[Table of Contents]

[ABOUT]

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

[FAQ]

- **[1]Q: Will this connect to my VR controller?**
- A: Yes! You can connect the hands to a mouse and keyboard if you felt 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.
  See [CONNECT AUTOHAND TO ANY CONTROLLER]

- **[2]Q: Why does the held object jitter sometimes?**
- A: **Make sure to add the HandStabilizer component to the camera.** This is a result of Unity's joint system. In order to reduce jitters you can increase the Default Solver Iterations under Unity's physics settings, 100 yields very good results.
  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.**
  See [PHYSICS SETTINGS]

- **[3]Q: Why is it rotating so slowly?**
- A: The default max angular velocity in the physics settings is far too slow. I recommend increasing it to at least 30
  See [PHYSICS SETTINGS]

- **[4]Q: Why is the sway messed up for one of the hands?**
- A: This is likely because your hand is inverted on the wrong axis, make sure the x is set to be the negative scale on the inverted hand.

- **[5]Q: Why isn't my custom hand rotated the right way?**
- A: In place of a public rotation offset, it is now required that you rotate the hand using the a **root** transform, which would be the first child of the hand and contains all the other children objects (under the object with hand.cs), as the pivot. See hand prefabs for example

- **[6]Q: Why is this object rotated and scaled weird when I grab it?**
- A: 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.

- **[7]Q: Sometimes when I grab an object it instantly releases.**
- 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 reduce the hands/objects mass.

- **[8]Q: 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. Apply the **Hand Physics Material** to each collider on the hand

- **[9]Q: Layer Missing Error?**
- Make sure that you have either imported one of the sub packages and included the physics settings, or manually import the required layers. **See [PHYSICS SETTINGS]**

[HAND SCRIPT]

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

- ### Follow Settings
    - **Follow** is what transform the hand should match using rigidbody movement
    - **Max Velocity** will cap the velocity at the given value
    - **Follow Position Strength** determines how much force to apply to the hand to move the hand to the follow position
    - **Follow Rotation Strength** determines how much torque to apply to the hand to move the hand to the follow rotation
    - **Max Follow Distance** is how far the hand can be from the follow transform before it is teleported directly to the position (this will prevent hand getting stuck and will only teleport if hand is not holding anything)

- ### Hand Settings
    - **Fingers** array should be filled with the **Finger** components attached to the root of each fingers transform
    - **Left** whether this is the left hand or not
    - **Look Assist Speed** this will increase how quickly the hand rotates to point the Palm Transform to look at the nearest **Grabbable** within **Reach Distance**
    - **Throw Power** determines how much to multiply the velocity of object on release
    - **Reach Distance** how far in unity units the hand can interact with a **Grabbable** (enable show gizmos to see hands rays)
    - **Ignore Release Time** how long in seconds the hand will ignore the colliders of an object when releasing it. Turning this value to zero may cause errors. This helps prevent clipping errors and throwing problems.

- ### Pose Settings
    - **Palm Transform** is used to determine where and in what direction to align the hand to the object it's grabbing. This should be an empty GameObject centered just in front of the palm of the hand with the blue forward arrow pointed away from the palm and the yellow up arrow pointed at the open fingers
    - **Grip Offset** is used to offset the default bend of each finger (0-1f). recommended ~0.1f
    - **Sway Strength** is how much the the fingers will wiggle depending on the hands velocity

- **Important Hand Functions**
    - **Grab**() The hand will grab the closest Grabbable object, if any, in front of 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.

* All public variables have tooltips and the script and functions are documented for further clarification.

[FINGER SCRIPT]

The **Finger** component is a required variable for the hand but it can also be individually controlled with some of its convenient public functions and variables

- **Bend Offset** is used to offset the default bend of the finger (0-1f)
- **Tip Radius** is used in conjunction with the **Tip** transform to sphere check if the finger has touched something. In other words it is acting as the fingers stopper. Use gizmos to see fingertip area (in light blue)
- **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.

