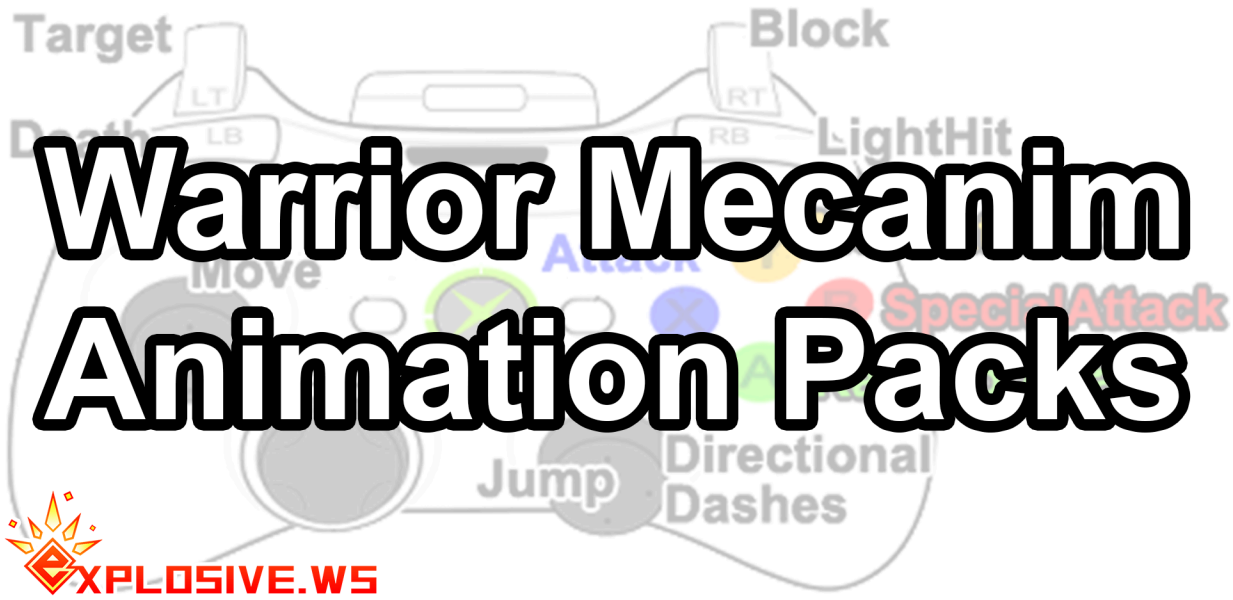


Warrior Mecanim Animation Packs

Warrior Mecanim Animation Pack ReadMe

Last Updated: **Jan 07, 2024**

It is highly recommended to watch Unity's Animation Tutorial Videos before using this asset if you're not familiar with Mecanim: **http://unity3d.com/learn/tutorials/topics/animation**

Controller Overview

The Warrior Mecanim Animation Pack controller includes several Unity components which control the character's position in the world, and its Mecanim animator.

Components

- `WarriorController` - This is the main controller which every other component references.
- `WarriorMovementController` - This component takes care of physics and drives the character in various movement states using SuperCharacterController.
- `SuperCharacterController` - This component handles ground detection and surface interaction.
- `WarriorTiming` - This component contains timing for locking the Warrior's movement and action during animation, and also timing for attack chaining windows for button presses.
- `AnimatorParentMove` - Enables Root Motion in the animation to drive the Warrior movement. It is automatically attached to the game object with the Animator component at runtime.
- `WarriorAnimatorEvents` - This component contains placeholder methods for animation events triggered by the animator. It is automatically attached to the game object with the Animator component at runtime.
- `WarriorData` - This component contains enum data for the Warrior's type, state, and animation triggers.

Demo Components

These scripts are optional.

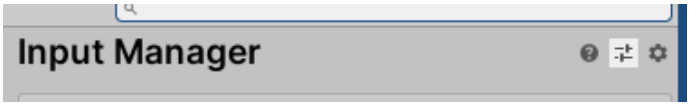
- `WarriorInputController` - This is a simple implementation of gamepad input and can be used wholesale or as a reference for defining your own input scheme.
- `WarriorInputSystemController` - Alternative InputSystem control scheme for using Unity's new InputSystem. Contained within the InputSystem Support package in the project folder, and requires the InputSystem Package to be installed via the Package Manager.
- `GUIControls` - This is an example for the demo scene and not intended for your game, but it is a great reference for how to trigger nearly any action.
- `IKHands` - Applies IK to the left hand for 2 handed weapons to keep the hand properly placed if differences in character proportion between rig and retargeted model are too great that the hand isn't properly positioned where it needs to be.

