Quick Start Guide

Introduction

This is a quick start guide to get up and running with Motion Matching for Unity (MxM). This guide will not go into detail on every system. It is intended to provide a brief overview of the package to get you started. Please see the user manual and tutorial videos for more detail.

This written guide is a supplement to the tutorial video accessible from the Tutorial Playlist.

Please note that the animations referred to in this quick start guide are no longer the animations used in the demo scene which have been replaced with better mocap data. However, The old animations can still be found within the package

Note: This quick start guide is intended as an introduction to motion matching. It is not recommended to attempt to complete it with your own animations. Please consult the user manual and advanced tutorial videos for custom setups.

Standalone Demo

A standalone demo is available to download and play. This shows some more advanced application of MxM in a full demo with sword combat, different stances and actions. The level and assets in the standalone demo are NOT included with MxM. The demo scene that ships with MxM is different.

The standalone demo can be downloaded for free here.

Note: Note: The Standalone Demo was developed with a much older version of MxM (v1.7beta) and will be updated soon to v2.2

Support

Discord - send me a private message once you have joined with your Invoice Number so I can give you verified status and access to the private discord channels.
Tutorial Videos - In depth tutorial videos on using motion matching for Unity
Quick Tip Videos - Quick tip videos for those small things that you probably overlooked.
Issue-tracker - Found a bug? Report it here.
Forum
Unity Connect
Asset Store Page

Note: For the best level of support please use the Discord and the git-hub issue tracker. It is the easiest way for me to verify that you are a valid customer. I will only provide support for verified customers as an anti piracy measure.

IMPORTANT: Import your animations correctly. MxM cannot fix your broken animations!

Pre-Requisites **(IMPORTANT)**

'Motion Matching for Unity 'requires Unity 2018.4x as a minimum. It also requires Unity's new job system and a few additional packages. You will need to install these packages manually through the Unity package manager.

Note: *Support for Unity's automatic installation of dependencies has been removed because it is an unreliable feature that causes more problems than it solves.*

- **Mathematics**
- **Collections** (Preview)
- **Burst**
- **Jobs** (Preview)

The following package is also recommended but not required:

- **PlayableGraphVisualizer** (Preview)

For the demo scene you will need to download the **Cinemachine** package to get the camera to work.

To download packages navigate to Window -> Package Manager in the Unity Editor. Select the required package and choose install. For the 'Preview' packages you will need to click on the 'Advanced' drop down and make sure 'Show Preview Packages' is checked before it will show up in your list.

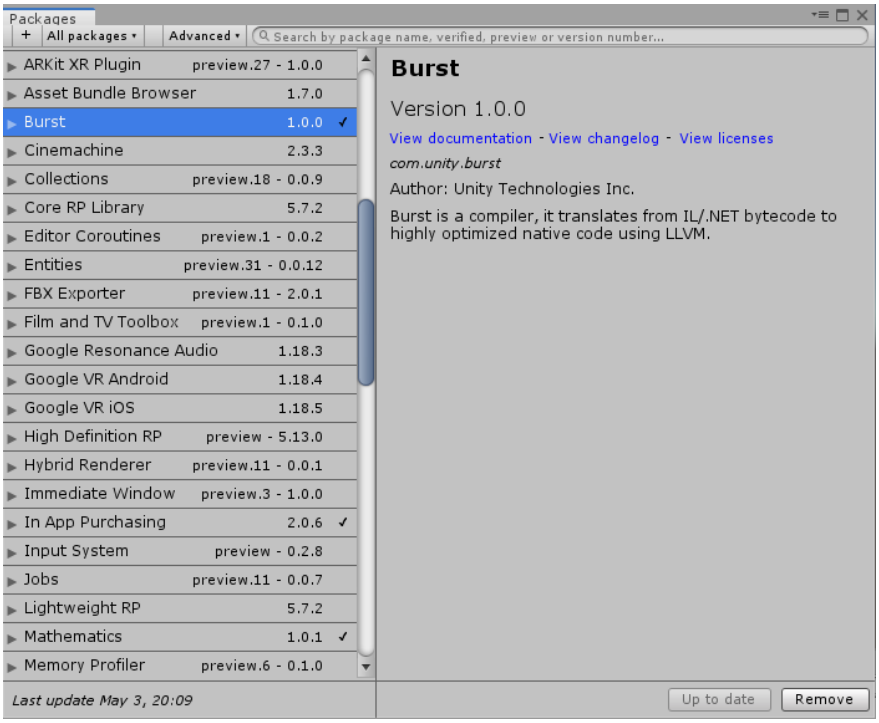

Figure 2 'Unity Editor package manager, install Burst, Mathematics, Collections & Jobs"

Import

Once you have purchased MxM you will be able to download it through the Asset Store window within Unity itself. Choose 'Window' / 'Asset Store' and navigate to 'My Assets' (figure 3). Find the Motion Matching for Unity package and choose 'Download'

