Adaptive Hands

Make any rigged hand into an adaptive hand with bendable fingers that conform to colliders in the world.

V 1.2.3

Incomplete documentation parts will be improved over time.

- Get the most up to date documentation by clicking here.
- Remember you can hover over fields in the “Inspector” window in Unity’s editor to read tooltip explanations of each field.
- If you have any questions or need assistance email support at intuitivegamingsolutions@gmail.com.

Table of Contents

NOTE: See 'API Reference.pdf' (online) if you are looking for source code documentation.

How to: Import the Package into a Unity Project

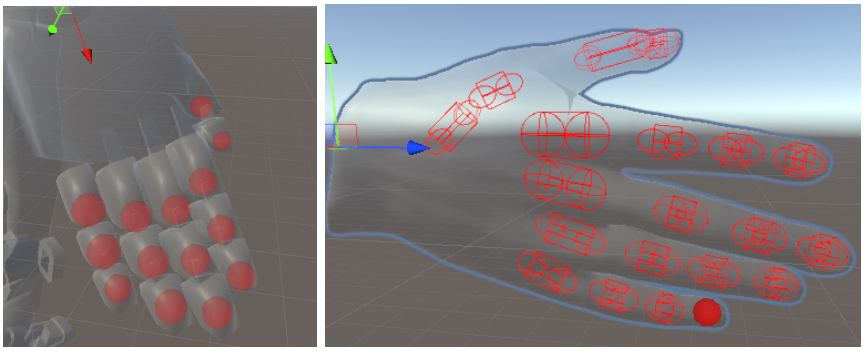
There are 2 ways to import the package.

a. (Recommended) Using the Unity Editor ‘Package Manager’.
    i. Open the Windows→Package Manager using the Unity editor toolbar.
    ii. In the upper-left corner of the Package Manager window select ‘Packages: My Assets’.
    iii. Search for “Adaptive Hands”’ in the list or use the search bar in the window.
    iv. Select the asset in the package manager, select ‘Download’.
    v. After the package has finished downloading click ‘Import’ to import it into the project.
b. Importing AdaptiveHands.unitypackage
    i. Using the Unity Editor’s toolbar select Assets→Import Package
    ii. In the file explorer that opens navigate to AdaptiveHands.unitypackage
    iii. Double click the package and import it.

Getting Started

3.a. Checking Out The Demos

- After importing the package simply load any one of the demo scenes to test it out.
- You can position any object with a Collider in the adaptive hands.
- You can visualize a hand's colliders by ensuring you have the appropriate gizmo enabled, such as KinematicHand, and selecting the hand GameObject or any finger.



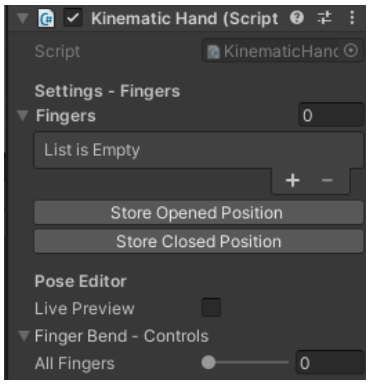
(left) The RobotKyle Humanoid demo hand in the 'Scene View' with all finger bone colliders of 0 length.
(right) The KinematicHand_Right demo hand in the 'Scene View' with all finger bone colliders of non-zero length except for the distal bone of the pinky.

- **The 'Kinematic Hand' demo** shows off a single hand that can be easily used for a visual hand in VR applications or similar uses.
- **The 'Humanoid' demo** shows off a humanoid character with both of its hands converted into adaptive hands allowing them to conform to colliders in the world.

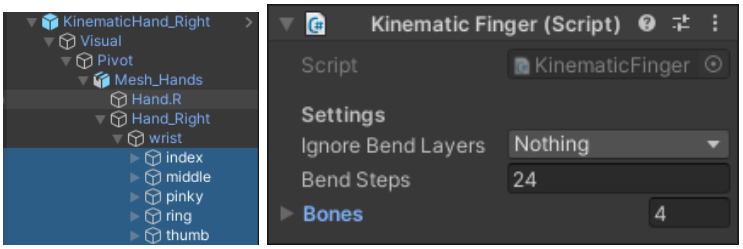
3.b. Configuring Custom Hands

(Youtube - Walkthrough)

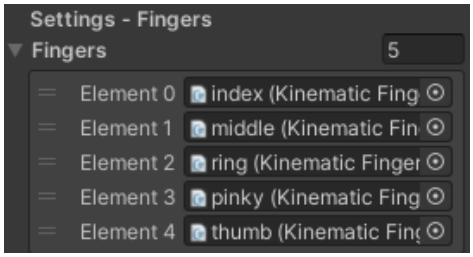
- ***If you want to set up 2 hands (for example the hands of a humanoid)*** it is easiest to only configure 1 hand and use the 'Hand Symmetry Tool'.
- ***Make sure to assign a 'Collider Length' of a non-zero value*** *to any finger bone that you want to use a capsule collision shape for instead of a sphere.*
- ***You can save time by editing finger bone colliders*** by using the 'Hand Collider Editor'.
- It is easy to configure your own custom hands for 'Adaptive Hands', just follow these steps:
    1. Navigate to the base GameObject of your hand(s) and attach a KinematicHand component. ***Add a component to all hands you want to configure,*** a KinematicHand component is needed even when using the 'Hand Symmetry Tool'.*.*



    2. Navigate to the top-most GameObject for each of your hand's fingers and attach a KinematicFinger component.

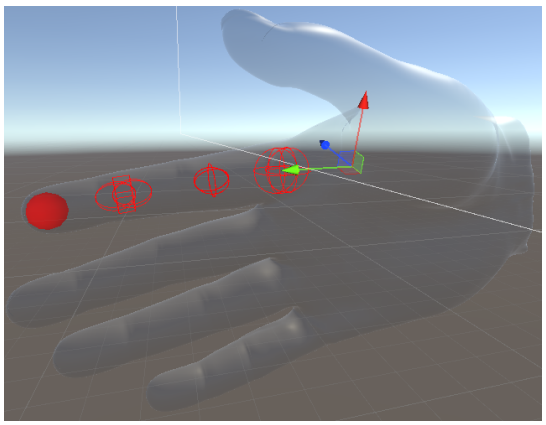


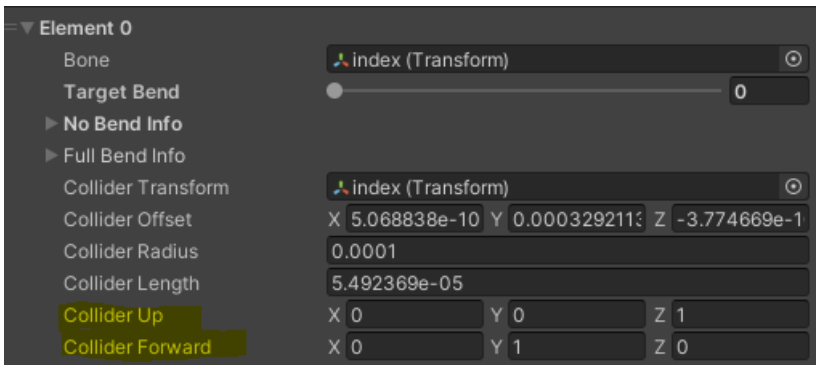
    3. Click and drag each of the top-most GameObjects for each of your hand(s) fingers into the 'Fingers' array of the KinematicHand component you added in step 1.



    4. Now select the top-most GameObjects for each of your hand's fingers) ***but this time 1 at a time (do not multi-select)*** to edit their bones and colliders.

