



uSequencer Documentation V 1.3.7

uSequencer In-Depth	1
Getting Started	1
What is uSequencer	1
Opening uSequencer	2
Creating New Sequences	2
Opening Existing Sequences	2
Automatically playing sequences on Application Start	3
Automatically playing sequences when a trigger volume is entered	3
Sequences	4
What is a sequence	4
Sequence Configuration	4
Duplicating Sequences	5
Scrubbing through a sequence	5
Zooming into parts of a sequence	5
Snapping	6
Starting a sequence	6
Pausing a sequence	6
Resuming a sequence	6
Stopping a sequence	7
Working with sequences through code	8
Sequence Set Up	8
Starting a sequence	9
Pausing a sequence	9
Resuming a sequence	9
Stopping a sequence	10
Skipping a sequence	10
Registering for callbacks on a sequence	10
Changing the playback mode	11
Setting the playback time	12
Affected Objects	13
What are Affected Objects	13
Adding new Affected Objects	13
Removing Affected Objects	14
Changing the Affected Object	14
Changing the Affected Object through code	14
Copying Affected Objects	15

Timelines	16
What is a Timeline	16
Adding Timelines	16
Removing Timelines	17
Copying Timelines	17
Event Timeline	18
What is a Event Timeline	18
Adding Events	18
Removing Events	19
Duplicating Events	19
Manipulating events	19
Observer Timeline	20
What is a Observer Timeline	20
Currently Active Camera	20
Adding new keyframes	20
Removing keyframes	21
Changing a keyframes camera	21
Duplicating a keyframe	21
Manipulating keyframes	21
Ignoring cameras	21
Object Path Timeline	23
What is a Object Path Timeline	23
Adding new path points	23
Removing path points	23
Manipulating Path Points	24
Path Easing	24
Different modes for object rotation	24
Property Timeline	25
What is a Property Timeline	25
Adding properties	25
Reading Properties	25
Removing properties	26
Viewing individual curves	27
Viewing multiple properties	28
Adding Keyframes	28
Modifying Keyframes	28

Modifying Tangents	29
Deleting Keyframes	29
Duplicating Keyframes	29
Extracting the starting value at runtime	29
Animation Timeline (Beta)	31
What are Animation Timelines	31
What is a track	31
Adding new tracks	31
Removing tracks	32
Adding new animation states	32
Removing animation states	33
Duplicating Animation States	33
Blending between multiple animation states	33
loop an animation state	33
Sequences As Prefabs	35
Creating your prefab	35
Updating your prefab	35
Prefab structure	35
Reusing your prefab in scenes.	36
Prefab Affected Object Late Binding	36
Instantiating your prefab using a sequence	37
Instantiating your prefab through code	37
Unity UI Integration	38
Overview	38
PlaySequence.cs	38
StopSequence.cs	38
PauseSequence.cs	38
SkipSequence.cs	39
SequencePreviewer.cs	39

uSequencer In-Depth

Getting Started

What is uSequencer

uSequencer is a powerful timeline creation and manipulation tool, all brought together into a easy to understand interface! You'll benefit from lightning fast performance, ultimate flexibility and beautifully designed professional-level features.

uSequencer has been built from the ground up to be a user friendly creation first tool. We want people to unleash their inner creativity and we believe that you now have the right tools to do so!

ABOUT USEQUENCER

The product possesses an impressive selection of professional-level concepts and features aimed at pushing the boundaries of your game and development process.

uSequencer is developed by a developer with years of professional experience in tools development and interactive entertainment production and brings three years of experience and creativity into your hands, we know you'll enjoy using uSequencer.

KEY USES

- Cutsscenes
- Background Animations
- Animation synchronisation
- QTEs
- Product Trailers
- On Rails Gameplay sections
- Professional Fly-throughs
- Anything else you can imagine!

uSequencer is a tool that will forever thrive on providing users what they want, we live and die by our users and if you need something from us, we'd love to hear from you!

You can contact us at any time through one of our many channels.

- Email support@wellfired.com
- Twitter <https://twitter.com/WellFired>
- Facebook <http://facebook.com/wellfired>
- Unity Forums <http://forum.unity3d.com/threads/unity-cutsscene-creator-camera-path-usequencer-released.140440/>
- Our Forum <http://usequencer.freeforums.org/>
- Our User Voice Page <http://usequencer.uservoice.com>

Opening uSequencer

There are two ways you can open uSequencer.

1. You can press the key combination cmd (or ctrl) + U.
2. You can open uSequencer through Unity's Menu Items at Window / Well Fired Development / uSequencer / Open uSequencer

Once uSequencer is open you will see that it acts like any other Unity window, it can be resized, docked, minimised, maximised or closed.

uSequencer will at first seem quite complex. You will however notice if you don't understand something, you can hover your mouse pointer over it and receive a useful tooltip!

Creating New Sequences

When you have uSequencer open, you can create new sequences very simply by clicking on the Create New Sequence button.

A rectangular button with a light gray background and a thin border. The text "Create New Sequence" is centered on the button in a dark gray, sans-serif font.

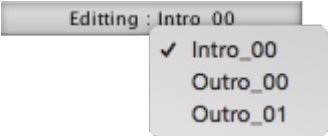
Opening Existing Sequences

You have two ways of opening already created sequences. The first is to select any sequence you have in Unity's hierarchy. This will automatically select that sequence for editing in uSequencer.

The easier way to open existing sequences is to use the quick selection menu. uSequencer's UI will display the sequence you are currently editing in its UI.

A rectangular button with a light gray background and a thin border. The text "Editing : Intro_00" is centered on the button in a dark gray, sans-serif font.

If you click this UI element, you will be presented with a popup showing you all sequences in the currently open scene.

A screenshot of the quick selection menu. It shows a button labeled "Editing : Intro_00" with a dropdown menu open. The dropdown menu is a light gray box containing three items: "Intro_00" with a checkmark, "Outro_00", and "Outro_01".

Sequence Name	Selection Status
Intro_00	Selected (checked)
Outro_00	Not selected
Outro_01	Not selected

You can simply click on any of the other sequences in this list to change between sequences in your scene.

Automatically playing sequences on Application Start

One of the next questions you might have is, “How can I play this new sequence when Play Mode is entered”.

Usually, this would require a single line of code, but we’ve taken away that necessity, providing you with a nice, simple way to start sequence Playback automatically.

Inside your uSequencer directory, you will find a folder called Quick Start Prefabs, inside this folder will be a simple prefab called Auto Play uSequence. If you drag this prefab into your scene, you can connect the sequence you’d like to play in Unity’s inspector.

Automatically playing sequences when a trigger volume is entered

Sequences can easily be played when an object enters a trigger volume, we at WellFired have taken away the hard work for you!

Inside the Quick Start Prefabs folder, you’ll see a number of useful prefabs, that you should be able to use out of the box. Simply drag these into your scene and set them up in Unity’s inspector! They even have the Collider needed on them!

PLAYER USEQUENCE TRIGGER

If you place one of these in your scene, the specified sequence will be triggered when an object with the Player Tag enters the collider on this object

In the inspector, you will need to set-up

- The Sequence To Play (Sequence) - This is the sequence you’d like to play when the main camera enters the collider on this object.

MAIN CAMERA USEQUENCE TRIGGER

If you place one of these in your scene, the specified sequence will be triggered when an object with the MainCamera tag enters the collider on this object

In the inspector, you will need to set-up

- The Sequence To Play (Sequence) - This is the sequence you’d like to play when the main camera enters the collider on this object.

OBJECT USEQUENCE TRIGGER

If you place one of these in your scene, the specified sequence will be triggered when a given object enters the collider on this object

In the inspector, you will need to set-up

- The Sequence To Play (Sequence) - This is the sequence you’d like to play when a given object enters the trigger volume on this object.
- The Trigger Object (GameObject) - When this object enters the collider on this object the sequence specified will be triggered

Sequences

What is a sequence

You can think of a sequence as a base object. The sequence drives all logic associated with a sequence. by default, they will update with your application flow, running from 0 seconds to x seconds in a linear fashion.

By default sequences live in your scene, and are essentially game objects, like everything else in your scene.

If you select a sequence in the Unity Hierarchy, the uSequencer window will update and start modifying that sequence automatically.

For the code-inclined the sequence class is WellFired.USSequencer

Sequence Configuration

uSequencer's sequences are quite complex, and they have a variety of customisation options. You can completely configure a sequence in the uSequencer window or in Unity's inspector. Here are some important configuration options.

NAME

You can rename a sequence by renaming the GameObject in Unity's Hierarchy

TICK WITH A FIXED UPDATE

If you want your sequence to update every fixed framerate frame, you should enable this option in Unity's Hierarchy. This might be useful if you need uSequencer to interact with rigidbodies, or you're working with uSequencers physics events, or, if you'd like your sequences to have direct frame by frame control.

DURATION

You can adjust the duration of the sequence, from it's default 10 seconds by altering the duration in the uSequencer UI.