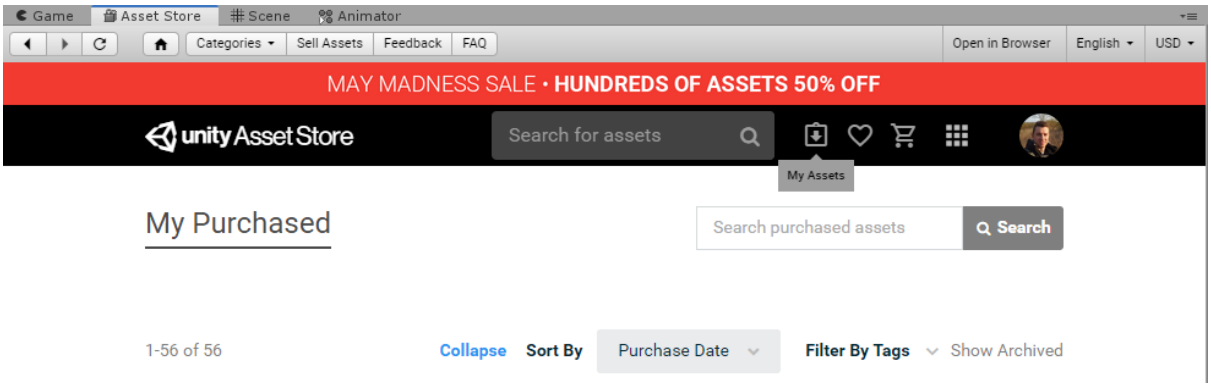

Figure 3 - 'Asset store embedded in the Unity Editor"

Once the download is complete choose import. You will be presented with the import window (figure 4). Leave everything checked and click the import button.

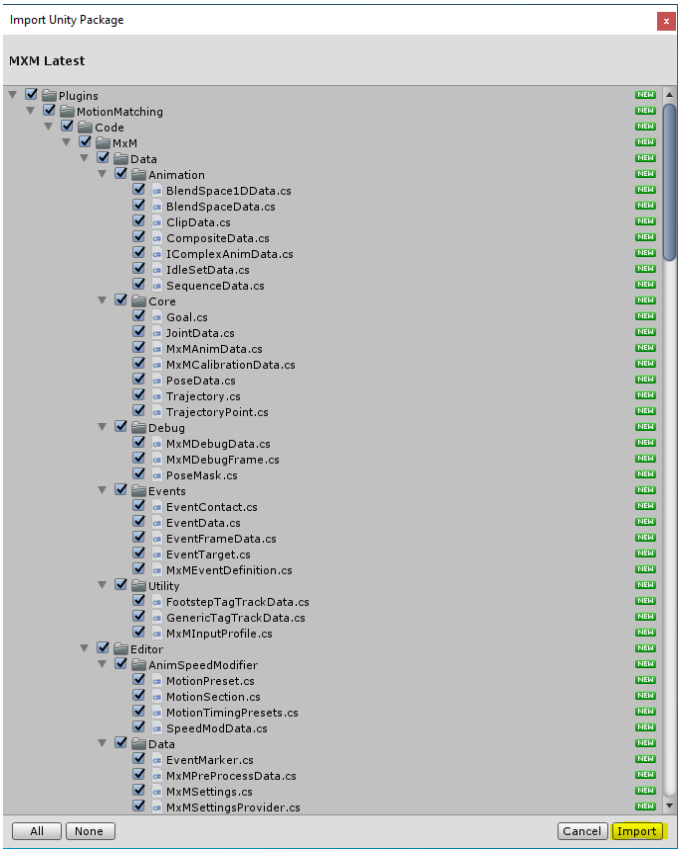

Figure 4 - 'Import window for MxM'

After a brief wait the package should be fully imported into your project.

Note: In some versions of 2019.3x +, the asset can be downloaded and imported from the package manager.

Demo Scene

In the project view, navigate to ‘Plugins / Motion Matching / Demo / Scenes’ and open the ‘MxMDemo’ scene (figure 5).

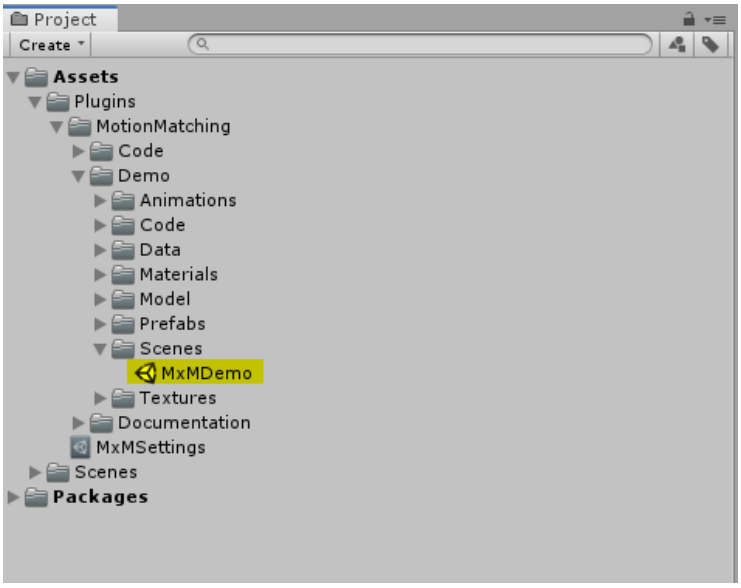

Figure 5 - ‘Demo scene location’

In the scene hierarchy select ‘Robot Kyle’ and notice the components in the inspector (Figure 6).