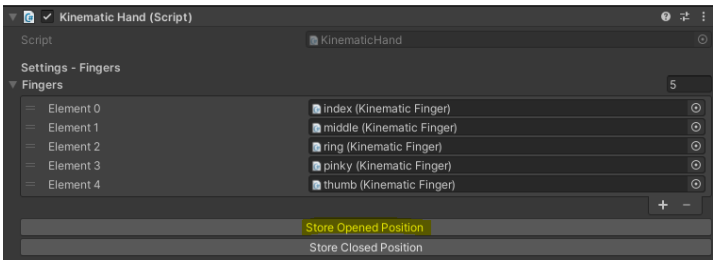
a. Add a bone entry for each bone that makes up your hand's finger to the 'Bones' array in the KinematicFinger component.
b. Drag each finger bone's GameObject into the appropriate 'Bone' and 'Collider Transform' fields and set up the settings for the finger bone.
    i. The top-most bone in the array ***must be the metacarpal (or proximal) bone*** and the last bone in the array must be the ***distal*** bone.
    ii. This is not necessarily required but the requirement is that the bones closer to the metacarpal appear in the 'Bones' array before the bones closer to the distal bones.
    iii. A good default 'Collider Radius' is 0.007.
    iv. Giving a length value to a finger bone collider will make it a capsule instead of a sphere.
    v. Giving a negative length value to a finger bone collider will invert it and make it into a capsule instead of a sphere.
    vi. The 'Collider Up' direction should match the axis that faces up from the knuckle in the Unity Editor.
    vii. The 'Collider Forward' direction should match the axis that goes along the bone in the Unity Editor.
    viii. You can now use the '[Hand Collider Editor]' to modify the hands colliders. *For finger bones that you want to use a capsule collision shape on make sure you assign them a non-zero length in the KinematicFinger inspector first.*





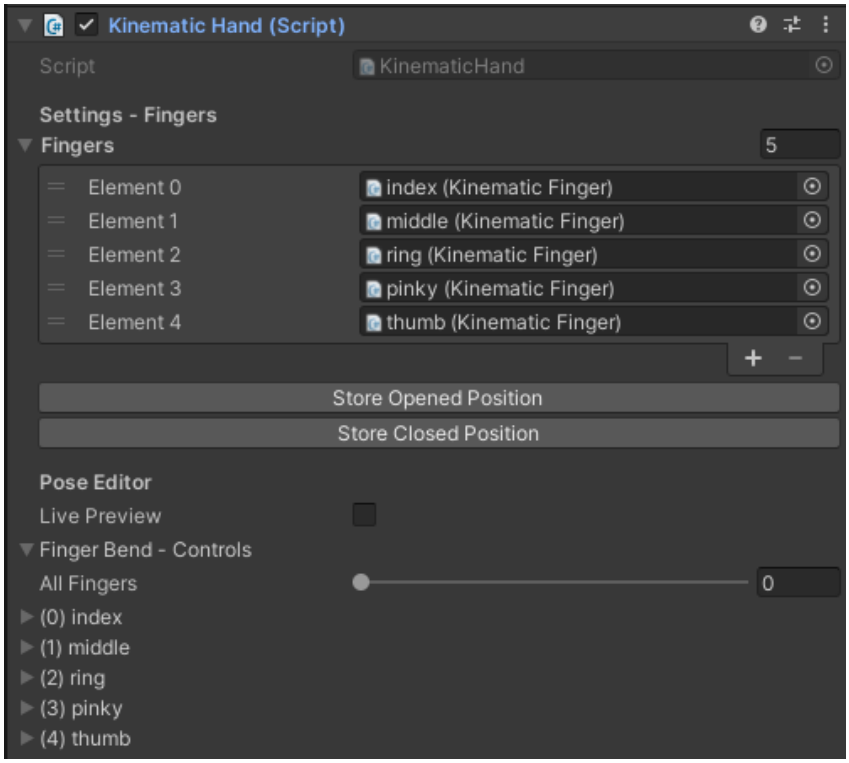
- Notice how the 'Collider Up' direction is '0, 1, 0' meaning forward in the y direction. As you can see this direction matches the green arrow in the Unity Editor that runs along the 'index.001' finger bone.

5. Finally before your hand is ready to use you must configure both the opened and closed position for it.
    a. While your hand is in a neutral position, navigate to the relevant hand component, such as KinematicHand, and select the 'Store Opened Position' button.



    b. Position the fingers on your hand as described in the 'Configuring Opened & Closed Positions' section of the documentation before selecting the 'Store Closed Position' button.
    c. Your hand is now fully configured and ready to use!

The KinematicHand Component



A screenshot of the KinematicHand component 'Inspector' pane in the Unity editor.

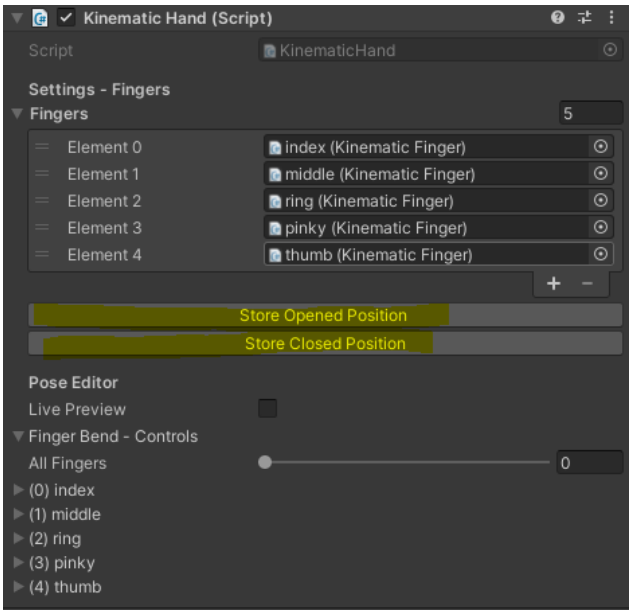
4.a. Description & Breakdown

- The KinematicHand component is the 'top-most' core component of 'Adaptive Hands'. It is responsible for managing bendable fingers.
- You can set the 'unbendObstructed' KinematicFinger booleans for all fingers that make up the hand at the same time using KinematicHand.SetUnbendObstructed(bool).
- Fields:

Field Name	Description	Default Value
Fingers	- An array of KinematicFingers that make up the hand.	new KinematicFinger() *(KinematicFinger[])*

4.b. Configuring Opened & Closed Positions

- The buttons that allow you to store a new opened or closed position for your adaptive hands can be found in the hand component itself, for example in the 'Inspector' pane for the KinematicHand component as shown in the screenshot below:

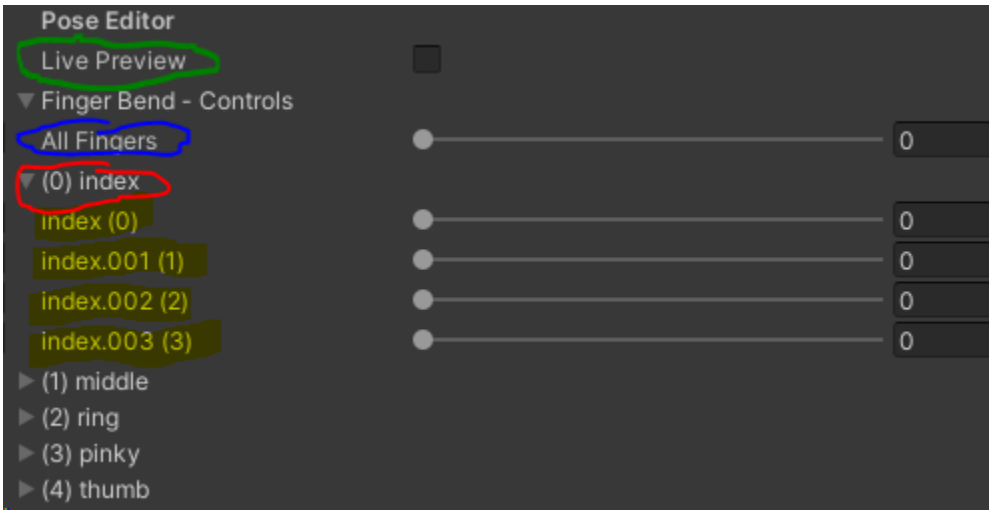


- The **Store Opened Position** button will save the current bend state for all bones in the 'Fingers' array of the hand as the 'opened' state for the hand.
    - A confirmation menu will appear after clicking this button asking if you are sure you want to overwrite the opened position of your hand.
    - An undo entry is registered for this operation.
- The **Store Closed Position** button will save the current bend state for all bones in the 'Fingers' array of the hand as the 'closed' state for the hand.
    - A confirmation menu will appear after clicking this button asking if you are sure you want to overwrite the opened position of your hand.
    - An undo entry is registered for this operation.
- To position your hands fingers it is recommended you follow these steps:

The best results come from positioning the thumb bone separately.