PLAYBACK RATE

You can adjust the playback Rate, from it's default 1 by altering the playback rate in the uSequencer UI.

The playback rate describes how quickly the sequence will playback, if 1 is the default value, 2 is 2x as fast, 4 is 4x as fast, and so on. In addition to speeding up a sequence you may also make it play in reverse by adjusting the playback rate to a negative value.

PLAYBACK MODE

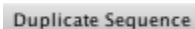
You can adjust the playback Mode directly within the uSequencer UI, Sequencer supports three different types of playback mode.

1. Normal - Sequences will play from start to end, then stop.
2. Looping - Sequences will play from start to end, then start again.

3. Ping Pong - Sequences will play from start to end, then in reverse, from end to start, this behaviour will loop.

Duplicating Sequences

Most sequences shouldn't be duplicated using Unity's in-built duplication functionality, instead, you can duplicate sequences directly in the uSequencer UI. To do this, first ensure you select the sequence you'd like to duplicate and then press the Duplicate Sequence button within the uSequencer UI.

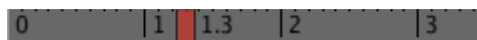
A rectangular button with a light gray background and a thin border, containing the text "Duplicate Sequence" in a dark gray sans-serif font.

Scrubbing through a sequence

One of the first things you're likely going to want to do is preview your sequence in Edit mode. This is easy to do with uSequencer and every single thing in uSequencer is previewable in edit mode.

You can scrub through a sequence in one of two ways, both of these have the same outcome.

- 1) You can click and drag on the scrub bar. The scrub bar is seen as the timeline view, it has the time markers on it and the red scrub head representing the current playback position. It looks something like this :



as you can see, the scrub head is at 1.3 seconds, representing that you are currently previewing your sequence at 1.3 seconds. If you'd like to preview your sequence at 2 seconds for instance, you can simply click where the time marker for 2 is.

- 2) You can manually set the RunningTime for more accurate control over what you are previewing. To specify the Running Time you can type directly into the Running Time field at the bottom of the uSequencer window.

A rectangular input field with a light gray background and a thin border. It is divided into two sections: the left section is labeled "Running Time" and the right section contains the numerical value "1.3".

Zooming into parts of a sequence

Sometimes, when you have a long sequence or you want fine grain control over your sequence, the default view is not sufficient. Suppose you want to 'zoom' into your sequence at 4 seconds, you can move your mouse over the scrub bar timeline marker at 4 seconds and scroll with your scroll wheel.

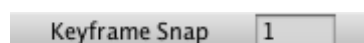
This should zoom in at 4 seconds, providing you with much more detail and allowing you to accurately edit your sequences.

Snapping


Everything that can be moved in uSequencer can be snapped in uSequencer. Snapping is often necessary when you want to line up many items in a sequence without manually typing exact values.

You can enable snapping in uSequencer by holding down the left shift button.

The default snapping value is 1, which will snap anything at regular 1 second intervals. Of course, 1 second might not be a perfect value for your use case, so you can modify this in the uSequencer UI. Simply by altering the Keyframe Snap.




Starting a sequence

If you want to sample the playback of a sequence, you can do this easily by pressing the  button. You'll notice when you press this button that it will change to a pause button and will turn red, this is to indicate to you that we're now in preview mode. Certain actions will be restricted in preview mode.


When you enter preview mode, the starting value of all objects in the sequence is captured and will be restored when you exit preview mode.

Pausing a sequence

When you're playing back a sequence, you might want to pause the playback, this can be easily achieved by pressing the  button. You'll notice, when paused the button remains red and changes to a Play button.


When the sequence is paused, preview mode is still active.

Resuming a sequence

If you've paused a sequence playback and you want to resume it from the current position, you can achieve this by pressing the  button.

When the sequence is running, preview mode is still active.

Stopping a sequence

When you are previewing a sequence in edit mode, you can stop the playback at any time, by pressing the  button. You'll notice when you press this button that uSequencer exits preview mode and all animated objects will be reverted back to their base stored state.

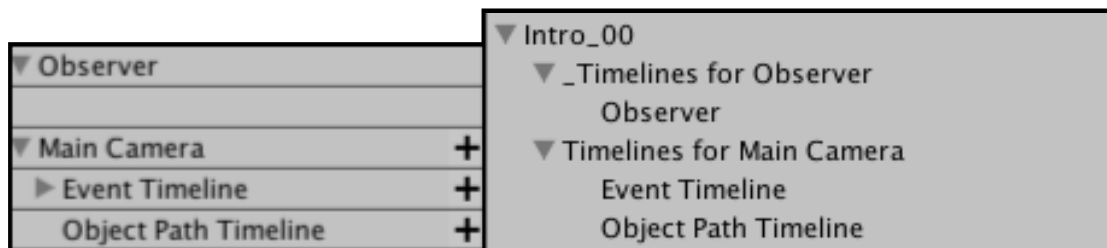
When you Stop a sequence, preview mode is exited and all objects are reverted back to their base states.

Working with sequences through code

Sequence Set Up

Sequences by default take full advantage of Unity's parent child system, a sequence object will contain lots of child objects (who will in turn contain child objects) that combined represent an entire sequence.

If you were to look at Unity's Hierarchy window, a sequence' child game objects would directly match what you see in uSequencer.



On the right is Unity's inspector and on the left is uSequencer's representation. The GameObject intro_00 doesn't show up in uSequencer, since that object is the Sequence you are viewing, but if you'll look at the other objects, you'll see they map directly. For completeness, I've added the class names in a table below. If this means nothing to you, simply ignore this information

uSequencer	Unity's Hierarchy
Observer	_Timelines for Observer (USTimelineContainer)
(Blank)	Observer (USObserverTimeline)
Main Camera	Timelines for Main Camera (USTimelineContainer)
Event Timeline	Event Timeline (USTimelineEvent)
Object Path Timeline	Object Path Timeline (USTimelineObjectPath)

One nice thing about this representation is that from any sequence object (USSequencer / USTimelineContainer / USTimelineEvent / USTimelineObjectPath), you can directly access any parent or child object.

For example, anything beneath the sequencer object, will be able to access it's containing sequence through a Sequence property :

```
USTimelineEvent.Sequence
USTimelineObjectPath.Sequence
USTimelineEvent.Sequence
USTimelineContainer.Sequence
```

Starting a sequence

If you want manual control over your sequence playback, you can achieve this quite simply.

Supposing you have a reference to a sequence and you want to play the sequence at runtime, through code, this can be achieved simply by calling the Play method, as demonstrated below.

```
using WellFired;

public class SampleSequenceControl
{
    public USSequencer sequence;

    public void StartSequence()
    {
        sequence.Play();
    }
}
```

Pausing a sequence

If you have a sequence that is currently playing back, you can pause playback of that sequence by simply calling the pause method, as demonstrated below.

```
using WellFired;

public class SampleSequenceControl
{
    public USSequencer sequence;

    public void PauseSequence()
    {
        sequence.Pause();
    }
}
```

Resuming a sequence

Resuming a paused sequence is a trivial affair, you simply need to call the Play method again, as demonstrated below.

```
using WellFired;

public class SampleSequenceControl
{
    public USSequencer sequence;

    public void ResumeSequence()
    {
        sequence.Play();
    }
}
```

Stopping a sequence

If you have a sequence that is playing back you can stop that sequence from playing back by simply calling the Stop method, as demonstrated below

```
using WellFired;

public class SampleSequenceControl
{
    public USSequencer sequence;

    public void StopSequence()
    {
        sequence.Stop();
    }
}
```

Skipping a sequence

It's quite common to skip sequences, maybe you've made a cutscene and implemented a skip cutscene button, or maybe you've some logic that has determined you no longer need to see the rest of the executing sequence.

Skipping a sequence is trivial and can be achieved with the SkipTimelineTo method, as demonstrated below.

This example will skip a sequence to the end, using the Property sequence.Duration.

```
using WellFired;

public class SampleSequenceControl
{
    public USSequencer sequence;

    public void SkipSequence()
    {
        sequence.SkipTimelineTo(sequence.Duration);
    }
}
```

Registering for callbacks on a sequence

If you're manually controlling your sequence it's often quite useful to receive notifications when certain criteria have been fulfilled. For example, if you want certain properties to apply to an object at the start of a sequence and then to un-apply when playback has finished, you can achieve this through callbacks.

