Combat Editor Instructions

Intro

CombatEditor is an plugin to manage the events based on the running animation clip such as particles, collider, motions, sfx . It have an editor window to easily add and remove the events, control the the start and the end time of these events, preview them in editor and run in playmode only with CombatController on the character. It just fetch the playable animations in the animator, doesn't control the translation of animations, so you can implement your own character controller.

Compared to traditional animation event workflow, it has adventages as below:

1. A tight window to control all of the event on animation, you can easily manage configs on animations by group, add, remove, swap, and change the start and end time of event are super easy. You don't have to use the default animation window which feels very hard to manage events.
2. Preview and visualize every type of event in scene mode, which makes it very convenient to control your particle and sfx in combat.
3. Join you project at very low cost. It doesn't control the translation the animation, it just fetch the playable data in the animator. So you can use your own character control with combat controller. You have to modify something, such as the trigger of sfx and physics control, but this is very easy.
4. You can write your own custom event very quickly by simply modify the template.
5. The Animation events on the editor is modular. You can reuse it on other characters with same animations.
6. Supports a strong range event, and this type will auto cut when animation are break and auto start if animation is enter after the event start time.
7. There are many details to accelerate the workflow. For example, loop preview animations, search to assign node on character, etc.

Table of Contents

Intro 1

Open CombatEditor Window 3

Set Up a Character 3

Preview Animation 4

Add AnimationEvents 4

Customlize you own Event 10

Add preview in scene mode to your event 14

Functions in preview template explained 15

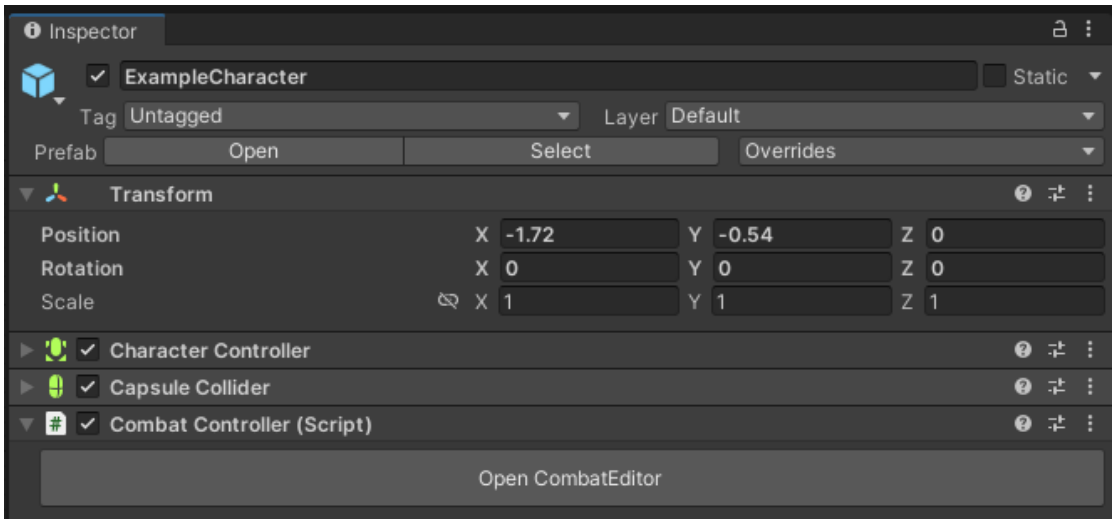
Complex preview: Instantiate an object with a handle 16

Open CombatEditor Window

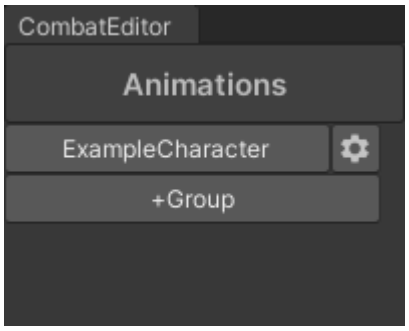
Please Open the editors in “Tools/CombatEditor” and “Tools/CombatInspector”, the CombatEditor is main editor window that manages the events on animation clips., and the CombatInspector shows the data of the events or config files in the CombatEditor.

Set Up a Character

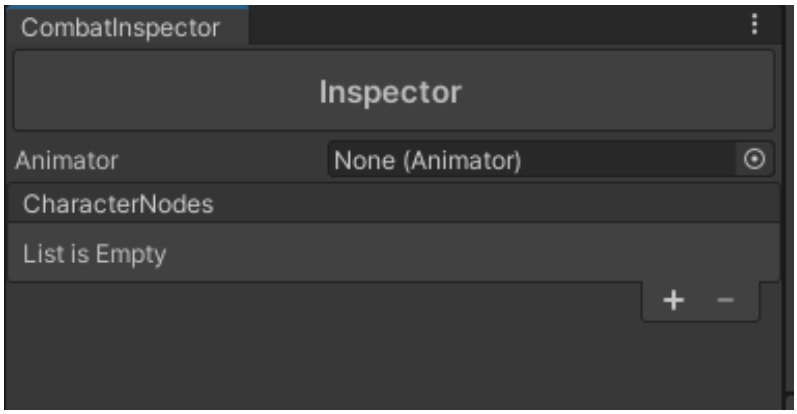
1.Create the Character and attach a “CombatController” script to it, place your animator gameObject as a child object of the character, and instantiate your character into a scene.(the editor doesn’t support prefabmode.)
2.Expand the CombatController component on your character.



