

[Table of Contents]

[FAQ]

[DISCORD]

[HANDS]

[SETUP NEW HAND]

[FINGERS]

[POSE AREAS]

[INTERACTION COMPONENTS]

[ADVANCED INFO]

[GRABBABLES]

[PLACE POINTS]

[DISTANCE GRABBING]

[CUSTOM GRABBABLE POSES]

[MOVEMENT SYSTEM]

[TELEPORTER]

[CLIMBING]

[PUSHING]

[CONTROLLER INPUT]

[HAND TRACKING CONTROLLERS]

Auto Hand 2.0 Documentation

Thank you for your purchase! Auto hand is a physics-based hand controller that calculates hand pose on grab. The Hand Component will take a Transform to follow and will use unity physics to match the hand to its position and rotation using rigidbody movements. It can be easily connected to any controller. All you need to do to make an object interactable with the hand is apply the Grabbable component. For any questions or issues, you can contact me at EarnestRobot@gmail.com

[JOIN THE DISCORD](#)

[CHECK OUT THE TRELLO FOR PLANNED FEATURES](#)

FAQ

Where are the Scenes?

In order to download the scene you need to pick a subpackage to download in **Autohand/Packages** the package will download the selected scene into **Autohand/Examples/Scenes** see the **SETUP.pdf** in **Autohand/Documentation**

Will this connect to my VR controller?

Yes! You can connect the hands to a mouse and keyboard if you feel like it. The only exception is that *this asset is designed to be used with a single trigger for grabbing*, as opposed to a five-finger controller like the index (however this will still work with the index trigger/button input).

I prefer a different button, how do I change this?

- Each **Hand** from the included packages has an [Input] **Hand Controller Link Component** attached to it that has public button values for the finger bending axis, the grab/release button, and the squeeze/unsqueeze button.
- The subcomponents under the hand (Finger Bender, Grab Pointer, Teleport Pointer and UI Pointer) all have **Controller Link** components on them

Why does the Hand / Held Grabbable jitter?

- The Majority of jitters are caused by Unity Physics not being precise enough. To correct this, you can increase the quality of the physics in the **Physics Settings**.
 - **Default Solver Iterations** and **Default Solver Velocity Iterations** should be increased, values above 100 yield very good results.
 - If Grabbables are jittering when being pulled apart by both hands reducing the break force so it breaks before jittering starts
 - Grabbables Rigidbody mass and drag will also affect jitter, objects of small mass and low drag are likely to wiggle more than objects of near-equal mass to the hand. Strongly recommend making small size/mass objects one-handed grab only.

Why isn't my hand position/rotation not relative to my real hand?

To adjust the hands pivot/position I recommend creating an empty object under the controller, assigning it the hand's follow, and offsetting its transform to get the hand to the ideal position

Layer Missing Error?

Make sure that you have either imported one of the sub packages and enabled the required settings from the setup wizard (**Window/AutoHand/Setup Window**), or manually import the required layers.

How can I use the Auto Pose system without physics/interaction?

The hand auto poses are generated in three steps:

1. The hand's body is aligned to the ideal grabbing pos/rot using physics and raycasts
2. The modular finger components bend until they hit the grabbing object
3. A fixed joint is created between the physics controlled hand and the grabbable

You can cut out steps one and three and just use the finger component to generate poses for whatever interaction system you're using. The finger components' open/closed poses can be individually saved by right-clicking the component.

See [FINGER SCRIPT]

Difference between the Right/Left hand?

The left hand can be an inverted right hand (or visa versa) by scaling the transform.x to -1. Make sure to set the **Left** toggle value on the hand true or false accordingly

Why is there a follow delay?

Because the hand moves using physics updates, it can sometimes lag depending on hand and physics settings. Increase the **Follow Position Speed**, but not too much without increasing the **Rigidbody Drag** can cause jitters. I also strongly recommend decreasing the physics timestep to at least 1/60, 1/90 for very smooth results and less jitters at higher follow values.

Why is it rotating so slowly?

The default **max angular velocity** in the **unity physics settings** is far too slow. I recommend increasing it to at least 40. The **hands follow rotation speed** can also be increased along with the **rigidbody angular drag** to find the right speed

Why is this object rotated and scaled weird when I grab it?

Unity sometimes has problems when rotating physics objects that are children of objects not scaled at (1, 1, 1). Make sure to unparent these objects and use [Fixed Joints](#) if you have to.

Sometimes when I grab an object it instantly releases or I keep losing grip.

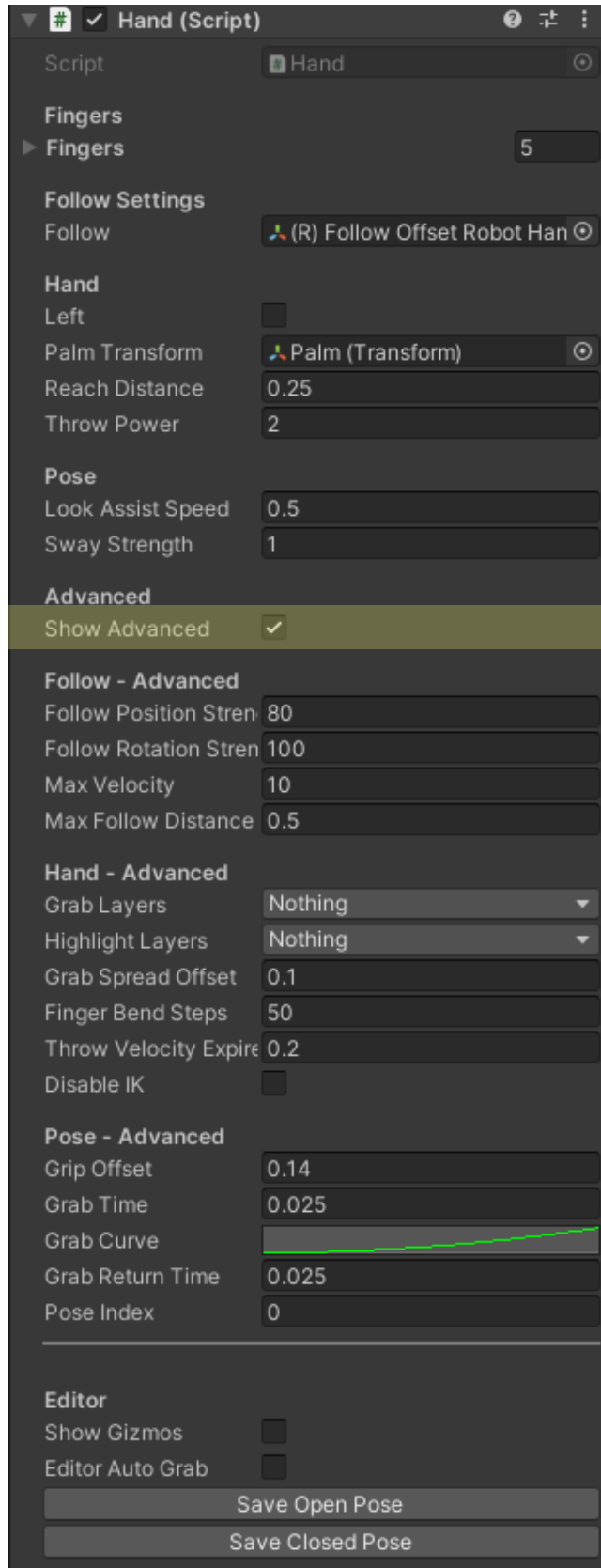
This is likely because the Break Force on the Grabbable is too low for the mass/speed of the hand and the object when grabbing. Turn this value up or adjust the hands/objects mass.