Registering for callbacks on a sequence is demonstrated below.

```

using UnityEngine;
using WellFired;

public class SampleSequenceControl
{
    public USSequencer sequence;

    public void RegisterCallbacks()
    {
        sequence.PlaybackStarted += SequenceStarted;
        sequence.PlaybackPaused += SequencePaused;
        sequence.PlaybackFinished += SequenceFinished;
        sequence.PlaybackStopped += SequenceStopped;
    }

    private void SequenceStarted(USSequencer sequence)
    {
        Debug.Log("{0} Has just started.", sequence.name);
    }

    private void SequencePaused(USSequencer sequence)
    {
        Debug.Log("{0} Has just been pause.", sequence.name);
    }

    private void SequenceFinished(USSequencer sequence)
    {
        Debug.Log("{0} Has finished playing back.", sequence.name);
    }

    private void SequenceStopped(USSequencer sequence)
    {
        Debug.Log("{0} Has been stopped.", sequence.name);
    }
}

```

Changing the playback mode

As well as changing the playback mode through the uSequencer UI, you can also do it through code, as demonstrated below.

```

using UnityEngine;
using WellFired;

public class SampleSequenceControl
{
    public USSequencer sequence;

    public void MakeSequenceLoop()
    {
        sequence.IsLopping = true;
    }

    public void MakeSequencePingPong()
    {
        sequence.IsPingPonging = true;
    }
}

```

Setting the playback time

It is quite possible that you will want fine control over the playback of your sequence. You may want to implement custom controls or provide specific UI to a user, for example.

It is also possible to set the playback time of a sequence at runtime, through code, as demonstrated below.

```
using WellFired;

public class SampleSequenceControl
{
    public USSequencer sequence;

    public void SetPlaybackTime(float newPlaybackTime)
    {
        sequence.RunningTime = newPlaybackTime;
    }
}
```


Affected Objects

What are Affected Objects

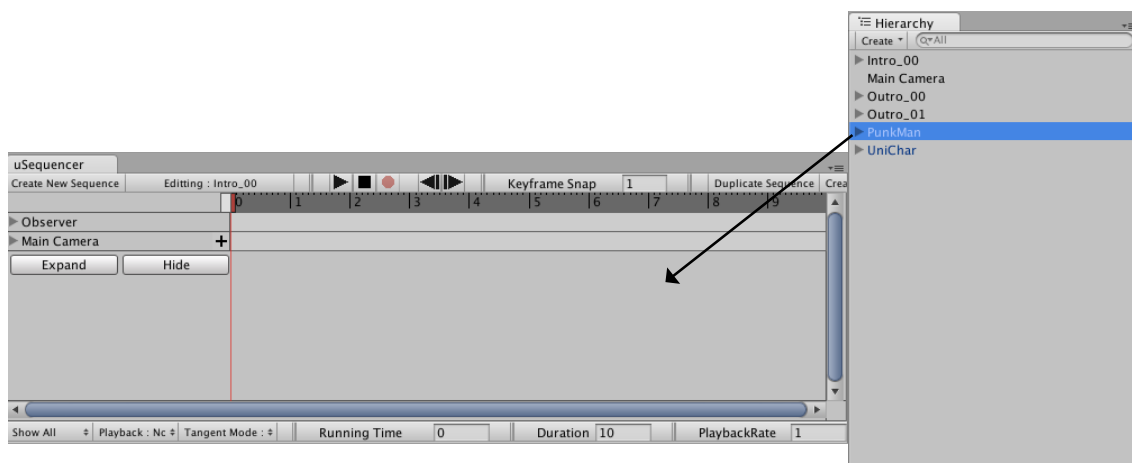
An affected object is any game object you add to a sequence. When creating sequences with uSequencer, you will first of all, add an object to the sequence.

Without knowing which objects to work on, uSequencer cannot do anything. Affected Objects are the starting point of your Sequencing.

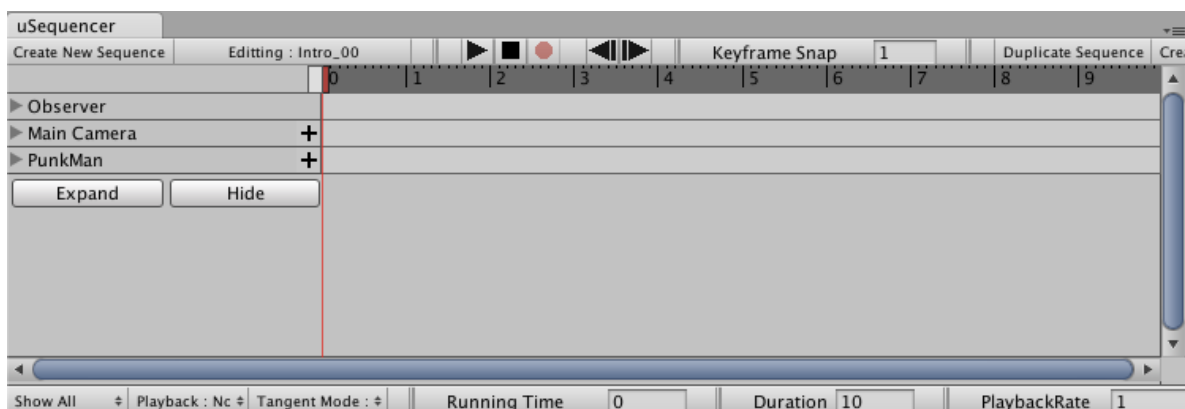
Suppose you want to animate a camera, you will need to add the camera to uSequencer, this camera then becomes an Affected Object. uSequencer will Affect this object in many different ways, outlined in the Timelines chapter.

Adding new Affected Objects

Adding new affected objects is a trivial matter, you simply drag them from the Unity Hierarchy into the uSequencer window.



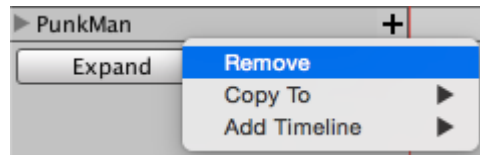
Once you've dragged your object into the uSequencer window, you will see you now have one more affected object in your uSequencer Window



Removing Affected Objects

In the same way that you might want to add Affected Objects, you will likely want to delete them also. This can be achieved through the Right Click Menu.

Right Click on your Affected Object in the uSequencer window and select Remove. This will remove your affected object from this sequence.



Changing the Affected Object

An Affected Object might be affected by animations, paths, events or anything else, and you may decide that you'd prefer to have this set of actions happen to another Affected Object than the one you initially decided to use.

Changing the Affected Object is simple. Suppose you have an Affected Object Camera 1, but you've decided you want to change it to Camera 2.

- 1) Select your Camera 2 in Unity's hierarchy
- 2) Right click on your Camera 1 in uSequencer
- 3) Select Update Affected Object : Camera 2

This is a really powerful feature, and it is often quite common to duplicate an entire Affected Object and then reassign it to a new Affected Object

Changing the Affected Object through code

It is also possible to change an affected object through code at runtime. The process for doing this is quite simple. You will need a reference to a USTimelineContainer object, this object contains the reference to a Affected Object that uSequencer uses.

Changing the affected object is trivial, as outlined below.

```
using WellFired;

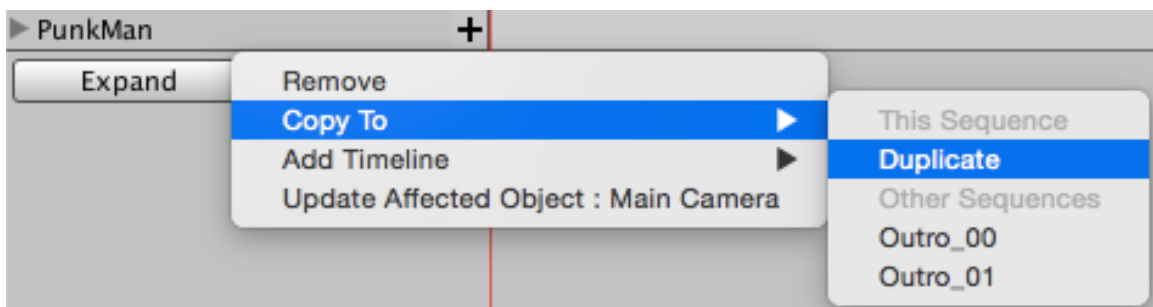
public class SampleSequenceControl
{
    public USTimelineContainer affectedObjectContainer;

    public void SetAffectedObject(Transform newAffectedObject)
    {
        affectedObjectContainer.AffectedObject = newAffectedObject;
    }
}
```

Copying Affected Objects

Sometimes it's desirable to copy an affected object and everything that affects that object. uSequencer knows of two copying methods. The first is an in-place copy, which is effectively a duplicate, the second is a send copy, which will copy your Affected object from one sequence to another.

Both are done quite simply through the uSequencer UI, simply by right clicking on the Affected Object you'd like to copy and selecting Copy to.



Copy To / Duplicate - will duplicate the Affected Object so you'll have two of them in one sequence.

Copy To / Sequence Name - will duplicate the Affected Object into another sequence.1515

Timelines

What is a Timeline

A timeline is a sequence of actions that happen to an Affected Object over a given period of time. Everything that uSequencer does involves a timeline of some sort. uSequencer comes with a selection of timelines for you to use.

