# FPS Hands Horror Pack

Documentation for asset version: 1.0

You can contact us here:
- support@tensori.fi

Thank you for buying this asset! Please contact us if this document or the included demo scenes do not answer your questions or you happen to come across bugs or other problems with this asset. Actionable feedback and Asset Store reviews are appreciated!

You can find a playable sample scene that contains a simple first person player character controller with animated hands & weapons here:
**FPS Hands Horror Pack/Demo/Demoscene**

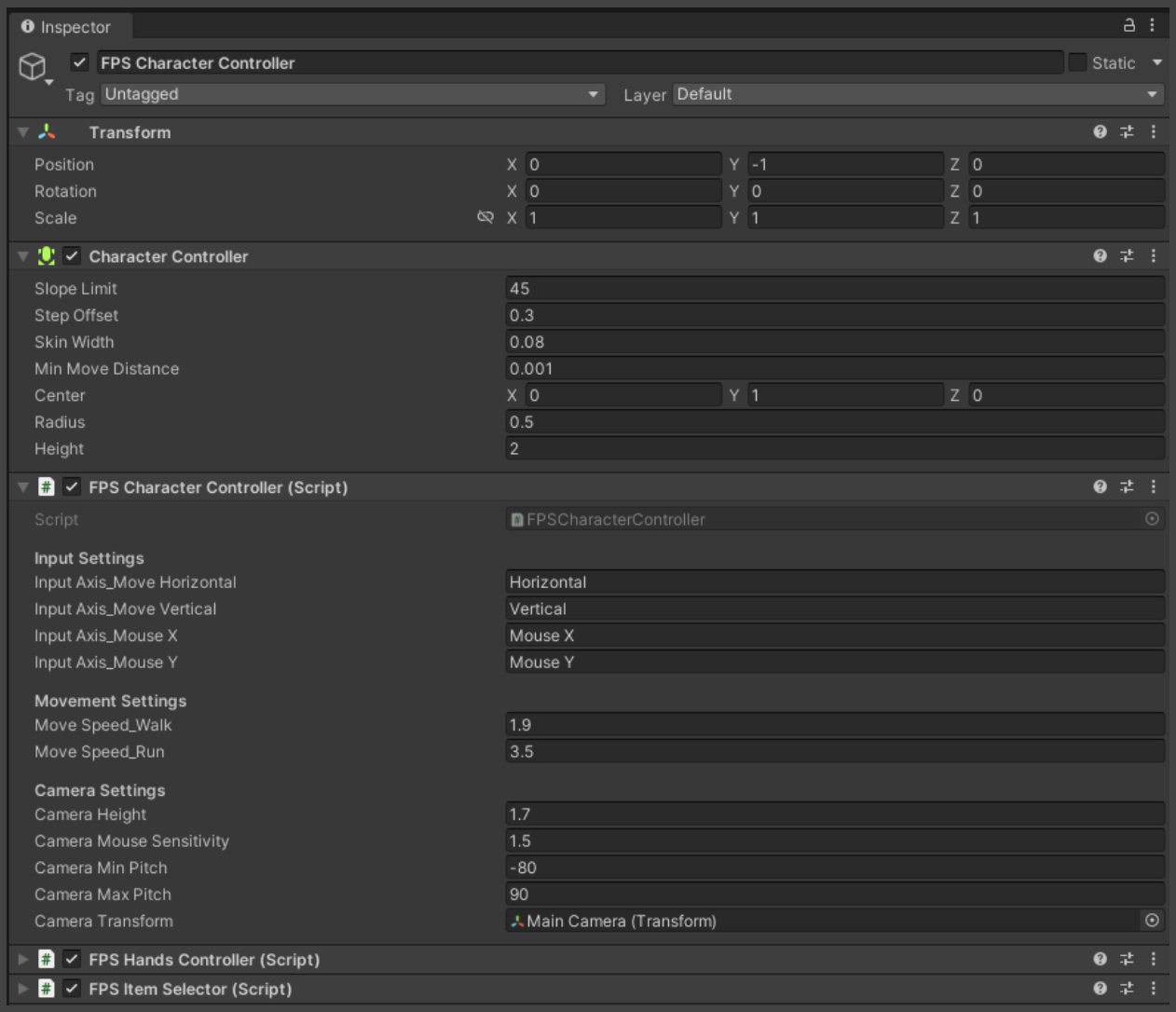
A plain scene with all included modular environment assets can be found here:
**FPS Hands Horror Pack/Demo/Prefab Showroom**

