❓ How does it work?

In this section we will learn how the system achieves such a nice blending.

Background

Usually, Unity developers use techniques such as *Layering* and *Additive Animation* to blend motions on the fly. However, this does not always work great, because animations are always applied in local space.

Tip: local space means relative to the parent transform.

Example: if we override the character sprinting animation with a static upper body pose, it will look horrible - the body will be moving around as if it was attached to the pelvis.

Imagine how complex your animator controller will be with all the different control parameters and animations. This big and convoluted structure will be difficult to maintain, and most importantly scale in the future.

Overview

Magic Blend takes a different approach - it operates in **character root bone space**. Why does it matter? As mentioned above, the local space rotation overrides do not always look good. While it will work for fingers and arms, it does not work for the lower body or spine.

Tip: root bone space means we will apply rotation and translation relative to the character root bone (Root or Armature).

On top of that, **Magic Blend** offers an easy way to adjust sliders on the fly and separate your poses from the animator controller. That's where the scalability kicks in - want to add a new weapon or item? Sure, no need to modify your animator controller, create a **Magic Blend Asset** and you are good to go.

Not even to mention the smooth blending between different poses - Mecanim Animator does not provide anything compared to that.

On the low level

The **Magic Blend** updates your character animation pose via these 3 main stages:

1. Copy base animation pose. Such a pose is usually an idle, used as a reference frame to compute dynamic additive data.
2. Copy overlay animation pose. This is your weapon or item pose. The system will copy its transforms in global space to blend in the final step.
3. Compute the dynamic additive between the locomotion pose (coming from your animator) and the base pose from the first step. Then, apply that additive result to the overlay pose. Finally, blend between additive overlay and current animator pose.

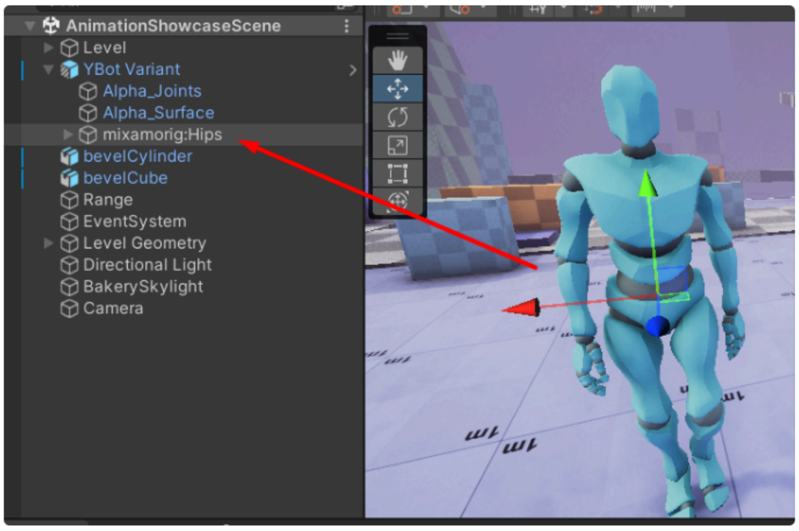
The system is powered by *Animation Jobs*, so it stays performant even when there are a lot of characters running around in the scene.

Tutorial

In this section we will learn how to use Magic Blending.

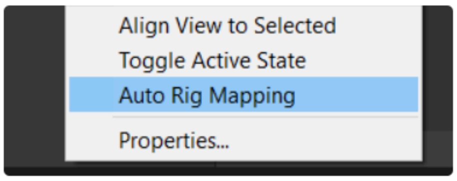
Create Rig Asset

First, we need to add our character to the scene and select its root bone:



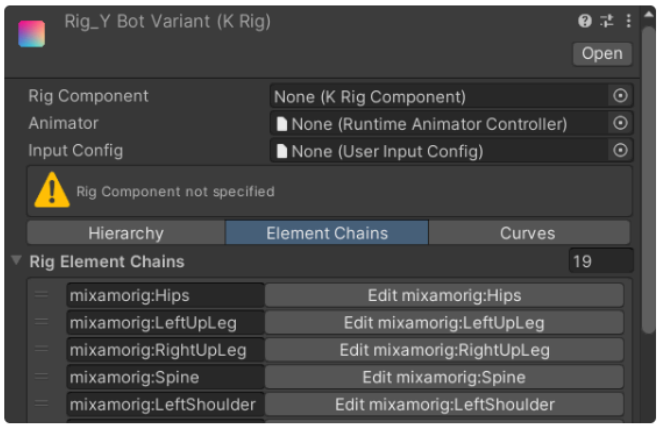
Root bone is the top transform in the hierarchy.