1. Select all *distal* finger bones (finger tips), bend them all at once to the desired position.
2. Select all *intermediate* finger bones (middle bones), bend them all at once to the desired position.
3. Select all *proximal* finger bones (between 1st and 2nd knuckle), bend them all at once to the desired position.
4. Select all *metacarpal* (between wrist and 1st knuckle) finger bones, bend them all at once to the desired position.
5. Position the thumb.
6. Tweak any finger bones you want to tweak and either click 'Store Opened Position' or 'Store Closed Position' and confirm depending on which position you want to store. If you accidentally overwrite a position do not worry! Simply use 'Undo'.
7. You can now preview your hand across all bend states by enabling 'Live Preview' in the 'Pose Editor' and modifying the 'All Fingers' bend value.

4.c. The 'Pose Editor'

- The 'Pose Editor' is simply built-in to the 'Inspector' pane of several components including KinematicHand and HandPoser.
- The 'Pose Editor' allows you to ***modify the bend value (between 0 and 1) for all finger bones that make up the hand.***
- Here is a partial breakdown of a screenshot of the 'Pose Editor' from the hand in the provided Demo_AdapativeHands_KinematicHand00 demo scene:
  - 

  - ***Live Preview*** **-** Controls whether or not this hands bend state is simulated in edit mode (while the game is not in play mode)
  - ***All Fingers*** **-** A slider that allows all finger bones to be controlled at the same time.
  - ***(0) index*** **-** A dropdown that holds all of the bones for this finger.
  - ***index (0), index.001 (1)*****, etc -** Sliders that control the target bend (0 to 1) for the respective KinematicFinger.Bones.
- Modifications made to any slider register undo operations.
- Multiple hands can have 'Live Preview' enabled at the same time.
- It is important to ***remember to keep 'Live Preview' mode turned on*** when editing poses in edit mode, otherwise the hand will not attempt to reach its bend targets unless in play mode.

The KinematicFinger Component

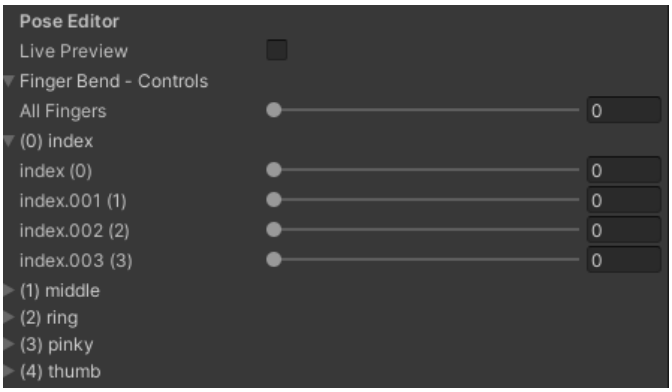
5.a. Description & Breakdown

- The KinematicFinger component describes a single finger of a KinematicHand and contains the definitions for each bone that makes up the finger and the collision information, and other relevant information, relating to it.
- Fields:

Field Name	Description	Default Value
Unbend Obstructed	- Controls whether or not the finger dynamically unbends when an obstruction comes into its bend path.	true *(Boolean)*
Ignore Bend Layers	- A LayerMask of layers to ignore when detecting collisions for this finger.	new LayerMask() *(LayerMask)*
Bend Steps	- The number of bend steps that a full 'bend' from 0 to 1 uses.	24 *(int)*
Bones	- An array of KinematicFinger.Bones that make up the finger.	new KinematicFinger.Bone() *(KinematicFinger.Bone[])*
Finger Move Rate	- The maximum speed the finger's bones can move with in units per second. (units/sec)	0.1 *(float)*
Finger Rotate Rate	- The maximum speed the finger's bones can rotate with. Given in degrees per second. (degs/sec)	360 *(float)*

5.b. Bend Sliders

- Bend sliders can be used to control the target bend value for any finger bone during runtime or in edit mode.
    - Remember that in 'Edit Mode' you must have 'Live Preview' enabled in order for bend targets to be visible without being in play mode.
- The 'Pose Editor' that can be found in multiple components such as the KinematicHand component and the HandPose component is where 'Bend Sliders' are found.
- As seen in the screenshot below:
    - There is an 'All Fingers' slider that allows all finger bone bend targets to be controlled at the same time.
    - There is a dropdown category for each KinematicFinger that makes up a KinematicHand.
    - There is a 'Bend Slider' for every bone that makes up a finger.

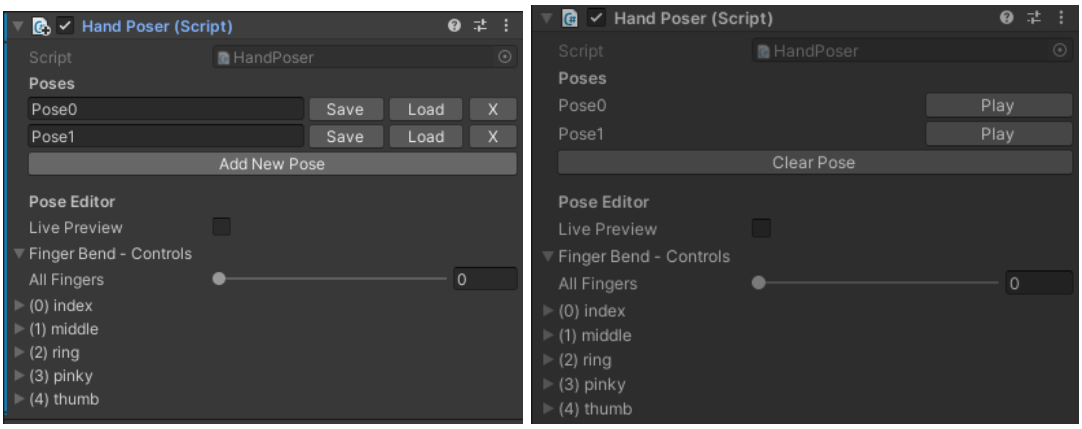


A screenshot showing the 'Pose Editor' where "Bend Sliders' can be found.

- Assuming the hand is in no pose when entering play mode the hand will maintain the bend targets specified in the 'Pose Editor' during edit mode.
- If you want to save a pose (save the bend targets for all finger bones in a hand) you can use the HandPoser component to do it.