The following pages contain explanations and usage notes for included code systems and setup requirements for the FPS Character Controller & FPS Hands Controller.

**Magenta colored terms:** C# code classes or components
**Orange colored terms:** visible & editable inspector properties

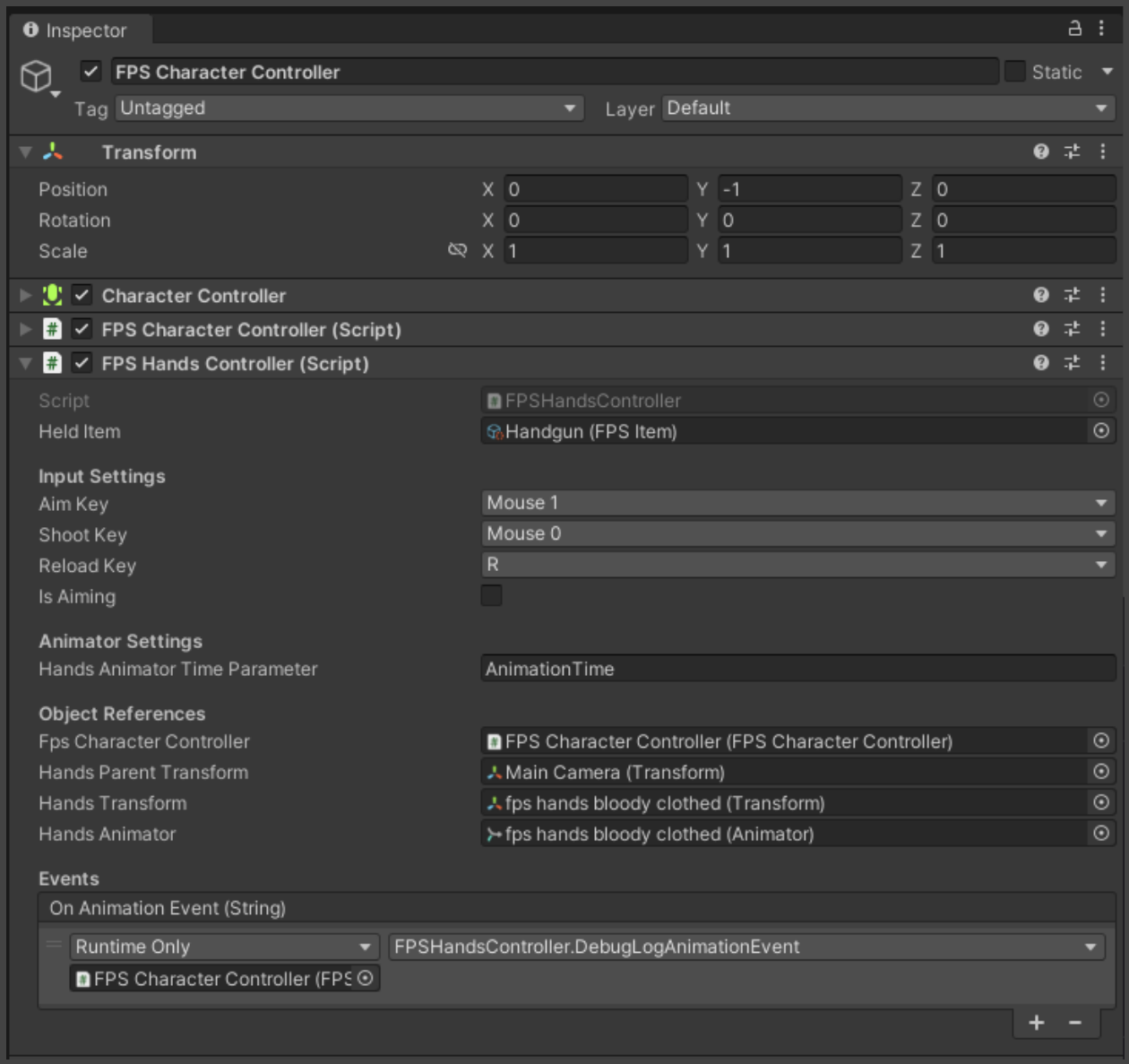
**FPS Character Controller** is a script that handles all character movement and camera rotation logic. The character object movement is implemented using a Character Controller component.