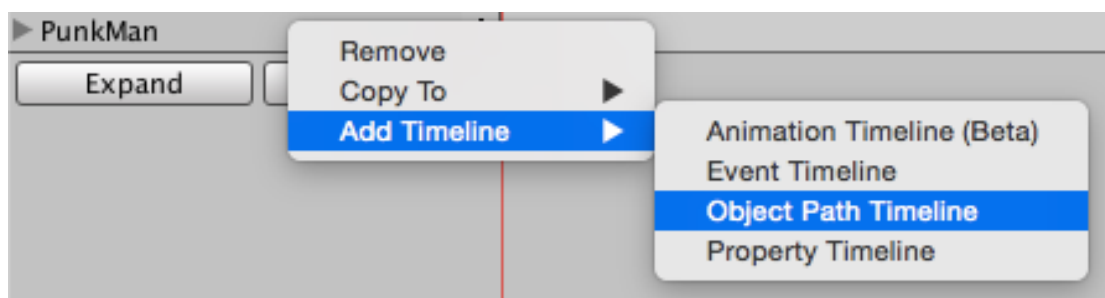
- Event Timeline - Powerful event timeline that can interact with any code. You can add your own events or choose from uSequencers built-in library of over 100 events.
- Object Path Timeline - The Object Path timeline is a timeline dedicated to creating and controlling paths for objects. This can be used for camera paths, but is not in anyway restricted to being used on cameras.
- Observer Timeline - Allowing you to direct the action, providing you with an easy, central way to maintain camera cuts.
- Property Timeline - The ultimate timeline, allows you to animate any property on any object, no limitations.
- Animation Timeline (beta) - One single timeline dedicated to all your animation needs, this is fully integrated with Mecanim and is completely edit mode preview-able.

If you're feeling adventurous, it's also possible to create your own custom timelines as seen in the technical documentation. http://www.wellfired.com/technicaldocumentation/usequencer/html/index.html#custom_timelines

Adding Timelines

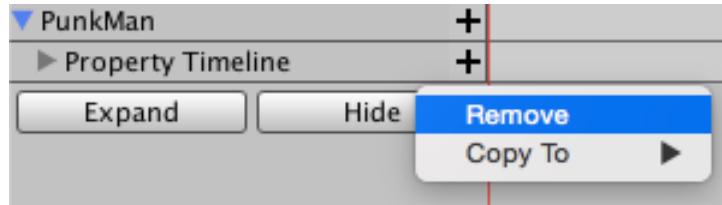
Of the entire collection of timelines, the only one that cannot be added / removed is the observer timeline, it's added by default to any sequences the user creates.

Adding any other timeline (Event / Animation / Object Path / Property) to an Affected Object is simple and can be done through the uSequencer Right Click Menu. Simply Right click on any Affected Object and select Add Timeline, as demonstrated below.



Removing Timelines

Along with adding new timelines to an Affected Object you will likely want to remove timelines. You can remove timelines from an Affected Object by right clicking on the timeline you'd like to remove and selecting remove.

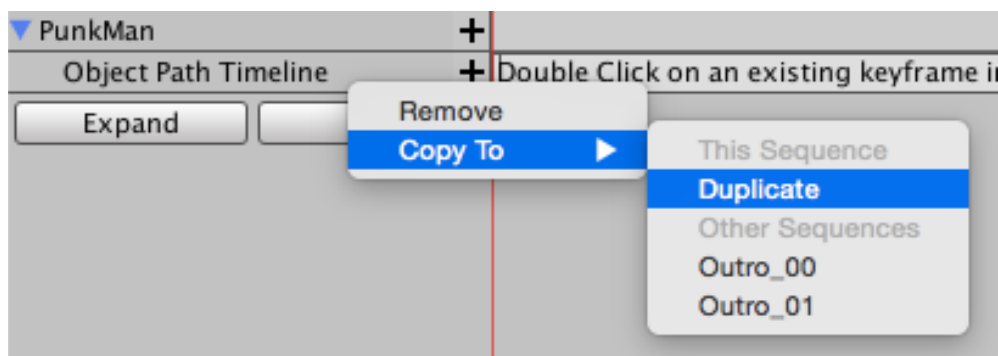


Copying Timelines

You've seen that it can be quite easy to add and remove timelines, but how about copying timelines? It actually works in the same way as copying Affected Objects.

uSequencer knows of two copying methods. The first is an in-place copy, which is effectively a duplicate, the second is a send copy, which will copy your Timeline from one sequence to another.

Both are done quite simply through the uSequencer UI, simply by right clicking on the Timeline you'd like to copy and selecting Copy to.



Copy To / Duplicate - will duplicate the Timeline so you'll have two of them in one sequence.

Copy To / Sequence Name - will duplicate the Timeline into another sequence.1717

Event Timeline

What is a Event Timeline

Event timelines give the user the power of over 100 different events, from a number of different categories. With an event timeline you have access to quick chunks of logic, this logic has already been written for you and you just need to utilise it!

You can, for instance:

- Play a legacy animation
- Open a new scene
- Apply some physics
- Run some particles
- Play some audio
- Spawn new objects
- Run special rendering effects
- Send messages to your runtime code.
- Plus many more!

Events in an event timeline generally all expose a Fire Time, this Fire Time is the time at which the logic for this event will be executed. Most events are Fire And Forget, meaning they just execute logic once and never again, but some can have a duration.

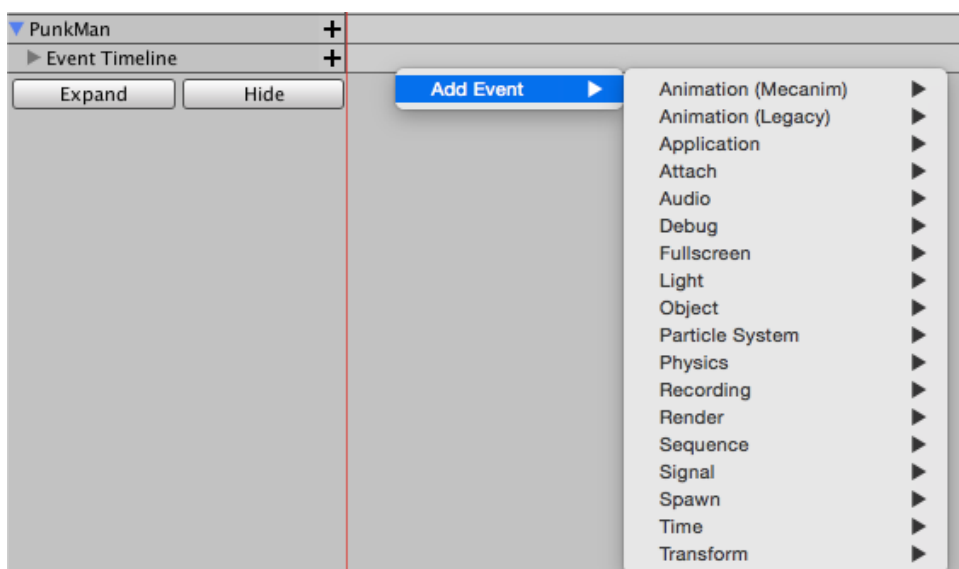
As well as uSequencer providing you with a list of over 100 events, you can of course, add your own events, have a look at our step by step guide on adding custom events http://www.wellfired.com/technicaldocumentation/usequencer/html/index.html#custom_events

Event Timelines can be added by right clicking the affected Object and selecting Add Timeline / Event Timeline.

Adding Events

Once you've added a new Event Timeline to uSequencer, adding events is really simple and again, done through uSequencers right click menu.

Now that you've got an event timeline, if you right click on the event timeline and select Add Event, you'll be presented with a list of categories, as shown below.

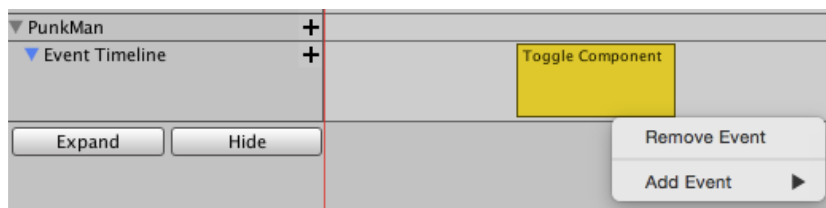


Each of these event categories has a handful of events, to add the one you'd like simply click on it.

Removing Events

As well as adding new events, you can also remove existing events, there are two ways to remove events. First you must select the event you'd like to remove.

Then you can either press the backspace or delete key or right click on the event and select Remove Event.



Duplicating Events

uSequencer allows the user to duplicate anything in any timeline with a simple key combination. Events in an event timeline are no different.

If you select multiple objects in an timeline and hold down the key combination Alt + Ctrl then drag your selection, you will duplicate that selection. The duplication works across many different types of objects, not just events.

Manipulating events

Events can be manipulated in the uSequencer window by dragging them around. When you drag an event in the uSequencer window, you are manipulating its Fire Time.

You can also manipulate any event by selecting it in the uSequencer window and then editing the event in the inspector.

Not every event has the same inspector parameters, but every event should have the following options.

Fire Time - At what time will this event fire.

