EBook Gratis

APRENDIZAJE ActionScript 3

Free unaffiliated eBook created from
Stack Overflow contributors.

#actionscrip

t-3

Tabla de contenido

Acerca de ... 1

Capítulo 1: Comenzando con ActionScript 3 ... 2

Observaciones ... 2

Versiones ... 2

Hay una versión única de Actionscript 3, llamada "ActionScript 3.0" ... 2

Examples ... 3

Descripción general de la instalación ... 3

Hola Mundo ... 3

Instalación flash desarrollada ... 4

Instalación de Apache Flex ... 5

Construyendo proyectos Flex o Flash en la línea de comando usando mxmlc ... 5

Un ejemplo de "Hello World" mostrado ... 6

Capítulo 2: Cargando archivos externos ... 7

Observaciones ... 7

Examples ... 7

Carga de imágenes externas / SWF con el cargador ... 7

Cargando un archivo de texto con FileStream (solo AIR runtime) ... 8

Capítulo 3: Datos binarios ... 9

Examples ... 9

Lectura de forma ByteArray a través de la interfaz IDataInput ... 9

Capítulo 4: Dibujo de mapas de bits ... 10

Examples ... 10

Dibuje un objeto de visualización en datos de mapa de bits ... 10

Dibuja un objeto de visualización con cualquier coordenada de punto de registro ... 10

Animando una hoja de sprites ... 11

Capítulo 5: Diseño de aplicación sensible ... 12

Examples ... 12

Solicitud de respuesta básica ... 12

Realiza procesos largos y no consigue aplicaciones que no responden ... 12

Capítulo 6: Entendiendo el "Error 1009: No se puede acceder a una propiedad o método de re ... 14

Introducción 14
Observaciones 14
Examples 14
Etapa no está disponible 14
Encasillado inválido 15
Objeto no ilustrado 15
Expresión multi-nivel 15
Resultado de la función sin procesar 15
Oyente del evento olvidado 15
Referencia no válida a un objeto basado en marco 16
Capítulo 7: Enviando y recibiendo datos de servidores 18
Examples 18
Hacer una solicitud desde Flash 18
Agregando variables a su solicitud 18
Alterar el método HTTP (GET, POST, PUT, etc.) 18
Mis datos de respuesta son siempre nulos, ¿qué significa "asíncrono"? 19
Peticiones de dominio cruzado 19
Capítulo 8: Fundamentos de desarrollo de juegos 21
Introducción 21
Examples 21
Carácter isométrico animando + movimiento. 21
los conceptos utilizados en este ejemplo: 21
Recursos: (no hay permiso para utilizar estos recursos con fines comerciales) 21
Código y Comentarios: 22
Referencias externas: 27
Capítulo 9: Generación de valor aleatorio 28
Examples 28
Número aleatorio entre 0 y 1 28
Número aleatorio entre valores mínimo y máximo 28
Angulo aleatorio, en grados 28
Valor aleatorio de una matriz 29
Punto aleatorio dentro de un círculo 29

Ángulo aleatorio, en radianes ... 30
Determinar el éxito de una operación de "porcentaje de probabilidad" ... 30
Crear un color al azar ... 30
Recorrer aleatoriamente el alfabeto ... 31
Aleatorizar una matriz ... 31
Capítulo 10: Los tipos ... 32
Examples ... 32
Tipo de fundición ... 32
El tipo de función ... 32
El tipo de clase ... 32
Anotando tipos ... 33
Revisando los tipos ... 33
Arrays mecanografiados ... 35
Capítulo 11: Manipulación de mapa de bits y filtrado ... 37
Introducción ... 37
Examples ... 37
Efecto umbral (monocromo) ... 37
necesario: ... 37
Capítulo 12: Mostrar lista de ciclo de vida ... 39
Observaciones ... 39
Examples ... 39
Añadido y eliminado del ciclo de vida de la etapa ... 39
Capítulo 13: Optimizando el rendimiento ... 41
Examples ... 41
Gráficos basados en vectores ... 41
Texto ... 41
Vector y para cada vs matrices y para ... 42
Eliminación rápida de elementos de matriz ... 42
Vectores en lugar de matrices ... 43
Reutilizando y agrupando gráficos ... 43
Capítulo 14: Patrón Singleton ... 45
Observaciones ... 45

Examples ... 45
Ejecutor Singleton a través de instancia privada ... 45
Capítulo 15: Programación orientada a objetos ... 46
Examples ... 46
Constructor "sobrecargado" a través del método estático ... 46
establecer y obtener funciones ... 46
Paquetes ... 47
Método de anulación ... 48
getter y setter ... 49
Capítulo 16: Trabajando con eventos ... 51
Observaciones ... 51
Examples ... 51
Eventos personalizados con datos de eventos ... 51
Manejo de eventos fuera de la lista de visualización ... 52
Manejo básico de eventos ... 52
Añade tus propios eventos ... 53
Estructura de evento de ratón simple ... 54
Capítulo 17: Trabajando con fecha y hora ... 55
Examples ... 55
Ante meridiem (AM) o Post meridiem (PM) para un reloj de 12 horas ... 55
Número de días en el mes especificado ... 55
Si el año especificado es año bisiesto ... 55
Si se observa el horario de verano ... 55
Fecha de inicio de hoy, a medianoche ... 55
Capítulo 18: Trabajando con la geometría ... 57
Examples ... 57
Obteniendo el ángulo entre dos puntos ... 57
Conseguir la distancia entre dos puntos ... 57
Convertir radianes a grados ... 57
Convertir grados a radianes ... 57
El valor de un círculo en grados y radianes ... 58
Moviendo un punto a lo largo de un ángulo ... 58

Determine si un punto está dentro de un área rectangular 58
Capítulo 19: Trabajando con sonido 60
Sintaxis 60
Examples 60
Deja de reproducir un sonido 60
Bucle infinito un sonido 60
Cargar y reproducir un sonido externo 61
Capítulo 20: Trabajando con temporizadores 62
Examples 62
Ejemplo de temporizador de cuenta regresiva 62
Intervalos y tiempos de espera 63
Ejemplo de temporizador aleatorio 64
Capítulo 21: Trabajando con Timeline 66
Examples 66
Hacer referencia a la línea de tiempo principal o la clase de documento desde otros MovieC 66
Capítulo 22: Trabajando con valores numéricos. 67
Examples 67
Si un número es un valor par 67
Si un número es un valor impar 67
Redondeo a la X más cercana 67
Errores de redondeo de números de punto flotante 68
Visualización de números con la precisión requerida 68
Capítulo 23: Trabajando con video 69
Examples 69
Cargar y reproducir archivo de video externo 69
Con NetStatusEvent 69
Capítulo 24: Trabajar con objetos de visualización 71
Sintaxis 71
Observaciones 71
Examples 71
Introducción a la lista de visualización 71
Orden Z / Capas 72

Eliminar objetos de visualización 73
Eventos 74
Adobe Animate / Flash Professional 74
Capas 74
Eliminar todos los objetos de la lista de visualización 74
Transición de cuadros a cambio manual de contenido 75
Capítulo 25: Usando la clase Proxy 77
Introducción 77
Examples 77
Implementación 77
Uso 80
Creditos 83

Acerca de

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

Capítulo 1: Comenzando con ActionScript 3

Observaciones

ActionScript 3 es el lenguaje de programación para los entornos de ejecución de Adobe Flash Player y Adobe AIR. Es un lenguaje basado en ECMAScript orientado a objetos que se utiliza principalmente para el desarrollo de aplicaciones nativas en dispositivos de escritorio (Windows / Mac) y móviles (iOS / Android).

Recursos de aprendizaje de Adobe: http://www.adobe.com/devnet/actionscript/learning.html

Historia y más detalles: https://en.wikipedia.org/wiki/ActionScript

Documentación en línea sobre clases y referencias: http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/package-detail.html

Versiones

Hay una versión única de Actionscript 3, llamada "ActionScript 3.0"

Versión flash	Nombre clave	Cambios y mejoras	Fecha de lanzamiento
Flash Player 9.x	Zaphod	Versión inicial	2006-06-22
Flash Player 10.0	Astro	introdujo el tipo `Vector.<T>` , los filtros de sombreado Adobe Pixel Bender en la clase `flash.filters.ShaderFilter` y su compatibilidad de hardware en varias CPU.	2008-10-15
Flash Player 10.1	Argo	introduje la clase `flash.events.TouchEvent` para trabajar con dispositivos multitáctiles y otro soporte de hardware para dispositivos móviles, como el acelerómetro.	2010-06-10
Flash Player 10.2	Picante	introdujo la clase `flash.media.StageVideo` y el marco general para trabajar con la reproducción de video por etapas en AS3.	2011-02-08
Flash Player 11	Serrano	agrega soporte H.264 a la transmisión de video sobre objetos `NetStream` en ambas direcciones.	2011-10-04

Versión flash	Nombre clave	Cambios y mejoras	Fecha de lanzamiento
		También agrega soporte SSL / TLS para conexión Flash con clase `SecureSocket` .	
Flash Player 11.4	Brannan	introdujo la clase `flash.system.Worker` y la capacidad de delegar el trabajo asíncrono a otros subprocesos en el cliente.	2012-08-10
Flash Player 11.8	Harrison	se eliminó el soporte de hardware (compilación JIT) para los filtros de sombreado de Adobe Pixel Bender, lo que reduce drásticamente el rendimiento de cualquier ejecución del filtro de sombreado PB.	2013-05-09

Examples

Descripción general de la instalación

ActionScript 3 se puede usar al instalar el SDK de Adobe AIR o el SDK de Apache Flex o como parte del producto Animate CC de Adobe *(anteriormente conocido como Flash Professional)* .

Adobe Animate CC es una solución de software profesional que se puede utilizar para crear proyectos AS3 utilizando herramientas visuales; una vez instalada, no es necesario seguir ningún paso para comenzar a crear proyectos AS3.

AIR SDK y Flex SDK se pueden usar con herramientas de línea de comandos o con varios IDE de terceros.

Además de Adobe Animate CC, hay otros cuatro IDEs populares capaces de trabajar con AS3. Estos IDE tienen sus propias instrucciones sobre cómo comenzar.

