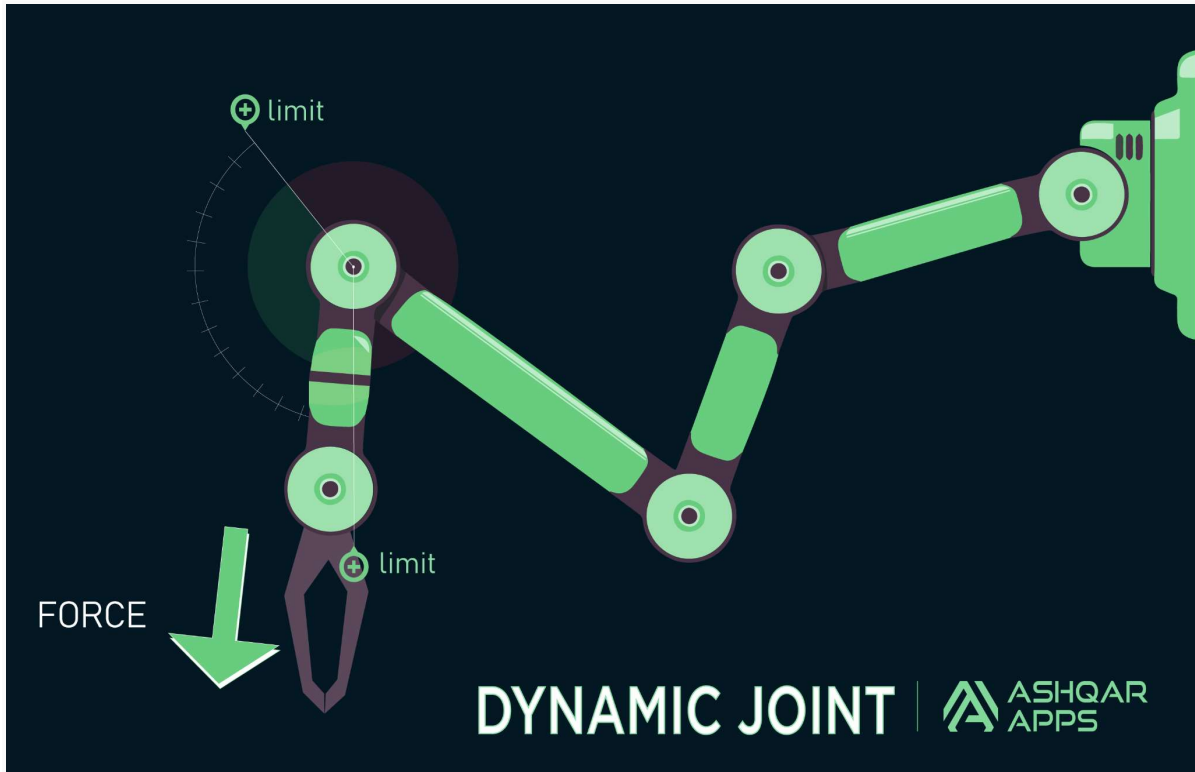


Dynamic Joint Pro for Unity

A fast, robust mass-spring dynamics system which supports **JOINT ROTATION LIMITS** in its dynamics simulation, making it the only particle-dynamics solution of its kind to solve for joints with strict hinge and swing-twist limits cheaply and continuously without needing to go through an external physics engine.



FEATURES

- **Dynamic Joint:** simulate individual joints with mass-spring dynamics under **gravity, inertia** or **spatial-relation targets**. Author dynamic properties for **positional** and **rotational** behaviors. Manually author center of mass direction.
- **Dynamic Chain:** simulate joint-chain hierarchies with mass-spring dynamics under different forces while preserving distance constraints.
- **Joint Limits:** Limit rotations on Dynamic Joints or on joints in Dynamic Chains while also being used in the dynamics solver directly to achieve continuity.
- **Dynamic Transform Filter:** Move & rotate a gameobject towards a target transform's position and rotation while simulating mass-spring dynamics to generate secondary motion & oscillations. Adjust positional and rotational behaviors separately by changing specific dynamics properties for each.
- **Layered forces:** simulate procedural motion using [Constant / Sin-Wave / Perlin-Noise] progression modes with custom amplitudes, frequency along time and frequency along chain to generate 'tentacle-like' motion
- **Elastic Stretch:** simulate stretchy bone-lengths in dynamic chains under forces controlled by **stretchiness** and **stiffness** values
- **Drag:** simulate air/water resistance in dynamics
- **Custom Distance Constraints:** add bespoke distance constraints between any two joint-particles even if they don't share a hierarchy.
- **Dynamic Colliders:** limit motion of dynamic-chain particles. Generate directly from Unity's native colliders, or set collision against a layer
- **Dynamic Inverse Kinematics:** drive end particles towards their targets in a physical way with full CCD-like IK solutions while still being affected by forces/collisions/dynamic secondary motion.

DYNAMICS SETUP

CORE DYNAMIC PARAMETERS:

Force Strength	<input type="text" value="1"/>
Damping	<input type="text" value="1"/>
Spring Strength	<input type="text" value="5"/>
Drag	<input type="text" value="0"/>

Force Strength: the multiplier applied to forces in the dynamics simulation which act upon each particle.

Spring Strength: Equivalent to spring stiffness, the ability of the particle to resist forces. The greater it is, the larger the forces needed to move the particles.

Damping: the reduction in the amplitude of the particle oscillations as a result of energy being drained from the system against resistive forces. The greater, the quicker the particles will slow down and converge.

Drag: simulates the resistance exerted by a fluid stream acting upon the particles as they travel. Increase this value to simulate the effect of particles traveling through fluids such as water.

DYNAMIC JOINT

Add a Dynamic Joint component onto a joint to simulate it with customisable dynamic properties. Author forces such as gravity or inertia. Drive forces based on custom spatial relations to achieve weight-shift secondary motion and suspension behaviors, all bounded by any authored joint limits.

#

✓

Dynamic Joint (Script)

?

↔

⋮

PreProcess

Script

DynamicJoint

Source

▼ Sources

Size

1

Element 0

Fan_Quad_Base (Transform)

Position Dynamics

Use Position Dynamics☐

► Position Dynamics

Rotation Dynamics

Use Rotation Dynamics☒

► Rotation Dynamics

Limits

Use Limits☐

Use Rotation Limits☒

Use Position Limits☒

Position Limits

Min Position Limits

X 0

Y 0

Z 0

Max Position Limits

X 0

Y 0

Z 0

Forces

Gravity

X 0

Y 0

Z 0

Mass Inertia

3

Simulation Mode

Simulation Mode

GRAVITY

Spatial Relations

Target Relative To

PARENT

Offset Eulers

X 0

Y 0

Z 0

► Custom Relative Points

BASE SETUP

Sources: The list of gameobjects to generate dynamic particles for and simulate as dynamic joints. If not defined or left empty, then upon runtime the sources will automatically be populated with the gameobject upon which the component exists.

PreProcess Button: With the component settings setup, press this to generate particles for each of the sources and to initialize and serialize them. This way, specific particle properties can be modified and saved in the editor, such as changing the direction of the center of mass of a particle. If never pressed, then the particles will be generated upon runtime with default settings.

DYNAMICS SETTINGS

Position Dynamics: The dynamics parameters which affect the translational aspect of the movement of the source gameobjects towards their simulated target positions. If **Use Position Dynamics** is disabled, the source gameobject positions will snap to their target positions without being driven there via dynamics hence losing positional secondary motion.