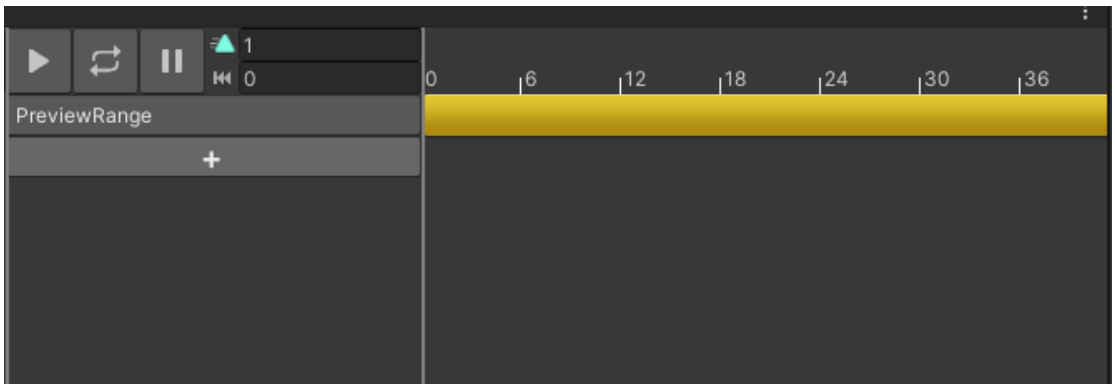
3.Click the “OpenCombatEditor” button to open CombatEditor. (You can also create combat editor by menu, which is in “Tools/CombatEditor” and “Tools/CombatInspector”.)
4.Now the combateditor looks like this.



5.Click the gear and assign the animator of your character in the CombatInspector window.



Preview Animation



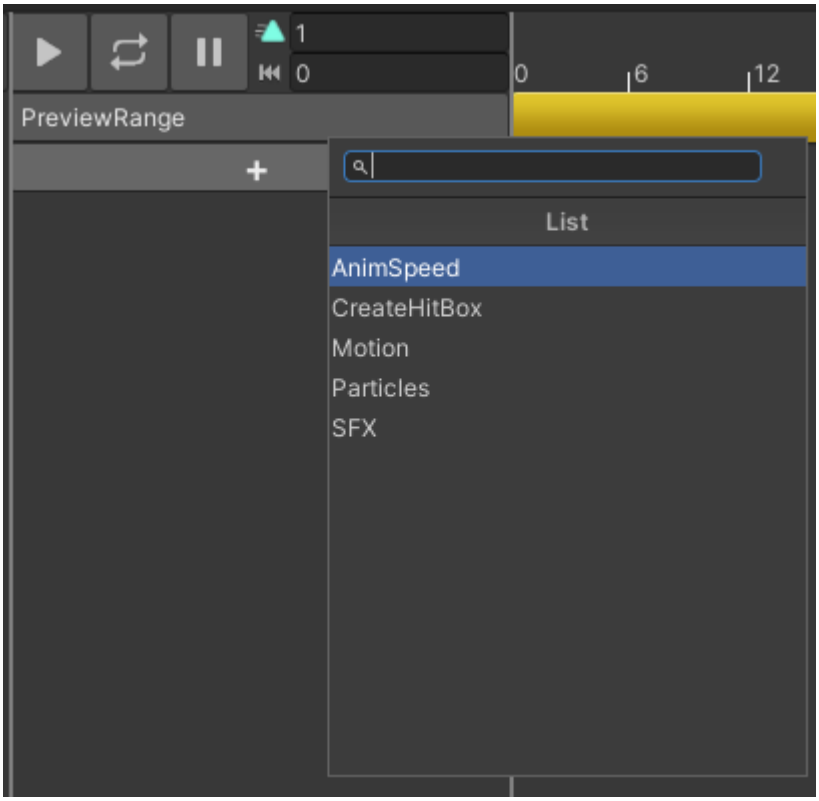
In this timeline window, you can press play button the preview the animation with effects on the character in scene, the loop button to loop play, and the pause button to stop. You can also drag on the timeline to preview.

The parameter at the right of the pause button is:

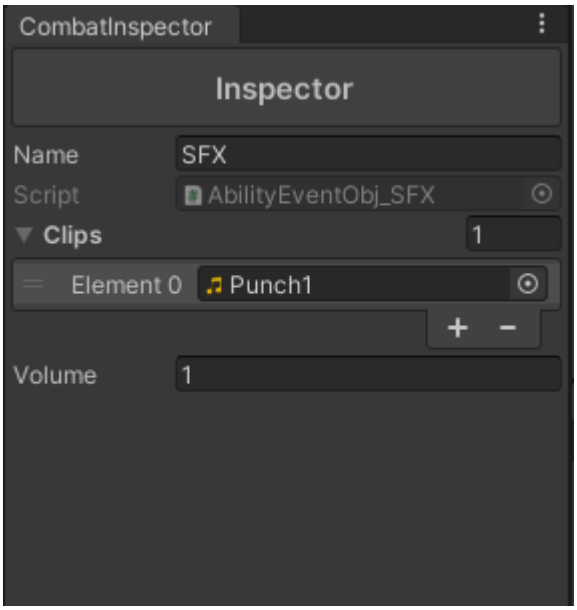
1. The preview animation speed multiplier.
2. The time between each loop.