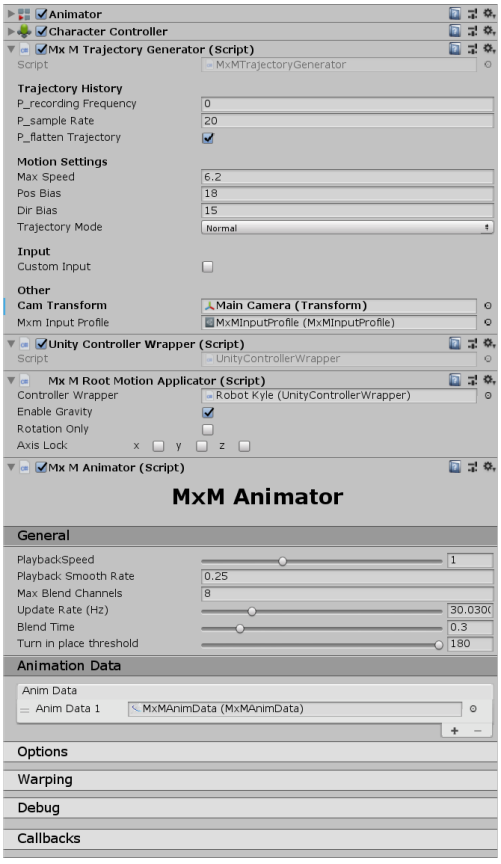

Figure 6 ‘Typical prefab with Motion Matching components”

The important components to notice are the following:

- **Animator (Mandatory)**
  For any animation in unity an Animator component is required. Motion Matching is no different. It doesn't, however, require an AnimationController to be slotted.

- **MxMAnimator (Mandatory)**
  This is the main component for motion matching and it is always required. It has all the settings and data for motion and it is what generates the actual animation. By itself, this component won't do anything. It needs another script to tell it what the desired future trajectory is. In this demo, it is done via the MxMTrajectoryGenerator component.

- **MxMTrajectoryGenerator (Mandatory - or Custom Alternative)**
  The MxMTrajectoryGenerator records past trajectory and calculates future desired trajectory based on player input. It communicates this trajectory (or goal) with the MxMAnimator so that MxM can match animations to it.

  This component has been made separate from the Animator for flexibility. It can be replaced with a custom trajectory generator that better suits your gameplay. See the full manual for details.

- **Character Controller (Not Mandatory)**
  This is the Unity basic 'Character Controller' component. It is not required at all for MxM to work and can be replaced with a custom controller of your choice, or no controller at all.

- **MxMRootMotionApplicator (Not Mandatory)**
  This component applies root motion and gravity to the character. However, also taking into account any animation warping (rotational or positional) generated from the MxMAnimator. This component has been made separate from the MxMAnimator for flexibility reasons.

  Without this component the MxMAnimator will apply root motion directly to the transform. However, with character controllers this is generally not desirable. The RootMotionApplicator is able to communicate with 'generic controller wrappers' so that the root motion can be applied in a way that the character controller needs to work properly.

  Note: Users can create custom root motion applicators that do not require a controller wrapper component.

- **Unity Controller Wrapper (Not Mandatory)**

This parameterless component is simply a wrapper for Unity's basic 'Character Controller' component. It is used by the MxMRootMotionApplicator to communicate root motion to the 'Character Controller'.

This wrapper is required because the MxMRootMotionApplicator was designed to be able to communicate with any controller through a wrapper. Custom controller wrappers can be created by inheriting from the GenericCControllerWrapper script.

Note: If you create your own RootMotionApplicator that communicates directly to a CharacterController of your choice, a wrapper is not required.

For this quick start guide, these components are more than sufficient but some customised alternatives may be necessary to get the most out of Motion Matching for your game.

Before we hit play, ensure that there is at least one MxMAnimData object inserted under the 'Animation Data' foldout of the MxMAnimator.

Click play, and you can run around the scene using 'WASD' and 'space' to do a jump. The character should also be able to automatically vault up, over and down from objects.

Note: The Vault Detector system in the demo is an example of how you could use MxM's warping tech to perform parkour like actions. It is a simple example only and not a part of 'Motion Matching for Unity' functionality.

Note: The animations used in this demo scene are the RawMocapData set supplied by Unity (Now no longer available from the asset store). They are in no way ideal for motion matching but are used here to demonstrate the system and how it can perform even without ideal animation data. Better results can be achieved with better quality mocap and even with quality cut clips if done right.

Click on the 'Gizmos' toggle button on the top right of the screen to see debug information. This will display a number of debugging gizmos that show exactly what is being matched.

- **Green** - These show the desired trajectory that is generated from gameplay input
- **Red** - These show the trajectory points of the current animation that MxM has chosen
- **Yellow** - These show the current position and local velocity (arrow) of the joints being matched. Take particular note how few joints are required.
- **Blue** - Root position
- **Cyan** - Current body velocity