Rotation Dynamics: The dynamics parameters which affect the rotational aspect of the rotation of source gameobjects towards their simulated target rotations. If **Use Rotation Dynamics** is disabled, the source gameobject rotations will snap to their target rotations without being driven there via dynamics hence losing rotational secondary motion.

LIMITS

Use Limits: global flag for whether to use position or rotation joint limits at all.

Use Rotation Limits: flag for whether to use rotation limits. If enabled, any joint limits on our gameobject will apply.

Use Position Limits: flag for whether to use position limits. If enabled, the position deltas from the target positions will be limited by the specified position limits.

Position Limits: The min & max position deltas of each joint will be limited by the specified **Min Position Limits** & **Max Position Limits**, specified in the local space of each joint's parent or spatial-relations frame.

FORCES

- **Gravity:** The gravity force applied to the particles. The greater the magnitude, the greater the strength. The strength is also modulated by *Force Strength* in the dynamics settings.
- **Mass Inertia:** The force applied to each particle to drive it towards its initial position and orientation relative to its parent space. The higher the value the more resistant to change from its initial pose. This force helps simulate weight-shift with secondary motion in moving systems.

Simulation Mode: A dropdown selection to toggle whether we are simulating in **GRAVITY** mode or in **SPATIAL_RELATIONS** mode.

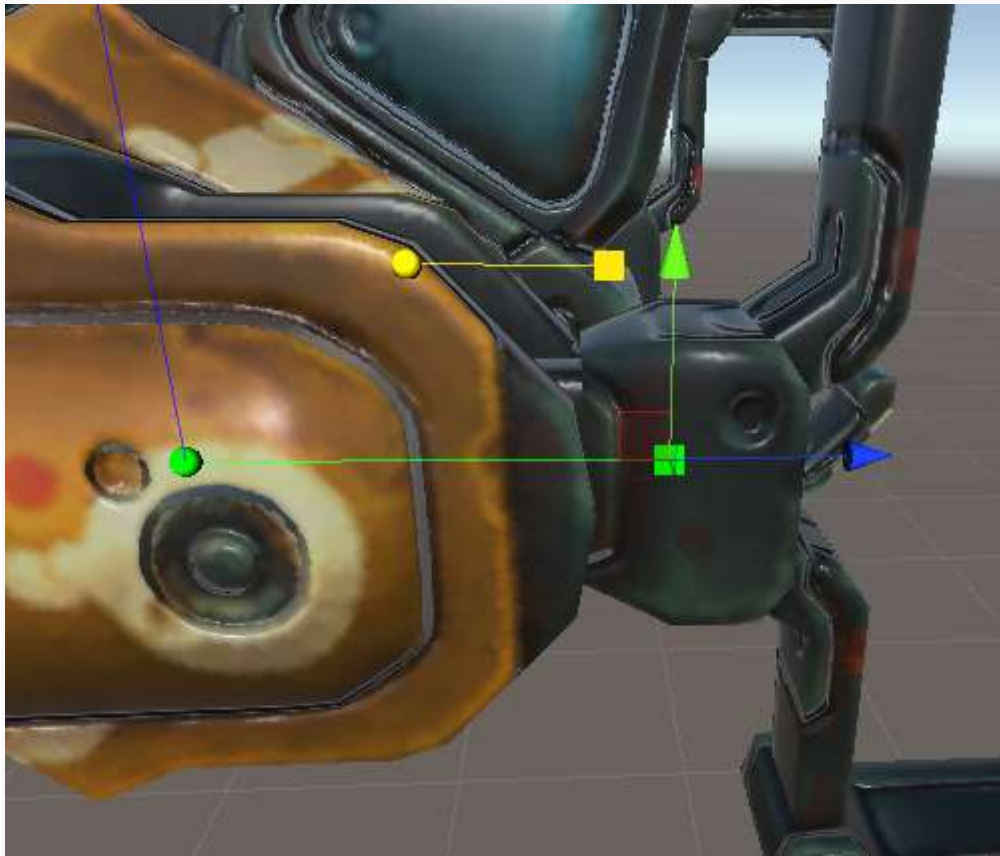
GRAVITY mode will simulate our joint by applying *Gravity* and *Mass Inertia* forces.

SPATIAL_RELATIONS mode will simulate by driving out joints towards targets based on spatial relations to other points.

- **Target Relative To:** Select whether our spatial relations are relative to the joint **PARENT**, **CUSTOM POINTS**, or **CUSTOM POINTS TRIANGULATED**.
- **PARENT** will simulate each joint by driving it towards its initial position and rotation relative to its parent gameobject.
- **CUSTOM POINTS** will simulate each joint by driving it towards its initial position and rotation relative to a list of custom gameobjects.
- **CUSTOM POINTS TRIANGULATED** is the same as above except relative to a triangulation between several custom points hence not affected by their changing rotations but only by their changing positions.

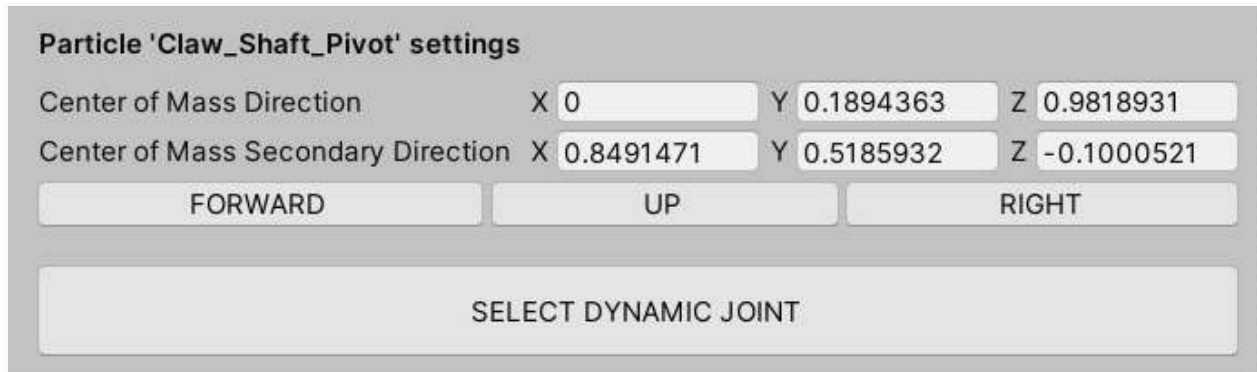
The screenshot shows a software interface with a grey background. At the top, there is a section titled 'Custom Relative Points' with a downward-pointing triangle icon. Below this, there is a sub-section titled 'Relative Points' also with a downward-pointing triangle icon. Under 'Relative Points', there are two rows of settings. The first row has the label 'Size' in blue text, followed by a text input field containing the number '1'. The second row has the label 'Element 0' in blue text, followed by a dropdown menu showing 'None (Transform)' and a circular icon with a dot in the center.

Offset Eulers: A constant rotation delta to be applied to the joint after resolving its rotation relative to its spatial relations targets.



If pre-processed and selected, each Dynamic Joint particle generated will be represented in the Scene View by a yellow sphere point and a vector representing the direction of **Center of Mass Direction** which will determine how gravity will act upon the object since there is no concept of hierarchy otherwise. Gravity will act to swing the joint gameobject to drive the direction of center of mass towards the direction of gravity. You can change the center of mass direction in the Scene View in the editor by selecting it and moving it with the movement handle.

