uSequencer – User Guide

Table of Contents

Overview

For more up to date information and tutorials please check the uSequencer homepage : https://sites.google.com/site/usequencer/

The uSequencer is cutscene authoring tool created for Unity, by a lover of Unity. Hopefully this tool will allow any user to quickly and efficiently create cutscenes for their games, apps, or products.

The main focus of uSequencer is to be easy to use, and to fit in seamlessly with unity's interface.

Key Features

- Full timeline based event sequences
- Compatible with all platforms.
- Compatible with Unity Indie and Unity Pro.
- Intuitive and easy to use User Interface (Simple context Menu and Drag Drop interaction).
- Out of the box events:
    - Audio Control
    - Physics Events (Impulse, Force)
    - Animation Control
    - Message Sending
    - Print To Screen
    - Instantiation
    - Many Many More
- Out of the box support for Playmaker and uScript!
- Prefab Support.
- Create and add your own events with incredible ease.
- Completely customizable in window event rendering.
- The sequencer will not add any sequencer scripts to your objects, it's completely contained.
- Full Undo / Redo support.

You can find more out about USequencer here :
http://www.usequencer.com

Here:
http://forum.unity3d.com/threads/140129-USequencer-Unity-Event-Sequencer-(Cutscenes)

or of course, you can always fire us an email :
support@wellfired.com

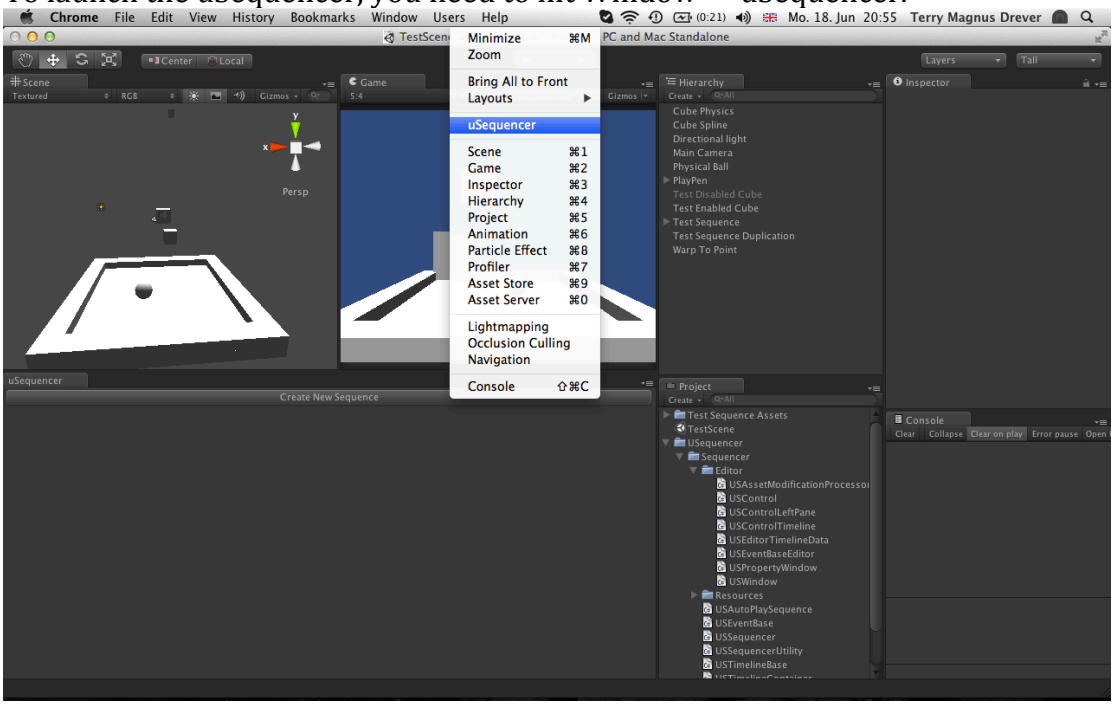
Free Vs Paid Version Differences

If you are using the free version of uSequencer, you have all the features of the paid version. The only difference is that your sequences will have a watermark in the top left corner and you don't have access to the source code. Other than this, you have a fully featured version of the uSequencer.

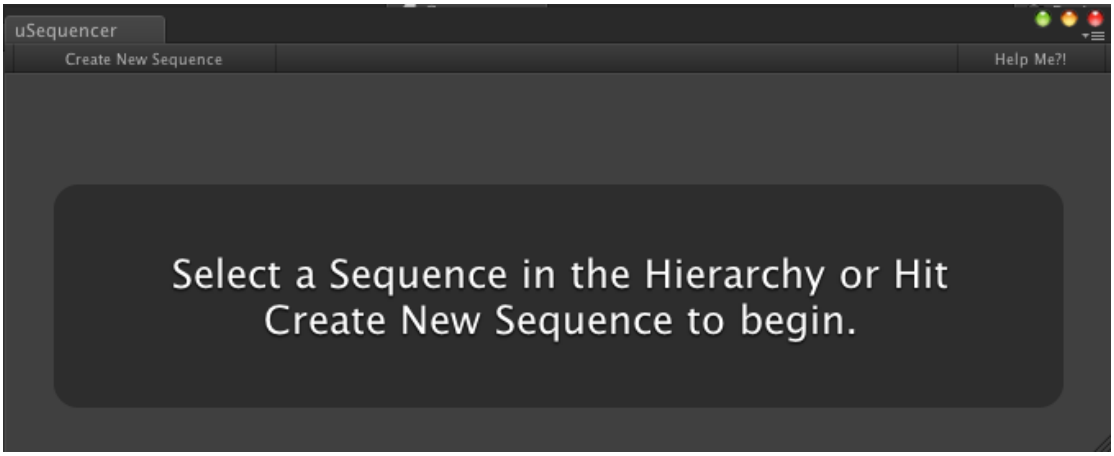
Installation and launching

When you download this package from the asset store, you shouldn't have to do anything to set up uSequencer, it should just work, out of the box.