Why isn't my hand smooth when I move it across surfaces?

Everything is based on Unity physics, so the **physics material** will affect how things interact. I recommend applying the Hand Physics Material to each collider on the hand.

[HAND COMPONENT]

The hand component is the core script that runs the hand. It has 5 essential functions to understand. The hand follows using physics settings



[FINGERS]

Fingers [Finger]: The modular finger scripts attached to each fingers root under the hand

[FOLLOW SETTINGS]

Follow [Transform]: The Hand's follow position/rotation target, usually the VR controller

[HAND SETTINGS]

Left [Bool]: Whether or not it's the left or right hand

Palm Transform [Transform]: The transform that represents the forward and upward direction of the palm

Reach Distance [Float]: Maximum distance for hand to grab

Throw Power [Float]: Multiplies throw velocity by this on release

[POSE SETTINGS]

SwayStrength[Float]: This will affect how much the fingers move with velocity

[ADVANCED SETTINGS]

Follow Position Speed [Float]: When turned too high can jitter (increase rigidbody drag if jittering)

Follow Rotation Speed [Float]: When turned too high can jitter(increase rigidbody angular drag if jittering)

Max Velocity [Float]: The maximum allowed velocity of the hand in any direction

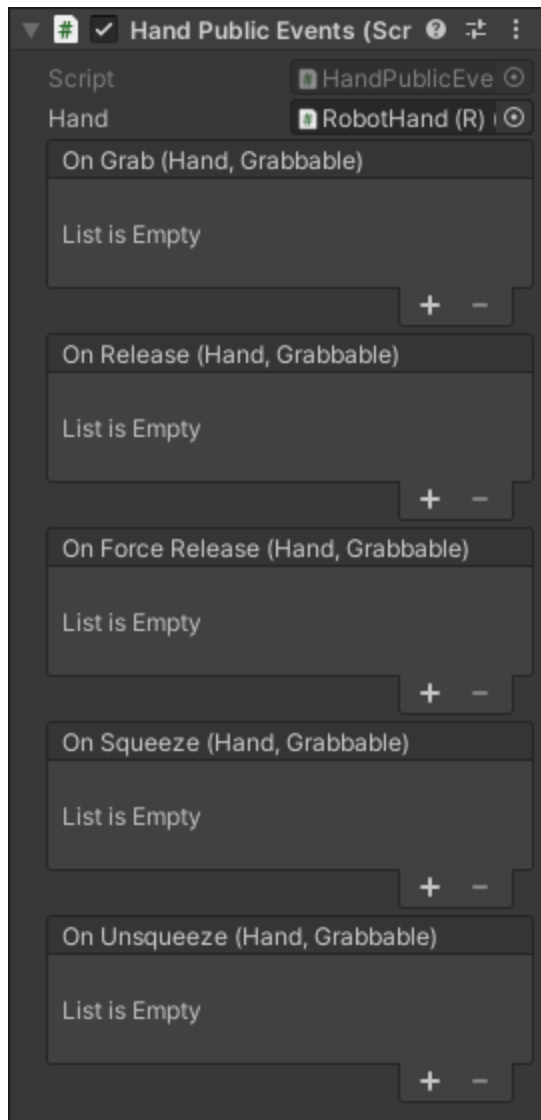
Max Follow Distance [Float]: Will return the hand to the follow target at this distance

[HAND - ADVANCED]

Grab Layers [LayerMask]: These are the layers the hand will check for grabbables

Highlight Layers [LayerMask]: These are the layers the hand will check for highlighting

Grab Spread Offset [Float]: This will affect the width of the grab area, increase for more range but potentially worse automatic grabbing results



Finger Bend Steps [Int]: How many physics checks are made when bending the fingers, decrease for slight performance increase but decreased pose quality

Throw Velocity Expires [Float]: Calculates release throw velocity based on this many seconds of time

DisableIK [Bool]: This will turn off all the automatic updates on the finger transform, like sway and bending, to allow things like animations

[POSE - ADVANCED]

GripOffset [Float]: This is how bent the fingers are (0-1) recommend a slight bend of 0.1-0.2 to allow for sway room

Grab Time [Float]: How many seconds it takes for the hand to grab something at max reach distance (Rec. < 0.1)

Grab Curve [AnimationCurve]: The animation curve for grabbing based on time ($x \cdot \text{grabTime}$) and pose animation point (y)

Grab Return time [float]: How many seconds the hand should take return to the follow point

Pose Index [int]: This value is used in conjunction with pose components to determine which poses to use when using multiple hand models in the same scenes. For example: Robot hand is pose Index 0, Human hand is 1.

[EDITOR SETTINGS]

Show Gizmos [Bool]: Whether or not to show all the hand grabbing rays

Editor Auto Grab [Bool]: When true the hand will try to check and grab what's in front of it in scene, in editor mode. Great for setting up static poses.

[EVENTS]

OnGrab [EVENT]: Called when the hand grabs this object

OnRelease [EVENT]: Called when the hand releases this object

OnSqueeze [EVENT]: Called when the hand squeeze function is called

OnUnsqueeze [EVENT]: Called when the hand unsqueeze function is called

Important Hand Functions:

Grab() The hand will grab the closest Grabbable object, if any, in front of the hand within reach distance. It will then call the OnGrab event on the Grabbable object

Release() The hand will the held object, if any, apply throw power. It will then call the OnRelease event on the Grabbable object

ForceReleaseGrab() This will release the hand without calling OnRelease event or applying throw strength

Squeeze() The hand will call the OnSqueeze event on the held object.

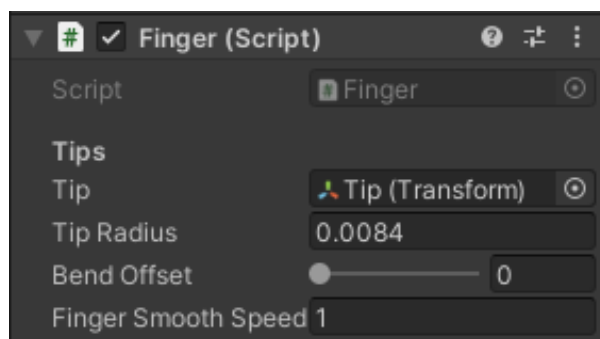
Unsqueeze() The hand will call the OnUnsqueeze event on the held object.