All scripts in the Warrior Mecanim Animation Pack use the namespace: `WarriorAnims`

Setup

Pre-Installation

Before attempting to use the pack, you must first ensure that the tags and inputs are correctly defined. There is an included **InputManager.preset** and **TagManager.preset which** contains all the settings that you can load in: **https://docs.unity3d.com/Manual/Presets.html**

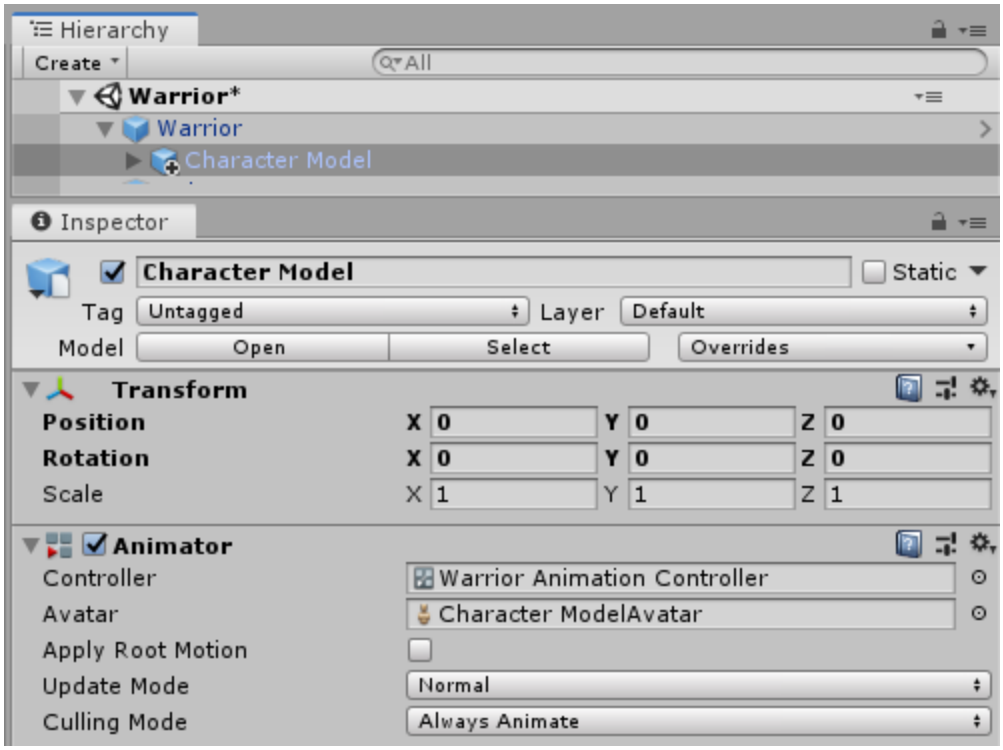


The required tags and inputs are as follows:

Replace Character Model

Simply drag in your character model underneath the main Warrior prefab, and then set the Controller property of the Animator component to the Warrior Animator in the prefab folder.

The `WarriorController` script will find the Animator at runtime and automatically switch the correct settings for the Animator, and also attach the `WarriorAnimatorEvents` script and `AnimatorParentMove` script.



Set Warrior Type

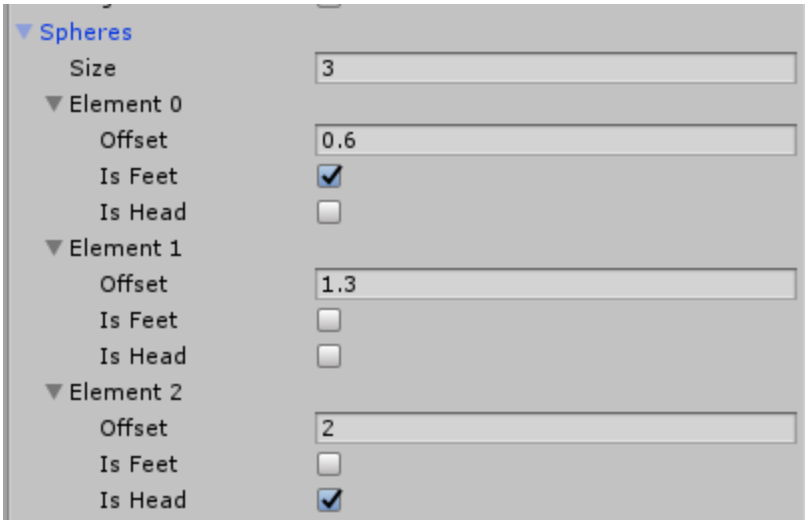
In the `WarriorController`, set the type of Warrior you're using. This will dictate which animations will play, and also control the timings for those animations for locking character movement while playing.