To launch the uSequencer, you need to hit Window -> uSequencer.



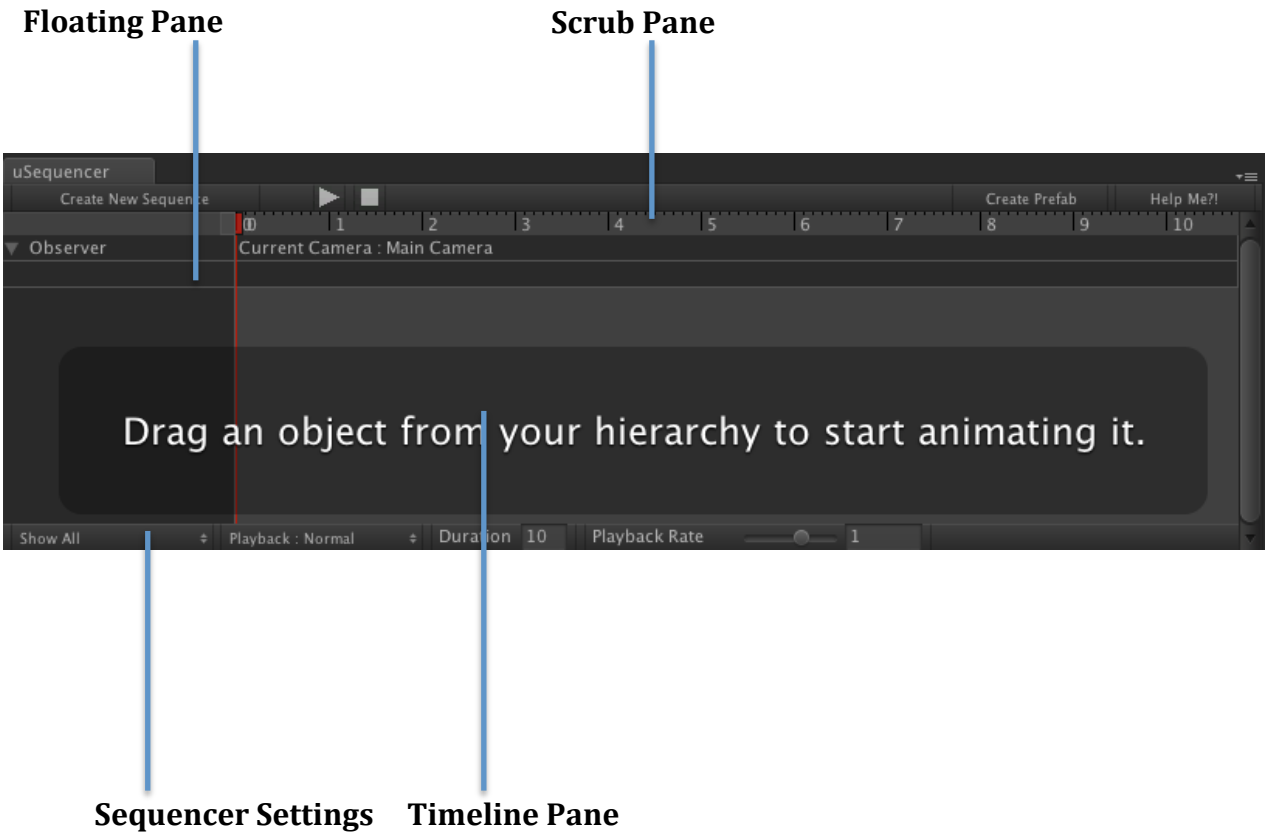
You should be presented with a window that can be docked anywhere and looks like :



Simply click Create New Sequence to start creating your cutscenes!

Main Screen Overview

After creating your first sequence, you might be a little overwhelmed by the options, let me break them down for you below.



Floating Pane : This floating pane will house the objects that your sequence will affect. Simple drag an object here to start sequencing.

Timeline Pane : You will probably do the majority of your work in the Timeline Pane, adding events, manipulating events.

Scrub Pane : Use this pane to scrub back and forth through a cutscene, you can also zoom in and out here.

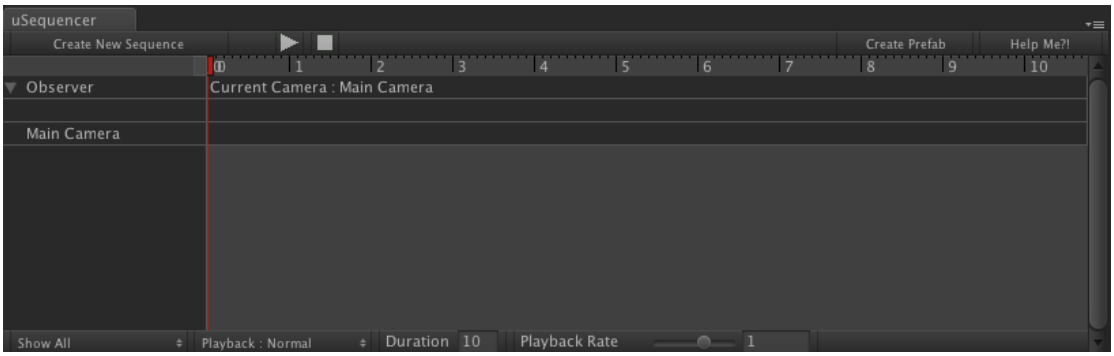
Control Overview

Most of the interaction with the uSequencer is done with the context menu (right click), or drag drop.

