FIMPOSSIBLE CREATIONS

# LEGS ANIMATOR

## USER MANUAL

## About Legs Animator

- Legs Animator is a multi-purpose plugin which uses Inverse Kinematics (IK) to simulate **different leg animation behaviors**.

- Plugin offers **many ways to configure** legs animating setup in order to work with different types of rig and in order to create different leg animations. You can make it work on humanoids, insects, animals or any other type of creature.

- Legs Animator component is providing a **highly customized inspector window** (GUI) to help use it without confusion since there are a **lot of parameters to play with**.

- Package is providing many example scenes presenting different features which can be **unpacked to project with** "Demo - Legs Animator" unitypackage file.

- Legs Animator offers methods for custom usage through code. These methods are starting with "User_" like "**User_SetIsGrounded**".

- You can combine Legs Animator with my other packages, like Tail/Spine/Look, Animator. Check the manual pages for more details about it.

*Contact and other links you will find in Readme.txt file*

# Index



**Tutorials Playlist on Youtube**

# 1: Getting Started

## Demo Scenes:

You can find demo scenes with many useful examples under:
“FImpossible Creations/**Plugins - Animating**/Legs Animator/Demo - Legs Animator/”
**after unpacking the “Demo - Legs Animator.unitypackage” file**.

## Base Requirements:

In order to make Legs Animator work with your character model, there are few simple requirements:

**One**: If you use a rigged model: Disable “Optimize Game Objects” in your model file.
(This option is disabled by default when importing models to unity, so in most cases you don’t need to check it)
It’s in the inspector window, “Rig” bookmark.
(with “Optimize Game Objects”, doing procedural animations is impossible)

**Two**: Your character model skeleton needs to be created in a standard-parented way. There are very rare rigs, where all skeleton bones are detached.
(99.99% models are using standard parented structure)
The Hips bone needs to be a parent of leg bones.

**Three**: Recommended for your model to face the Z-Forward axis (unity standard, asset store models require it, in order to be visible on the store), to avoid problems with helper algorithms. You can always create additional transform and adjust your model rotation within local space.

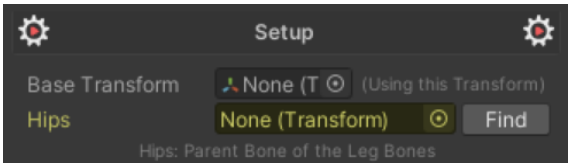
**Four**: Keep in mind that different models will require individual tweaking, of different Legs Animator’s parameters, to get best out of it. Adding component -> Assigning Bones -> Run -> will not give you immediately, beautiful and perfect animation.

### Plugin is computing leg animation fast but…

But you still need to be careful. Computing one leg cost is similar to one Unity’s CharacterController movement cost. If you set up a spider-like creature (8 legs), the cost is 8 times bigger! So you see that using a legs animator on two-legged creatures is much less expensive. There is potential for DOTS implementation in the future, so there may be big performance boost update in the future.

## Preparing the Component:

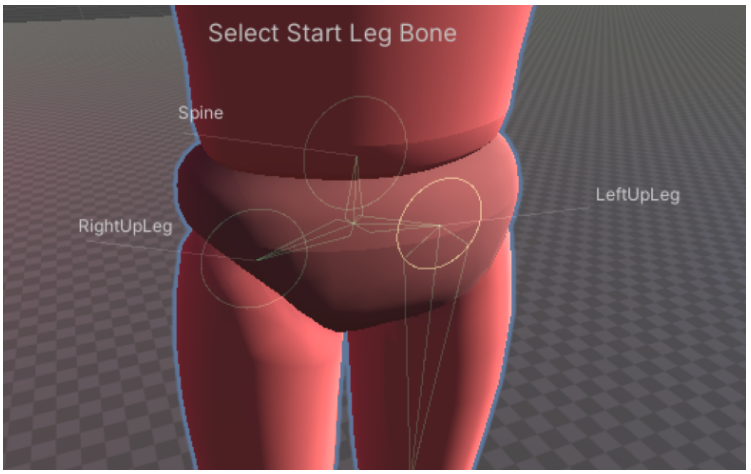
The component you will use is called **Legs Animator**.
Select your character object on the scene, hit “Add Component” and go to “Fimpossible Creations -> **Legs Animator**” or write “Legs Animator” in the search prompt and select it.




After adding component to your character, Legs Animator's initial requirement is **defining hips bone**, which needs to be **a parent of the leg bones**.
Open your character hierarchy, find skeleton structure and drag & drop hips bone into the legs animator “Hips” field. You can use the “Find” button, but always verify if the right bone was selected after hitting the button.

**The next thing to do is setting up legs.**
Hit the “+ Add Leg” button to add a field for one of the legs.
Hit its number button to select it and display more options + display scene gizmos which can help you setting up the leg.

Scene view will display bones which are child of the hips to select to be the upper leg. Then you will see more icons in all found child bones of the upper leg to define which one is the lower leg and foot.



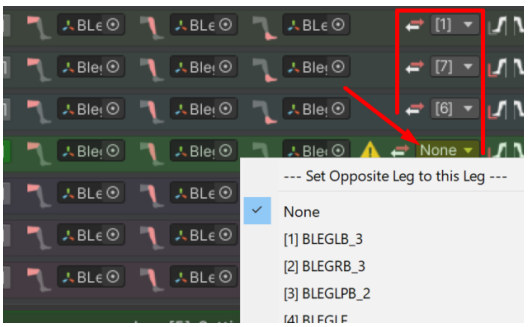

You can also try using automatic legs detection with the button on the bottom, **but always verify the result of this action**.

