Overview

This tutorial will guide you through adding tags to fragments and how to preview a sequence. It uses samples that are part of the SDK. Feel free to follow along with the step by step instructions.

Before doing this tutorial you should know how to open the editor, how to load a preview setup and what fragments and fragmentIDs are. For a tutorial on this see Mannequin Editor Tutorial 1 - Preview Setup, Fragments and Saving.

The tutorial will use a lot of pictures and a simple color code to distinguish areas to interact with and areas to look at:

Tags

Assigning Tags to Fragments

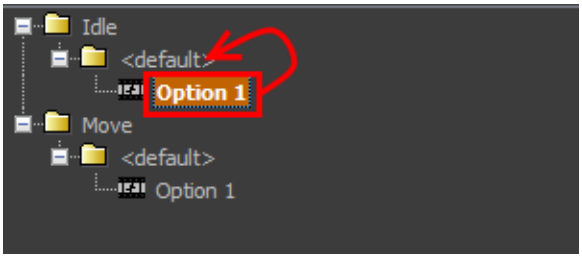
Load the preview file for the second tutorial called **"sdk_tutorial2preview.xml"**. See Loading the Preview File (but this time the file is called "sdk_tutorial2preview.xml").

If the file manager pops up you can press "Undo Changes to Selected Files" to ignore your previous changes.

You will notice that this time we create 2 fragments (one under the FragmentID "Idle" and one under the FragmentID "Move").

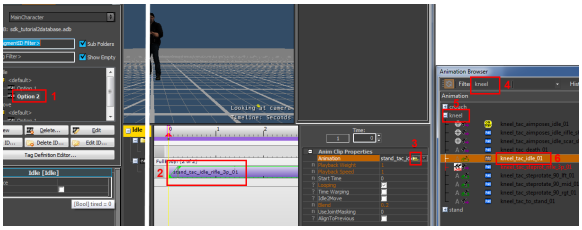
The bit under the fragment browser displays something new: you can now assign what are called *Tags* to the fragments. What does this mean? In the previous tutorial we just made a couple of variations for a FragmentID. When the game requests that FragmentID ("Idle" in our example), fragments are basically chosen randomly from those variations. Now we can start adding "tags" to the fragment to limit the cases in which certain fragments get picked. For example, we can mark a fragment with the tag "tired" so it only gets selected when the character is "tired". Or we can mark certain fragments with either "kneeling" or "standing" so we can create different stance variations for the same animation.

Let's try this last example now. Currently the fragment inside "Idle" has an animation where the character is standing. Let's make a variation that is kneeling. The simplest way to do this is by first copying the old fragment using right-mouse-button drag-and-drop:



Next we modify the animation in this fragment to point to the kneel_tac_idle_01 animation:
For example like this:

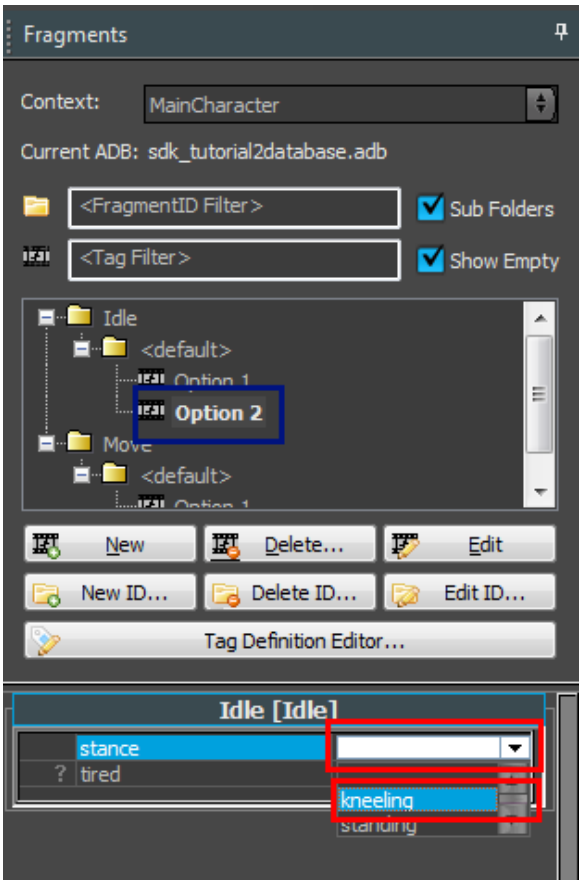
1. Make sure option 2 is selected in the fragment editor. This should already be the case, look for "Fullbody: (2 of 2)" which specifies that the second option is selected). If not the case, double click "Option 2".
2. Select the animation clip (should already be selected, it gets a green border when selected).
3. Open up the animation browser.
4. Optionally use a filter (we typed "kneel" in the example below).
5. Find the animation in the kneel section.
6. Select the animation into the clip by double-clicking.