Floating Pane

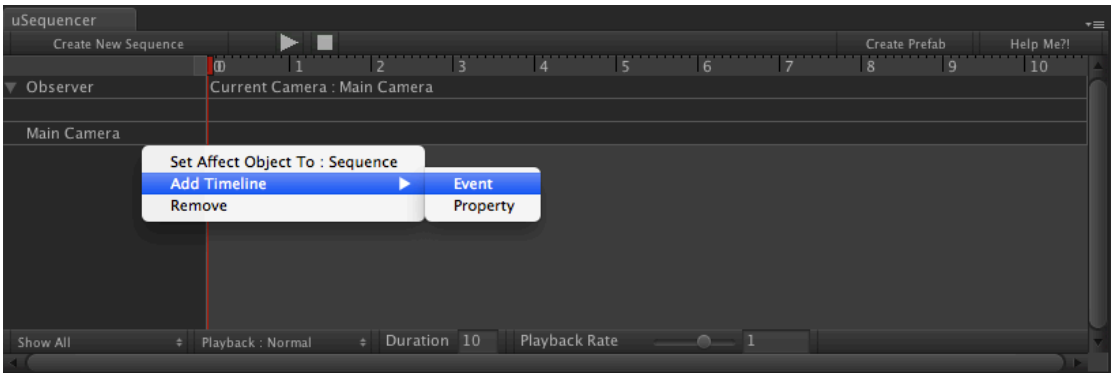
You can drag objects from your scene directly into the floating pane, if you do this, you will see an object with the same name in your sequencer, from here you.

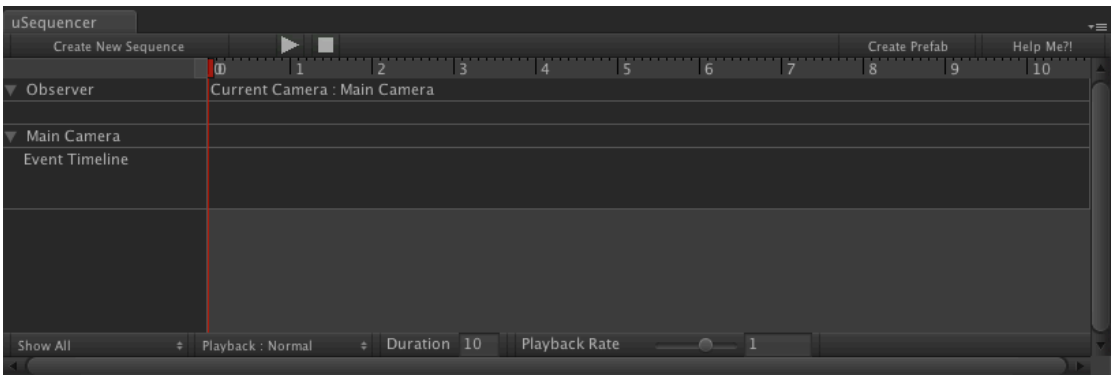
You can use your scroll wheel to navigate up and down your timeline list.



You can now context click on your object to reveal a menu allowing you to add a timeline to this object (Note, you can add as many timelines as you want, or need.)

You can rename these timelines, by selecting them in the sequencer, this will change the currently selected object in the inspector window. Simply rename the timeline from here.