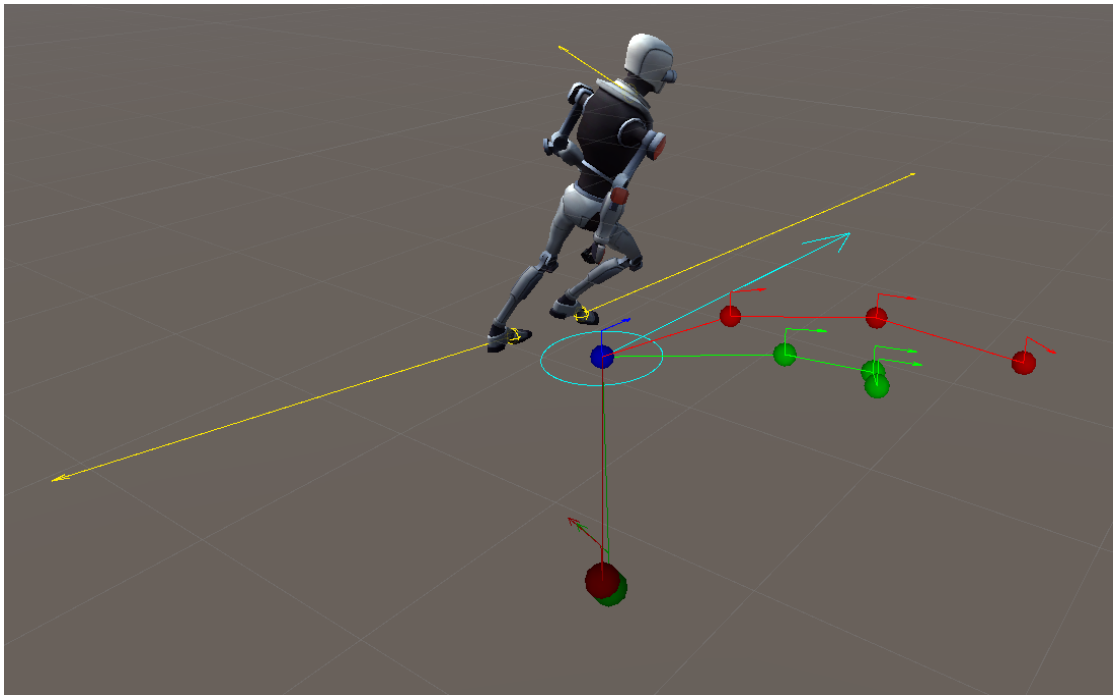

Figure 7 ‘MxM in action with debug gizmos showing’

Creating the Demo Scene Animator Yourself

In this section we will quickly learn how to setup a character similar to the demo scene without much effort at all. I have made very minimal changes to the RawMocapData provided by Unity, only removing some completely broken animations and mirroring some one sided clips.

Note: In this quick start guide we won't finesse the animation as much as has been done in the demo scene, it is only intended to get you familiar with the basics.

The PreProcessor

1. Right click in your project view and choose **Create / MxM / PreProcessor**
2. Select the new asset (MxMPreProcessData) and note the inspector (Figure 8)
3. In the 'General' foldout, drag and drop the Robot Kyle ***model*** (not prefab) into the 'Target Model' slot
4. In the 'Trajectory Configuration' foldout click the 'Add Trajectory Point' button four (4) times and set the trajectory point timings as shown in Figure 8.

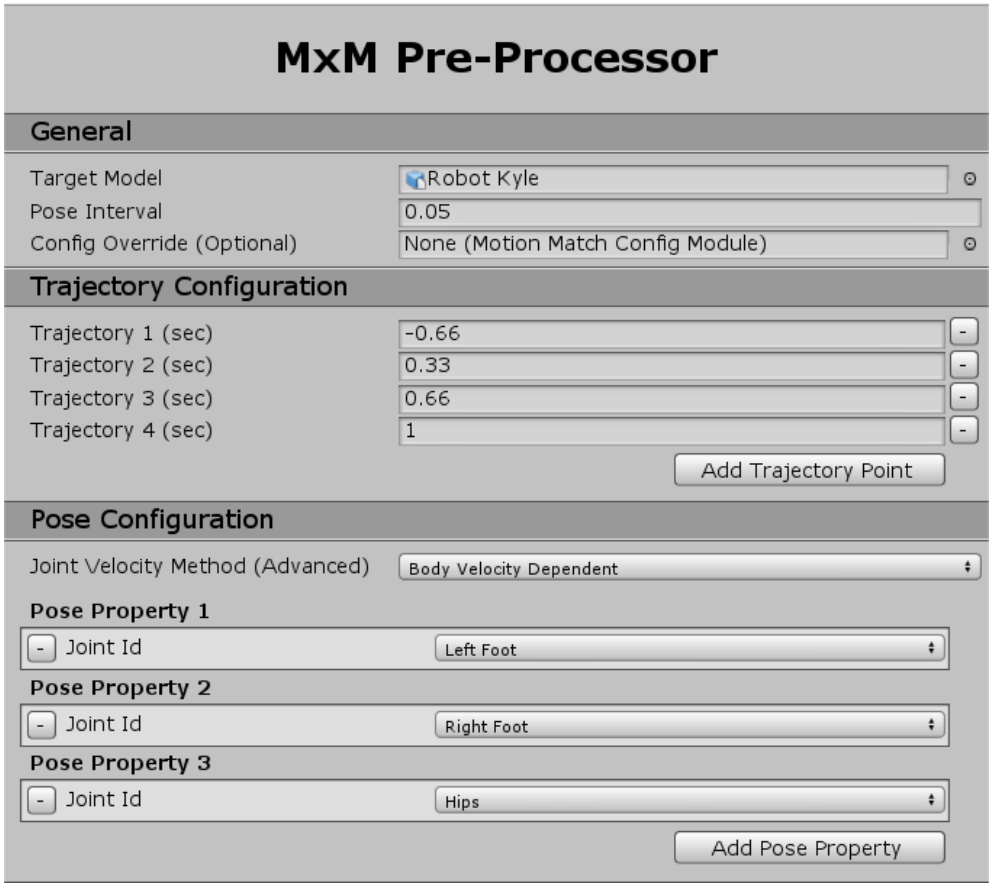

Figure 8 'Basic pre-processor config settings'

