle.graphics.drawCircle(75, 75, 50);

var blueCircle:Shape = new Shape();
blueCircle.graphics.beginFill(0x0000FF); //blue
blueCircle.graphics.drawCircle(100, 100, 50);

this.addChild(redCircle);
this.addChild(greenCircle);
this.addChild(blueCircle);
```

Dato che il metodo `addChild` aggiunge il figlio sopra a tutto il resto nello stesso genitore, otterrai questo risultato con gli oggetti a strati nello stesso ordine in cui usi addChild:



Se vuoi che un bambino abbia livelli diversi rispetto ai suoi fratelli, puoi usare `addChildAt` . Con `addChildAt` , si passa in un altro parametro che indica l'indice (ordine z) a cui dovrebbe essere assegnato il figlio. `0` è la posizione / livello più in basso.

```
this.addChild(redCircle);
this.addChild(greenCircle);
this.addChildAt(blueCircle,0); //This will add the blue circle at the bottom
```



Ora il cerchio blu è sotto i suoi fratelli. Se in seguito, si desidera modificare l'indice di un figlio, è possibile utilizzare il metodo `setChildIndex` (sul genitore del figlio).

```
this.setChildIndex(redCircle, this.numChildren - 1); //since z-index is 0 based, the top most
position is amount of children less 1.
```

Questo riorganizzerà il cerchio rosso in modo che sia sopra ogni altra cosa. Il codice sopra produce lo stesso risultato di `this.addChild(redCircle)` .

Rimozione di oggetti di visualizzazione

Per rimuovere oggetti, hai i metodi `removeChildAt` `removeChild` e `removeChildAt` , nonché il metodo `removeChildren` .

```
removeChild(redCircle); //this will take redCircle off the display list

removeChildAt(0); //this will take the bottom most object off the display list

removeChildren(); //this will clear all children from the display list
```

```
removeChildren(1); //this would remove all children except the bottom most

removeChildren(1,3); //this would remove the children at indexes 1, 2 & 3
```

eventi

Quando un bambino viene aggiunto all'elenco di visualizzazione, alcuni eventi vengono attivati su quel bambino.

- `Event.ADDED`
- `Event.ADDED_TO_STAGE`

Al contrario, ci sono anche gli eventi di rimozione:

- `Event.REMOVED`
- `Event.REMOVED_FROM_STAGE`

Adobe Animate / Flash Professional

Quando si ha a che fare con le timeline FlashProfessional / Adobe Animate, l'aggiunta di qualcosa alla timeline gestisce automaticamente le sfumature dell'elenco di visualizzazione. Hanno aggiunto e rimosso dalla lista di visualizzazione automaticamente dalla timeline.

Tuttavia, è bene tenere presente che:

> Se si manipola tramite codice la parentela di un oggetto di visualizzazione creato dalla timeline (utilizzando addChild / setChildIndex), tale figlio non verrà più rimosso automaticamente dalla timeline e dovrà essere rimosso tramite codice.

stratificazione

Ci possono essere situazioni in cui si decide che un set di oggetti di visualizzazione deve sempre trovarsi sopra un altro set di oggetti, ad esempio frecce sopra la testa, esplosioni su qualcosa appena esploso, ecc. Per eseguire questa operazione il più semplice possibile, è necessario designare e crea un set di `Sprite` , disponili in ordine dal basso verso l'alto, quindi aggiungi tutti gli oggetti di "sopra" impostati su un livello superiore a quello usato per gli oggetti di "sotto".

```
var monsters:Vector.<Monster>;
var bullets:Vector.<Bullet>; // desired: bullets strictly above monsters
var monsterLayer:Sprite=new Sprite();
var bulletLayer:Sprite=new Sprite();
addChild(monsterLayer);
addChild(bulletLayer);
```

Quindi, ogni volta che aggiungi un `Monster` all'elenco di visualizzazione, aggiungilo a `monsterLayer` e ogni volta che aggiungi un `Bullet` , aggiungi a `bulletLayer` per ottenere l'effetto desiderato.

Rimuovi tutti gli oggetti dall'elenco di visualizzazione

Se scegli come target Flash Player 11+, il metodo removeChildren incorporato è il modo migliore per rimuovere tutti i bambini:

```
removeChildren(); //a start and end index can be passed
```

Per le applicazioni legacy, lo stesso può essere realizzato con un ciclo:

```
while (numChildren > 0) {
    removeChildAt(0);
}
```

Passaggio da frame a commutazione del contenuto manuale

All'inizio, uno sviluppatore di Flash utilizza i frame, come sono nativamente disponibili in Flash Player, per ospitare vari schermi della loro applicazione (il più delle volte è un gioco). Alla fine potrebbero incappare in un problema che qualcosa va storto proprio perché hanno usato i frame, e trascurato le difficoltà che ne derivano, e cercano modi per mantenere la loro struttura di frame, ma anche rimuovere l'ostacolo dell'utilizzo di frame con le sue complicazioni. La soluzione consiste nell'utilizzare le classi discendenti di `Sprite` o i fotogrammi esportati come `MovieClip` con un singolo fotogramma (con quelli progettati in Adobe Flash CS) e passare manualmente i contenuti con `addChild()` e `removeChild()` .

La classe manager dovrebbe avere tutte le sue classi frame figlio pronte e ogni volta che viene chiamata una transizione, è possibile utilizzare una funzione simile a questa:

```
var frames:Vector.<DisplayObject>; // this holds instances to ALL children
var currentFrame_alt:int; // current frame. Can't use the property
function changeFrame(frame:int):void {
    removeChild(frames[currentFrame_alt]);
    addChild(frames[frame]);
    currentFrame_alt=frame;
}
```

Tutti i bambini possono sia inviare che ascoltare eventi con `Event.ADDED_TO_STAGE` utilizzato come punto di ingresso per qualsiasi cosa accada dopo `gotoAndStop()` che punta a tale frame e qualsiasi transizione in uscita può essere codificata come eventi basati su stringhe che vengono ascoltate nella classe `Main` , che poi esegue la transizione.

```
frames[0].addEventListener("startGame",startGame); // assuming frame 0 is a "Play" button
function startGame(e:Event):void {
    changeFrame(1); // switch to frame 1 - will display frames[1]
}
```

Ovviamente, l'insieme di stringhe dovrebbe essere predefinito, ad esempio, lo schermo introduttivo potrebbe avere due pulsanti per iniziare il gioco, "Inizia partita" e "Inizio disattivato", ad esempio, ei pulsanti dovrebbero inviare diversi eventi, che verranno quindi gestiti diversamente nella classe manager.

Questo modello può andare più in profondità di cui hai bisogno. Se qualsiasi frame del progetto

contiene un MovieClip con più frame, questo metodo può anche essere disaccoppiato in sprite.

Leggi Lavorare con gli oggetti di visualizzazione online: https://riptutorial.com/it/actionscript-3/topic/1628/lavorare-con-gli-oggetti-di-visualizzazione

Capitolo 11: Lavorare con i timer

Examples

Esempio di conto alla rovescia

```
package {
import flash.events.TimerEvent;
import flash.utils.Timer;

public class CountdownTimer extends Timer {

    public var time:Number = 0;

    public function CountdownTimer(time:Number = Number.NEGATIVE_INFINITY, delay:Number =
1000) {
        super(delay, repeatCount);

        if (!isNaN(time))
            this.time = time;

        repeatCount = Math.ceil(time / delay);

        addEventListener(TimerEvent.TIMER, timerHandler);
        addEventListener(TimerEvent.TIMER_COMPLETE, timerCompleteHandler);
    }

    override public function start():void {
        super.start();
    }

    protected function timerHandler(event:TimerEvent):void {
        time -= delay;
    }

    protected function timerCompleteHandler(event:TimerEvent):void {
    }

    override public function stop():void {
        super.stop();
    }

    public function dispose():void {
        removeEventListener(TimerEvent.TIMER, timerHandler);
        removeEventListener(TimerEvent.TIMER_COMPLETE, timerCompleteHandler);
    }

}
}
```

Questo `CountdownTimer` estende il `Timer` e viene utilizzato esattamente allo stesso modo, ad eccezione del tempo in cui il conto alla rovescia.

Esempio di utilizzo:

```
var timer:CountdownTimer = new CountdownTimer(5000);
timer.addEventListener(TimerEvent.TIMER, timerHandler);
timer.addEventListener(TimerEvent.TIMER_COMPLETE, completeHandler);
timer.start();

function timerHandler(event:TimerEvent):void {
    trace("Time remaining: " + event.target.time);
}

function completeHandler(event:TimerEvent):void {
    trace("Timer complete");
}
```

Sopra l'esempio produrrebbe:

```
[trace] Time remaining: 4000
[trace] Time remaining: 3000
[trace] Time remaining: 2000
[trace] Time remaining: 1000
[trace] Time remaining: 0
[trace] Timer complete
```

Intervalli e timeout

```
import flash.utils.*;
var intervalId:uint=setInterval(schroedingerCat,1000);
// execute a function once per second and gather interval ID
trace("Cat's been closed in the box.");
function schroedingerCat():void {
    if (Math.random()<0.04) {
        clearInterval(intervalId); // stop repeating by ID
        trace("Cat's dead.");
        return;
    }
    trace("Cat's still alive...");
}

var bombId:uint;
function plantBomb(seconds:Number):uint {
    trace("The bomb has been planted, and will blow in "+seconds.toFixed(3)+" seconds!");
    var id:uint=setTimeout(boom,seconds*1000); // parameter is in milliseconds
    return id;
}
function defuseBomb(id:uint):void {
    clearTimeout(id);
    trace("Bomb with id",id,"defused!");
}
function boom():void {
    trace("BOOM!");
}
```

`setInterval()` viene utilizzato per eseguire attività ripetute in modo asincrono come intervalli specificati. Viene utilizzato l'oggetto `Timer` interno, il valore restituito di tipo `uint` è il suo ID interno, tramite il quale è possibile accedere e interrompere la ripetizione chiamando `clearInterval()` . `setTimeout()` e `clearTimeout()` funzionano in modo simile, ma la chiamata alla funzione fornita viene eseguita una sola volta. È possibile fornire argomenti aggiuntivi per entrambe le funzioni di

impostazione, che verranno trasmesse alla funzione in ordine. Il numero di argomenti e il loro tipo non vengono controllati in fase di compilazione, quindi se si fornisce una combinazione strana di argomenti o una funzione che li richiede e non ne riceve nessuno, viene generato un errore "Errore # 1063: mancata corrispondenza del numero di argomenti".

È possibile eseguire entrambe le attività di `setInterval` e `setTimeout` con oggetti `Timer` regolari, utilizzando 0 o 1 per la proprietà `repeatCount` , 0 per le ripetizioni indefinite, 1 per uno.

Esempio di timer casuale

