FIMPOSSIBLE CREATIONS

Ragdoll Animator 2

USER MANUAL

About Ragdoll Animator 2

- Ragdoll Animator 2 is a plugin which provides tools for **precisely adjusting ragdoll related components**, like **rigidbodies**, **colliders** and **configurable joints**.
It provides tools for **detailed and easy control** over **physical body animation**.

- Plugin offers many ways to handle **ragdoll body behavior**, like animating physical bones **with currently played animation**, simulating **full body physics**, playing physical animation **when having attached body parts to the different joints**, adding **physical impacts** to the selected bones, **dragging** full ragdoll body by it's leg, and more.

- Ragdoll Animator 2 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 - Ragdoll Animator 2" unitypackage file.

- Ragdoll Animator 2 offers methods for custom usage through code. These methods are starting with "User_" like "**User_AddAllBonesImpact**".

- You can combine Ragdoll Animator 2 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 on Youtube

1: Getting Started

Core Logic of Ragdoll Animator:

Here some information, to give you better insight, before using Ragdoll Animator.

Ragdoll Animator is generating and using unity components such as:

- **Configurable Joins**
- **Rigidbodies**
- **Colliders**

During runtime, Ragdoll Animator is **generating a separated skeleton** hierarchy on the scene, with physical components on it.
It is required to keep the original animated skeleton and physical body **separated**, to simulate physical animation properly, without animator - physics conflicts.

If you need it, you can use the **Pre-Generated Ragdoll Dummy** option, to generate ragdoll hierarchy on the scene, **before entering playmode**.

In the inspector view you will get access to default physical components parameters, which will be assigned across all ragdoll skeleton bones. These values will be updated on the physical body even during runtime, so you can watch their effect on the animation.

Construct bookmark is providing a complex and user friendly system, for ragdoll dummy definition, colliders tweaking and physical components tweaking.
Thanks to that, you can prepare a precise ragdoll setup.

You can control the behavior of the ragdoll dummy and shape it to your needs, thanks to playmode **Motion** properties. You will find them in the inspector view, these parameters can be controlled through code as well. Some parameters require calling **myRagdollAnimator**.**User_UpdateAllBonesParametersAfterManualChanges()** to make changes happen.

Plugin is prepared with many "**User_**" methods which will help you to design wanted ragdoll dummy behavior.

Ragdoll Animator has an **Extra Features system**, which gives the possibility to write custom ragdoll control scripts, **without affecting core code of the plugin**.
You will find many useful **Extra Features** which come with the plugin.
These **Extra Features** can solve many behavior cases, and in result you will need to do much less coding to get desired results. Check **Extra Features List.pdf** file for list of provided Extra Features.

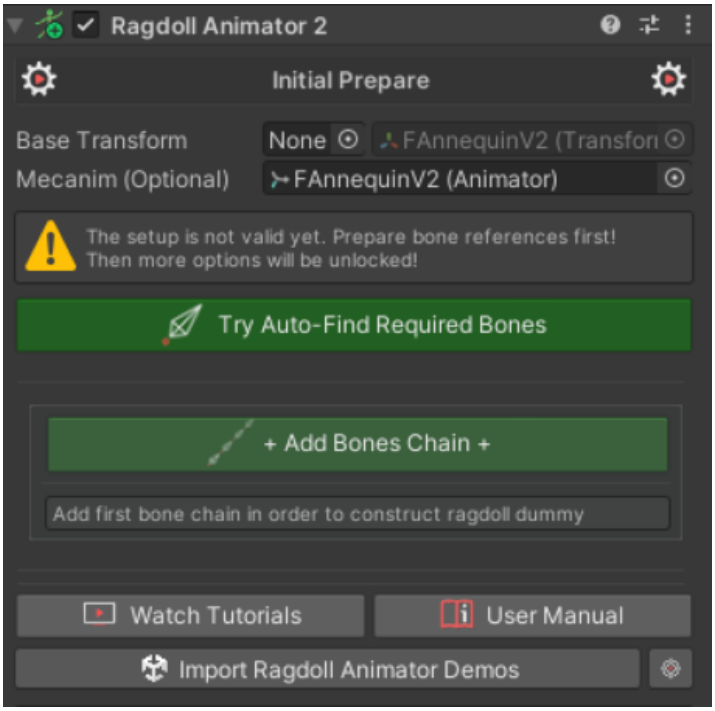
Demo Scenes:

You can find demo scenes with many useful examples under:
"FImpossible Creations/**Plugins - Animating**/Ragdoll Animator 2/Demo - Ragdoll Animator 2"
after unpacking the "Demo - Ragdoll Animator 2.unitypackage" file.

In order to make many automatic features work properly with your model, it needs to face **Z-Forward** axis (it's Unity Engine standard, all models on the asset store meets this criteria)
If it's not, you can always add an extra empty object, which will have a rotated character as a child.

Preparing the Component:

The component you will use is called **Ragdoll Animator 2**.
Select your character object on the scene, hit "Add Component" and go to "Fimpossible Creations -> **Ragdoll Animator 2**" or write "Ragdoll Animator" in the search prompt and select it.



First you will see the initial preparation view.
You need to prepare at least one bone chain in order to unlock more options.

You can leave **Base Transform** empty, if you added Ragdoll Animator to your character controller object. If you added it to some child object, drag & drop main object here.

Mecanim is another optional field, but it is useful to have it assigned, since it will provide many useful features.
using **Mecanim** tech name, instead of **Animator** to not confuse it with Ragdoll **Animator** name

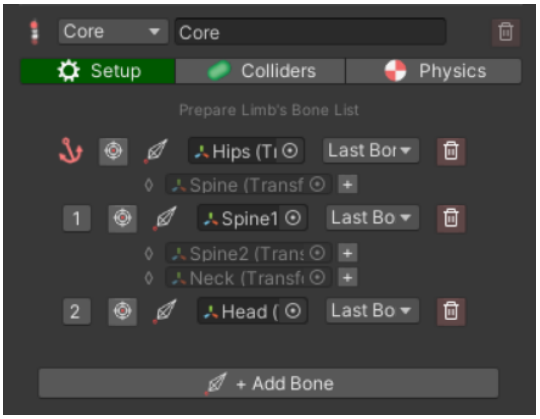
If you're using humanoid rig, you can hit **Try Auto-Find Required Bones** button, to generate default ragdoll body setup automatically.

If you're using other type of rig, then the **auto-find** button will try to identify spine, arms and legs by basic analysis of your character's hierarchy. This option can generate a few unwanted bone chains, like adding ear bones as arms etc. so you need to check the whole setup after using this option.

Third approach will be adding all bone chains manually, which you can start by hitting the **Add Bones Chain** button.

Ragdoll Construct - Setup :

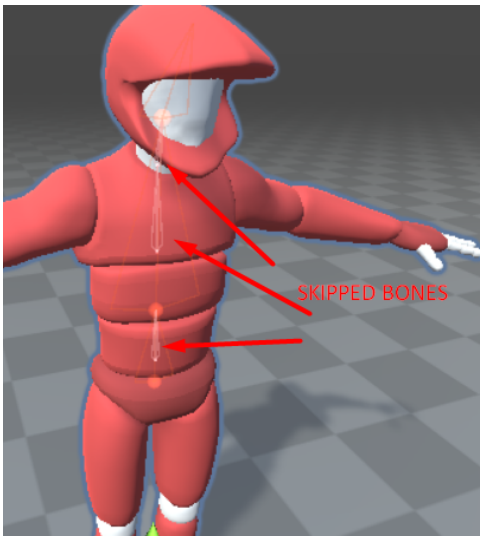
If you want to quickly test a Ragdoll Animator with an automatically generated humanoid rig construct, **you can skip this manual part for now**, and go to **Inspector View Navigation** (page 10)



Read this part, to get an idea how to modify ragdoll structure.

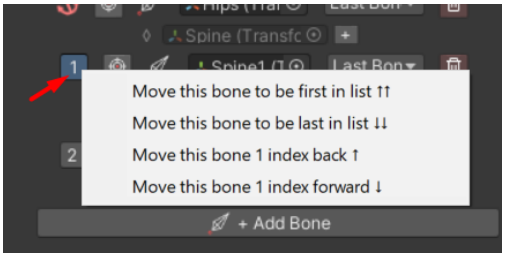
When creating the whole ragdoll setup manually, you need to start with the **Core** bones chain. **Core** contains hips -> spine -> up to head. First bone of the **Core** will be treated as **Anchor Bone** , which holds the whole ragdoll dummy connected.

Hit **Add Bone** for a new bone field. Algorithm will automatically assign the child bone of the previous bone. You can also drag & drop bones from the hierarchy view onto the inspector window.