- **Throw Multiplayer** will determine how much to amplify the throw after the hand throw amplifier is applied. Turn this down to reduce flinging of heavy objects.
- **Joint Break Force** and torque determine how much force is required to break the fixed joint connecting the hand and its held object. The OnJointBreak event will be called whenever this joint breaks, unless the **Pull Apart Break Only** bool is true, in which case it will only be called if the joint is broken when being held by two or more hands. (This is good for simulating pulling something apart)
- **Allow Many Hands** will determine if this object can be held with more than one hand. If this is false and you try to grab it with the other hand it will swap hands instead.

[GRABBABLE POINT]

Grabbable Points allow for forced point grabbing of an object using auto grab feature

- **Right** the point where the right hand will be placed before grabbing
- **Left** the point where the left hand will be placed before grabbing
- **Editor Hand** used to help orient the left and right points (will automatically move the hands back to where they were before)

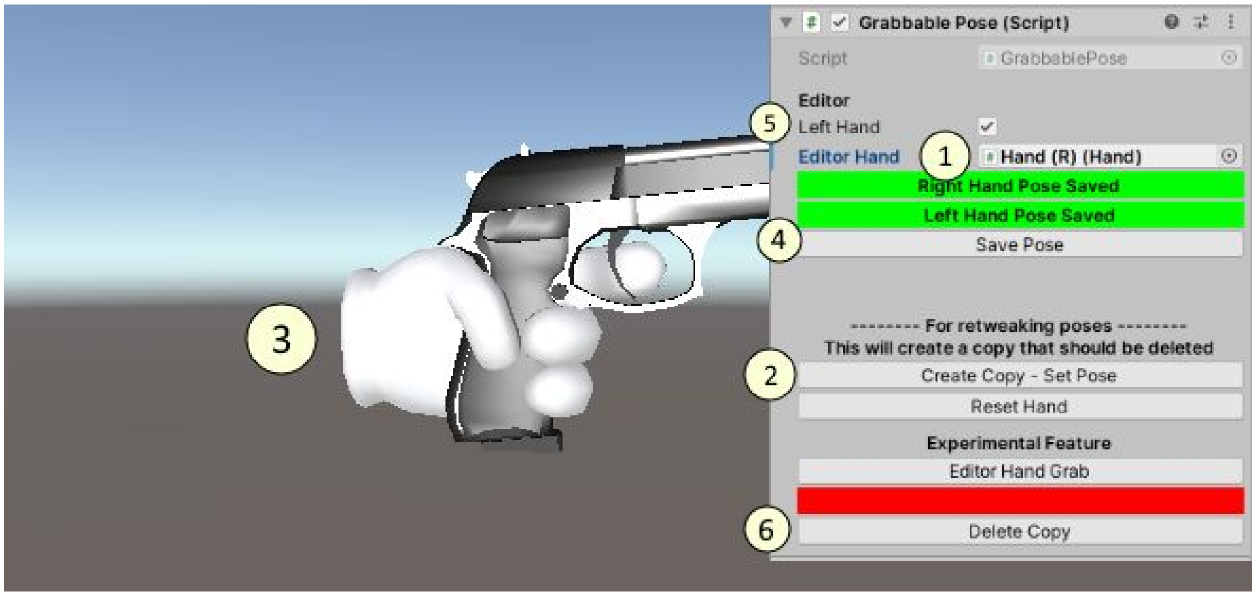
[PLACE POINTS]

- **Place Radius** the radius in which an object can be placed
- **Offset** offsets the position of the placed object
- **Place Names** only objects that contain the given name (Case sensitive) will be allowed in this place-point . For example, “Car” will allow for “RedCar” or “BlueCar” or “Cart”
- **Make Kinematic** whether or not the object should be made kinematic when placed.
- **Start Placed** will automatically place and parent the given object on start.
- **Joint** allows for the option of jointing this given rigidbody with the object in place.
- **On Highlight()** is called whenever a held grabbable enters the place radius
- **On Stop Highlight()** is called whenever it leaves the place radius
- **EditorGrabbable** is used to test where an object will be placed in the editor

[STATIC POSES]

