

# MAGGIE

***M**ultiprocess **A**ctionScript **G**eneric **G**ame **I**nterface **E**ngine*

*versión 0.1*

*12/12/2009*

## Índice

Clase Maggie .....	3
Clase Process .....	8
Clase Audio .....	17
Clase TextLabel .....	18
Clase Signal .....	22

# Clase Magie

Esta es la clase principal de la cual debe heredar la clase inicial del juego.

Cabecera:

```
public function Magie(frequency:uint = 0)
```

Donde:

**frequency:** Define la frecuencia del reloj principal del juego. Si es cero, cada proceso definirá su frecuencia individual y no habrá un reloj central que los controle.

Ejemplo:

```
public class Demo extends Magie {  
    public function Demo() {  
        super(25);  
    }  
}
```

## Métodos de manejo de gráficos:

<b>getPixel</b>	Devuelve el color del píxel de una imagen en unas coordenadas específicas.
<pre>public final function getPixel(graph:Class, x:int, y:int, width:uint = 0,                                 height:uint = 0):uint</pre> <p>Donde:</p> <p>graph: Imagen a examinar. x: Coordenada x. y: Coordenada y. width: Si se pasa como argumento, la imagen se expande / contrae según el valor indicado. height: Si se pasa como argumento, la imagen se expande / contrae según el valor indicado.</p>	

<b>setBackgroundImage</b>	Define una imagen de fondo para el juego.
<pre>public final function setBackgroundImage(image:Class):void</pre> <p>Donde:</p> <p>image: La imagen a mostrar.</p>	

<b>deleteBackgroundImage</b>	Elimina la imagen de fondo.
<pre>public final function deleteBackgroundImage():void</pre>	

### Métodos de manejo de procesos:

<b>existProcess</b>	Devuelve true si existe el proceso pasado como argumento.
<pre>public final function existsProcess (process:Process) :Boolean</pre> <p>Donde:</p> <p>process: Proceso del cual se quiere comprobar si existe.</p>	
<b>existsType</b>	Devuelve true si existe al menos un proceso de esa clase.
<pre>public final function existsType (type:Class) :Boolean</pre> <p>Donde:</p> <p>type: Clase por la cual se buscarán los procesos.</p>	

### Métodos de manejo de fuentes:

<b>loadFont</b>	Carga la fuente indicada.
<pre>public final function loadFont (font:Class) :void</pre> <p>Donde:</p> <p>font: Fuente que se quiere cargar.</p>	
<b>existFont</b>	Devuelve true si existe la fuente pasada como parámetro.
<pre>public final function existFont (name:String) :Boolean</pre> <p>Donde:</p> <p>name: Nombre de la fuente.</p>	

### Métodos de manejo del ratón:

<b>mouseEnable</b>	Muestra el ratón en la pantalla.
<pre>public final function mouseEnable () :void</pre>	
<b>mouseDisable</b>	Oculto el ratón en la pantalla.
<pre>public final function mouseDisable () :void</pre>	

<b>getMouseX</b>	Devuelve la coordenada X de la posición del ratón.
<pre>public final function getMouseX():int</pre>	

<b>getMouseY</b>	Devuelve la coordenada Y de la posición del ratón.
<pre>public final function getMouseY():int</pre>	

### **Métodos de manejo de texto:**

<b>addText</b>	Agrega un texto en la pantalla y devuelve su referencia.
<pre>public final function addText(x:int = 0, y:int = 0, text:String = "",                              color:Object = "0x000000",                              font:String = "Verdana",                              size:uint = 10,                              bold:Boolean = false,                              center:Boolean = false):TextLabel</pre> <p>Donde:</p> <ul style="list-style-type: none"> <li>x:           Coordenada X</li> <li>y:           Coordenada Y</li> <li>text:        Texto a mostrar</li> <li>color:       Color del texto</li> <li>font:        Fuente del texto</li> <li>size:        Tamaño del texto</li> <li>bold:        Negrita</li> <li>center:      Texto centrado</li> </ul>	

<b>deleteAllText</b>	Elimina todos los textos que hay en la pantalla.
<pre>public final function deleteAllText():void</pre>	