Dummy setup is flexible, you can skip a few spine bones, and ragdoll will work properly, it even will be more performant, but the physical animation matching will be less precise.

Next to the bone transform field, there is a **Bone ID** field, which is optional, but will be useful for shooter games, to identify hitted bone for different hit effects.



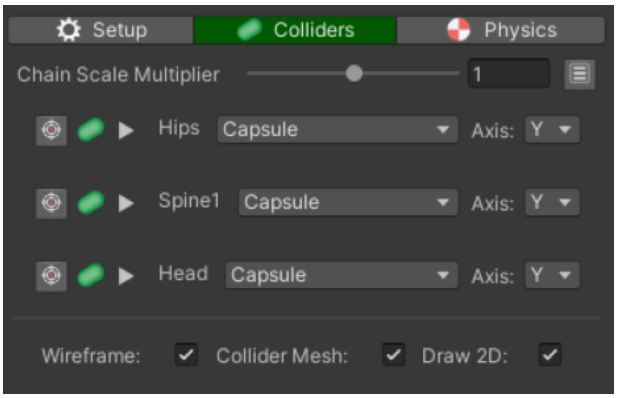
You can hit the bone number button, to display bone order change options, in case you added some bones in the wrong order.

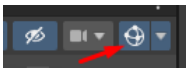
Hit right mouse button on the Setup or button to display helper options.

On the bottom, you will see the **Detach** toggle.
With this option enabled, ragdoll dummy will generate ragdoll bones unparented, which works a bit faster and is not generating glitches when using kinematic bones.

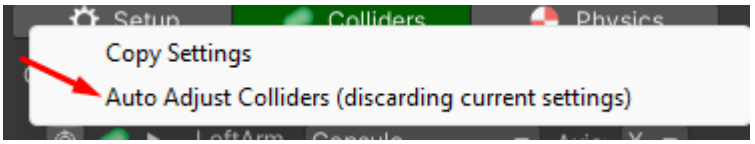
When you hit button, you will be redirected to the bone collider editor.

Ragdoll Construct - Colliders :



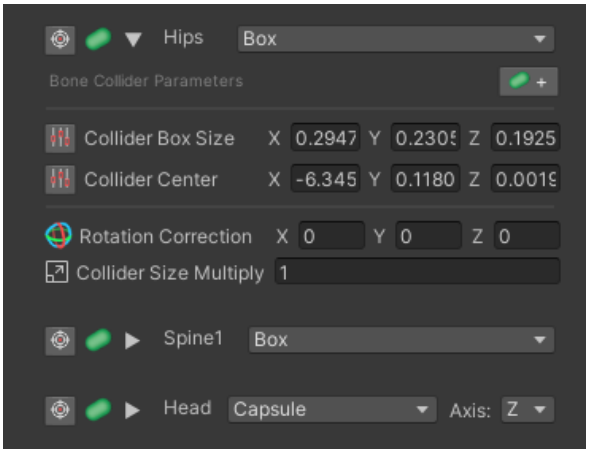
Under this bookmark, you will define **each bone collider size**, **type** and its **positioning**.
Colliders you see in scene gizmos are representation of collider components which will be generated runtime. If you don't see any gizmos, make sure you have gizmos enabled in the scene view: 

After adding all bones for the chain, it is advised to use **auto-colliders adjusting**.
You can call it by selecting the right option after hitting  button next to the **Chain Scale Multiplier**, or by hitting **Colliders** bookmark **using the right mouse button**.



It will adjust collider sizes to match bones hierarchy, and after doing so, you can quickly jump to more precise adjustments, to match your character model.

Expand bone collider to see more options, related to the selected collider type.
Check the scene view to watch the effect of the parameters.
Click on the  button to use sliders for more responsive adjustments.
Using **Rotation Correction** can be very helpful for bones, which are rotated in the wrong direction.
It will trigger generating additional game object, with a rotated collider inside bone transform.



Use **Collider Size Multiply** to quickly scale up/down collider size in all axes.

Use  button to add extra colliders to the bone. It can be useful if you want to add a neck collider for the upper chest bone collider, when not using neck bone in the dummy setup.

When editing bones chain like **arm** or **leg**, you will notice  button, which identifies **symmetry**. Click on it, to copy collider settings to the found symmetrical bone. When you click it with the right mouse button, the inspector view will be redirected to the

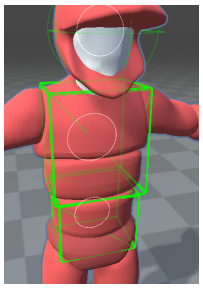
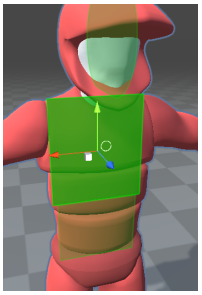
symmetry bone editing. Some symmetrical bones will still need a few tweaks tho, for example the ones with rotation correction involved.

 Button is a quick “solo” collider editor, hiding all other expanded colliders.

You can also click on the bone name, next to the  button, to display collider settings in one fixed position in the inspector window.

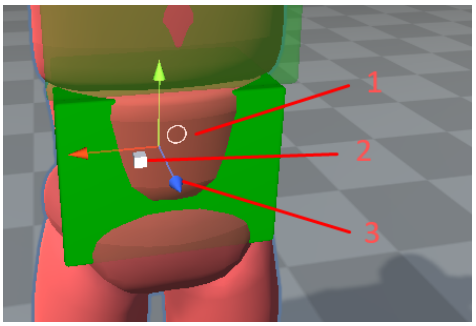
Colliders Scene Gizmos:

On the bottom, you can toggle colliders display mode:

Wireframe:  Mesh:  2D: 

to easier navigate on the scene, and adjust colliders precisely.

You can focus scene view editing, just on one collider at a time.
You will see position/scale handles on the scene when selecting collider for editing:



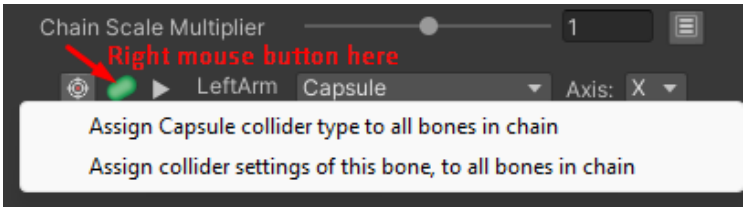
1: Click on the **circle** to deselect the collider, after that you will see a similar **circle** gizmo **on the rest of the colliders** - **click on them to select another collider for editing**.

2: Click on the **box** to switch into a **scale editing** tool, when editing scale you will see **arrow** gizmo instead of box, click on the **arrow** to switch back to the collider **position editing**.

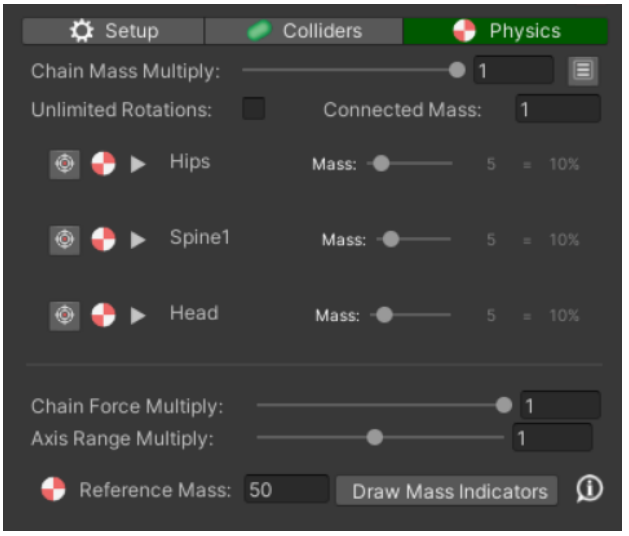
3: Edit collider position/scale using tools handles.
This is affecting position/scale values you see in the inspector window.

After all setup, you can use **Chain Scale Multiplier** to quickly size up each collider in the chain.

Hit **right mouse button** on the collider  icon to display helper options for defining bones chain:



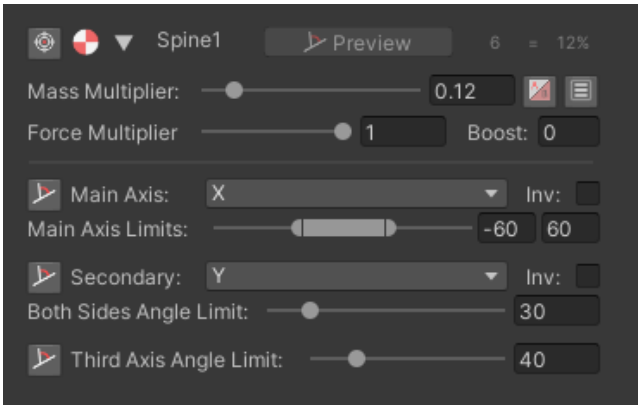
Ragdoll Construct - Physics :



There you will prepare **each bone physical parameters**, to animate them properly during playmode.