5. In the ‘Pose Configuration’ foldout click the ‘Add Pose Property’ button 3 times. Using the provided drop downs for each property select the bones to match including
    a. Left Foot
    b. Right Foot
    c. Neck
6. In the ‘Animation Data’, ‘Composites’ foldout, drag and drop all the animations contained in the *‘MotionMatching/Demo/Animations/Locomotion’* folder (figure 9)
7. In the ‘Animation Data’, ‘Idle Sets’ foldout, drag and drop the idle animation contained in the *‘Plugins/MotionMatching/Demo/Animations/Idle’* folder (figure 9)

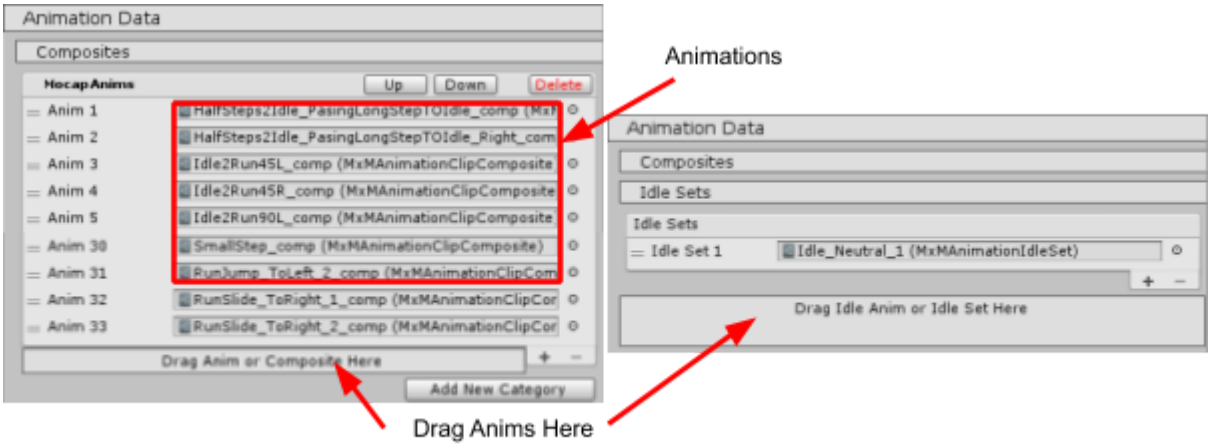

Figure 9 left - ‘Animation composites’, right - ‘Idle Sets’

Note: In the animation data, there are some cut clips. Cut clips require additional work when compared to mocap because they lack continuous animation to calculate trajectory points from. For these animations we need to specify how the trajectory should be calculated. This can be done in two (2) ways. Firstly, the trajectory can be extrapolated as shown in Figure 10, or the trajectory could be taken from another animation as shown in Figure 11.

To extrapolate the trajectory of an animation do the following:

8. In this new list of animations, double click on the ‘Idle2Run135L_comp’ animation. This will open up an editor as seen in Figure 10.
9. Uncheck ‘Ignore Edges’ to expand the options
10. Check ‘Extrapolate.

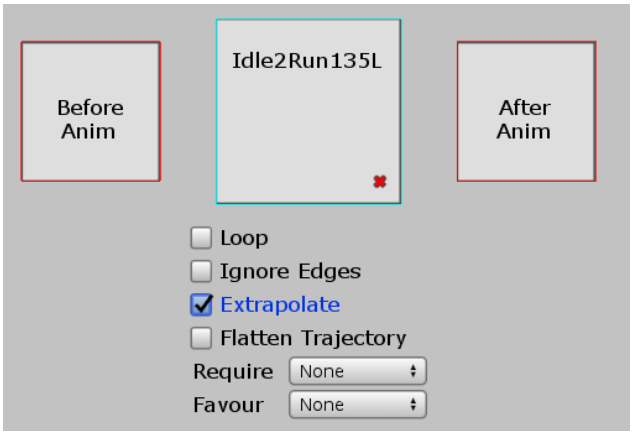

Figure 10 'Extrapolate the trajectory'

To calculate the future trajectory by stealing it from a different animation do the following:

11. In the composite editor, uncheck 'Ignore Edges'.
12. Drag an animation that you want to steal the future trajectory from into the 'After Anim' box of the composite.
13. This can be done for the past trajectory as well by dragging an animation into the BeforeAnim slot.

Note: The demo scene only uses extrapolation. Please see the PreProcessData file for the demo for exact settings for each animation. This file is found in *Plugins/MotionMatching/Demo/Data/Pre-ProcessData*

Note: The demo scene only uses extrapolation. Please see the PreProcessData file for the demo for exact settings for each animation. This file is found in *Plugins/MotionMatching/Demo/Data/Pre-ProcessData*

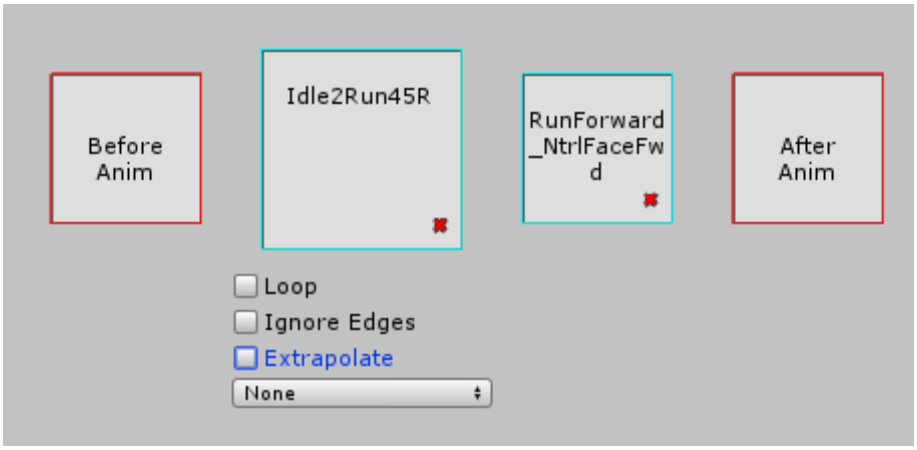

Figure 11 'Animation composite editor'

14. Double click on the idle animation in the Idle Sets foldout. This will open another editor (Figure 12) similar to that of the composite. For the 'quick start' just ensure that Min Loops and Max Loops is set to 1 and you can close this editor.