- Flash Builder (por Adobe - basado en Eclipse)
- IntelliJ IDEA (Por Jetbrains)
- FlashDesarrollar
- FDT (Eclipse Plugin)

Hola Mundo

Una clase de documento de ejemplo que imprime "Hola, Mundo" en la consola de depuración cuando se crea una instancia.

```
import flash.display.Sprite;

public class Main extends Sprite {

    public function Main() {
        super();
```

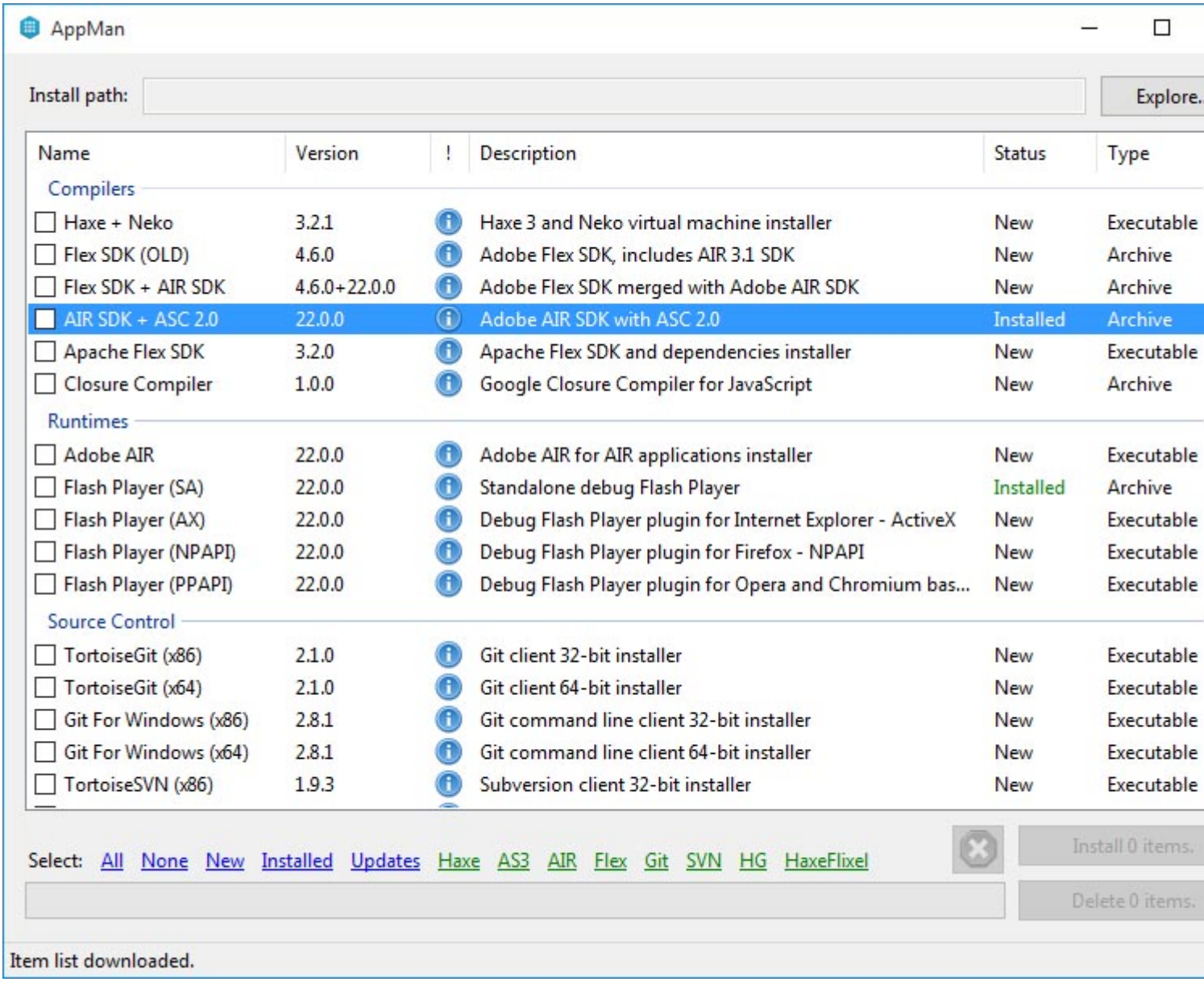
```
        trace("Hello, World");
    }

}
```

Instalación flash desarrollada

FlashDevelop es un IDE multiplataforma de código abierto creado en 2005 para desarrolladores de Flash. Sin costo, es una forma muy popular de comenzar a desarrollar con AS3.

Para instalar FlashDevelop:

1. Descarga el archivo de instalación y ejecuta el instalador.
2. Una vez completada la instalación, ejecute FlashDevelop. En el primer lanzamiento, debería aparecer la ventana de `App Man` que le pide que elija qué SDK y qué herramientas instalar.



Si AppMan no se abre automáticamente, o si desea agregar algo más tarde, ábralo seleccionando 'Instalar software' en el menú 'Herramientas'.

Verifique el elemento **AIR SDK + ACS 2.0** (en la sección 'Compilador') y el elemento **Flash Player (SA)** en la sección 'Tiempo de ejecución' (más cualquier otra cosa que desee instalar). Haga clic en el botón de instalación.

3. Una vez que se instala el SDK, probemos creando un proyecto de hello world. Comience creando un nuevo proyecto (desde el menú *Proyecto*)
4. Elija el **proyector AIR AS3** de la lista y asígnele un nombre / ubicación.
5. En el panel del administrador de proyectos (seleccione 'Administrador de proyectos' en el menú de vista si no está visible), expanda la carpeta **src** y abra el archivo `Main.as`
6. En el archivo `Main.as` , ahora puede crear un primer programa de ejemplo como Hello World
7. Ejecute su proyecto haciendo clic en el ícono de reproducción, o presionando `F5` , o `Ctrl+Enter` . El proyecto se compilará y cuando termine, aparecerá una ventana en blanco (esta es su aplicación). En la ventana de salida de FlashDevelop, debería ver las palabras: **Hola mundo** .

¡Ya está listo para comenzar a desarrollar aplicaciones AS3 con FlashDevelop!

Instalación de Apache Flex

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

1. Descarga el instalador del SDK
2. Ejecute el instalador del SDK. La primera pregunta que se le hará es el directorio de instalación.

   - en una Mac, use `/Applications/Adobe Flash Builder 4.7/sdks/4.14.0/`
   - en una PC, use `C:\Program Files(x86)\Adobe Flash Builder 4.7\sdks\4.14.0`

   Tendrá que crear las carpetas 4.14.0. Presione Siguiente. Aceptar licencias de SDK e instalar.

Instrucciones específicas de IDE para la configuración de Apache Flex:

- Flash Builder
- IntelliJ IDEA
- FlashDesarrollar
- FDT

Construyendo proyectos Flex o Flash en la línea de comando usando mxmlc

El compilador Flex (`mxmlc`) es una de las partes más importantes del SDK de Flex. Puede editar el código AS3 en cualquier editor de texto que desee. Cree un archivo de clase principal que se extienda desde `DisplayObject` .

Puede desencadenar compilaciones en la línea de comando de la siguiente manera:

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

Si necesita compilar un proyecto de Flash (a diferencia de Flex), puede agregar una referencia a la biblioteca de Flash de la siguiente manera (deberá tener instalado el IDE de Adobe Animate):

```
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"
```

O en 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 ejemplo de "Hello World" mostrado

```
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);
        }
    }
}
```

Esta clase usa la clase `TextField` para mostrar el texto.

Lea Comenzando con ActionScript 3 en línea: https://riptutorial.com/es/actionscript-3/topic/1065/comenzando-con-actionscript-3

Capítulo 2: Cargando archivos externos

Observaciones

En algunos casos, su aplicación no puede manipular el contenido de los recursos cargados desde un recurso externo (por ejemplo, transformar imágenes). No puedo recordar con certeza cuáles son, pero estoy bastante seguro de que está relacionado con la carga de contenido entre dominios.

Examples

Carga de imágenes externas / SWF con el cargador

1. Crear un objeto Loader:

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

2. Añadir escuchas en el cargador. Los estándares están completos y los errores de seguridad / io

```
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. Cargue el archivo deseado:

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

4. Crea tus manejadores de eventos:

```
 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
 }
```

> La carga con la clase Loader es asíncrona. Esto significa que después de llamar a `loader.load` la aplicación continuará ejecutándose mientras se carga el archivo. El contenido de su cargador no estará disponible hasta que el cargador `Event.COMPLETE` evento `Event.COMPLETE` .

---

importaciones necesarias:

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

Cargando un archivo de texto con FileStream (solo AIR runtime)

Un ejemplo simple sobre cómo leer un archivo de texto UTF de forma síncrona.

```
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();
```

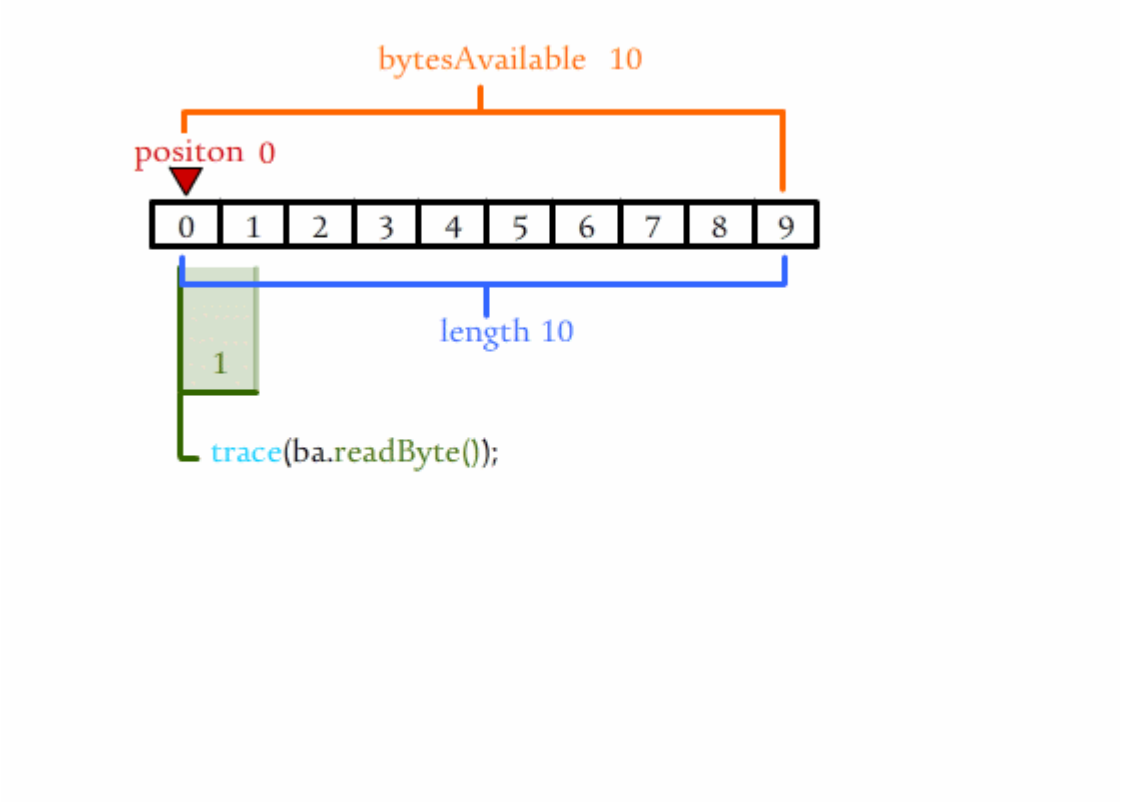
Lea Cargando archivos externos en línea: https://riptutorial.com/es/actionscript-3/topic/1694/cargando-archivos-externos

Capítulo 3: Datos binarios

Examples

Lectura de forma ByteArray a través de la interfaz IDataInput.

La siguiente animación muestra lo que sucede cuando utiliza los métodos de la interfaz IDataInput para acceder a los datos de ByteArray y otras clases que implementan esta interfaz.



Lea Datos binarios en línea: https://riptutorial.com/es/actionscript-3/topic/9464/datos-binarios

Capítulo 4: Dibujo de mapas de bits

Examples

Dibuje un objeto de visualización en datos de mapa de bits

Una función auxiliar para crear una copia de mapa de bits de un objeto. Esto se puede usar para convertir objetos vectoriales, texto o Sprite anidados complejos en un mapa de bits aplanado.