Bone Rigidbody **Mass** is a very important parameter. It is defining bone heaviness in the whole ragdoll dummy structure.
Target mass is defined using percentage of the **Reference Mass** Value. Thanks to this approach, you can quickly change the whole skeleton mass, to experiment with physical behavior of the whole skeleton, using different body mass values. You can also trace anatomical values which are also defined with percentages.

After expanding bone physics, you see fields like this:



Don't be scared of the axis fields, it will give you a clear preview on the scene view, which will be explained in a while.

The **Mass Multiplier** is **the same** parameter as the **Mass** slider you see, when bone physics settings are not expanded.

The button is switching display for **advanced physics settings** for a single bone, which you don't need to bother about, until you realize that your project needs much more detailed physical setup per bone.

The button will display few helper options which sometimes can speed up your setup (especially when setting up tail bones chain)

You will see the same option, when hitting right mouse button on the icon, next to the bone name.

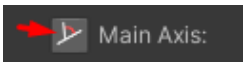
Force Multiplier is controlling physical **Joints Spring** powers for this single bone.
You can set it lower to make a single bone weaker, or use **Boost** to make it stronger.

Now the **Axis settings**. Character bone rotations should be limited to prevent unnatural, weird poses, like the knee should not be able to rotate itself forward. Doing it with built in unity configurable joints gizmos, was always chaotic for me. Approach for adjusting each axis, one by one, with a live preview on the model, seems to be the best way for that.

Beware making limits too small, the physical animation matching will be affected-blocked by the joint limits in some cases.

Configurable joints are using 3 kings of rotation limits.
The **Main Axis** can be limited with a positive and negative value range, while the second axis and third one can be limited only by +- angle. The **Third Axis** is always the cross axis of the **Main** and **Second Axis**.
So it is important to choose the right **Main Axis**, the one which needs a different positive and negative angle for rotations.

X Y and Z axes will rotate differently, depending on the initial bones rotations setup of your character's skeleton. **So X axis can react differently than X axis on another character model/bone.**

In order to preview your axis limit settings, hit the 

button, to activate preview for a certain axis. You can activate a few previews at the same time, clicking on the axis activation icon, using the right mouse button, **but it's recommended doing tweaks only with one axis preview activation**. With preview activated, you will see target bone rotation animation on the scene view. To make it more responsive, hit the 

button to see rotation of the bone on the character model.

You can tweak limits to be a bit smaller than you feel it should be, so later you can enable **angular limit springs**, which will allow bones to rotate a bit more, but with spring power pushing them back. This approach can give you more smooth and less rigid motion.

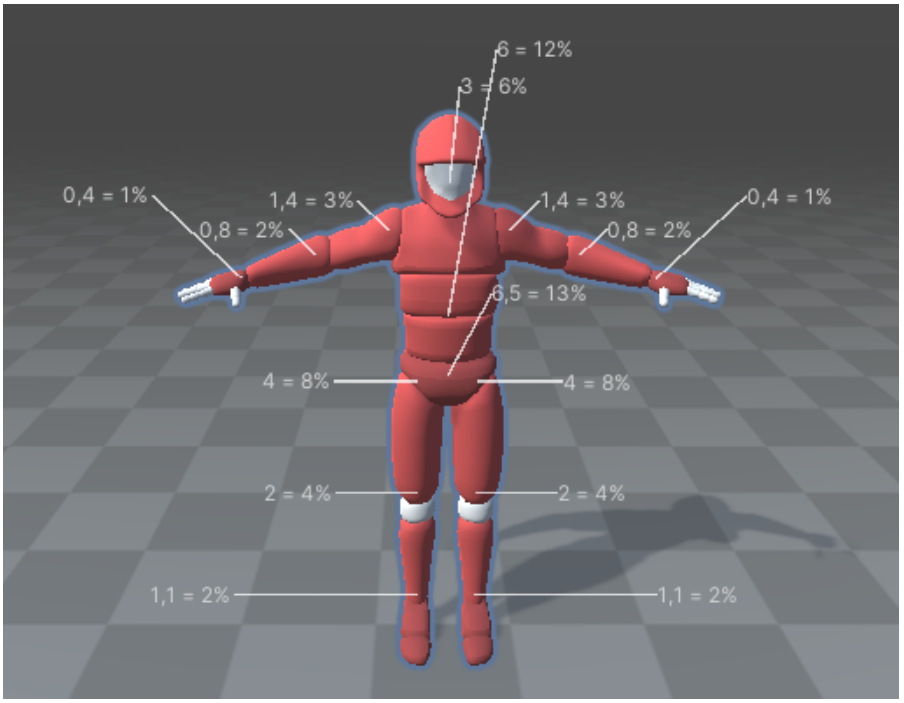
Chain Mass Multiply on the top, is changing mass value for all bones in the chain, which can serve as a quick mass tweak.
Unlimited Rotations switch, will disable joint axis limits. You can use it if your game character will never fall unconsciously on the ground, but use just physical animation matching.

Connected Mass scale multiplier, is a parameter which affects physics of the connected joints. When it is set lower, the body will be lighter and physical animation matching will be quicker, but you can control this value for the whole dummy under bookmarks which will be described in further parts of this manual.

Chain Force Multiply is changing **Joint Spring** values, for all bones of this chain. You can lower it down to make for example limp arm
(remember about no spring damping and connected mass override at least 1 for right gravity effect)

On the bottom you see a shortcut for **Reference Mass** value and **Mass Indicators Button**.

When you hit the **Mass Indicators Button**, you will see each bone mass description on the scene, which can help you build the right anatomical rigidbody mass setup for your character.



2: Inspector View Navigation

When you prepare basic Ragdoll Construct, you will unlock other Ragdoll Animator's bookmarks. 

Under **Setup** you will prepare target behavior for Ragdoll Animator and set reference values for rigidbodies and joints.

Under **Construct** you're preparing Ragdoll skeleton structure, colliders and each bone individual physics settings. There you also control whole bone chains (limbs) parameters.

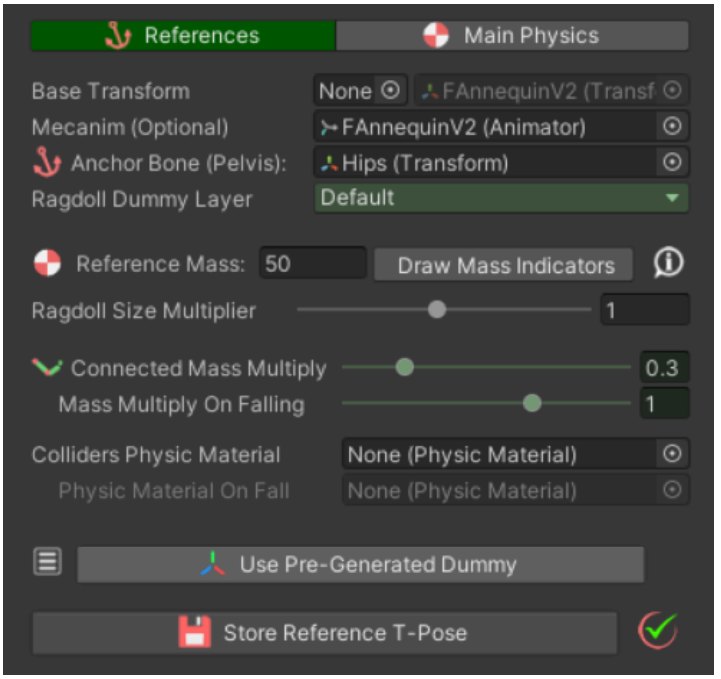
Under **Motion** you will tweak the most common playmode Ragdoll behavior parameters.

Under **Extra Features** bookmark you will add additional less and more complex behaviors for the Ragdoll Dummy (you can code your own extra features)

3: Setup

After preparing basic ragdoll Construct, you should visit this bookmark in the first place. You will find here two sub-bookmarks: **References** and **Main Physics**.

Under **References**, you will find a few fields you saw in the **Getting Started** manual page.



The new field is **Ragdoll Dummy Layer**.
This is a **very important value**, which defines the target **Game Object Layer** for the physical body of your character.
It is important, because in many cases your project will require a **layer for the dummy**, which will **not collide with the character controller layer**. Check unity's **Layer Based Collision** docs for more insight.

Ragdoll Size Multiplier is a shortcut for quick tweaking of all bone's collider sizes.