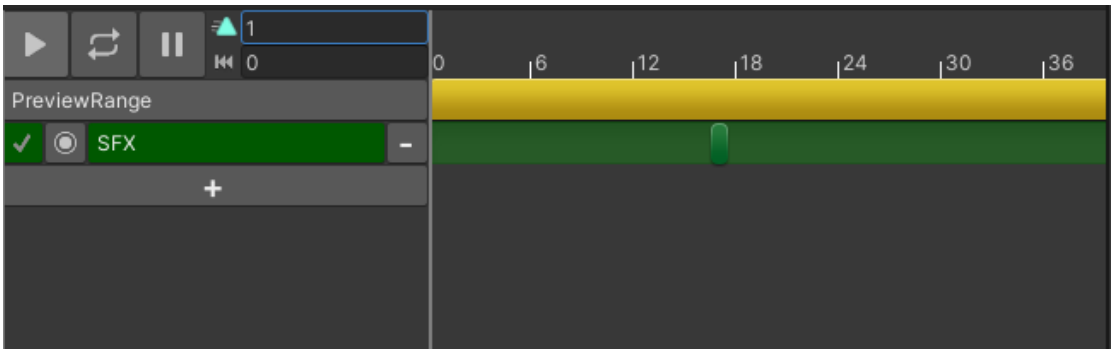
Add AnimationEvents

Now it is time to add some events. Press “+” button below the Preview Range button, and you can see a Dropdown which looks like this:



We can take a look at the SFX event at first, which is very simple. Add an SFX event by click the add button like in the pictures above and assign random SFX in the CombatInspector window like in the pictures below.





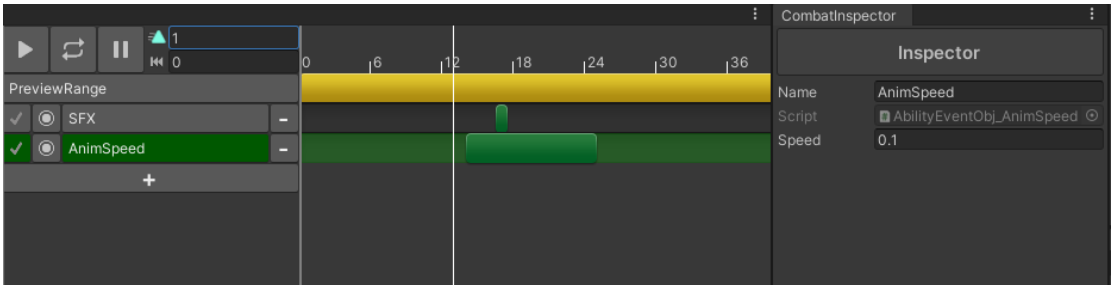
You can drag on the track of SFX, and the SFX spawn time will change.

Click play button, and the character will play the animation and when the animation is on the SFX trigger time, the editor will trigger the target SFX.

Next, we can take a look at animationSpeed event, which is the range event.

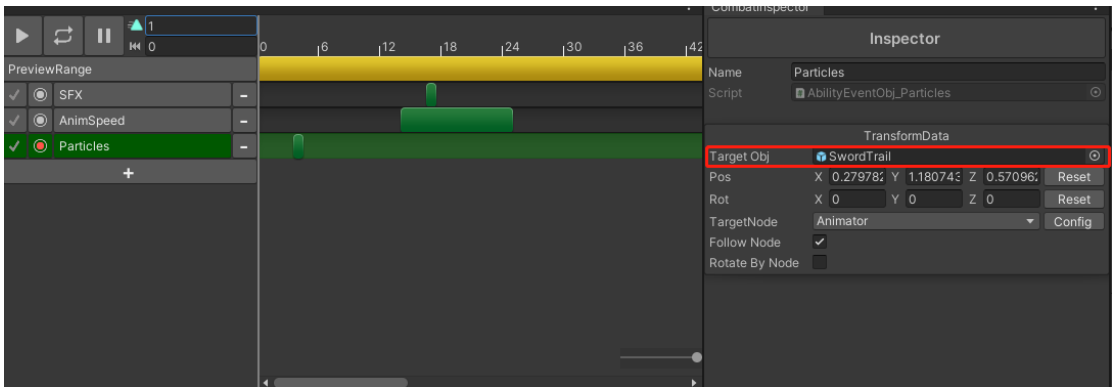
Add an AnimationSpeed event and assign it speed to 0.1f.

To control a range on the track, use left mouse button to control its start time, and right mouse button to its endtime.