[FINGER SCRIPT]

The Finger component is a modular script used as a core piece of the pose generation system. This component is responsible for containing the open/closed poses for the hand and it manages the finger bending/stopping algorithm for generating automatic poses.

Individual finger bending input is managed through the various **FingerBenderLink** scripts, examples of this can be found under the root of the examples hands, scripts attached to the **Finger Bender Object**

You can use this component without the **Hand component** by saving the Open/Closed poses through right clicking the **Finger component** in the inspector.



Tip [Transform]: This is the center of the tip radius and grab bumper

Bend Offset [Float]: Adjust the bend offset of the individual finger. 0 is no bend, 0.5 is half bend

Tip Radius [Float]: The radius, used for finger collision, drawn in gizmos as a blue sphere

Smooth Speed [Float]: Decrease for slower finger movement, Increase for snappier finger movement.

Important Functions:

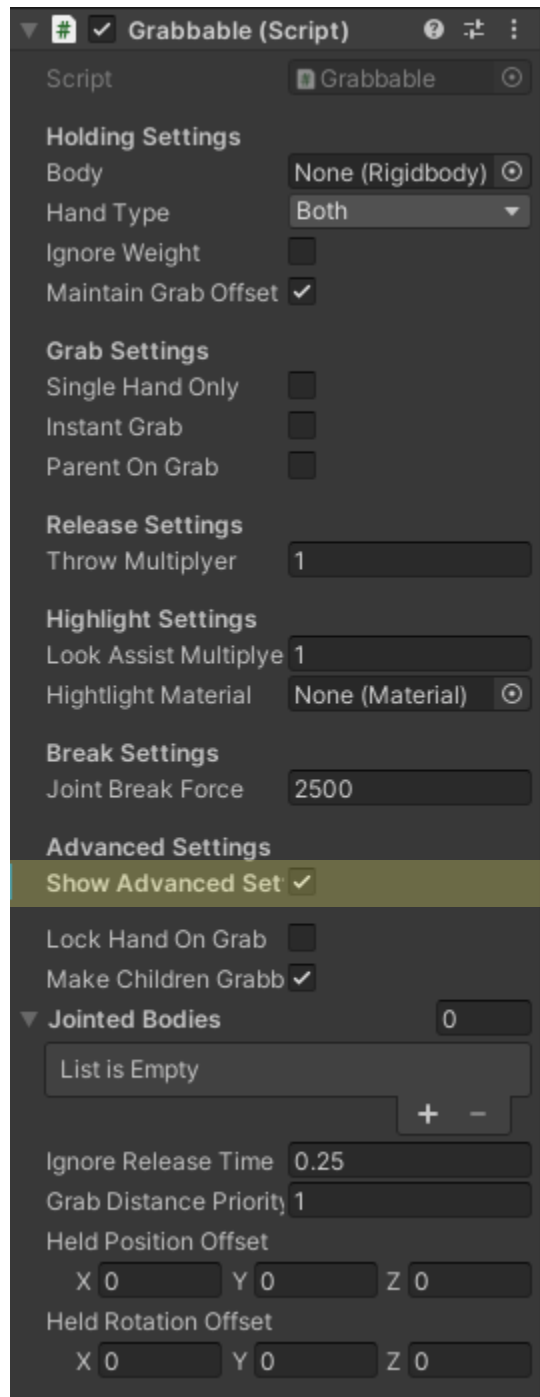
BendFingerUntilHit() This will bend the finger until it hits something, then it will stop

UpdateFingerBend() Takes input 0-1f, will bend the finger, if hits something it will stop

SetFingerBend() Takes input 0-1f and will force bend the finger without doing physics checks (0 is open finger, 1 is fully closed finger)

[GRABBABLE SCRIPT]

The grabbable script should be attached to any object that the hand can pick up. It has public Unity Events that will be called by the hand when interacted with.



[HOLDING SETTINGS]

Body [Rigidbody]: The rigidbody references, if left blank will default to local GetComponent.

Allows for custom grab settings per collider.

Hand Type [HandType]: Whether this can be held by right/left/both/no hand

Ignore Weight [Bool]: Whether is object is practically weightless while being held

Maintain Grab Offset [Bool]: When true, the hand will create an offset instead of returning back to the follow point, good for grabbable gadgets and grabbable joints.

[GRAB SETTINGS]

Single Hand Only [Bool]: Whether or not this object can be held with only one hand at a time

Instant Grab [Bool]: When true, the hand will instantly return to its follow position on grab

Parent On Grab [Bool]: When true, the object will be parented to the hands parent, allow you to move freely and teleport with the object

[RELEASE SETTINGS]

Throw Multiplier [Float]: multiplies throw velocity by this on release

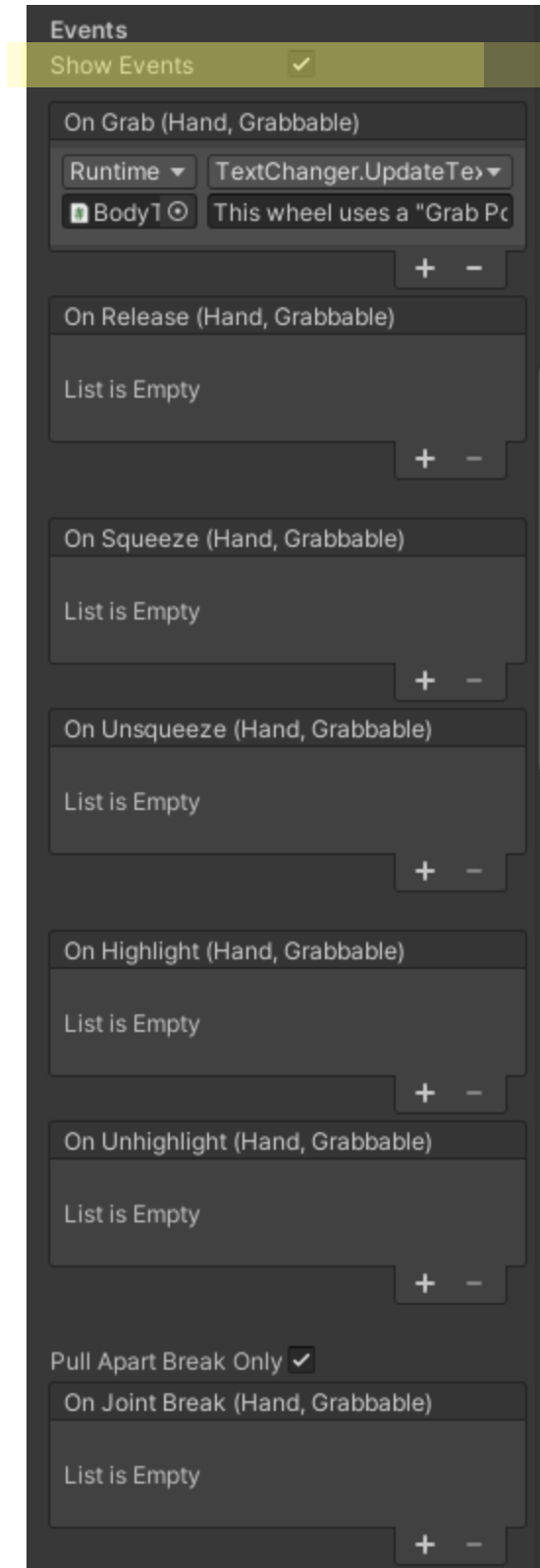
Joint Break Force [Float]: The force required to break the connection between the hand and grabbable