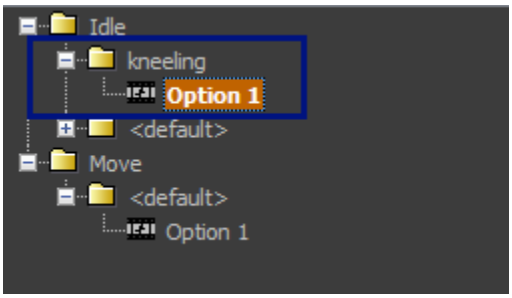
Now we tag this fragment with the tag "kneeling".

Again, make sure you have the fragment in boldface.

Next select "kneeling" from the "stance" combobox:



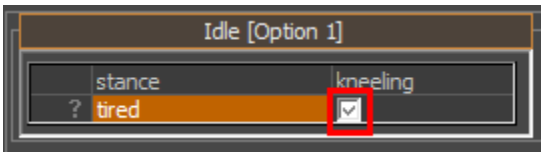
Note how this change affects the fragment browser. What used to be "option 2" in the "default" section now became "option 1" in the "kneeling" section.



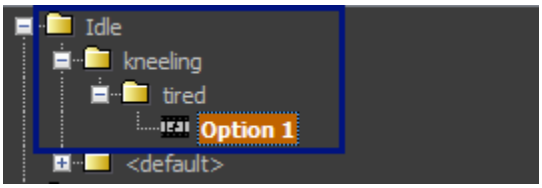
Now whenever the game requests "Idle" with tag "kneeling", it will get the new fragment. But if the game requests "Idle" with tag "standing" instead, it will take the default option.

> You might wonder why you need these "<default>" options once you gave all options appropriate tags. If we know the only options are "standing" and "kneeling", can't we simply tag that other fragment as "standing" and get rid of the default option? It's a matter of style but it might be a good idea to have some kind of "default" to fall back to. For example whenever a new stance gets added it's useful to have some good fallback behavior. If the system cannot find a matching fragment you might get T-poses. It depends on your production philosophy whether you want to enforce T-poses for missing fragments (which forces bugs to be found) or want to "sweep it under the rug" with a default option and only fix it up when needed.

We call "stance" a *Tag Group*. Some tags are inside tag groups, some other tags, like "tired", are not. Putting tags in a group like that makes sure you can only select one of the options in the group at the same time. So you cannot be both "standing" and "kneeling" at the same time in this setup, but you *can* b e both "kneeling" and "tired". Let's see what happens when we do that, let's also select "tired":

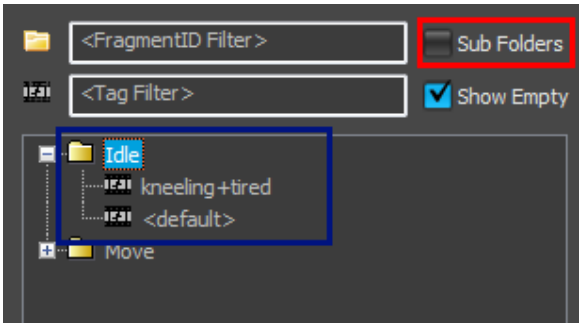


And this is how this shows up in the fragment browser:



Our fragment now became an option within "tired", and that sits within "kneeling" (this gives you a hint that "kneeling" is in some sense more important than "tired", which is the case, "kneeling" has higher priority than "tired" in this setup).

If you find the folders for the tags confusing and you just want to see the whole list of fragments you can use the "Sub Folders" checkbox:



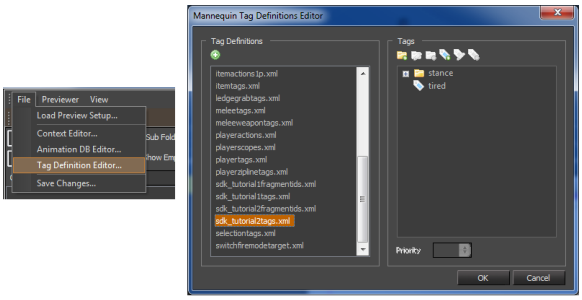
Note how our fragment with the tags "kneeling" and "tired" now shows up as "kneeling+tired".