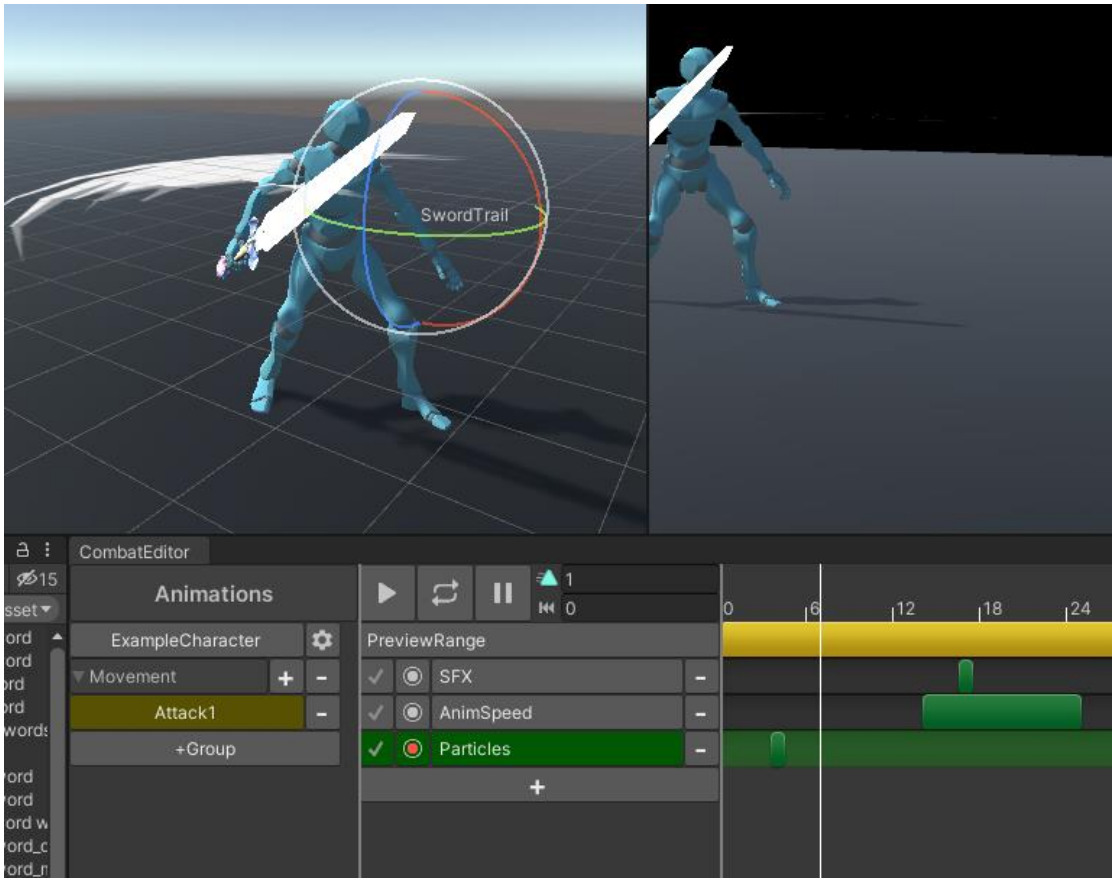


Click play, and the animation speed will multiply 0.1f when in this range.

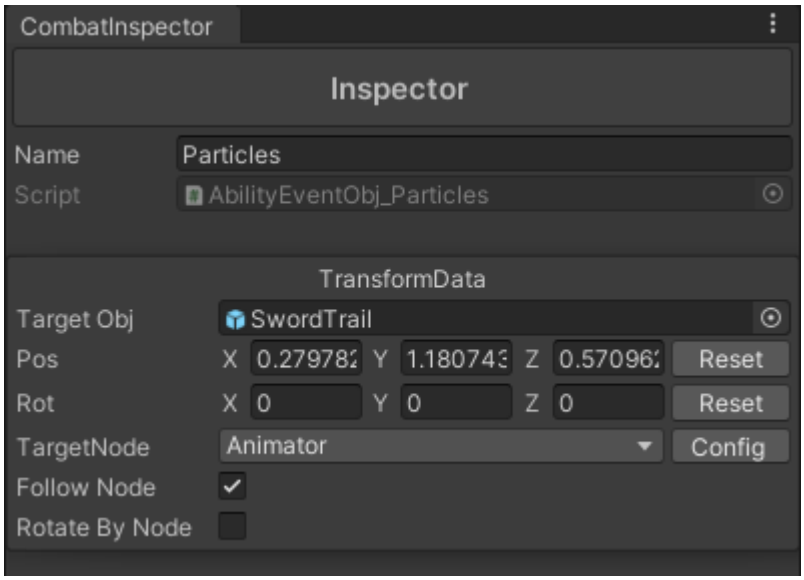
Now we can take a look at animationevent that is more complex. Add a particle event to the animation and assign your particle to the TargetObj values.



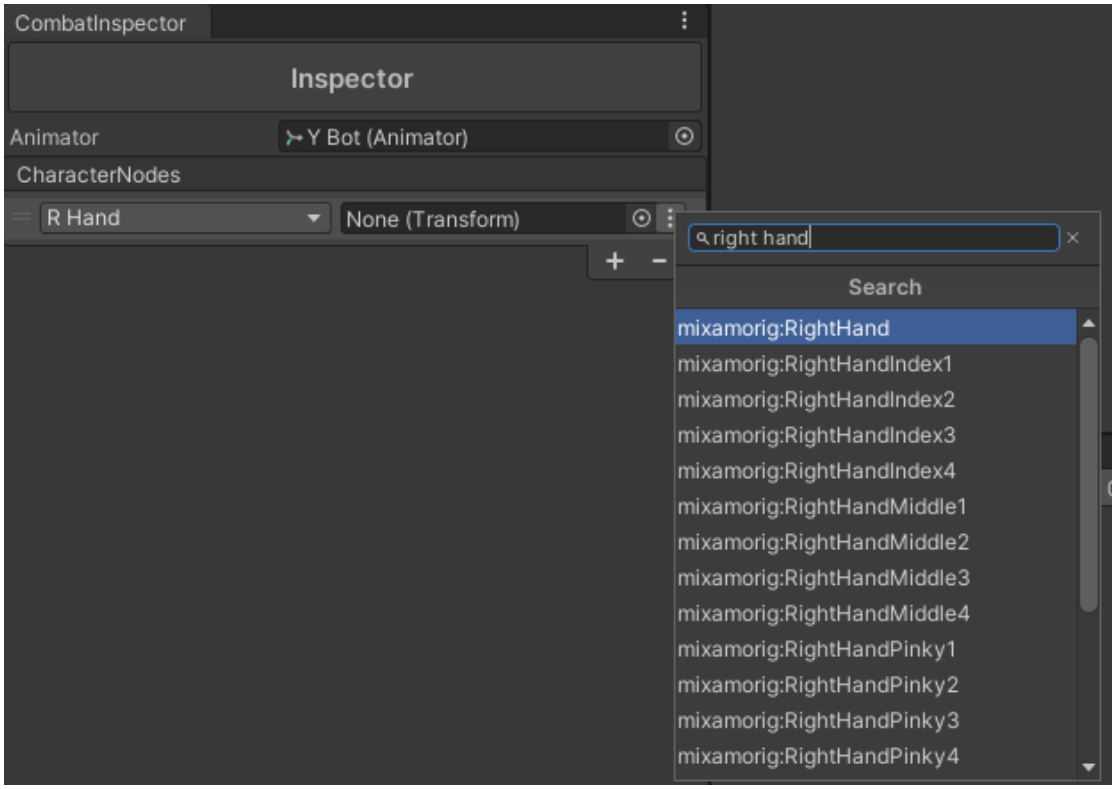
Click the Record button on the left side of particles, and this will show the offset handles of particle in the scene. Press W to control the particle position, press E to control the particle rotation, and this will be auto recorded.