[ADVANCED SETTINGS]

Lock Hand On Grab [Bool]: When true, the hand will become kinematic while holding

Make Children Grabbable [Bool]: Each collider needs a Grabbable or a GrabbableChild with reference. When true, all the children colliders will be referenced to this grabbable component

Jointed Bodies [List]: Connect jointed rigidbodies attached to this grabbable here, so when you move/teleport they are stable



EVENTS

OnGrab [EVENT]: Called when the hand grabs this object

OnRelease [EVENT]: Called when the hand releases this object

OnSqueeze [EVENT]: Called when the hand squeeze function is called

OnUnsqueeze [EVENT]: Called when the hand unsqueeze function is called

OnHighlight [EVENT]: Called when the hand highlight function is called

OnUnhighlight [EVENT]: Called when the hand unhighlight function is called

Pull Apart Break Only [Bool]: When true, the hand will never call the joint break event unless being held by more than one hand

OnJointBreak [EVENT]: Called when the joint breaks, used to simulate pull-apart

Important Functions:

HandRelease() This will force the hand to release this object and apply velocity

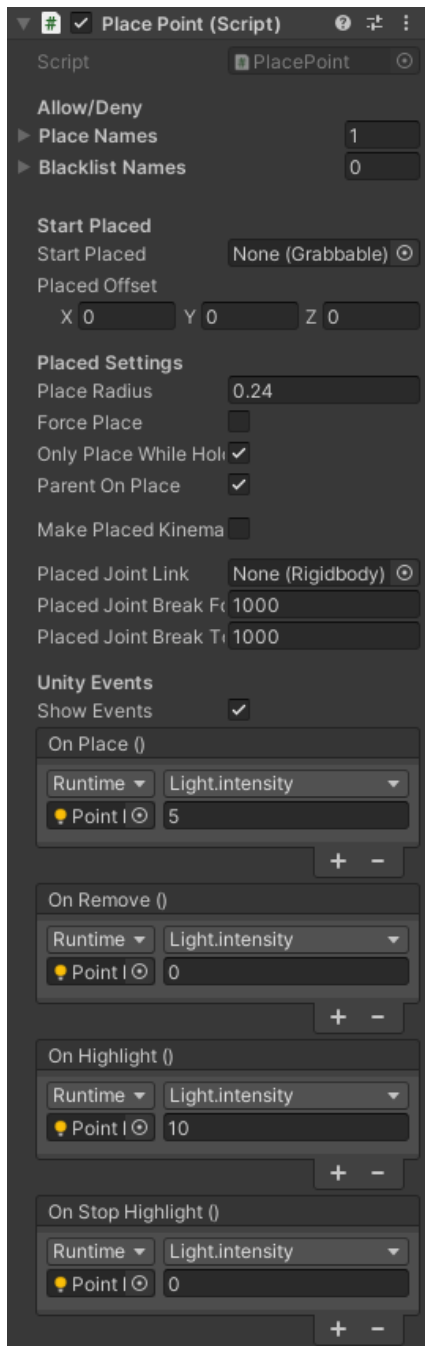
ForceHandsRelease() This will force the hand to release this object and not apply velocity, like it was dropped not thrown.

GetHeldCount() Returns how many hands are holding this grabbables.

GetHeldBy() Returns which hands are holding this grabbable

[PLACE POINTS]

The Place Point Component is a tool that allows for custom grabbable placement settings through a trigger collider. For completely stable connection use isKinematic, for a physics breakable connection, use a rigidbody, for no connection just use neither.



[ALLOW/DENY SETTINGS]

Place Names [String[]]: Only objects that have **names** that *CONTAIN* any of the given strings(Case sensitive) will be allowed to place

Blacklist Names [String[]]: Objects that have **names** that *CONTAIN* any of the given strings(Case sensitive) will NOT be allowed to place

[PLACEMENT SETTINGS]

Place Radius [Float]: The radius in which an object can be placed

Force Place [Bool]: Whether or not the object will be forced released and put into the place point as soon as it enters the trigger

Parent On Place [Bool]: Whether or not to parent the grabbable to the point when placed

[PLACED SETTINGS]

Start Placed [Grabbable]: Leave blank for nothing, connects this grabbable on start, can be used to test offset

Placed Offset [Vector3]: Offsets the position of the placed object allowed in this place-point

Make Placed Kinematic [Bool]: Whether or not to make the grabbable kinematic in the place point until grabbed again

Placed Joint Link [Rigidbody]: allows for the option of jointing the given rigidbody with the object in place, this will have the effect of a weighted connected when break force is adjusted.

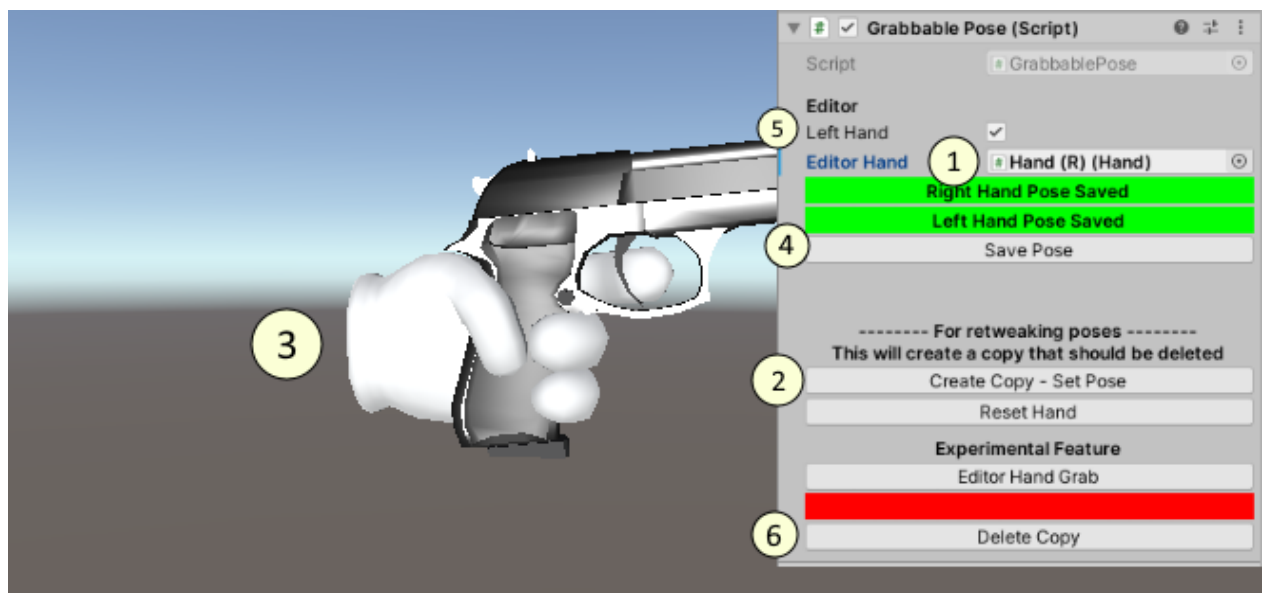
*By default Place Points require a sphere collider, for more advanced place bounds, reduce the place radius to 0 and add custom trigger colliders to the object

[GRABBABLE POSES]

These components can all be used to create custom poses with the hand.