When setting up creatures like spiders, with many legs, after setting up one leg, **you can speed up the process**.

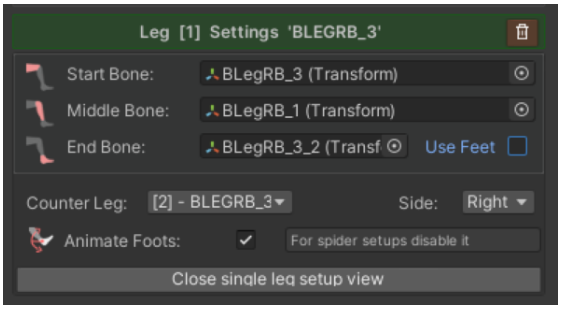

You can use the current leg setup as a scheme for other legs.
Use the button to mimic selected leg structure on other leg start bones in hips or manually drag & drop start leg bones into the “Leg Top Bone” field.

After setting up leg bones, it’s important to **define leg side**: if it’s left or right leg. Sometimes the plugin's algorithm will detect it automatically.

Another really important thing is defining opposite legs. It is used by the animating algorithm gluing feature, to prevent raising two legs in the same moment.




With one of the legs selected, you will see more options down below.




Use  button to remove the leg, if you added it by mistake. You can enable experimental feet animating, to raise heel up instead of move whole foot up. You can switch off Animate Foots if your character doesn't have feet bones - like insects. (this field is visible in Motion/Main tab too)

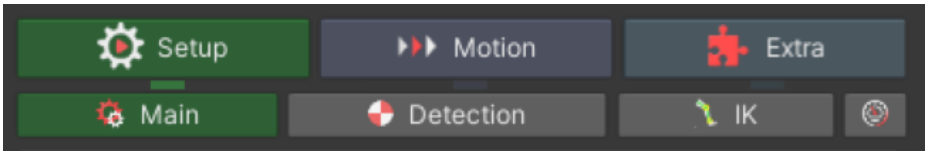
More IK specific settings per leg can be found in other sub-bookmark of the Setup bookmark.

**Scale reference** value if very important. Most of the algorithm's intensity factors rely on this value. Basically it's a reference to the average size of your character. Try setting it up to be half of the side of your character.

After setting up leg bones you will notice the mentioned bookmarks on the top of the inspector window.

## Navigating in the inspector window:

The Inspector window of Legs Animator is divided into few bookmarks and sub-bookmarks. Main bookmarks are **Setup**, **Motion** and **Extra**.

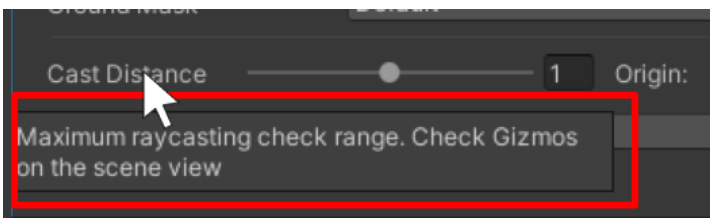



Under **Setup** , you will find settings for the base behavior of the Legs Animator algorithms. Define legs structure, leg settings, physics settings, IK settings and few Optimization options.
After setting up all things, you will rarely go back to this bookmark.

Under **Motion** , you will adjust many animation specific parameters to make it look best on the character or creature you apply legs animator on. You will also add custom behavior modules in this bookmark.

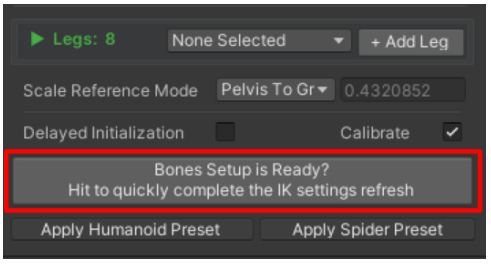
The **Extra** bookmark gives a few more useful settings to play with. There are important references like unity Animator reference or Rigidbody reference which can be used to ease up use of Legs Animator without much coding.

You can always help yourself by **checking the tooltips** on the parameter fields. Most of the parameters are documented. (tooltips are not displayed during playmode)



## Further Setup :

Now, when the legs are setted up, you need to refresh IK foot parameters, which you can do by pressing the button:



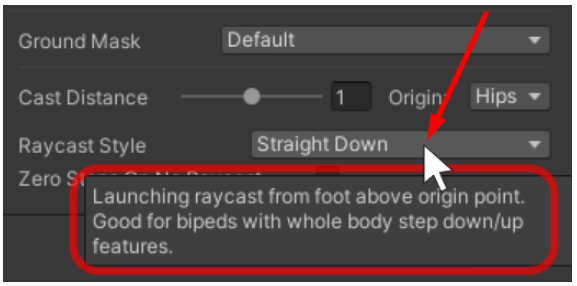

Or if you don't see it, **you can go to IK sub-bookmark and hit the button**, on the right to the "IK Leg Details". It will sync feet orientation and ankle height settings.

(Check tooltip for "Delayed Initialization" description.)