Set Target

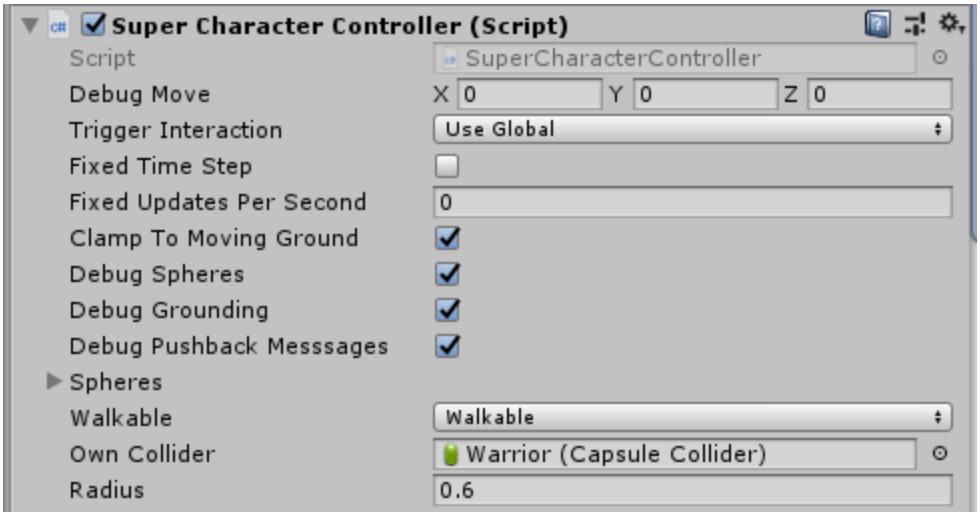
The `WarriorController` script needs a target object for purposes of targeting/strafing.

Adjust Collider and Super Character Controller Spheres

If needed, adjust the Capsule Collider for your character, and also adjust the Super Character Controller Sphere's objects to proper size and position.