- **Grabbable Poses** are used to set a custom saved grab pose.
- **Grabbable Pose Advanced** are used to set custom poses that work around a radius and length
- **Pose Area** is a trigger collider that can be used to set the hands pose, while not holding, when in this collider. **STRONGLY RECOMMEND PUTTING A TRIGGER SPHERE AROUND HAND TO PREVENT POSE FLICKERING (SEE EXAMPLE HANDS)**
- **Pose Driver** is used to save and call a local pose through code

[\(NEW\) - Advanced Grabbable Pose Instructional Video](#)

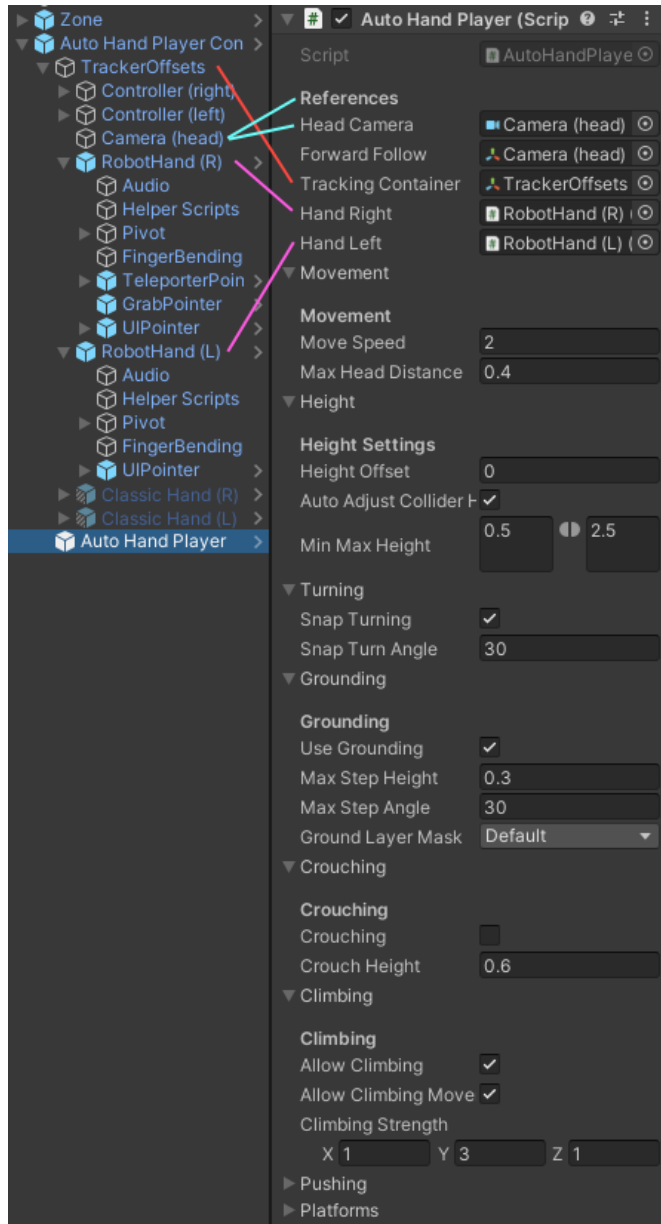


***THIS APPLIES TO THE GRABBABLE POSE AND POSE AREA COMPONENTS**

1. Make sure you're in a scene view, attach a hand in the scene for editor copy
2. Create a copy for editing purposes
(By default Editor Auto Grab is enabled on the hand copy)
3. Shape the hand, positioning the Hand Object, then adjust the individual finger rotations/positions, anything with or under a Finger Component Object to your liking
4. Save Pose
5. Do the same for the other hand (you can invert hand x Scale to -1 to do this)
6. Delete Editor Copy
7. (Optional) Apply overrides to prefab to save for all instances

[AUTO HAND PLAYER]

When setting up the **Hand Player**, all tracking objects and the Auto Hands should be placed under a “Tracking Container”, in this case named *Tracked Offsets*. Make all the appropriate connections under the body section.



[Climbing]

Applying the **Climbable** Component to a Grabbable object will allow you to climb if the player controller has climbing enabled

You can increase climbing speed per axis through the climb speed values on the **HandPlayer** and or per **Climbable**

Climbable Grabbables should be set as Kinematic through the **Rigidbody**

You can increase breaking force to infinity if hand keep breaking

Making the Hands Max Distance too low can affect climbing

[Pushable]

Objects with the **Pushable** Component will allow you to push against them to move the body with your hands

Can be applied to static or heavy **Grabbables** to anchor the Player Body

[Player Platform]

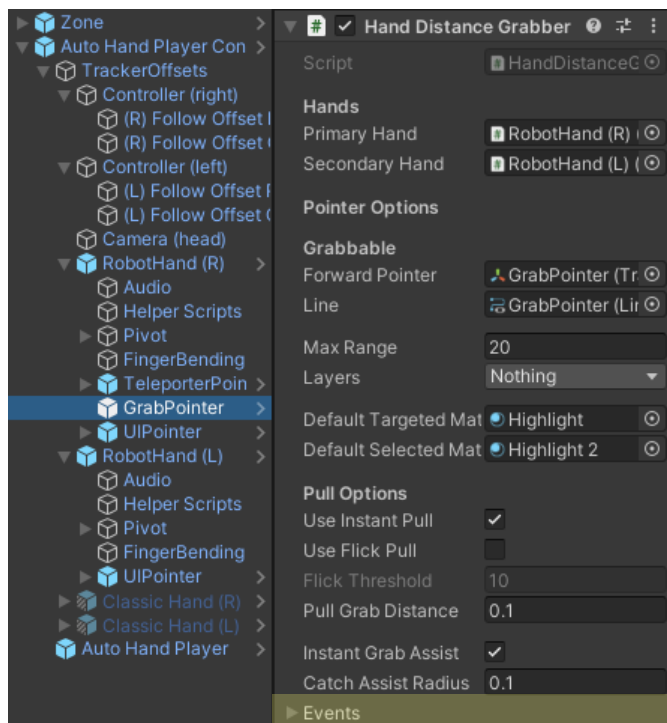
Objects with the **Player Platform** Component will move the player with is when it moves

Documentation for each value on this component can be found in the tooltips
(by hovering over the value in the inspector)