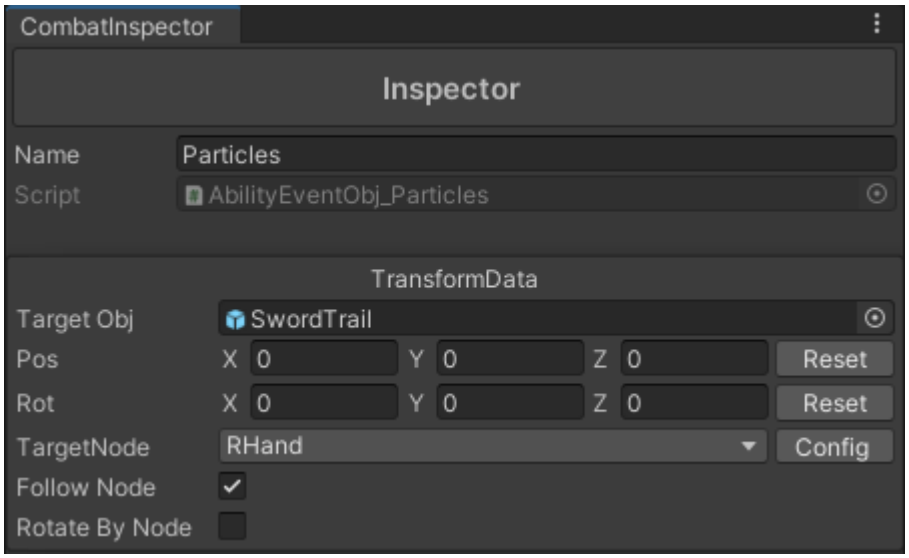
By default , the particle's offset is based on the animator position and rotation, and it will not follow anything after instantied.



If you want the particle to instantiated relative to a transform on the character, you should at first config nodes. Click the config to go to the node configuration.



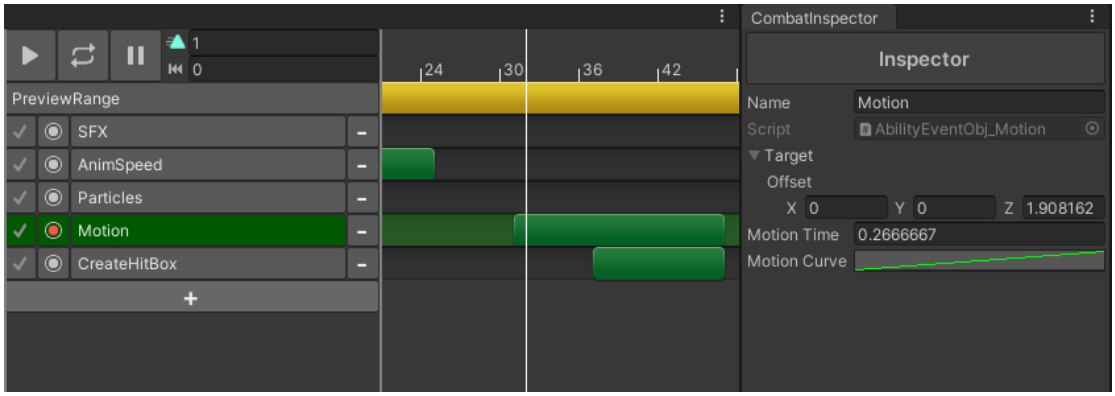
Click the 3points button on the right to search and assign transform node which is the child of the CombatController, or just drag the target transform to it.



Now you can assign the TargetNode to the node you just created.

If the Follow Node is on, this particle will follow the node's transform in preview and in playmode, else it will not move after instantiated.

If the RotateNode is on, the particle's rotation is relative to the node's rotation.