Calibrate option is turned on just in case some of the model bones are not animated. If you turn off "Calibrate" and your bones are not animated, it will result in a bone twisting glitch. Turning off "Calibrate" if not needed, can give you a small boost in performance.

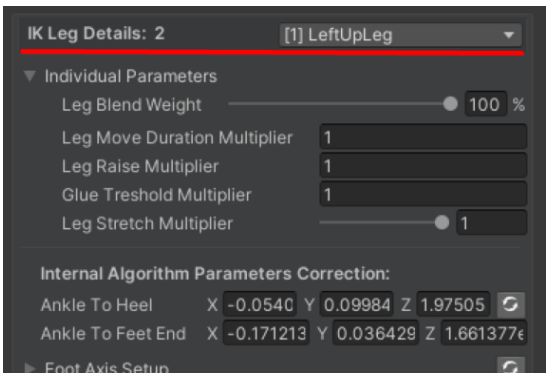
Under **Detection** sub-bookmark, you will define ground under foots detection.

First, define colliders layers mask to detect raycast hits on. Then adjust position to start raycast from, follow the scene gizmos for details.
Enter with a mouse cursor on the parameters and selected options to display description.




You can skip using raycasting and animate legs, just on a flat plane with "Raycast Style" set to "No Raycasting". "Zero Steps On No Raycast" will do the same but when no ground is detected.

In **IK** sub-bookmark, you can adjust IK algorithm specific settings like knees hint computing. (check different ones to find one best fitting to your model, "Default" and "Leg" are best)
Max Stretching will limit how straightened can get the leg + it will adjust hips towards stretched limb when using hips stretch preventer settings.




IK Leg Details gives you the possibility to modify extra animating settings per single leg, which can be very useful for creatures with multiple legs.

Randomizing these settings will give you a more detailed leg animation style.

The “Internal Algorithm Parameters Correction” is automatically computed based on the editor pose of the skeleton. If you encounter some errors like twisted feet on ground, you can try accessing these values to correct it.

In the **Optimization**  sub-bookmark you will find parameters for easily disable Legs Animator component on camera distance or visibility in cameras (including scene view camera) to save performance when character is not displayed.

In the future versions there will be implemented LOD settings for smarter performance adjustments.

# 2: Working With Legs Animator

Legs Animator is offering different behaviors for leg animating.
You can selectively enable different features.
You can use leg ik step adjust on uneven terrain, use auto leg animating basing on automatic attachment points with or without raycasting, or you can use legs animator just for 360 automatic movement animating etc.

Let's go through **Motion**'s  sub-bookmarks one by one.


## Motion / **Main**  :

There you will find "**Legs Animator Blend**" parameter to fade on/off whole legs algorithm influence on the character model.

The rest of the settings are related to uneven terrain foot step aligning.
If you disable raycasting, these settings will not be used.

"**Animate Foots**" is exactly the same parameter you saw in the Setup category.
If you enable it, you will have the possibility to **limit foot's Yaw** (Y axis) rotation, which effect can be noticed mostly, when using gluing and during rotating characters in place. We will talk about gluing later.

The next parameters are helpers for step alignment motion.
"**Smooth Sudden Steps**" will prevent instant change of step height position for the leg, which you can observe in the S_DEMO_LegsAnimator_SmoothSuddenSteps demo scene.
"**Leg Elevate**" will raise the leg over the ground if in the current frame of played animation, the leg is not touching the ground and the ground is just below foot.
"**Foot Rotation Blend**" if controlling foot alignment on steep slope rotation for the foot bone.
"**Foot Rotation Rapidity**" controls how quickly the rotation change should be animated.

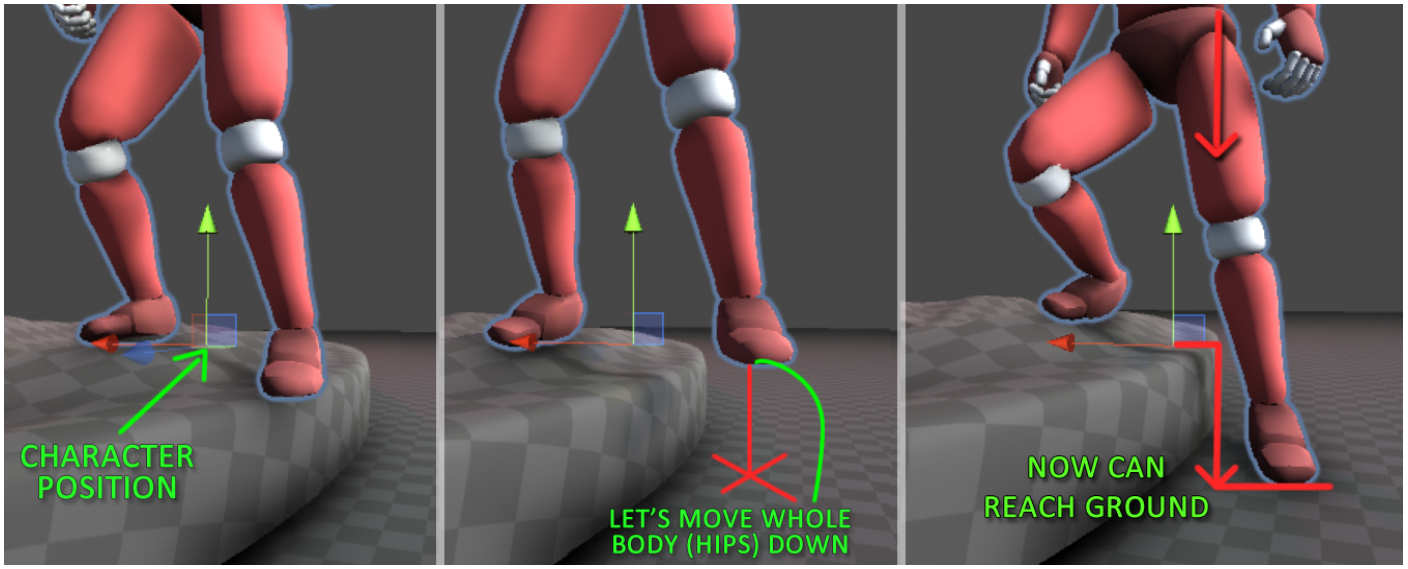
**Step Points Overlap Radius** is a simple implementation of push-out collision between feet. Check scene gizmos to identify push out radius.