The HandPoser Component

6.a. Description & Breakdown



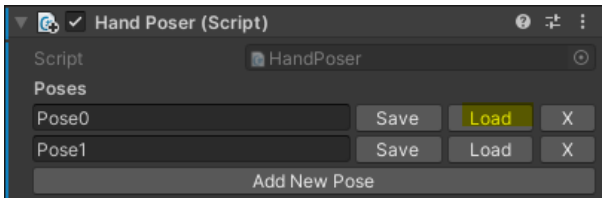
*(left) A screenshot showing the HandPoser components 'Inspector' pane **as seen in edit mode**.*
*(right) A screenshot showing the HandPoser components 'Inspector' pane **as seen in play mode**.*

- The HandPoser is a component responsible for saving and loading preset finger bend states (poses) for a given hand.
- The 'Poses' list shows a list of all poses currently saved for your hand.
    - It is important to note that poses are specific to a single hand and may not be copied between hands.
- The 'Poses' list **in edit mode:**
    - The text field is where you enter the ***case-sensitive name*** for the pose.
    - The 'Save' button saves the current hand bend state (all individual finger bend states) under the pose corresponding to the button you clicked.
        - This button will ask for confirmation before carrying out the action.
    - The 'Load' button loads the saved pose setting all finger bend targets in the hand to those stored in the loaded pose.
        - This button will ask for confirmation before carrying out the action.
    - The 'X' button deletes the corresponding pose.
        - This button will ask for confirmation before carrying out the action.
- The 'Poses' list **in play mode:**
    - The 'Play' button sets the current pose for the HandPoser to the pose that corresponds to the button you clicked.
        - Poses may also be modified using HandPoser.SetPose(string pPoseName) or many other methods found in the scripting API.
    - The 'Clear Pose' button only appears if 'HandPoser.CurrentPoseIndex != HandPoser.POSE_NONE'.
        - The 'Clear Pose' button resets HandPoser.CurrentPoseIndex back to andPoser.POSE_NONE essentially putting the HandPoser into 'sleep' mode as it will no longer override the hands pose.
        - The current hand pose can also be cleared using the scripting API method HandPoser.Clear().

- The 'Add New Pose' button will automatically add a new pose named 'Pose #' with the # being the index of the newly added pose.
    - The newly added pose will have the hand's current bend states automatically saved in it.
- The 'Pose Editor' section is the same as the one shown in the KinematicHand component. Refer to this section of the documentation for more information.
- An ***important note*** to remember:
    - ***Loading a pose in edit mode*** simply ***overrides the current finger bone bend states*** with the values loaded from the relevant pose.
    - ***Playing a pose in play mode*** sets the HandPoser.CurrentPoseIndex field which ***forces the finger bone bend states*** to the values loaded from the relevant pose ***every frame***.

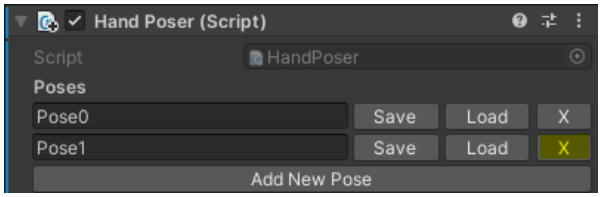
6.b. Editing Poses

- Remember when editing poses you should have 'Live Preview' mode turned on so the hand bends while not in play mode.
- You can use the 'Load' button to ***load an existing pose*** to start working from that point.



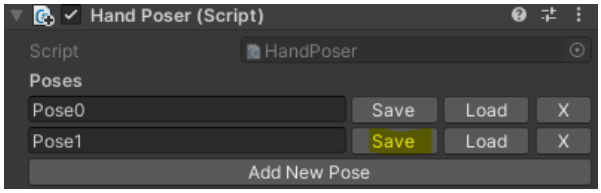
Pressing the highlighted 'Load" button would overwrite the current finger bone bend states with the values loaded from 'Pose0'.

- You may effectively ***duplicate a pose*** by first loading the source pose and then clicking 'Save' on your destination pose, or simply creating a new pose after loading the source pose.
- You may ***delete an existing pose*** by using the 'X' button.



Pressing the highlighted 'X' button would delete 'Pose1'.

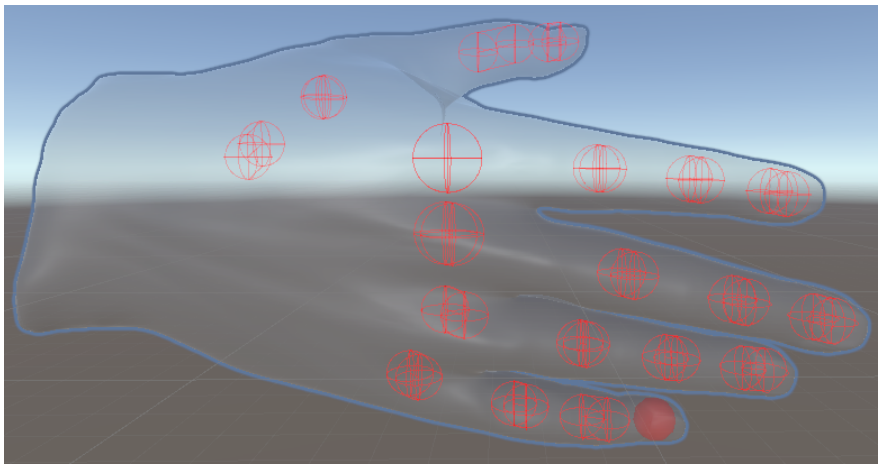
- To ***save a modified pose*** use the 'Pose Editor' to modify the finger bone bend values to your desired states and then click the 'Save' button corresponding to the pose you want to save in, for example to save in 'Pose 1' in the screenshot below I would click the highlighted button.



Pressing the highlighted 'Save' button would save the current finger bone bend values into 'Pose1'.

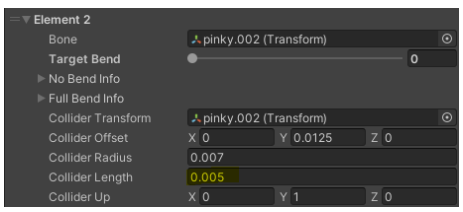
Hand & Fingers - Visual Breakdown

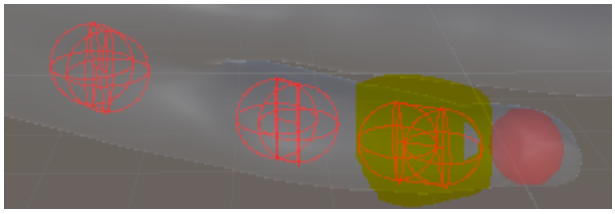
7.a. Visualizing Hand & Finger Colliders



A screenshot showing the Editor visualization of an adaptive hand.

- The wireframe red capsules represent finger bones with a length != 0.

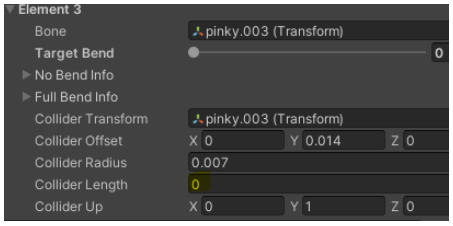


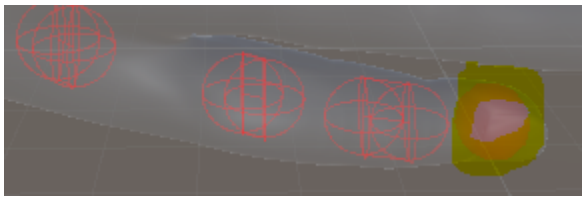


(left) A screenshot showing a finger definition with a length != 0.
(right) A screenshot showing the highlighted capsule representing the finger bone defined in the left screenshot.

- Bones with a length not equal to 0 detect collisions using capsules. This is slower than sphere collision detection but in general the quality of collision detection is better.
- The radius of the wireframe spheres at the tips of the capsule represent the radius of the finger bone.

- The solid red spheres represent finger bones with a length == 0.