Connected Mass Multiply is **Joint's Connected Mass Scale** value applied to the ragdoll dummy bones during **Standing Mode**. **When this value is lower**, joints are handling bone animation more lightly, making it quicker and more Physical Animation Matching friendly. **When this value is higher**, the physical body animation will be more heavy.

Mass Multiply On Falling is the same parameter as above, but this value will be applied when Ragdoll enters falling mode. Thanks to higher value the ragdoll will naturally and heavily fall on the ground instead of looking like a super light character.

Colliders Physic Material is **Physic Material** which will be applied to all colliders of the ragdoll dummy. You can use physic materials with zero friction to make character body parts easily slide on the obstacles, minimizing jitter effects, but it's better to use frictional materials during falling mode, otherwise the character will behave like it's falling on the slippery ice.

Physic Material On Fall same as above, but applied during falling mode. When left empty, fall mode will use 'Colliders Physical Material' value.

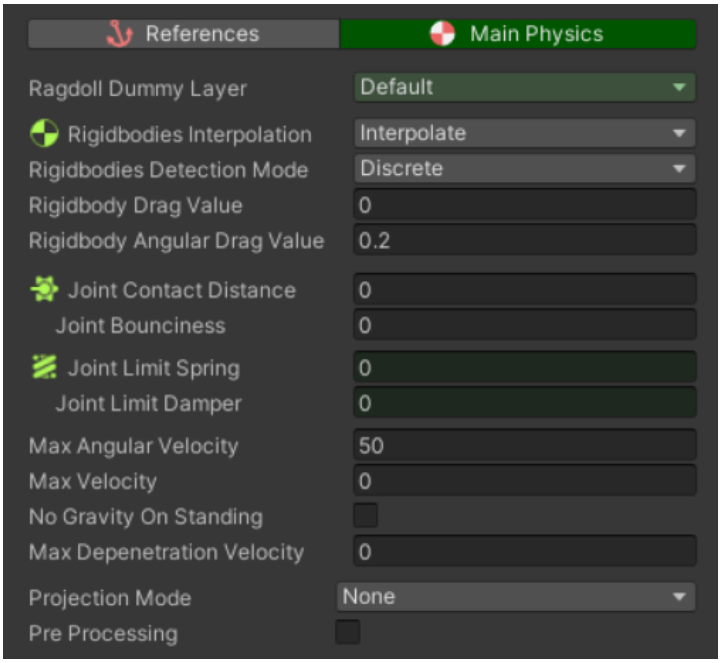
Ragdoll Logic is a whole component logic switch. **Active Ragdoll** is all features ragdoll animating, while **Just Bone Components** is dedicated for basic setup, like turning on ragdoll on death, disabling animator and letting unity configurable joints components control animator bones. It is the most optimal way if you just need death ragdoll animations. You can start with the Ragdoll Animator component disabled, and enable it on character death.

When you hit the  button, you will see helper options, like resetting humanoid character to T-Pose, which is required by unity joints to initialize properly.
Option to call automatic colliders and physics parameters adjusting and option to copy all setup from other Ragdoll Animator components if using the same hierarchy.

Use Pre-Generated Dummy button will generate target physical body skeleton on the scene, during editor mode. So you can use references to the joints/rigidbodies/colliders to make customized scene setup, or modify dummy, add components for the playmode.

Store Reference T-Pose is a helper button, which holds information about a character's default pose, from which unity joints should initialize. It's very important when using reconstruction mode, which will be described later.

Now **Main Physics**  sub-bookmark



There you can tweak target parameters for unity physical components of ragdoll dummy.

Define default settings for unity **Rigidbodies**.
You can define individual settings for each bone under Construct -> Physics -> Advanced Settings view.

Next settings are related with Joints, like **Joint Contact Distance** and **Joint Bounciness**, which are not as useful for the ragdolls.

On the other hand, **Joint Limit Spring** and **Joint Limit Damper** are very useful for ragdolls. Tweaking these values is recommended, since unity joints limits are very stiff by default, which can result in jitter. When you set a limit spring value like 500,

the bone will be allowed to rotate a bit more than its limit, but spring will push it back smoothly.

Max Angular Velocity is **rigidbody max rotation speed**. By default unity limits it very strictly like 7, but for animated ragdolls it's better when it's a bit greater.

Max Velocity is max allowed **rigidbody speed**. When set as zero, it will use unity's default velocity limit.

No Gravity On Standing will turn off **gravity for ragdoll dummy rigidbodies** when using standing animating mode. In some cases it can give you more precise physical animation matching.
Max Depenetration Velocity will apply **this value to all ragdoll dummy rigidbodies**. Which can make overlapped colliders be pushed out more smoothly.

Projection Mode is switching **the same named parameter** for ragdoll dummy's joints. Recommended disabled if not needed.

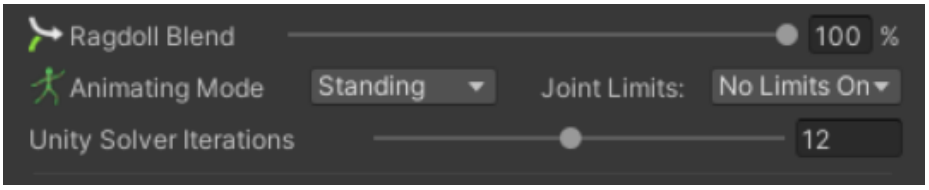
Pre Processing is switching **the same named parameter** for ragdoll dummy's joints. Recommended disabled if not needed.

Purpose of the Setup bookmark is to prepare main values for target ragdoll behavior, and go back to it only if some main tweak is required.
After doing the whole setup, you will visit this bookmark only sometimes.

4: Motion

This bookmark is divided into two vertical parts, and three sub-bookmarks, but don't worry. You will quickly get its flow.

First the top vertical part.



It will stay the same when changing sub-bookmarks, which will be described later.

Ragdoll Blend is a value with which you can control transition between ragdolled, and not ragdolled character animation.
When some feature is modifying this value (like optimization features), you will see small gray value next to it:

Animating Mode is the main ragdoll behavior switch.

You can choose modes between:

- Standing - Restricted anchor bone (hips) position, for active animated ragdoll
- Fall - Free fall bones, so character will fall on the ground + be animated
- Sleep - Same as Fall mode, but turning ragdoll off when bones motion stops
- Off - Simply turned off all ragdoll animator behavior and physics

You could notice "on fall"called properties, on the previous manual pages.
Such parameters are synchronized with the **Animating Mode** switch.
Standing and falling behavior requires use of different parameters, to make motion look right. It is important to prepare them according to your project needs.

Joint Limits is a very important parameter, when you want to use standing, animated ragdoll characters. The joint axis limits can prevent characters doing physically driven poses as the animator poses. Often animation clips are rotating bones in the

way that they're passing the joint limits. On the other hand, limits should be used for the falling / falling to sleep mode characters, to prevent weird bone rotations.

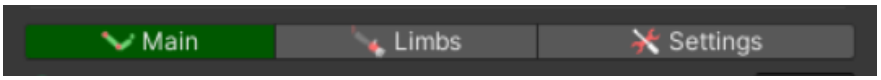
You can choose joint limits mode between:

- **No Limits** - Disabling all limits completely
- **No Limits On Standing Mode** - Disable limits only in Standing Mode
- **All Limits** - Enabling joint limits all the time, not matter animating mode

It's recommended to use **No Limits On Standing Mode** switch, but if your project needs different dummy behavior, you can tweak it to your needs.

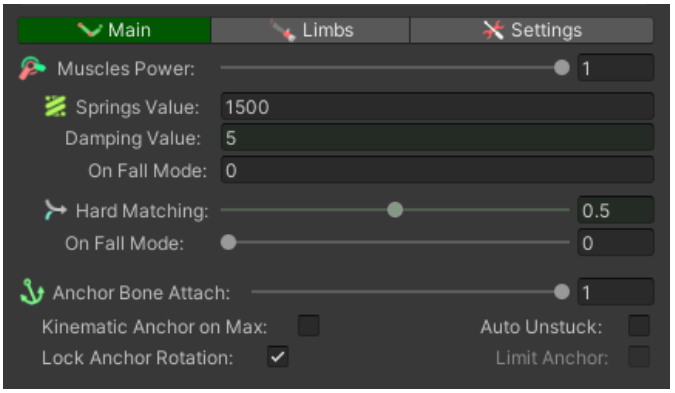
Unity Solver Iterations is target **Rigidbody solver iterations** for all ragdoll dummy rigidbodies. Set it higher for more precise physics, but beware, since it costs more.
You can use **Optimization Extra Feature** to change solver iterations count over distance to the camera.

Now the three sub-bookmarks of the **Motion** category.