# Motion / **Hips** :

This sub-bookmark divides into 3 more categories:


Motion / Hips / **Body Adjust** :

There you can configure whole body adjustment when raycasting is detecting ground not so far below under the feet.

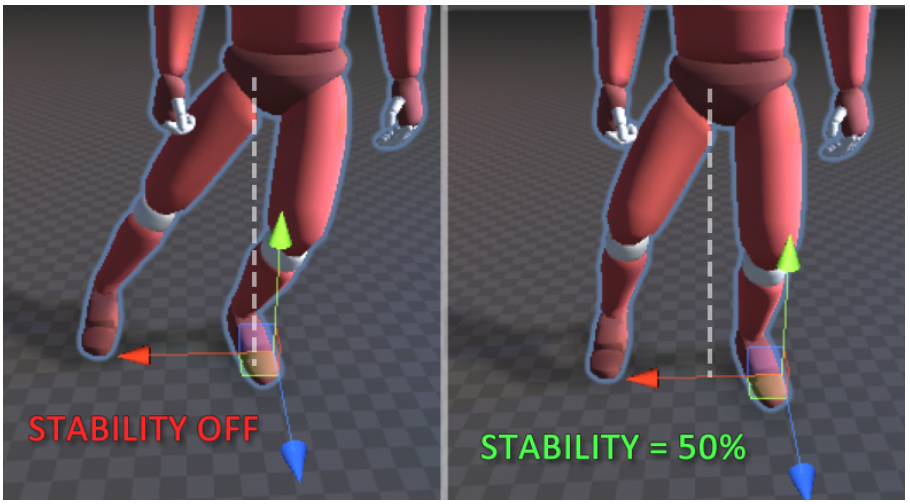



You can also adjust there, how rapidly legs animator should fade out on changing **.IsGrounded** flag to false.

Motion / Hips / **Stability**  :

There you can configure an algorithm, which is adjusting hips position in response to the current legs state and current legs animation.

**Stabilize Center Of Mass** is controlling the amount of applied hips stability offset. Its effect is most visible when using gluing. If leg attachment points are far from hips, then the algorithm will push hips to keep a stable pose of the character. Stabilizing the Center Of Mass will very rarely go over 50%.




**Animation is Stable Pose** is a stability reference point blend.
If it's zero, then the stability point is computed based on the initial T-Pose hips relation with feet. When it's higher then stability point is computed dynamically, basing on the middle point of the feet positions.

**Push Hips on Leg Move** is a multiplicator for gluing step animation influence on the hips push. The push amount is controlled by the curve set under the gluing bookmark.
The **N:** is toggle for normalizing push power, which can make motion less chaotic when your character has many legs and many pushes are happening.
If the push motion seems to make character hips go too much up and down, you can use **Push Y Blend** to calm it down.

**Hips Stretch Preventer** is separately checking current legs stretch amount, caused by procedural positions offsets and if stretch is too big, it tries to pull hips towards it to prevent too big stretch of the leg.

**Stabilize On Is Moving** can calm down the stability algorithm when the character moves. It's using the Leg Animator **.IsMoving** flag to define when a character should use this parameter.

Motion / Hips / **Elasticity**  :

There you can adjust the flow of the hips' elastic animation.
It's using the same approach as my other package Bones Stimulator or as the elasticity settings of Animation Designer but applying it to the hips.
Thanks to this algorithm, the hips motion feels smooth and realistic.
The default settings are universal in all cases, but you can change it to make hips react more rapidly or slower.
Increasing **Motion Influence**, will inherit world movement of the character, on the hips animation. You can increase it, change XZ to zero to automatically support legs bending on character jump/land.

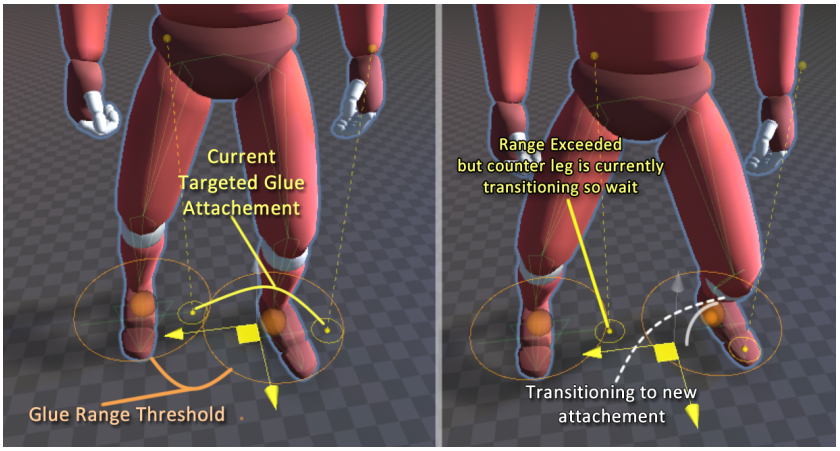
## Motion / **Gluing**  :

This sub-bookmark divides into 2 more categories:


Before entering these categories, let's describe the basics of the gluing feature.

