# Estructura orxANIMSET

## Sumario

### Animation Set

```
[AnimationSetTemplate]
AnimationList = AnimationTemplate1#AnimationTemplate2#...
LinkList      = LinkTemplate1#LinkTemplate2#...
Frequency     = <float>
```

### Animation

```
[AnimationTemplate]
DefaultKeyDuration = <float>
KeyData<N>         = GraphicTemplate
KeyDuration<N>     = <float>
KeyEventName<N>    = <string>
KeyEventValue<N>   = <float>
```

### Animation link

```
[LinkTemplate]
Source      = SourceAnimationTemplate
Destination = DestinationAnimationTemplate
Priority    = <int>
Property    = immediate | cleartarget
```

## Detalles

### Animation

Una lista de propiedades disponibles para una animación:

- `AnimationList`: Lista de todas las animaciones (nombres de configuración) que son partes del set(conjunto). Hasta 256 animaciones pueden ser definidas. Esta propiedad *necesita* ser definida.
- `LinkList`: Lista de todos lo enlaces [1)] que son parte del set. Hasta 256 enlaces puede ser definidos. Esta propiedad *necesita* ser definida.
- `Frecuency`: Frecuencia relativa que será aplicada a todas las animaciones de ese set cuando son reproducidas. Su valor por defecto es 1.0, que dice que la animacion será reproducida con la sincronización definida en la configuración.

### Animation set

Tenemos aquí una lista de propiedades disponibles para un set de animaciones:

- `DefaultDuration`: Duración por defecto para todas las llaves definidas, en segundos. Esta duración puede ser reemplazada localmente por cualquier llave definida. Por defecto su valor es 0.0 y necesita ser definida a menos que cada llave tenga un valor local.
- `KeyData<N>`: Todas las llaves necesitan ser definidas secuencialmente, empezando con `KeyData1`. Ninguna brecha es permitida en los números. Para cada llave, el valor es el orxGRAPHIC que será rendereado cuando el cursor de animación apunta a esa llave, cuando es reproducida. Al menos una llave (`KeyData1`) necesita ser definida.
- `KeyDuration<N>`: Para cualquier llave definida (con `KeyData<N>`), una duración, en segundos, puede ser especificada. Si ninguna es especificada, la `LlaveDuraciónPorDefecto(DefaultKeyDuration)` será usada.
- `KeyEventName<N>`: Los eventos personalizados se pueden asociar a llaves en animaciones. Permiten sincronizarse con una parte específica de las animaciones.
- `KeyEventValue<N>`: Valor opcional para un evento. Por defecto este valor es 0.0.

Ejemplo:

```
[MyAnimation]
DefaultKeyDuration = 0.1
KeyData1           = MyGraphic1
KeyData2           = MyGraphic2
KeyDuration2       = 0.2; <= esta llave se mostrará el doble de veces que
las otras llaves
KeyData3           = MyGraphic1; <= Reusando el mismo contenido que para la
primera llave
...
KeyEventName1      = SetSoundVolume
KeyEventValue1     = 1.0
KeyEventName2      = SetSoundVolume
KeyEventValue2     = 0.0
KeyEventName3      = Explode
...
```

### Animation link

Vemos aquí una lista de propiedades disponibles para un enlace de animación:

- `Source/Destination`: Define la fuente y destino de las animaciones para el enlace, ej. el comienzo y final de una posible transición. Esas propiedades *necesitan* ser definidas.
- `Priority`: Define la propiedad de un enlace, el puede ser `inmediato(immediate)`, `cleartarget(?)`. Si un enlace es `immediate` quiere decir que la transición ocurrirá inmediatamente en lugar de esperar a que el final de la animación de origen tome lugar. Si un enlace es `cleartarget` significa que va a restablecer la animación del objeto de destino

actual, cuando el enlace se utiliza. Por defecto ninguna de estas propiedades son definidas, lo cual explica que la transición ocurrirá *después* de la fuente de la animación fue completamente reproducida y no restablecerá el objetivo actual del objeto.

Ejemplo:

```
[LinkSitAnimLoop]
Source      = SitAnim
Destination = SitAnim
Priority    = 15; <= esta es nuestra transición escogida si ninguna
animación es solicitada, este es un ciclio para siempre =)

[LinkSitToStandUp]
Source      = SitAnim
Destination = StandUpAnim
Property    = immediate; <=  esto significa que no esperaremos por el final
del ciclo de SitAnim antes de lanzar  StandUpAnim

[LinkStandUpToRun]
Source      = StandUpAnim
Destination = RunAnim
```

Ahora en el código, si estamos reproduciendo la animación `SitAnim` y emitimos

```
orxObject_SetTargetAnim(MyHero, "RunAnim");
```

MyHero primero reproducirá inmediatamente la animación `StandUpAnim` y entonces irá a por la animación `RunAnim` sin tener que esperar por ningún mensaje de animación ni nada. Puedes escuchar por el evento `orxEVENT_TYPE_ANIM` si quieres ser notificado cuando una transición ocurra.

Entre dos animaciones, el camino calculado tomará el enlace de más alta prioridad en cada paso y si dos enlaces con la misma prioridad son encontrados, el tomará entonces el que tenga el camino más corto (ej. la menor cantidad de enlaces necesarios para llegar a su destino).

**PD: Si no te sientes a gusto con este sistema de gráfo, todavía se puede definir todas las animaciones por separado y luego hacer todas las transiciones de forma manual cada vez que ha sido la animación actual reproducida completamente.**

[1)]
un link es una transicion disponible de una animación a otra