Note: **Camera Transform** should not be parented under any moving game objects. If possible, set it as a scene root object. The code will update its position and rotation.




It is possible to use your own movement / camera logic with the hands & item animations from this pack, but you have to modify **FPS Hands Controller** code to replace the few calls to **FPS Character Controller** with your own implementation.

**FPS Hands Controller** is a script that manages first person hands and all animated items / weapons. By default, the script uses legacy Unity Input Manager API for user input detection. If needed, replacing this input logic with custom code should be fairly straightforward.




**Held Item** is a reference to a scriptable object asset that contains all settings for a single item. An item prefab from this asset is instantiated under the hands transform hierarchy during runtime. Changing the **Held Item** during runtime will destroy the previous item object and instantiate a new, correct one from the new referenced **FPS Item** asset. Please scroll down for more information on **FPS Item** assets.

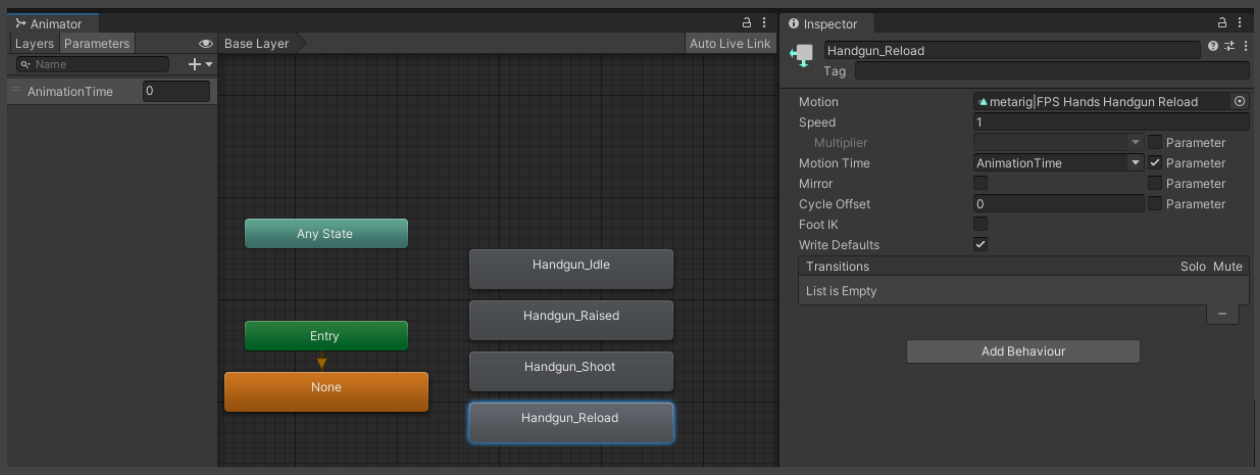
**Hands Transform** is the root transform of the animated first person hands. It is **not** needed to parent this transform manually under the camera transform, for example. **Hands Parent Transform** is the position and rotation target for the hands transform and the code will update them automatically.

**Hands Animator Time Parameter** is the name of the animator parameter (float) used to control the linear position of all animation states from the referenced **Hands Animator** component.