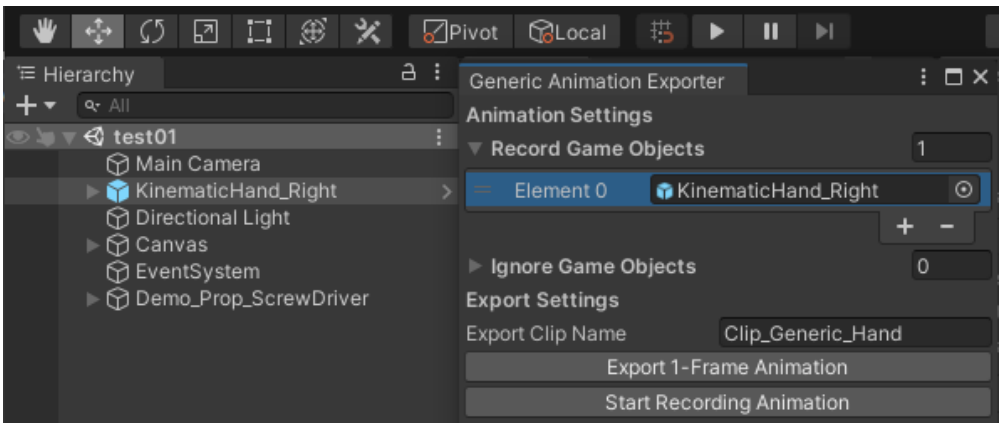


(left) A screenshot showing a finger definition with a length == 0.
(right) A screenshot showing the highlighted sphere representing the finger bone defined in the left screenshot.

- Bones with a length == 0 detect collisions using spheres. This is faster with better performance but in general the quality of collision detection is worse.
- The radius of the solid sphere visualizes the radius of the sphere collider for the finger bone.

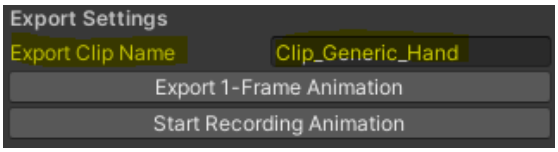
Generic Animation Exporter

- The 'Generic Animation Exporter' allows you to export single or multi-frame animations for any GameObjects in your scene including adaptive hands.
- To record a Generic AnimationClip using 'Adaptive Hands' simply follow these steps:
    1. Navigate to 'Tools → Adaptive Hands → Generic Animation Exporter' and drag any GameObject you want to record from your scene's 'Hierarchy' pane into the 'Record Game Objects' field.



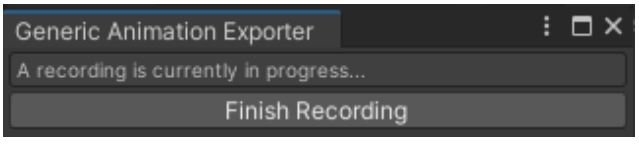
A screenshot showing the 'Hierarchy' pane next to the 'Generic Animation Exporter' window with the KinematicHand_Right GameObject added to the 'Record Game Objects' array.

    2. Next add any ***child objects*** of 'Record Game Objects' that you do not want to be recorded to the 'Ignore Game Objects' list. *You may skip this step if you do not want to ignore any GameObjects.*
    3. Enter the name of the animation clip you want to export under the 'Export Settings' category.



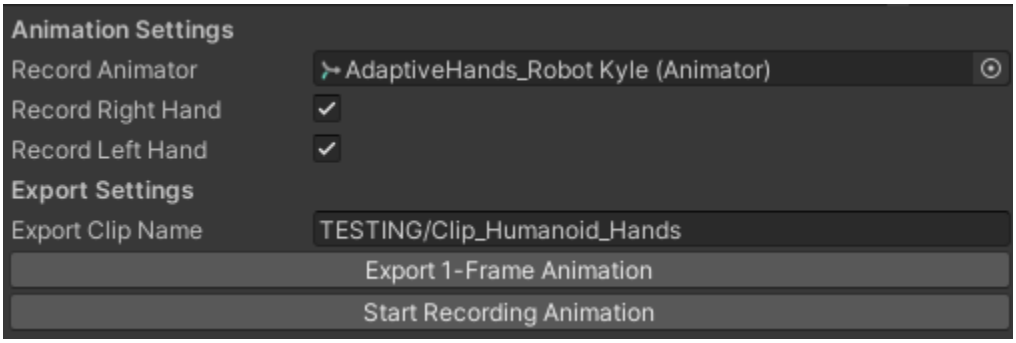
A screenshot showing the 'Export Clip Name' field in the 'Generic Animation Exporter' window..

        - Note that you may change the export path for exported animation clips by going to 'Tools → Adaptive Hands → Editor Settings'. The default directory is *'Assets/'*.
    4. Click one of the two options to record your desired animation:
        a. **Export 1-Frame Animation** - exports a generic AnimationClip with only a single frame, the current Transform properties of the 'Record Game Objects' specified and their children.
        b. **Start Recording Animation** - begins recording a generic AnimationClip with the Transform properties of the 'Record Game Objects' specified and their children each frame until '**Finish Recording**' is clicked.



A screenshot showing the 'Generic Animation Exporter' window while recording an animation.

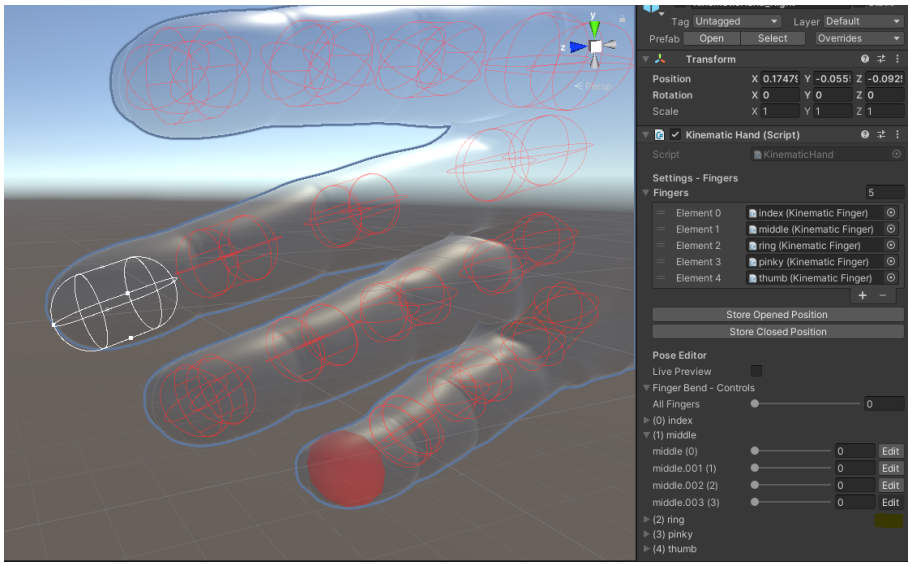
Humanoid Animation Exporter



- Steps to exporting a humanoid animation:
    1. Simply drag a humanoid Animator (with an Avatar referenced) into the 'Record Animator' field.
    2. Check off the boxes for whichever hand(s) you want to record.
    3. Click 'Export 1-Frame Animation' to export a single-frame (hand pose) animation, otherwise click 'Start Recording Animation' to begin recording a multi-frame (hand pose) animation.
    4. If you started a multi-frame animation click 'Stop Recording Animation' when you are done recording your animation.
    5. You can freely modify your exported animation in the Unity 'Animation' window.

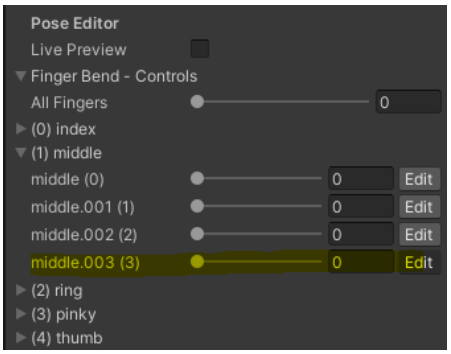
The Hand Collider Editor

(Youtube Demo)



A screenshot of the middle finger tip in 'Edit' mode using the hand collider editor.