> The order in which the fragments are listed reflects the order in which they get selected. If there are multiple equivalent matches the first one in the list that matches gets chosen. For example, say there is a tag called "tired" and a tag called "scared". You have one fragment that is tagged "tired", and another fragment is tagged "scared". Now the game looks for a fragment for a character that is both "tired" *and* "scared". If "tired" and "scared" have the same priority it's pretty much undefined which fragment will get chosen (it's currently related to the order in which they are defined), but the editor will show you the fragments in the selection order. Fragments that are first in the list get selected first.

> Selecting the empty option within the tag group "Stance" means "I do not care which stance it is for, it's a default, a fallback". Similarly, leaving the "tired" tag unchecked means "I do not care whether or not you are tired". It specifically does *not* mean "not tired". If you want to be able to specify fragments specifically for a "not tired" state you need to add a new tag (e.g. called "notTired").

Editing Tag Definitions

Though we will not explain this in this tutorial, you can edit the tags in the *Tag Definition Editor* which can be found in the File menu:



Previewing Sequences

What are Preview Sequences

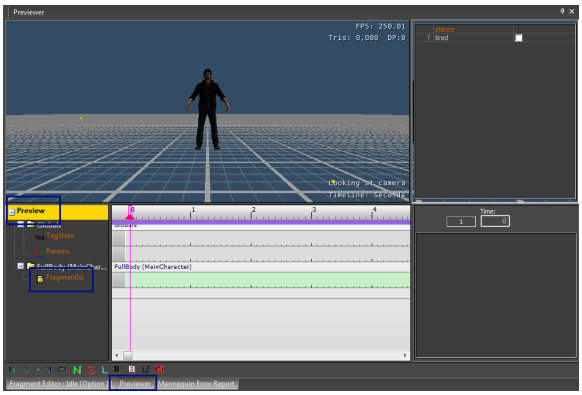
Up until now we just worked on little 'fragments' of your game's animation (literally). These will be sequenced together by the game code, but the CryMannequin editor also provides a way to preview how this sequencing would look even without running the game. Actually the preview sequences or are not used in the game at all! This is useful for all kinds of things:

- Working out 'what if' scenarios. e.g. "How would it look if the game would request the Move after Idle while kneeling?".
- Analyzing Bugs: You can record a real sequence from the game and load it up in the previewer. Or you can set up a sequence yourself to mimic what the game code does and see if it works outside of the game.
- Testing: Setting up little test scenarios and reloading them later on to see if they still work. You can also preview which transitions the system will pick (see the next tutorial)

Creating a preview sequence

To try out and preview how your fragments will look in the game the editor has the Previewer. You can find this one as a tab on the right side of the CryMannequin editor, please select it:

This is how it will look:



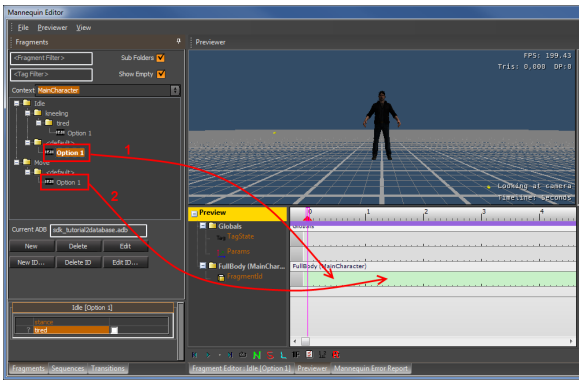
Note that it looks quite similar to the Mannequin Fragment Editor we worked in before. But there are a couple of details that give it away (apart from the tab at the bottom that says "Previewer").

Instead of the name of the fragment on a colored background it now simply says "Preview" and instead of having animation & procedural layers we now have a track labeled "FragmentID".

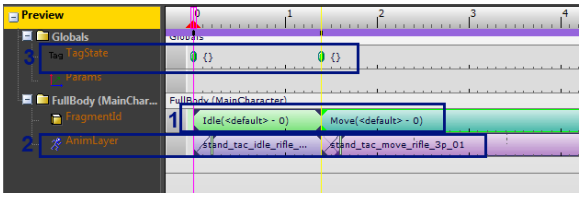
Let's see if we can set up a little sequence where we play our default "Idle" followed by our default "Move".