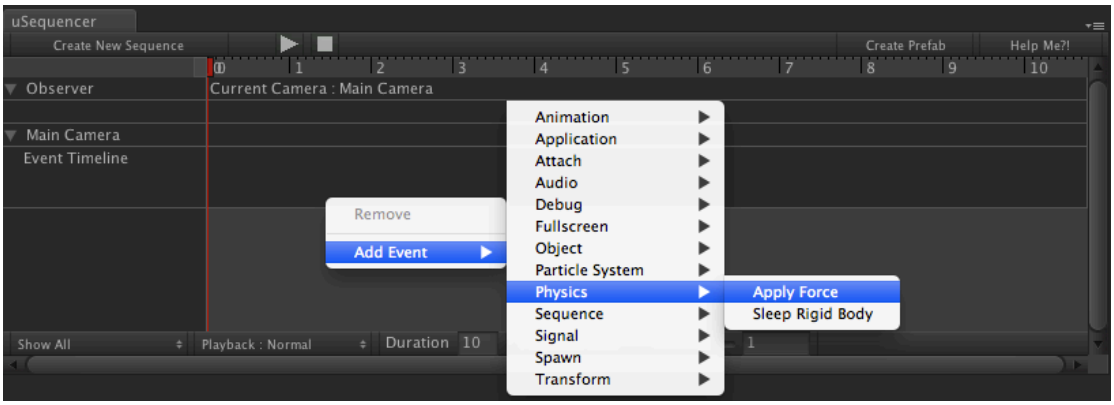




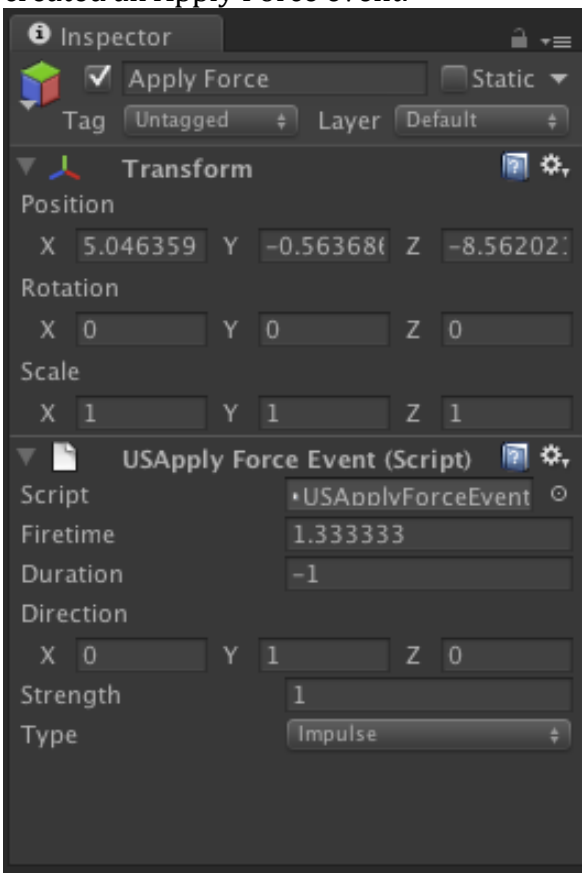
Timeline Pane

You will spend most of your time in the timeline pane. The events x Position in the timeline pane corresponds to the time at which this event will trigger.

Most actions that can be done in the timeline pane, will be done with the context menu (Right-Click).



Once you've added a new event, if you click on this event in the timeline, you will see the inspector changes to represent your event. In the example above, we created an Apply Force event.



You see here, we can change important properties for the Apply Force event, including the strength and the Type.

Click dragging an event in the timeline, will change the time at which that event get's triggered.

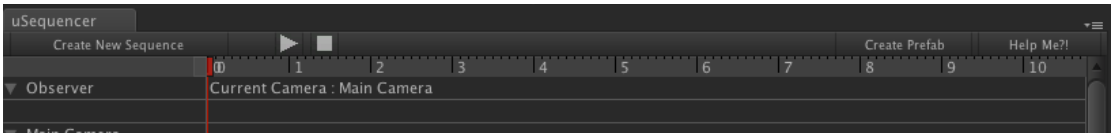
Hierarchy manipulation

Another great feature of the uSequencer is that you can manipulate the timelines through the hierarchy.

If for instance, you've set up an amazing timeline affecting one object, and you want to duplicate that timeline and have it affect another object, you simply need to copy and paste within the hierarchy. Furthermore, you can drag, timelines, timeline containers and events between different sequences, giving you complete control over what your cutscene affects, and how!

Observer Timeline

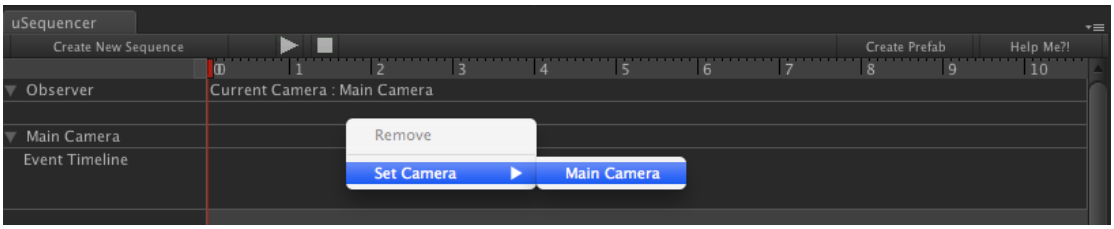
The observer timeline is automatically added to every cutscene you create with the uSequencer. It is visible in the timeline pane.



You can interact with the Observer timeline through the context menu (Right Click Menu).

Through this context menu, you can delete keyframes, or add new keyframes. The context menu will list any camera in your scene.

It will also sate the currently active camera at the top.



Data Layout

All Sequencer properties and objects will exist only in the hierarchy pane. Every object that relates to the sequence will be a child of the parent object (The Sequence).

Note: since all uSeqeuncer elements are objects in the hierarchy pane, you can change their properties in the inspector as well as in the uSequencer window.

An Example Data layout is below.

Sequence (USSequencer)
- Timelines for Object 1 (USTimelineContainer)
  - Event Timeline (USEventTimeline)
    - Event 1 (USEventBase)
- Timelines for Object 2 (USTimelineContainer)
  - Event Timeline (USEventTimeline)
  - Property Timeline (USPropertyTimeline)

As you can see, a sequence contains many Timeline containers, each of these hold the timelines for a specific GameObject in your scene.

Each Timeline Container holds the timelines for that GameObject in your scene.

Since all data is laid out in the hierarchy, it is possible for the user to drag drop between TimelineContainers, and Timelines. The user can drag whole timelines, timeline containers, or individual events.

Timelines Overview

The uSequencer consists of timelines, or rather timelines make up the bulk of the uSequencer and the sequence that is produced. An overview of all the timelines is below. A timeline is basically a sequence of time, running from 0 seconds to the duration of your cutscene.

Event Timeline

The event timeline is the basic timeline, it contains events and is the most basic of timelines. If you create custom events, or use events that the uSequencer provides, they will show up here.

You can interact with the event timeline through the context menu (Right Click Menu). Adding of events is possible through this context menu.

Property Timeline

A property timeline is a timeline that lets you manipulate an objects properties, you can use this to create movement curves, rotation curves, or to keyframe any other property on an object. (For example the range on a light, or exposed properties in custom scripts).

To use a property timeline, you must find the property you wish to animate, in the property timeline and push the little + button next to the property. You can then add keyframes by scrubbing the sequence to the time you'd like to add a keyframe and manually manipulating this property in the inspector.

You can edit curves by dragging keyframes, or by manually manipulating keyframe tangents. There are also a lot of options available to the developer through the context (Right Click Menu).

Observer Timeline

Observer timelines are automatically added to every created cutscene.