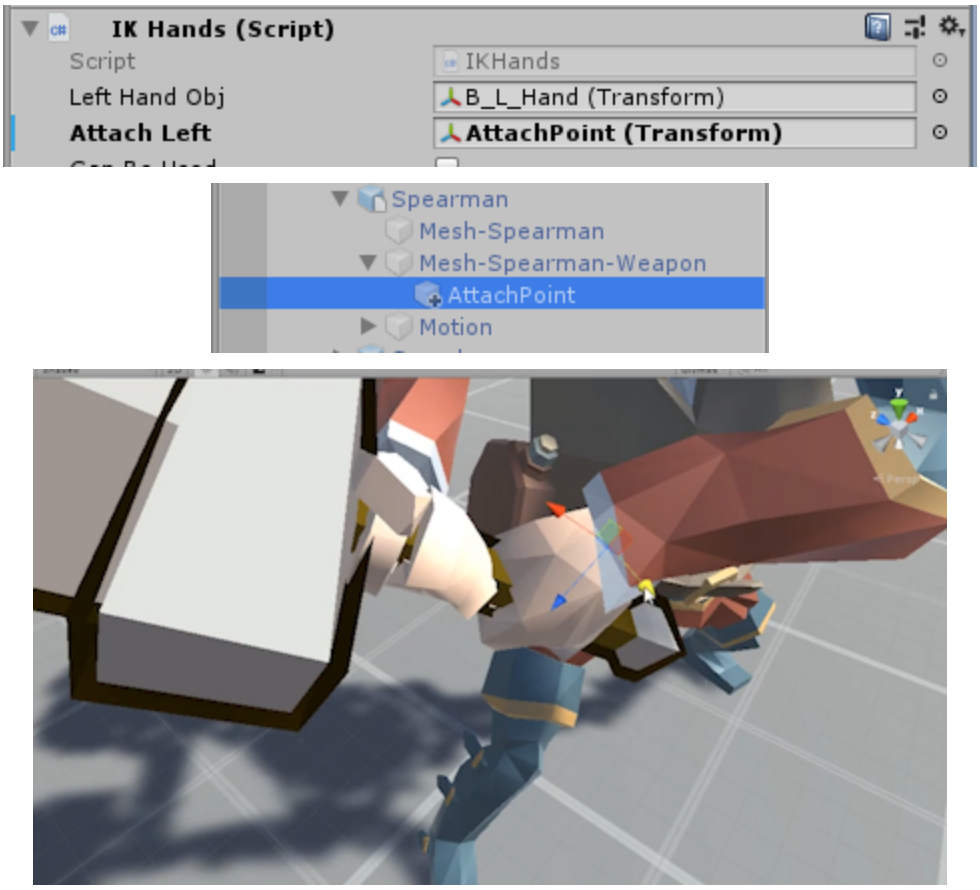


Set Super Character Controller Script Walkable Layer and Own Collider.



Set IK hand

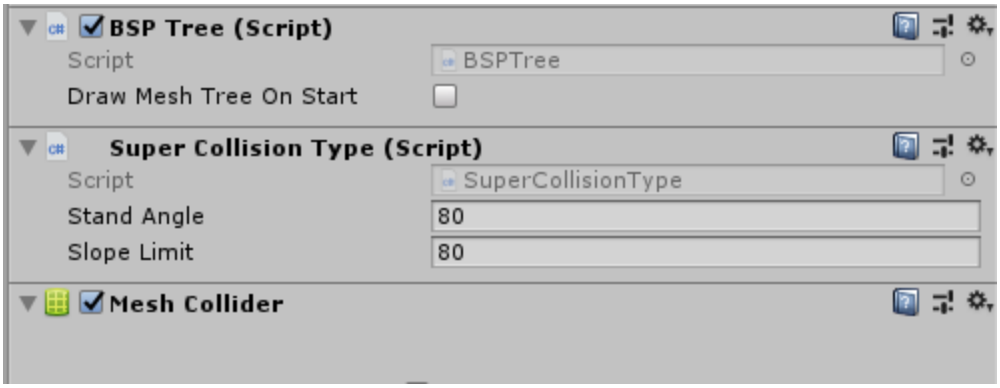
If you're using IKHands script then you need to set your character's left hand joint as the script's Left Hand Obj and also add an empty gameObject to your character's right hand which is connected to the Attach Left field.



You may need to position the AttachPoint gameObject in runtime when you see the hand on the weapon to get it just right, and then you can copy the Transform and paste the values back again after stopping the game.

Setup World Colliders

For any objects that you want the character to walk over or collide with, set them as Layer: Walkable, and make sure this is set the same in the `SuperCharacterController` script. Objects with primitive colliders on them such as sphere, box, or capsule colliders won't need any additional settings, but any object with a *Mesh Collider* needs the `BSPTree` script attached to it. If you want to control the allowable slope height for the object, you can attach the `SuperCollisionType` script to it as well.



Animation Events

Note that there are animation events for many of the animations. If you're using the WarriorController script, it will automatically attach the WarriorAnimatorEvents script to the gameobject in your character's hierarchy which contains the Animator component. The WarriorAnimatorEvents script is used to trigger weapon visibility when Sheathing/Unsheathing weapons, and also has blank Methods if you wish to trigger sounds, effects, and/or other code upon impact, etc.

Animator Parameters

Moving: Set in `WarriorMovementController` if there's movement input and character motion.
Targeting: Set in `WarriorController` if targeting.
Stunned: Set in `WarriorController` if targeting. If true, GetHit transitions to Stunned animation.
Blocking: Set in `WarriorController` if pressing block.
Animation Speed: Global adjustment for all animations, set in `WarriorController`
Weapons: Set in `WarriorController`, same as WeaponSwitch function.
Jumping: Set in `WarriorMovementController`, 0 grounded, 1 jump, 2 falling, 3 double jump.
Velocity X: Set in `WarriorMovementController`, character's sideways speed.
Velocity Z: Set in `WarriorMovementController`, character's forward/backward speed.
Action: Set in `WarriorController`. This is used by the various animation triggers to determine which animation to play.
Trigger Number: Set in `WarriorController`. This determines which node is triggered via the "Trigger" trigger.

Questions, comments, requests or suggestions:

Contact.