```
package {
    import flash.events.TimerEvent;
    import flash.events.TimerEvent;
    import flash.utils.Timer;

    public class RandomTimer extends Timer {

        public var minimumDelay:Number;
        public var maximumDelay:Number;
        private var _count:uint = 0;
        private var _repeatCount:int = 0;

        public function RandomTimer(min:Number, max:Number, repeatCount:int = 0) {
            super(delay, repeatCount);

            minimumDelay = min;
            maximumDelay = max;
            _repeatCount = repeatCount;
        }

        override public function start():void {
            delay = nextDelay();
            addEventListener(TimerEvent.TIMER, timerHandler);
            super.start();
        }

        private function nextDelay():Number {
            return (minimumDelay + (Math.random() * (maximumDelay - minimumDelay)));
        }

        override public function stop():void {
            removeEventListener(TimerEvent.TIMER, timerHandler);
            super.stop();
        }

        protected function timerHandler(event:TimerEvent):void {
            _count++;
            if ((_repeatCount > 0) && (_count >= _repeatCount)) {
                stop();
                dispatchEvent(new TimerEvent(TimerEvent.TIMER_COMPLETE));
            }
            delay = nextDelay();
        }

        override public function reset():void {
            _count = 0;
            super.reset();
        }
```

```
    }
}
```

Questo `RandomTimer` estende `Timer` e viene utilizzato esattamente allo stesso modo, ad eccezione del fatto che viene inviato a intervalli casuali.

Esempio di utilizzo, invio casuale tra 1 e 5 secondi:

```
var t:int = getTimer();

var timer:RandomTimer = new RandomTimer(1000, 5000);
timer.addEventListener(TimerEvent.TIMER, timerHandler);
timer.start();

function timerHandler(event:TimerEvent):void {
    trace("Time since last dispatch: " + (getTimer() - t));
    t = getTimer();
}
```

Sopra l'esempio produrrebbe:

```
[trace] Time since last dispatch: 1374
[trace] Time since last dispatch: 2459
[trace] Time since last dispatch: 3582
[trace] Time since last dispatch: 1335
[trace] Time since last dispatch: 4249
```

Leggi Lavorare con i timer online: https://riptutorial.com/it/actionscript-3/topic/2798/lavorare-con-i-timer

Capitolo 12: Lavorare con i video

Examples

Carica e riproduci file video esterni

---

riferimento : `NetConnection` , `NetStream` , `Video`

argomenti correlati : Lavorare con il suono

Esempio base di riproduzione di un file video esterno (FLV, MP4, F4V). Il codice riprodurrà anche i file audio M4A.

```
var nc:NetConnection = new NetConnection();
nc.connect(null);

var ns:NetStream = new NetStream(nc);

var myVideo:Video = new Video();
addChild(myVideo);

myVideo.attachNetStream(ns);

ns.play("http://www.yourwebsite.com/somefile.mp4");
```

Si noti il codice utilizzato un `nc.connect.null` ? Questo perché, in questo caso, non è necessario creare una connessione bidirezionale peer-to-peer (ad esempio: come previsto in un'app di chat video) poiché stiamo riproducendo un file memorizzato.

Impostando `nc.connect.null` è necessario fornire un collegamento a un file che si trova su un server Web o locale (stessa posizione / cartella) al file SWF in esecuzione.

- Per un file **web** utilizzare: `ns.play("http://www.yourwebsite.com/somefile.mp4");`
- Per un file **locale** utilizzare: `ns.play("somefile.mp4");`

Con NetStatusEvent

```
package {
    import flash.events.NetStatusEvent;
    import flash.net.NetStream;
    import flash.net.NetConnection;
    import flash.events.Event;
    import flash.media.Video;
    import flash.display.Sprite;
    public class VideoWithNetStatus extends Sprite {
        private var video:Video = new Video();
        private var nc:NetConnection;
```

```
        private var ns:NetStream;

        public function VideoWithNetStatus() {
            nc = new NetConnection();
            nc.addEventListener(NetStatusEvent.NET_STATUS, onStatus);
            nc.connect(null);//or media server url
        }

         private function onStatus(e:NetStatusEvent):void{
             switch(e.info.code){
                 case 'NetConnection.Connect.Success':
                     connectStream();
                 break;
                 default:
                     trace(e.info.code);//to see any unhadled events
             }
         }
         private function connectStream():void{
             ns = new NetStream(nc);
             ns.addEventListener(NetStatusEvent.NET_STATUS, onStatus);
             addChild(video);
             video.attachNetStream(ns);
             ns.play('url/to/video.flv');
         }
    }
}
```

Leggi Lavorare con i video online: https://riptutorial.com/it/actionscript-3/topic/2406/lavorare-con-i-video

Capitolo 13: Lavorare con il suono

Sintassi

- Sound.play (startTime: Number = 0, loop: int = 0, sndTransform: flash.media: SoundTransform = null): SoundChannel // Riproduce un suono caricato, restituisce un SoundChannel

Examples

Interrompi la riproduzione di un suono

```
import flash.net.URLRequest;
import flash.media.Sound;
import flash.media.SoundChannel;
import flash.events.Event;

var snd:Sound; = new Sound();
var sndChannel:SoundChannel
var sndTimer:Timer;

snd.addEventListener(Event.COMPLETE, soundLoaded);
snd.load(new URLRequest("soundFile.mp3")); //load after adding the complete event

function soundLoaded(e:Event):void
{
    sndChannel = snd.play();

    //Create a timer to wait 1 second
    sndTimer = new Timer(1000, 1);
    sndTimer.addEventListener(TimerEvent.TIMER, stopSound, false, 0, true);
    sndTimer.start();
}

function stopSound(e:Event = null):void {
    sndChannel.stop(); //Stop the sound
}
```

Infinito looping un suono

```
import flash.net.URLRequest;
import flash.media.Sound;
import flash.events.Event;

var req:URLRequest = new URLRequest("filename.mp3");
var snd:Sound = new Sound(req);

snd.addEventListener(Event.COMPLETE, function(e: Event)
{
    snd.play(0, int.MAX_VALUE); // There is no way to put "infinite"
}
```

Inoltre, non è necessario attendere il caricamento del suono prima di chiamare la funzione `play()` . Quindi questo farà lo stesso lavoro:

```
snd = new Sound(new URLRequest("filename.mp3"));
snd.play(0, int.MAX_VALUE);
```

E se vuoi davvero eseguire il loop del tempo inifinite per qualche motivo (`int.MAX_VALUE` suono di 1s per circa 68 anni, senza contare la pausa che un mp3 causa ...) puoi scrivere qualcosa del genere:

```
var st:SoundChannel = snd.play();
st.addEventListener(Event.SOUND_COMPLETE, repeat);
function repeat(e:Event) {
    st.removeEventListener(Event.SOUND_COMPLETE, repeat);
    (st = snd.play()).addEventListener(Event.SOUND_COMPLETE, repeat);
}
```

`play()` restituisce una nuova istanza dell'oggetto `SoundChannel` ogni volta che viene chiamata. Lo assegniamo alla variabile e ascoltiamo il suo evento SOUND_COMPLETE. Nell'evento callback, il listener viene rimosso dall'oggetto `SoundChannel` corrente e ne viene creato uno nuovo per il nuovo oggetto `SoundChannel` .

Carica e riproduce un suono esterno

```
import flash.net.URLRequest;
import flash.media.Sound;
import flash.events.Event;

var req:URLRequest = new URLRequest("click.mp3");
var snd:Sound = new Sound(req);

snd.addEventListener(Event.COMPLETE, function(e: Event)
{
    snd.play();
}
```

Leggi Lavorare con il suono online: https://riptutorial.com/it/actionscript-3/topic/2043/lavorare-con-il-suono

Capitolo 14: Lavorare con la geometria

Examples

Ottenere l'angolo tra due punti

Usando la matematica di vaniglia:

```
var from:Point = new Point(100, 50);
var to:Point = new Point(80, 95);

var angle:Number = Math.atan2(to.y - from.y, to.x - from.x);
```

Usando un nuovo vettore che rappresenta la differenza tra i primi due:

```
var difference:Point = to.subtract(from);

var angle:Number = Math.atan2(difference.y, difference.x);
```

> Nota: `atan2()` restituisce i radianti, non i gradi.

Ottenere la distanza tra due punti

Usando la matematica di vaniglia:

```
var from:Point = new Point(300, 10);
var to:Point = new Point(75, 40);

var a:Number = to.x - from.x;
var b:Number = to.y - from.y;

var distance:Number = Math.sqrt(a * a + b * b);
```

Utilizzo della funzionalità integrata di `Point` :

```
var distance:Number = to.subtract(from).length; // or
var distance:Number = Point.distance(to, from);
```

Convertire i radianti in gradi

```
var degrees:Number = radians * 180 / Math.PI;
```

Conversione di gradi in radianti

```
var radians:Number = degrees / 180 * Math.PI;
```

Il valore di un cerchio in gradi e radianti

- Un intero cerchio è di `360` gradi o `Math.PI * 2` radianti.
- La metà di questi valori segue per essere di `180` gradi o di `Math.PI`
- Un quarto è quindi `90` gradi o `Math.PI / 2` radianti.

Per ottenere un segmento come percentuale di un intero cerchio in radianti:

```
function getSegment(percent:Number):Number {
    return Math.PI * 2 * percent;
}

var tenth:Number = getSegment(0.1); // One tenth of a circle in radians.
```

Spostare un punto lungo un angolo

Supponendo di aver l'angolo che desideri muoversi dentro e un oggetto con `x` e `y` i valori che si desidera spostare:

```
var position:Point = new Point(10, 10);
var angle:Number = 1.25;
```

È possibile spostarsi lungo l'asse `x` con `Math.cos` :

```
position.x += Math.cos(angle);
```

E l'asse `y` con `Math.sin` :

```
position.y += Math.sin(angle);
```

Ovviamente puoi moltiplicare il risultato di `Math.cos` e `Math.sin` per la distanza che vuoi percorrere:

```
var distance:int = 20;

position.x += Math.cos(angle) * distance;
position.y += Math.sin(angle) * distance;
```

> Nota: l'angolo di input deve essere in radianti.

Determina se un punto si trova all'interno di un'area rettangolare

Puoi verificare se un punto si trova all'interno di un rettangolo usando `Rectangle.containsPoint()` :

```
var point:Point = new Point(5, 5);
var rectangle:Rectangle = new Rectangle(0, 0, 10, 10);

var contains:Boolean = rectangle.containsPoint(point); // true
```

Leggi Lavorare con la geometria online: https://riptutorial.com/it/actionscript-3/topic/3201/lavorare-

con-la-geometria

Capitolo 15: Lavorare con la timeline

Examples

Fare riferimento alla timeline principale o alla classe del documento da altri MovieClip

Nella timeline di qualsiasi `DisplayObject` collegato come discendente dell'albero di visualizzazione, è possibile utilizzare la proprietà `root` . Questa proprietà punta alla linea temporale principale nel caso in cui non ci sia una classe di documento personalizzata o la classe del documento se ne definisce una.

Poiché `root` è tipizzato `DisplayObject` , il compilatore non ti permetterà di accedere a metodi o proprietà personalizzati definiti sulla timeline principale o all'interno della tua classe di documenti come:

```
root.myCustomProperty = 10;
root.myCustomMethod();
```

Per ovviare a ciò, è possibile digitare `root` alla classe del documento nel caso in cui si abbia una classe di documento:

```
(root as MyDocumentClass).myCustomMethod();
```

Oppure `MovieClip` nel caso di nessuna classe di documento:

```
(root as MovieClip).myCustomMethod();
```

Il motivo per cui casting `MovieClip` funziona qui è perché `MovieClip` è `dynamic` . Ciò significa che il compilatore consente di dichiarare su di esso proprietà e metodo di runtime, evitando errori di compilazione quando si tenta di accedere a proprietà o metodi che non sono definiti esplicitamente su `MovieClip` . Il lato negativo di questo è che si perde tutta la sicurezza del tipo in fase di compilazione. Stai molto meglio dichiarando una classe di documenti e lanciando su di essa.

Leggi Lavorare con la timeline online: https://riptutorial.com/it/actionscript-3/topic/2459/lavorare-con-la-timeline

Capitolo 16: Lavorare con valori numerici

Examples

Se un numero è un valore pari

```
function isEven(n:Number):Boolean {
    return ((n & 1) == 0);
}
```

Esempi:

```
isEven(1); // false
isEven(2); // true

isEven(1.1); // false
isEven(1.2); // false
isEven(2.1); // true
isEven(2.2); // true
```

Se un numero è un valore dispari

```
function isOdd(n:Number):Boolean {
    return ((n & 1) == 1);
}
```

Esempi:

```
isOdd(1); // true
isOdd(2); // false

isOdd(1.1); // true
isOdd(1.2); // true
isOdd(2.1); // false
isOdd(2.2); // false
```

Arrotondare alla X più vicina

Per arrotondare un valore al multiplo più vicino di x:

```
function roundTo(value:Number, to:Number):Number {
    return Math.round(value / to) * to;
}
```

Esempio:

```
roundTo(8, 5); // 10
roundTo(17, 3); // 18
```

Errori di arrotondamento dei numeri in virgola mobile

```
/**
 * @param n Number to be rounded.
 * @param precision Decimal places.
 * @return Rounded Number
 */
function roundDecimal(n:Number, precision:Number):Number {
    var factor:int = Math.pow(10, precision);
    return (Math.round(n * factor) / factor);
}
```

Esempi:

```
trace(0.9 - 1); // -0.09999999999999998

trace(roundDecimal(0.9 - 1, 1));     // -0.1
trace(roundDecimal(0.9 - 1, 2));     // -0.1

trace(roundDecimal(0.9 - 1.123, 1)); // -0.2
trace(roundDecimal(0.9 - 1.123, 2)); // -0.22
trace(roundDecimal(0.9 - 1.123, 3)); // -0.223
```

Visualizzazione dei numeri con precisione richiesta

```
var a:Number=0.123456789;
trace(a);                       // 0.123456789
trace(a.toPrecision(4));        // 0.1235
trace(a.toFixed(4));            // 0.1235
trace(a.toExponential(4));      // 1.2345e-1
trace(a.toString(16));          // 0 - works for integer part only
var b:Number=12345678.9876543; // a bigger number to display rounding
trace(b);                       // 12345678.9876543
trace(b.toPrecision(4));        // 1.235e+7
trace(b.toFixed(4));            // 12345678.9877
trace(b.toExponential(4));      // 1.2345e+7
trace(b.toString(16));          // bc614e
b=1.0e+16;
trace(b.toString(36));          // 2qgpckvng1s
```

Leggi Lavorare con valori numerici online: https://riptutorial.com/it/actionscript-3/topic/1899/lavorare-con-valori-numerici

Capitolo 17: Manipolazione e filtraggio bitmap

introduzione

In questo argomento puoi imparare un po 'a manipolare i **bitmapdata** e l'elaborazione visiva, lavorare con i pixel e iniziare con i filtri degli effetti.

Examples

Effetto soglia (monocromatico)

---

necessario:

1. comprensione dei dati Bitmap e Bitmap

---

cos'è la soglia

> Questa regolazione prende tutti i pixel di un'immagine e ... li spinge verso il bianco puro o il nero puro

cosa dobbiamo fare

> ecco una **demo live** di questo esempio con alcune modifiche aggiuntive come l'utilizzo di un'interfaccia utente per modificare il livello di soglia in runtime.

soglia in azione script 3 dalla documentazione ufficiale as3

Verifica i valori dei pixel in un'immagine rispetto a una soglia specificata e imposta i pixel che passano il test ai nuovi valori di colore. Utilizzando il metodo threshold (), è possibile isolare e sostituire gli intervalli di colori in un'immagine ed eseguire altre operazioni logiche sui pixel dell'immagine.

La logica di test del metodo threshold () è la seguente:

1. Se ((pixelValue & mask) operazione (soglia e maschera)), quindi impostare il pixel su colore;
2. Altrimenti, se copySource == true, imposta il pixel sul valore di pixel corrispondente da sourceBitmap.

Ho appena commentato il seguente codice con nomi esattamente come descrizione citata.

```
import flash.display.BitmapData;
import flash.display.Bitmap;
import flash.geom.Rectangle;
import flash.geom.Point;

var bmd:BitmapData = new wildcat(); // instantied a bitmapdata from library a wildcat
var bmp:Bitmap = new Bitmap(bmd); // our display object to previewing bitmapdata on stage
addChild(bmp);
monochrome(bmd); // invoking threshold function

/**
    @param bmd, input bitmapData that should be monochromed
*/
function monochrome(bmd:BitmapData):void {
    var bmd_copy:BitmapData = bmd.clone(); // holding a pure copy of bitmapdata for
comparation steps
    // this is our "threshold" in description above, source pixels will be compared with this
value
    var level:uint = 0xFFAAAAAA; // #AARRGGBB. in this case i used RGB(170,170,170) with an
alpha of 1. its not median but standard
    // A rectangle that defines the area of the source image to use as input.
    var rect:Rectangle = new Rectangle(0,0,bmd.width,bmd.height);
    // The point within the destination image (the current BitmapData instance) that
corresponds to the upper-left corner of the source rectangle.
    var dest:Point = new Point();
    // thresholding will be done in two section
    // the last argument is "mask", which exists in both sides of comparation
    // first, modifying pixels which passed comparation and setting them all with "color"
white (0xFFFFFFFF)
    bmd.bitmapData.threshold(bmd_copy, rect, dest, ">", level, 0xFFFFFFFF, 0xFFFFFFFF);
    // then, remaining pixels and make them all with "color" black (0xFF000000)
    bmd.bitmapData.threshold(bmd_copy, rect, dest, "<=", level, 0xFF000000, 0xFFFFFFFF);
    // Note: as we have no alpha channel in our default BitmapData (pixelValue), we left it to
its full value, a white mask (0xffffffff)
}
```

Leggi Manipolazione e filtraggio bitmap online: https://riptutorial.com/it/actionscript-3/topic/8055/manipolazione-e-filtraggio-bitmap

Capitolo 18: Nozioni di base sullo sviluppo del gioco

introduzione

[! [inserisci la descrizione dell'immagine qui] [1]] [1] **nozioni** di **base** sullo sviluppo del gioco. ---------- *Nota* : questa serie di esercitazioni / articoli contiene molti concetti che possono essere forniti come argomenti separati prima. dobbiamo rinfrescarli nella mente e imparare un po 'di implementare le parti più critiche di un videogioco tramite actionscript-3. [1]: https://i.stack.imgur.com/CUIsz.png

Examples

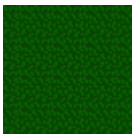
carattere isometrico che anima + movimento

① i concetti utilizzati in questo esempio:

Classe	uso
`URLRequest` + `Loader` + `Event`	Caricamento mappa atlante (sprite) dal percorso esterno.
`BitmapData` + `Sprite` + `beginBitmapFill` + `Matrix` + `stageWidth & stageHeight`	disegnare risorse caricate su bitmapdata, usando bitmap piastrellate, disegnando con la trasformazione.
`MovieClip` + `scrollRect` + `Bitmap` + `Rectangle`	creare e ritagliare un Movie Clip di carattere usando Bitmap come timeline.
`KeyboardEvent`	rilevamento degli input dell'utente
`Event.EXIT_FRAME`	implementando la funzione Loop del gioco

---

② Risorse: **(nessun permesso per l'utilizzo di queste risorse per scopi commerciali)**



③ Codice e commenti:

> **Nota:** FPS 15 Utilizzato per questo tutorial, è consigliato, ma se necessario, è necessario modificare alcune parti del codice da soli.

all'inizio dobbiamo scaricare le nostre risorse da url esterni.