```
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);
```

Dibuja un objeto de visualización con cualquier coordenada de punto de registro

```
    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 al margen: para que `getBounds()` devuelva valores válidos, el objeto debe tener acceso al escenario al menos una vez; de lo contrario, los valores son falsos. Se puede agregar un código para asegurar que la `source` pasada tenga etapa, y si no, se puede agregar a la etapa y luego quitarse nuevamente.

Animando una hoja de sprites

Una hoja de sprites por definición es un mapa de bits que contiene una animación determinada. Los juegos antiguos utilizan hojas de sprites de tipo de cuadrícula, es decir, cada cuadro ocupa una región igual, y los bordes están alineados por los bordes para formar un rectángulo, probablemente con algunos espacios desocupados. Más tarde, para minimizar el tamaño del mapa de bits, las hojas de sprites se "empaquetan" eliminando espacios en blanco adicionales alrededor del rectángulo que contiene cada fotograma, pero cada fotograma es un rectángulo para simplificar las operaciones de copia.

Para animar una hoja de sprites, se pueden utilizar dos técnicas. Primero, puede usar `BitmapData.copyPixels()` para copiar una determinada región de su hoja de sprite en un `Bitmap` visualizado, produciendo un personaje animado. Este enfoque es mejor si utiliza un solo `Bitmap` mostrado que hospeda toda la imagen.

```
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 segunda técnica se puede utilizar si tiene muchos `Sprite` o `Bitmap` en la lista de visualización, y comparten la misma hoja de sprites. Aquí, el búfer de destino ya no es un solo objeto, sino que cada objeto tiene su propio búfer, por lo que la estrategia adecuada es manipular la propiedad `bitmapData` los mapas de bits. Sin embargo, antes de hacer esto, la hoja de sprites se debe cortar en marcos individuales.

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

Lea Dibujo de mapas de bits en línea: https://riptutorial.com/es/actionscript-3/topic/2814/dibujo-de-mapas-de-bits

Capítulo 5: Diseño de aplicación sensible

Examples

Solicitud de respuesta básica

```
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
        }
    }
}
```

Realiza procesos largos y no consigue aplicaciones que no responden.

Hay situaciones en las que necesita calcular algo realmente grande en su aplicación Flash, sin interrumpir la experiencia del usuario. Para esto, necesita diseñar su largo proceso como un proceso de varios pasos con estado guardado entre iteraciones. Por ejemplo, debe realizar una actualización en segundo plano de muchos objetos internos, pero si desea actualizarlos todos a la vez con un simple `for each (var o in objects) { o.update(); }` , Flash brevemente (o no tan brevemente) deja de responder al usuario. Por lo tanto, debe realizar una o varias actualizaciones por fotograma.

```
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
    }
}
```

El procesamiento avanzado puede incluir usar `getTimer()` para verificar el tiempo transcurrido y permitir otra iteración si el tiempo no se agotó, dividiendo `update()` en varias funciones si la actualización es demasiado larga, mostrando el progreso en otros lugares, ajustando el proceso de iteración para adaptarse a los cambios del Lista de objetos a procesar, y muchos más.

Lea Diseño de aplicación sensible en línea: https://riptutorial.com/es/actionscript-3/topic/1615/diseno-de-aplicacion-sensible

Capítulo 6: Entendiendo el "Error 1009: No se puede acceder a una propiedad o método de referencia de objeto nulo"

Introducción

Un error 1009 es un error general que surge cuando intenta recibir un valor de una variable o propiedad que tiene un valor `null` . Los ejemplos proporcionados exponen varios casos donde surge este error, junto con algunas recomendaciones sobre cómo mitigar el error.

Observaciones

El temido y frecuentemente preguntado "Error 1009: No se puede acceder a una propiedad o método de referencia de objeto nulo" es una señal de que algunos de los datos aparecen nulos, pero se intentó utilizarlos como un objeto poblado. Hay muchos tipos de problemas que pueden causar este comportamiento, y cada uno debe compararse con el código donde surgió el error.

Examples

Etapa no está disponible

A veces, los desarrolladores escriben algún código que desea acceder a la `stage` o etapa de Flash para agregar oyentes. Puede funcionar por primera vez, luego, de repente, deja de funcionar y produce el error 1009. El código en cuestión puede incluso estar en la línea de tiempo, ya que es la primera iniciativa para agregar código allí, y muchos tutoriales que aún existen usan Capa de código de línea de tiempo para colocar el código.

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

La razón por la que este código no funciona es simple: primero se crea una instancia del objeto de visualización, luego se agrega a la lista de visualización, y mientras está fuera de la lista de visualización, la `stage` es nula.

Peor aún si el código de esta manera:

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

se coloca en la línea de tiempo. Incluso puede funcionar durante algún tiempo, mientras que el objeto `Main` se golpea para poner en escena a través de GUI. Luego, su SWF se carga desde otro SWF y, de repente, el código se rompe. Esto sucede porque los cuadros del `Main` se construyen

de una manera diferente cuando el SWF es cargado directamente por el jugador y cuando la carga se procesa de forma asíncrona. La solución es utilizar el escucha `Event.ADDED_TO_STAGE` , colocar todo el código que se ocupa de la etapa, y colocar el oyente en un archivo AS en lugar de la línea de tiempo.

Encasillado inválido

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

Si dicho oyente se adjunta a un objeto que no es descendiente de `MovieClip` (por ejemplo, un `Sprite`), la conversión a error fallará y cualquier operación posterior con su resultado arrojará el error 1009.

Objeto no ilustrado

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

Aquí, se declara una referencia de objeto, pero nunca se le asigna un valor, ya sea con una `new` o con una asignación de un valor no nulo. Solicitar sus propiedades o método resulta en un error 1009.

Expresión multi-nivel

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

Aquí, cualquier objeto anterior al punto puede terminar siendo nulo, y el uso de métodos que devuelven objetos complejos solo aumenta la complicación para depurar el error nulo. En el peor de los casos, si el método es propenso a fallas extrañas, por ejemplo, recuperar datos a través de la red.

Resultado de la función sin procesar

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

`getChildByName()` es una de las muchas funciones que pueden devolver nulo si se produce un error al procesar su entrada. Por lo tanto, si está recibiendo un objeto de cualquier función que posiblemente pueda devolver nulo, primero verifique si hay nulo. Aquí, una propiedad se consulta instantáneamente sin verificar primero si `s` es nulo, esto generará el error 1009.

Oyente del evento olvidado

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

Este ejemplo moverá el `childMC` (agregado a `Main` en el tiempo de diseño) pero lanzará instantáneamente un 1009 tan pronto como se `gotoAndStop()` , si ese `childMC` no existe en el cuadro 2. La razón principal de esto es que siempre que pasa una cabeza de reproducción un cuadro clave (un cuadro que no hereda el conjunto de objetos del cuadro anterior), ya sea mediante `gotoAndStop()` , `gotoAndPlay()` con el cuadro de destino separado del cuadro actual por un cuadro clave, o por la reproducción normal si el SWF es un En la animación, los contenidos del marco actual se **destruyen** y los nuevos contenidos se crean utilizando los datos almacenados desde la GUI. Por lo tanto, si el nuevo marco no tiene un elemento secundario denominado `childMC` , la solicitud de propiedad devolverá un valor nulo y se lanzará 1009.

El mismo principio se aplica si agrega dos escuchas de eventos, pero elimina solo una, o agrega una escucha a un objeto, pero intente eliminar de otro. La llamada `removeEventListener` no le avisará si el objeto no tiene un detector de eventos respectivo adjunto, así que lea el código que agrega y elimina los escuchas de eventos con cuidado.

También `setInterval()` cuenta: al usar objetos del `Timer` , al llamar a `setInterval()` y a `setTimeout()` también se crean escuchas de eventos, y estos también deben borrarse correctamente.

Referencia no válida a un objeto basado en marco

A veces, se llama a `gotoAndStop()` en medio del código que hace referencia a algunas propiedades basadas en marcos. Pero, **justo después de cambiar el marco,** todos los enlaces a las propiedades que existían en el marco actual se invalidan, por lo que cualquier procesamiento que los involucre debe terminarse de inmediato.

Hay dos escenarios generales de tal procesamiento: Primero, un bucle no termina después de la llamada `gotoAndStop()` , como aquí:

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

Aquí, un error 1009 significa que el MC del `player` fue destruido al procesar la llamada `gotoAndStop()` , pero el bucle continúa, y refiere el enlace ahora nulo para obtener `hitTestObject()` . Si la condición diría `if (bullet.hitTestObject(player))` lugar, el error sería # 2007 "El parámetro hitTestObject no debe ser nulo". La solución es colocar una declaración de `return` justo después de llamar a `gotoAndStop()` .

El segundo caso son múltiples escuchas de eventos en el mismo evento. Me gusta esto:

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