Now right-click and select **Auto Rig Mapping**:



Click that button.

After this, a **Rig Asset** will be generated in the active folder:



Rig_Y Bot Variant (K Rig)

Open

Rig Component None (K Rig Component)

Animator None (Runtime Animator Controller)

Input Config None (User Input Config)

Rig Component not specified

Hierarchy | Element Chains | Curves

Rig Element Chains 19

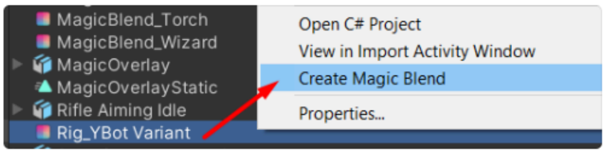
mixamorig:Hips	Edit mixamorig:Hips
mixamorig:LeftUpLeg	Edit mixamorig:LeftUpLeg
mixamorig:RightUpLeg	Edit mixamorig:RightUpLeg
mixamorig:Spine	Edit mixamorig:Spine
mixamorig:LeftShoulder	Edit mixamorig:LeftShoulder

Rig Asset contains information about your character skeleton.

The **Rig Asset** plays an important role - it stores the information about the character's hierarchy as a Scriptable Object.

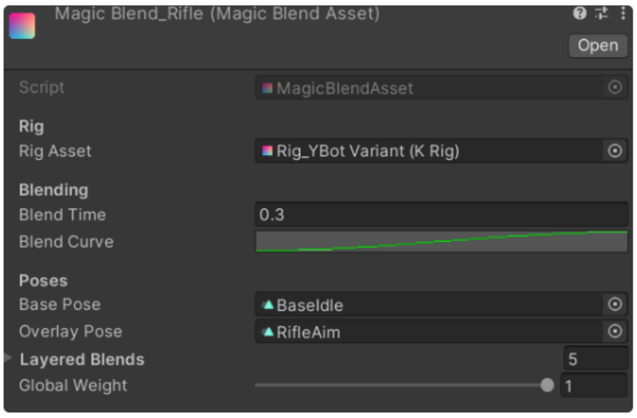
Create Magic Blend Asset

Right-click on your **Rig Asset** and select **Create Magic Blend**:



Hit that button.

Now, a **Magic Blend Asset** has been generated in the active folder. Let's see what properties it has:


Magic Blend_Rifle (Magic Blend Asset)

Open

Script MagicBlendAsset

Rig

Rig Asset Rig_YBot Variant (K Rig)

Blending

Blend Time 0.3

Blend Curve

Poses

Base Pose BaseIdle

Overlay Pose RifleAim

Layered Blends 5

Global Weight 1

Magic Blend Asset.

Blend Time defines the time in seconds to blend this asset in.

Blend Curve defines how the blending will be handled.

Base Pose is an Animation Clip that defines the idle pose. This pose will be "subtracted" from the animator controller pose to compute additive animation dynamically.

Overlay Pose is your weapon or item Animation Clip pose.

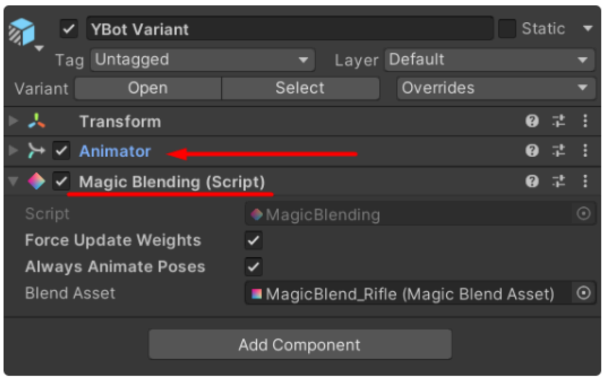
Layered Blends define the layers that will be blended in real-time. By default, the system generates:

- Lower Body
- Spine
- Head
- Arms
- Fingers

Global Weight controls the universal influence of the system. If set to zero - nothing will be applied, where 1 means the system will be fully active.