If a particle is selected in the Scene View, a custom UI section will become visible for it in the inspector:



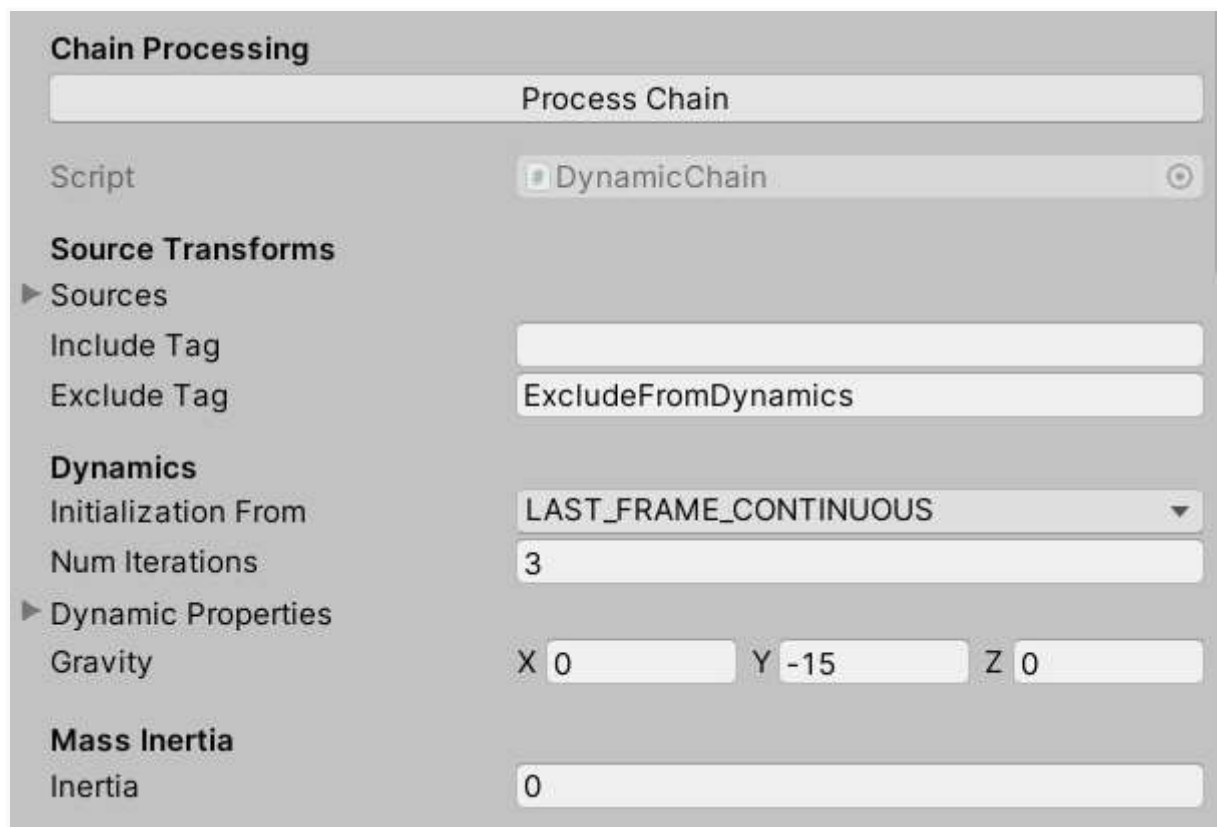
The image shows a custom inspector UI for a particle named 'Claw_Shaft_Pivot'. It features two rows of input fields for direction vectors. The first row is for the 'Center of Mass Direction' with X=0, Y=0.1894363, and Z=0.9818931. The second row is for the 'Center of Mass Secondary Direction' with X=0.8491471, Y=0.5185932, and Z=-0.1000521. Below these are three buttons labeled 'FORWARD', 'UP', and 'RIGHT'. At the bottom is a large button labeled 'SELECT DYNAMIC JOINT'.

Particle 'Claw_Shaft_Pivot' settings			
Center of Mass Direction	X	0	Y 0.1894363 Z 0.9818931
Center of Mass Secondary Direction	X	0.8491471	Y 0.5185932 Z -0.1000521
		FORWARD	UP RIGHT
SELECT DYNAMIC JOINT			

Here the **Center of Mass Direction** Vector can be manually changed in the inspector directly, or set automatically with the **FORWARD / UP / RIGHT** buttons which will set it to the respective local axis in the joint transform space.

DYNAMIC CHAIN

Generate dynamic chains upon hierarchies of joints and simulate them under different forces such as gravity or inertia. Layer custom forces with different propagation modes such as SIN-WAVE and PERLIN NOISE to create procedural motion such as 'tentacle motion'. Constrain particle motion by authoring custom distance constraints or adding colliders. Constrain particles with position constraints or drive end particles towards targets using dynamic INVERSE KINEMATICS.



The image shows a configuration panel titled "Chain Processing". It contains several sections with input fields and dropdown menus:

- Process Chain**: A button labeled "Process Chain".
- Script**: A dropdown menu showing "DynamicChain" with a circular icon to its right.
- Source Transforms**: A section header.
- Sources**: A section header.
- Include Tag**: An empty text input field.
- Exclude Tag**: A text input field containing "ExcludeFromDynamics".
- Dynamics**: A section header.
- Initialization From**: A dropdown menu showing "LAST_FRAME_CONTINUOUS".
- Num Iterations**: A text input field containing "3".
- Dynamic Properties**: A section header.
- Gravity**: Three text input fields for X, Y, and Z coordinates. X is "0", Y is "-15", and Z is "0".
- Mass Inertia**: A section header.
- Inertia**: A text input field containing "0".

Sources: the root gameobjects to generate dynamic chains from. Particle chains will automatically be generated from each source transform traversing all the children of the object and creating a particle for each child in its hierarchy.

Include Tag: if not empty, while traversing the hierarchy of each gameobject, only generate particles for child gameobjects which are tagged with a tag of this string.

Exclude Tag: if not empty, while traversing the hierarchy of each gameobject, do not generate particles for child gameobjects which are tagged with a tag of this string.

Process Chain Button: With the component settings setup, press this to generate chain particles for each of the sources and to initialize and serialize them. This way, specific particle properties can be modified and saved in the editor, such as changing the radius of a particle. If never pressed, the chain particles will be generated upon runtime with default settings.

Initialization From: How to initialize particle transforms each frame of the simulation.

- **LAST_FRAME_CONTINUOUS:** do not reset transforms between frames.
- **STATIC_POSE:** reset transforms of each particle gameobject to the initial static pose rotations.
- **SOURCE_ANIMATION:** reset transforms of each particle gameobject to the pose of the gameobject as driven by an active source animation.

Gravity: The gravity force applied to the particles. The greater the magnitude, the greater the strength. The strength is also modulated by *Force Strength* in the dynamics settings.

Mass Inertia: The force applied to each particle to drive it towards its initial position and orientation relative to its parent space. The higher the value the more resistant to change from its initial pose. Designed to work when *Initialization From* is set to either *STATIC_POSE* or *SOURCE_ANIMATION*. If set to *SOURCE_ANIMATION*, the forces will drive the particles towards their animation targets.

EXTENDED DYNAMIC PARAMETERS

Start Weight	1
End Weight	1
Surface Friction	0
Joint Limit Friction	0
Prevent Stretch	<input checked="" type="checkbox"/>
Stretchiness	1
Stiffness	1

Start Weight: the Spring Strength Multiplier for particles at the start of the dynamic chain.