To do this we drag both fragments onto the FragmentID track:

1. Drag the default "Idle" fragment onto the beginning of the FragmentID track.
2. Drag the default "Move" fragment a bit further on the FragmentID track.

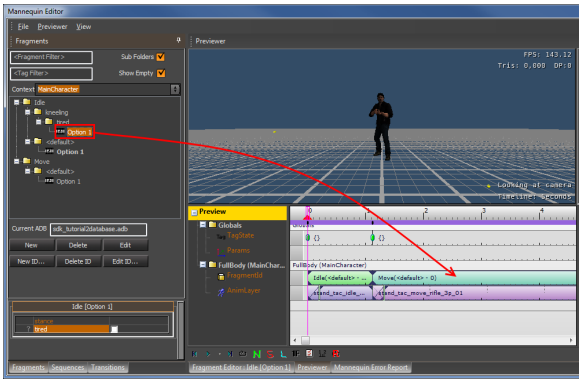


A lot happened automatically while you were dragging, let's explain what happened:



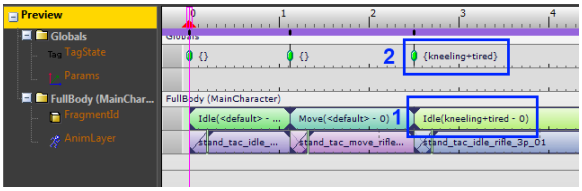
1. You added "Idle" and "Move" on the FragmentID track. This mimics what the game does to play animation: it just requests "Idle" and then "Move". The previewer mimics the selection process, and as we are not requesting any specific tags (we'll do that later) it just picks the default fragments for both FragmentIDs.
2. The system shows which actual animation clips those fragments translate into and shows that on an animation layer.
3. When we dropped the fragments this also created two dots (*keys*) on the *TagState* track. What they mean should become clear in the next step...

Now let's drag the special fragment that has both kneeling and tired tags onto the track too:



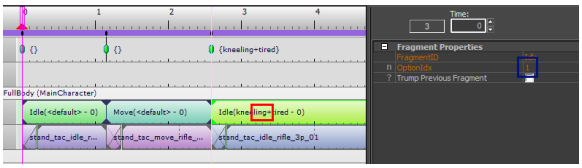
The fragment appears, and notice:

1. The fragment is now labeled "kneeling+tired - 0", as we would expect.
2. The TagState track now contains a key that is labeled "kneeling+tired". This mimics the fact that the game has to somehow request those tags too, along with the FragmentID.



Option Index

If you select a fragmentID key you can change the option index (OptionIdx) in case you want to select a different option:



Here we have OptionIdx = 1, so the first option is selected.

If you set OptionIdx = 0 the system will pick an option randomly and the FragmentID key will not display a number at all.

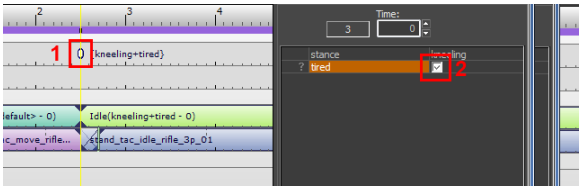
> If you use an option index which is higher than the number of options it is divided by the number of options and the remainder determines the actual option. For example if you have 3 options and you set OptionIdx=5, option 2 gets chosen.

In case you are wondering, we'll talk about the "trump previous fragment" option in a later section.

TagState Keys

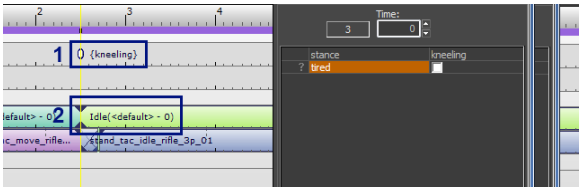
Let's play a bit with this TagState to get a feeling of how it works.