Under the **Main** sub-bookmark, you will tweak ragdoll bones - muscles behavior.
Under **Limbs** you will quickly adjust muscles power per bone chain.
Under **Settings** you will set target animating settings.

Let's cover the **Main** sub-bookmark:
This bookmark will be the most frequently visited one, in order to tweak ragdoll behavior during playmode.



Muscles Power is the main multiplicator for all joint rotation driving parameters which will be described below. It's not recommended to set this value zero. If you want to simulate faint behavior, set it pretty low, but not zero. Zero value gives completely limp motion.

Springs Value is the default **spring** for configurable joints rotation drive. It's power which is used by each bone to match desired pose.
You can use **Extra Features** to transition this value to a lower value in fall.

Damping Value is the default **damping** for configurable joints rotation drive.
It's making bones motion less jiggly. When you set it too high, the motion will give noticeably delayed animation.

On Fall Mode is **Damping Value** applied for physical joints on fall mode.
It is important to set damping low in fall. If damping would be high during falling mode, it would result in slow mo-like falling animation.

Hard Matching is an additional physical animation matching factor.
When set above zero value, it is activated, **which results in slightly higher cost for physics calculations**. It will push/force rotate bones towards real animator pose including physical collisions and factors.

On Fall Mode is **Hard Matching** applied for calculations on fall mode.
It's recommended to set it zero on fall mode, since it can make falling animation look not natural.

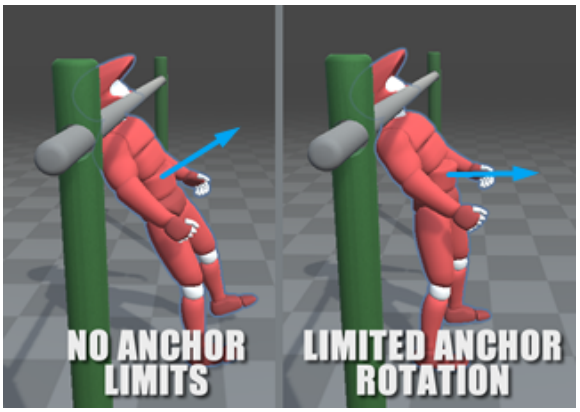
 Anchor Bone Attach is working only during **Standing Mode**.
It is controlling the anchor bone (most likely hips bone which holds the whole skeleton mass) position + rotation joint powers. When set to the max, it will follow the animator's motion without physical motion delay. When set lower, motion can be delayed, but body collisions will be treated more smoothly. **When set to zero**, the character will fall -> **Treated as Fall Mode even during Standing Mode!** You will get different behavior when using **Anchor Bone Attach** zero but using **Hard Matching**, since **Hard Matching** will push anchor bone towards animator pose as extra force.

Kinematic Anchor on Max will make the anchor bone's rigidbody **kinematic** when hard matching is set to the max (1). Kinematic anchor will result in different body behavior. It will overlap all collisions no matter what, the collisions will be ignored, but its animation will be perfectly matching animator pose and be stable. In addition, movement of the body will have no motion effect on the rest of the bones - no interia.

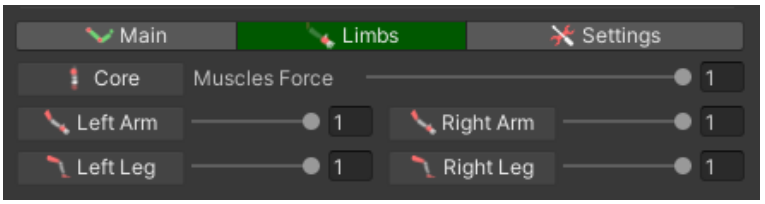
Auto Unstuck will teleport body towards current controller (**base transform**) position, when some obstacle locks physical dummy way back. It can happen if an anchor with **frictional physic material** + **Lock Anchor Rotation** is applied.

Lock Anchor Rotation will enable **rigidbody rotation lock constraints** for the anchor bone, which can give smoother motion, when the body collides with some scene obstacle. When rotation is not locked, anchor is more sensitive for jittering motion on overlap collision. (yes it feels like it should be jittering when rotations are locked, but its not)

Limit Anchor will enable rotation joints limits as tweaked under **Construct -** anchor bone. It is disabled by default, since you usually don't like to limit anchor bone rotations - it could be destructive for physical animation matching. But you can find it useful, if you can let anchor bone rotation, but want to limit it a bit.



Limbs:

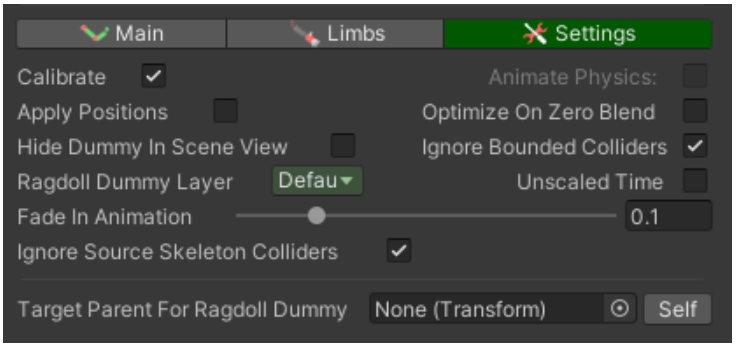


Under the **Limbs** Sub-Category there are not so many things to do. In the future versions of Ragdoll Animator 2 there may be added more properties tho.

As for now, you can control each chain muscles force multiplier.
You can also click on the chain name button, to quickly jump into chain editing **Construct** bookmark.

Settings:

There you can tweak settings which are directly related to your character setup.



Calibrate is preparing all bones for procedural animation.
If your character bones are not animated/not using unity animator, you need to leave this property enabled, otherwise bones motion will look wrong.
When you're sure that source skeleton bones are animated, you can disable it, to get a tiny bit better performance.

Animate Physics is changing the way how the Ragdoll Animator update loop works. It is adapting to the physical update loop (fixed update).
You need to enable it if you use **Animate Physics** on the unity animator. Otherwise you can experience flickering motion on the character. This option is grayed out when you have **Mecanim** field assigned, it will automatically change **Animate Physics** based on the unity animator settings.

Apply Positions is changing positions of the skeleton bones, accordingly with physical joints positions. Sometimes it can provide more precise poses but sometimes it can cause weird stretching on the model.

Optimize On Zero Blend will disable all ragdoll calculations, when **Ragdoll Blend** comes to zero value. When this option is not enabled, zero blend will still compute physical motion under the hood.

Hide Dummy In Scene View is a cosmetic property, which will hide ragdoll dummy object on the scene view, to make hierarchy cleaner.

Ignore Bounded Colliders is checking all dummy colliders on the stars if they're too near each other, and ignoring collisions between them. It's recommended to keep this property enabled, to avoid self collision issues.

Ragdoll Dummy Layer is the same property field as in the previous bookmarks.

Unscaled Time makes dummy calculations being updated in the unscaled time, so unaffected by Time.scale.

Fade In Animation is solving initial unity components jiggle effect, by automatically transitioning **Ragdoll Blend** on Ragdoll Animator enable.

Ignore Source Skeleton Colliders is calling **Physics.IgnoreCollision** between ragdoll dummy colliders and all colliders found in the source skeleton. It's recommended to use different collision layer for skeleton bone and ragdoll dummy, but you can still use this option to ensure no collision between these colliders.

Target Parent For Ragdoll Dummy can be used to put a generated ragdoll dummy under a targeted parent object.

5: Ragdoll Extra Features

This bookmark is not required to get Ragdoll Animator working.
You can use this plugin without using any **Extra Features**, but these features can be a huge help for your project, so it's worth trying them.

You will find a list of **Extra Features** which comes with the plugin, inside "Ragdoll Animator 2 - Extra Features List.pdf" file, to get basic information about each modular feature script.

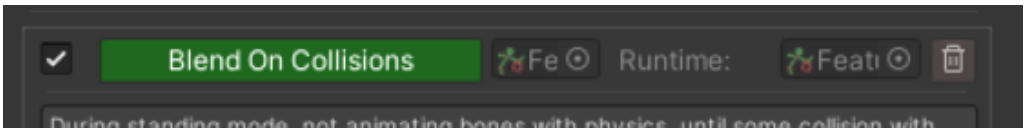
You can also code your own Extra Feature, **without affecting core code of the plugin**, which will be described at the end of this manual.

Toggle on the top is switching On and Off all Extra Features (the initialization on start still will be called)
You can find it useful when debugging performance cost of the component when using performance measure tool

 button will display a list of Extra Features you can add to your Ragdoll Animator component. It is reading all feature assets placed under:
FImpossible Creations\Plugins - Animating\Ragdoll Animator 2\Ragdoll Features

 button is adding an empty field for a Ragdoll Feature. You can use it to select a custom project file reference, if you don't want to put it inside Ragdoll Animator's default features directory.