```
const src_grass_tile_url:String = "https://i.stack.imgur.com/sjJFS.png";
const src_character_atlas_url:String = "https://i.stack.imgur.com/B7ztZ.png";

var loader:Loader = new Loader();
loader.contentLoaderInfo.addEventListener(Event.COMPLETE, setGround);
loader.load(new URLRequest(src_grass_tile_url));
```

---

setGround verrà salvato una volta che `src_grass_tile_url` è stato caricato e pronto per l'uso. in seguito implementando setGround per ottenere risorse e disegnarlo come sfondo del gioco

```
function setGround(e:Event):void {
    /* drawing ground */
    /* loader is a displayObject, so we can simply draw it into the bitmap data*/
    /* create an instance of Bitmapdata with same width and height as our window*/
    /* (also set transparent to false because grass image, does not contains any transparent
pixel) */
    var grass_bmd:BitmapData = new BitmapData(loader.width, loader.height, false, 0x0);
    /* time to draw */
    grass_bmd.draw(loader); // drawing loader into the bitmapData
    /* now we have to draw a tiled version of grass_bmd inside a displayObject Sprite to
displaying
       BitmapData on stage */
    var grass_sprite:Sprite = new Sprite();
    // for drawing a bitmap inside sprite, we must use <beginBitmapFill> with graphic property
of the sprite
    // then draw a full size rectangle with that Fill-Data
    // there is a repeat mode argument with true default value so we dont set it true again.
    // use a matrix for scalling grass Image during draw to be more cute!
    var mx:Matrix = new Matrix();
    mx.scale(2, 2);
    grass_sprite.graphics.beginBitmapFill(grass_bmd, mx);
    grass_sprite.graphics.drawRect(0, 0, stage.stageWidth, stage.stageHeight);
    // now add sprite to displayobjectcontainer to be displayed
    stage.addChild(grass_sprite);

    // well done, ground is ready, now we must initialize our character
    // first, load its data, i just re-use my loader for loading new image, but with another
complete handler (setCharacter)
    // so remove existing handler, then add new one
    loader.contentLoaderInfo.removeEventListener(Event.COMPLETE, setGround);
    loader.contentLoaderInfo.addEventListener(Event.COMPLETE, setCharacter);
    loader.load(new URLRequest(src_character_atlas_url));
}
```

il codice è dannatamente commentato, dopo averlo terminato, è tempo di implementare il personaggio. il carattere contiene anche una risorsa che deve essere caricata allo stesso modo. così alla fine di `setGround` ci stiamo dirigendo verso il `setCharacter` che è un altro richiamo completo.

```
function setCharacter(e:Event):void {
    // let assuming that what is really character!
    // a set of images inside a single image!
    // that images are frames of our character (also provides movement for different
directions)
    // first load this
    var character_bmd:BitmapData = new BitmapData(loader.width, loader.height, true, 0x0); //
note character is transparent
    character_bmd.draw(loader);
    // take a look at sprite sheet, how many frames you see?
    // 42 frames, so we can get width of a single frame
    const frame_width:uint = character_bmd.width / 42; // 41 pixels
    // as i show you above, to displaying a BitmapData, we have to draw it using a
DisplayObject,
    // another way is creating a Bitmap and setting its bitmapdata
    var character_bmp:Bitmap = new Bitmap(character_bmd);
    // but its not enough yet, a movieClip is necessary to cliping and animating this bitmap
(as a child of itself)
    var character_mc:MovieClip = new MovieClip();
    character_mc.addChild(character_bmp);
    character_bmp.name = "sprite_sheet"; // setting a name to character_bmp, for future
accessing
    character_mc.scrollRect = new Rectangle(0, 0, frame_width, character_bmd.height); //
cliping movieclip, to dusplaying only one frame
    character_mc.name = "character"; // setting a name to character_mc, for future accessing
    stage.addChild(character_mc); // adding it to stage.
    // well done, we have a character, but its static yet! 2 steps remaining. 1 controlling 2
animating
    // at first setting a control handler for moving character in 8 directions.
    stage.addEventListener(KeyboardEvent.KEY_DOWN, keyDown);
    stage.addEventListener(KeyboardEvent.KEY_UP, keyUp);
}
```

ora il personaggio è pronto per il controllo. è visualizzato bene e pronto per il controllo. quindi gli eventi della tastiera sono collegati e ascoltano i tasti freccia come il seguente codice:

```
// we storing key stats inside <keys> Object
var keys:Object = {u:false, d:false, l:false, r:false};
function keyDown(e:KeyboardEvent):void {
    switch (e.keyCode) {
        case 38: //up
            keys.u = true;
            break;
        case 40: //down
            keys.d = true;
            break;
        case 37: //left
            keys.l = true;
            break;
        case 39: //right
            keys.r = true;
            break;
    }
```

```
}
function keyUp(e:KeyboardEvent):void {
    switch (e.keyCode) {
        case 38: //up
            keys.u = false;
            break;
        case 40: //down
            keys.d = false;
            break;
        case 37: //left
            keys.l = false;
            break;
        case 39: //right
            keys.r = false;
            break;
    }
}
```

`keys:Object` memorizza la variabile booleana di 4 per ogni tasto freccia, il processo di spostamento deve essere eseguito all'interno della funzione di aggiornamento (Loop) del gioco, quindi dobbiamo passare le statistiche della tastiera ad esso. lascia implementare la funzione **Loop** .

```
// initialize game Loop function for updating game objects
addEventListener(Event.EXIT_FRAME, loop);

// speed of character movement
const speed:Number = 5;
// this function will be called on each frame, with same rate as your project fps
function loop(e:Event):void {
    if (keys.u) stage.getChildByName("character").y -= speed;
    else if (keys.d) stage.getChildByName("character").y += speed;
    if (keys.l) stage.getChildByName("character").x -= speed;
    else if (keys.r) stage.getChildByName("character").x += speed;
}
```

la velocità è una costante di aiuto, definisce la velocità del carattere. il codice sopra presenta un semplice movimento in 8 direzioni con questa priorità bassa: `Up > Down Left > Right` . quindi se la freccia su e giù viene premuta nello stesso tempo, il personaggio si sposta solo verso l' *alto* (non si blocca).

molto bene!!! solo un passo rimanente, l'animazione, la parte più importante di questo tutorial

cos'è veramente l'animazione? un set di fotogrammi chiave che contiene almeno un fotogramma consente di creare il nostro oggetto keyframs, che contiene il nome dei fotogrammi chiave
e anche alcuni dati sul frame iniziale e finale di questo keyframe
Nota , nei giochi isometrici, ogni fotogramma chiave contiene 8 direzioni (può essere ridotto a 5 con l'uso di flipping)

```
var keyframs:Object = {
    idle: {up:[0,0], up_right:[1,1], right:[2,2], down_right:[3,3], down:[4,4]}, // [2,2]
means start frame is 2 and end frame is 2
    run: {up:[5,10], up_right:[11,16], right:[17,22], down_right:[23,28], down:[29,34]}
};
```

dovremmo ignorare i fotogrammi rimanenti, questo esempio fornisce solo inattività ed esegue l'animazione
ad esempio il fotogramma iniziale dell'animazione inattiva con direzione destra, è:
<keyframs.idle.right [0]>
ora consente di implementare la funzione Animator

```
var current_frame:uint;
function animate(keyframe:Array):void {
    // how it works
    // just called with a keyframe with direction (each frame),
    // if keyframe is what is already playing, its just moved to next frame and got updated
(or begning frame for loop)
    // other wise, just moved to begining frame of new keyframe
    if (current_frame >= keyframe[0] && current_frame <= keyframe[1]) { // check if in bound
        current_frame++;
        if (current_frame > keyframe[1]) // play back if reached
            current_frame = keyframe[0];
    } else {
        current_frame = keyframe[0]; // start new keyframe from begining
    }
    // moving Bitmap inside character MovieClip
    var character:MovieClip = stage.getChildByName("character") as MovieClip;
    var sprite_sheet:Bitmap = character.getChildByName("sprite_sheet") as Bitmap;
    sprite_sheet.x = -1 * current_frame * character.width;
}
```

leggere i commenti della funzione precedente, tuttavia il lavoro principale di questa funzione è lo spostamento di *sprite_sheet* `Bitmap` all'interno del *personaggio* `MovieClip` .

sappiamo che ogni aggiornamento dovrebbe essere fatto all'interno della funzione Loop, quindi invocheremo questa funzione da Loop con i relativi keyframe. questa è la funzione Loop aggiornata:

```
// speed of character movement
const speed:Number = 8;
var last_keyStat:Object = {u:false, d:false, l:false, r:false}; // used to getting a backup of
previous keyboard stat for detecting correct idle direction
// this function will be called on each frame, with same rate as your project fps
function loop(e:Event):void {
    if (keys.u) stage.getChildByName("character").y -= speed;
    else if (keys.d) stage.getChildByName("character").y += speed;
    if (keys.l) stage.getChildByName("character").x -= speed;
    else if (keys.r) stage.getChildByName("character").x += speed;

    // animation detection
    if (keys.u && keys.l) { animate(keyframs.run.up_right); flip(true); }
    else if (keys.u && keys.r) { animate(keyframs.run.up_right); flip(false); }
    else if (keys.d && keys.l) { animate(keyframs.run.down_right); flip(true); }
    else if (keys.d && keys.r) { animate(keyframs.run.down_right); flip(false); }
    else if (keys.u) { animate(keyframs.run.up); flip(false); }
    else if (keys.d) { animate(keyframs.run.down); flip(false); }
    else if (keys.l) { animate(keyframs.run.right); flip(true); }
    else if (keys.r) { animate(keyframs.run.right); flip(false); }
    else {
        // if character dont move, so play idle animation
        // what is the best practice to detecting idle direction?
        // my suggestion is to sotring previous keyboard stats to determining which idle
```

```
direction is correct
        // do any better thing if possible (absolutely is possible)
        // i just simply copy it from above, and replaced (keys) with (last_keyStat) and (run)
with (idle)
        if (last_keyStat.u && last_keyStat.l) { animate(keyframs.idle.up_right); flip(true); }
        else if (last_keyStat.u && last_keyStat.r) { animate(keyframs.idle.up_right);
flip(false); }
        else if (last_keyStat.d && last_keyStat.l) { animate(keyframs.idle.down_right);
flip(true); }
        else if (last_keyStat.d && last_keyStat.r) { animate(keyframs.idle.down_right);
flip(false); }
        else if (last_keyStat.u) { animate(keyframs.idle.up); flip(false); }
        else if (last_keyStat.d) { animate(keyframs.idle.down); flip(false); }
        else if (last_keyStat.l) { animate(keyframs.idle.right); flip(true); }
        else if (last_keyStat.r) { animate(keyframs.idle.right); flip(false); }
    }
    // update last_keyStat backup
    last_keyStat.u = keys.u;
    last_keyStat.d = keys.d;
    last_keyStat.l = keys.l;
    last_keyStat.r = keys.r;
}
```

leggi i commenti, ci limitiamo semplicemente a rilevare un vero keyframe tramite le statistiche della tastiera. poi fai anche la stessa cosa per rilevare l'animazione inattiva. per le animazioni inattive non abbiamo input da utilizzare per rilevare quale carattere di direzione è attivo, quindi una variabile helper simle potrebbe essere utile per memorizzare lo stato precedente della tastiera (last_keyStat).

---

inoltre c'è una nuova funzione `flip` che è un'altra funzione di aiuto usata per simulare animazioni mancanti (sinistra + up_left + down_left) anche questa funzione fa alcune correzioni che vengono commentate di seguito:

```
// usage of flip function is because of Movieclip registration point, its a fix
// as the registration point of MovieClip is not placed in center, when flipping animation
(for non existing directions inside spritesheet)
// character location changes with an unwanted value equal its width, so we have to prevent
this and push it back or forward during flip
function flip(left:Boolean):void {
    var character:MovieClip = stage.getChildByName("character") as MovieClip;
    if (left) {
        if (character.scaleX != -1) {
            character.scaleX = -1;
            character.x += character.width; // comment this line to see what happen without
this fix
        }
    } else {
        if (character.scaleX != 1) {
            character.scaleX = 1;
            character.x -= character.width; // comment this line to see what happen without
this fix
        }
    }
}
```

il nostro lavoro sta finendo qui. un ringraziamento speciale per Editor's che rende questo

tutorial più irrinunciabile. anche Ecco una demo dal vivo di questo tutorial più un link esterno del codice completo.

④ Riferimenti esterni:

- **codice completo**
- **Dimostrazione dal vivo**

Leggi Nozioni di base sullo sviluppo del gioco online: https://riptutorial.com/it/actionscript-3/topic/8237/nozioni-di-base-sullo-sviluppo-del-gioco

Capitolo 19: Ottimizzazione delle prestazioni

Examples

Grafica basata vettoriale

La grafica vettoriale è rappresentata da una pletora di dati che devono essere calcolati dalla CPU (punti vettoriali, archi, colori, ecc.). Qualsiasi cosa diversa dalle forme semplici con punti e linee rette minimali consumerà grandi quantità di risorse della CPU.

C'è un indicatore "Cache as Bitmap" che può essere attivato. Questo flag memorizza il risultato del disegno di DisplayObject basato su vettori per ridisegni molto più veloci. La trappola di questo è che se ci sono trasformazioni applicate all'oggetto, l'intera cosa deve essere ridisegnata e ricollocata. Questo può essere più lento di non accenderlo affatto se sono applicate trasformazioni fotogramma per fotogramma (rotazione, ridimensionamento, ecc.).

Generalmente, il rendering della grafica tramite bitmap è molto più performante rispetto all'utilizzo della grafica vettoriale. Librerie come flixel ne approfittano per rendere gli sprite su una "tela" senza ridurre il framerate.

Testo

Il rendering del testo consuma molta CPU. I caratteri sono resi in modo simile alla grafica vettoriale e contengono molti punti vettoriali per ogni personaggio. Text alterazione fotogramma per fotogramma *si* degrada le prestazioni. Il flag "Cache as bitmap" è estremamente utile se usato correttamente, il che significa che devi evitare:

- Modifica del testo frequentemente.
- Trasformazione del campo di testo (rotazione, ridimensionamento).

Tecniche semplici come il wrapping degli aggiornamenti di testo in un'istruzione `if` fanno la differenza principale:

```
if (currentScore !== oldScore) {
    field.text = currentScore;
}
```

Il testo può essere renderizzato usando il renderer anti-alias incorporato in Flash, o usando "font dispositivo". L'uso di "tipi di carattere dispositivo" rende il rendering del testo molto più veloce, sebbene faccia apparire il testo frastagliato (alias). Inoltre, i caratteri del dispositivo richiedono che il font sia preinstallato dall'utente finale, oppure che il testo possa "scomparire" sul PC dell'utente, sebbene appaia bene sul tuo.

```
field.embedFonts = false; // uses "device fonts"
```

Vettore e per ogni vs matrici e per

Usando il tipo `Vector.<T>` e il ciclo `for each` loop è più performante di un array convenzionale e `for` loop:

Buono:

```
var list:Vector.<Sprite> = new <Sprite>[];

for each(var sprite:Sprite in list) {
    sprite.x += 1;
}
```

Male:

```
var list:Array = [];

for (var i:int = 0; i < list.length; i++) {
    var sprite:Sprite = list[i];

    sprite.x += 1;
}
```

Rimozione veloce degli elementi dell'array

Se non si richiede che un array sia in un ordine particolare, un piccolo trucco con `pop()` ti consentirà enormi guadagni in termini di prestazioni rispetto a `splice()` .

Quando si `splice()` un array, l'indice degli elementi successivi in quell'array deve essere ridotto di 1. Questo processo può consumare una grande quantità di tempo se l'array è grande e l'oggetto che si sta rimuovendo è più vicino all'inizio di quell'array .

Se non ti interessa l'ordine degli elementi nell'array, puoi invece sostituire l'elemento che vuoi rimuovere con un oggetto `pop()` dalla fine dell'array. In questo modo, gli indici di tutti gli altri elementi dell'array rimangono gli stessi e il processo non degrada in termini di prestazioni man mano che la lunghezza dell'array aumenta.

Esempio:

```
function slowRemove(list:Array, item:*):void {
    var index:int = list.indexOf(item);

    if (index >= 0) list.splice(index, 1);
}

function fastRemove(list:Array, item:*):void {
    var index:int = list.indexOf(item);

    if (index >= 0) {
        if (index === list.length - 1) list.pop();

        else {
            // Replace item to delete with last item.
            list[index] = list.pop();
        }
```

```
    }
}
```

Vettori invece di matrici

Flash Player 10 ha introdotto il vettore. <*> Tipo di elenco generico più veloce della matrice. Tuttavia, questo non è completamente vero. Solo i seguenti tipi di vettore sono più veloci rispetto alla controparte Array, a causa del modo in cui sono implementati in Flash Player.

- `Vector.<int>` - Vettore di numeri interi a 32 bit
- `Vector.<uint>` - Vettore di numeri interi senza segno a 32 bit
- `Vector.<Double>` - Vettore di galleggianti a 64 bit

In tutti gli altri casi, l'uso di una matrice sarà più performante rispetto all'utilizzo di Vettori, per tutte le operazioni (creazione, manipolazione, ecc.). Tuttavia, se desideri "scrivere con decisione" il tuo codice, puoi utilizzare i Vettori nonostante il rallentamento. FlashDevelop ha una sintassi che consente agli elenchi a discesa di completamento del codice di funzionare anche per gli array, utilizzando `/*ObjectType*/Array` .

```
var wheels:Vector.<Wheel> // strongly typed, but slow

var wheels:/*Wheel*/Array // weakly typed, but faster
```

Riutilizzare e mettere in comune la grafica

La creazione e la configurazione di oggetti `Sprite` e `TextField` in fase di esecuzione può essere costosa se ne stai creando centinaia di migliaia su un singolo frame. Quindi un trucco comune è "mettere in comune" questi oggetti per un successivo riutilizzo. Ricorda che non stiamo solo cercando di ottimizzare il tempo di creazione (`new Sprite()`) ma anche la configurazione (impostazione delle proprietà di default).

Diciamo che stavamo costruendo un componente di lista usando centinaia di oggetti TextField. Quando hai bisogno di creare un nuovo oggetto, controlla se un oggetto esistente può essere riutilizzato.

```
var pool:Array = [];

if (pool.length > 0){

    // reuse an existing TextField
    var label = pool.pop();

}else{
    // create a new TextField
    label = new TextField();

    // initialize your TextField over here
    label.setDefaultTextFormat(...);
    label.multiline = false;
    label.selectable = false;
}
```

```
// add the TextField into the holder so it appears on-screen
// you will need to layout it and set its "text" and other stuff seperately
holder.addChild(label);
```

Successivamente, quando si distrugge il componente (o lo si rimuove dallo schermo), ricordarsi di aggiungere nuovamente le etichette non utilizzate nel pool.

```
foreach (var label in allLabels){
    label.parent.removeChild(label); // remove from parent Sprite
    pool.push(label); // add to pool
}
```

Nella maggior parte dei casi è preferibile creare un pool per utilizzo anziché un pool globale. Svantaggi per la creazione di un pool globale è necessario reinizializzare l'oggetto ogni volta per recuperarlo dal pool, per annullare le impostazioni eseguite da altre funzioni. Questo è altrettanto costoso e praticamente nega l'aumento delle prestazioni dell'utilizzo del pool in primo luogo.

Leggi Ottimizzazione delle prestazioni online: https://riptutorial.com/it/actionscript-3/topic/2215/ottimizzazione-delle-prestazioni

Capitolo 20: Progettazione di applicazioni reattive

Examples

Applicazione di base reattiva

```
package
{
    import flash.display.Sprite;
    import flash.display.StageAlign;
    import flash.display.StageScaleMode;
    import flash.events.Event;

    public class Main extends Sprite
    {
        //Document Class Main Constructor
        public function Main()
        {
            //Sometimes stage isn't available yet, so if not, wait for it before proceeding
            if (!stage) {
                addEventListener(Event.ADDED_TO_STAGE, stageReady);
            }else {
                stageReady();
            }
        }

        protected function stageReady(e:Event = null):void {
            //align the stage to the top left corner of the window/container
            stage.align = StageAlign.TOP_LEFT;
            //don't scale the content when the window resizes
            stage.scaleMode = StageScaleMode.NO_SCALE;

            //listen for when the window is resized
            stage.addEventListener(Event.RESIZE, stageResized);
        }

        protected function stageResized(e:Event):void {
            //use stage.stageWdith & stage.stageHeight to repostion and resize items
        }
    }
}
```

Esecuzione di lunghi processi e non ottenere un'applicazione non rispondente

Ci sono situazioni in cui devi calcolare qualcosa di veramente grande nella tua applicazione Flash, senza interrompere l'esperienza dell'utente. Per questo, è necessario concepire il tuo lungo processo come un processo a più fasi con stato salvato tra le iterazioni. Ad esempio, è necessario eseguire un aggiornamento in background di molti oggetti interni, ma se si desidera aggiornarli tutti contemporaneamente con un semplice `for each (var o in objects) { o.update(); }` , Flash

brevemente (o non brevemente) non risponde all'utente. Quindi, è necessario eseguire uno o più aggiornamenti per frame.