1. Select this key .
2. Toggle the "tired" tag off.

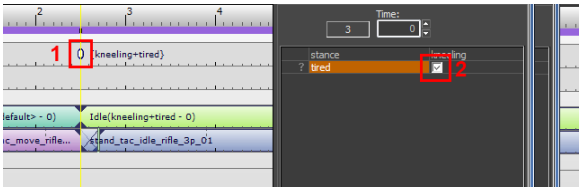


The result is:

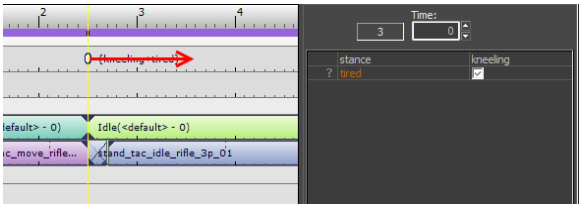
1. The key is now called simply "kneeling" (that is the only tag that is still set inside the key).
2. The fragment that is selected is now the default one again, "<default> - 0". This fragment now is the 'best match' for the game's request.



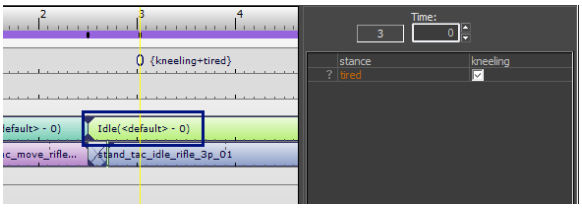
Let's play some more. Turn the "tired" tag back on inside that key:

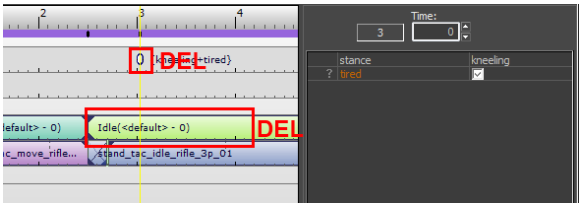


Move the key a bit to the right:



This simulates a situation in which the game requests those tags ("kneeling+tired") *after* requesting the FragmentID "Idle". This means that at the moment "Idle" is requested the tags are not set, and the default fragment will be chosen:

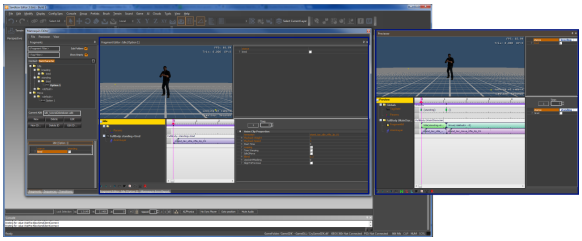




Undocking the Previewer

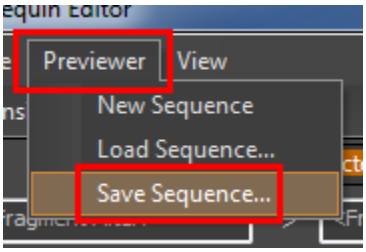
If you want you can pull out the Previewer from the mannequin editor and put it alongside the rest of the editor. This makes it possible to work on individual fragments and see immediately how this would change the sequence.

You can also dock the previewer somewhere else if you want to. For example here is the Previewer alongside the Fragment Editor:



Saving & Loading Preview Sequences

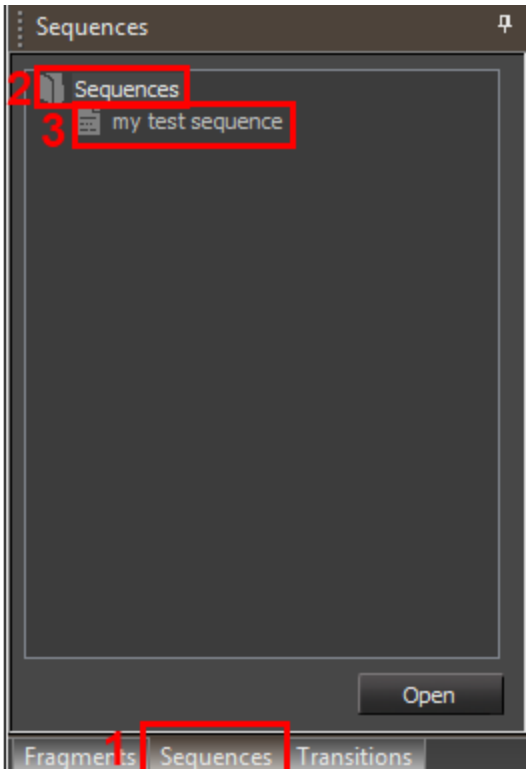
You can save the sequence we just set up using the "Previewer/Save Sequence..." menu item. Give it a nice name and save it, for example name it "my test sequence".