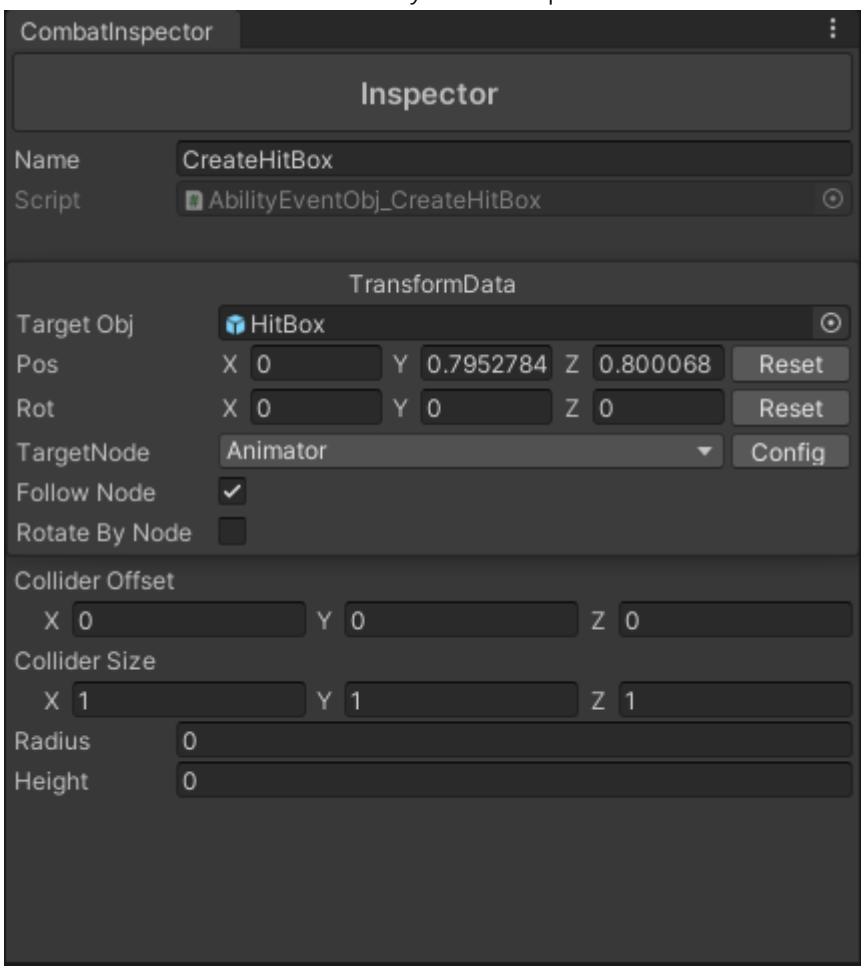


The motion is the character movement data in the animation. You can also click the record button and control the handle in scene to assign the target distance. And don't forget to give it a curve, or the character will not move.

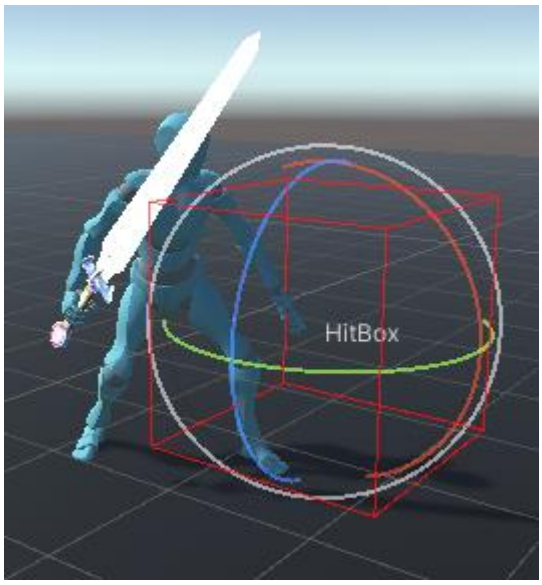
The Motion use the Move function in MoveExecutor to move, so if you want to implement your own physics, you can change the code in the script "MoveExecutor.cs".

Currently, there should only be one motion event in timeline editor.

The CreateHitbox event works very similar to particle event.



The targetObj requires an object with Hitbox Component and Sphere collider or Box Collider. After assign the hitbox, you can, like particles , control the position and rotation offset in scene. In addition, you can change the offset and size of the collider by handle.



Customlize you own Event

For example, we want to create a event that prints HelloWorld string.
Create a script, and rename it to AbilityEventObj_HelloWorld.
Find the AbilityEventTemplate txt file in CombatEditor/AbilityEventObj, and paste it to the AbilityEventObj_HelloWorld.

```
using UnityEngine;
[AbilityEvent]
[CreateAssetMenu(menuName = "AbilityEvents / FillInType")]
public class AbilityEventObj_FillInType : AbilityEventObj
{
    (FillInDatas)


    public override EventTimeType GetEventTimeType()
    {
        return FillInTimeType;
    }
    public override AbilityEventEffect Initialize()
    {
        return new AbilityEventEffect_FillInType(this);
    }
}
public partial class AbilityEventEffect_FillInType : AbilityEventEffect
{
    public override void StartEffect()
    {
        base.StartEffect();
    }
    public override void EffectRunning()
    {
        base.EffectRunning();
    }
    public override void EndEffect()
    {
        base.EndEffect();
    }
}
```

Replace all the “FillInType” to “HelloWorld”, which is the event name.

```
public class AbilityEventObj_HelloWorld : AbilityEventObj
{
    (FillInDatas)


    public override EventTimeType GetEventTimeType()
    {
        return FillInTimeType;
    }
    public override AbilityEventEffect Initialize()
    {
        return new AbilityEventEffect_HelloWorld(this);
    }
}
```

In the FillInDatas, you can set some datas required to display, for example, the particle event need the particle obj datas. In this case, we just need a string.

