Using Keyframe Sequence to effect Custom Interpolation

See a few examples of what keyframe sequence can do.

Overview

The primary use for an SKKeyframeSequence object is to animate properties on particles emitted by an SKEmitterNode object. When a keyframe object is assigned to an appropriate property on the emitter node, particles determine their values by sampling the keyframe sequence. The sequence replaces the normal simulation performed by the emitter node. Only certain properties can be animated using a keyframe sequence. The following table lists the particle emitter properties and the corresponding class for the data that must be stored in the sequence. All value objects stored in a sequence must have the same class.

TABLE 1 EMITTER PROPERTIES FOR KEYFRAME SEQUENCES

SKEmitterNode property	Value class
particleColorSequence	SKColor
particleColorBlendFactorSequence	NSNumber containing a floating point value.
particleScaleSequence	NSNumber containing a floating point value.
particleAlphaSequence	NSNumber containing a floating point value.

The time values stored in the sequence are specified in a normalized range of 0.0 to 1.0, where 0 is the time when the particle was created and 1.0 is the time when the particle dies. You can provide keyframe values for the entire lifetime of the particle or for a subset of the particle's lifetime. If you choose to cover only a subset of the particle's lifetime, the repeatMode property determines how samples are determined for time values that lie outside the specified range.

The `interpolationMode` property determines how samples between the keyframe values are calculated.

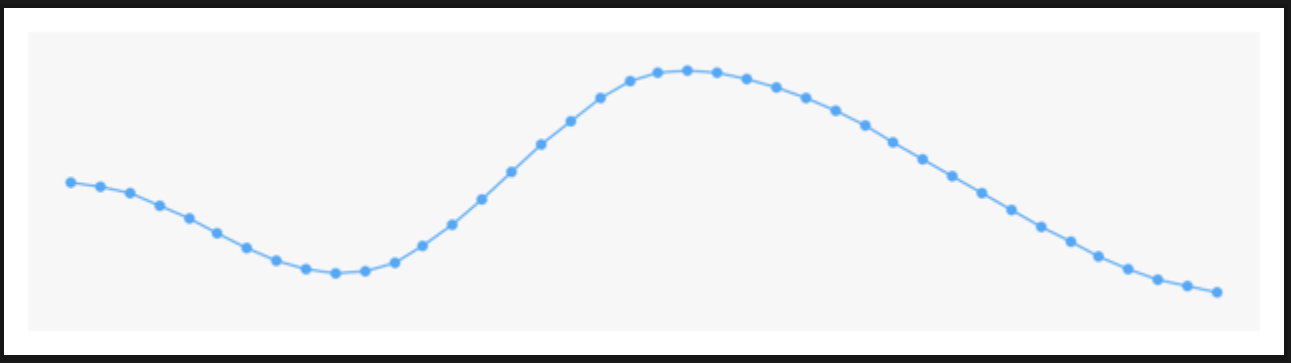
SKKeyframeSequence isn't limited to working only with `SKEmitterNode`, you can use the `sample(atTime:)`method to generate values interpolated between keyframes for other applications. The following code shows how to create a sequence containing four keyframes and how to iterate over the interpolated values:

LISTING 1 CREATING INTERPOLATED VALUES

```
let sequence = SKKeyframeSequence(keyframeValues: [5, 1, 10, 0],
                                  times: [0, 0.25, 0.5, 1])
sequence.interpolationMode = .spline
stride(from: 0, to: 1, by: 0.025).forEach {
    let value = sequence.sample(atTime: CGFloat($0))
}
```

Using `SKInterpolationMode.spline` interpolation, the code above creates the following graph in a playground:

FIGURE 1 PLAYGROUND GRAPH GENERATED BY SKKEYFRAMESEQUENCE



SKKeyframeSequence can also interpolate colors. The following code shows how to initialize a sequence with four colors and sample the interpolated values:

LISTING 2 CREATING INTERPOLATED COLORS

```
let colorSequence = SKKeyframeSequence(keyframeValues: [SKColor.green,
                                                        SKColor.yellow,
                                                        SKColor.red,
                                                        SKColor.blue],
                                       times: [0, 0.25, 0.5, 1])
colorSequence.interpolationMode = .linear
stride(from: 0, to: 1, by: 0.001).forEach {
    let color = colorSequence.sample(atTime: CGFloat($0)) as! SKColor
}
```

The samples for each color generated by the code above can be used to create a gradient:

FIGURE 2 GRADIENT GENERATED BY KEYFRAME SEQUENCE INTERPOLATED COLORS