```
stage.addEventListener(Event.ENTER_FRAME,func2);
function func1(e:Event):void {
    if (condition()) {
        gotoAndStop(2);
    }
}
```

Aquí, si la `condition()` es verdadera, el primer oyente ejecutará `gotoAndStop()` , pero el segundo escuchador aún se ejecutará, y si esa persona hace referencia a los objetos en el marco, se lanzará un error 1009. La solución es evitar múltiples escuchas en un solo evento, en un solo objeto, es mejor tener un oyente que maneje todas las situaciones en ese evento y pueda terminar adecuadamente si se necesita un cambio de marco.

Lea Entendiendo el "Error 1009: No se puede acceder a una propiedad o método de referencia de objeto nulo" en línea: https://riptutorial.com/es/actionscript-3/topic/2098/entendiendo-el--error-1009--no-se-puede-acceder-a-una-propiedad-o-metodo-de-referencia-de-objeto-nulo-

Capítulo 7: Enviando y recibiendo datos de servidores

Examples

Hacer una solicitud desde Flash

Las clases `URLRequest` y `URLLoader` trabajan juntas para realizar solicitudes de Flash a recursos externos. La `URLRequest` define información sobre la solicitud, por ejemplo, el cuerpo de la solicitud y el tipo de método de solicitud, y el `URLLoader` referencia a esto para realizar la solicitud real y proporcionar un medio de notificación cuando se recibe una respuesta del recurso.

Ejemplo:

```
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.
}
```

Agregando variables a su solicitud

La clase `URLVariables` permite definir los datos que se enviarán junto con una `URLRequest` .

Ejemplo:

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

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

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

Puede enviar la solicitud a través de un `URLLoader` o abrir la URL de la solicitud con las variables adjuntas en la cadena de consulta mediante el comando `navigateToURL` .

Alterar el método HTTP (GET, POST, PUT, etc.)

La clase `URLRequestMethod` contiene constantes para los distintos tipos de solicitud que puede realizar. Estas constantes deben asignarse a la propiedad del `method` `URLRequest` :

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

```
request.method = URLRequestMethod.POST;
```

> Tenga en cuenta que solo `GET` y `POST` están disponibles fuera del tiempo de ejecución de AIR.

Mis datos de respuesta son siempre nulos, ¿qué significa "asíncrono"?

Cuando Flash realiza una solicitud de datos de una fuente externa, esa operación es *asíncrona* . La explicación más básica de lo que esto significa es que los datos se cargan "en segundo plano" y activan el controlador de eventos que asigna a `Event.COMPLETE` cuando se recibe. Esto puede suceder en cualquier momento de la vida de su aplicación.

Sus datos **NO** estarán disponibles inmediatamente después de llamar a `load()` en su `URLLoader` . **Debe** adjuntar un detector de eventos para `Event.COMPLETE` e interactuar con la respuesta allí.

```
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.
```

No puedes evitar esto con pequeños trucos como usar `setTimeout` o similar:

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

Peticiones de dominio cruzado

Flash no cargará datos de un dominio que no sea el que está ejecutando su aplicación a menos que ese dominio tenga una política de dominio cruzado XML ya sea en la raíz del dominio (por ejemplo, `http://somedomain.com/crossdomain.xml`) o en algún lugar donde se encuentre puede apuntar con `Security.loadPolicyFile()` . El archivo crossdomain.xml es donde puede especificar dominios que pueden solicitar datos a su servidor desde una aplicación Flash.

Ejemplo del crossdomain.xml *más permisivo* :

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

Tenga en cuenta que este ejemplo **no debe utilizarse en entornos de producción** , use una instancia más restrictiva.

Un crossdomain.xml específico más restrictivo se verá así, por ejemplo:

```
<?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>
```

Recursos:

- La especificación del archivo de política de crossdomain .

Lea Enviando y recibiendo datos de servidores en línea: https://riptutorial.com/es/actionscript-3/topic/1893/enviando-y-recibiendo-datos-de-servidores

Capítulo 8: Fundamentos de desarrollo de juegos

Introducción

[! [ingrese la descripción de la imagen aquí] [1]] [1] **conceptos básicos** del desarrollo de juegos. ---------- *Nota* : este conjunto de tutoriales / artículos contiene muchos conceptos que pueden proporcionarse antes como temas separados. tenemos que refrescarlos mentalmente y aprender un poco de la implementación de las partes más críticas de un videojuego a través de actionscript-3. [1]: https://i.stack.imgur.com/CUIsz.png

Examples

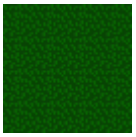
Carácter isométrico animando + movimiento.

① los conceptos utilizados en este ejemplo:

Clase	Uso
`URLRequest` + `Loader` + `Event`	Cargando mapa atlas (sprite) desde la ruta externa.
`BitmapData` + `Sprite` + `beginBitmapFill` + `Matrix` + `stageWidth & stageHeight`	dibujo de recursos cargados a bitmapdata, Utilizando bitmaps en mosaico, dibujando con transformación.
`MovieClip` + `scrollRect` + `Bitmap` + `Rectangle`	Creando y recortando movieclip personaje utilizando Bitmap como línea de tiempo.
`KeyboardEvent`	detectar entradas de usuario
`Event.EXIT_FRAME`	implementando la función de bucle del juego

---

② Recursos: **(no hay permiso para utilizar estos recursos con fines comerciales)**



③ Código y Comentarios:

> **Nota:** FPS 15 Se utiliza para este tutorial, se recomienda, pero si necesita más, debe modificar parte del código por su cuenta.

Al principio debemos descargar nuestros recursos desde urls externos.

```
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 se llamará una vez que `src_grass_tile_url` esté cargado y listo para su uso. en la aplicación de seguimiento de setGround para obtener recursos y dibujarlos como fondo del juego

```
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));
}
```

El código está muy bien comentado, después de que hayamos terminado con el terreno, es hora de implementar el carácter. El carácter también contiene un recurso que debe cargarse de la misma manera. así que al final de `setGround` nos dirigimos al `setCharacter` que es otra llamada

completa.

```
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);
}
```

Ahora el personaje está listo para controlar. Se muestra bien y listo para controlar. así se adjuntan los eventos del teclado y se escuchan las teclas de flecha como el siguiente código:

```
// 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` almacena la variable booleana de 4 por cada tecla de flecha, el proceso de movimiento debe realizarse dentro de la función de actualización (Loop) del juego, por lo que debemos pasarle las estadísticas del teclado. Permite implementar **la** función **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 velocidad es una constante auxiliar, define la velocidad del carácter. el código anterior presenta un movimiento simple de 8 direcciones con esta prioridad baja: `Up > Down Left > Right` . por lo tanto, si se presionan las flechas hacia arriba y hacia abajo al mismo tiempo, el personaje solo se mueve hacia *arriba* (no se congela).

¡¡¡bien hecho!!! Solo queda un paso, animación, la parte más importante de este tutorial.

¿Qué es realmente la animación? un conjunto de fotogramas clave que contiene al menos un fotograma
permite crear nuestro objeto de fotogramas clave, que contiene el nombre de los fotogramas clave
y también algunos datos sobre el inicio y el final del cuadro de este fotograma clave
Tenga en **cuenta que** en los juegos isométricos, cada fotograma clave contiene 8 direcciones (se puede reducir a 5 con el uso de voltear)

```
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]}
};
```

debemos ignorar los cuadros restantes, este ejemplo solo proporciona animación inactiva y de

ejecución
por ejemplo, el cuadro de inicio de la animación inactiva con dirección a la derecha, es:
<keyframs.idle.right [0]>
ahora permite implementar la función animadora

```
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;
}
```

lea los comentarios de la función anterior, sin embargo, el trabajo principal de esta función es mover *sprite_sheet* `Bitmap` dentro del *carácter* `MovieClip` .

Sabemos que cada actualización debe realizarse dentro de la función Loop, por lo que invocaremos esta función desde Loop con los fotogramas clave relacionados. Esta es la función Loop actualizada:

```
// 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;
}
```

Lea los comentarios, simplemente detectamos un verdadero fotograma clave a través de las estadísticas del teclado. Luego, haga lo mismo para detectar la animación inactiva. para las animaciones inactivas no tenemos ninguna entrada clave que podamos usar para detectar en qué dirección está el carácter, por lo que una variable de ayuda sencilla podría ser útil para almacenar el estado anterior del teclado (last_keyStat).

---

también hay una nueva función `flip` que es otra función auxiliar que se usa para simular animaciones faltantes (left + up_left + down_left) también esta función hace algunas correcciones que se comenta a continuación:

```
// 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
        }
    }
}
```

Nuestro trabajo está terminando aquí. Un agradecimiento especial para los editores que hacen que este tutorial sea más difícil de guardar. También aquí hay una demostración en

vivo de este tutorial más un enlace externo de código completo.

④ Referencias externas:

- **código completo**
- **Demo en vivo**

Lea Fundamentos de desarrollo de juegos en línea: https://riptutorial.com/es/actionscript-3/topic/8237/fundamentos-de-desarrollo-de-juegos

Capítulo 9: Generación de valor aleatorio

Examples

Número aleatorio entre 0 y 1

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

produce un número aleatorio distribuido uniformemente entre 0 (inclusive) y 1 (exclusivo)

Ejemplo de salida:

- 0.22282187035307288
- 0.3948539895936847
- 0.9987191134132445

Número aleatorio entre valores mínimo y máximo

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

Esta función se llama pasando un rango de valores mínimo y máximo.

Ejemplo:

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

Ejemplos de salidas:

- 1.661770915146917
- 2.5521070677787066
- 9.436270965728909

Angulo aleatorio, en grados

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

Ejemplos de salidas:

- 31.554428357630968
- 230.4078639484942
- 312.7964010089636

Valor aleatorio de una matriz

Suponiendo que tenemos una matriz `myArray` :

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

Tenga en cuenta que usamos `int` para convertir el resultado de `Math.random()` en un int porque valores como `2.4539543` no serían un índice de matriz válido.

Punto aleatorio dentro de un círculo

Primero define el radio del círculo y su centro:

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

Luego genera un ángulo aleatorio en *radianes* desde el centro:

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

Luego genere un radio efectivo del punto devuelto, para que esté dentro del `radius` dado. Un simple `Math.random()*radius` no funcionará, porque con esta distribución los puntos `Math.random()*radius` terminarán en el círculo interior de la mitad del radio la mitad del tiempo, pero el cuadrado de ese círculo es una cuarta parte del original. Para crear una distribución adecuada, la función debería ser así:

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

Esta función produce un valor que tiene su función de probabilidad que aumenta linealmente desde 0 en cero hasta el máximo en el `radius` . Ocurre porque una suma de valores aleatorios tiene una función de densidad de probabilidad igual a la convolución de todas las funciones de densidad individual de los valores aleatorios. Estas son algunas matemáticas extendidas para una persona de grado promedio, pero se presenta un tipo de GIF para dibujar una gráfica de la función de convolución de dos funciones de densidad de distribución uniformada explicadas como " señales de caja ". El operador `if` pliega la función resultante sobre su máximo, dejando solo un gráfico en forma de diente de sierra.

Esta función se selecciona porque el cuadrado de una franja de círculo ubicada entre el `radius=r` y el `radius=r+dr` aumenta linealmente al aumentar `r` y la constante muy pequeña `dr` modo que `dr*dr<<r` . Por lo tanto, la cantidad de puntos generados cerca del centro es menor que la cantidad de puntos generados en el borde del círculo por el mismo margen, ya que el radio del área central es menor que el radio del círculo completo. Entonces, en general, los puntos se distribuyen uniformemente en todo el círculo.

Ahora, consigue tu posición aleatoria:

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

Para obtener un punto al azar en el círculo (en el borde del círculo de un radio determinado), use el `radius` lugar de `rad` .

PD: El ejemplo terminó siendo sobrecargado por la explicación de las matemáticas.

Ángulo aleatorio, en radianes

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

Ejemplos de salidas:

- 5.490068569213088
- 3.1984284719180205
- 4.581117863808207

Determinar el éxito de una operación de "porcentaje de probabilidad"

Si necesita tirar para obtener un `true` o `false` en una situación de "x% de probabilidad", use:

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

Utilizado como

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

Crear un color al azar

Para obtener *cualquier* color al azar:

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

Si necesita más control sobre los canales rojo, verde y azul:

```
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;
```

Aquí puede especificar su propio rango para `r` , `g` y `b` (este ejemplo es de 0-255).

Recorrer aleatoriamente el 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);
}
```

Ejemplo de salida:

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

Aleatorizar una matriz

```
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);
```

Ejemplo de salida

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

Este método es conocido como shuffle de Fisher-Yates .

Lea Generación de valor aleatorio en línea: https://riptutorial.com/es/actionscript-3/topic/1441/generacion-de-valor-aleatorio

Capítulo 10: Los tipos

Examples

Tipo de fundición

La conversión de tipos se realiza con el operador `as` :

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

O envolviendo el valor en `Type()` :

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

Si el lanzamiento falla con `as` , el resultado de ese lanzamiento es `null` . Si la conversión falla al ajustarse en `Type()` , se lanza un `TypeError` .

El tipo de función

Las funciones son del tipo `Function` :

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

Pueden ser referenciadas por otras variables con el tipo `Function` :

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

Y se pueden pasar como argumentos para parámetros cuyo tipo es `Function` :

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

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

El tipo de clase

Las referencias a las declaraciones de clase se escriben en `Class` .

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

Puede usar las variables con tipo de `Class` para crear instancias de esa clase:

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

Esto puede ser útil para pasar un argumento de tipo `Class` a una función que podría crear una instancia de la clase proporcionada:

```
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);
```

Anotando tipos

Puede decirle al compilador el tipo de un valor anotándolo con `:Type` :

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

Parámetros de función y tipos de retorno también pueden ser anotados:

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

El intento de asignar un valor con un tipo no coincidente resultará en un `TypeError` :

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

Revisando los tipos

Puede usar el operador `is` para validar si un valor es de un tipo determinado:

```
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.
```

También hay un `instanceof` operador (en desuso), que funciona casi idéntica a `is` , excepto que *devuelve `false` en la comprobación de interfaces implementadas* y tipos int / uint.

El `as` operador también se puede utilizar por igual que `is` operador. Esto es especialmente útil si utiliza algún IDE inteligente como FlashDevelop que le dará una lista de todas las propiedades

posibles del tipo de objeto explícito. Ejemplo:

```
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);
}
```

Para obtener el mismo efecto con `is` que iba a escribir (sligthly menos 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);
}
```

Hemos de tener en cuenta que en la comprobación coditions con `as` operador, el valor dado será el puño se convirtió al tipo especificado y luego el resultado de esta operación se comprobará si no falsa, así que tenga cuidado al usarlo con posibles falsos valores / NaN:

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

La siguiente tabla muestra algunos valores y tipos básicos con el resultado de los operadores de tipo. Las celdas verdes se evaluarán como verdaderas, las rojas y las falsas causarán errores de compilación / tiempo de ejecución.

		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

Arrays mecanografiados

- Recibirá `TypeError` tipo compilación si intenta insertar valores que no sean T en la colección.
- Los IDE proporcionan información de sugerencias de tipo útil para los objetos dentro de una instancia de `Vector.<T>` .

Ejemplos de crear un `Vector.<T>` :

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

---

[1] Los vectores en realidad solo brindan mejoras notables en el rendimiento sobre las matrices cuando se trabaja con tipos primitivos (`String` , `int` , `uint` , `Number` , etc.).

Lea Los tipos en línea: https://riptutorial.com/es/actionscript-3/topic/2803/los-tipos

Capítulo 11: Manipulación de mapa de bits y filtrado

Introducción

En este tema, puede aprender un poco sobre la manipulación de datos de **mapas de** bits y el procesamiento visual, trabajar con píxeles y comenzar a **utilizar** filtros de efectos.

Examples

Efecto umbral (monocromo)

---

necesario:

1. entendiendo bitmap y bitmap data

---

que es el umbral

> Este ajuste toma todos los píxeles de una imagen y ... los empuja a blanco puro o negro puro

Que tenemos que hacer

> Aquí hay una **demostración en vivo** de este ejemplo con algunos cambios adicionales, como usar una interfaz de usuario para cambiar el nivel de umbral en tiempo de ejecución.

umbral en el script de acción 3 de la documentación oficial as3

> Prueba los valores de los píxeles en una imagen contra un umbral específico y establece los píxeles que pasan la prueba a nuevos valores de color. Usando el método de umbral (), puede aislar y reemplazar los rangos de color en una imagen y realizar otras operaciones lógicas en píxeles de imagen.
>
> **La lógica de prueba del método del umbral () es la siguiente:**
>
> 1. Si ((pixelValue & mask) operación (umbral & mask)), entonces configure el pixel en color;
> 2. De lo contrario, si copySource == true, configure el píxel en el valor de píxel correspondiente de sourceBitmap.

Acabo de comentar el siguiente código con exactamente los nombres como descripción citada.

```
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)
}
```

Lea Manipulación de mapa de bits y filtrado en línea: https://riptutorial.com/es/actionscript-3/topic/8055/manipulacion-de-mapa-de-bits-y-filtrado

Capítulo 12: Mostrar lista de ciclo de vida

Observaciones

La animación basada en cuadros en Flash y AIR implementa el siguiente ciclo de vida:

- `Event.ENTER_FRAME` se envía
- Código de constructor de los objetos de visualización de niños se ejecutan
- `Event.FRAME_CONSTRUCTED` se envía
- Se ejecutan acciones de cuadro en el símbolo `MovieClip`
- Acciones de fotograma en niños `MovieClip` símbolos son ejecutados
- `Event.EXIT_FRAME` se envía
- `Event.RENDER` se envía

Examples

Añadido y eliminado del ciclo de vida de la etapa

```
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);
    }

}
}
```

Lea Mostrar lista de ciclo de vida en línea: https://riptutorial.com/es/actionscript-3/topic/1877/mostrar-lista-de-ciclo-de-vida

Capítulo 13: Optimizando el rendimiento

Examples

Gráficos basados en vectores

Los gráficos basados en vectores están representados por una gran cantidad de datos que deben ser computados por la CPU (puntos vectoriales, arcos, colores, etc.). Cualquier otra cosa que no sean formas simples con puntos mínimos y líneas rectas consumirá grandes cantidades de recursos de CPU.

Hay un indicador "Caché como mapa de bits" que se puede activar. Esta bandera almacena el resultado de dibujar el DisplayObject basado en vectores para volver a dibujar mucho más rápido. El inconveniente de esto es que si hay alguna transformación aplicada al objeto, todo debe ser redibujado y vuelto a almacenar en caché. Esto puede ser más lento que no activarlo en absoluto si se aplican transformaciones cuadro por cuadro (rotación, escalado, etc.).

En general, la representación de gráficos utilizando mapas de bits es mucho más eficaz que el uso de gráficos vectoriales. Las bibliotecas como flixel aprovechan esto para representar sprites en un "lienzo" sin reducir la tasa de cuadros.

Texto

La renderización de texto consume mucha CPU. Las fuentes se representan de manera similar a los gráficos vectoriales y contienen muchos puntos vectoriales para cada personaje. Alterar el texto fotograma a fotograma *se* degradará el rendimiento. El indicador "Caché como mapa de bits" es extremadamente útil si se usa correctamente, lo que significa que debe evitar:

- Alterando el texto con frecuencia.
- Transformando el campo de texto (rotando, escalando).

Técnicas simples como envolver actualizaciones de texto en una declaración `if` harán una gran diferencia:

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

El texto se puede representar utilizando el renderizador suavizado incorporado en Flash o usando "fuentes de dispositivo". El uso de "fuentes de dispositivo" hace que el texto se reproduzca mucho más rápido, aunque hace que el texto aparezca irregular (con alias). Además, las fuentes del dispositivo requieren que la fuente sea preinstalada por su usuario final, o el texto puede "desaparecer" en la PC del usuario aunque parezca correcto en la suya.

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

Vector y para cada vs matrices y para

El uso del tipo `Vector.<T>` y `for each` bucle es más eficaz que un arreglo convencional y `for` bucle:

Bueno:

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

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

Malo:

```
var list:Array = [];

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

    sprite.x += 1;
}
```

Eliminación rápida de elementos de matriz

Si no necesita que una matriz esté en ningún orden en particular, un pequeño truco con `pop()` le proporcionará enormes ganancias de rendimiento en comparación con `splice()` .

Cuando `splice()` una matriz, el índice de los elementos subsiguientes en esa matriz debe reducirse en 1. Este proceso puede consumir una gran cantidad de tiempo si la matriz es grande y el objeto que está eliminando está más cerca del comienzo de esa matriz .

Si no le importa el orden de los elementos en la matriz, puede reemplazar el elemento que desea eliminar con un elemento que saque `pop()` del final de la matriz. De esta manera, los índices de todos los demás elementos de la matriz siguen siendo los mismos y el proceso no se degrada en el rendimiento a medida que aumenta la longitud de la matriz.

Ejemplo:

```
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();
```

```
        }
    }
}
```

Vectores en lugar de matrices

Flash Player 10 introdujo el tipo de lista genérica Vector. <*> Que era más rápida que la matriz. Sin embargo, esto no es del todo cierto. Solo los siguientes tipos de vectores son más rápidos que las contrapartes de la matriz, debido a la forma en que se implementan en Flash Player.

- `Vector.<int>` - Vector de enteros de 32 bits
- `Vector.<uint>` - Vector de enteros sin signo de 32 bits
- `Vector.<Double>` - Vector de flotadores de 64 bits

En todos los demás casos, el uso de una matriz será más eficaz que el uso de vectores, para todas las operaciones (creación, manipulación, etc.). Sin embargo, si desea "escribir con fuerza" su código, puede usar Vectores a pesar de la desaceleración. FlashDevelop tiene una sintaxis que permite que los menús desplegables de finalización de código funcionen incluso para Arrays, utilizando `/*ObjectType*/Array` .

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

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

Reutilizando y agrupando gráficos.

Crear y configurar objetos `Sprite` y `TextField` en tiempo de ejecución puede ser costoso si está creando cientos de miles de estos en un solo marco. Por lo tanto, un truco común es "agrupar" estos objetos para reutilizarlos posteriormente. Recuerde que no solo estamos tratando de optimizar el tiempo de creación (`new Sprite()`) sino también la configuración (configuración de las propiedades predeterminadas).

Digamos que estábamos construyendo un componente de lista usando cientos de objetos TextField. Cuando necesite crear un nuevo objeto, verifique si un objeto existente puede ser reutilizado.

