orxANIMSET structure

Summary

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
```

Details

Animation set

Here's a list of the available properties for an animation:

- `AnimationList`: List of all the animations (config names) that are part of this set. Up to 256 animations can be defined. This property *needs* to be defined.
- `LinkList`: List of all links [1] that are part of this set. Up to 256 links can be defined. This property *needs* to be defined.
- `Frequency`: Relative frequency that will be applied to all animations of this set when they're played. Its default value is 1.0, which means the animation will be played with the timing defined in the config.

Animation

Here's a list of the available properties for an animation set:

- `DefaultKeyDuration`: Default duration for all defined keys, in seconds. This duration can be overridden locally for any defined key. By default its value is 0.0 and needs to be defines unless every key has a local value.
- `KeyData<N>`: All the keys needs to be defined sequencially, starting with the `KeyData1`. No gap is allowed in the numbers. For every key, the value is the orxGRAPHIC that will be rendered when the animation cursor points on this key, when played. At least one key (`KeyData1`) needs to be defined.
- `KeyDuration<N>`: For any key defined (with `KeyData<N>`), a duration, in seconds, can be specified. If none is defined, the `DefaultKeyDuration` will be used instead.
- `KeyEventName<N>`: Custom events can be associated to keys in animations. They allow to synchronize with specific part of animations.
- `KeyEventValue<N>`: Optional value for an event. By default its value is 0.0.

Example:

```
[MyAnimation]
DefaultKeyDuration = 0.1
KeyData1           = MyGraphic1
KeyData2           = MyGraphic2
KeyDuration2       = 0.2; <= this key will be displayed twice the time of
the other keys
KeyData3           = MyGraphic1; <= Reusing the same content as for the
first key
...
KeyEventName1      = SetSoundVolume
KeyEventValue1     = 1.0
KeyEventName2      = SetSoundVolume
KeyEventValue2     = 0.0
KeyEventName3      = Explode
...
```

Animation link

Here's a list of the available properties for an animation link:

- `Source/Destination`: Defines the source and destination animations for the link, ie. the start and end of a possible transition. These properties *need* to be defined.
- `Priority`: Defines the priority of this link, rangin from 0 (lowest) to 15 (highest). Priorities are mostly used to define default transitions when no target animation is specified for the current object. By default its value is 8 which means a medium priority.
- `Property`: Defines the property of the link, it can either be `immediate`, `cleartarget`. If a link is `immediate` it means the transition will occur immediately instead of waiting for the end of the source animation before taking place. If a link is `cleartarget` it means that it will reset

object's current target animation when the link is used. By default none of these properties are defined, which means the transition will occur *after* the source animation has been completely played and won't reset object's current target.

Example:

```
[LinkSitAnimLoop]
Source      = SitAnim
Destination = SitAnim
Priority    = 15; <= this is our transition of choice if no target anim is
requested, we just cycle here for ever =)

[LinkSitToStandUp]
Source      = SitAnim
Destination = StandUpAnim
Property    = immediate; <= that means we won't wait for the end of the
SitAnim cycle before going to StandUpAnim

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

Now in the code, if we're currently playing the `SitAnim` and we issue

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

MyHero will first play immediately the `StandUpAnim` and then go to the `RunAnim` without you having to wait for any animation message or anything. You can listen to the `orxEVENT_TYPE_ANIM` events if you still want to be notified when transitions occur.

Between two animations, the calculated path will take the highest priority links at each step and if two links with the same priority are found, it will then take the one that leads to the shortest path (ie. the least amount of links needed to reach destination).

NB: If you don't feel at ease with this graph system, you can still define all your animations separately and then do all the transitions manually every time the current animation has been completely played (listening to the `orxANIM_EVENT_STOP` event, for example).

[1)]
a link is an available transition from an animation to another one