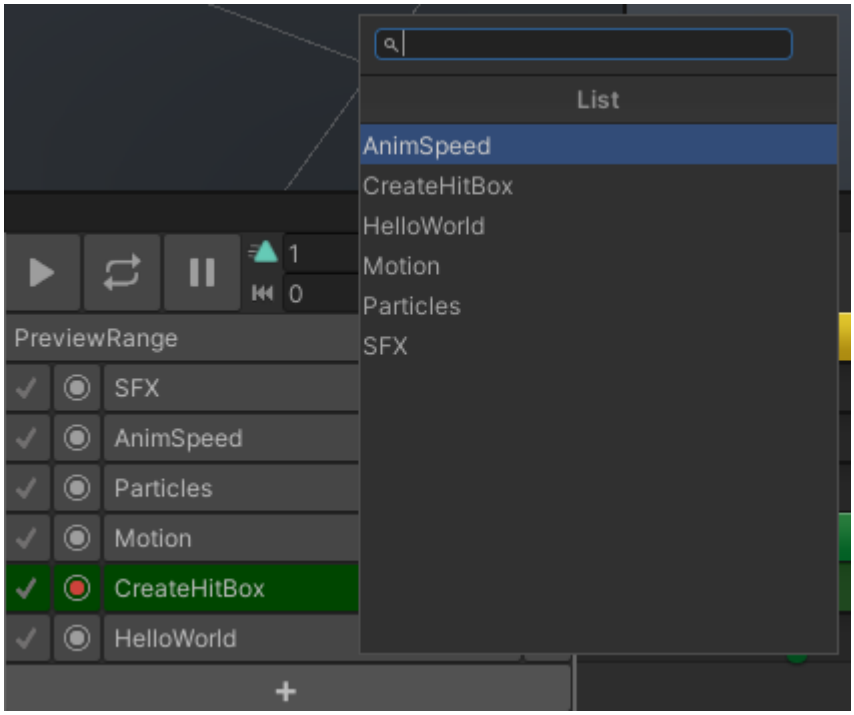
The Event type defines whether this event is a range or a point. In this case, this event only trigger in one frame, so this should be a point.

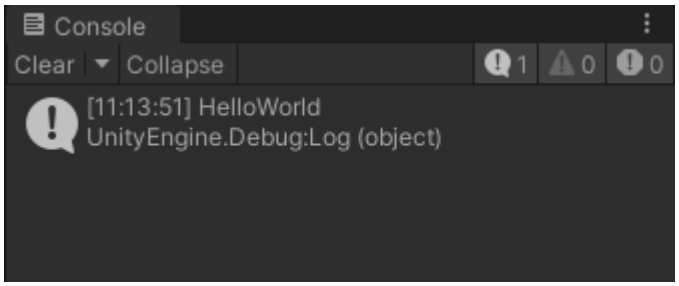
```
public class AbilityEventObj_HelloWorld : AbilityEventObj
{
    string DisplayString = "HelloWorld";

    public override EventTimeType GetEventTimeType()
    {
        return EventTimeType.EventTime;
    }
    public override AbilityEventEffect Initialize()
    {
        return new AbilityEventEffect_HelloWorld(this);
    }
}
```

```
public partial class AbilityEventEffect_HelloWorld : AbilityEventEffect
{
    public override void StartEffect()
    {
        base.StartEffect();
        Debug.Log(TargetObj.DisplayString);
    }
    public override void EffectRunning()
    {
        base.EffectRunning();
    }
    public override void EndEffect()
    {
        base.EndEffect();
    }
}
```

On StartEffect, we show the string in the EventObject.





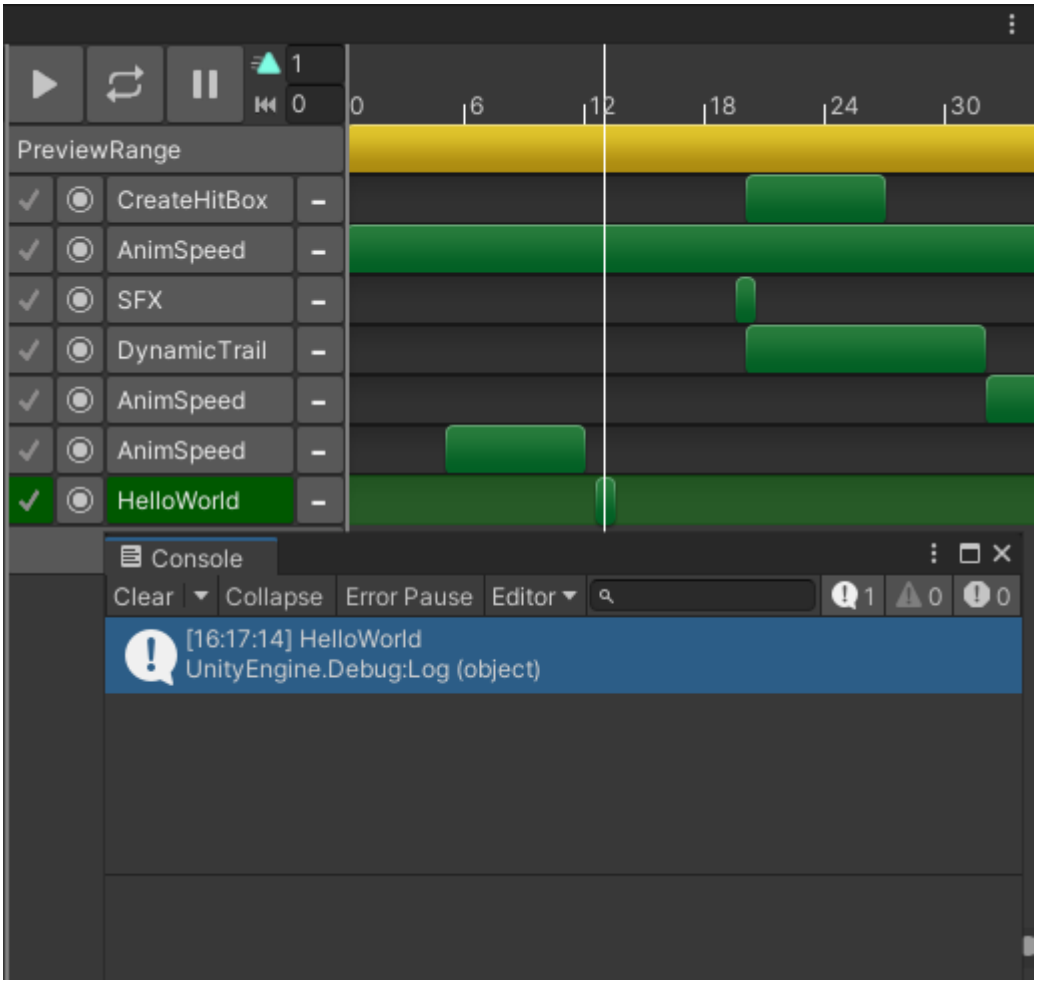
Add this Event to the track, and at the event start time, you can see a the log “HelloWorld”.