Fire On Skip - Should this event be triggered if the cutscene has been skipped.

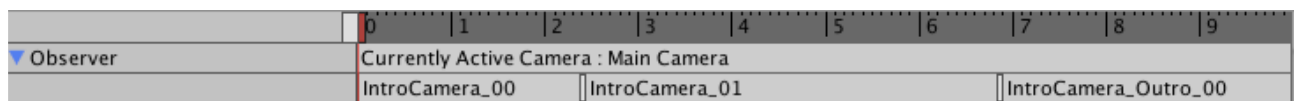
Observer Timeline

What is a Observer Timeline

An observer timeline is a timeline within uSequencer that is dedicated to providing the user a nice interface for dealing with camera cuts.

Unity tells us that only one camera must be active at a time, uSequencer will try to ensure this is always true, by enabling and disabling cameras for you. This will be achieved by toggling on and off the camera component on any cameras in your scene.

An example of a populated observer timeline.

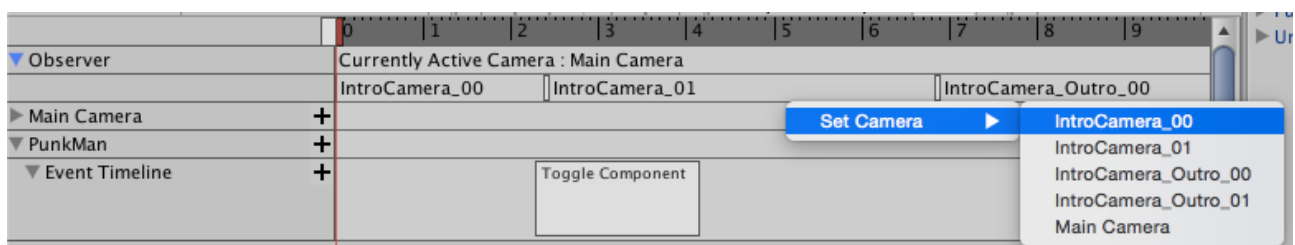


Currently Active Camera

The Observer Timeline will always show you the currently 'active' camera, or the currently enabled camera, so that, at a quick glance you can see what camera is currently rendering.

Adding new keyframes

Each keyframe in the Observer Timeline represents a camera cut. when a sequences running time passes the fire time of a keyframe, the camera cut will occur. Adding new keyframes to the observer timeline is straightforward, simply right click at the time you'd like to add a new keyframe and select Set Camera / (your camera). As seen below :



This would, for example add a new keyframe at 5 seconds, setting the camera to IntroCamera_00.

Removing keyframes

Keyframes can be removed from the observer timeline in a number of ways.

- 1) If you select the keyframe you'd like to remove and press the backspace or delete key.
- 2) Right click on the keyframe you'd like to remove and select Remove Observer Keyframe.

Changing a keyframes camera

If you have a keyframe setup already and would like to edit the camera cut, right click on the keyframe you want to change and select Set Camera / (your new camera)

Duplicating a keyframe

uSequencer allows the user to duplicate anything in any timeline with a simple key combination. Keyframes in an observer timeline are no different.

If you select multiple objects in an timeline and hold down the key combination Alt + Ctrl then drag your selection, you will duplicate that selection. The duplication works across many different types of objects, not just observer keyframes.

Manipulating keyframes

Observer keyframes can be edited by dragging them left and right in the Observer Timeline, this will affect the FireTime of the keyframe. You'll notice also that as you drag the keyframe left or right that the Currently Active Camera will update to reflect the current situation

Ignoring cameras

Sometimes, you will want the observer timeline to ignore certain cameras (UI Cameras, for instance). You can do that simply by putting those cameras on their own layer and ensuring uSequencer doesn't modify cameras on those layers, as described below.

- 1) Ensure the cameras you want to ignore are on their own layer. (If you're unfamiliar with Unity's layering concept, the following page will help you understand it better : <http://docs.unity3d.com/Documentation/Components/Layers.html>)
- 2) Locate your sequence in the hierarchy, expand it.
- 3) Your sequence will have a child object named : "_Timelines For Observer" expand it also.
- 4) The _Timelines For Observer object will have a child named Observer, select this.

- 5) Unity's inspector should be updated now, to show you the properties for this observer timeline.
- 6) The inspector will have a property "Layers To Ignore", you should now set this to ignore the layer you created in step 2.

Object Path Timeline

What is a Object Path Timeline

Most often, when you're creating a sequence, you'll want at least one object to follow a path.

A path is selection of connected 3D nodes that have an distinct start and end. Each of these nodes is connected with a curve in 3D space. Each node is connected to the next node, either linearly or by a curve.

When creating paths, you can also optionally alter the easing whilst moving along the path and you can control the orientation of the object.

Upon adding a new Object Path Timeline, two keyframes will be automatically added for you at the position of your Affected Object.

Adding new path points

Once you have an Object Path Timeline in your sequence, you should see two keyframes in your Unity scene view at the position of your Affected Object with the text Start and End next to them, as shown below.



You are free to move around these Start and End positions in the Scene View.

Of course, a curve isn't a curve with only two points. You can add more path points to a curve by double clicking on an existing path point in the Scene View. There are no restriction to how many path points you can add to an Object Path Timeline.

Removing path points

Removing Path Points is as simple as adding them, simply click on the Path Point you want to remove and press the backspace or delete key.

Manipulating Path Points

Path points can be dragged around in the scene view.

Each path will also have two points that aren't part of the path, one at the start and one at the end, these are the tangent points and just help you to control the shape of your curve.

Path Easing

Each path has easing options associated with it, you can select from over 30 easing options, such as Ease-In-Ease-Out, Ease-In, or Bounce.

Changing the easing options on an Object Path Timeline is easy.

- 1) Click the word Object Path Timeline in the uSequencer window, which should select the object path timeline for you in Unity's Hierarchy.
- 2) In Unity's inspector window, modify the Easing Type, choosing your desired easing method.

You can scrub or playback your sequence at any time so you can live preview your easing changes.

Different modes for object rotation

Each Affected Object on an Object Path can optionally have it's orientation maintained by the Object Path Timeline. One very common use case, is that you would move an object along a path and look at a static position.

You have three options when controlling object rotation on an object path.

- 1) Manual Orientation - The Object Path Timeline will not modify the objects orientation in any way, leaving that, instead up to the designer.
- 2) Look At Transform - The Affected Object will be looking at a specific object for the duration of This Object Path Timeline. The object you look at can be anything with a Transform component, the object can be static or moving, the Object Path Timeline will look at it regardless.
- 3) Look Ahead - The Object moving along this Object Path Timeline will look forward along the Path.

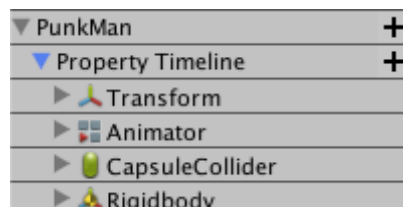
Property Timeline

What is a Property Timeline

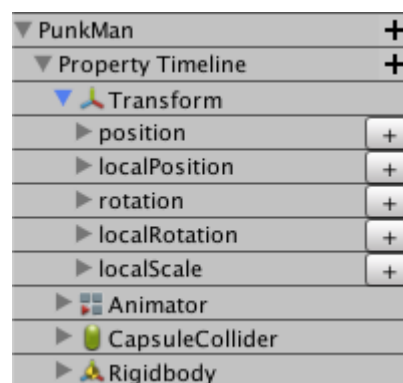
A property timeline gives you direct control over any object in your game. You can actually animate anything, with complete control over the animation curve. Property Timelines are without a doubt the most powerful feature of uSequencer, providing you an easy to understand interface whilst giving you all the control.


Adding properties

When you've added a property timeline to an Affected Object, the next step will be to add some properties. When you expand the property timeline, you'll see you have all the components on your Affected Object.



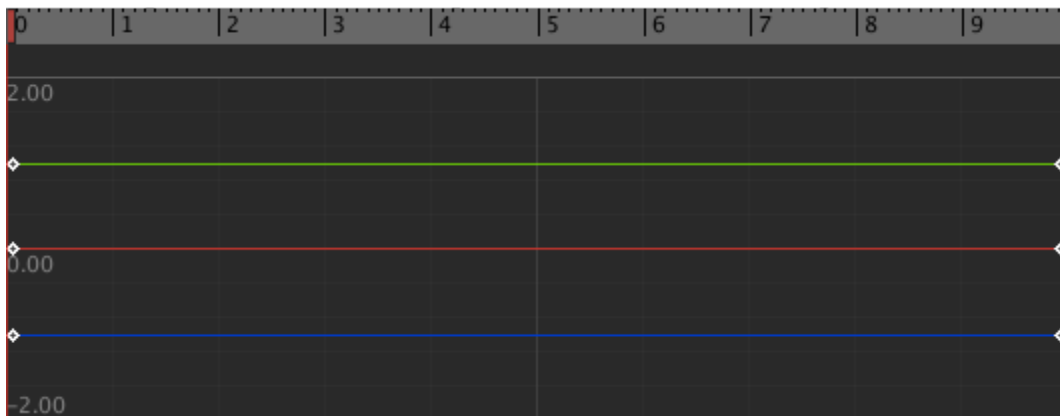
You can animate any value on any component listed under your affected object. Suppose for instance, you want to animate the local position of your Affected Object, you can expand the Transform component.