```
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);
```

Más adelante, cuando esté destruyendo su componente (o quitándolo de la pantalla), recuerde volver a agregar etiquetas no utilizadas a la agrupación.

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

En la mayoría de los casos, es mejor crear un grupo por uso en lugar de un grupo global. Las desventajas de crear un grupo global es que necesita reinicializar el objeto cada vez que lo recupere del grupo, para anular la configuración realizada por otras funciones. Esto es igualmente costoso y, en primer lugar, niega el aumento de rendimiento del uso de la agrupación.

Lea Optimizando el rendimiento en línea: https://riptutorial.com/es/actionscript-3/topic/2215/optimizando-el-rendimiento

Capítulo 14: Patrón Singleton

Observaciones

El patrón de singleton tiene el objetivo de permitir que solo exista una instancia de una clase en un momento dado.

La prevención de la creación de instancias directas a través del constructor suele evitarse al hacerla privada. Sin embargo, esto no es posible en As3 y, por lo tanto, deben utilizarse otras formas de controlar el número de instancias.

Examples

Ejecutor Singleton a través de instancia privada

En este enfoque, se accede al single a través del método estático:

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

Para imponer solo una instancia del singleton, una variable estática privada retiene la instancia, mientras que cualquier intento adicional de instanciar una instancia se aplica dentro del constructor.

```
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()");
    }

}
}
```

Lea Patrón Singleton en línea: https://riptutorial.com/es/actionscript-3/topic/1437/patron-singleton

Capítulo 15: Programación orientada a objetos

Examples

Constructor "sobrecargado" a través del método estático

La sobrecarga de constructores no está disponible en As3.

Con el fin de proporcionar una forma diferente de recuperar una instancia de una clase, se puede proporcionar un método `public static` para que sirva como un "constructor" alternativo.

Un ejemplo de esto es `flash.geom.Point` , que representa un objeto de punto 2D. Las coordenadas para definir el punto pueden ser

- **cartesiano** en el constructor regular

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

  ejemplo de uso:

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

- **polar** en un método estático

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

  ejemplo de uso:

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

  Debido a que no es un constructor real, no hay una `new` palabra clave.

establecer y obtener funciones

Para garantizar la encapsulación, las variables de los miembros de una clase deben ser `private` y solo deben ser accesibles al `public` través de métodos `public` acceso a `get` / `set` . Es una práctica común prefijar campos privados con _

```
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 veces ni siquiera es necesario crear un campo `private` para un par de `get` / `set` .
Por ejemplo, en un control como un grupo de radio personalizado, necesita saber qué botón de opción está seleccionado, sin embargo, fuera de la clase, solo necesita una forma de `get` / `set` solo el valor seleccionado:

```
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;
        }
    }
}
```

Paquetes

Los paquetes son paquetes de clases. Cada clase debe ser declarada dentro de un paquete usando la declaración del `package` . La declaración del `package` va seguida del nombre de su paquete, o seguida de nada en el caso de agregar clases al paquete de nivel superior. Los subpaquetes se crean utilizando la delimitación de puntos (.) A la declaración del paquete le sigue un bloque que contendrá *una sola definición de* `class` . Ejemplos:

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

package world {
    // A package named world.
}

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

Los paquetes deben correlacionarse con la estructura de archivos de las clases en relación con la raíz de origen. Suponiendo que tiene una carpeta raíz de origen llamada `src` , lo anterior podría representarse correctamente en el sistema de archivos como:

```
src
    TopLevelClass.as

    world
        ClassInWorldPackage.as
        AnotherClassInWorldPackage.as

        monsters
            Zombie.as
```

Método de anulación

Cuando `extend` una clase, puede `override` métodos que la clase heredada define usando la palabra clave de `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!');
    }
}
```

Ejemplo:

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

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

Puede usar la palabra clave `super` para hacer referencia al método original de la clase que se hereda. Por ejemplo, podríamos cambiar el cuerpo de `AnotherExample.test()` a:

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

Resultando en:

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

La anulación de los constructores de clases es un poco diferente. La palabra clave de `override` se omite y el acceso al constructor heredado se realiza simplemente con `super()` :

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

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

También puede anular métodos `get` y `set` .

getter y setter

Getters y setters son métodos que se comportan como propiedades. significa que tienen una estructura funcional, pero cuando se usan, se usan igual que las propiedades:

Estructura de las funciones getter :

que deberían tener `get` palabra clave después de `function` de palabras clave y antes del nombre de la función, sin ningún argumento, especifica un tipo de retorno y debe devolver un valor:

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

Sintaxis :

para obtener el valor de un captador, la sintaxis es la misma que obtener un valor de una propiedad (no se utilizan parens `()`).

```
trace(myValue);
```

Estructura de las funciones de setter :

deberían haber `set` palabra clave después de la palabra clave de la `function` y antes del nombre de la función, con un argumento y sin retorno de valor.

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

Sintaxis :

para establecer el valor de un definidor, la sintaxis es la misma que para establecer un valor en una propiedad (usando el signo igual `=` luego valor).

```
myValue=desiredValue;
```

configurando un captador y definidor para un valor :

> Nota: si creas solo getter o solo setter con un nombre, esa propiedad sería de solo lectura o solo de set.

para hacer que una propiedad sea legible y programable, debe crear un captador y un establecedor con:

1.el mismo nombre.
2.el mismo tipo (tipo de valor de retorno para el captador y tipo de valor de entrada (argumento) para el configurador,

> Nota: los captadores y los definidores no deben tener un nombre igual al de otras propiedades o métodos.

Uso de getters y setters:

El uso de captadores y definidores en lugar de las propiedades normales tiene muchas ventajas:

1. **Hacer propiedades de solo lectura o solo de set:**
por ejemplo, número de hijos en un objeto de visualización. no puede ser setable

2. **Accediendo a propiedades privadas:**
un ejemplo:

```
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. **cuando se requiere algún cambio después de establecer un valor:**
en este ejemplo, se debe notificar el cambio de esta propiedad:

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

y muchos otros usos

Lea Programación orientada a objetos en línea: https://riptutorial.com/es/actionscript-3/topic/2042/programacion-orientada-a-objetos

Capítulo 16: Trabajando con eventos

Observaciones

Los eventos son piezas de datos que un programa puede crear, intercambiar y reaccionar. El flujo de eventos asíncronos se distribuye a través de la lista de visualización por el motor de Flash como una reacción ante eventos externos, como los movimientos del mouse u otro marco que se muestra. Cada otro flujo de eventos y todo el procesamiento de eventos es síncrono, por lo que si un fragmento de código ha generado un evento, todas las reacciones en él se procesan antes de que se ejecute la siguiente línea de código, también si hay varios oyentes de un evento, todos ellos Habría corrido antes de que el próximo evento pudiera ser procesado.

Hay varios eventos importantes asociados con la programación de Flash. `Event.ENTER_FRAME` se genera antes de que Flash dibuje otro marco, señala la lista de visualización completa para prepararse para dibujarse, y se puede usar como un temporizador síncrono. `MouseEvent.CLICK` y sus hermanos se pueden usar para recibir información del mouse del usuario, y `TouchEvent.TOUCH_TAP` es un análogo para las pantallas táctiles. `KeyboardEvent.KEY_DOWN` y `KEY_UP` proporcionan medios para recibir la entrada del usuario desde el teclado, sin embargo, su uso en el departamento móvil es casi imposible debido a que los dispositivos no tienen teclado físico. Finalmente, `Event.ADDED_TO_STAGE` se distribuye una vez que un objeto de visualización recibe acceso a la etapa, y se incluye en la lista de visualización global que recibe la totalidad de los eventos que pueden subir y bajar la lista de visualización.

La mayoría de los eventos en Flash son componentes específicos. Si está diseñando su propio componente que usará eventos de Flash, use una clase descendiente `flash.events.Event` y sus propiedades de `String` estáticas para crear el conjunto de eventos de su componente.

Examples

Eventos personalizados con datos de eventos.

```
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;
        }
    }
}
```

Para enviar un evento personalizado:

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

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

Para escuchar eventos personalizados:

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

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

Manejo de eventos fuera de la lista de visualización

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

public class AbstractDispatcher extends EventDispatcher {

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

}
}
```

Para enviar un evento en una instancia:

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

Para escuchar eventos en una instancia:

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

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

Manejo básico de eventos

Flash despacha **Events** para la mayoría de sus objetos. Uno de los eventos más básicos es **ENTER_FRAME** , que se distribuye (a la velocidad de cuadros del SWF) en cada objeto de la lista de

visualización.

```
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 !");
}
```

Esta función se llamará de forma asíncrona en cada fotograma. Esto significa que la función que asigna como el controlador de eventos `onEnterFrame` se procesa antes que cualquier otro código ActionScript que se adjunte a los marcos afectados.

Añade tus propios eventos

Puede crear sus propios eventos y enviarlos, extendiendo la clase `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);
    }
}
```

Luego puede enviarlo y escucharlo, usando un `EventDispatcher` . Tenga en cuenta que la mayoría de los objetos flash son despachadores de eventos.

```
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!"));
```

Tenga en cuenta que el método de `clone` es obligatorio si desea volver a ver su evento.

Estructura de evento de ratón simple

A través del uso de `event types` de `event types` , puede reducir fácilmente la acumulación de código que ocurre a menudo cuando se definen eventos para muchos objetos en el escenario filtrando eventos en una función en lugar de definir muchas funciones de manejo de eventos.