You can find the Observer Timeline at the top of the uSequencer Pane. This timeline allows you to directly and global control every camera in your scene. The observer Timeline will automatically turn off all none specified cameras, allowing you to implement camera cuts and detect which camera is active at a global level.

Example Scenes

Each copy of uSequencer comes with a bunch of example scenes, these serve as common tutorials for you to investigate. A description of each of these follows

01 Basic Examples

These basic examples showcase some of the most basic functionality in uSequencer.

BasicExample01-PropertiesAndEvents

This example scene shows you how Property and Event timelines can be used to achieve basic camera movement, and Particle System triggering.

BasicExample02-ObserverTrack

This example scene shows you how an Observer Track can be used to toggle between active cameras.

BasicExample03-Gameplay To Cutscene

This example scene shows you how you can seamlessly transition from a Gameplay Camera to a cutscene camera.

02 Basic Prefab Examples

These slightly more advanced samples show how you might use Prefabs in multiple scenes, or how you might, retarget uSeqeuncer Cutscenes.

01 Original Cutscene

This sample scene, is simply a basic cutscene, we used this cutscene to base our other scenes.

02 Instanced Modified Prefab 01

This sample scene uses the cutscene created in 01 Original Cutscene, however it has been retargeted to a sphere.

03 Instanced Modified Prefab 02

This sample scene uses the cutscene created in 01 Original Cutscene, however it has been retargeted to a cylinder.

03 Basic Additive Scenes

These sample scenes show how you might use an Additive Scene to share a cutscene between two scenes.

It is important that, before you run these scenes, you should add them to the Build Settings (File -> Build Settings).

01 Original Cutscene

This cutscene is the original scene, it will be additively loaded into another scene, and then played.

02 Additively Loading Cutscene

This cutscene will additively load the cutscene 01 Original Cutscene.

04 Intermediate Examples

These sample scenes use uSequencers basic functionality and add extra niceties on top.

01 Simple Camera Look

A simple Cutscene that moves a camera along a spline and allows you to mouse look.

Sharing Between Scenes

Sharing uSequencer cutscenes between multiple scenes, is quite easy. If you decide you'd like to share a sequence between multiple scenes, then you will have two choices Going forward.

1) Use Prefabs.
2) Load additional scenes additively.

Using Prefabs

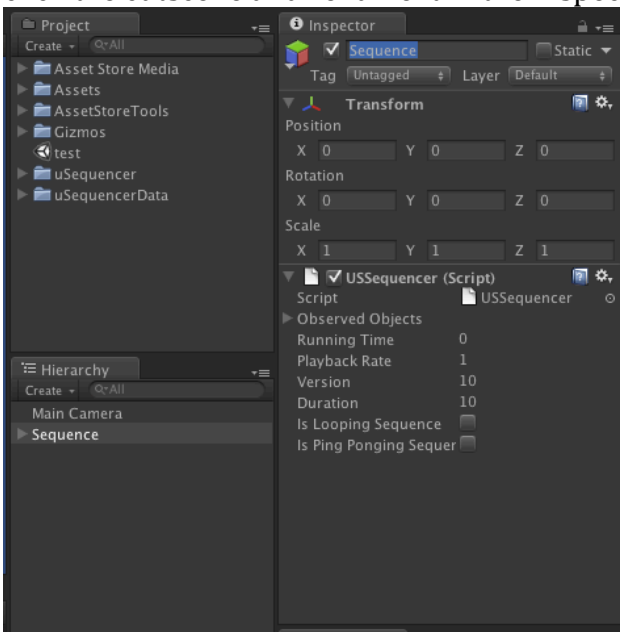
uSequencer come with build in prefab support. If you decide you'd like to share a sequence between multiple scenes, this can be achieved easily, however, it is strongly recommended that you have a full understanding of how prefabs work before you continue.

When you have finished a cutscene and you'd like to share it between multiple scenes, it is important that you follow these steps.

1) Name your cutscene appropriately.
2) Store the prefab somewhere sensible.
3) Add your prefab to any scenes that require it.
4) Re make the cutscene -> Affected Object association.

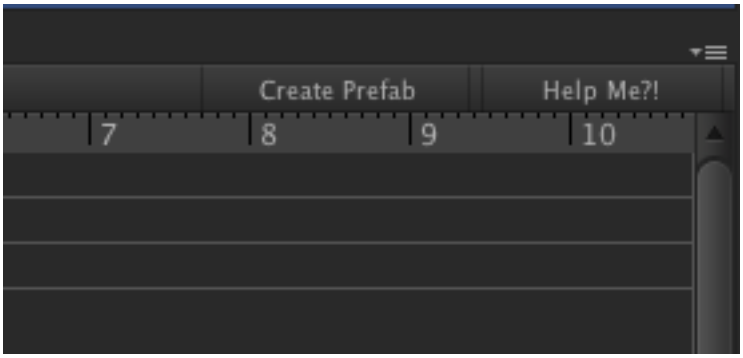
Naming your cutscene.

You can name your custcene, the same way you'd name anything in Unity, simply click the cutscene and rename it in the inspector.



Store the Prefab somewhere sensible

You create your prefab with the Create Prefab button, at the top right of the sequence window.