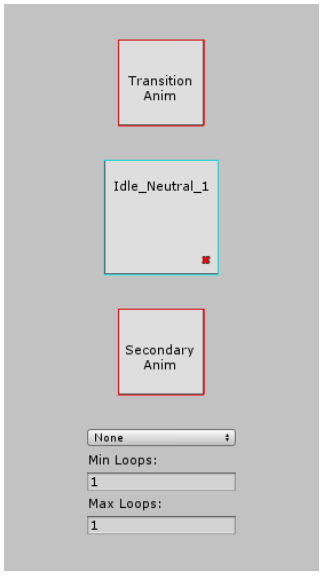

Figure 12 'Idle set editor'

15. From the *'MotionMatching/Demo/Animations/Locomotion/BlendSpace_Anims'* folder drag and drop the following animation clips into the BlendSpaces foldout:
    a. RunForward_NtrlFaceFwd
    b. WalkFWD

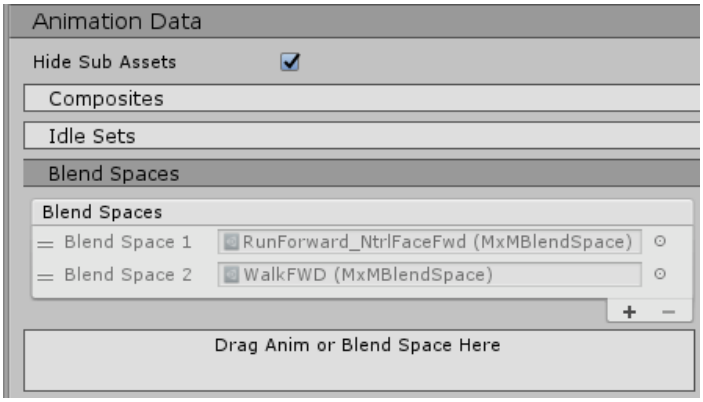

Figure 13 'Blendspaces Animations'

16. Double click on one of the blend spaces to open the blend space editor (Figure 14 & 15). Ensure that on both blend spaces, the 'Type' is set to 'Scatter X' with a value of 0.1 on the top toolbar.

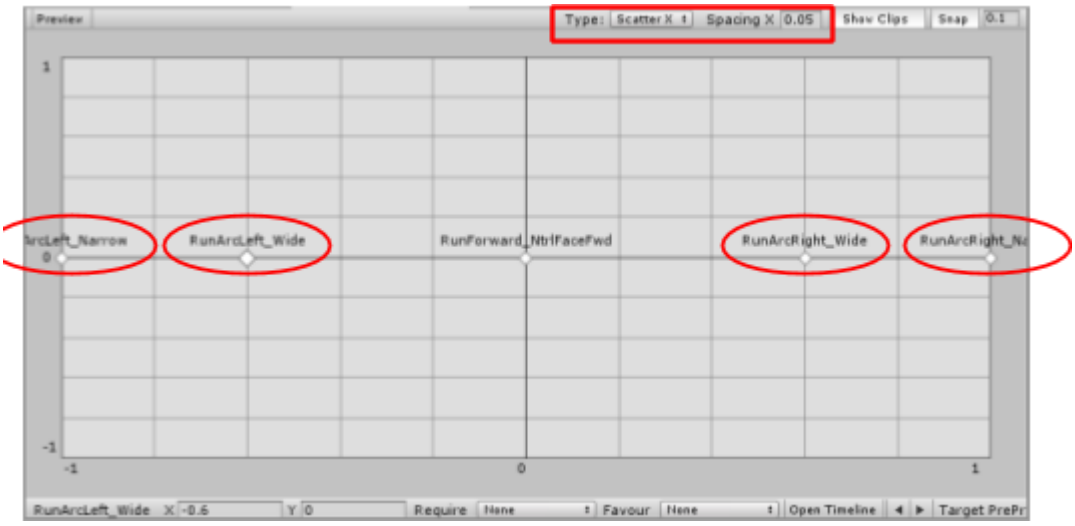

Figure 14 'Running BlendSpace Setup'