Magic Blending Component

Go to your character prefab and add a **Magic Blending** component. Make sure you add it to the same Game Object, to which the Animator is attached:



Script: MagicBlending
Force Update Weights: ✓
Always Animate Poses: ✓
Blend Asset: MagicBlend_Rifle (Magic Blend Asset)

Magic Blending.

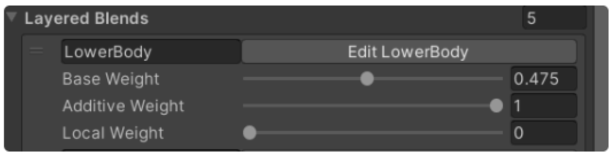
All you have to do is assign the **Magic Blend Asset** in this component. Now you can start the game and play around with the values in the **Magic Blend Asset**!

⭐ Workflow

In this section we will cover the features of Magic Blending.

Blend Sliders

The **Magic Blend Asset** offers a very convenient way to adjust blending parameters on the fly:



Layered Blends 5
LowerBody — Edit LowerBody
Base Weight 0.475
Additive Weight 1
Local Weight 0

Weight sliders.

Base Weight controls a blend between Animator and Magic Blend pose. If set to zero, the Animator will be fully used, if set to 1 - the overlay pose will fully override this bone chain.

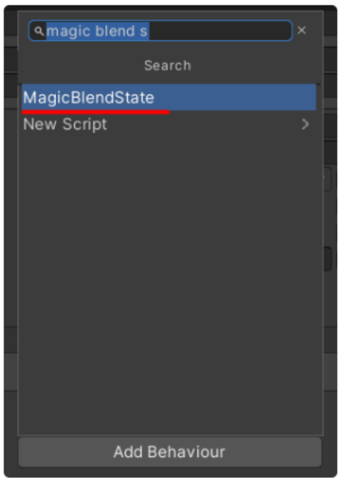
Additive Weight controls the dynamic additive influence. If set to zero, no additive is applied, if set to 1 - it will be added to the overlay pose.

Local Weight overrides everything with the overlay pose. It is only useful for fingers or hands. If set to zero, it will not be applied, and if set to one - it will fully override the bone chain with the overlay pose.

There are no restrictions on what sliders to use and how. You need to tweak the settings and see what works best for your character pose.

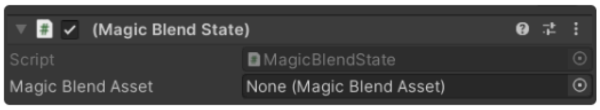
Animation States

It is possible to bind **Magic Blend Assets** to a specific animation state in your controller. To do so, click on the desired state and add a **Magic Blend State**:



Blend State component.

Now, assign the blend asset, and the system will automatically sync it with the rest of the animator controller:



Script: MagicBlendState
Magic Blend Asset: None (Magic Blend Asset)

Choose your asset here.

Tip: you can add multiple empty states to your animator, and set up transitions between them. You don't have to specify the animation clips, as they will be supplied from the selected Magic Blend Asset.

This is a very simple way to integrate the **Magic Blending** with your project without modifying the code. However, sometimes we run into situations when it is vital to change a **Magic Blend Asset** in runtime.

Code Integration

You can swap **Magic Blend Asset** by simply replacing the asset in the inspector. If you want to do a similar thing in the code, here is how you can achieve that:

YourComponent.cs

```
private MagicBlending _magicBlending;

private void Start()
{
    _magicBlending = GetComponent<MagicBlending>();
}

private void ChangeMagicBlendAsset(MagicBlendAsset newAsset)
{
    _magicBlending.UpdateMagicBlendAsset(newAsset, true);
}
```

That is it! Now you know how to fully use **Magic Blending** in your project.