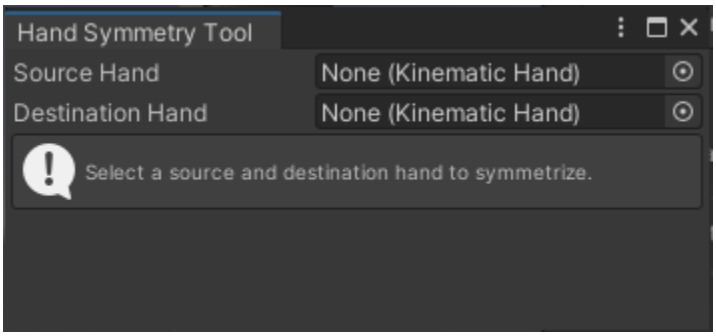
- The 'Hand Collider Editor' is an easy-to-use tool that allows you to quickly size and place your finger bone colliders where you need them using handy visual handles in the scene view.
- The 'Hand Collider Editor' can be used by going to the 'Pose Editor' in your KinematicHand component's 'Inspector' pane and clicking the 'Edit' button beside any finger bone's bend slider.



A screenshot of the 'Hand Collider Editor' settings in the Pose Editor section of the KinematicHand component's 'Inspector' pane..

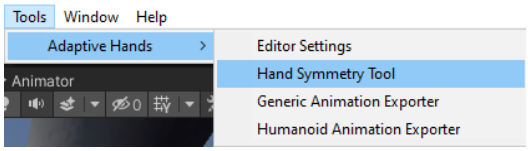
The Hand Symmetry Tool

(Youtube - Hand Symmetry Tool)

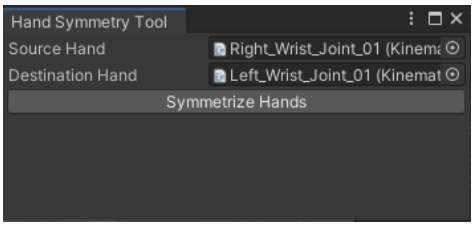


A screenshot of the 'Hand Symmetry Tool' window.

- The 'Hand Symmetry Tool' can be accessed by navigating to Tools → Windows → Help in the Unity Editor toolbar.



- To use the 'Hand Symmetry Tool' follow these steps:
    1. Click and drag the ***hand that is fully configured*** into the ***Source Hand*** field of the editor window.
    2. Ensure the hand that is not configured (the destination hand) has a KinematicHand component attached to it.
    3. Click and drag the ***hand that is not configured*** into the ***Destination Hand*** field of the editor window. After this step the 'Symmetrize Hands' button should appear.



*A screenshot showing a fully filled out 'Hand Symmetry Tool' window.**

    4. Click the '***Symmetrize Hands'*** button.

You are now done configuring your symmetric hand!

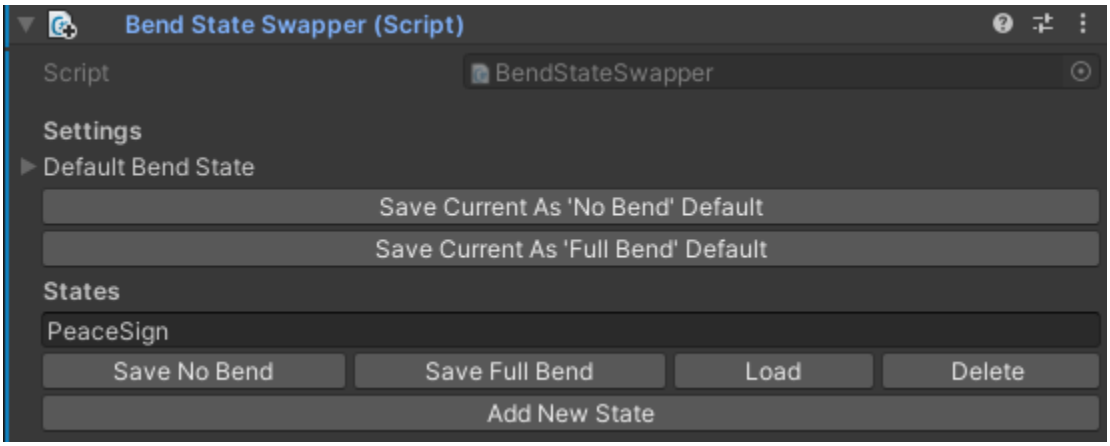
Issues?

- Check console for logs messages and warnings, if the editor was unable to find symmetric limbs send the console logs to us so we can add any unsupported naming conventions to the system.

- Alternatively you can code a custom editor script that adds (an) entry(s) to EditorSystemUtility.SYMMETRY_TEXT_REPLACEMENT_LOOKUP to manually convert your custom type. The syntax is as follows:

```
// new EditorSystemUtility.ReplacementEntry(leftText, rightText, replaceMode)
EditorSystemUtility.SYMMETRY_TEXT_REPLACEMENT_LOOKUP.Add(new
EditorSystemUtility.ReplacementEntry(
   "my_left_bone", // The 'leftText' bone name to replace with 'rightText' when flipping from left symmetry.
   "My_right_bone", // The 'rightText' bone name to replace with 'leftText' when flipping from right
symmetry.
   EditorSystemUtility.ReplacementMode.Replace // Normal replacement, replaced using text.Contains().
));
```

The BendStateSwapper Component

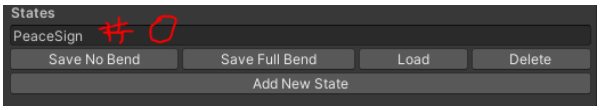


A screenshot showing the 'Inspector' pane of the BendStateSwapper component in the Unity Editor.

- The BendStateSwapper component allows you to save and load 'full' and 'no' bend states for your adaptive hand and swap between them in real time while your application is running or in the editor.
- This component allows you to create more complex poses without adding extra complexity to the KinematicHand component.
- This component must be attached to the same GameObject as a KinematicHand component as it will drive the adaptive hands bend states..
- Check out this Youtube demo that shows the component in action.
- After adding this component it will automatically inherit the default 'no bend' and 'full bend' information from the relevant KinematicHand component.
- **In Edit Mode:**
    - The '**Save Current As 'No Bend' Default**' button saves the current hand state as the default 'No Bend' state.
    - The '**Save Current As 'Full Bend' Default**' button saves the current hand state as the default 'Full Bend' state.
    - The '**Add New State**' button adds a new state named "State#" to the 'States' list. The new state will automatically inherit the default bend state settings from this components 'Default Bend States' setting.
    - The '**Save No Bend**' button that appears under the name of a state allows you to save the current hand state as the 'No Bend' state for the relevant state.
    - The '**Save Full Bend**' button that appears under the name of a state allows you to save the current hand state as the 'Full Bend' state for the relevant state.
    - The '**Load**' button that appears under the name of a state allows you to load a bend state overriding the 'no bend' and 'full bend' states of the relevant KinematicHand with those of the loaded state.
    - The '**Delete**' button allows a state to be deleted.
- **In Play Mode:**

- The '**Load**' button that appears to the right of each state's name may be used to load a state while the application is running.