Gluing is generating attachment points when a character's leg is touching ground or is very close to the ground. It can be used to keep character feet stable on the ground.

Leg attachment is smoothly fading out when foot is above ground level.
Leg attachment can transition to a new attachment position with complex leg movement animation, which can be customized with curves in "Idle Glue Motion".
Leg attachments are transitioning when condition to detach is triggered. There are multiple conditions, but the most noticeable one is **Glue Range Threshold**, **which you can debug-view in the scene view when Motion/Glue sub-bookmark is selected:**



**When the character is moving**, you can **switch to the secondary gluing style**, which will not animate leg movement, but smoothly transitioning feet towards attachment point. It's dedicated for attaching legs to ground during movement animations. It can solve foot sliding effect on animations, which movement speed is not synchronized with character movement speed + it can solve feet sliding on uneven terrain movement.

**Glue Fade In Speed** has the biggest influence on the just mentioned Movement Gluing. Raise this value to make feet attach to ground faster. If movement animations of your character are problematic to keep movement gluing effective, it's better to lower this value to give the algorithm more time to define if the leg should be attached to the ground already.

### Motion / **Gluing / Main Glue**  :

**Allow Glue Below Foot** is giving an easier threshold of heel-ground height difference to start gluing foot. Increase it or increase Scale Reference if your character seems to not always attach to ground as wanted. Increasing this value will also make Movement Gluing trigger to-ground transitioning sooner.

**Glue Fade Out Speed** is defining how rapidly the foot will detach. Effect of this parameter is mostly noticeable with Movement Gluing.

**Unglue On [Angle]** is an extra condition to detach foot when the foot ankle is angled too much.

**Allow Glue Drag** is a really important parameter which allows feet to drag towards hips when stretched, skipping different gluing conditions.
It depends on your character movement dynamics and speed but it's recommended to check legs animator behavior with this value increased.

**Swing Helper** can prevent attaching the leg to the ground when the leg is swinging in the opposite direction to the current movement direction. Requires rigidbody velocity read or set legsAnimator.**User_SetDesiredMovementDirection** through scripting.
(you can simply assign character velocity, to be Desired Movement Direction)

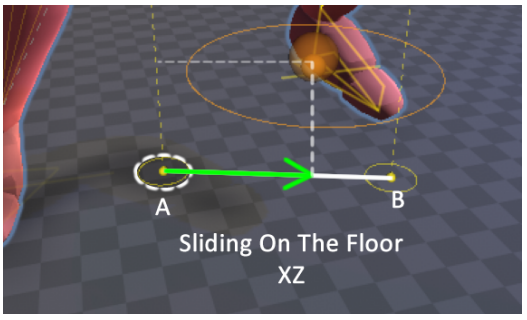
**Floor Level** is gluing attachment detection floor offset level. You can increase this value if attaching is executed too rarely or lower if attaching is executed too often.

Motion / **Gluing / Idle Glue Motion**  :

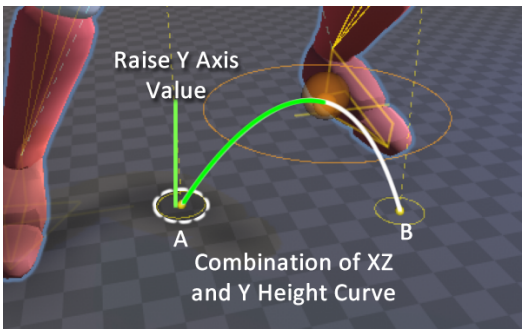
**Idle Glue Motion** is a set of parameters, which defines the character of the Idle Gluing attachment transition animation. (In this window you see simple preview of the target animation)
It can be set to simulate human-like legs animation during rotating in place, or to simulate spider-like leg movement.
You can also set a push effect on hips, during leg transition animation.
You should consider blending it down if you need your character to slide.
(you can also use **.IsSliding** flag for smooth gluing fade out)

**Step Move Duration** is the main leg transition speed parameter. It changes if the step transition is shorter and takes up to the duration value in seconds if it's a long distance step.

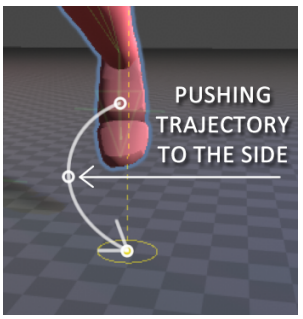
**Move To Goal Curve** is curve of animation transition from attachment A to attachment B in XZ axis, so without leg raising animation.




**Raise Y Axis Curve** is curve of animation transition from attachment A to attachment B in Y-Height axis.




**Spherize Track** can be used to apply extra side axis offset during animating leg transition towards attachment B.
It's simply pushing the trajectory to one of the sides instead of moving the leg straight towards target position without additional curves in motion.