```
private var processing:Boolean;      // are we in the middle of processing
private var lastIndex:int;           // where did we finish last time
var objects:Vector.<UpdatingObject>; // the total list of objects to update
function startProcess():Boolean {
    if (processing) return false; // already processing - please wait
    startProcessing=true;
    lastIndex=0;
    if (!hasEventListener(Event.ENTER_FRAME,iterate))
        addEventListener(Event.ENTER_FRAME,iterate); // enable iterating via listener
}
private function iterate(e:Event):void {
    if (!processing) return; // not processing - skip listener
    objects[lastIndex].update(); // perform a quantum of the big process
    lastIndex++; // advance in the big process
    if (lastIndex==objects.length) {
        processing=false; // finished, clear flag
    }
}
```

L'elaborazione avanzata può includere l'utilizzo di `getTimer()` per controllare il tempo trascorso e consentire un'altra iterazione se il tempo non si esaurisce, dividere `update()` in più funzioni se l'aggiornamento è troppo lungo, visualizzando i progressi altrove, regolando il processo di iterazione per adattarsi ai cambiamenti del elenco di oggetti da elaborare e molti altri.

Leggi Progettazione di applicazioni reattive online: https://riptutorial.com/it/actionscript-3/topic/1615/progettazione-di-applicazioni-reattive

Capitolo 21: Programmazione orientata agli oggetti

Examples

Costruttore "sovraccaricato" tramite metodo statico

L'overloading del costruttore non è disponibile in As3.

Per fornire un modo diverso per recuperare un'istanza di una classe, è possibile fornire un metodo `public static` per fungere da "costruttore" alternativo.

Un esempio è `flash.geom.Point` , che rappresenta un oggetto punto 2D. Le coordinate per definire il punto possono essere

- **cartesiano** nel costruttore regolare

  ```
  public function Point(x:Number = 0, y:Number = 0)
  ```

  esempio di utilizzo:

  ```
  var point:Point = new Point(2, -.5);
  ```

- **polare** in un metodo statico

  ```
  public static function polar(len:Number, angle:Number):Point
  ```

  esempio di utilizzo:

  ```
  var point:Point = Point.polar(12, .7 * Math.PI);
  ```

  Perché non è un vero costruttore, non c'è una `new` parola chiave.

imposta e ottieni funzioni

Per garantire l'incapsulamento, le variabili membro di una classe dovrebbero essere `private` e essere accessibili al `public` tramite metodi pubblici di accesso `get` / `set` . È una pratica comune anteporre i campi privati a _

```
public class Person
{
    private var _name:String = "";

    public function get name():String{
        return _name;
        //or return some other value depending on the inner logic of the class
```

```
    }


    public function set name(value:String):void{
        //here you may check if the new value is valid
        //or maybe dispatch some update events or whatever else
        _name = value;
    }
```

A volte non è nemmeno necessario creare un campo `private` per una coppia `get` / `set` .
Ad esempio in un controllo come un gruppo radio personalizzato è necessario sapere quale pulsante di opzione è selezionato, tuttavia al di fuori della classe è necessario solo un modo per `get` / `set` solo il valore selezionato:

```
public function get selectedValue():String {
    //just the data from the element
    return _selected ? _selected.data : null;
}
public function set selectedValue(value:String):void {
    //find the element with that data
    for (var i:int = 0; i < _elems.length; i++) {
        if (_elems[i].data == value) {
            _selected = _elems[i];//set it
            processRadio();//redraw
            return;
        }
    }
}
```

Pacchi

I pacchetti sono pacchetti di classi. Ogni classe deve essere dichiarata all'interno di un pacchetto utilizzando la dichiarazione del `package` . L'istruzione del `package` è seguita dal nome del pacchetto o seguita da nulla in caso di aggiunta di classi al pacchetto principale. I sotto-pacchetti sono creati usando la delimitazione punto (`.`). L'istruzione del pacchetto è seguita da un blocco che conterrà *una singola definizione di* `class` . Esempi:

```
package {
    // The top level package.
}

package world {
    // A package named world.
}

package world.monsters {
    // A package named monsters within a package named world.
}
```

I pacchetti devono essere correlati alla struttura dei file delle classi relative alla radice di origine. Supponendo di avere una cartella radice di origine chiamata `src` , quanto sopra potrebbe essere correttamente rappresentato nel filesystem come:

```
src
    TopLevelClass.as

    world
        ClassInWorldPackage.as
        AnotherClassInWorldPackage.as

        monsters
            Zombie.as
```

Metodo prioritario

Quando `extend` una classe, puoi eseguire l' `override` metodi che la classe ereditata definisce utilizzando la parola chiave `override` :

```
public class Example {
    public function test():void {
        trace('It works!');
    }
}

public class AnotherExample extends Example {
    public override function test():void {
        trace('It still works!');
    }
}
```

Esempio:

```
var example:Example = new Example();
var another:AnotherExample = new AnotherExample();

example.test(); // Output: It works!
another.test(); // Output: It still works!
```

È possibile utilizzare la parola chiave `super` per fare riferimento al metodo originale dalla classe che viene ereditata. Ad esempio, potremmo cambiare il corpo di `AnotherExample.test()` in:

```
public override function test():void {
    super.test();
    trace('Extra content.');
}
```

Con il risultato di:

```
another.test(); // Output: It works!
                //         Extra content.
```

Sostituire i costruttori di classi è un po 'diverso. La parola chiave `override` viene omessa e l'accesso al costruttore ereditato avviene semplicemente con `super()` :

```
public class AnotherClass extends Example {
    public function AnotherClass() {
```

```
        super(); // Call the constructor in the inherited class.
    }
}
```

Puoi anche sostituire i metodi `get` e `set` .

getter e setter

Getter e setter sono metodi che si comportano come proprietà. significa che hanno una struttura funzionale ma quando vengono utilizzati, vengono utilizzati come proprietà:

Struttura delle funzioni getter :

essi dovrebbero avere `get` parola chiave dopo la `function` parola chiave e prima il nome della funzione, senza argomenti, un tipo di ritorno specificato e deve restituire un valore:

```
public function get myValue():Type{
    //anything here
    return _desiredValue;
}
```

Sintassi :

per ottenere il valore da un getter, la sintassi è la stessa di ottenere un valore da una proprietà (non vengono utilizzati parens `()`).

```
trace(myValue);
```

Struttura delle funzioni di setter :

dovrebbero avere `set` parola chiave parola chiave dopo `function` e prima del nome funzione, con un argomento e nessun valore restituito.

```
public function set myValue(value:Type):void{
    //anything here
    _desiredProperty=value;
}
```

Sintassi :

per impostare il valore di un setter, la sintassi è uguale all'impostazione di un valore su una proprietà (utilizzando il segno di uguale `=` quindi il valore).

```
myValue=desiredValue;
```

impostare un getter e setter per un valore :

> Nota: se crei solo getter o solo setter con un nome, quella proprietà sarà di sola lettura o solo set.

per rendere una proprietà leggibile e impostabile, è necessario creare un getter e un setter con:

1. lo stesso nome
2. lo stesso tipo (tipo di valore di ritorno per il getter e tipo di valore di input (argomento) per il setter,

> Nota: getter e setter non dovrebbero avere un nome uguale ad altre proprietà o metodi.

Utilizzo di getter e setter:

L'utilizzo di getter e setter invece delle normali proprietà ha molti vantaggi:

1. **creazione di proprietà di sola lettura o di sola impostazione:**
ad esempio il numero di bambini in un oggetto di visualizzazione. non può essere impostato.

2. **accesso alle proprietà private:**
un esempio:

```
private var _private:Type=new Type();
//note that function name "private" is not same as variable name "_private"
public function get private():Type{
    return _private;
}
```

3. **quando è necessario apportare alcune modifiche dopo aver impostato un valore:**
in questo esempio, la modifica di questa proprietà deve essere notificata:

```
public static function set val:(input:Type):void{
    _desiredProperty=input;
    notifyValueChanged();
}
```

e molti altri usi

Leggi Programmazione orientata agli oggetti online: https://riptutorial.com/it/actionscript-3/topic/2042/programmazione-orientata-agli-oggetti

Capitolo 22: Singleton Pattern

Osservazioni

Il modello singleton ha l'obiettivo di consentire l'esistenza di una sola istanza di una classe in qualsiasi momento.

Prevenire l'istanziazione diretta tramite il costruttore di solito impedisce di renderlo privato. Tuttavia, questo non è possibile in As3 e quindi altri modi per controllare il numero di istanze devono essere utilizzati.

Examples

Singleton enforcer tramite istanza privata

In questo approccio, il singolo è accessibile tramite il metodo statico:

```
Singleton.getInstance();
```

Per applicare solo un'istanza del singleton, una variabile statica privata conserva l'istanza, mentre eventuali ulteriori tentativi di istanziazione di un'istanza vengono applicati all'interno del costruttore.

```
package {

public class Singleton {

    /** Singleton instance */
    private static var _instance: Singleton = new Singleton();

    /** Return singleton instance. */
    public static function getInstance():Singleton {
        return _instance;
    }

    /** Constructor as singleton enforcer. */
    public function Singleton() {
        if (_instance)
            throw new Error("Singleton is a singleton and can only be accessed through
Singleton.getInstance()");
    }

}
}
```

Leggi Singleton Pattern online: https://riptutorial.com/it/actionscript-3/topic/1437/singleton-pattern

Capitolo 23: tipi

Examples

Digitare Casting

Il casting del tipo viene eseguito con l'operatore `as` :

```
var chair:Chair = furniture as Chair;
```

O avvolgendo il valore in `Type()` :

```
var chair:Chair = Chair(furniture);
```

Se il cast fallisce con `as` , il risultato di tale cast è `null` . Se il cast fallisce avvolgendo in `Type()` , viene lanciato un errore `TypeError` .

Il tipo di funzione

Le funzioni sono del tipo `Function` :

```
function example():void { }
trace(example is Function); // true
```

Possono essere referenziati da altre variabili con il tipo `Function` :

```
var ref:Function = example;
ref(); // ref.call(), ref.apply(), etc.
```

E possono essere passati come argomenti per parametri il cui tipo è `Function` :

```
function test(callback:Function):void {
    callback();
}

test(function() {
    trace('It works!');
}); // Output: It works!
```

Il tipo di classe

I riferimenti alle dichiarazioni di classe sono tipizzati `Class` :

```
var spriteClass:Class = Sprite;
```

È possibile utilizzare variabili digitate `Class` per creare istanze di tale classe:

```
var sprite:Sprite = new spriteClass();
```

Questo può essere utile per passare un argomento di tipo `Class` a una funzione che potrebbe creare e istanza della classe fornita:

```
function create(type:Class, x:int, y:int):* {
    var thing:* = new type();

    thing.x = x;
    thing.y = y;

    return thing;
}

var sprite:Sprite = create(Sprite, 100, 100);
```

Tipi di annotazione

Puoi dire al compilatore il tipo di un valore annotandolo con `:Type` :

```
var value:int = 10; // A property "value" of type "int".
```

I parametri delle funzioni e i tipi di ritorno possono anche essere annotati:

```
// This function accepts two ints and returns an int.
function sum(a:int, b:int):int {
    return a + b;
}
```

Se si tenta di assegnare un valore con un tipo non corrispondente, verrà `TypeError` un `TypeError` :

```
var sprite:Sprite = 10; // 10 is not a Sprite.
```

Controllo dei tipi

È possibile utilizzare l'operatore `is` per verificare se un valore è di un determinato tipo:

```
var sprite:Sprite = new Sprite();

trace(sprite is Sprite); // true
trace(sprite is DisplayObject); // true, Sprite inherits DisplayObject
trace(sprite is IBitmapDrawable); // true, DisplayObject implements IBitmapDrawable
trace(sprite is Number); // false
trace(sprite is Bitmap); // false, Bitmap inherits DisplayObject
                         // but is not inherited by Sprite.
```

C'è anche un `instanceof` operatore (sconsigliato), che funziona quasi identica a `is` la differenza che *restituisce* `false` *quando si controlla per le interfacce implementate* e tipi int / uint.

Il `as` operatore può anche utilizzato proprio come `is` operatore. Ciò è particolarmente utile se si utilizzano IDE intelligenti come FlashDevelop che forniranno un elenco di tutte le proprietà

possibili del tipo di oggetto esplicito. Esempio:

```
for (var i:int = 0; i < a.length; i++){
    var d:DisplayObject = a[i] as DisplayObject;
    if (!d) continue;
    d.//get hints here
    stage.addChild(d);
}
```

Per ottenere lo stesso effetto con `is` scrivere (leggermente meno conveniente):

```
for (var i:int = 0; i < a.length; i++){
    if (a[i] is DisplayObject != true) continue;
    var d:DisplayObject = a[i] as DisplayObject;
    stage.addChild(d);
}
```

Tieni presente che quando si controllano le codifiche con `as` operatore `as` , il valore dato verrà convertito in un tipo specificato e il risultato di tale operazione verrà controllato se non falso, quindi fai attenzione quando lo utilizzi con possibili valori falsi / NaN:

```
if(false as Boolean) trace("This will not be executed");
if(false as Boolean != null) trace("But this will be");
```

La tabella seguente mostra alcuni valori e tipi di base con il risultato di operatori di tipi. Le celle verdi valuteranno il true, il rosso al false e il greay causeranno errori di compilazione / runtime.

		Sprite	IBitmapDrawable	Object	Class	uint	int
new Sprite()	as	[object Sprite]	[object Sprite]	[object Sprite]	null	null	null
	is	true	true	true	false	false	false
	instanceof	true	false	true	false	false	false
	(<columnName>)	error	error	error	error	error	error
true	as	null	null	true	null	null	null
	is	false	false	true	false	false	false
	instanceof	false	false	true	false	false	false
	(<columnName>)	error	error	error	error	error	error
false	as	null	null	false	null	null	null
	is	false	false	true	false	false	false
	instanceof	false	false	true	false	false	false
	(<columnName>)	error	error	error	error	error	error
0.3	as	null	null	0.3	null	null	null
	is	false	false	true	false	false	false
	instanceof	false	false	true	false	false	false
	(<columnName>)	error	error	error	error	error	error
1	as	null	null	1	null	1	1
	is	false	false	true	false	true	true
	instanceof	false	false	true	false	false	false
	(<columnName>)	error	error	error	error	error	error
-1	as	null	null	-1	null	null	-1
	is	false	false	true	false	false	true
	instanceof	false	false	true	false	false	false
	(<columnName>)	error	error	error	error	error	error
NaN	as	null	null	NaN	null	null	null
	is	false	false	true	false	false	false
	instanceof	false	false	true	false	false	false
	(<columnName>)	error	error	error	error	error	error
"string"	as	null	null	string	null	null	null
	is	false	false	true	false	false	false
	instanceof	false	false	true	false	false	false
	(<columnName>)	error	error	error	error	error	error
Boolean	as	null	null	[class Boolean]	[class Boolean]	null	null
	is	false	false	true	true	false	false
	instanceof	false	false	true	true	false	false
	(<columnName>)	true	true	true	true	true	true
String	as	null	null	[class String]	[class String]	null	null
	is	false	false	true	true	false	false
	instanceof	false	false	true	true	false	false
	(<columnName>)	[class Sprite]	[class IBitmapDrawable]	[class Object]	[class Class]	[class uint]	[class i
Number	as	null	null	[class Number]	[class Number]	null	null
	is	false	false	true	true	false	false
	instanceof	false	false	true	true	false	false
	(<columnName>)	NaN	NaN	NaN	NaN	NaN	NaN
int	as	null	null	[class int]	[class int]	null	null
	is	false	false	true	true	false	false
	instanceof	false	false	true	true	false	false
	(<columnName>)	0	0	0	0	0	0
uint	as	null	null	[class uint]	[class uint]	null	null
	is	false	false	true	true	false	false
	instanceof	false	false	true	true	false	false
	(<columnName>)	0	0	0	0	0	0
Class	as	null	null	[class Class]	[class Class]	null	null
	is	false	false	true	true	false	false
	instanceof	false	false	true	true	false	false
	(<columnName>)	[class Sprite]	[class IBitmapDrawable]	[class Object]	[class Class]	[class uint]	[class i
Object	as	null	null	[class Object]	[class Object]	null	null
	is	false	false	true	true	false	false
	instanceof	false	false	true	true	false	false
	(<columnName>)	[class Sprite]	[class IBitmapDrawable]	[class Object]	[class Class]	[class uint]	[class i
IBitmapDrawable	as	null	null	[class IBitmapDrawable]	[class IBitmapDrawable]	null	null
	is	false	false	true	true	false	false
	instanceof	false	false	true	true	false	false
	(<columnName>)	error	error	error	error	error	error
Sprite	as	null	null	[class Sprite]	[class Sprite]	null	null
	is	false	false	true	true	false	false
	instanceof	false	false	true	true	false	false
	(<columnName>)	error	error	error	error	error	error

Metrici tipizzato

- Riceverai `TypeError` tipo compile se tenti di inserire valori non T nella collezione.
- Gli IDE forniscono utili informazioni sull'hint di tipo per gli oggetti all'interno di un'istanza `Vector.<T>` .

  `Vector.<T>` .

Esempi di creazione di un `Vector.<T>` :

```
var strings:Vector.<String> = new Vector.<String>(); // or
var numbers:Vector.<Number> = new <Number>[];
```

---

[1] I vettori in realtà forniscono miglioramenti delle prestazioni notevoli rispetto agli array quando si utilizzano tipi primitivi (`String` , `int` , `uint` , `Number` , ecc.).

Leggi tipi online: https://riptutorial.com/it/actionscript-3/topic/2803/tipi

Capitolo 24: Utilizzo della classe proxy

introduzione

Per prima cosa, devo dire. **C'è** una ragione per questa cosa Proxy, nonostante la sua utilità apparente, non è evidenziato abbastanza su Internet.

Non è **possibile** utilizzarlo per guardare una proprietà casuale di una classe / istanza casuale. È consentito utilizzare questa tecnica solo sottoclassando la classe **Proxy** .

Alcune operazioni non prevedono eccezioni, quindi è necessario comprendere completamente cosa stai facendo e perché lo stai facendo, e il tuo codice **deve** essere assolutamente pulito e privo di errori.

Examples

Implementazione

L'altra cosa riguardante la classe Proxy, e il motivo per cui non è così popolare, è che è piuttosto difficile capire un problema che ha esattamente bisogno di una classe dinamica con un accesso controllabile alle sue proprietà e metodi dinamici come soluzione più appropriata. Ogni volta che ho provato a utilizzare Proxy, ho finito per ricorrere a qualcos'altro, più semplice e più controllabile.

Tuttavia, non scoraggiarci. Mi piace l'idea di indirizzare gli ultimi elementi dell'array con indici [-1], [-2], ecc. In **Python** . Potrebbe non essere una grande impresa, ma è bello usare questo piuttosto che un lungo e goffo **someArray [someArray.length - 1]** . Vediamo cosa possiamo fare al riguardo.

```
package
{
    import flash.utils.Proxy;
    import flash.utils.flash_proxy;

    /**
     * Pyaray the Tentacled Whisperer of Impossible Secrets.
     */

    dynamic public class PyArray extends Proxy
    {
        private var data:Array;

        public function PyArray(...args:Array)
        {
            if (args.length == 0)
            {
                data = new Array;
            }
            else if ((args.length == 1) && (args[0] is Array))
            {
                data = args[0];
```

```
        }
        else
        {
            data = args;
        }
    }

    // This is a getter proxy to all the available Array
    // elements and properties and, sometimes, methods.
    override flash_proxy function getProperty(name:*):*
    {
        var anIndex:int = name;

        // Handle the int indices of the Array elements.
        if (anIndex == name)
        {
            // Handle the -1, -2, etc indexing.
            if (anIndex < 0) anIndex += data.length;

            if (anIndex >= data.length) return null;
            if (anIndex < 0) return null;

            return data[anIndex];
        }

        // Handle the existing public Array properties.
        if (data.hasOwnProperty(name)) return data[name];

        // Handle the Array methods addressed via ["member"] access.
        try
        {
            if (data[name] is Function) return data[name];
            else throw new Error;
        }
        catch (fail:Error)
        {
            trace("[PyArray] is unable to resolve property \"" + name + "\".");
        }

        return null;
    }

    // This will set either elements, or settable properties.
    override flash_proxy function setProperty(name:*, value:*):void
    {
        var anIndex:int = name;

        // Handle the int indices of the Array elements.
        if (anIndex == name)
        {
            // Handle the -1, -2, etc indexing.
            if (anIndex < 0) anIndex += data.length;