End Weight: the *Spring Strength* multiplier for particles at the end of the dynamic chain. Values are assigned as a gradient of *start weight* to *end weight* from the start of the chain to the end of the chain. E.g. if end weight is a high value, then particles at the end of the chain will resist forces more than those at the start of the chain and oscillate less.

Surface Friction: the friction multiplier applied when colliding with dynamic colliders. Friction is applied as a damping effect.

Joint limit friction: the friction applied when particles are clamped to the boundaries of their joint rotation limits. Friction is applied as a damping effect.

Prevent Stretch: If enabled, the bone-length between each particle and its parent are strictly enforced to be fixed. If not the bone-length is free to stretch.

Stretchiness: The degree to which the bone-length can stretch for particles. Active only when *prevent stretch* is disabled.

Stiffness: The resistance of the particles to deviating from the target stretch. The greater the value, the more closely they will stick target stretch / bone-lengths.

Limits

Use Joint Limits ☒

Joint Limit Strength 1

Solve Twist Dynamics ☒

Max Twist Rate

IK

Enable IK ☐

IK Target

Strict IK ☐

Use Joint Limits: if enabled, apply rotation limits based on Dynamic Joint Limits which exist as components on any particle gameobject during the dynamics simulation.

Joint Limit Strength: the joint limit strength to apply. If less than 1, apply joint rotation limits as soft limits, not strictly applying them but partially applying them.

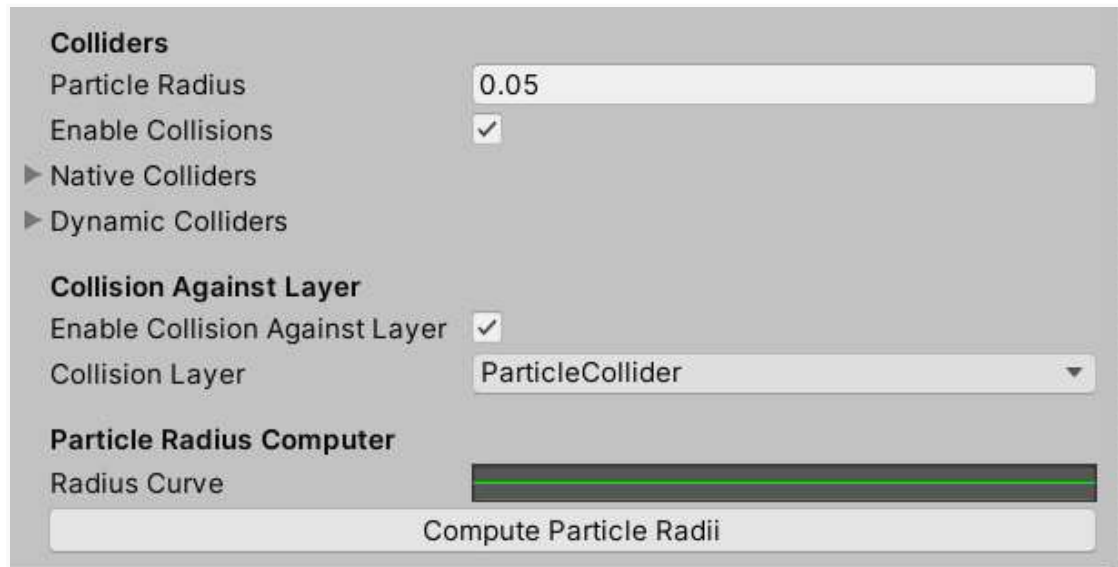
Solve Twist Dynamics: The dynamics simulation is based on particles applying swing rotations to move their child particles towards their targets. This excludes twist. For particles with joint limits applied, if this setting is enabled then apply twist rotations along the twist axis (vector of a particle to its child) in order to also move its grandchild to its target. Swing rotation solves for the child of a particle, twist rotation solves for its grandchild.

Max Twist Rate: The max twist rate in degrees per second that the particles can twist. Swing rotations are limited by particle dynamics settings as they represent the translational motion of the particles. We need this additional value to limit big or sudden twists of particles which aren't captured by the dynamics settings otherwise. Only relevant if *Solve Twist Dynamics* is enabled.

Enable IK: if enabled, solve Inverse Kinematics for any particle which has an IK constraint and target enabled.

IK Target: The gameobject representing the default target for any particle with an IK constraint enabled. If a particle is IK constrained and has no override target defined, then use this target.

Strict IK: When disabled, solve IK under dynamics. This way, even when reaching for target, forces, secondary motion and collisions are still active upon the chain. If enabled, ignore all dynamics and colliders to strictly achieve the IK target without secondary motion and dynamic effects.



Particle Radius: The default base radius to generate each particle with when generating via *Process Chain* button. Each particle can have its radius overridden separately,

Enable Collisions: Toggle whether to enable collider interactions between particles and colliders in the dynamics simulation.

Native Colliders: Populate this with specific Unity-native colliders to specify which colliders the particles should interact with. Dynamic Colliders will automatically be generated as an intermediate structure.

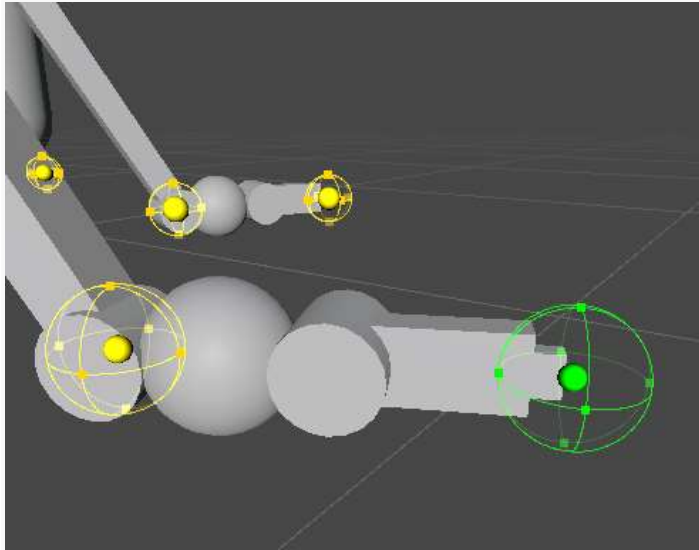
Dynamic Colliders: Populate this with Dynamic Collider components directly if manually authored.

Enable Collision Against Layer: Toggle this to activate collision detection between particles and ALL colliders existing on a selected **Collision Layer**.

Radius Curve: A curve representing particle radii from root to end particles when generated via *Process Chain* button. Is a fixed line by default, so that all are the same. Press **Compute Particle Radii** to regenerate particle radii based on the curve without regenerating particles via *Process Chain*.

PARTICLE SPECIFIC SETTINGS

With the Dynamic Chain component selected and the particles generated (via *Process Chain* inspector button) , the scene view should display every particle generated as a yellow sphere. These are selectable. You can change the radius of each particle via the radius handle within the scene view, or select the particle which will toggle it with a green color and display its relevant Inspector fields.