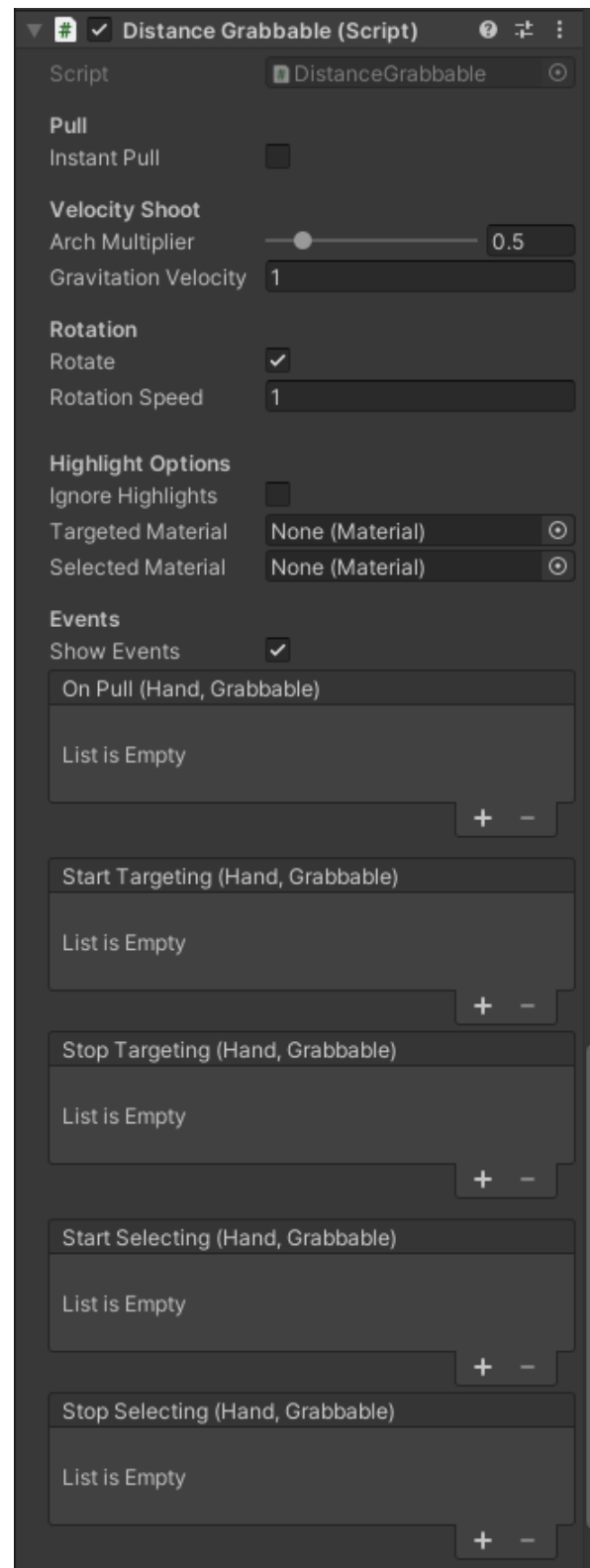
[DISTANCE GRABBING]

Distance Grabbing is achieved through the **Hand Distance Grabber** Component used by the hand or tracked controller, in conjunction with the **Distance Grabbable** Component applied to each grabbable to allow for distance grabbing

***Hand Distance Grabber** goes under the hand or tracked controller*



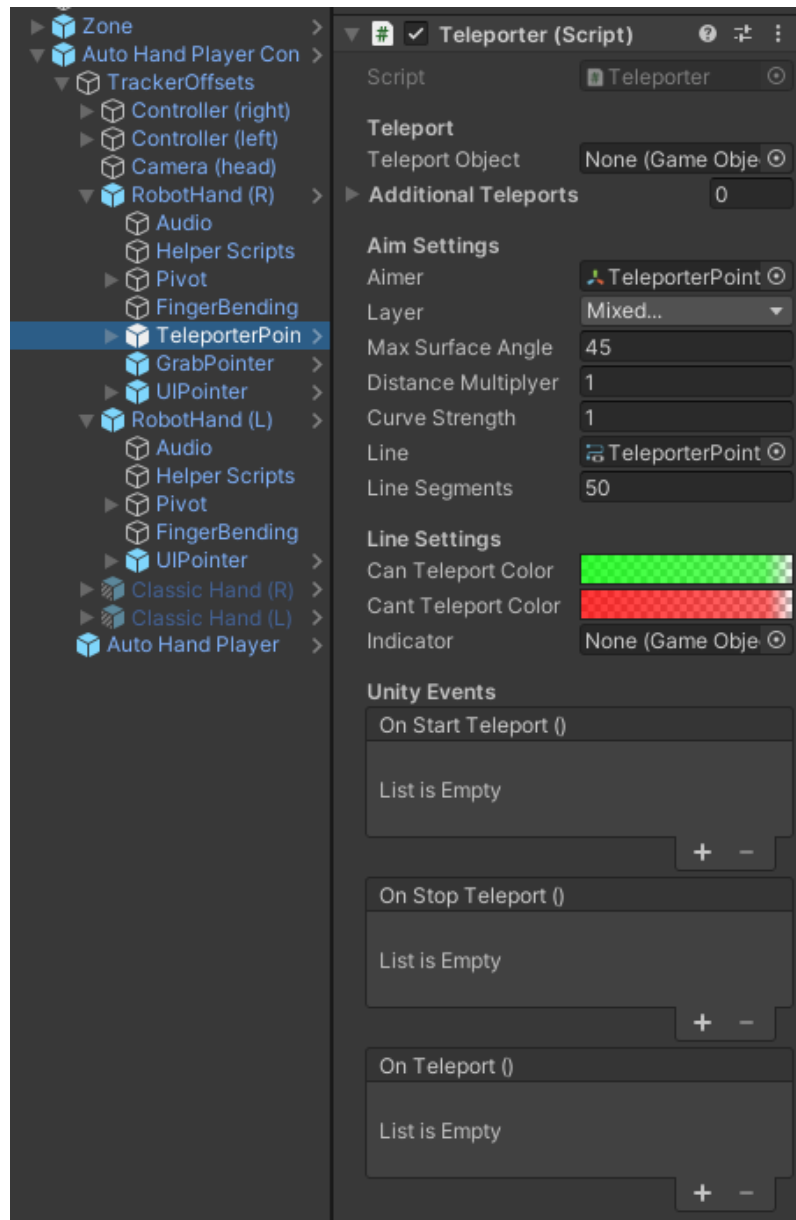
Documentation for each value on this component can be found in the tooltips (by hovering over each value in the inspector)



[TELEPORTER]

The Component in the demo can be found under the Right Hand
(Player/TrackerOffsets/Hand (R)) in the Demo scene