**Min/Max Foot Raise** is the height factor for Raise Y Axis Curve, multiplied by Scale Reference Value. Min value is applied when the attachment step is short and max when attachment step is long.

**Allow Speedups** is rushing the attachment transition progress if one of the opposite legs are getting stretched or ankles angle is rotated too much by being attached. Helpful for dynamic characters.

**Allow Detach Before** is allowing the opposite leg to start transition just before the current leg transition finishes. Can be useful for spider-like creatures.

**Push Hips On Leg Move** is applying hips stability impact push on transition happening to create more realistic motion overally.

**Foot Rotation Curve** is applying pitch rotation on foot (up-down), when transition is happening.

### Motion / **Glue / Glue Mode**:

**Glue Mode** is defining which gluing style should be used. The described above Idle Glue or Movement Glue. You can choose “Automatic” mode, which will switch modes automatically using the **.IsMoving** flag which can be set in several ways.
(animator parameter, coding or velocity is moving module)

# Motion / **Modules**  :

**Modules** are custom, scripted classes which can extend features of Legs Animator.
If you're an experienced programmer, you can code your own module.
In the future versions of Legs Animator, there will come more modules for you.

Hit the  button and choose module from the list to add it and adjust its variables or hit “+” button to add an empty slot for module and choose one using the project file.

Some of the modules are containing variables within the module file, like lists or custom class structures which are not supported by module displayer inside Legs Animator inspector. For example, the module “Impulses On Stop And On Land” is containing push impulse settings data within, or “IK 360 Movement” is containing adjustable values for driving procedural animation.

In the initial version of the Legs Animator, there are included modules like:
- Insect Legs Helper (helping detaching legs during gluing more like a spider creature)
- Impulses On Stop And On Land (calling impulses when IsMoving/IsGrounded flags are changing)
- IK 360 Movement (automatic strafing / 360 movement based on single movement animation!)
- Animator Curves Glue (using animator curves to control gluing triggering)
- Auto Align Body Matrix (aligning raycasting/rotation on the steep slopes)

and a few other minor ones.

## IK 360 Movement Module:

This module is the most complex one.
It gives the possibility to use legs animator for generating 360 movement based on single movement animation.

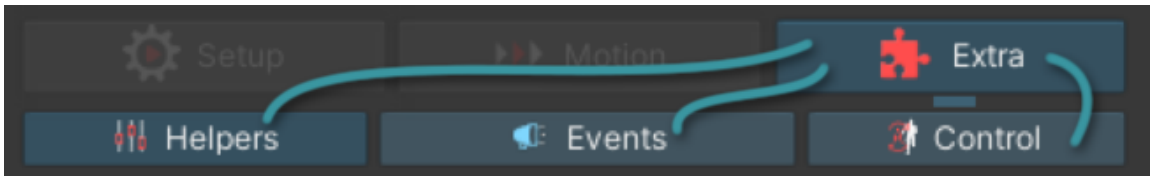
If you find its quality good enough, you can say goodbye to the complex blend trees and limit them to just simple walk/jog/run/spring motion blending. The rest will be handled by IK 360 Movement Module and Legs Animator.

Module requires desired movement direction vector. It needs to be a world space direction. You can provide it using **legsAnimator**.**User_SetDesiredMovementDirection**.

But if you want to control it, **without entering legs animator classes**, you can define **two animator float variables** for x and z direction, assign their names in the module setup and it will be read as the desired movement direction.

The last approach, which does **not require any coding** is using **rigidbody reference**, which velocity will be used to drive the IK 360 Movement desired direction.

# Now the **Extra** sub-bookmarks.




## Extra / **Helpers** :

Providing useful parameters for extra adjustments and corrections.

**Extra Pelvis Offset** will move hips in local space towards provided offset value.
As an example, you can set Y value -0.2 to crouch a bit. You can also control this value with custom scripts if you wish.

**Repose Gluing After** will force gluing reattaching after character stops after movement and after waiting **[value]** seconds.

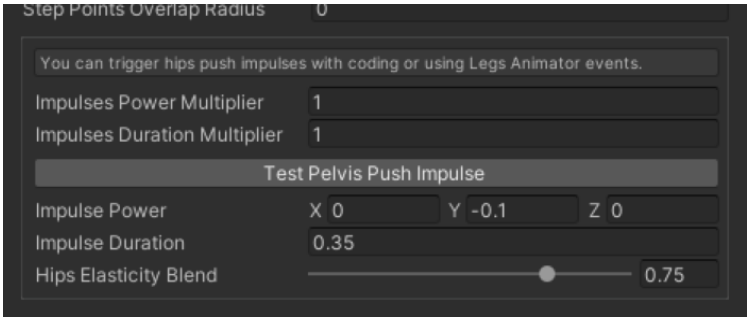
**Glue Only On Idle** will automatically disable gluing when character **.IsMoving**.
If your characters' animations are problematic to adapt dynamic gluing without any effort, you can decide to use gluing only during idling.

**Local World Up** is automatically computing raycasting orientation along with base transform rotation. It's required for planet movement/walking on the walls for insects / on the ceiling, retargeting raycasting.

## Push Impulses:

Push impulses is a basic api for legs animator, to trigger extra animation on the legs animator hips. You can use it for animating jump-landing leg bend or push hips on stopping running etc.