If you want to creat a ranged event, just replace the EventType to EventRange. Do what you want to do In StartEffect(), EffectRunning, EndEffect(), these 3 functions will call when you enter, stay, and exit the range in this animation. EndEffect() will auto call if the animator is not on this animationclip but the event have already start, for e.g, if you are attacking and you are on hit, then many events that are running should stop.

Add preview in scene mode to your event

Create the a preview is more complex than simple events. For example, we want to the console to log a message when in the scene.

1. Copy and paste the code CombatEditor/AbilityPreview/AbilityEventPreview_Template.txt. Rename all the “FillInType” to “HelloWorld”.(this requires the AbilityObj_Helloworld.cs in the previous section.)
2. There are many functions that looks scary in this script. But you don’t need many of them if you just want to make something simple. In this case, you just need to Debug.Log(“HelloWorld”) in the OnStartFrame.



3.After that, you can see a “Hello world” shows everytime you preview on the target frame.

Functions in preview template explained

1. Init Preview() : This functions is called when create the preview. Reset preview happens when you select the ability, toggle preview button, change the property in abilityObj, etc. Usually you need to instantiate what you want to preview here and set the preview transform parent to preview group.
2. OnStartFrame() : This function calls when you drag the frame to the startframe of the event. It is used for something like trigger some sound.
3. PreviewRunning(float CurrentTimePercentage): This function is called everytime you drag the animation percentage or play, the CurrentTimePercentage is the percentage of the target animationclip.
4. PreviewUpdateFrame(float CurrentTimePercentage): This function is called everytime you drag the animation percentage or play, but it is only called once on the same frame.
5. PreviewRunningInScale(float ScaledPercentage): his function is called everytime you drag the animation percentage or play, but the Scaled Percentage is the percentage time that after time scaled event on this ability. That means, ScaledPercentage * AnimationClip.length is the real time in preview. It is used to calculate the true preview time of particles after time scale.
6. NeedStartFrameValue(): If this returns true, the preview class will sample the animation at start frame and get some data. For example, if you want to spawn something in a frame, you need the data of that frame everytime the preview refreshes.
7. GetStartFrameDataBeforePreview(): You can fetch some data at the startframe in this function if NeedStartFrameValue() returns true. During this function, the transform data on character is calculated based on the animation of startframe. You can see the example use of these case in the AbilityEventPreview_InstantiateObj.cs.
8. DestroyPreview(): DestroyPreview is called everytime the preview is destroyed, for e.g, when you want to save the prefab, exit the scene, close the window, change current inspected ability, change property of abilityObj, the preview all need to be reset. So remember to clear the preview object, or change the property modified in this preview to the default value.
9. BackToStart(): Called when you click the stop animation button. When you click play or stop animation button, the previews are not destroyed, but this function is called. For e.g, the dynamic trail of weapon should clear after you click the stop animation button.

Complex preview: Instantiate an object with a handle

In this plug-in, many abilityEvent need to instantiate an object that is relative to nodes on character. For example, the particles, the light, the motion…and the transform data of the instantiated object could be modified by using default handles in scene. This is very complex to achieve, but you can inherit the InstantiateObj event class to do it easily. So how to do that?

1. Create a AbilityEventObj by template, like mentioned in page "Customlize you own event."
2. The script by default inherits from AbilityEventObj, please inherit from AbilityEventObj_InstantiateObj. In StartEffect(), you can call Obj.ObjData.CreateObject(_combatController) to create the object in playmode.
3. Create a AbilityEventPreview by template, like mentioned in page "Add Preview in scene mode to your event".
4. The script by default inherits from AbilityEventPreview, make it inherit from AbilityEventPreview_CreateObjWithHandle. Now, the field InstantiedObject is the object that is created in preview.
   Now you can create an preview object with handle!