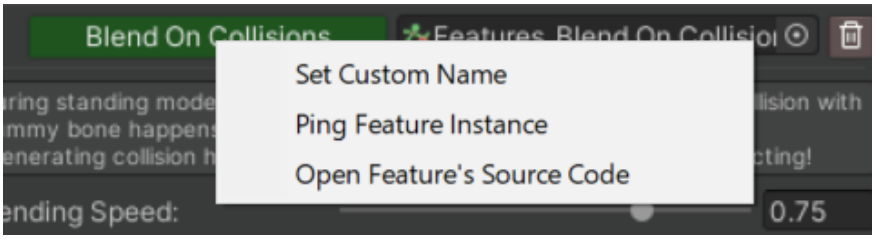
At runtime, Ragdoll Animator is generating copy of the Ragdoll Feature asset file, which you see in the window as below:



Some Ragdoll Features are updated dynamically, some are not. The ones updated dynamically can be disabled with the toggle next to the name button.

Click on the feature name button to display its info and settings.
You can have only one feature selected at the time, so when you click on the other feature name, the current one will be closed.

Hit the right mouse button on the feature name button to display a few utility options.



You can assign custom name to the feature, to get their reference when coding, using **myRagdollAnimator**.Handler.**GetExtraFeatureHelper**("Your Custom Name")

When using coding to get Extra Feature, you will notice that you can get a feature helper and feature instance itself. This helper is a basic class to manage Extra Feature. In order to turn On/Off feature, you need a feature helper reference.
featureHelper.Enabled = …

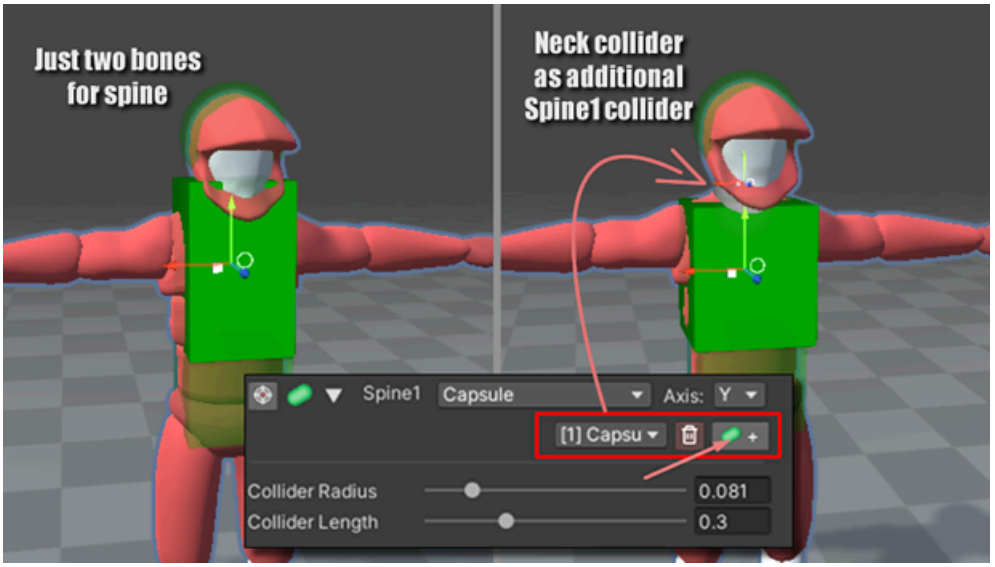
If you want to get and change one of the variables you see in the inspector window, you need to call
var featureVariable = featureHelper.RequestVariable("Variable Name")
You need to check the feature source script file, for the right variable names.
You should find them under the Editor_InspectorGUI() method.

to change variable value or get value, call:
featureVariable.SetValue(value) featureVariable.GetFloat() featureVariable.GetVector3()

6: Setup Tips

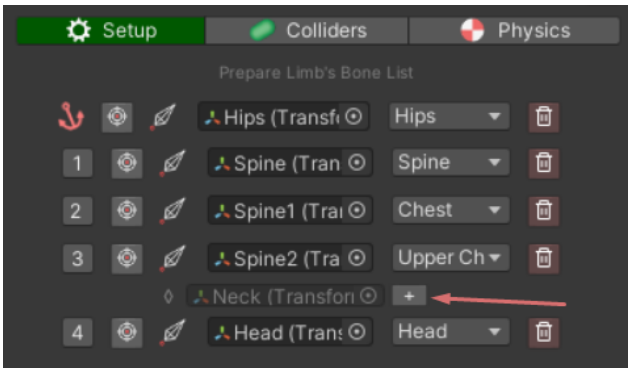
When preparing ragdoll structure, you need to keep in mind a few things.
Using many bones in the bones chain, sometimes can be harder to control and it will take more time to compute them. If you don't need very precise physical animation matching, it's recommended to use just a few bones in the core (hips -> spine -> head) chain and skip shoulder bones.

When tweaking ragdoll colliders, remember about additional colliders option, for a single bone. Thanks to that you can generate a more precise body setup.

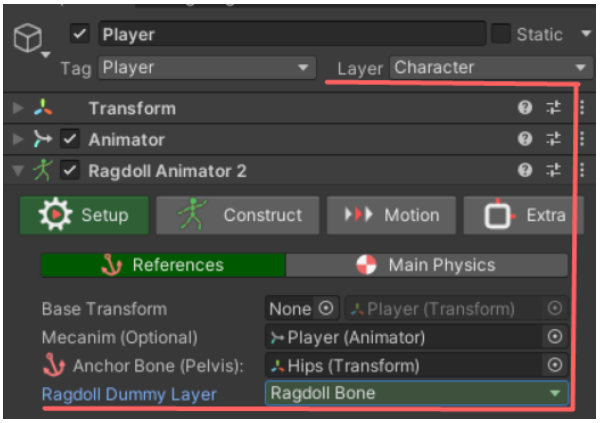


When you use more spine bones, or including shoulder bones in the structure, you need to provide right mass values to them and add more force to keep them balanced during animation. You probably will need to set higher **springs value** and higher **damping** to avoid animation delay and too much jiggly motion.

You can quickly add lacking bones in the chain, simply by clicking on the + buttons on found skipped bones:
Then hit right mouse button on the **Setup** button or hit the  button, to display utility options, then choose first option
(Call auto adjust <Colliders> and <Physics>)



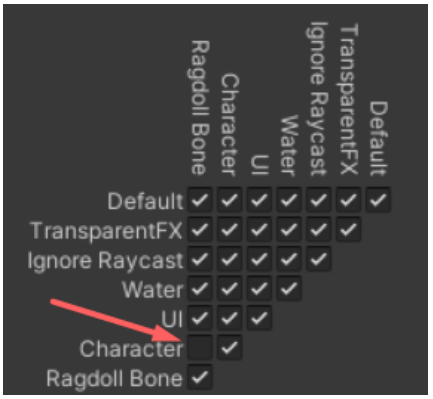
and you should be good to go. Just check colliders for precise tweaking and check physics axis limits.



When using controller colliders for your character + ragdoll dummy, you need to make sure that your character collider will not overlap with ragdoll body colliders.

To solve this, it's recommended to use different layers for character and different for ragdoll dummy.

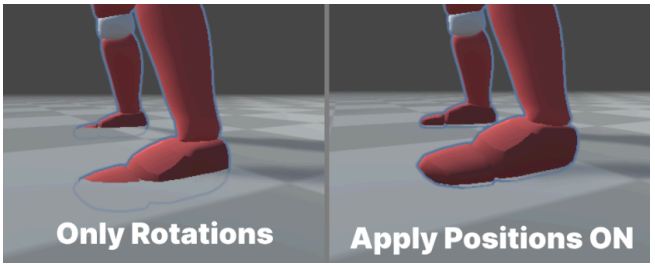
Then go to Edit -> Project Settings -> Physics
And **ignore collisions between these layers**.



You can also solve this by using **Extra Feature**.
Go to **Extra**, hit  button and select Collisions/**Ignore Collisions With**, add slot for a collider and select your character controller collider. Now Ragdoll Colliders will ignore this character collider.

It was mentioned in the previous part of the manual, but it's worth doing it again.
User **Setup** -> **Main Physics**, you will find **Joint Limit Spring** property.
It will minimize the possibility for limit jitter, when bones are rotated to their limits, allowing them to pass limit range, with some power pushing them back. In such a case you can use the bone chain's **Angle Limit Multiplier**, set it for example 0.8 to make limits smaller, as space for the additional spring limit range.