Make sure that you have a float animator parameter in your hands' animator that is used exclusively for controlling the **Motion Time** parameters within animation states. Every animation state that you want to change the duration of easily, needs to have the **Motion Time** parameter controlled by your float animator parameter.

**The same logic should be applied when setting up animations for both hands and items. Both animators are synced and updated via code by using animator parameters & Motion Time feature.**

Check the image below and study the included "FPS Hands AC" and "Weapon AC" animator controllers for a valid setup.

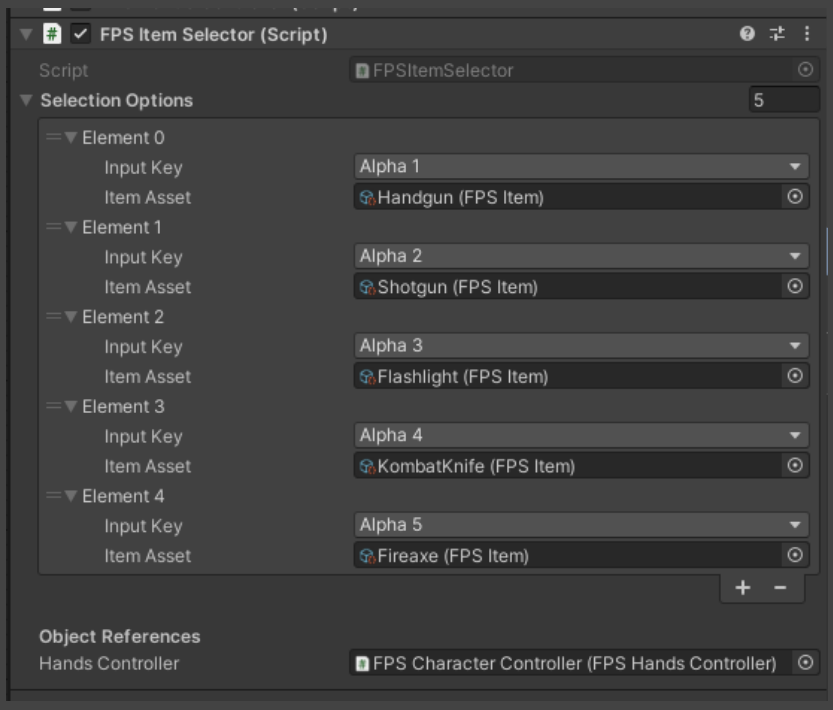


**OnAnimationEvent** is a **UnityEvent** that is invoked every time an **Animation Event** is triggered during attack & reload animations. The event has a string parameter that can be used to bind multiple different timed actions to an animation. See **FPS Item** section for more info on **Animation Events**.
Example use cases:

- Trigger weapon raycast logic at a specific time
- Play visual effects
- Enable follow up attack animations

---

**FPS Item Selector** is a simple script that updates the **Held Item** parameter from the referenced **FPS Hands Controller** script when an input key event is registered.




You can easily connect any custom item selection logic by calling the **SetHeldItem(FPSItem item)** -method from **FPS Hands Controller** component.

**FPS Item** is a scriptable object code asset. It contains all relevant animation settings for a single animated item (held by FPS Hands) and a reference to an item prefab that is spawned during runtime.