### **Métodos de manejo de audio:**

<b>playAudio</b>	Reproduce un audio en el juego.
<pre>public final function playAudio(sound:Class, loop:Boolean = false):Audio</pre> <p>Donde:</p> <ul style="list-style-type: none"> <li>sound: Sonido a reproducir.</li> <li>loop: Reproducción infinita.</li> </ul>	

<b>stopAllAudio</b>	Detiene la reproducción de todos los audios en el juego.
<pre>public final function stopAllAudio():void</pre>	

### ***Métodos de manejo de música:***

<b><i>playMusic</i></b>	Reproduce una música en el juego.
<pre>public final function playMusic(sound:Class):void</pre> <p>Donde:</p> <p>sound: Música a reproducir.</p>	
<b><i>stopMusic</i></b>	Detiene la reproducción de la música del juego.
<pre>public final function stopMusic():void</pre>	
<b><i>restartMusic</i></b>	Reinicia la reproducción de la música del juego.
<pre>public final function restartMusic():void</pre>	
<b><i>isPlayingMusic</i></b>	Devuelve true si se está reproduciendo la música en el juego.
<pre>public final function isPlayingMusic():Boolean</pre>	

### ***Métodos de manejo de alarmas:***

<b><i>alarm</i></b>	Fija una alarma en el juego.
<pre>public final function alarm(func:Function, time:uint):void</pre> <p>Donde:</p> <p>func: Música a reproducir.</p> <p>time: Tiempo de la alarma.</p>	

### Métodos de manejo de funciones de red:

<b>dataRequest</b>	Realiza una petición al servidor.
<pre>public final function dataRequest(url:String, complete:Function,                                   fail: Function      = null,                                   parameters:Object    = null,                                   method:String        = "post"):void</pre> <p>Donde:</p> <p>url: Dirección destino de la petición. complete: Función que se llama en caso de que la petición se realice correctamente. fail: Función que se llama en caso de que la petición falle. parameters: Parámetros de la petición. method: Método con el cual se realizará la petición. [GET - POST]</p>	

### Métodos de propiedades del juego:

<b>getScreenWidth</b>	Devuelve el ancho de la pantalla.
<pre>public final function getScreenWidth():uint</pre>	

<b>getScreenHeight</b>	Devuelve el alto de la pantalla.
<pre>public final function getScreenHeight():uint</pre>	

<b>setFps</b>	Fija los frames por segundo del juego.
<pre>public final function setFps(fps:uint):void</pre> <p>Donde:</p> <p>fps: Valor de los frames por segundo.</p>	

<b>random</b>	Devuelve un número aleatorio comprendido entre los parámetros <i>min</i> y <i>max</i> .
<pre>public final function random(min:int, max:int):int</pre> <p>Donde:</p> <p>min: Valor mínimo del resultado. max: Valor máximo del resultado.</p>	

<b>showMemory</b>	Muestra por consola la memoria utilizada por el juego en ese momento.
<pre>public final function showMemory():void</pre>	

## Clase Process

Esta clase permite crear procesos a lo largo del juego, los cuales se ejecutan en paralelo y pueden interactuar entre ellos.

Cabecera:

```
public function Process(frequency:uint = 25, x:int = 0, y:int = 0,
                        graph:Class = null, angle:uint = 0)
```

Donde:

frequency: Define la frecuencia proceso. Si se ha definido un reloj central, este parámetro no se utiliza.  
x: Coordenada inicial X del proceso.  
y: Coordenada inicial Y del proceso.  
graph: Gráfico inicial del proceso.  
angle: Ángulo inicial del proceso.

Ejemplo:

```
public class Proceso extends Process {
    public function Proceso (x:int, y:int, image:Class) {
        super(25, x, y, image);
    }
}
```

Si se desea que el proceso implemente una lógica iterativa, se deberá sobrescribir el siguiente método:

<b>run</b>	Método principal que implementa la lógica del proceso.
<pre>public final function run():void</pre>	

### ***Métodos de comunicación ente procesos:***

<b>onFreeze</b>	Fija un listener para cuando el proceso se congela.
<pre>public final function onFreeze (func:Function):void</pre>	
Donde:	
func: Función a la que se llama cuando el proceso se congela.	