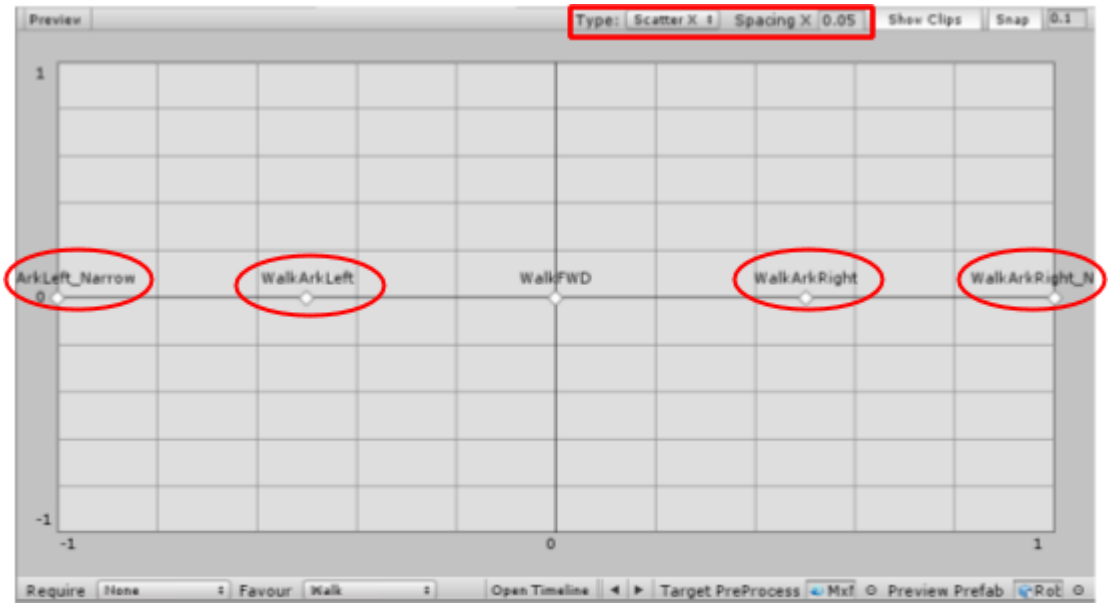

Figure 15 'Walking BlendSpace Setup'

17. For the running and walking blend spaces shown in Figure 14 and 15 respectively drag and drop the circled animations into the blend space. Position them as shown in the images above.

Note: Cut clips suffer from lacking coverage, particularly when it comes to arcing runs. The blendspace system in MxM helps fix this coverage gap by using the blended poses in-between arcs and forward runs.

18. Go to the last foldout *'Pre-Process'* and click the *'Pre-Process Animation Data'* button (figure 16)
19. Choose where to save the results of Pre-Processing (it generates an MxMAnimData asset)

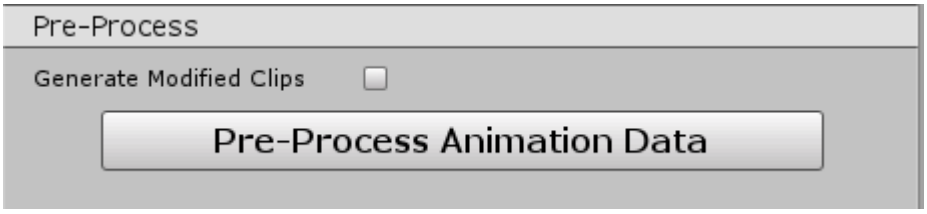

Figure 16 'Pre-process'

The AnimData

MxMAnimData is predominantly just data that does not get modified. It does, however, have a custom inspector showing metrics of the data stored within and it also allows you to create custom calibrations for motion matching.

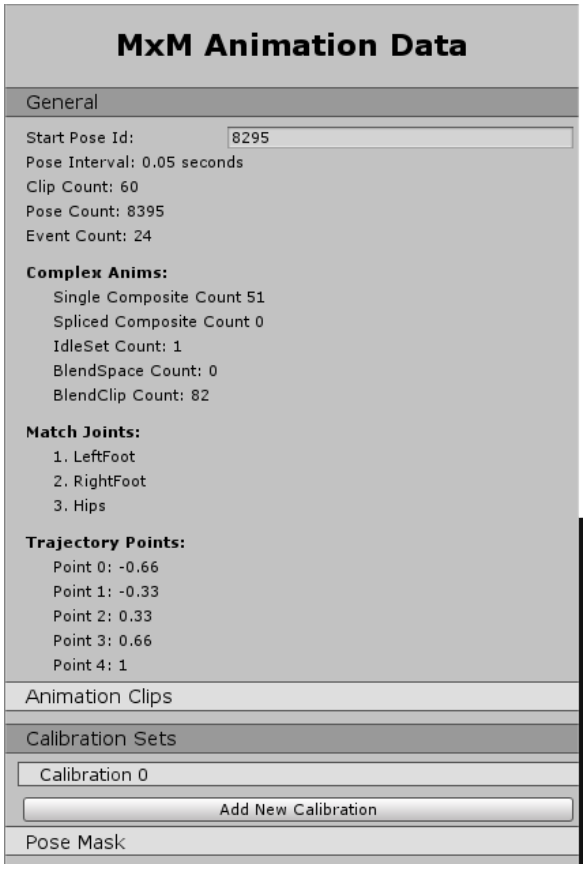

Figure 17 'The MxMAnimData inspector'

Calibration is out of scope for this quick start guide. For now just copy the calibration settings shown in Figure 18.