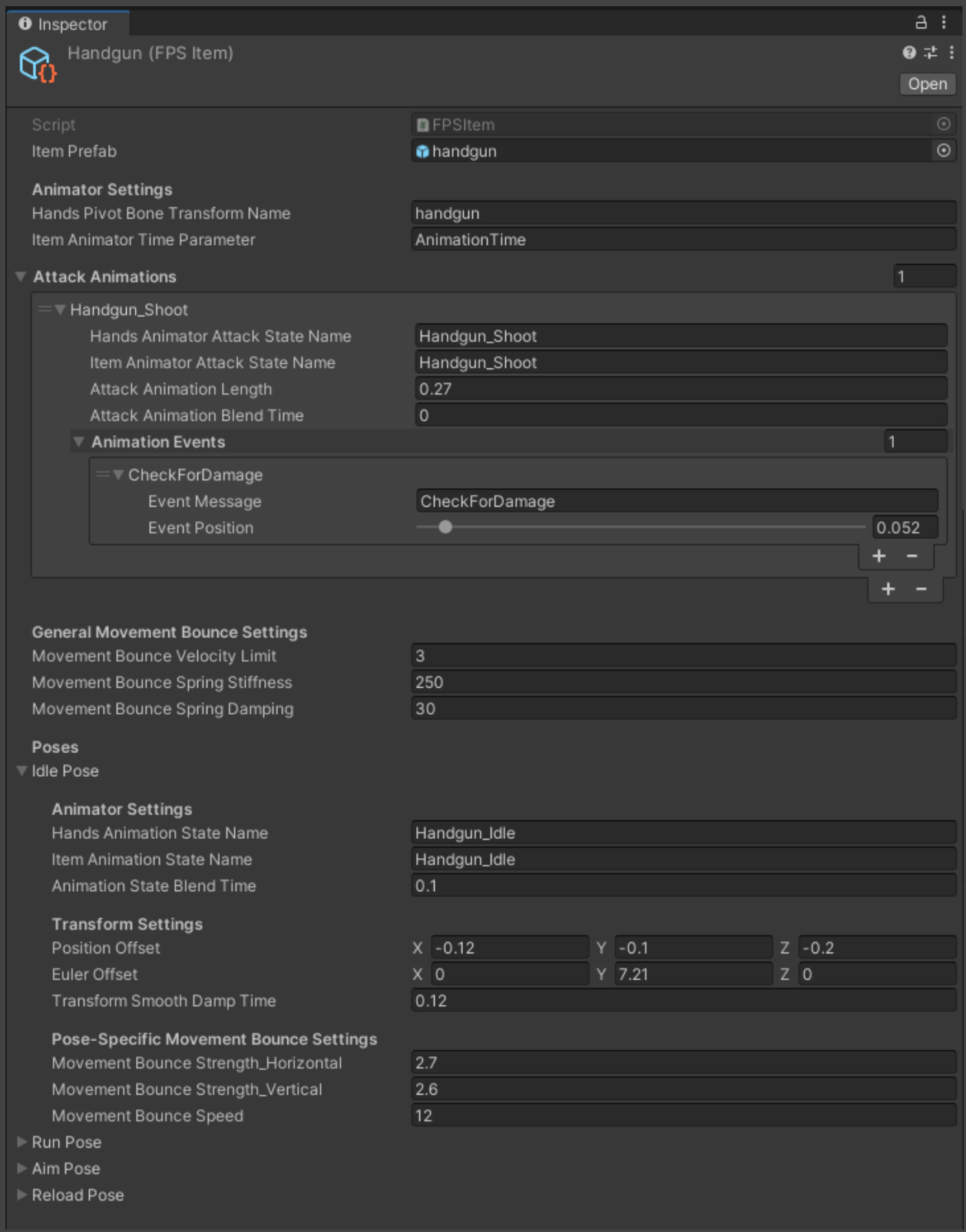
You can create a new **FPS Item** asset by right clicking over the Project-window and navigating the opened menu and selecting the option:
**Create -> Tensori -> FPSHorrorPack -> New FPS Item Asset**

(Scroll down for an image of inspector panel)

**Item Prefab** is the prefab that is instantiated during runtime. The code will search the hands' transform hierarchy for an object that has a matching name with **Hands Pivot Bone Transform Name** parameter. The found child transform will be used as a parent transform for the instantiated item.

In this pack, there is a separate bone transform within the hands' armature for each item, but generally using any other bone transform as a target parent is perfectly valid as long as you use that specific bone when creating animations.

**Item Animator Time Parameter** is the name of the animator parameter (float) used to control the linear position of all animation states from the instantiated item's animator component if one is available.