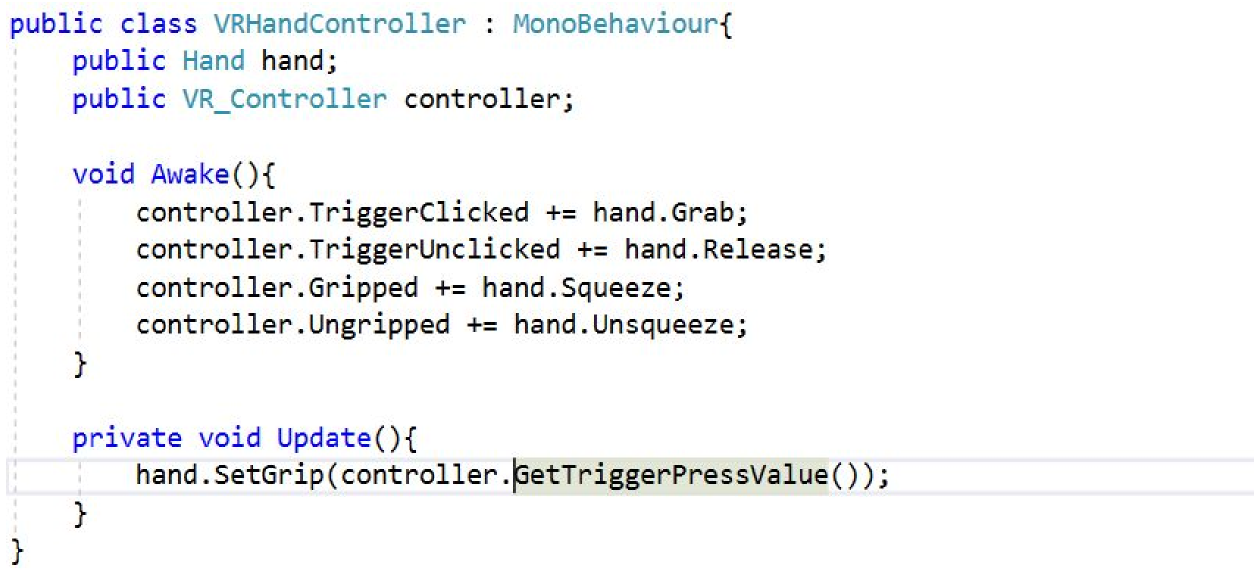
1. Make sure you're in a scene view, attach a hand in the scene for editor copy
2. Create a copy for editing purposes

   By default **Editor Auto Grab** is enabled on the hand copy
3. Shape the hand, positioning the **Hand Component 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



[CONNECT TO ANY CONTROLLER]