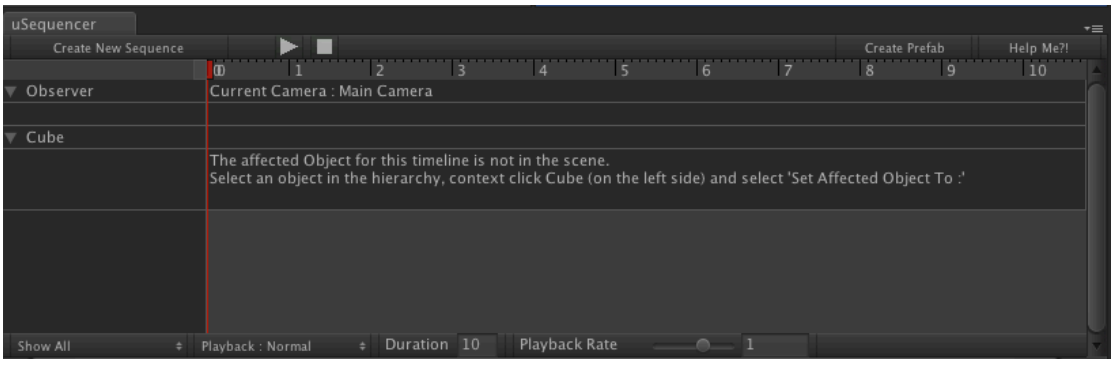
When you press Create Prefab, a popup will appear, asking where you'd like to save the prefab. The prefab will have a default location (inside the uSequencer Data Directory), but you can save this anywhere you like.

Add your prefab to any scenes that require it

You do this just as you do any other prefab in Unity, simply drag from the project view to the hierarchy.

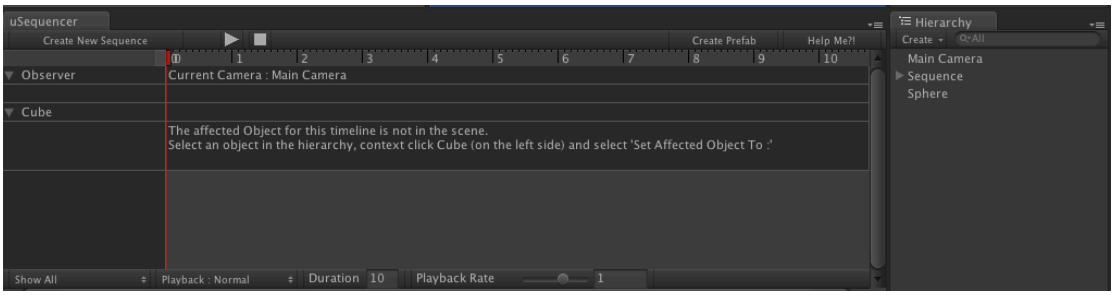
Re make the cutscene -> Affected Object association

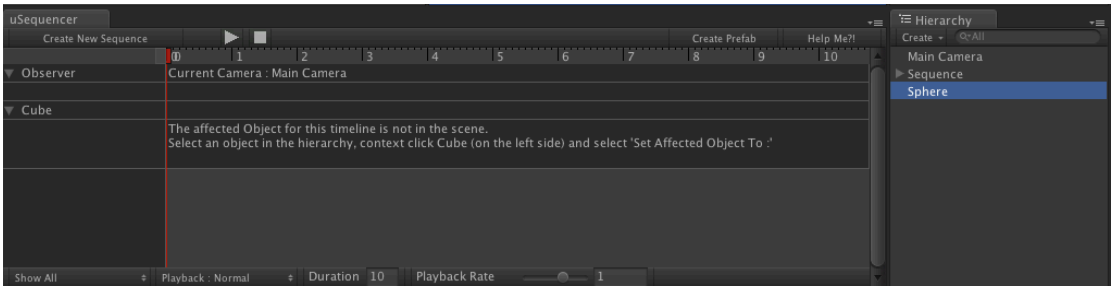
You'll see that your imported cutscenes look a bit funny in the uSequencer window. Something like this :

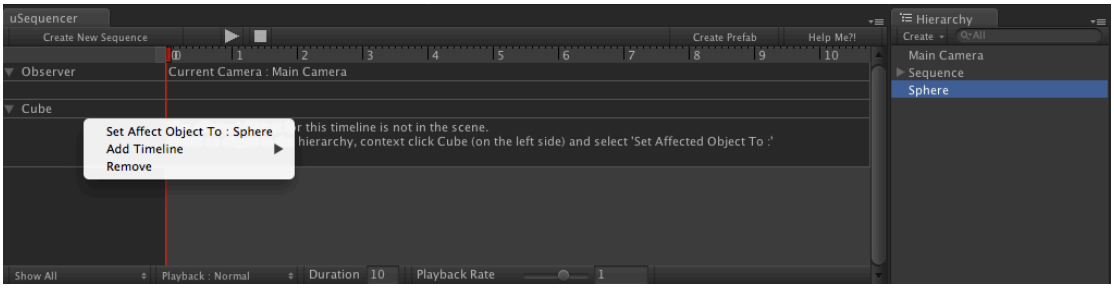


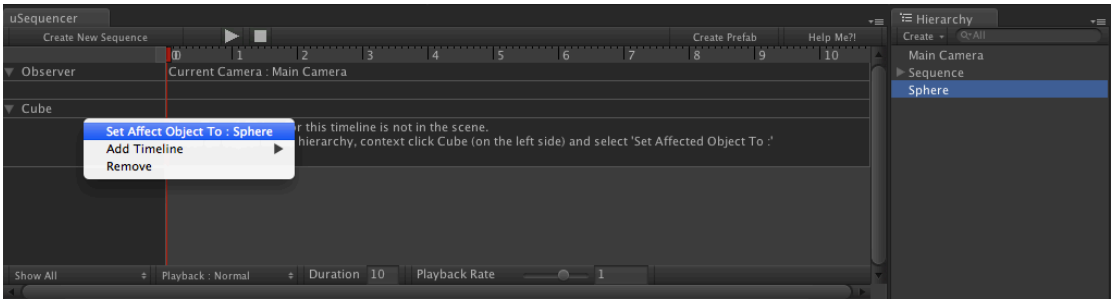
This is because the cutscene relies on data within the scene to run. Each scene is different and has different data. It is a necessary step that you remake the connections. You can do this simply by selecting the new object you'd like the Timelines for Cube to affect in the hierarchy. You will then right click on the Timeline for cube and select "Set Affected Object To : "