Inspector
Handgun (FPS Item)
Open
Script
FPSItem
Item Prefab
handgun
Animator Settings
Hands Pivot Bone Transform Name
handgun
Item Animator Time Parameter
AnimationTime
Attack Animations
1
Handgun_Shoot
Hands Animator Attack State Name
Handgun_Shoot
Item Animator Attack State Name
Handgun_Shoot
Attack Animation Length
0.27
Attack Animation Blend Time
0
Animation Events
1
CheckForDamage
Event Message
CheckForDamage
Event Position
0.052
General Movement Bounce Settings
Movement Bounce Velocity Limit
3
Movement Bounce Spring Stiffness
250
Movement Bounce Spring Damping
30
Poses
Idle Pose
Animator Settings
Hands Animation State Name
Handgun_Idle
Item Animation State Name
Handgun_Idle
Animation State Blend Time
0.1
Transform Settings
Position Offset
X -0.12
Y -0.1
Z -0.2
Euler Offset
X 0
Y 7.21
Z 0
Transform Smooth Damp Time
0.12
Pose-Specific Movement Bounce Settings
Movement Bounce Strength_Horizontal
2.7
Movement Bounce Strength_Vertical
2.6
Movement Bounce Speed
12
Run Pose
Aim Pose
Reload Pose

**Attack Animations** is a list of animation settings (**FPSItem.AttackAnimationSettings**) for all available attack animations.
If there are more than one available attack animations, the animations will play in a sequence whenever the user presses an attack input while an attack animation is already playing. The sequence will restart from the beginning if an attack input is pressed when the last listed attack animation is playing.

- **Hands Animator Attack State Name:** name of the animation state in hands' animator component

- **Item Animator Attack State Name:** name of the animation state in item's animator component

- **Attack Animation Length:** duration of the animation in seconds (for both hands and the item) - controlled by updating the animation states' **Motion Time** parameters

- **Attack Animation Blend Time**: duration of animator crossfade to this animation state in seconds (for both hands and the item)

- **Animation Events** are timed messages that are triggered during animation playback. You can subscribe to **FPS Hands Controller's OnAnimationEvent** in code or manually from the inspector view. **Event Position** is a normalized time during the animation timeline when the event gets triggered.
  **0.0f = start of animation, 1.0f = end of animation.**

**General Movement Bounce Settings** are a set of parameters that are used to generate wave-like swaying motion for the hands & item when the player character moves. A spring formula with simple trigonometric functions is used to generate this motion.
Please scroll down to **Poses** section for pose-specific options.

- **Movement Bounce Velocity Limit** is the maximum magnitude and the effective limit of the player's movement velocity that is used within the movement bounce code calculations.
  Note: this settings will **not** interfere with the player's movement velocity in any way - it is used for internal calculations only.

- **Movement Bounce Spring Stiffness** controls the strength or the "sharpness" of the simulated movement bounce spring.

- **Movement Bounce Spring Damping** controls the stopping force or the "smoothness" of the simulated movement bounce spring.

**Pose (FPSItem.ItemPose)** is a set of animation settings for a single action or a stance. **FPS Hands Controller** validates and selects the used pose every frame and uses it to animate the hands and item.

- **Hands Animation State Name:** name of the animation state in hands' animator component

- **Item Animation State Name:** name of the animation state in item's animator component

- **Transform Settings** control the hands' transform position and rotation offsets in this pose. All calculations behave like local space transformations. For example: changing the **Position Offset** X-axis will move the hands horizontally sideways and the Y-axis will change the vertical position in screen space.
  Note: Units are in Unity's world space.

- **Pose-Specific Movement Bounce Settings** affect the movement bounce motion in this pose. The strength parameters control the magnitude of the motion. X- and Y-axis settings are separated.

- Reload pose has additional animation settings. **Animation Length** and **Animation Events** work identically as in **Attack Animations**. See the section on **Attack Animations** for more details on these options.