Imagina que tenemos 10 objetos en el escenario llamados `object1` , `object2` ... `object10`

Podrías hacer lo siguiente:

```
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;
    }
}
```

Los beneficios de este método incluyen:

1. No es necesario especificar la cantidad de objetos para aplicar eventos.
2. No es necesario saber específicamente con qué objeto interactuó y aún así aplicar la funcionalidad.
3. Fácilmente aplicando eventos a granel.

Lea Trabajando con eventos en línea: https://riptutorial.com/es/actionscript-3/topic/1925/trabajando-con-eventos

Capítulo 17: Trabajando con fecha y hora

Examples

Ante meridiem (AM) o Post meridiem (PM) para un reloj de 12 horas

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

Número de días en el mes especificado.

```
/**
 * @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;
}
```

Si el año especificado es año bisiesto

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

Si se observa el horario de verano.

```
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;
}
```

Fecha de inicio de hoy, a medianoche.

```
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);
}
```

Lea Trabajando con fecha y hora en línea: https://riptutorial.com/es/actionscript-3/topic/1926/trabajando-con-fecha-y-hora

Capítulo 18: Trabajando con la geometría

Examples

Obteniendo el ángulo entre dos puntos

Usando las matemáticas de vainilla:

```
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 nuevo vector que representa la diferencia entre los dos primeros:

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

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

> Nota: `atan2()` devuelve radianes, no grados.

Conseguir la distancia entre dos puntos.

Usando las matemáticas de vainilla:

```
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);
```

Usando la funcionalidad incorporada de `Point` :

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

Convertir radianes a grados

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

Convertir grados a radianes

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

El valor de un círculo en grados y radianes.

- Un círculo completo es `360` grados o radianes de `Math.PI * 2` .
- La mitad de esos valores sigue siendo `180` grados o radianes `Math.PI`
- Un cuarto es entonces `90` grados o `Math.PI / 2` .

Para obtener un segmento como un porcentaje de un círculo entero en radianes:

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

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

Moviendo un punto a lo largo de un ángulo

Suponiendo que tiene el ángulo que desea mover y un objeto con los valores `x` e `y` que desea mover:

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

Puedes moverte a lo largo del eje `x` con `Math.cos` :

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

Y el eje `y` con `Math.sin` :

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

Por supuesto, puede multiplicar el resultado de `Math.cos` y `Math.sin` por la distancia que desee recorrer:

```
var distance:int = 20;

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

> Nota: El ángulo de entrada debe estar en radianes.

Determine si un punto está dentro de un área rectangular

Puede probar si un punto está dentro de un rectángulo 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
```

Lea Trabajando con la geometría en línea: https://riptutorial.com/es/actionscript-3/topic/3201/trabajando-con-la-geometria

Capítulo 19: Trabajando con sonido

Sintaxis

- Sound.play (startTime: Number = 0, loops: int = 0, sndTransform: flash.media: SoundTransform = null): SoundChannel // Reproduce un sonido cargado, devuelve un SoundChannel

Examples

Deja de reproducir un sonido

```
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
}
```

Bucle infinito un sonido

```
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"
}
```

Tampoco es necesario esperar a que se cargue el sonido antes de llamar a la función `play()` . Así que esto hará el mismo trabajo:

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

Y si realmente quieres hacer un bucle de sonido en tiempo inifinito por alguna razón (`int.MAX_VALUE` emitirá el sonido de un bucle durante aproximadamente 68 años, sin contar la pausa que causa un mp3 ...) puedes escribir algo como esto:

```
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()` devuelve una nueva instancia del objeto `SoundChannel` cada vez que se llama. Lo asignamos a la variable y escuchamos su evento SOUND_COMPLETE. En el caso de la devolución de llamada, el escucha se elimina del objeto `SoundChannel` actual y se crea uno nuevo para el nuevo objeto `SoundChannel` .

Cargar y reproducir un sonido externo.

```
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();
}
```

Lea Trabajando con sonido en línea: https://riptutorial.com/es/actionscript-3/topic/2043/trabajando-con-sonido

Capítulo 20: Trabajando con temporizadores

Examples

Ejemplo de temporizador de cuenta regresiva

```
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);
    }

}
}
```

Este `CountdownTimer` extiende el `Timer` , y se usa exactamente igual, excepto que el tiempo cuenta atrás.

Ejemplo de uso:

```
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");
}
```

El ejemplo anterior daría como resultado:

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

Intervalos y tiempos de espera

```
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()` se utiliza para realizar tareas repetidas de forma asíncrona como intervalos especificados. Se utiliza el objeto `Timer` interno, el valor devuelto de tipo `uint` es su ID interna, mediante el cual puede acceder y detener la repetición llamando a `clearInterval()` . `setTimeout()` y `clearTimeout()` funcionan de manera similar, pero la llamada a la función suministrada se realiza solo una vez. Puede proporcionar argumentos adicionales a ambas funciones de configuración,

que se pasarán a la función en orden. El número de argumentos y su tipo no se verifican en el momento de la compilación, por lo que si proporciona una combinación extraña de argumentos, o una función que los requiere y no recibe ninguno, se genera un error "Error # 1063: No coincide el conteo de argumentos".

Puede realizar ambas actividades de `setInterval` y `setTimeout` con objetos `Timer` normales, usando 0 o 1 para la propiedad `repeatCount` , 0 para repeticiones indefinidas, 1 para una.

Ejemplo de temporizador aleatorio

```
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();
        }
```

```
    }
}
```

Este `RandomTimer` amplía el `Timer` y se usa exactamente igual, excepto que se distribuye a intervalos aleatorios.

Ejemplo de uso, despacho aleatorio entre 1 y 5 segundos:

```
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();
}
```

El ejemplo anterior daría como resultado:

```
[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
```

Lea Trabajando con temporizadores en línea: https://riptutorial.com/es/actionscript-3/topic/2798/trabajando-con-temporizadores

Capítulo 21: Trabajando con Timeline

Examples

Hacer referencia a la línea de tiempo principal o la clase de documento desde otros MovieClips

En la línea de tiempo de cualquier `DisplayObject` que se adjunta como descendiente del árbol de visualización, puede utilizar la propiedad `root` . Esta propiedad apunta a la línea de tiempo principal en el caso de que no haya una clase de documento personalizada, o la clase de documento si define una.

Debido a que la `root` se escribe `DisplayObject` , el compilador no le permitirá acceder a métodos personalizados o propiedades definidas en la línea de tiempo principal o dentro de su clase de documento como:

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

Para solucionar esto, puede escribir la `root` de su clase de documento en el caso de que tenga una clase de documento:

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

O `MovieClip` en el caso de no clase de documento:

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

La razón por la que el `MovieClip` de `MovieClip` funciona aquí es porque `MovieClip` es `dynamic` . Esto significa que el compilador permite que las propiedades y el método de tiempo de ejecución se declaren en él, evitando errores de tiempo de compilación al intentar acceder a propiedades o métodos que no están definidos explícitamente en `MovieClip` . La desventaja de esto es que pierde toda la seguridad del tipo de tiempo de compilación. Es mucho mejor declarar una clase de documento y convertir a eso.

Lea Trabajando con Timeline en línea: https://riptutorial.com/es/actionscript-3/topic/2459/trabajando-con-timeline

Capítulo 22: Trabajando con valores numéricos.

Examples

Si un número es un valor par

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

Ejemplos:

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

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

Si un número es un valor impar

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

Ejemplos:

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

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

Redondeo a la X más cercana

Para redondear un valor al múltiplo de x más cercano:

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

Ejemplo:

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

Errores de redondeo de números de punto flotante

```
/**
 * @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);
}
```

Ejemplos:

```
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
```

Visualización de números con la precisión requerida.

```
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
```

Lea Trabajando con valores numéricos. en línea: https://riptutorial.com/es/actionscript-3/topic/1899/trabajando-con-valores-numericos-

Capítulo 23: Trabajando con video

Examples

Cargar y reproducir archivo de video externo

---

referencia : `NetConnection` , `NetStream` , `Video`

temas relacionados : trabajando con sonido

Ejemplo básico de reproducción de un archivo de video externo (FLV, MP4, F4V). El código también reproducirá archivos de 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");
```

Observe el código utilizado en `nc.connect.null` ? Esto se debe a que, en este caso, no es necesario crear una conexión de igual a igual (p. Ej., Como se espera en una aplicación de video chat) ya que estamos reproduciendo un archivo almacenado.

Al configurar `nc.connect.null` , es necesario proporcionar un enlace a un archivo que esté en un servidor web o que sea local (misma ubicación / carpeta) al SWF en ejecución.

- Para usar un archivo **web** : `ns.play("http://www.yourwebsite.com/somefile.mp4");`
- Para un uso de archivo **local** : `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');
         }
    }
}
```

Lea Trabajando con video en línea: https://riptutorial.com/es/actionscript-3/topic/2406/trabajando-con-video

Capítulo 24: Trabajar con objetos de visualización

Sintaxis

1. `addChild(child)` : agrega un nuevo elemento al árbol secundario de este objeto como el elemento superior.
2. `addChildAt(child, index)` : agrega un nuevo elemento al árbol hijo de este objeto en una posición específica. El ítem de abajo tiene un índice de 0.
3. `getChildAt(index)` : devuelve un hijo con un índice determinado.
4. `getChildIndex(child)` devuelve el índice de un hijo *directo* de este objeto. De lo contrario se lanza una excepción.
5. `removeChild(child)` : elimina el hijo directo especificado del árbol hijo de este objeto. Lanza una excepción si el padre del niño suministrado no es igual a `this` .
6. `removeChildAt(index)` : elimina un elemento secundario seleccionado por índice en lugar de por referencia. Se lanza la excepción si el árbol hijo no es tan ancho.
7. `removeChildren(beginIndex:int = 0, endIndex:int = 0x7fffffff))` : se agrega en Flash Player 11, elimina un subconjunto de hijos por rango de índice o todos los hijos si se llama sin parámetros.
8. `setChildIndex(child,index)` : cambia el índice del niño al nuevo valor, cambiando a todos los niños para ocupar el lugar liberado.
9. `swapChildren(child1,child2)` : intercambia las posiciones de los dos niños en la lista de visualización, sin afectar las posiciones de otros niños.
10. `swapChildrenAt(index1,index2)` : intercambia a los niños ubicados por sus índices.

Observaciones

La lista de visualización es en realidad un árbol, y se visualiza con el primer algoritmo de profundidad. Cualquier objeto enumerado anteriormente se mostrará anteriormente, y puede ser ocultado por los objetos enumerados más adelante. Todas las técnicas que se pueden usar contra un árbol pueden aplicarse al trabajo con la lista de visualización.

Examples

Introducción a la lista de visualización

En AS3, los activos de visualización no están visibles hasta que se agregan a la Lista de visualización.

El tiempo de ejecución de AIR / Flash tiene una estructura de visualización jerárquica (relación padre-hijo donde los hijos pueden tener sus propios hijos), siendo el `stage` el padre de nivel superior.

Para agregar algo a la lista de visualización, use `addChild` o `addChildAt` . Este es un ejemplo básico de cómo dibujar un círculo y agregarlo a la lista de visualización:

```
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`
```

Para ver el objeto en el ejemplo anterior, `this` (el contexto del código) también debe estar en la lista de visualización, así como cualquier padre que tenga. En AS3, el `stage` es el padre más superior.

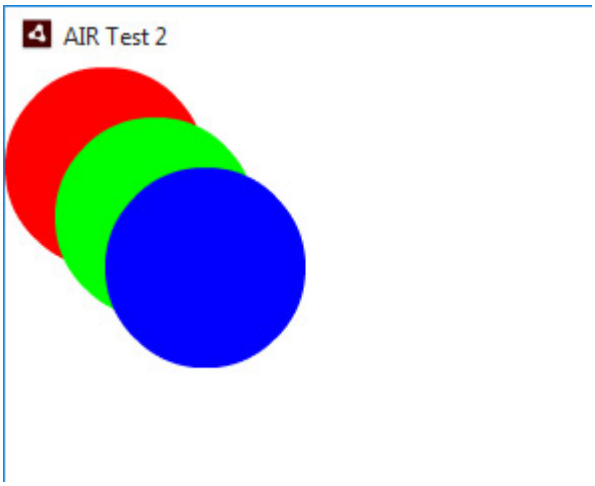
> Los objetos de visualización solo pueden tener un padre. Entonces, si un hijo ya tiene un padre y lo agrega a otro objeto, se eliminará de su padre anterior.

Orden Z / Capas

Digamos que usted replicó el código del ejemplo anterior, por lo que tuvo 3 círculos:

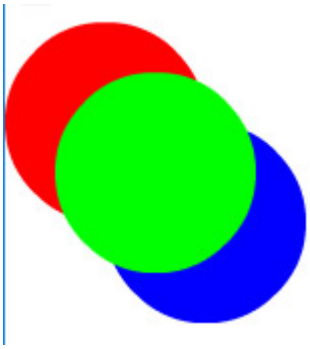
```
var redCircle:Shape = new Shape();
redCircle.graphics.beginFill(0xFF0000); //red
redCircle.graphics.drawCircle(50, 50, 50); //graphics.endFill is not required

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);
```

Dado que el método `addChild` agrega el elemento secundario encima de todo lo demás en el mismo padre, obtendrá este resultado con los elementos en capas en el mismo orden en que usa addChild:



Si querías un hijo en capas diferente en relación a sus hermanos, puedes usar `addChildAt` . Con `addChildAt` , pasa otro parámetro que indica el índice (orden z) en el que debe estar el hijo. Siendo `0` la posición / capa más baja.

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



Ahora el círculo azul está debajo de sus hermanos. Si más adelante desea cambiar el índice de un niño, puede usar el método `setChildIndex` (en el padre del niño).

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

Esto reorganizará el círculo rojo para que esté por encima de todo lo demás. El código anterior produce exactamente el mismo resultado que `this.addChild(redCircle)` .

Eliminar objetos de visualización

Para eliminar objetos, tiene los `removeChild` converse `removeChild` y `removeChildAt` , así como el método `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
```

Eventos

Cuando se agrega un niño a la lista de visualización, algunos eventos se activan en ese niño.

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

Por el contrario, también están los eventos eliminar:

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

Adobe Animate / Flash Professional

Cuando se trata de líneas de tiempo de FlashProfessional / Adobe Animate, agregar algo a la línea de tiempo maneja los matices de la lista de visualización automáticamente. Se agregaron y eliminaron de la lista de visualización automáticamente por la línea de tiempo.

Sin embargo, es bueno tener en cuenta que:

> Si manipula mediante el código la filiación de un objeto de visualización creado por la línea de tiempo (utilizando addChild / setChildIndex), ese hijo ya no será eliminado automáticamente por la línea de tiempo y deberá eliminarse mediante el código.

Capas

Puede haber situaciones en las que decida que un conjunto de objetos de visualización siempre debe estar por encima de otro conjunto de objetos, por ejemplo, flechas sobre cabezas, explosiones sobre algo que acaba de explotar, etc. Para realizar esto de la forma más simple posible, debe designar y cree un conjunto de `Sprite` s, organícelos en orden de abajo hacia arriba, luego agregue todos los objetos del conjunto "anterior" a una capa superior a la utilizada para los objetos del conjunto "inferior".

```
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);
```

Luego, cada vez que agregue un `Monster` a la lista de visualización, agréguelo a `monsterLayer` , y cada vez que agregue una `Bullet` , agregue a `bulletLayer` para lograr el efecto deseado.

Eliminar todos los objetos de la lista de visualización

Si se dirige a Flash Player 11+, el método incorporado removeChildren es la mejor manera de eliminar todos los hijos:

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

Para aplicaciones heredadas, lo mismo se puede lograr con un bucle:

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

Transición de cuadros a cambio manual de contenido

Al principio, un desarrollador de Flash utiliza marcos, ya que están disponibles de forma nativa en Flash Player, para alojar varias pantallas de su aplicación (la mayoría de las veces es un juego). Eventualmente, pueden tropezar con el problema de que algo sale mal exactamente porque han usado marcos, y han pasado por alto las dificultades que surgen de esto, y buscar formas de conservar la estructura de su marco pero también eliminar el obstáculo de usar marcos con sus complicaciones. La solución es usar clases descendientes de `Sprite` , o marcos exportados como `MovieClip` s con un solo marco (para los que se diseñan en Adobe Flash CS), y cambiar manualmente los contenidos con `addChild()` y `removeChild()` .

La clase del administrador debe tener todas sus clases de marco hijo listas, y siempre que se llame una transición, se puede usar una función similar a esta:

```
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;
}
```

Todos los elementos `Event.ADDED_TO_STAGE` pueden enviar y escuchar eventos con `Event.ADDED_TO_STAGE` utilizado como punto de entrada para lo que ocurra después de que `gotoAndStop()` ese marco, y cualquier transición saliente se puede codificar como eventos basados en cadenas, que se escuchan en `Main` clase `Main` . que luego realiza la transición.

```
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]
}
```

Por supuesto, el conjunto de cadenas debe estar predefinido, por ejemplo, la pantalla de introducción puede tener dos botones para iniciar el juego, "Iniciar juego" y "Iniciar silenciado", por ejemplo, y los botones deben enviar diferentes eventos, que luego se manejarán. diferente en la clase de gerente.

Este patrón puede ir tan profundo como sea necesario. Si algún fotograma del proyecto contiene

un MovieClip con múltiples fotogramas, también se puede desacoplar en sprites con este método.

Lea Trabajar con objetos de visualización en línea: https://riptutorial.com/es/actionscript-3/topic/1628/trabajar-con-objetos-de-visualizacion

Capítulo 25: Usando la clase Proxy

Introducción

Primero, tengo que decir. **Hay** una razón por esto proxy, a pesar de su utilidad aparente, no se resalta suficientemente en Internet.

No **puede** usarlo para ver una propiedad aleatoria de una clase / instancia aleatoria. Solo se le permite usar esta técnica subclasificando la clase **Proxy** .

Algunas operaciones no esperan excepciones, por lo que necesita comprender completamente qué está haciendo y por qué lo hace, y su código **debe** ser absolutamente limpio y sin errores.

Examples

Implementación

La otra cosa sobre la clase Proxy, y por qué no es tan popular, es que es bastante difícil comprender un problema que necesita exactamente una clase dinámica con un acceso controlable a sus propiedades y métodos dinámicos como la solución más adecuada. Cada vez que intentaba usar Proxy, terminaba recurriendo a otra cosa, más simple y más controlable.

Sin embargo, no nos desanimemos. Me gusta la idea de abordar los últimos elementos de la matriz mediante los índices [-1], [-2], etc. en **Python** . Puede que no sea una gran hazaña, pero se siente bien usar eso en lugar de un largo y torpe **someArray [someArray.length - 1]** . Veamos que podemos hacer al respecto.

```
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
        }
    }
}
```

Lea Usando la clase Proxy en línea: https://riptutorial.com/es/actionscript-3/topic/10631/usando-la-clase-proxy

Creditos

S. No	Capítulos	Contributors
1	Comenzando con ActionScript 3	BadFeelingAboutThis, Community, Jason Sturges, joshtynjala, Kit Grose, null, Programmer Dancuk, Vesper
2	Cargando archivos externos	BadFeelingAboutThis, Marty
3	Datos binarios	Paweł Audionysos
4	Dibujo de mapas de bits	BadFeelingAboutThis, Marty, Vesper, www0z0k
5	Diseño de aplicación sensible	BadFeelingAboutThis, null, Vesper
6	Entendiendo el "Error 1009: No se puede acceder a una propiedad o método de referencia de objeto nulo"	Vesper, www0z0k
7	Enviando y recibiendo datos de servidores	Marty, null, xims
8	Fundamentos de desarrollo de juegos	payam_sbr
9	Generación de valor aleatorio	alebianco, BadFeelingAboutThis, HITMAN, Jason Sturges, Marty, mnoronha, null, Vesper, xims
10	Los tipos	BadFeelingAboutThis, joshtynjala, Marty, Paweł Audionysos
11	Manipulación de mapa de bits y filtrado	payam_sbr, VC.One
12	Mostrar lista de ciclo de vida	Jason Sturges, mnoronha
13	Optimizando el rendimiento	Community, Marty

14	Patrón Singleton	commovere, Jason Sturges, mnoronha, null
15	Programación orientada a objetos	HITMAN, Marty, null, www0z0k
16	Trabajando con eventos	blue112, Jason Sturges, mnoronha, null, VC.One, Vesper, Zze
17	Trabajando con fecha y hora	Jason Sturges, mnoronha
18	Trabajando con la geometría	Marty, mnoronha, www0z0k
19	Trabajando con sonido	BadFeelingAboutThis, blue112, Paweł Audionysos, VC.One
20	Trabajando con temporizadores	Jason Sturges, mnoronha, Vesper, www0z0k
21	Trabajando con Timeline	Marty
22	Trabajando con valores numéricos.	Jason Sturges, Jonny Henly, Marty, Vesper
23	Trabajando con video	VC.One, www0z0k
24	Trabajar con objetos de visualización	BadFeelingAboutThis, Jason Sturges, Vesper
25	Usando la clase Proxy	Organis