You can see that every property on the Transform component is listed below the transform component. (position, localPosition, rotation, localRotation and localScale). To animate the localPosition, you simply press the  button.




Reading Properties


Property animations are visualised as curves in the right hand pane, when you add a new property, you will be presented with a curve editing area in the uSequencer UI.

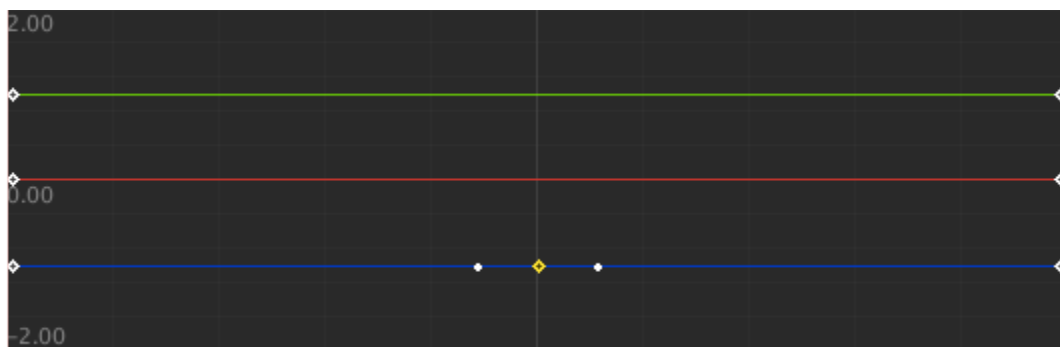


uSequencer automatically builds curves for all property components. The property `localPosition` has three components. (x, y and z), you'll see that the curves are effectively a straight line, that's because all of the  (keyframes) have the same value.


Each  can have a different value this would in turn modify the curve.

Each  has it's own unique tangent. A curve is simply a bunch of  with unique tangents. The tangent describes how the value will change as time flows over that .

When you select a  the colour of that  will change, and the tangent will also be visible, you can see from this image I've added a new  and selected it, so that it's tangent is visible.



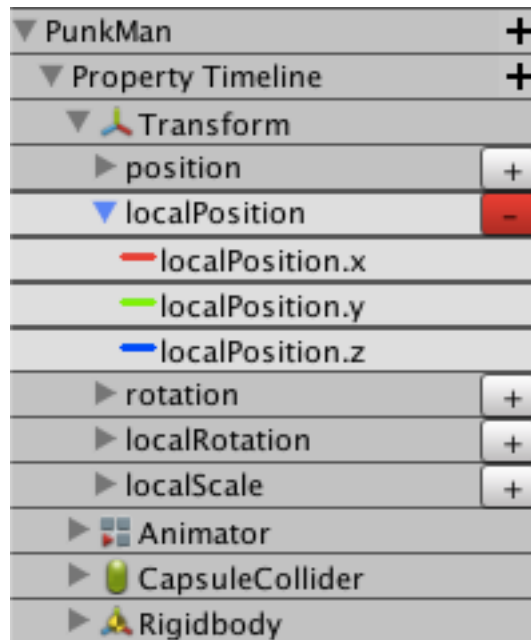
Removing properties

If you need to remove any properties you no longer need (and all curves associated with them), you can do so simply by pressing the  button that should be right next to the property name.

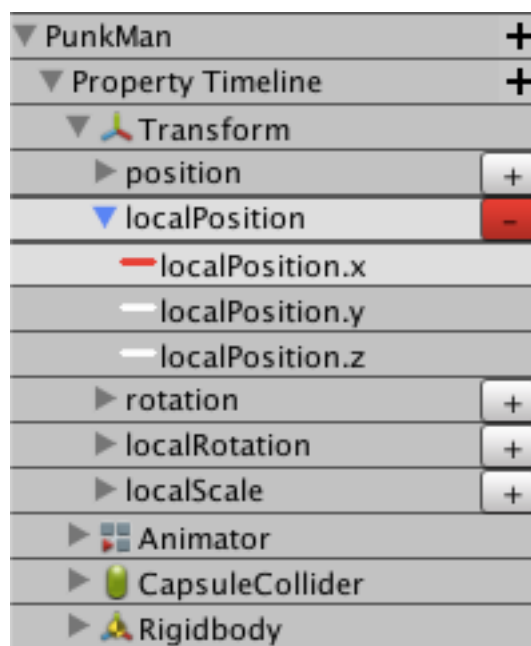
Viewing individual curves

If you're modifying a property that has more than one curve, or you want to view just a small subset of those curves, you can easily do that.

For example, using the `localPosition` as an example again, expand the property that you'd like finer grain control over.



For example, if you only want to view the red `localPosition.x` curve, you can simply deselect the others by clicking on them. Click on `localPosition.y` and `localPosition.z` to deselect them.





The curve area should now be updated to reflect this change and will only be displaying the red curve.

Note that all actions that occur will only occur to any curves that are visible. If you for instance add / remove / edit keyframes, these changes will only apply to the visible curves. This is a very useful feature.


Viewing multiple properties





As well as viewing individual curves you can view multiple different properties across multiple different components if you'd like to, all you need to do is click on any component or curve you'd like to view / edit, they will all be combined in the same curve editing space.

Adding Keyframes

A curve is useless without  on it. There are a number of ways to add  to an existing curve. They generally involve scrubbing to a specific time first of all.





using the localPosition as our example, if you start by ensuring the curves for localPosition are visible, then scrubbing your sequence to 5 seconds.


You now have three choices for adding  at the given time.

- 1) Pressing the k key on your keyboard will automatically add  at the scrub heads position for all visible curves.
- 2) If you modify the value from within Unity, for example, using unity's inspector to manipulate your Affected Objects position,  will automatically be added and / or updated to the new values for you. This is our preferred method for adding .
- 3) If you right click on the curve editor at a given time, you will see the option Add key. This option will automatically add a  at the given time.





Modifying Keyframes

You can modify  in three ways.


- 1) Drag them around in the curve editor, this will let you edit the fire time and the value of that .
- 2) Clicking on a  will expose editable properties in Unity's Inspector, you can modify these directly.
- 3) If you single click on a  uSequencer will automatically set the scrub head to that time. You can now modify the property directly within Unity, for instance, in the inspector. This will automatically modify any visible  at that time for that component. This is the preferred method for modifying keyframes.

You can modify  using methods 2 and 3 whilst multi selecting.

Modifying Tangents

As well as modifying , you can also modify a  tangents. In order to see a  tangent, it must be the only thing selected in uSequencers UI, in order to achieve this, left click anywhere in the uSequencer window where there is nothing to click on, then single click on the  you'd like to modify the tangents for.

You now have three options for modifying the tangents.

- 1) You can drag the tangents up and down to modify them directly.
- 2) You can modify the value of the tangents in Unity's Inspector.
- 3) You can right click on any  keyframe to be given a list of all predefined tangents options.

Flatten - Flattening a keyframe will give you a nice ease in / ease out for that transition.

Smooth - If you smooth a keyframe, the entry and exit to that keyframe will be nice and smooth

Tangent Linear - This will give you a nice linear transition to the next value, the linear transition will provide you a constant speed.


Tangent Constant - A constant tangent will give you one value immediately until the next keyframe is passed, then the new value will snap.

You can modify Tangents using methods 2 and 3 whilst multi selecting tangents.

Deleting Keyframes

Keyframes can also be deleted, simply by selecting the keyframes you'd like to delete, then pressing the backspace or delete key on your keyboard.

Duplicating Keyframes

uSequencer allows the user to duplicate anything in any timeline with a simple key combination.  in an property timeline are no different.

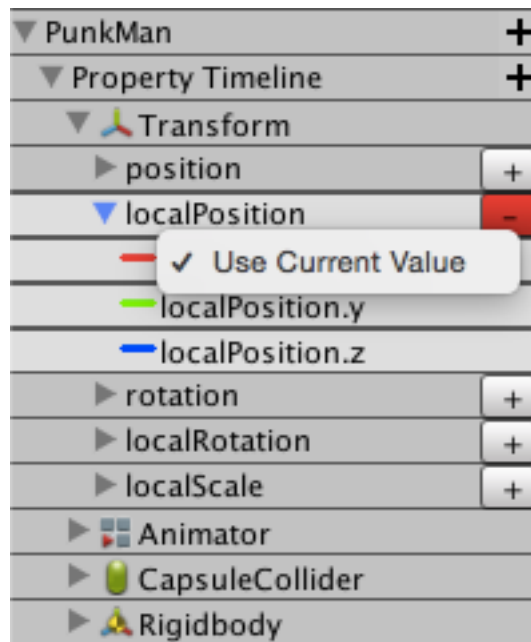
If you select multiple objects in an timeline and hold down the key combination Alt + Ctrl then drag your selection, you will duplicate that selection. The duplication works across many different types of objects, not just keyframes.