<b><i>onSleep</i></b>	Fija un listener para cuando el proceso se duerme.
<pre>public final function onSleep (func:Function) :void</pre> <p>Donde:</p> <p>func: Función a la que se llama cuando el proceso se duerme.</p>	
<b><i>onWakeUp</i></b>	Fija un listener para cuando el proceso se despierta.
<pre>public final function onWakeUp (func:Function) :void</pre> <p>Donde:</p> <p>func: Función a la que se llama cuando el proceso se despierta.</p>	
<b><i>onFinish</i></b>	Fija un listener para cuando el proceso finaliza.
<pre>public final function onFinish (func:Function) :void</pre> <p>Donde:</p> <p>func: Función a la que se llama cuando el proceso finaliza.</p>	
<b><i>letMeAlone</i></b>	Mata a todos los procesos excepto al que invoca a esta función.
<pre>public final function letMeAlone () :void</pre>	
<b><i>isSleeping</i></b>	Devuelve true si el proceso esta dormido.
<pre>public final function isSleeping () :Boolean</pre>	
<b><i>isFrozen</i></b>	Devuelve true si el proceso esta congelado.
<pre>public final function isFrozen () :Boolean</pre>	
<b><i>isAwake</i></b>	Devuelve true si el proceso esta despierto.
<pre>public final function isAwake () :Boolean</pre>	
<b><i>sleep</i></b>	Duerme al proceso.
<pre>public final function sleep () :void</pre>	

<b>freeze</b>	Congela al proceso.
<pre>public final function freeze():void</pre>	

<b>wakeUp</b>	Despierta al proceso.
<pre>public final function wakeUp():void</pre>	

<b>finish</b>	Finaliza el proceso.
<pre>public final function finish(sendSignal:Boolean = true):void</pre> <p>Donde:</p> <p>sendSignal: Envía una señal indicando que el proceso ha finalizado.</p>	

### **Métodos de manejo de ángulos:**

<b>getAngle</b>	Devuelve el ángulo actual del proceso.
<pre>public final function getAngle():uint</pre>	

<b>addAngle</b>	Suma una cantidad al ángulo actual del proceso.
<pre>public final function addAngle(value:int):void</pre> <p>Donde:</p> <p>value: Suma al ángulo actual el valor pasado como parámetro.</p>	

<b>setAngle</b>	Fija el ángulo del proceso.
<pre>public final function setAngle(newAngle:int):void</pre> <p>Donde:</p> <p>newAngle: Nuevo valor que tomará el ángulo del proceso.</p>	

<b>advance</b>	Avanza el proceso en el ángulo correspondiente.
<pre>public final function advance(value:int):void</pre> <p>Donde:</p> <p>value: Valor según cual avanza el proceso en su ángulo correspondiente.</p>	

<b>getAngleWith</b>	Devuelve el ángulo entre el proceso actual y el indicado.
<pre>public final function getAngleWith (process:Process) :uint</pre> <p>Donde:</p> <p>process: Proceso con el cual se calcula el ángulo.</p>	

### ***Métodos de manejo de imágenes:***

<b>setImage</b>	Fija la imagen del proceso.
<pre>public final function setImage (graph:Class = null) :void</pre> <p>Donde:</p> <p>graph: Imagen que va a tomar el proceso. Si no se indica, el proceso se queda sin imagen.</p>	

<b>setAlpha</b>	Fija la transparencia de la imagen del proceso.
<pre>public final function setAlpha (value:Number) :void</pre> <p>Donde:</p> <p>value: Valor de de la transparencia de la imagen. Valor comprendido entre 0 y 1.</p>	

<b>getWidth</b>	Devuelve el ancho de la imagen.
<pre>public final function getWidth() :uint</pre>	

<b>getHeight</b>	Devuelve el alto de la imagen.
<pre>public final function getHeight() :uint</pre>	

<b>setScale</b>	Fija la escala de la imagen.
<pre>public final function setScale (value:Number) :void</pre> <p>Donde:</p> <p>value: Valor de la escala que tomará la imagen del proceso.</p>	

<b>setScaleX</b>	Fija la escala en X.
<pre>public final function setScaleX(value:Number):void</pre> <p>Donde:</p> <p>value: Valor de la escala en X que tomará la imagen del proceso.</p>	

<b>setScaleY</b>	Fija la escala en Y.
<pre>public final function setScaleY(value:Number):void</pre> <p>Donde:</p> <p>value: Valor de la escala en Y que tomará la imagen del proceso.</p>	