Particle 27 settings
Transform: l4

DESELECT

GO TO GAMEOBJECT

Target Constraint

NONE

POSITION CONSTRAINT



INVERSE KINEMATICS

TARGET FORCE

Target Position

X 0Y 0Z 0

Target Transform

 None (Transform) 

Constrain Rotation

☐

Particle Radius

0.05

Custom Distance Constraints

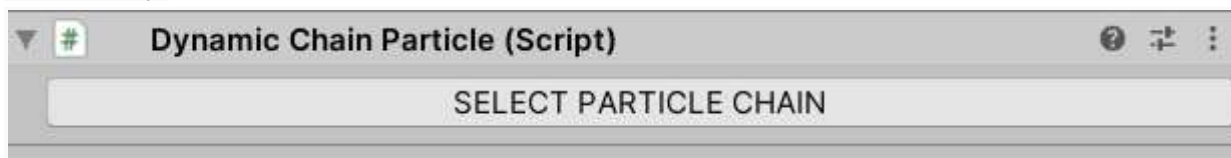
Author New Distance Constraint

With a specific particle selected, these modifiable fields should become visible within the Inspector UI:

Deselect (button): deselect the particle and go back to showing the full Dynamic Chain settings within the Inspector instead.

Go to GameObject (button): Select the gameobject upon which the particle was generated in the scene view and inspector. This is a quick way to jump to the gameobject in order to modify any joint limits.

Conversely:



Any gameobject which has a particle generated for it should have a **Dynamic Chain Particle** component existing on it, which should display a **SELECT PARTICLE CHAIN** button which when pressed will select the gameobject which contains the Dynamic Chain component which drives this particle.

Target Constraint: there are 4 position constraint modes for a particle.

- **NONE:** no constraint, default, free to move
- **POSITION CONSTRAINT:** apply a position constraint onto a target position or transform (transform taking priority, use position only if no transform is defined)
- **INVERSE KINEMATICS:** apply an IK constraint towards a target position or transform (transform taking priority, use position only if no transform is defined). If **Constraint Rotation** is enabled, additionally enforce a rotation on the end particle to match that of the target transform rotation. Only one particle per branch can have an IK constraint enabled.
- **TARGET FORCE:** Similar to Position Constraint, except drive particle towards a target via a force, with force strength defined by a **Target Force Strength** field.

Particle Radius: Define the particle radius. Used for collisions.

Author New Distance Constraint: Press this button to start creating a distance constraint relative to the selected particle.

Custom Distance Constraints

Select a particle to constrain this with.

This will show in inspector when you press the button. Select the other particle to constraint relative to in the Scene View.



When you do a constraint will be created displaying a line between the two and a square button in the Scene View representing the constraint. It will be fixed distance by default.

In the particle specific UI, a constraints section will be displayed:

Constraint 1:	
Constraint Type	<input type="text" value="FIXED_DISTANCE"/>
Value	<input type="text" value="0.781179"/>
<input type="button" value="Delete Constraint"/>	

The **Constraint Type** is of three types:

FIXED_DISTANCE: enforce a fixed distance **Value** between the 2 particles

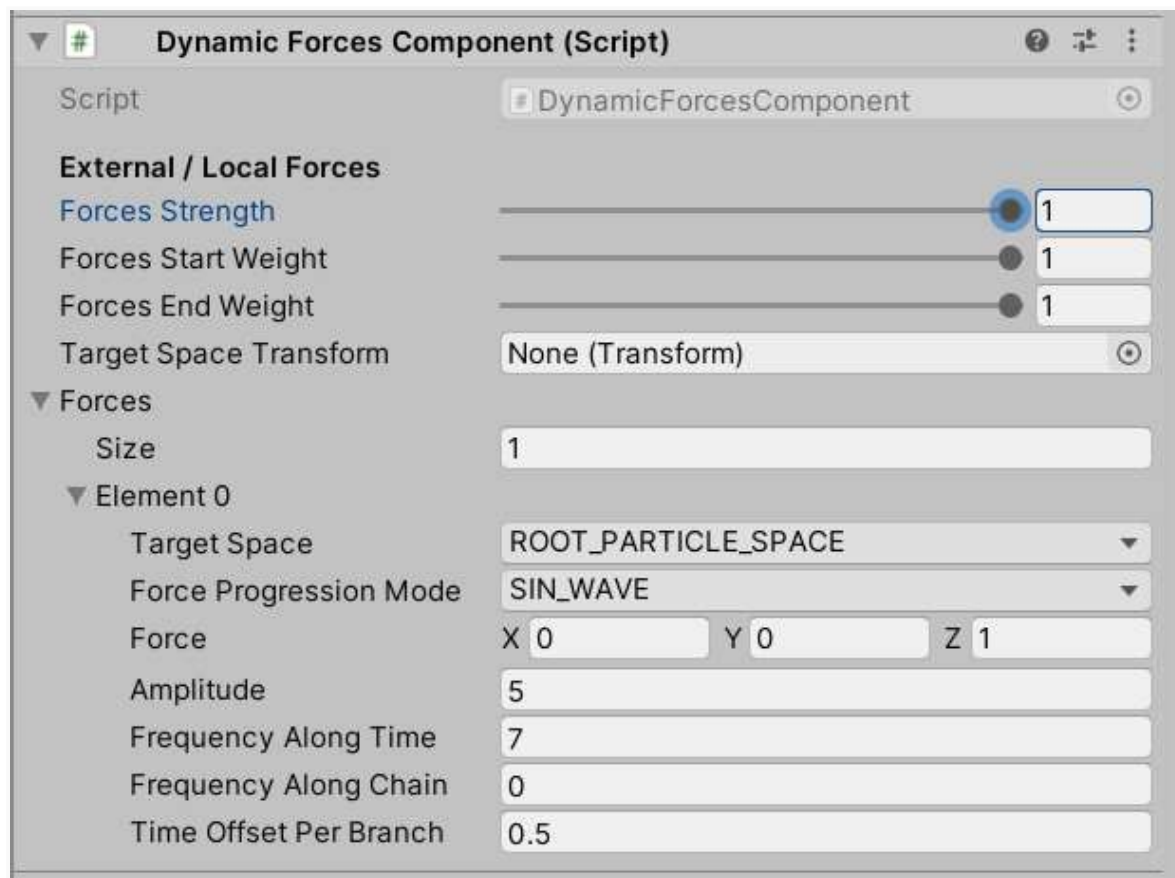
MIN_DISTANCE: Distance between the 2 particles can be greater than this distance value but clamped if less than this value

MAX_DISTANCE: Distance between the 2 particles can be less than this distance value but clamped if greater than this value.

Press the **Delete Constraint** button to remove this constraint.

DYNAMIC FORCES COMPONENT

In order to apply external layered forces onto particles in either the Dynamic Joint or Dynamic Chain components, add a Dynamic Forces Component onto the gameobject containing the Dynamic Joint or Dynamic Chain components. As long as they coexist on the same gameobject, the defined forces will be applied as layered forces on top of Gravity and Mass Inertia.



Forces Strength: The global strength multiplier of all the external forces defined.

Forces Start Weight: The force strength multiplier upon the particles at the root / start of Dynamic Chains.

Forces End Weight: The force strength multiplier upon the particles at end of Dynamic Chains. Strength is applied as a gradient from Start Weight to End Weight on particles based on their distance to the root of the chain.