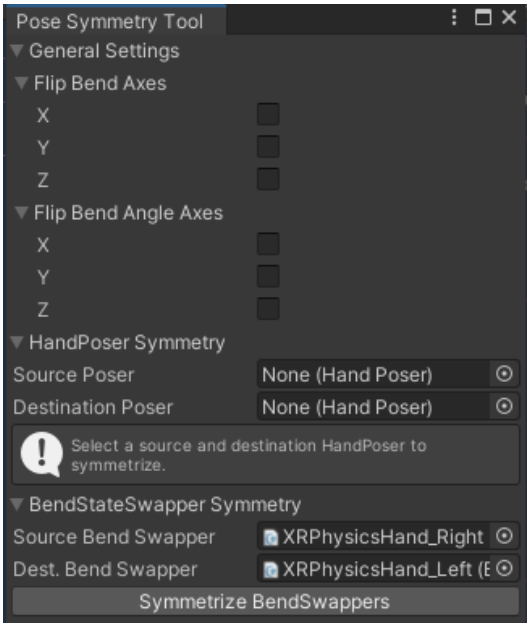
- **Via The Scripting API:**
  - BendStateSwapper.SetState(string pStateName)
    - This method allows you to set the bend swap state by its name.
  - BendStateSwapper.SetStateByIndex(int pIndex)
    - This method allows you to set the bend swap state by its index. The index can be determined by the order it shows up in the components 'Inspector' starting from 0.



  - BendStateSwapper.ClearState()
    - This method clears any bend swap state, restoring the default state.

The Pose Symmetry Tool

(Youtube Demo)



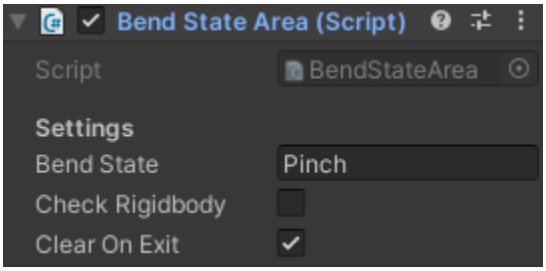
- Allows you to easily copy HandPoser or BendStateSwappers from a hand to a symmetrical hand.
- Simply navigate to 'Tools → Adaptive Hands → Pose Symmetry Tool' in the Unity Editor toolbar and use the easy to understand menu to symmetrize HandPoser or BendStateSwapper components.
- Check out this youtube walkthrough for an example video on how to use the 'Pose Symmetry Tool'.

Extras

14.a. Grab System

- A free extension included with 'Adaptive Hands' that implements advanced grabbing and throwing mechanics.
- Simply double click the 'Extra_GrabSystem.unitypackage' file to import the grab system into your project.
- Documentation
- API Reference

14.b. The BendStateArea Component



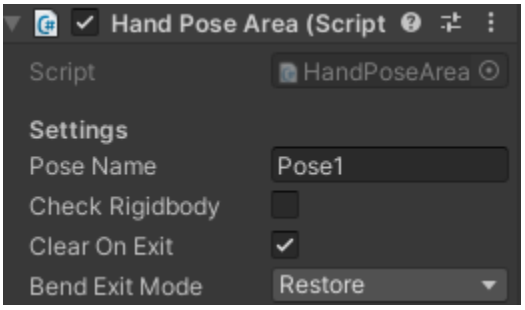
A screenshot of the BendStateArea component 'Inspector' pane in Adaptive Hands v1.2.0

- Sets any enabled BendStateSwapper who enters some trigger's bend state by name.
- The 'BlockBendStateSwapDelegate' event may be used to block the bend state swap by flipping the third 'ref bool' argument to false.
-

Setting	**Description**	**Default Value**
Bend State	A string that represents the name of the bend state which any BendStateSwapper who triggers 'OnTriggerEnter()' should be put into.	"" *(string)*
Check Rigidbody	Should colliders that cause an 'OnTriggerEnter()' event be checked if no BendStateSwapper component is found on them directly and they have a valid 'attachedRigidbody' reference? WARNING: This may conflict with 'Check Rigidbody' as having child colliders leave will clear the bend state even if other colliders in the Rigidbody compound collider remain in the area.	false *(bool)*
Check Trigger Stay	An option that allows for triggers to be considered as 'entering' the pose during	false *(bool)*

	OnTriggerStay(). Comes with some performance impact but allows the hand pose area to trigger hands that become eligible while already inside the pose area.	
Clear On Exit	Should the bend state be cleared when the collider (or a compound collider in the case of a Rigidbody compound collider with 'checkRigidbody' enabled) exits the trigger? WARNING: This may conflict with 'Check Rigidbody' as having child colliders leave will clear the bend state even if other colliders in the Rigidbody compound collider remain in the area.	true *(bool)*

14.c. The HandPoseArea Component



A screenshot of the HandPoseArea component 'Inspector' pane in Adaptive Hands v1.2.0

- Sets any enabled HandPoser who enters some trigger's pose by name.
- The 'BlockHandPoseDelegate' event may be used to block the hand pose area by flipping the third argument 'ref bool' to true.
-

Setting	**Description**	**Default Value**
Pose Name	A string that represents the name of the pose that any HandPoser who triggers 'OnTriggerEnter()' should be put into.	"" *(string)*
Check Rigidbody	Should colliders that cause an 'OnTriggerEnter()' event be checked if no HandPoser component is found on them directly and they have a valid 'attachedRigidbody' reference? WARNING: This may conflict with 'Clear On Exit' as having child colliders leave will clear the pose even if other colliders in the Rigidbody compound collider remain in the area.	false *(bool)*
Check Trigger Stay	An option that allows for triggers to be considered as	false *(bool)*

	'entering' the pose during OnTriggerStay(). Comes with some performance impact but allows the hand pose area to trigger hands that become eligible while already inside the pose area.	
Clear On Exit	Should the pose be cleared when the collider (or a compound collider in the case of a Rigidbody compound collider with 'checkRigidbody' enabled) exits the trigger? WARNING: This may conflict with 'Clear On Exit' as having child colliders leave will clear the pose even if other colliders in the Rigidbody compound collider remain in the area.	true *(bool)*
Bend Exit Mode	What to do with the fingers 'bend' states when exiting the area. **None - Do nothing on exit.** **Zero -** Zero on exit. **Restore -** Cache on enter, restore on exit.	BendExitMode.Restore *(HandPoseArea.BendExitMode)*

FAQ

(Frequently Asked Questions)

***Q:* The hands position is not changing while in edit mode.**
A: You have likely forgotten to enable 'Live Preview' mode.

- 'Live Preview' mode can be enabled through the KinematicHand or HandPoser component 'Inspector' pane.

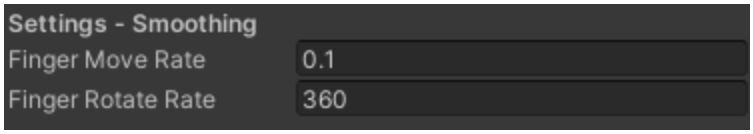
- 'Live Preview' can be activated for multiple hands at the same time.

***Q:* Can I set my hand colliders up so there are no gaps?**
A: You can absolutely set your hand collider sup so there are no gaps in between finger bone colliders however this does not always produce the most visually appealing hand results – it is usually preferable to find a balance that leads to the best hand placement for your use case instead of trying to perfectly model a realistic hand with your colliders.

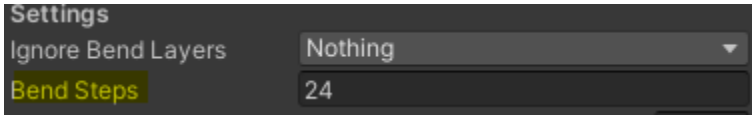
***Q:* Why do exported humanoid hand animations not match the visual hand when a metacarpal bone has been bent?**
A: In Unity Humanoids only have the following hand bones: distal, intermediate, proximal… they do not have metacarpal bones. If you have positioned your hand by bending the metacarpal bones and then recorded a humanoid animation it is possible for the generated animation to not perfectly match the hand's current state. A generic animation can be generated instead to capture the position exactly in this case.

***Q:* How can I smooth the finger bending?**
A: There are multiple ways to smooth finger bending:

- The best way to smooth finger bending is using the 'Finger Move Rate' and 'Finger Rotate Rate' settings found in the 'Inspector' pane for a KinematicFinger component.