During playmode, in the window down below you can debug-test impulse settings to call on legs animator.




You can use **legsAnimator**.**User_AddImpulse()** to call impulse through custom coding.

Use **LegsAnimator**.**PelvisImpulseSettings** class to define impulse impact settings.

## Extra / **Events** :

You can call custom methods when the algorithm detects leg touching ground.
**The detection is not perfectly precise in the initial version of Legs Animator.**

**Event Execute Sooner** will force the algorithm to call events sooner, before fully confirming leg touching ground.

**Step Info Receiver** is a reference to a game object with the **IStepInfoReceiver** interface implemented. With this interface you will get access to more detailed information about the executed step. Info like ankle position, raycast etc.

## Extra / **Control**  :

There you can define extra helping references for the Legs Animator algorithms, set animator parameters to control Legs Animator through unity Animator and adjust a few other things.

**Grounded/Moving/Sliding Parameter** are names of bool variables to be read from the provided **Mecanim** animator reference.
This variable will control **legsAnimator**.**IsMoving .IsGrounded** and **.IsSliding** flags.

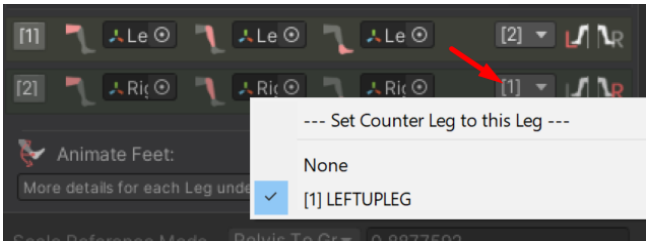
The **Rigidbody** reference will be used by extra features and by some of the scripted modules.
Enable **Use Rigidbody For Is Moving** to control **.IsMoving** legs animator flag without any coding.

During playmode, on the bottom of Extra/Control sub-bookmark you will see debugging for a few variables.

# 3: Setup Tips

**Character is stepping in place:** If after setup, your character is doing steps, when standing in place with idle animation, this can mean that gluing ranges are too small.
You can fix it by increasing “Allow Glue Below Foot” or raising up “Gluing Floor Level” a bit. It can also alert that the “Scale Reference” value is set too small.

**Character is doing a double leg step**: If the character is raising two legs at the same time, when you move it around, that means “opposite legs” are not set up. Go back to setup and choose opposite legs to pair the legs:




**Character is snapping feet to the ground too often/when raising leg during walk animations:** That means the played animation is rising at a very low height during the movement cycle. Algorithm is using thresholds for snapping feet to the ground and you can adjust them, to make snap synchronized better with your own animations.
**Allow Glue Below Foot** is making feet snapping more often, so you can lower it. **Gluing Floor Level** works in a similar way.
**Swing Helper** will prevent snapping when the leg is swinging towards the desired movement direction. You can also make **Glue Fade In Speed** lower, to give the algorithm a bit more time before it confirms that it should fully snap feet.

And the last thing which can help you in such a case, is using **animation curves**.
You can hard-define height of foot in the animation clip time, so the algorithm doesn't need to predict it. It requires a bit more work, since you need to provide a precise height animation curve for the animation clip, define variables in the unity’s Animator, and then read it with the **Utilities/Animation Curves Glue** module.

**Character keeps leg snapped for too long:** You can solve it by using lower **Glue Range Threshold** - to make leg detach sooner, you can also increase **Allow Glue Drag** to make leg shift a bit on being stretched. **Hips Stretch Preventer** parameter, under Stability sub-bookmark also can help, by pushing hips to minimize leg stretch.

**Character foot is rotated in a weird way when standing on the ground:** It may be caused because the initial pose of your model has rotated feet on start. You can simply rotate the foot bones in the editor to be flat-on-ground and it should be solved.

## Debugging Performance Costs:

Legs Animator plugin is offering a quick view on the cost of the legs animator component.

You can enable debug display, by hitting the small  icon, which is visible during playmode. Now the bottom of the inspector window will display time required by the plugin to compute all leg animating stuff.
You can switch different features of Legs Animator and analyze how it affects performance of the plugin, to select the most optimal settings for your setup.

Performance measurement of any plugin is very unstable, so it's better to do a few measurements of min/max ticks, to make sure that your change really affected the performance cost. Cost can jump by a few ticks back and forth, giving different results if the character stands in place and different when the character is moving.

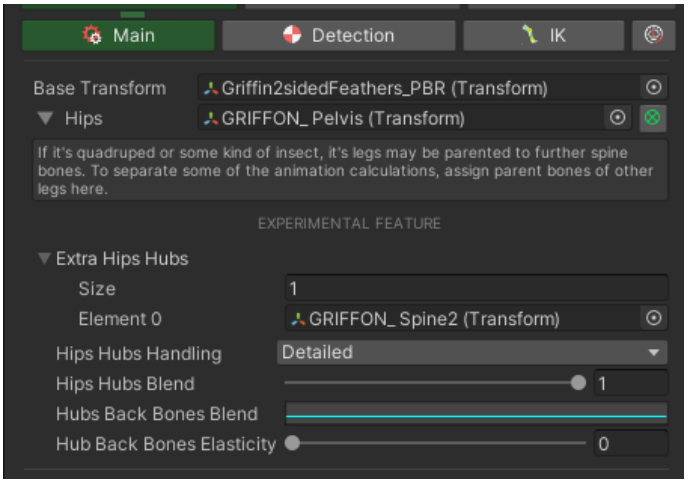
## Hips Hubs (Experimental):