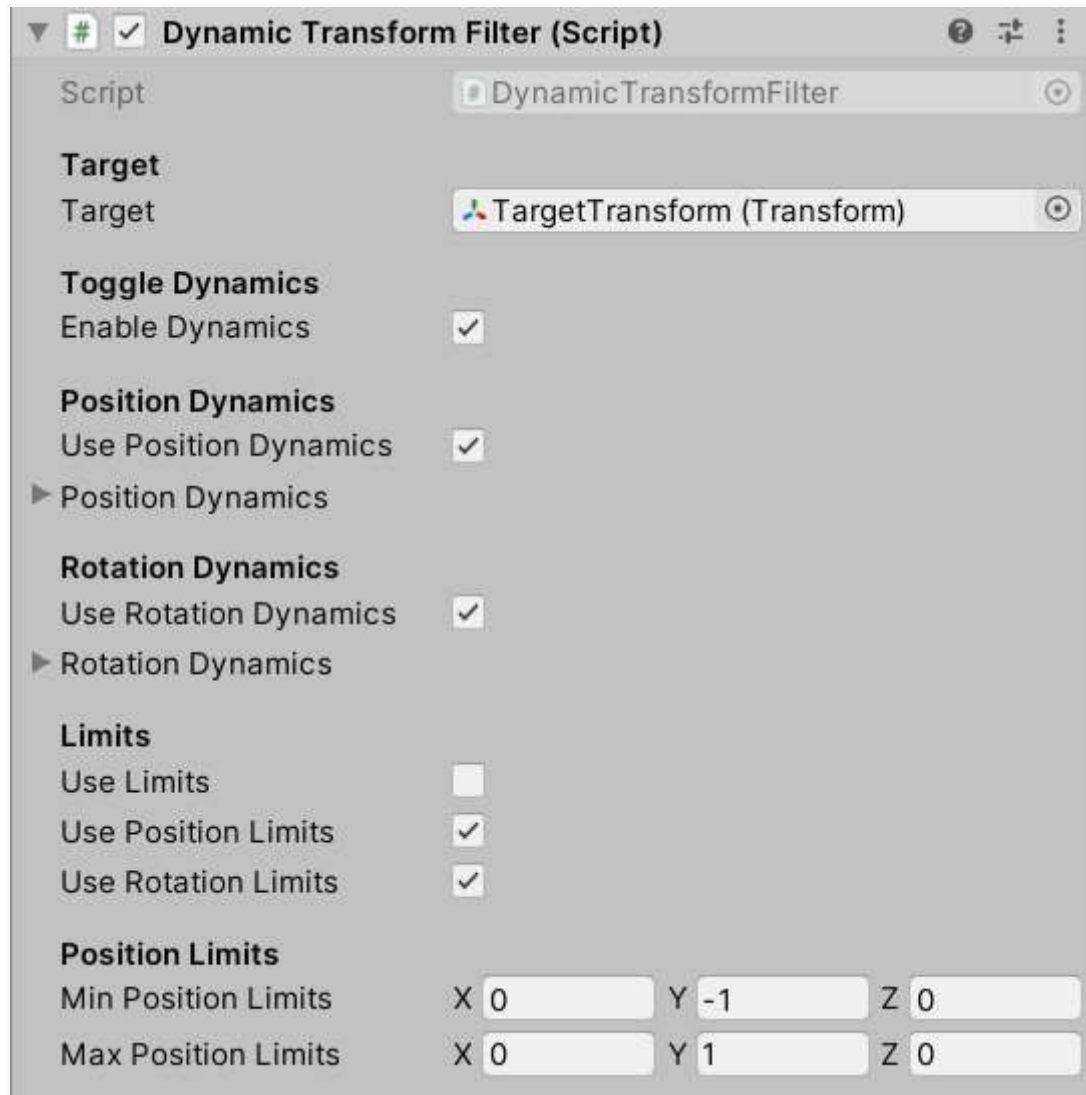
Forces: The list of forces to be applied.

Local Force Settings

- **Force:** the vector of the force
- **Target Space:** the space in which the force vector is applied:
 - **WORLD_SPACE:** Vector defined in world space
 - **TARGET_TRANSFORM_SPACE:** Vector defined in the local space of a transform defined in **Target Transform Space**
 - **ROOT_PARTICLE_SPACE:** Vector defined in the local space of the root particle gameobject
 - **ROOT_PARTICLE_PARENT_SPACE:** Vector defined in the local space of the root particle gameobject's parent
 - **PER_PARTICLE_SPACE:** Vector defined in the local space of each particle's gameobject
- **Force Progression Mode:** The progression mode of the force:
 - **CONSTANT:** apply force as a constant in the Force direction
 - **SIN_WAVE:** apply force as a Sin Wave along the Force direction
 - **PERLIN_NOISE:** apply force as a random bumpy noise along the Force direction
- **Amplitude:** The strength multiplier of the force
- **Frequency Along Time:** The frequency of the force wave over time for forces of SIN_WAVE progression mode.
- **Frequency Along Chain:** The frequency of the force wave along the particle chain hierarchy based on its distance from the root, for forces of SIN_WAVE progression mode. This helps to generate tentacle-like motion.
- **Time Offset Per Branch:** the time offset for forces of SIN_WAVE progression applied for a Dynamic chain with multiple branches to desync their motion. This helps to add random time offsets of sin wave motion per branch so that they don't all move in the same direction at the same time in sync.

DYNAMIC TRANSFORM FILTER

Move & Rotate a gameobject towards a target transform's position and rotation while simulating mass-spring dynamics to generate secondary motion & oscillations. Adjust positional and rotational behaviors separately by changing specific dynamics properties for each. Applies on the gameobject which contains the component.



Target: The target transform to move and rotate our gameobject towards while going through the dynamics filter.

Enable Dynamics: Toggle whether to enable dynamics to apply. If disabled, our gameobject position and rotation will snap to the target's transform without going through any dynamics filter hence without any secondary motion or oscillations.

Position Dynamics: The dynamics parameters which affect the translational aspect of the movement of our gameobject towards the target transform position. If **Use Position Dynamics** is disabled, the gameobject position will snap to the target transform position without being driven there via dynamics hence losing positional secondary motion.

Rotation Dynamics: The dynamics parameters which affect the rotational aspect of the rotation of our gameobject towards the target transform rotation. If **Use Rotation Dynamics** is disabled, the gameobject rotation will snap to the target transform rotation without being driven there via dynamics hence losing rotational secondary motion.

Use Limits: global flag for whether to use position or rotation joint limits at all.

Use Rotation Limits: flag for whether to use rotation limits. If enabled, any joint limits on our gameobject will apply.

Use Position Limits: flag for whether to use position limits. If enabled, the position deltas from the target transform position will be limited by the specified position limits.

Position Limits: The min & max position deltas from the target transform position will be limited by the specified **Min Position Limits** & **Max Position Limits**, specified in the local space of the target transform.

COLLIDERS

Dynamic Collider Native: Unity native colliders are supported by the Dynamic Chain component. Simply drag them into the Native Colliders list field. At runtime, a DynamicColliderNative component will automatically be added to these colliders as an intermediate representation. To modify their behaviors, you can manually add a Dynamic Collider Native component to a gameobject which contains a Collider component in advance. Here you can modify a few settings.