Extracting the starting value at runtime

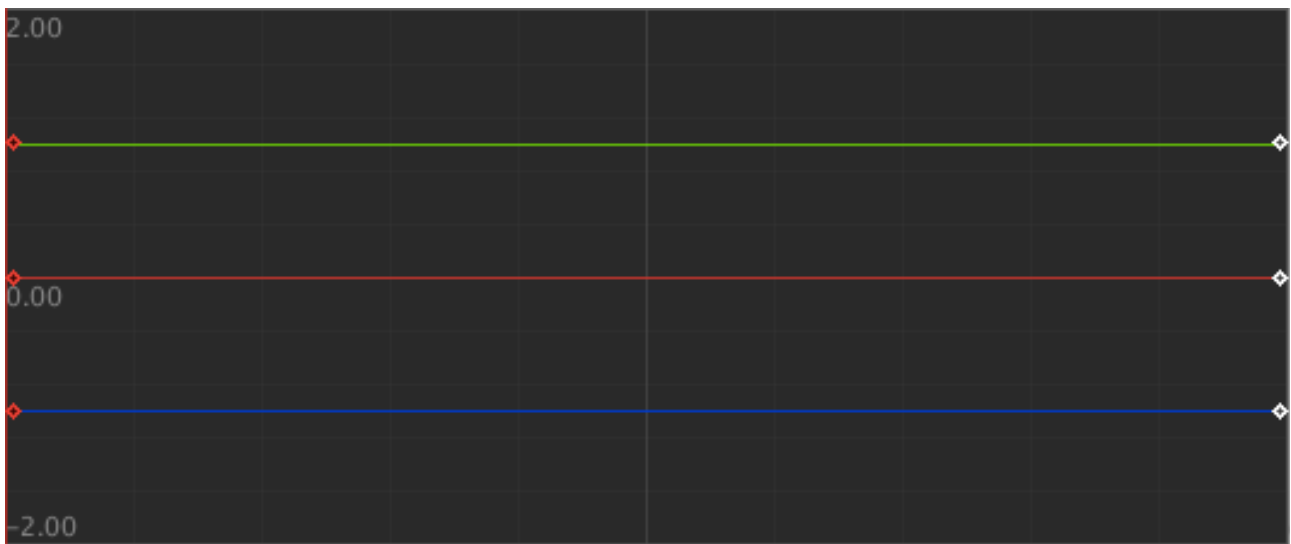
When making curves with the property timeline, it is not always possible ahead of time, to know the starting value of a curve. This could happen for example, when you are modifying a value at runtime, then expect a sequence to start and animate from the value.

This is a problem with a solution!

Find the property you'd like to set to take the runtime value in the uSequencer UI, for instance, you might pick the `localPosition`, right click on the property and select Use Current Value.



When you enable this option, you should notice that your initial keyframes will turn red, this is to indicate to you that the value will be taken at runtime.



Animation Timeline (Beta)

What are Animation Timelines

Animation Timelines are dedicated to bringing you live editing / viewing of Unity's Mecanim animation system. Note, these timelines are still in beta and whilst fully functional might change in the near future.

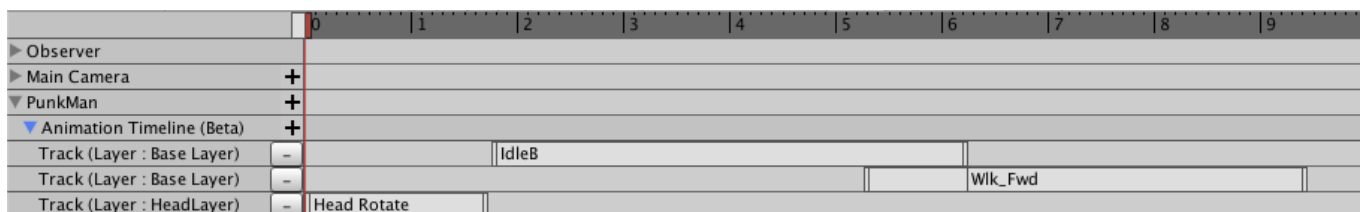
With Animation Timelines, you can play Mecanim Animation states, blend Mecanim Animation States, Play with individual layers all at edit time!

For the first time ever you can use a sequencing tool to view Mecanim events in line with the rest of your sequence.

Animation Timelines also fully support Root Motion. you're characters will also move and rotate in edit mode.

What is a track

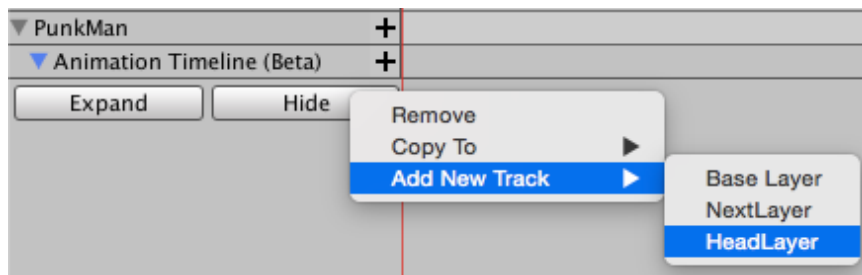
A track is analogous to Mecanim Layers. Whilst you can only have one layer, you can have multiple tracks using one layer, for example, the following complex Sequence.



You'll see that this Animation Timeline uses three tracks, Base Layer twice and Head Layer. You can have multiple tracks utilising one layer, simply for ease of reading. In this example, you can see that this Timeline will play the IdleB Animation state and the crossfade into Wlk_Fwd for approximately one second. All this will happen on the base Layer. We will also play A Head Rotate animation on the Head Layer.


Adding new tracks

When you have an animation Timeline, adding new tracks is simple. If you right click on your Animation Timeline, you will have an option Add New Track / (your layer). Simply click on the layer you would like to add.



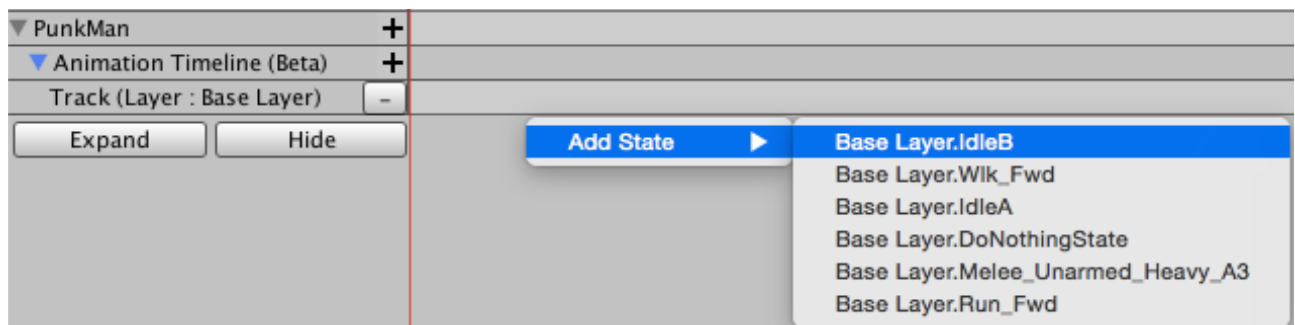
This will add your track.

Removing tracks

If you decide you'd no longer like an Animation Track and any of it's associated events, you can remove it simply by clicking the  button next to the track you'd like to remove

Adding new animation states

If you'll remember each track represents one Animation Layer, if you'd like to add an animation from that layer onto your track, you can do so simply by right clicking on that track and selecting Add State / (your animation state).



This will automatically add the state to that track for you.

There is no limit to how many states you can add to one track. You can however spread your states out amongst multiple tracks, so as to avoid clutter in your Animation Timeline.

Removing animation states

You can remove any unwanted animation states by simply selecting them in the uSequencer UI and pressing the backspace or delete key on your keyboard.

Duplicating Animation States

uSequencer allows the user to duplicate anything in any timeline with a simple key combination. Animation States in an Animation Timeline are no different.

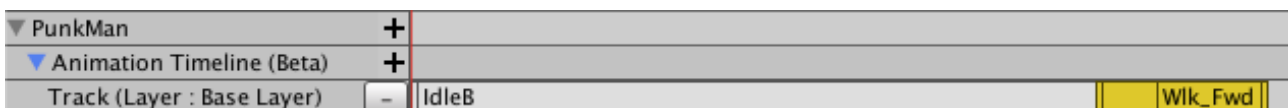
If you select multiple objects in an timeline and hold down the key combination Alt + Ctrl then drag your selection, you will duplicate that selection. The duplication works across many different types of objects, not just Animation States.