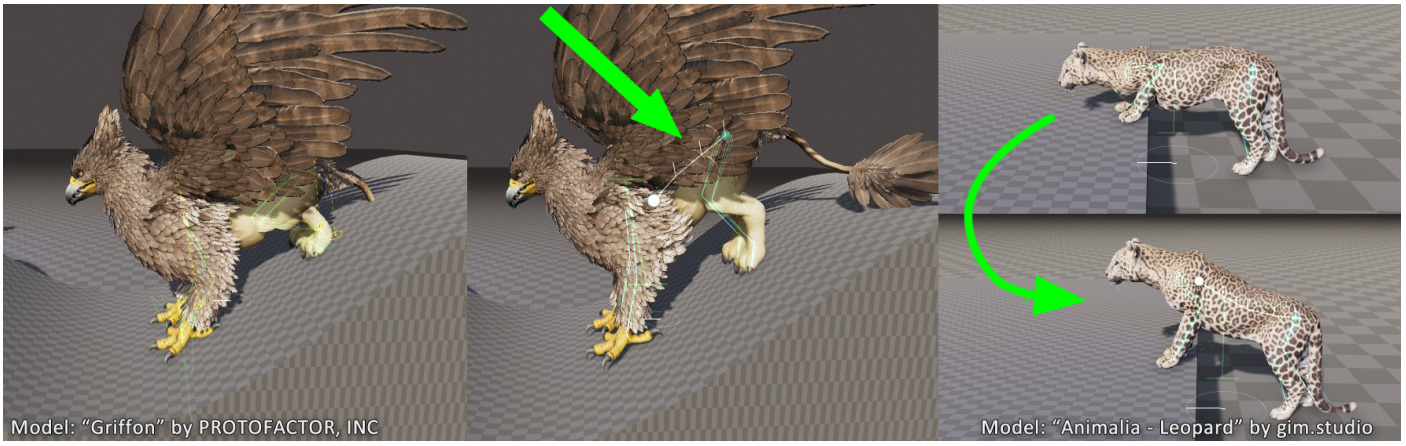
If you want to apply Legs Animator to the quadruped creature, you can try using the **Hips Hubs** feature. It's experimental and can cause some unwanted offsets on the legs, but it will be polished in the future versions of Legs Animator.

If the front legs of your creature are contained by different spine bones than first legs in the spine chain, then you can assign the parent of the front legs.

The hips stability/leg push and other calculations will be calculated separately for the new hips hub.

This feature gives the possibility to bend spine bones across the creature when reaching higher / lower steps. Don't forget to adjust "Max Step Up" parameter to activate step-up spine bending:



# 4: Legs Animator simultaneously with other plugins

If you want, you can use Legs Animator in sync with my other plugins, like:

- **Spine Animator** (for quadrupeds and other creatures)
- **Leaning Animator** (for extra leaning for humanoids)
- **Look Animator** (with look animator + legs animator you can allow rotating more backbones for animals)

You can use **Animation Designer** to create gluing animation curves in an easier way than unity built in animation curves workflow.

You can use the other plugins like **Tail Animator**, **Bones Stimulator** or **Ragdoll Animator** without interference as well.

# 5: Custom Modules API

Since the Legs Animator system is applicable to many types of characters, it requires a lot of different features, to improve motion on individual cases.
Because of that, there are many small features which are useful only on specific kinds of characters. There is a special modules system, to easily implement new features for the Legs Animator system, without editing the core of the plugin.

In order to create a custom module, you need to create the script file and inherit from the type of **LegsAnimatorControlModuleBase**. (namespace: FIMSpace.FProceduralAnimation)
Now in order to make modules do something, you need to override methods, depending what you want to do.
Each overridable method is documented, so write **override** and check your programming environment tooltips.

Now, in order to add your module to the Legs Animator update stacks, you need to create an instance of the module. To do this, add line like:
**[CreateAssetMenu(fileName = "MyModule", menuName = "CREATE INSTANCE OF MY MODULE")]**
On top of 'public class' declaration. After the project compiles, you will find the '**CREATE INSTANCE OF MY MODULE**' menu option if you hit the right mouse button somewhere in the project browser. Select your menu option and module instance will be created. Now you can enter Legs Animator, go to Motion/Custom Modules, hit "+" button and select reference to your module instance.
If you want to display variables in the selected module view, you need to write custom editor gui code. You can read about it here or here, or just check other modules to see how you can implement it in the Legs Animator module, since it's simplified.

If you take a look in the provided Legs Animator modules, you will see implementation of **LegsAnimator.Variable** class, which allows to set up individual variables per Legs Animator, per Module, to adjust in the inspector window.
If you create standard variables just inside module class, like you do for MonoBehaviours and other classes, then these variables will be contained per Module Instance, and you will see them in the inspector window of the selected module instance.

**LegsAnimator.Variable** supports just a few basic types of data, so if you need to hold more complex data for modules, you can keep them inside module instance and create multiple module instances for your project.

When the game starts, the legs animator is generating a copy of the project file module instance. So after starting, all variables of the module are computed per Legs Animator.

**If you like this package please visit my [asset store page] for more or write a review for this asset ;)**