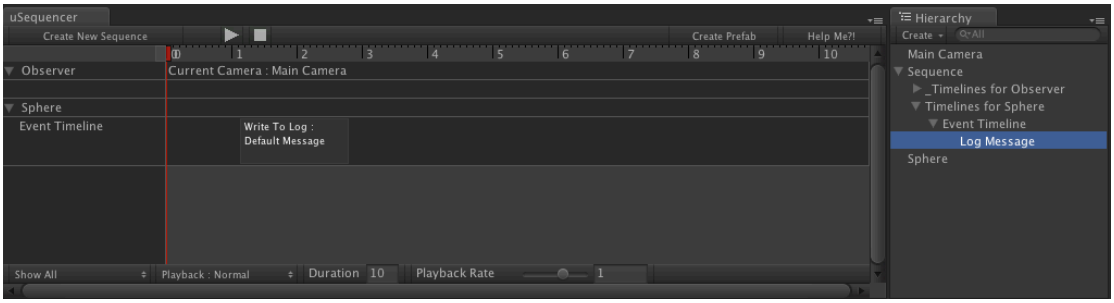
This sounds complex, but is described in the following illustrations.











You'll see that the timeline that used to affect a Cube Object, now affects a Sphere object. This allows you to perform Prefab Specialisation, on a per instance basis.

Things to watch out for when you are working with prefabs

1) It is important to either save your Prefab to a unique location, or give it a unique name.
2) Each prefab sequence will also have a ScriptableObjects folder next to it. These two are associated with one another. The Sequence object can be used in your scenes, whereas the ScriptableObjects folder holds all data.
3) You will only have to repeat step 4) Re make the cutscene -> Affected Object association, if you manually press the Update Prefab Button. You will however, only have to press this update prefab button, if you've added new events, or timelines. Modifying existing data does not require this.

Load Additional Scenes Additively

This option will be a lot easier than using prefabs, and will be suitable for most cases. The approach is quite simple.

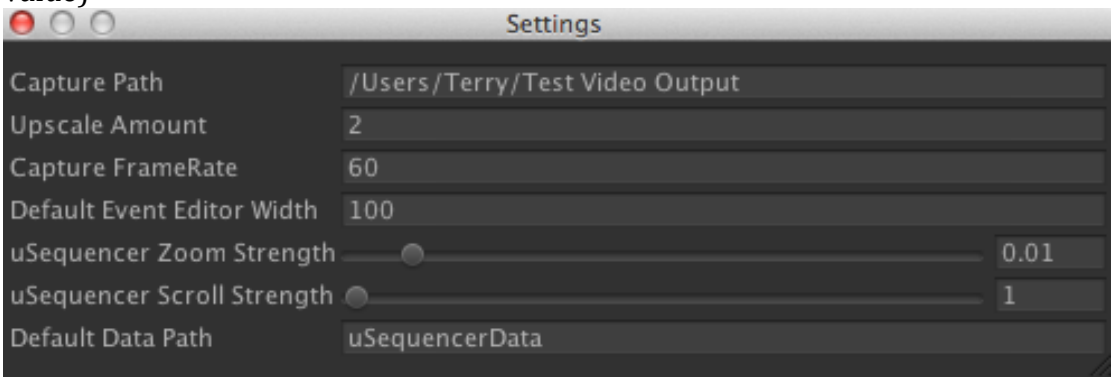
1) Create your amazing cutscene, with only cutscene data.
2) In your main scene, call the Application.LoadSceneAdditive function

This will load your cutscene into your current scene.

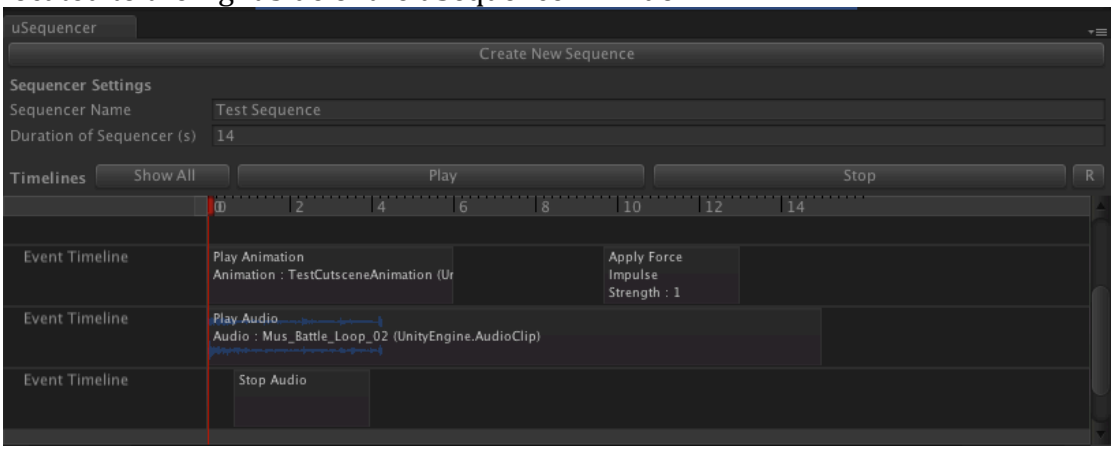
Recording a HD Video within uSequencer

uSequencer uses Unity's inbuild ScreenCapture functionality to capture High Resolution Videos. You can use uSequencer in built recording functionality to record at specified framerates, regardless of your machines capabilities, and super high resolution, this is ideal for sending to media, or showing of your game running as smooth as possible on your website.