Blending between multiple animation states

uSequencer will try to automate the blending of animation states for you.

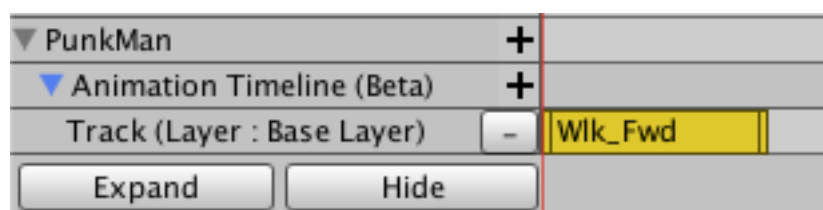
When you add two states to the same layer and have one overlap the end of another, uSequencer will automatically set up the blend for you.

The following example shows a blend between the IdleB Animation State and the Wlk_Fwd Animation State, when you drag the Animation State Wlk_Fwd, you'll see that the blend time is visualised and the name of the new State will always be rendered after the blend.

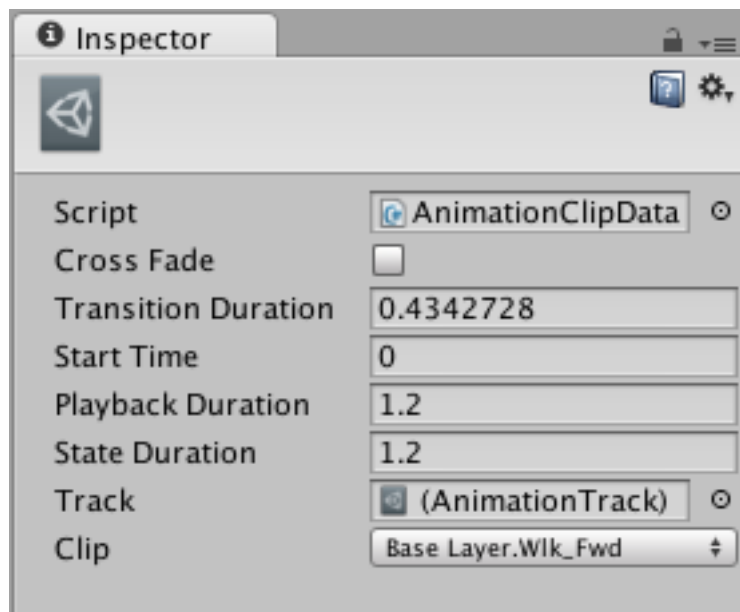


loop an animation state

Suppose you have a Walk Animation called Wlk_Fwd, the state will last approximately 1.2 seconds, but you've decided you want it to animate for 5 seconds, you can achieve this quite simply by selecting the Animation you'd like to loop in the uSequencer UI.



This will show you the Animation State Options in Unity's Inspector.




You can modify the Playback duration to the desired value, in this case, you'd enter 5. you're state will now loop for 5 seconds and this should be reflected in uSequencers UI.

Sequences As Prefabs

Creating your prefab

As with most things in Unity, uSequencer is a good citizen and sequences can be used as prefabs! This is great when you need to implement functionality multiple times and you want to save time! A great example of this, is using uSequencer to create a nice, reusable door opening animation.

Creating prefabs with uSequencer is a simple affair.


- 1) Ensure you've made your wonderful sequence!
- 2) Select your sequence in uSequencer
- 3) Press the Create Prefab button  .
- 4) Select a folder to save your prefab.

Your prefab should now be created and ready to go.

Updating your prefab

If you have already have a prefab in your scene and you've made some local changes, you will likely want to apply those changes to all instances of your prefab.

This is simple to do within uSequencer.

- 1) Select your sequence in uSequencer
- 2) Press the Update Prefab button  .
- 3) Select a folder to save your prefab.

Your local changes will now have been applied to your prefab and all changes should have been migrated to any instances you have in any scenes.

Prefab structure

uSequencers prefabs, along with any Unity prefabs exist in Unity's project view. uSequencer will automatically create a folder for you with the same name as your sequence. Under this folder will exist everything that your sequence needs to execute. The folder structure will be outlined below, do you can better understand what each file does.

- + Sequence Name
 - + Sequence Prefab (This is your sequence' game object)
 - + ScriptableObjects (Folder containing all sequence data)
 - + (All the scriptable objects that this sequence needs to execute).

It's quite important that you don't manually modify this folder, uSequencer will automatically update the files for you, if you use the uSequencer UI to Create and update your prefabs.

Reusing your prefab in scenes.

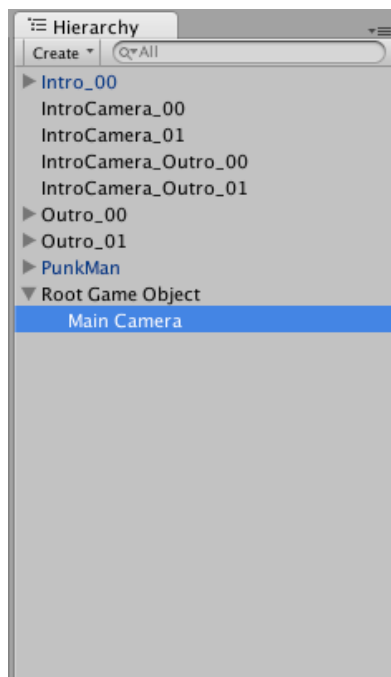
If you've gone to the effort of creating a sequence, you're going to want to add this sequence to various scenes, to use a prefab'd sequence in a scene, simply drag the sequence game object into your scene of choice.

Prefab Affected Object Late Binding

Unity has a one way relationship, items in the scene can know about items in your project pane, items in your project cannot know about anything in your scene however.

This is a problem for uSequencer, since your prefab'd sequence will need to operate on objects living in your scene. When you Update or Create a prefab, uSequencer will serialise out all transform paths of all Affected Objects, then when a prefab is instantiated the Affected Objects will be found using this string lookup.

For example, suppose your sequence is animating an object called Main Camera which is a child of Root Game Object, as demonstrated below.



The serialised path will contain the string "Root Game Object/Main Camera", your sequence will then be saved and when you create a new instance "Root Game Object/Main Camera" will be found.

You don't need to do anything about this, uSequencer will take care of it automatically for you, it is however, very important that you understand how this works, otherwise, it will seem like magic.

Instantiating your prefab using a sequence

Prefab'd sequences can be instantiated using an event timeline. If you add an event timeline and a event Spawn / Spawn Prefab, you can customise this event in Unity's inspector.

Instantiating your prefab through code

If you would like to instantiate a sequence through code, this can be done quite simply, using Unity's built in instantiation code.

As demonstrated below.

```
using UnityEngine;
using WellFired;

public class SampleSequenceControl
{
    public USSequencer sequencePrefab;

    public void SpawnSequenceAndStartPlaying()
    {
        var instantiatedGameObject =
GameObject.Instantiate(sequencePrefab.gameObject) as GameObject;
        var instantiatedSequence =
instantiatedGameObject.GetComponent<USSequencer>();

        instantiatedSequence.Play();
    }
}
```

Unity UI Integration

Overview

With the introduction of Unity 4.6, we now have some wonderful UI components to utilise. Wouldn't it be great if, for instance, we could use a UI button to Play, Pause or Stop a sequence? Well, now we can, with the new UI Integration

uSequencers Unity UI Integration comes in the form of some useful Components you can attach to your new UI buttons. Let's outline each component below.

Each component has two fields in the inspector. The first will be the sequence you wish to operate on and the second is a toggle that tells the script to manage the state of the button, depending on what the sequence is doing. We recommended you keep this on.

PlaySequence.cs

The Play sequence component does exactly what it says on the tin, it will start a sequence playing back. To use this component, you simply need to add it to your newly create Unity UI Button and assign the sequence you'd like to play in the inspector.

If the script is maintaining the state of the button, the button will only be enabled when the sequence is in a valid state and can be played.

StopSequence.cs

The Stop sequence component will stop any currently playing sequence. To use this component, you simply need to add it to your newly create Unity UI Button and assign the sequence you'd like to stop in the inspector.

If the script is maintaining the state of the button, the button will only be enabled when the sequence is in a valid state and can be stopped.

PauseSequence.cs

The Pause sequence component will pause any currently playing sequence. To use this component, you simply need to add it to your newly create Unity UI Button and assign the sequence you'd like to pause in the inspector.

If the script is maintaining the state of the button, the button will only be enabled when the sequence is in a valid state and can be paused.

SkipSequence.cs

The Skip sequence component will skip any currently playing sequence. To use this component, you simply need to add it to your newly create Unity UI Button and assign the sequence you'd like to skip in the inspector.

If the script is maintaining the state of the button, the button will only be enabled when the sequence is in a valid state and can be skipped.

SequencePreviewer.cs

This component can be attached to any Unity Slider and will allow the user to scrub through any currently playing sequence. To use this component you simply need to attach it to any existing Unity UI Slider.