To load a sequence you can use the "Previewer/Load Sequence..." menu item or even easier: just open it from the Sequences tab on the left side.

1. Select the Sequences tab.

2. Double-click the folder icon to open the folder (the sequence you saved before should now show up).
3. Double-click your sequence to load it (or select it and press the Open button below).

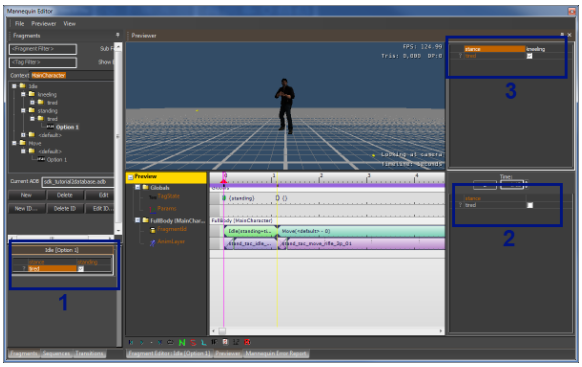


Preview Filter Tags

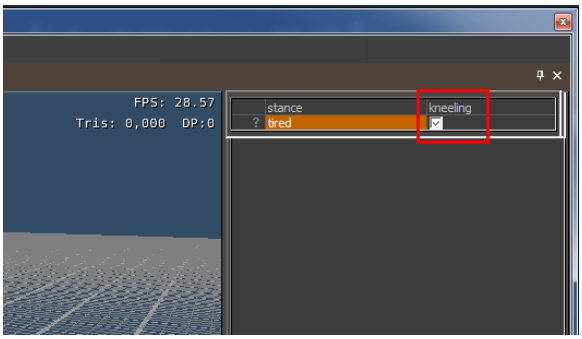
Using tagstate keys while setting up a preview sequence can get tedious. The drag and drop helps in setting up those keys, but there are cases where you just want the same tags to be set all the time. Or you'd like to set up a generic sequence and want to see how the sequence would look under different circumstances. For example you might want to set up a basic motion sequence like "Idle -> Move -> Idle" both for "standing" and "kneeling". Maintaining all those different sequences can get annoying when the number of stances gets higher and higher. It's easier if you could set up a generic sequence and then fill in the stance tag afterwards. This is where the *Preview Filter Tags* come into play.

At this point you might have noticed that that there are 3 areas in the editor which look very similar. But don't let this confuse you, they all have different meanings! Two of them we already talked about, the third one displays the *Preview Filter Tags*. Refer to the picture below:

1. These are the tags associated with the Fragment we are currently editing (the boldface fragment in the Fragment Browser). See the section on Assigning Tags to Fragments.
2. These are the tags set in a TagState key which we put on the TagState track in the Previewer. See the section on TagState Keys.
3. These are the *Preview Filter Tags*.

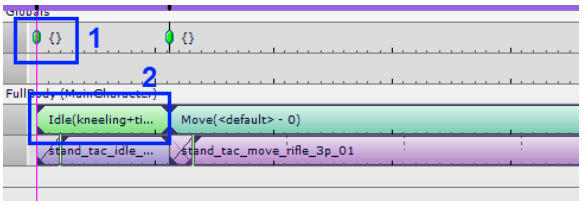


Let's play with it. Let's select both "kneeling" and "tired" in it:



Now note how this change in the default tagstate *changes* the sequence in the previewer.

1. Even though the first TagState key is set up to be empty, it is labeled "{}"...
2. ...the kneeling+tired fragment gets selected.



What we did now is add "kneeling" and "tired" to the Preview Filter Tags. These work as a default for the TagState. So the tags that are set by default for the whole sequence are "kneeling" and "tired".

The TagState keys still *override* these default tags though. Say you have "standing" selected in the default tagstate you can still override it with a "kneeling" TagState Key. An empty tagstate key, in other words a key that is labeled "{}", overrides nothing at all.