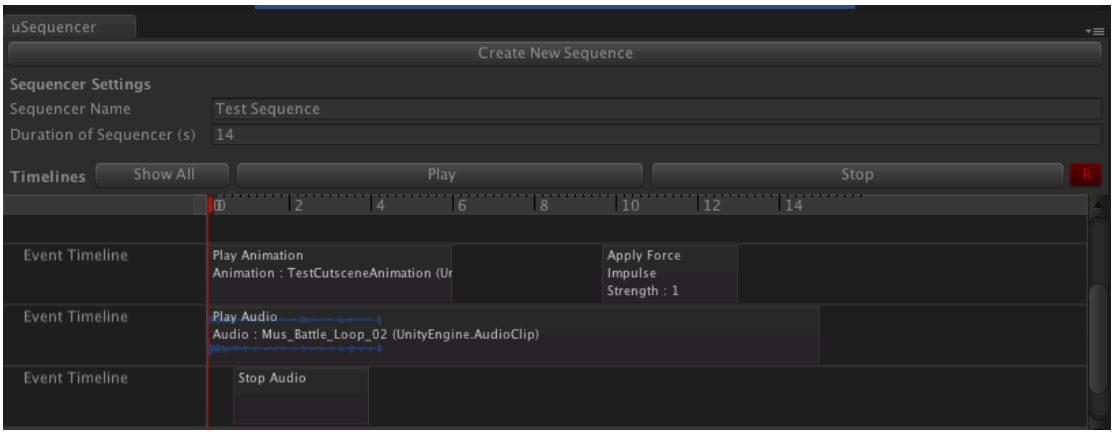
The settings menu for the Recording process can be found in the menu Edit -> uSequencer Preferences.
Capture Path – The output Path, all images will go here
Upscale Amount – How much to upscale the resolution by (960 x 540 with an upscale of 2 is 1920 x 1080)
Capture FrameRate – The framerate you'd like to capture (60 is a nice smooth value)



To record in editor, you must press play and arm your uSequencer sequence. The arm button only shows whilst you are playing in editor. It's the small R button located to the right side of the uSequencer Window.



The button will turn red when the uSequencer is armed



All you need to do now is hit play and wait until the sequence has finished playing through. Once it's done, you can press stop.

You should have a collection of many large image files in your Capture Path.

Now all you need to do, is use your favourite video package to combine all the images located in your Capture Path into a movie. (I use Timelapse Assembler, though you can also use Quicktime Pro).

Editor Extension

It's very simple to extend the uSequencer, and one of the main features.

Adding new Events

To add a new event, simply create a new script file and inherit from USEventBase. Suppose, for instance you wanted an event that would write a debug message to the Log. You can do this, with a very small class:

```
using UnityEngine;
using System.Collections;

[USequencerEvent("Debug/Log Message")]
public class USMessageEvent : USEventBase {
     public string message = "Default Message";

     public override void FireEvent()
     {
          Debug.Log(message);
     }

     public override void ProcessEvent(float deltaTime)
     {

     }
}
```

Your class must implement the FireEvent and ProcessEvent functions.

This class uses the C# attribute USequencerEvent() to define in which sub menu this item will be placed if you context click on the Timeline pane.

For instance, there will be a menu option Debug -> Log Message, in the context menu. Clicking this will add a USMessageEvent to your sequencer.

Optionally, you can implement the following functions within your class, for further control:

```
     public virtual void PauseEvent() { ; }
     public virtual void ResumeEvent() { ; }
     public virtual void StopEvent() { ; }

     public virtual void EndEvent() { ; }
     public virtual void UndoEvent() { ; } // Called when
you scrub back in time, to before this event fired
```

Although, you don't have too.

Customising the display of your new events

As well as implementing new events, you can customize how they look in the uSequencer. If you would like to do this, you need to place an editor script in the editor folder and inherit from USEventBaseEditor :

```
using UnityEditor;
using UnityEngine;
using System.Collections;

[CustomEditor(typeof(USMessageEvent))]
public class USMessageEventEditor : USEventBaseEditor
{
     new public Rect RenderEvent(Rect myArea, USEventBase
thisEvent)
     {
          USMessageEvent messageEvent = thisEvent as
USMessageEvent;

          if (!messageEvent)
               Debug.LogWarning("Trying to render an
event as a USMessageEvent, but it is a : " +
thisEvent.GetType().ToString());

          DrawDefaultBox(myArea, thisEvent);

          GUILayout.BeginArea(myArea);
               GUILayout.Label("Write To Log :",
defaultBackground);
               if (messageEvent)
                    GUILayout.Label(messageEvent.message,
defaultBackground);
          GUILayout.EndArea();

          return myArea;
     }
}
```

You can fill this function with whatever you like, you're class must have the c# attribute CustomEditor and you **must also return** the grab able area, so that the event can be drag dropped.

You can combine your event rendering class with Unity's own editor extension, for instance, the Apply Force event also implements OnSceneGUI, so you can change the force direction in the scene View!

Unity 4.0 Package

This Unity 4.0 package contains extra uSequencer events for Unity 4.0 users, you can access these simply by extracting the package. They will then be available for you to use through uSequencer.

Upgrade Problems?
You must remove your uSequencer directory (with Unity closed), before upgrading.

If you experience further problems, don't hesitate to get in touch.

Questions, comments, thoughts OR suggestions?

Feel free to email me at support@wellfired.com and if you like this software, you could always give me a nice review!