TweenMaker

Public Member Functions

```
TweenMaker CreateChain ()
```

```
      void SetCustomEasingFunction (Func< float, float > function)
```

Static Public Member Functions

```
static TweenMaker Create (GameObject owner)
```

Public Attributes

```
          float duration = 1.0f
```

```
          float delay = 0.0f
```

```
           bool loop = false
```

```
           bool pingPong = false
```

```
EasingDirection easingDirection = EasingDirection.easeOut
```

```
Action< float > onUpdate
```

```
         Action onComplete
```

Properties

```
Easing easing [set]
```

Detailed Description

A minimal, action-based tween utility for Unity.

What does action-based mean?

With TweenMaker all tweening is done through an update action called `onUpdate`. This action has a single parameter: an eased value that represents how complete the tween is. What you do with that value is up to you; it's perfect for feeding into Unity's Lerp and Slerp functions.

TweenMaker does *not* have functions to move, rotate, scale, fade, set colors, manage objects, fold laundry, etc. You do everything yourself, in the `onUpdate` and `onComplete` functions.

This is better for you(because you don't have to trawl through documentation to find out how to do something), and it's better for me (because I don't have to write the documentation).

Member Function Documentation

```
static TweenMaker Create ( GameObject owner )
```

static

Static constructor.

Parameters

`owner` The Unity GameObject that will host the TweenMaker component.

Returns

The TweenMaker component

This is the principal constructor for TweenMaker; all TweenMaker examples start here.

Example:

```
var tween = TweenMaker.Create(gameObject);
tween.duration = 2.0f;
tween.easing = Easing.Back;
tween.easingDirection = EasingDirection.easeInOut;
```

Although you can use any GameObject to host the component, it's a good idea to use the object that is being animated because deleting that object will also delete any TweenMaker tweens still running.

The component destroys itself when it is complete.

```
TweenMaker CreateChain ( )
```

Creates another TweenMaker tween that will start when this tween completes.

The new TweenMaker is hosted by the same GameObject.

Returns
The chained TweenMaker component.

```
void SetCustomEasingFunction ( Func< float, float > function )
```

Specify the easing function explicitly.

Parameters

`function` The easing function.

Example:

```
tween.SetCustomEasingFunction((p) => {
    return Mathf.Abs(Mathf.Sin(p* Mathf.PI* 6.5f)) * p;
});
```

This is an optional advanced feature that should only be used if you want to spend several days fiddling with math calls.

Member Data Documentation

```
float duration = 1.0f
```

Duration in seconds.

```
float delay = 0.0f
```

Delay in seconds.

```
bool loop = false
```

Whether to continuously loop the tween.

```
bool pingPong = false
```

Whether to pingPong (ie: play backwards after playing forwards).

```
EasingDirection easingDirection = EasingDirection.easeOut
```

Easing direction.

Action<float> onUpdate

Called every update with an eased value between 0 and 1.

This is where most of your tween logic will live.

Typical usage:

```
var startRotation = mainCamera.rotation;
var endRotation = Quaternion.Euler(60.0f, 0.0f, 0.0f);

tween.onUpdate = (t) =>
{
    mainCamera.rotation = Quaternion.SlerpUnclamped(startRotation, endRotation, t);
};
```

Action onComplete

Called when the tween is complete.

Optional

Property Documentation

```
Easing easing
```

set

Specify the easing function by type.

This is the normal way to specify easing.

The documentation for this class was generated from the following file:

- TweenMaker.cs