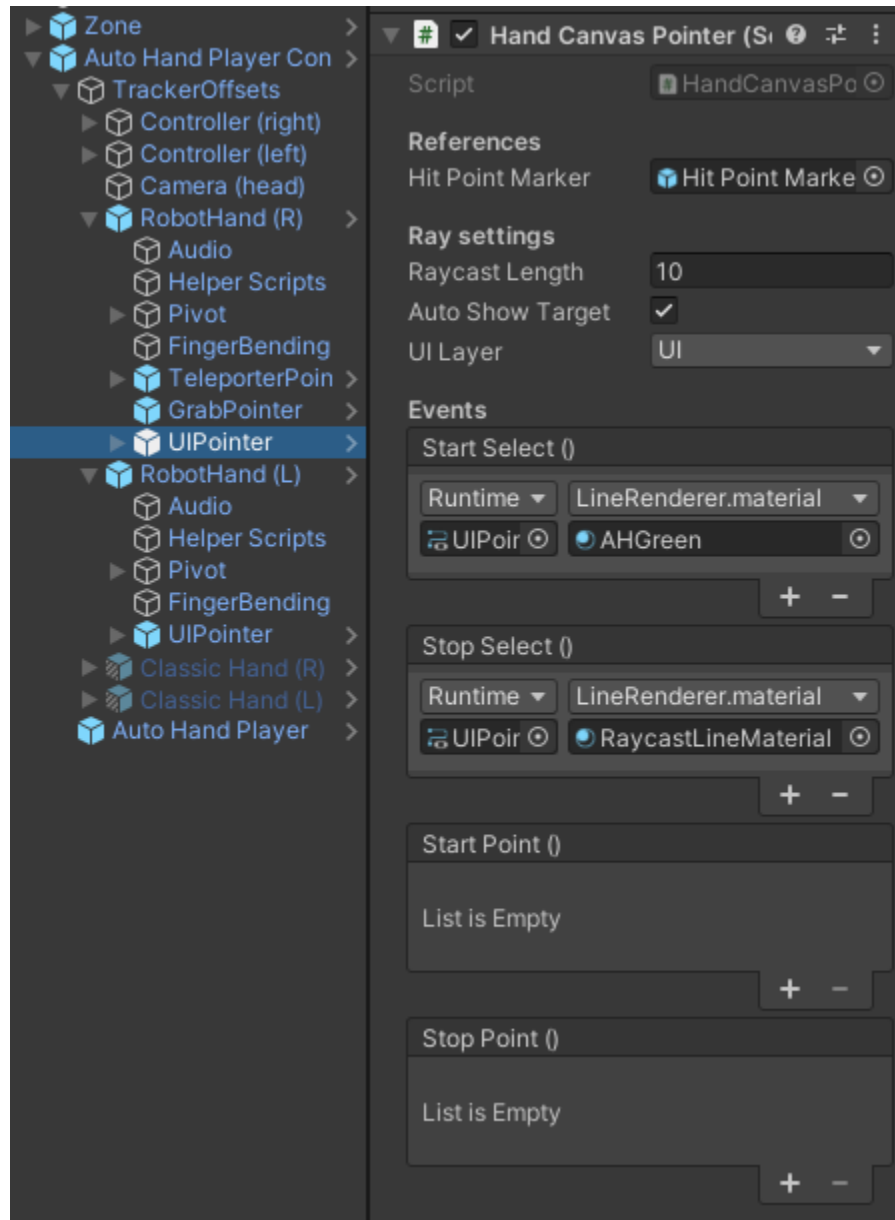
Documentation for each value on this component can be found in the tooltips (by hovering over each value in the inspector)



Input is achieved through the TeleporterControllerLink Component under this Teleporter Component

[UI INTERACTION]

This Component can be added under multiple hands



Documentation for each value on this component can be found in the tooltips (by hovering over each value in the inspector)

[CONNECT TO ANY CONTROLLER]

This is for custom input setup outside of the included demo packages, if you are looking to use XR, SteamVR, or OVR follow the **SETUP** documentation instead. All these packages come with a **HandControllerLink** script attached to each hand, that allows for custom input setting adjustments. Auto Hand will connect to any controller with ease. There are only up to 5 essential functions that should be linked to their respective controller events.

(This is a pseudo code examples)

```
public class VRHandController : MonoBehaviour{
    public Hand hand;
    public VR_Controller controller;

    void Awake(){
        controller.TriggerClicked += hand.Grab;
        controller.TriggerUnclicked += hand.Release;
        controller.Gripped += hand.Squeeze;
        controller.Ungripped += hand.Unsqueeze;
    }

    private void Update(){
        hand.SetGrip(controller.GetTriggerPressValue());
    }
}
```

The controller link system demonstrates that in order to connect a controller to the hand all you would need to do is call the

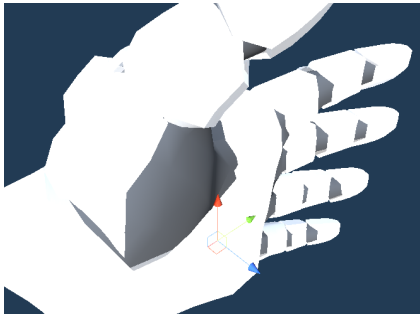
- hand.Grab() function when the trigger is pressed.
- hand.Release() function when the trigger is released.
- hand.SetGrip(0-1) takes the controller trigger current press state and bends the fingers along with how much the trigger is pressed

[CONNECT TO NEW HAND]

→ → → [WATCH THE SETUP VIDEO](#) ← ← ←

1. Create new prefab with a rigged hand model
2. Apply Hand component to root of hand prefab
3. Apply Finger component to the root of each finger transform
4. Add each finger to the fingers array on the Hand component
5. Add an empty gameobject to the last child of each finger and connect to the "tip" value on finger
6. Use on gizmos to see finger-tip radius
7. Adjust its position and the tip radius (value on Finger component) until the wire sphere gizmo covers the fingerprint of the finger and do this for each finger
8. Shape hand rig into its completely open position (flat open palm)
9. Click the *Save Opened Hand* button on the bottom of the Hand component
10. Shape hand rig into its completely closed position (Fist)
11. Click the *Save Closed Hand* button on the bottom of the Hand component
12. Create an empty gameobject and center it just above the palm of the hand with blue forward arrow pointed away from the palm of the hand and the yellow arrow pointed towards open finger

IMPORTANT CHANGE TO PALM ROTATION NOT INCLUDED IN VIDEO



palm transforms local rotation should be like this now including y axis pointed towards open fingers

13. Drag it into the "Palm transform" slot on Hand component
14. (Recommended) Add colliders to the skeleton (refer to hand prefab for example)
15. Turn off gravity on the hands rigidbody, increase the mass (recommended ≥ 10) and turn on continuous detection