When your character body parts are overlapping with other colliders, when bones are tweaked properly and it shouldn't happen, that means unity physics are forcing joint positions in this particular case. You can enable **Apply Positions** in such a case, but it is recommended to not use this toggle if not needed.



When Ragdoll Animator is initializing, it's triggering physical transition from initial pose, to currently played clip, which can result in jiggly motion on start.
You can enable the **Fade In Animation** parameter and adjust fade in time, to hide the physics jiggle before it calms down.

If you want to make character physical body parts being unaffected by controller translation, you need to make anchor bone kinematic, using **Kinematic Anchor on Max** property. It is more stable this way, but it has downsides though. Unity velocity calculations are disabled for kinematic bodies, and when switching from standing mode to falling mode, anchor rigidbody velocity will be zero instead of keeping current motion velocity. It can be unwanted for characters in movement + starting to fall mode, making the character stop instead of falling in the same direction as it was moving. In such cases you can try applying hips impact with fixed frame delay, like **User_AddAllBonesImpact** -> **waitExtraFixedSteps** = 1 or 2.

You can switch ragdoll dummy bone **rigidbodies to be kinematic**.
During **standing mode**, it will make bone follow animator pose without any delay, but will be unaffected by other colliders. On the other hand, during **falling mode**, it will **lock bone in its position and rotation**, which can be used as an **attached bone**. Kinematic switch works properly with **detached chains**. When using **not detached chains with skipped hierarchy bones** it can cause **destructive motion for the dummy**, so beware.

Kinematic bones can also help you in maintaining source animation properly with physics. You can use **Extra Features** to switch **kinematic feet / hands**. It will result in properly animated limbs during standing mode. **It is super useful to switch kinematic feet for animals**. This way you can set Anchor Bone Attach zero and drive the whole skeleton by foot positioning.

When preparing ragdoll setup, especially for precise animation matching, skipping bones in chains will result in smaller or bigger differences in comparison to source animator pose. You can deal with it in a few ways.

One is adding all animator bones in a chain.

Second is choosing just the bones which have the biggest influence on the resulting pose, so you don't need to add all bones, but just the most pose-important ones.

Third option is to use **Reconstruction Mode**. It will generate additional kinematic rigidbodies in skipped bones and rotate them to match animator pose.
It is an experimental feature though, so test it in all required actions for your project, to make sure if it is not generating some weird effects in some cases.
What's more, when using **Reconstruction Mode**, switching from standing to falling mode can generate additional **Garbage Collector** peaks, since it needs to do a few more operations during the switch. To use **Reconstruction Mode**, you need to use **Extra Feature** under **Features/Reconstruction Mode**.

7: Working with Ragdoll Animator

When working on complex physical interactions, you will need to do coding.
That also requires clear access and many physical operations to be called on the Character's Ragdoll. Ragdoll Animator plugin comes with helper solutions for that.

First let's cover fundamental steps to work with **Ragdoll Animator code**.

The Ragdoll Animator component is placed under **FIMSpace.FProceduralAnimation namespace**.

If you want to use **assembly definitions**, you can unpack them using a unitypackage file like All Fimpossible Creations Plugins Assembly Definitions.unitypackage.
Then Ragdoll Animator will be found under **AD_Fimp.Animating** assembly definition.

Ragdoll Handler:

Ragdoll Animator is **NOT using MonoBehaviour** as the **main logic class**, it is using the **RagdollHandler** class, which you can access through **myRagdollAnimator2.Handler**.

MonoBehaviour is just an object which forwards **Start() Update() and other ticks** to the **RagdollHandler** class. So, if that matters, you're able to implement **RagdollHandler** within your classes without much additional work.
You just need to define handler as variable, implement **IRagdollAnimator2HandlerOwner** interface (just returning a RagdollHandler reference), and display **RagdollHandler editor GUI** using the **RagdollHandlerEditor class**. Use methods like **RagdollHandlerEditor**.GUI_DrawSetupCategory(), but for more check **RagAnim.StartGUI.cs** file.

Ragdoll Animator 2 comes with many extension methods for the **IRagdollAnimator2HandlerOwner** interface. All of the extension methods are starting with **User_** name. Check the *Ragdoll Animator 2 - Extra Features List.pdf* file for list of the user methods.

Next to extension methods are variables and methods of **RagdollHandler**.
Some of them can turn out to be handy for you. Only there you can access bone chains, iterate through bone setups and modify their parameters.
After doing manual changes in the bone chains or bone setups, you need to apply them on the ragdoll dummy. To do it you need to call a method like **User_UpdateAllBonesParametersAfterManualChanges**.

This method is called automatically when editing parameters in the inspector window, that's why runtime preview is instant.

Ragdoll Actions:

If you want to make your character fall, being knocked out of standing mode, use **User_SwitchFallState**. After that you can also call **User_AddAllBonesImpact** or **User_AddChainImpact** to apply impact on the whole bones chain or **User_AddBoneImpact** for a single bone impact.
To get **bone reference** for **AddBoneImpact**, use:
myRagdollAnimator.**Handler**.GetChain().**GetBone**(index)
When the character is falling on the ground, it is worth it to transition the muscles power value to be low, for less stiff falling motion, you can do it by calling **User_FadeMusclesPowerMultiplicator**.
You can also make it automatic with use of **Extra Features**, like **Helpers/Muscles Force On Fall Mode** or **Features/Fall-Get Up Animate**.

If you want to implement Get Up behavior after falling on the ground, using scripting, you need to use a few User_ extension methods.

You can implement Get Up behavior without coding though.
You can use Extra Feature for it, which can be found under **Features/Auto Get Up**.
For get up animation, you should also use the **Features/Fall-Get Up Animate** feature in combination with **Auto Get Up** feature.

But if the extra features are not giving you enough flexibility you need for your character movement controller, you will need methods as below:

User_CanGetUpByRotation - to get body rotation for get up animation, in addition you can provide canBeNone true, to avoid getting up when the character is lying on the side.

User_GetChainBonesAverageTranslation - to check if the body stopped falling and is stable lying on the ground.

User_ProbeGroundBelowAnchorBone - to check ground contact under a ragdoll dummy anchor bone, in case it is still over ground.

Then **reposition your character controller** to ground hit point or current ragdoll dummy anchor bone position.
Calculate target character controller rotation, basing on the anchor bone position and feet position: ragdoll.**User_GetMappedRotationHipsToLegsMiddle**();
Now reset your character movement script velocity/position/rotation.
Then you can play "Get Up" animation and call **User_TransitionToStandingMode**.

You can take a look at **RAF_AutoGetUp.cs** extra feature file code, or **Demo_Ragd_AutoGetup** demo script for more detailed insight.

Collision Events and Bone Indicators:

You probably would like to call some actions when ragdoll bones collisions are happening. To do it, you can use **IRagdollAnimator2Receiver** interface implementation and **Collisions/Collision Events** Extra Feature, or **Collisions/Collision Messages** Extra Feature + method like
void **RagdollAnimator2BoneCollision**(**RA2BoneCollisionHandler** hitted)
(requires FIMSpace.FProceduralAnimation using)
inside your class.
There are few **bone indicator** classes, inheriting from **RagdollAnimator2BoneIndicator**. Some of them are implementing collision events. When using Extra Features mentioned above, the **RA2BoneCollisionHandler** (inheriting **RagdollAnimator2BoneIndicator**) will be generated on the **ragdoll dummy bones**, to forward collision events further on.

When ragdoll bone colliders contact other colliders, you will receive hit information with the **RA2BoneCollisionHandler** instance. This instance contains precious details like parent **Ragdoll Handler**, **Bone Setup**, **Bone Type**, **Parent Bones Chain** reference, which you can use to identify action to perform on collision.

When you do **raycasting**, to get for example bullet hit collision, you can try calling **raycastHit**.**Collider**.**GetComponent**<**RagdollAnimator2BoneIndicator**>
and use the collision indicator data for your gameplay use.
But sometimes, **ragdoll dummy pose can differ from visible animator pose**, like when using **Blend On Collision** **Extra Feature**.
To make raycast detection more precise, you can **add bone indicators on the source skeleton** (you also need to toggle adding colliders to the source skeleton) as well and use a layer mask to detect only source skeleton colliders.

Debugging Performance Costs:

Ragdoll Animator plugin is offering a quick view on the cost of the ragdoll 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 animating stuff.
You can switch different features of Ragdoll Animator and analyze how it affects performance of the plugin, to select the most optimal settings for your setup.