### ***Métodos de detección de colisiones:***

<b>collision</b>	Devuelve la referencia de un proceso si éste colisiona con él.
<pre>public final function collision(type:Class = null):Process</pre> <p>Donde:</p> <p>type: Solo verifica colisiones con procesos de esa clase.</p>	

<b>collisionWith</b>	Devuelve true una si hay colisión con el proceso indicado.
<pre>public final function collisionWith(process: Process):Boolean</pre> <p>Donde:</p> <p>process: Proceso con el cual se comprueba si hay colisión.</p>	

### ***Métodos de manejo de teclado:***

<b>key</b>	Devuelve true si se ha presionado la tecla indicada.
<pre>public final function key(code:uint):Boolean</pre> <p>Donde:</p> <p>code: Código de la tecla que se comprueba.</p>	

<b>keyUp</b>	Devuelve true si se ha presionado la tecla UP.
<pre>public final function keyUp () :Boolean</pre>	
<b>keyDown</b>	Devuelve true si se ha presionado la tecla DOWN.
<pre>public final function keyDown () :Boolean</pre>	
<b>keyLeft</b>	Devuelve true si se ha presionado la tecla LEFT.
<pre>public final function keyLeft () :Boolean</pre>	
<b>keyRight</b>	Devuelve true si se ha presionado la tecla RIGHT.
<pre>public final function keyRight () :Boolean</pre>	
<b>keySpace</b>	Devuelve true si se ha presionado la tecla SPACE.
<pre>public final function keySpace () :Boolean</pre>	
<b>keyEscape</b>	Devuelve true si se ha presionado la tecla ESCAPE.
<pre>public final function keyEscape () :Boolean</pre>	
<b>keyEnter</b>	Devuelve true si se ha presionado la tecla ENTER.
<pre>public final function keyEnter () :Boolean</pre>	

### ***Métodos de manejo de movimiento y distancias:***

<b>moveDown</b>	Mueve la imagen del proceso hacia abajo.
<pre>public final function moveDown (value:int) :void</pre>	
Donde:	
value: Distancia a mover.	

<b><i>moveUp</i></b>	Mueve la imagen del proceso hacia arriba.
<pre>public final function moveUp (value:int) :void</pre> <p>Donde:</p> <p>value: Distancia a mover.</p>	
<b><i>moveLeft</i></b>	Mueve la imagen del proceso hacia la izquierda.
<pre>public final function moveLeft (value:int) :void</pre> <p>Donde:</p> <p>value: Distancia a mover.</p>	
<b><i>moveRight</i></b>	Mueve la imagen del proceso hacia la derecha.
<pre>public final function moveRight (value:int) :void</pre> <p>Donde:</p> <p>value: Distancia a mover.</p>	
<b><i>getX</i></b>	Devuelve la coordenada X del proceso.
<pre>public final function getX() :int</pre>	
<b><i>getY</i></b>	Devuelve la coordenada Y del proceso.
<pre>public final function getY() :int</pre>	
<b><i>setX</i></b>	Fija la coordenada X del proceso.
<pre>public final function setX(value:int) :void</pre> <p>Donde:</p> <p>value: Nuevo valor de la coordenada.</p>	

<b>setY</b>	Fija la coordenada Y del proceso.
<pre>public final function setY(value:int):void</pre> <p>Donde:</p> <p>value: Nuevo valor de la coordenada.</p>	

<b>getDistance</b>	Devuelve la distancia entre el proceso actual y el indicado.
<pre>public final function getDistance(process:Process):uint</pre> <p>Donde:</p> <p>process: Proceso con el cual se calcula la distancia.</p>	

<b>getDistanceX</b>	Devuelve la distancia respecto del eje X entre el proceso actual y el indicado.
<pre>public final function getDistanceX(process:Process):uint</pre> <p>Donde:</p> <p>process: Proceso con el cual se calcula la distancia.</p>	

<b>getDistanceY</b>	Devuelve la distancia respecto del eje Y entre el proceso actual y el indicado.
<pre>public final function getDistanceY(process:Process):uint</pre> <p>Donde:</p> <p>process: Proceso con el cual se calcula la distancia.</p>	