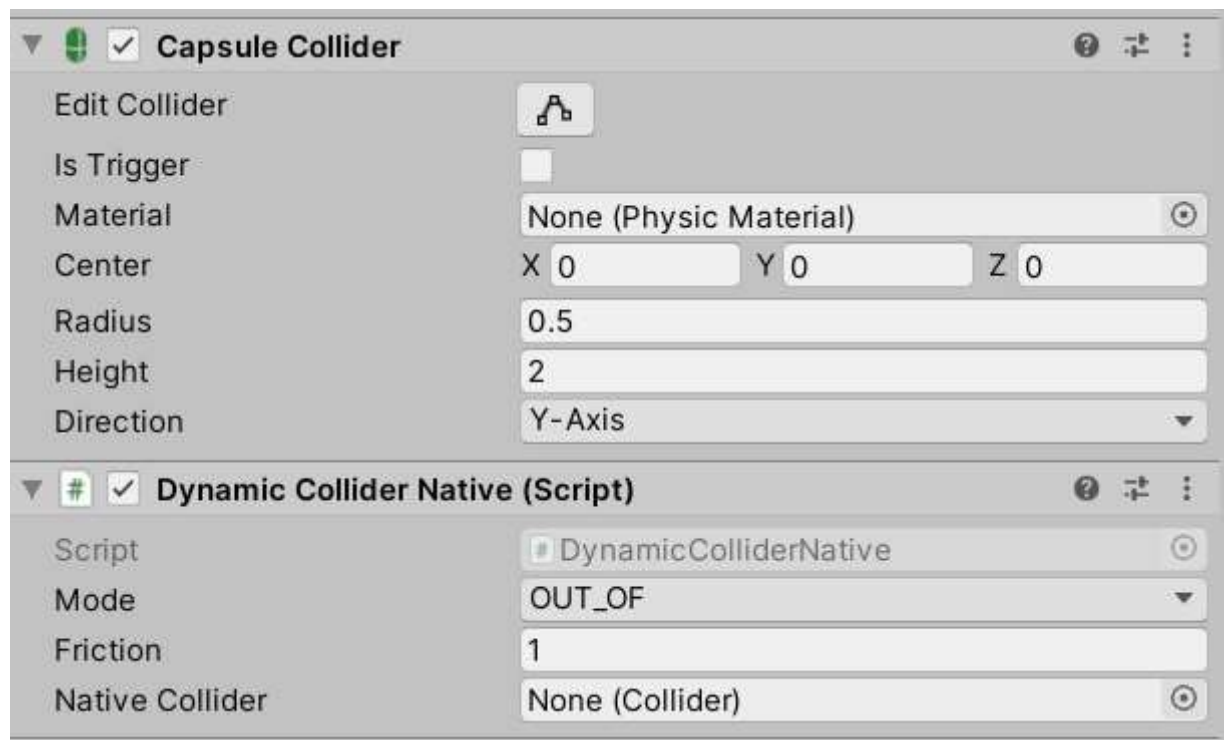
Mode: Choose a particle collision interaction mode:

OUT_OF (default): Particles should remain outside of the collider.

INTO: Particles should remain inside the collider.

ONTO: Particles should be projected onto the surface bounds of the collider at all times.

Friction: the surface friction multiplier to be applied to particles as they collide, which combined with the particle friction multiplier in Dynamic Chain, will affect the dynamic motion i.e. the damping as particles slide across the surface.



There are other Dynamic Colliders which can be setup on empty game objects and have their offsets and dimensions manually defined outside of Unity's native colliders. These include:

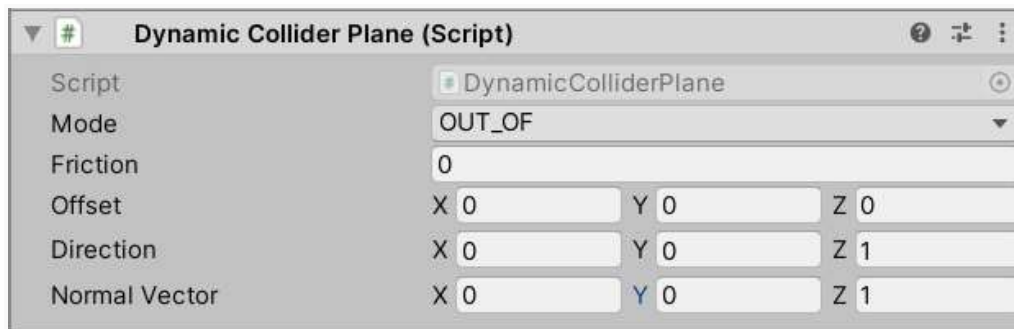
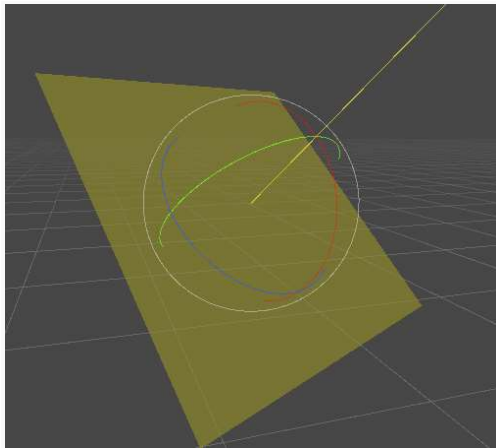
DynamicColliderSphere

DynamicColliderCapsule

DynamicColliderCone

DynamicColliderPlane

Plane colliders don't natively exist in Unity, therefore a **DynamicColliderPlane** can be added to a gameobject to represent one.



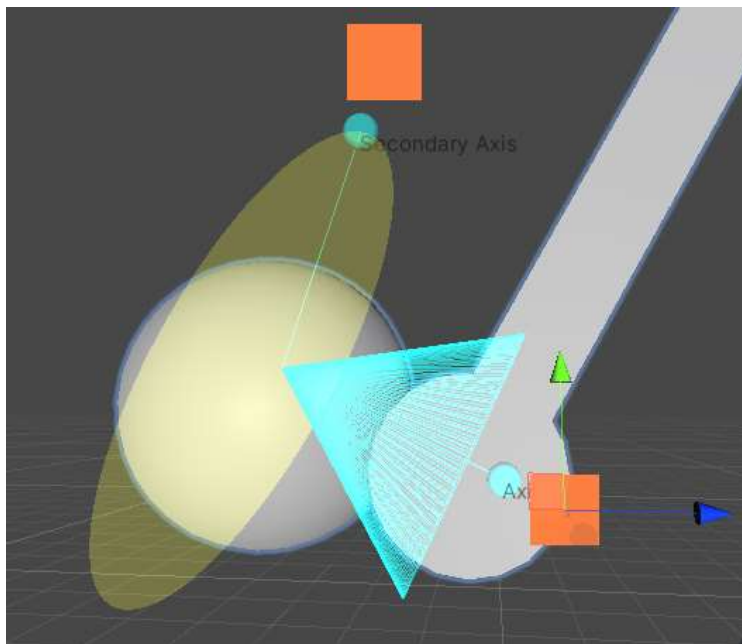
In this plane collider's case, a position **Offset** and a **Normal Vector** offset can be specified. They are bound to the gameobject's transform otherwise.

To add these colliders to a *Dynamic Chain*, drag the gameobjects which contain them as a component into the **Dynamic Colliders** list field instead of *Native Colliders*.