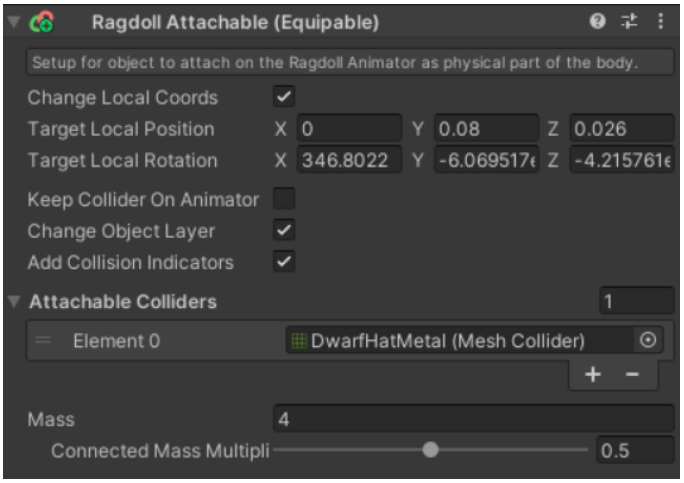
Performance measurement of any plugin is very unstable, so it's better to do a few measurements of min/max ticks, to make sure that your change really affected the performance cost. Cost can jump by a few ticks back and forth, giving different results if the character stands in place and different when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

You should also trace the unity profiler physics bookmark, since these values are not included in the component measurements.

8: Additional Utility Components

Attachables:

Attachables is a helper system for the Ragdoll Animator 2, making possible attaching items like weapons / helmets or other stuff to the physical body and transferring these objects' collisions / mass settings onto the physical ragdoll body structure.



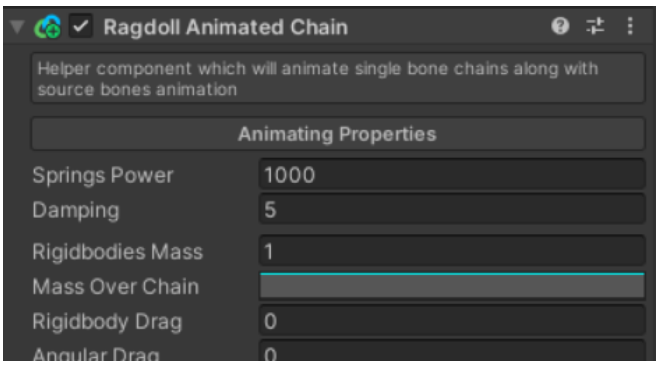
You will define **Attachable Item Setup**, by adding a **Ragdoll Attachable** component to your item game object.

There you define its settings.
It's target local position/rotation when attached to the bone (optional)
Collision settings and physical settings like Mass.

Now in order to use it with the Ragdoll Animator, you need to use coding or use Helpers/**Switch Attachable** Extra Feature.
For coding, you need to access the **Ragdoll Handler**, and call **WearAttachable** or **UnwearAttachable**.

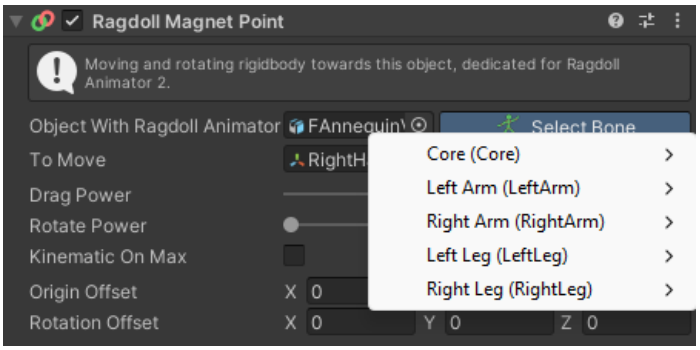
Single Animated Physical Chain:

There is a helper component, which will provide you a fraction of Ragdoll Animator's code, in order to **generate colliders-rigidbodies and joints** on any transform chain, then **animate it with physics**. You will find there similar properties like in the Ragdoll Animator GUI.
But this component is dedicated for automatic, quick bone chain setup, starting from the first parent bone and adding all components along, towards the last child bone you assign. You can't select which in-between bones



should be included in the chain or not, all bones in the provided hierarchy will be included. **It can be a super fast setup for example for tail bones chain**.
Just assign the first tail bone under **First Parent Bone** and end bone of the tail under **End Child Bone**, hit **pre-generate**, adjust **Radius,** hit **Pre-Generate** and there you go.

Magnet Point:



This component can help you **drag ragdoll dummy selective bones**, **towards desired position**. It works similarly like **Spring Joint**. When you assign an object with **Ragdoll Animator** to the field, you will get the possibility to select bones which will be dragged runtime. It's really useful when you're not using a pre-generated ragdoll dummy, so you don't have access yet, to the physical bones.

Basic Joints Chain Generator:

This is a simplified version of the **Single Animated Physical Chain** component.
This one is just generating a selected type of joints and colliders over provided bones without handling animating it with source animator. It can be used for example for generating physical joints structure for rope bones.

Rest of the Ragdoll Animator Utility Components:

Transfer Joint To Ragdoll Bone: Copying provided **joint** into ragdoll dummy bone, which you choose in the inspector using the view selector button. Thanks to that, you can add unity joint to the ragdoll dummy bone, without needing to pre-generate.

Ragdoll Bone as Parent: Changing this object parent, to be runtime generated physical dummy bone, which you can choose using inspector view selector button.

Ignore Collision Between Colliders: Basic helper component to trigger collisions ignore on runtime.

Set Joint Connection Body: Use selector in the inspector view, to choose ragdoll dummy bone, which rigidbody will be assigned as the **connected body** of your scene **joint** component.

9: Ragdoll Animator simultaneously with other plugins

If you want, you can use Ragdoll Animator in sync with **my other plugins**:

- **Spine Animator** (for quadrupeds and other creatures)
- **Tail Animator** (for tails animation)
- **Legs Animator** (for procedural legs animation)
- **Look Animator** (spine and head rotation to simulate looking towards target)

You need to keep in mind that physical animation will be not as precise as direct procedural animation generated by the components.
Physical Animation will always have slight delay, it will be affected by physical factors like colliders friction, muscle reaction to raise bones towards target pose, rotations limit assigned in the ragdoll setup, rigidbody drag parameters etc.
Also it's recommended to use **Detached** mode for each ragdoll bones chain, when using procedural animation plugins, especially when using Tail Animator.

10: Ragdoll Extra Features API

Since the Animated Ragdolls are often chained with complex and various game mechanics, it may need many specific operations to make it work as intended. Adding tons of use-case features within the core of ragdoll animator, will bloat its logic with unnecessary actions, used only in specific cases, making component code hard to read, hard to control and hard to develop further on. So the Ragdoll Animator Extra Features API comes in help.

Thanks to this API, you can add extra functionalities to the plugin, without affecting core code. It gives access to the most important variables and the update tick actions.

In order to create a custom Ragdoll Animator Extra Feature, you need to create the script file and inherit from the type of **RagdollAnimatorFeatureBase**.
(namespace: FIMSpace.FProceduralAnimation)
You can also inherit type as **RagdollAnimatorFeatureUpdate** to automatically inherit update loops support, or inherit **RagdollAnimatorFeatureCollisions** for easy implementation of bones collision hit events handling.

Now in order to make a feature 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, or check how the other extra feature scripts are made.

Now, in order to find your extra feature under Ragdoll Animator inspector view, you need to create an instance of the feature. To do this, add line like:
[CreateAssetMenu] On top of 'public class' declaration. After the project compiles, you will find the '**Create MyModuleScriptName**' 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 Ragdoll Animator, go to Extra, hit "+" button and select reference to your module instance.

You can also put the generated feature asset file under
FImpossible Creations\Plugins - Animating\Ragdoll Animator 2\Ragdoll Features
Then you will see its name when hitting the  button.

If you want to display variables in the selected extra feature view, you need to write custom editor GUI code. You can read about it here or here, or just check other extra

features to see how you can implement it in the Ragdoll Animator feature, since it's simplified.

If you take a look in the provided Ragdoll Extra Features, you will see implementation of **FUniversalVariable** class, which allows to set up individual variables per Ragdoll Animator component, per Extra Feature, to adjust in the inspector window.
If you create standard variables just inside the feature class, like you do for MonoBehaviours and other classes, then these variables will be contained per Extra Feature Instance, and you will see them in the inspector window of the selected module instance. These variables will be project global variables.
You can use private variables during runtime though, because during runtime, there is an individual extra feature instance.

FUniversalVariable supports just a few basic types of data, so if you need to hold more complex data for extra features, you can keep them inside extra feature instance and create multiple module instances for your project.

You will find a list of **Extra Features** which comes with the plugin, inside "Ragdoll Animator 2 - Extra Features List.pdf" file, to get basic information about each modular feature script.

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