- Another option if you find hand movements to be too 'jumpy' is to increase the number of 'bend steps'. More bend steps requires more performance but leads to smoother finger bending.



Q: I am having trouble selecting capsule and sphere handles while editing my colliders.
A: We use Unity's built-in editor handles to allow you to modify your colliders visually in the scene view. Try these suggestions to fix selection troubles:

1. It is likely your model is blocking the handle. Use the 'Layers' dropdown to temporarily lock the layer your model is in to disable selections on it.

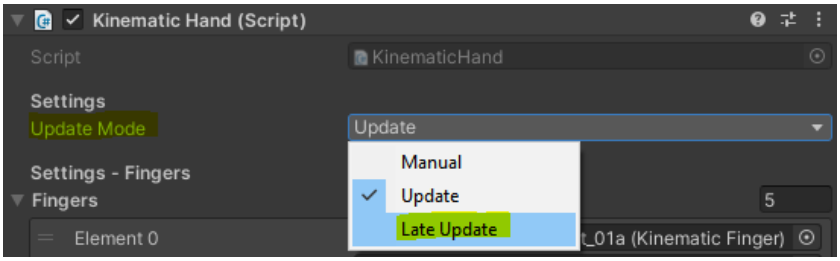
2. Move your camera to a different position/orientation to get a clearer view from the center of your camera to the handle point you are trying to select.
3. Use the 'eye' icon in the 'Hierarchy' pane to temporarily hide the visual part of your mesh.
4. Close the 'hand collider editor' and re-open it by unselecting 'Edit' and reselecting it.
   - If none of these steps work check your Unity editor preferences to confirm you have your 'handle size' and related settings set correctly.

Q: When I increase the bend value my fingers disappear.
A: Solution: save a proper opened and closed position for your hand. *(Youtube Tutorial)*

- This occurs when you have forgotten to 'Save Opened Position' and/or 'Save Closed Position' on your KinematicHand therefore the finger bend is interpolating between one or more invalid states. You can easily fix this by simply saving an opened and closed position for your custom hand.

- Follow this quick youtube walkthrough that takes you through the steps.

Q: My animations are overriding the adaptive hand.
A: This is because the Animator is applying the animation after the 'Update' loop. You can easily solve this by navigating to the 'Inspector' pane for your KinematicHand component and *setting the 'Update Mode' to 'Late Update'*.



A screenshot showing the 'Kinematic Hand' Inspector with 'Late Update' being hovered-over.

Q: After updating the 'Mesh_Hands.fbx" mesh my hands are invisible! How can I fix this?
A: Unfortunately Unity does not handle updating the SkinnedMeshRenderer for mesh instances where you have used the 'Unpack Completely' feature to unpack them. In these cases you must fix your prefab manually by re-adding the new, non-unpacked mesh to your prefab. Here is a youtube tutorial showing the steps to fix it.

Q: I can't bend my hands fingers in 'Live Preview' mode. Even in the prefab editor.
A: Assuming your hand has been correctly configured this happens when some Collider from your scene is physically blocking the fingers from bending. It may be an invisible collider such as a BoxCollider from the ground in your scene. *So why does this still happen in the prefab editor?* When we test for collisions in edit mode, even in the prefab editor, it seems that collisions with scene objects are still detected. Try disabling whatever object the prefab is colliding with in your scene, ***or alternatively,*** you can simply load a blank scene before re-entering the prefab editor.

Q: How can I 'freeze' my hand's finger bend values in place
A: There are a huge variety of ways you can do this using some methods including:

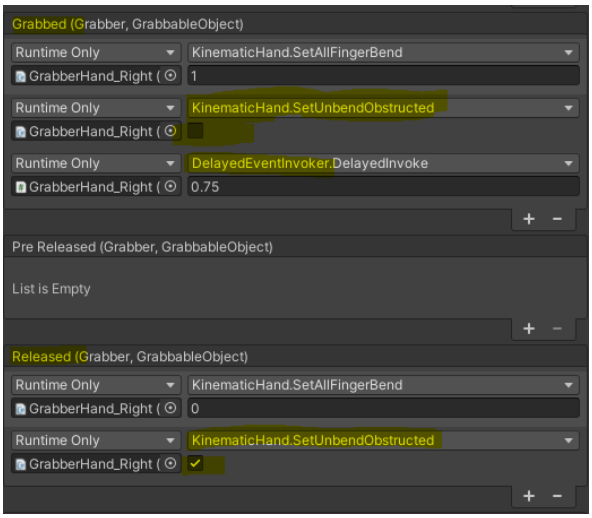
- KinematicHand.SetUnbendObstructed(false)
  - Disables 'unbend obstructed' for all fingers in the kinematic hand.
- KinematicHand.UpdateFullbend()
  - Updates the KinematicHand enough times to simulate a full bend.
- KinematicHand.SetAllFingerBendToCurrent()
  - Sets all finger bone bend targets to their current true values.

Using these three methods in combination will result in your hand no longer unbending fingers that have become obstructed by objects, immediately update enough times to reach its current bend target, and replace bend targets with their current values. This will effectively 'freeze' your hand.

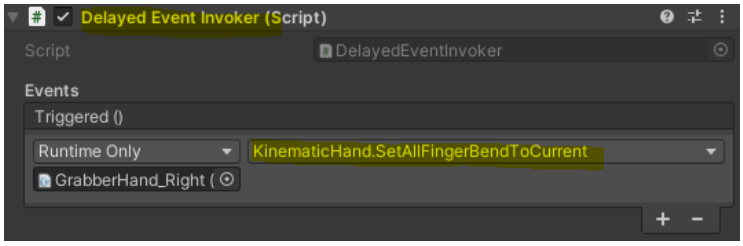
Here is a quick youtube video showing one way to do this.

Q: How can I 'freeze' my hand's finger bend values in place after some time passes?
A: To do this you simply need to:

1. Invoke KinematicHand.SetAllFingerBendToCurrent() this function will replace the target bend values for all fingers in the hand with the current values. This method can be invoked through C# script or through Unity editor events as shown in the 'Grab System' demo in the 'GrabberHand_Right' prefab and the screenshot below.
2. Disable 'Unbend Obstructed' for all KinematicFingers in the hand. This can also be done to all fingers at once using the KinematicHand.SetUnbendObstructed(bool) method.



A screenshot showing the Grabber component in the grab system demo configured to both disable 'Unbend Obstructed' in the fingers hands and invoke the 'delayed invoker' from the screenshot below when the grabber grabs something. It undoes any relevant settings on release..



A screenshot showing a 'DelayedEventInvoker' component configured to set all finger bend targets to their current values when invoked..

Q: How can I prevent desyncing between my 'Adaptive Hands' and physics world objects like Rigidbodys and Colliders?
A: For performance reasons Unity made it so Transform changes are only synced to the physics world when necessary. Because of this behavior you may notice objects that are children of the hand can lead to 'jittery' solving.
The solutions are either:

a) Solve the bending and then 'freeze' the bend states in place as described in various other parts of this documentation.

b) Use some custom script and the following steps to update the adaptive hands after syncing the physics world with Transforms:
   i) Set all of your KinematicHand components to 'Manual' update mode.
   ii) Use a script like the one provided below to update your hands. (This comes at a performance cost which is why Unity introduced this behavior in the first place.)

```
using AdaptiveHands;
using UnityEngine;

/// <summary>Manually manages the updating of KinematicHand components after syncing the physics world with current Transforms.</summary>
public class PhysSyncedHandUpdater : MonoBehaviour
{
    [Header("Settings")]
    [Tooltip("An array of KinematicHands that are updated by this component.")]
    public KinematicHand[] hands;

    // Unity callback(s).
    void LateUpdate()
    {
        Physics.SyncTransforms();
        foreach (KinematicHand hand in hands)
        {
            if (hand != null)
                hand.UpdateHand();
        }
    }
}
```