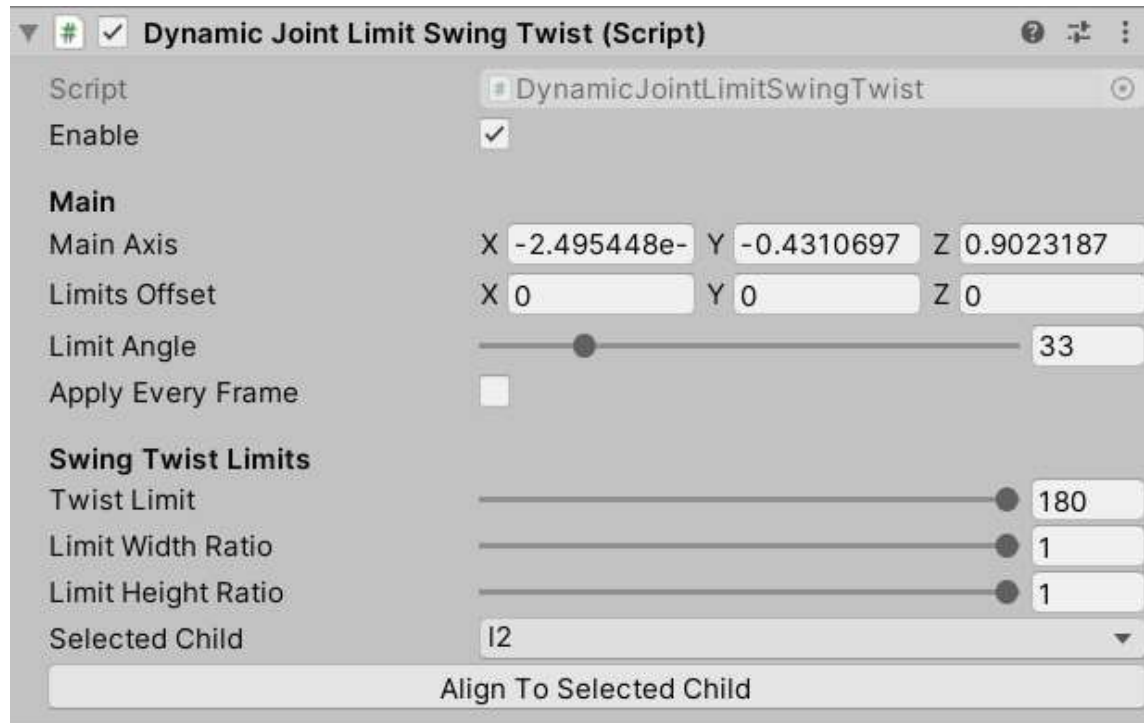
JOINT LIMITS

DYNAMIC JOINT LIMIT SWING TWIST:

Swing-Twist limits constrain joint rotations to a maximum angle along 2 axes of swing, and a maximum twist angle along an orthogonal twist axis. In order to add this joint limit, select the joint gameobject for which you wish to add the limit and add a **DynamicJointLimitSwingTwist** component onto it.



Use the Scene-View handlers to modify the direction of the **Main Axis** and its orthogonal secondary axis. The twist limit range is displayed as a yellow tinted arc along a circle-plane, the swing is displayed as a green cone.



Limit Angle sets the max swing angle from the Main Axis.

Twist Limit sets the max twist angle around the Main Axis.

Limit Offset sets a rotation offset for the joint limit relative to the main and secondary axes defined.

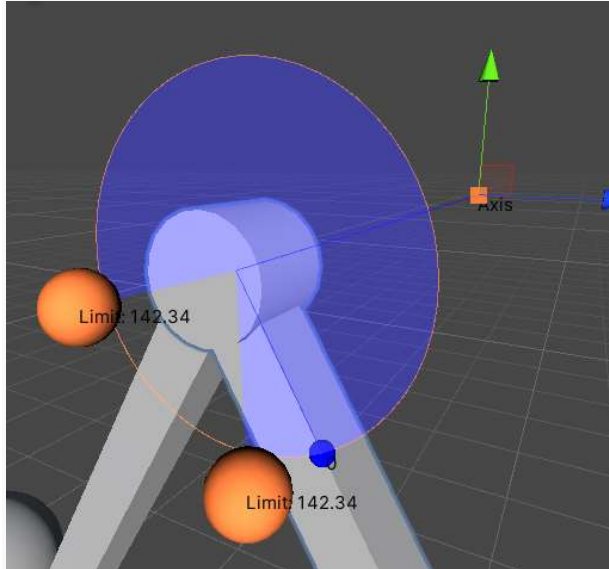
Limit Width Ratio / Limit Height Ratio set the ratios of maximum swing along the 2 dimensions of swing, creating an ellipsoid boundary for the swing instead of a circular boundary..

Apply Every Frame enforces joint limits in the LateUpdate step regardless of use within the dynamics tools, making these joint limits usable in other scenarios outside of this tool.

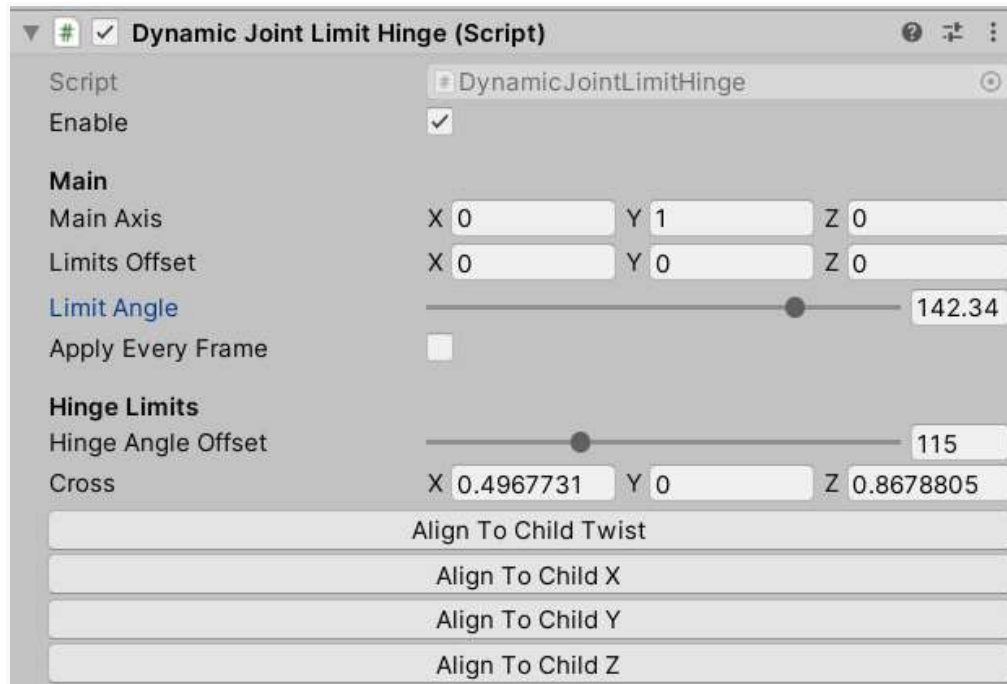
Align to Selected Child button aligns the main axis to point towards the **Selected Child** gameobject of its hierarchy to auto point it.

DYNAMIC JOINT LIMIT HINGE:

Hinge limits constrain joint rotations along a single axis, or *hinge*.



A **Main Axis** is defined similar to how it's defined in the swing-twist limit, modifiable directly with Scene-View handles. The limit angles in either direction can also be modified via scene view handles.



Limit Angle defines the maximum angle of rotation around the main axis.

Limits Offset defines a rotation offset of the limit relative to the main axis.

Hinge Angle Offset defines an offset along the main axis to limit the rotation from.

A number of buttons to auto-orient the hinge axis are available in editor, aligning either to a local axis X, Y or Z of the game object frame, or to align towards a child gameobject in its hierarchy.

Trailer

https://www.youtube.com/watch?v=4Wz_MLca36Y

Video Tutorials

https://www.youtube.com/watch?v=vqX_A1WQISc

<https://www.youtube.com/watch?v=2Gc1mjXh9VM>

<https://www.youtube.com/watch?v=qhrmB4omQ1Y>

<https://www.youtube.com/watch?v=gRNL6Xis3Hg>