            // In case the element index is out of range,
            // the PyArray will extend its data Array.
            // if (anIndex >= data.length) return;
            if (anIndex < 0) return;

            data[anIndex] = value;

            return;
```

```
        }

        // Handle the existing (or dynamic) public Array properties.
        try
        {
            data[name] = value;
        }
        catch (fail:Error)
        {
            trace("[PyArray] is unable to set property \"" + name + "\".");
        }

        return;
    }

    // This allows to delete PyArray elements with "delete" operator.
    override flash_proxy function deleteProperty(name:*):Boolean
    {
        var anIndex:int = name;

        // Handle the int indices of the Array elements.
        if (anIndex == name)
        {
            // Handle the -1, -2, etc indexing.
            if (anIndex < 0) anIndex += data.length;

            if (anIndex >= data.length) return false;
            if (anIndex < 0) return false;

            data.splice(anIndex, 1);

            return true;
        }

        // Handle the dynamic public Array properties.
        try
        {
            delete data[name];
            return true;
        }
        catch (fail:Error)
        {
            trace("[PyArray] is unable to delete the \"" + name + "\" property.");
        }

        return false;
    }

    // This proxies any attempt to call a method on PyArray directly to data Array, thus
    // all Array methods (including "toString" method called through trace) are available.
    override flash_proxy function callProperty(name:*, ...rest):*
    {
        try
        {
            return (data[name] as Function).apply(data, rest);
        }
        catch (fail:Error)
        {
            trace("[PyArray] is unable to resolve method \"" + name + "\".");
            return null;
        }
```

```
        }

        // This allows PyArray to handle for..in and for..each..in loops.
        // The initial call starts with zero, so we need to do this +1 -1 magic
        // in order for enumeration to work correctly. I'm not happy with this either.
        override flash_proxy function nextNameIndex(index:int):int
        {
            if (index >= data.length) return 0;
            else return index + 1;
        }

        // This method handles the for..in loop.
        override flash_proxy function nextName(index:int):String
        {
            return (index - 1).toString();
        }

        // This method handles the for..each..in loop.
        override flash_proxy function nextValue(index:int):*
        {
            return data[index - 1];
        }
    }
}
```

uso

```
package
{
    import flash.display.Sprite;

    /**
     * Daemonette of Slaanesh.
     *
     * It is minor female demon, vaguely human-like, but with crab-like pincers instead of
hands.
     * She wears a rather indecent skimpy leather bikini, moves quickly and casts deadly
spells!
     */

    public class Slaanesh extends Sprite
    {
        public function Slaanesh()
        {
            // Lets initialize the PyArray.
            var PA:PyArray = new PyArray(1,2,3,4,5,4,3,2,1,"Foo");

            // Basic check: get the last element.
            trace(PA[-1]);
            // output:
            // Foo

            // This will map to the 0-based third element.
            trace(PA[2.0]);
            // output:
            // 3

            // This should not get us anywhere.
            trace(PA[2.1]);
```

```
// output:
// [PyArray] is unable to resolve property "2.1".
// null

// This should return the length of the data Array.
trace(PA["length"]);
// output:
// 10

// This should return the length of the data Array.
// This will not compile unless PyArray class is marked "dynamic".
trace(PA.length);
// output:
// 10

// This will map to indexOf method of data Array via getProperty method.
trace(PA["indexOf"]);
// output:
// function Function() {}

// This will map to indexOf method of data Array via getProperty method.
// This will not compile unless PyArray class is marked "dynamic".
trace(PA.indexOf);
// output:
// function Function() {}

// This is a try to access a non-existent property.
// This will not compile unless PyArray class is marked "dynamic".
trace(PA.P124);
// output:
// [PyArray] is unable to resolve property "P124".
// null

// This is a try to call a non-existent method via callProperty method.
// This will not compile unless PyArray class is marked "dynamic".
trace(PA.P124());
// output:
// [PyArray] is unable to resolve method "P124".
// null

// Basic check: calling a proxied method via callProperty method.
trace(PA.indexOf(5));
// output:
// 4

// An attempt to replace an Array method with a random value.
// This will not compile unless PyArray class is marked "dynamic".
PA.indexOf = 123;
// output:
// [PyArray] is unable to set property "indexOf".

// An attempt to assign a random value to a random property.
// It will succees because Array, as a dynamic class, allows so.
// This will not compile unless PyArray class is marked "dynamic".
PA.indexOfz = 123;
trace(PA.indexOfz);
// output:
// 123

// An attempt to assign an Array element via negative indexing.
// This trace works fine because toString method is also proxied.
```

```
            PA[-3] = "Hello";
            trace(PA);
            // output:
            // 1,2,3,4,5,4,3,Hello,1,Foo

            // An attempt to delete Array elements via normal and negative indexing.
            // This operation is mapped via deleteProperty method.
            delete PA[-4]; // deletes "3" before "Hello"
            delete PA[0];  // deletes "1" at the start.
            trace(PA);
            // output:
            // 2,3,4,5,4,Hello,1,Foo

            // An attempt to delete a non-dynamic method reference.
            // There's no error output, AS3 must be handling this internally.
            delete PA.indexOf;
            trace(PA.indexOf);
            // output:
            // function Function() {}

            // An attempt to set an element out of index range.
            PA[10] = "123abc";
            trace(PA);
            // output:
            // 2,3,4,5,4,Hello,1,Foo,,,123abc

            var aText:String;

            // This is a test of for..in loop, Array elements are
            // enumerated via nextName and nextNameIndex methods.
            aText = ""

            for (var aKey:String in PA)
                aText += aKey + ":" + PA[aKey] + " ";

            trace(aText);
            // output:
            // 0:2 1:3 2:4 3:5 4:4 5:Hello 6:1 7:Foo 8:undefined 9:undefined 10:123abc

            // This is a test of for..each..in loop, Array elements are
            // enumerated via nextValue and nextNameIndex methods.
            aText = "";

            for each (var aValue:* in PA)
                aText += aValue + " ";

            trace(aText);
            // output:
            // 2 3 4 5 4 Hello 1 Foo undefined undefined 123abc
        }
    }
}
```

Leggi Utilizzo della classe proxy online: https://riptutorial.com/it/actionscript-3/topic/10631/utilizzo-della-classe-proxy

Capitolo 25: Visualizza elenco ciclo di vita

Osservazioni

Le animazioni basate su frame in Flash e AIR implementano il seguente ciclo di vita:

- `Event.ENTER_FRAME` viene inviato
- Viene eseguito il codice costruttore di oggetti di visualizzazione per bambini
- `Event.FRAME_CONSTRUCTED` viene inviato
- Le azioni del frame nel simbolo `MovieClip` vengono eseguite
- Azioni frame nei bambini I simboli `MovieClip` vengono eseguiti
- `Event.EXIT_FRAME` viene inviato
- `Event.RENDER` viene inviato

Examples

Aggiunto e rimosso dal ciclo di vita della fase

```
package {
import flash.display.Sprite;
import flash.events.Event;

public class Viewport extends Sprite {

    /** Constructor */
    public function Viewport() {
        super();

        // Listen for added to stage event
        addEventListener(Event.ADDED_TO_STAGE, addedToStageHandler);
    }

    /** Added to stage handler */
    protected function addedToStageHandler(event:Event):void {
        // Remove added to stage event listener
        removeEventListener(Event.ADDED_TO_STAGE, addedToStageHandler);

        // Listen for removed from stage event
        addEventListener(Event.REMOVED_FROM_STAGE, removedFromStageHandler);
    }

    /** Removed from stage handler */
    protected function removedFromStageHandler(event:Event):void {
        // Remove removed from stage event listener
        removeEventListener(Event.REMOVED_FROM_STAGE, removedFromStageHandler);

        // Listen for added to stage event
        addEventListener(Event.ADDED_TO_STAGE, addedToStageHandler);
    }

    /** Dispose */
    public function dispose():void {
```

```
        // Remove added to stage event listener
        removeEventListener(Event.ADDED_TO_STAGE, addedToStageHandler);

        // Remove removed from stage event listener
        removeEventListener(Event.REMOVED_FROM_STAGE, removedFromStageHandler);
    }

}
}
```

Leggi Visualizza elenco ciclo di vita online: https://riptutorial.com/it/actionscript-3/topic/1877/visualizza-elenco-ciclo-di-vita

Titoli di coda

S. No	Capitoli	Contributors
1	Introduzione a ActionScript 3	BadFeelingAboutThis, Community, Jason Sturges, joshtynjala, Kit Grose, null, Programmer Dancuk, Vesper
2	Caricamento di file esterni	BadFeelingAboutThis, Marty
3	Dati binari	Paweł Audionysos
4	Disegno di bitmap	BadFeelingAboutThis, Marty, Vesper, www0z0k
5	Generazione di valore casuale	alebianco, BadFeelingAboutThis, HITMAN, Jason Sturges, Marty, mnoronha, null, Vesper, xims
6	Informazioni su "Errore 1009: impossibile accedere a una proprietà oa un metodo di riferimento a un oggetto nullo"	Vesper, www0z0k
7	Invio e ricezione di dati dai server	Marty, null, xims
8	Lavorare con data e ora	Jason Sturges, mnoronha
9	Lavorare con gli eventi	blue112, Jason Sturges, mnoronha, null, VC.One, Vesper, Zze
10	Lavorare con gli oggetti di visualizzazione	BadFeelingAboutThis, Jason Sturges, Vesper
11	Lavorare con i timer	Jason Sturges, mnoronha, Vesper, www0z0k
12	Lavorare con i video	VC.One, www0z0k
13	Lavorare con il suono	BadFeelingAboutThis, blue112, Paweł Audionysos, VC.One
14	Lavorare con la geometria	Marty, mnoronha, www0z0k

15	Lavorare con la timeline	Marty
16	Lavorare con valori numerici	Jason Sturges, Jonny Henly, Marty, Vesper
17	Manipolazione e filtraggio bitmap	payam_sbr, VC.One
18	Nozioni di base sullo sviluppo del gioco	payam_sbr
19	Ottimizzazione delle prestazioni	Community, Marty
20	Progettazione di applicazioni reattive	BadFeelingAboutThis, null, Vesper
21	Programmazione orientata agli oggetti	HITMAN, Marty, null, www0z0k
22	Singleton Pattern	commovere, Jason Sturges, mnoronha, null
23	tipi	BadFeelingAboutThis, joshtynjala, Marty, Paweł Audionysos
24	Utilizzo della classe proxy	Organis
25	Visualizza elenco ciclo di vita	Jason Sturges, mnoronha