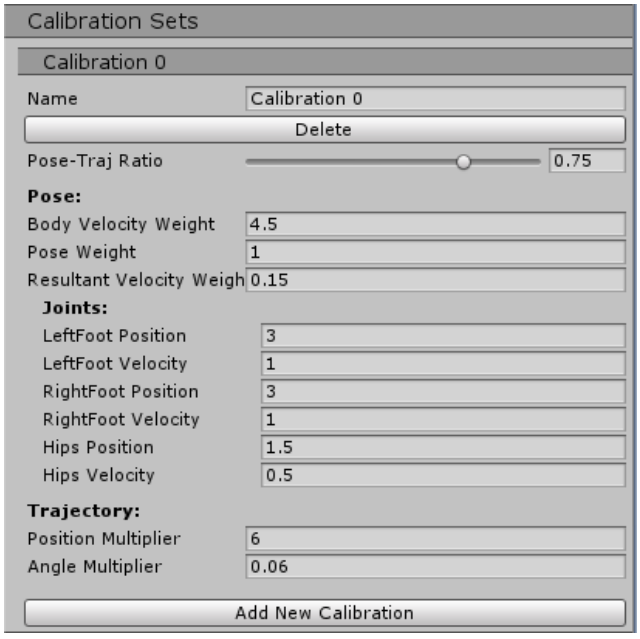

Figure 18 'The MxMAnimData Calibration'

Create Your Prefab

1. Drag and drop the 'Robot Kyle' model into the scene view
2. Attach an Animator component without an Animator Controller
3. Attach an MxMAnimator component
4. Attach an MxMTrajectoryGenerator component
5. Drag and drop you newly created MxMAnimData file into the 'Animation Data' foldout (Figure 19 left)
6. For the MxMAnimator and MxMTrajectoryGenerator ensure the settings look like those shown in Figure 19

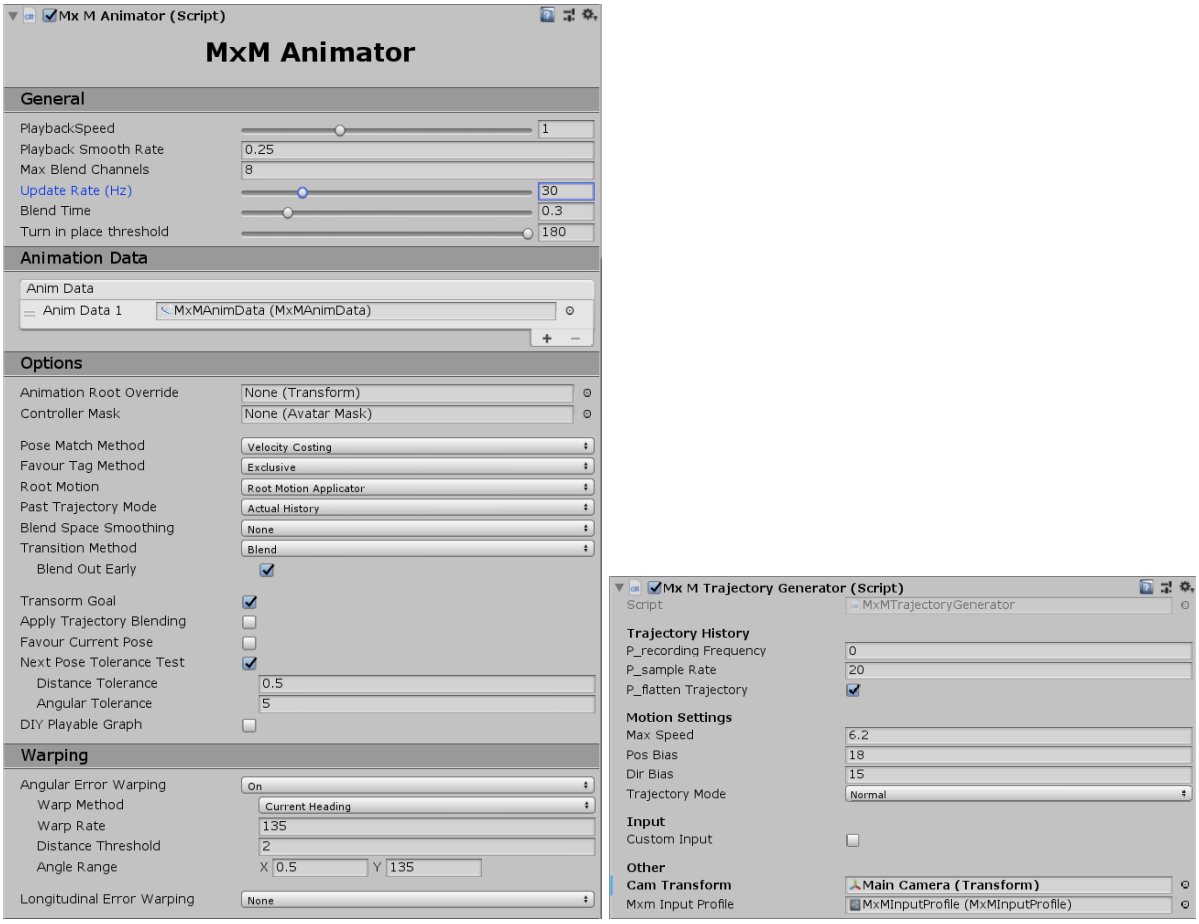

Figure 19 left - 'MxMAnimator settings' right - 'MxMTrajectoryGenerator settings'

The setup is now complete and you can press play to control your Motion Matching robot. Make sure you disable the Robot Kyle that was originally in the scene so you don't get confused.

Note: The other components in the demo scene are not necessary for this quick start to work. However, feel free to copy the structure and AnimData of the Demo Scene Kyle to get an exact replication.

Note: The Kyle in the demo scene has had a lot more work done on it to further improve the animation output. This includes tagging with the timeline editor (not covered in this guide) and setting up events for vaulting. Please see the User Manual and tutorial videos for more details on the more advanced features of MxM.

Conclusion

This quick start guide has instructed you on how to set up your first motion matching animator in Unity. I hope this has shown you just how quick motion matching is to get complex animations. There is of course a lot more to motion matching than just this simple overview so please watch the video tutorials and read the user manual for more detail.