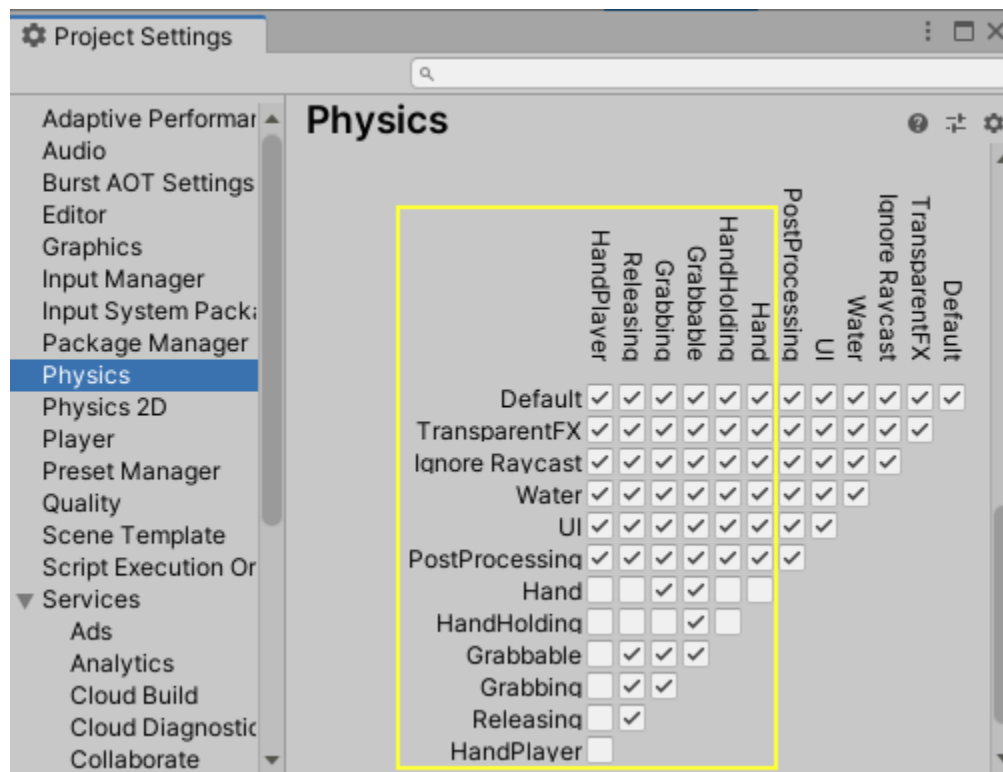
[PHYSICS SETTINGS]

Auto Hand at its core is a physics based system and requires some changes to the default Unity Physics Settings to work as designed. Some things are just general recommendations to improve the overall performance, but the physics layers are a requirement and help to prevent some critical bugs. You can change the name of the grabbable layers and more information on each one's significance near the top of the Hand.cs component.

*** ALL SETTINGS SHOULD BE APPLIED WITHOUT OVERRIDING ANYTHING THROUGH THE SETUP WIZARD** (should automatically popup or find at **Windows/Autohand/**)

Required layer settings

In Edit/Project Settings/Physics



These are the required physics layers. Don't worry about applying these layers to the objects in the editor, all layers will be applied through the setupwizard at runtime.

The strongly suggested physics settings

In Edit/Project Settings/Physics

- Enable adaptive force to allow for better rigidbody stacking
- Increase Default Solver Iterations for more precise stacking / reduced jittering
- Increase Default Solver Velocity Iterations for more precise throws and
- Increase Angular Velocity to at least 40 (default is too slow to match controller rotation speed)

*If you want to use the finger pose system without the physics interaction system, refer to **FAQ** or **Finger Component**.

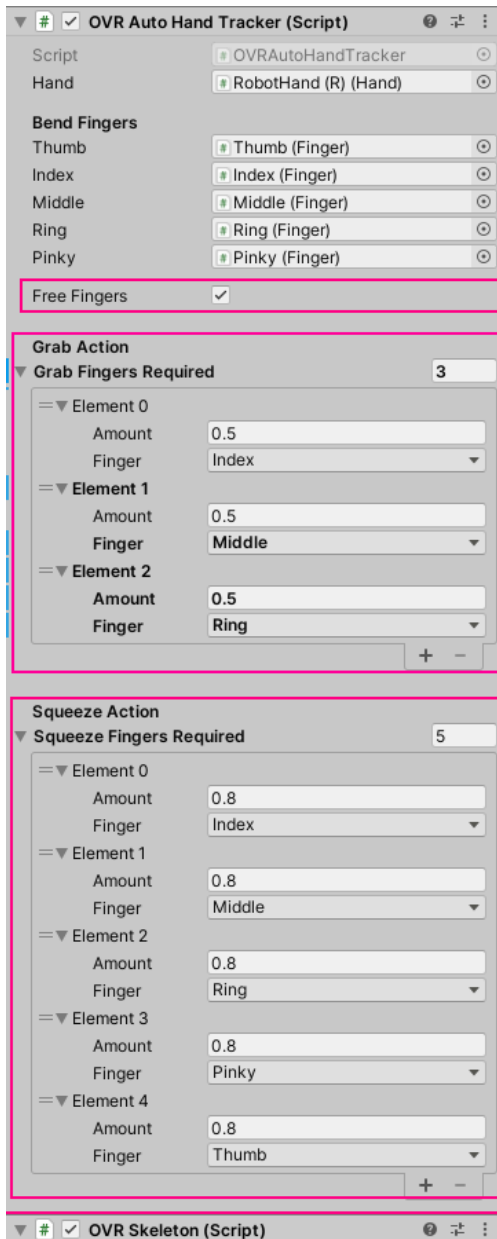
[ADVANCED INFO]

- The Hand works by interpolating each finger between the saved opened and closed position. For the best results an open pose on the hand should be open to the point of being a flat hand. You can set the "Grip Offset" value to adjust the default pose of the hand so you don't have a pancake hand when not grabbing, and some room for movement sway, but still have good grabbing results.
- **Gadgets:** are designed using a base reader class with some value customizability. The **Hinge Angle Reader** base uses a Hinge Joint and reads its angle limits. The **Configurable Limit Reader** reads the Configurable Joint Limit. Usually the reader will default from (1 to -1) representing open/closed, with the center at 0.
There are a handful of user friendly premade gadgets with easy to use Unity Events in the demo scene.
 - You can find all the included gadgets by searching for **PhysicsGadget**.

- Physics will work better with a reduced fixed timestep recommended min. 1/60, 1/90 for very good results
- You can bend individual fingers by changing their offset value on the Finger component
- The closer to the surface of the finger the fingertip is, the closer the grabbing results
- You can increase the hands "strength" by increasing the rigidbody mass
- Rigidbody drag can help reduce jitters and smooth movements when increasing follow speed. You can adjust the drag and the hand follow speeds to get different follow results.
- The follow speed, max velocity, and hand model rotation pivot will all affect throw feel

[HAND TRACKING CONTROLLERS]

This applies to Index and Oculus hand tracking



Make sure to enable **Hand Tracking Support** for Oculus Under **Oculus/OculusProjectConfig**

Whether or not fingers not required for grab action should have free movement while holding

Represents the fingers required being bent past a certain range to call grab action

Represents the fingers required being bent past a certain range to call squeeze action

This is required for Oculus Hand Tracking