### **Métodos de manejo de efectos:**

<b>fadeIn</b>	Ejecuta el efecto <i>fade in</i> a la imagen del proceso.
<pre>public final function fadeIn(from:Number, until:Number, step:Number,                              frequency:uint, onFinish:Function = null):void</pre> <p>Donde:</p> <p>from: Valor límite inicial.</p> <p>until: Valor límite final.</p> <p>step: Valor con la cual se incrementa el fade.</p> <p>frequency: Frecuencia con la cual se ejecuta el efecto.</p> <p>onFinish: Función que se llama al terminar la ejecución del efecto.</p>	

<b><i>fadeOut</i></b>	Ejecuta el efecto <i>fade out</i> a la imagen del proceso.
<pre>public final function fadeOut(from:Number, until:Number, step:Number,                              frequency:uint, onFinish:Function = null):void</pre> <p>Donde:</p> <p>from: Valor límite inicial.  until: Valor límite final.  step: Valor con la cual se decrementa el fade.  frequency: Frecuencia con la cual se ejecuta el efecto.  onFinish: Función que se llama al terminar la ejecución del efecto.</p>	

***Los siguientes métodos son iguales que los de la clase Maggie:***

<b><i>getScreenWidth</i></b>	<b><i>dataRequest</i></b>	<b><i>playAudio</i></b>	<b><i>playMusic</i></b>	<b><i>addText</i></b>
<b><i>getScreenHeight</i></b>	<b><i>getMouseX</i></b>	<b><i>stopAllAudio</i></b>	<b><i>stopMusic</i></b>	<b><i>deleteAllText</i></b>
<b><i>getPixel</i></b>	<b><i>getMouseY</i></b>	<b><i>existsProcess</i></b>	<b><i>restartMusic</i></b>	<b><i>loadFont</i></b>
<b><i>setBackgroundImage</i></b>	<b><i>mouseEnable</i></b>	<b><i>existsType</i></b>	<b><i>isPlayingMusic</i></b>	<b><i>existFont</i></b>
<b><i>deleteBackgroundImage</i></b>	<b><i>mouseDisable</i></b>	<b><i>setFps</i></b>	<b><i>alarm</i></b>	<b><i>random</i></b>



## Clase Audio

Esta clase permite el manejo de sonidos en el juego. Sin embargo no debe instanciarse de forma directa sino a través del método **playAudio**, el cual devuelve la referencia al audio correspondiente para, a través de ella, poder manipular el sonido mediante los siguientes métodos:

<b>play</b>	Reproduce el audio.
<pre>public final function play():void</pre>	
<b>stop</b>	Detiene la reproducción del audio.
<pre>public final function stop():void</pre>	
<b>finish</b>	Elimina el audio.
<pre>public final function finish():void</pre>	
<b>isPlaying</b>	Devuelve true si el audio se está reproduciendo.
<pre>public final function isPlaying():Boolean</pre>	
<b>onFinish</b>	Fija un listener para cuando el audio finaliza.
<pre>public final function onFinish(func:Function):void</pre>	
Donde:	
func: Función a la que se llama cuando el audio finaliza.	

## Clase TextLabel

Esta clase permite el manejo de texto en el juego. Al igual que la clase **Audio**, no debe instanciarse de forma directa sino a través del método **addText**, el cual devuelve la referencia al texto correspondiente para, a través de ella, poder manipularlo mediante los siguientes métodos:

### *Métodos de manejo de posición y movimiento:*

<b>setX</b>	Fija la coordenada X del texto.
<pre>public final function setX(value:int):void</pre> <p>Donde:</p> <p>value: Nuevo valor de la coordenada.</p>	
<b>setY</b>	Fija la coordenada Y del texto.
<pre>public final function setY(value:int):void</pre> <p>Donde:</p> <p>value: Nuevo valor de la coordenada.</p>	
<b>getX</b>	Devuelve la coordenada X del texto.
<pre>public final function getX():int</pre>	
<b>getY</b>	Devuelve la coordenada Y del texto.
<pre>public final function getY():int</pre>	
<b>moveUp</b>	Mueve el texto hacia arriba.
<pre>public final function moveUp(value:int):void</pre> <p>Donde:</p> <p>value: Distancia a mover.</p>	

<b>moveDown</b>	Mueve el texto hacia abajo.
<pre>public final function moveDown (value:int) :void</pre> <p>Donde:</p> <p>value: Distancia a mover.</p>	

<b>moveLeft</b>	Mueve el texto hacia la izquierda.
<pre>public final function moveLeft (value:int) :void</pre> <p>Donde:</p> <p>value: Distancia a mover.</p>	