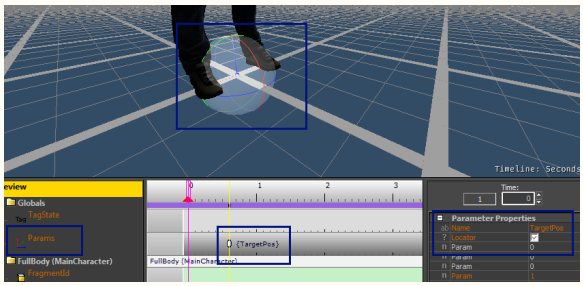
Params Track

Work in progress

The game can set certain 'parameters' that get picked up by the clips. By adding keys on the params track you can control these parameters in the previewer.

Parameters have names (eg. 'TargetPos'). Which parameters and how they are used depends on the specific clips.

For example a 'positioning' clip takes a parameter that is a location&orientation in space to slide towards.



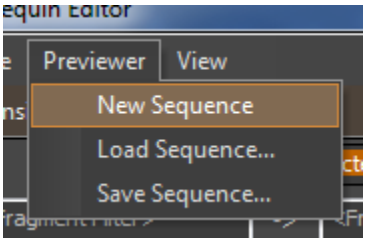
Fill in the name needed by the procedural clip or game. For the procedural alignment clips (all the PositionAdjustXXX clips) this parameter name *must be "*TargetPos" as in the screenshot above.

Select the "Locator" option for the gizmo to appear.

Use the buttons on the toolbar below, or they keys "W" or "E", to select translation or rotation mode for the gizmo.

New Sequence

If you would like to clear a sequence you can use the "New Sequence..." menu item:



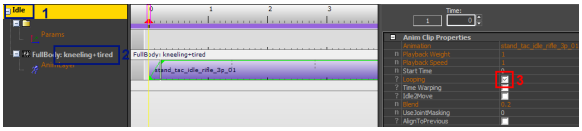
Trumping

There is a certain complication which we carefully avoided until now. When the game requests FragmentIDs, it is not guaranteed that the new fragment will immediately start. It is possible that the default transition between the previous fragment and the new fragment will delay the selection of the fragment. For example to finish an animation, or to wait until the correct foot is on the ground. In some cases this is fine, in other cases you really want to play the new fragment immediately. And this is what we call *Trumping*: basically just skipping the waiting period.

This can be simulated in the previewer too. Let's set up a little scenario where this happens.

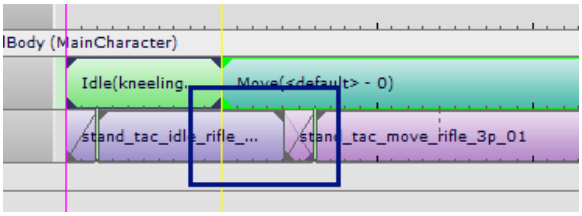
Open the Idle (kneeling+tired) fragment up in the fragment editor.

1. Just in case: Double check the fragmentID...
2. ...and make sure you have the correct variation, kneeling+tired, selected.
3. Select the first animation clip and uncheck the Looping setting.



Now go back to the previewer and look at a simple sequence (if you lost your sequence just drag the "Idle (Kneeling+Tired)" and "Moving" fragments into the previewer again as before.

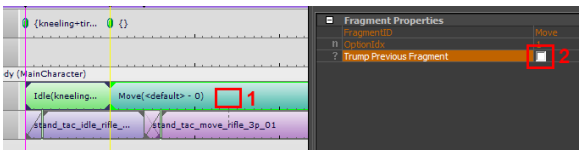
You might need to move the "Move" FragmentID key a bit to get exactly what is shown in the screenshot below, but you should notice that now the move animation is *delayed*!



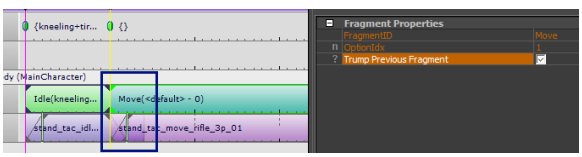
This means that the new animation only starts after the previous one has ended. This is the default behavior for non-looping fragments. Up til now we didn't set up sequences with non-looping fragments so it simply didn't show up yet.

It also means that this is the default behavior in the game. Whenever the game wants to skip this waiting period it sets things up in order to "trump". We can simulate this by:

1. Selecting the Move fragmentID key.
2. Checking the "Trump Previous Fragment" option.



The result is the following, where the animation starts immediately:



This concludes this tutorial.

Where to Go Next

You can continue with Mannequin Editor Tutorial 3 - Transitions.

Or you can review the concepts we touched: Mannequin Tag Definition Editor, Sequence File (xml), etc.