<b>moveRight</b>	Mueve el texto hacia la derecha.
<pre>public final function moveRight (value:int) :void</pre> <p>Donde:</p> <p>value: Distancia a mover.</p>	

### ***Métodos de manejo de propiedades del texto:***

<b>setText</b>	Fija el valor del texto.
<pre>public final function setText (value:String) :void</pre> <p>Donde:</p> <p>value: Nuevo valor del texto.</p>	

<b>getText</b>	Devuelve el valor actual del texto.
<pre>public final function getText () :String</pre>	

<b>getWidth</b>	Devuelve el ancho del texto.
<pre>public final function getWidth () :uint</pre>	

<b>getHeight</b>	Devuelve el alto del texto.
<pre>public final function getHeight () :uint</pre>	

<b>setBackgroundColor</b>	Fija el color de fondo del texto.
<pre>public final function setBackgroundColor (color:uint) :void</pre> <p>Donde:</p> <p>color: Nuevo color del fondo del texto.</p>	
<b>removeBackgroundColor</b>	Elimina el color de fondo del texto.
<pre>public final function removeBackgroundColor () :void</pre>	
<b>setBorderColor</b>	Fija el color del borde del texto.
<pre>public final function setBorderColor (color:uint) :void</pre> <p>Donde:</p> <p>color: Nuevo color del borde del texto.</p>	
<b>removeBorderColor</b>	Elimina el color del borde del texto.
<pre>public final function removeBorderColor () :void</pre>	
<b>setColor</b>	Fija el color del texto.
<pre>public final function setColor (value:String) :void</pre> <p>Donde:</p> <p>value: Nuevo color del texto.</p>	
<b>getColor</b>	Devuelve el color del texto.
<pre>public final function getColor () :String</pre>	
<b>setFont</b>	Fija la fuente del texto.
<pre>public final function setFont (value:String) :void</pre> <p>Donde:</p> <p>value: Nueva fuente del texto.</p>	

<b>getFont</b>	Devuelve la fuente del texto.
<pre>public final function getFont() :String</pre>	
<b>setSize</b>	Fija el tamaño del texto.
<pre>public final function setSize(value:uint):void</pre> <p>Donde:</p> <p>value: Nuevo tamaño del texto.</p>	
<b>getSize</b>	Devuelve el tamaño del texto.
<pre>public final function getSize():uint</pre>	
<b>setBold</b>	Fija en negrita el texto.
<pre>public final function setBold(bold:Boolean = true):void</pre> <p>Donde:</p> <p>bold: Fija en negrita de texto.</p>	
<b>isBold</b>	Devuelve true si el texto está en negrita.
<pre>public final function isBold():Boolean</pre>	
<b>setCenter</b>	Fija la alineación del texto.
<pre>public final function setCenter(center:Boolean):void</pre> <p>Donde:</p> <p>center: Fija la alineación del texto.</p>	
<b>isCenter</b>	Devuelve true si el texto está centrado.
<pre>public final function isCenter():Boolean</pre>	
<b>remove</b>	Elimina el texto.
<pre>public final function remove():void</pre>	

## Clase Signal

Esta clase permite detectar que tipo de señal recibe una función y que proceso fue el que la provocó. Todos los procesos que esperen una señal de otro proceso, deberán implementar su función destino de la siguiente manera:

```
private function destino(signal:Signal) {  
    ...  
}
```

Los siguientes métodos están disponibles para esta clase:

<b><i>getProcess</i></b>	Devuelve la referencia del proceso que provocó la señal.
<pre>public final function getProcess(type:Class = null):*</pre> <p>Donde:</p> <p>type: Realiza un cast según la clase pasada como parámetro.</p>	
<b><i>isFinish</i></b>	Devuelve true si la señal es de finalización.
<pre>public final function isFinish():Boolean</pre>	
<b><i>isSleep</i></b>	Devuelve true si la señal es de dormir.
<pre>public final function isSleep():Boolean</pre>	
<b><i>isWakeUp</i></b>	Devuelve true si la señal es de despertar.
<pre>public final function isWakeUp():Boolean</pre>	
<b><i>isFreeze</i></b>	Devuelve true si la señal es de congelar.
<pre>public final function isFreeze():Boolean</pre>	