Hand can connect to any controller with ease. There are only up to 5 essential functions that should be linked to their respective controller events.



```
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

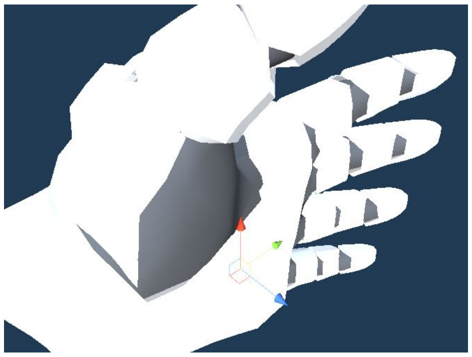
- hand.Grab() function when the trigger is pressed.
- hand.Release() function when the trigger is released.
- hand.SetGrip(0-1) takes the controller trigger current press state and bends the fingers along with how much the trigger is pressed

[CONNECT TO NEW HAND]

https://www.youtube.com/watch?v=6kDEctIKeTM

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

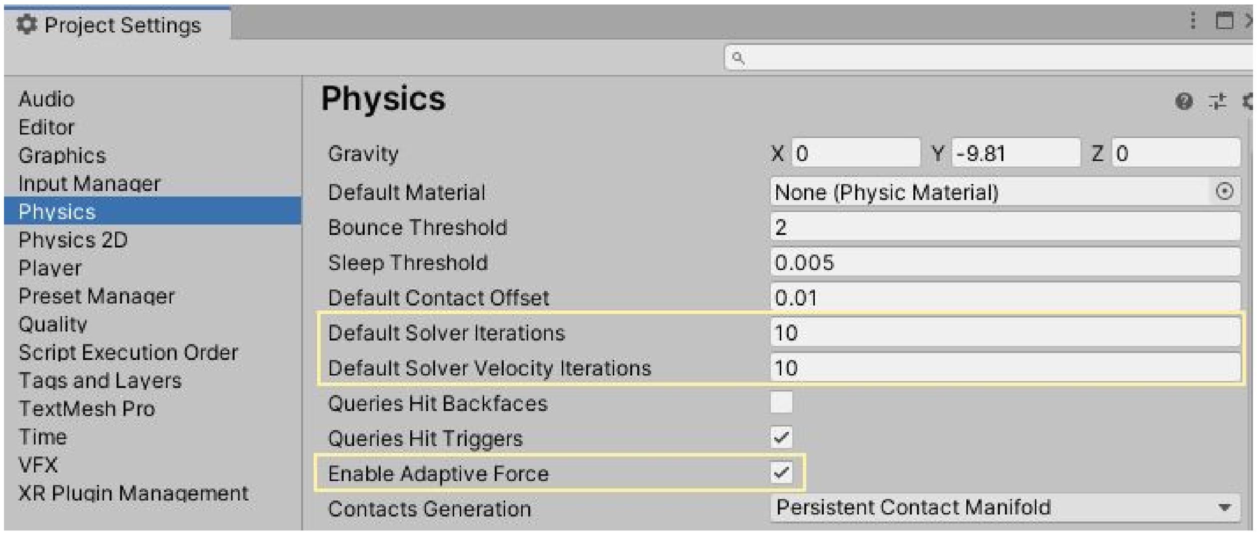
    IMPORTANT CHANGE TO PALM ROTATION NOT INCLUDED IN VIDEO

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

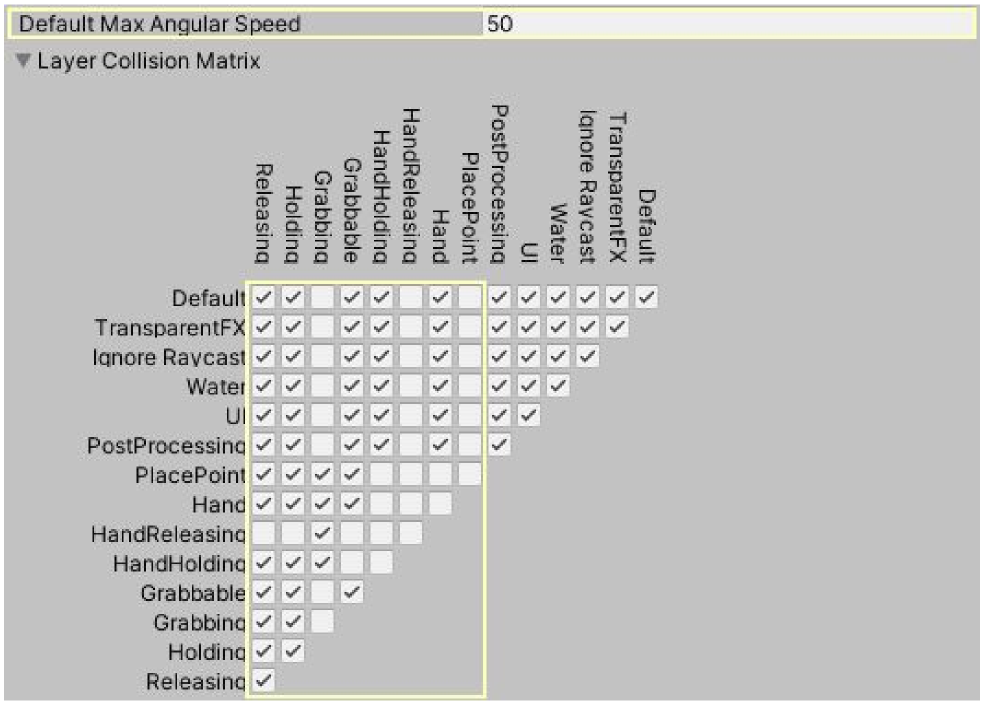
In Edit/Project Settings/Physics *These settings are included with the sub-packages



Physics

Gravity	X 0 Y -9.81 Z 0
Default Material	None (Physic Material)
Bounce Threshold	2
Sleep Threshold	0.005
Default Contact Offset	0.01
Default Solver Iterations	10
Default Solver Velocity Iterations	10
Queries Hit Backfaces	
Queries Hit Triggers	✓
Enable Adaptive Force	✓
Contacts Generation	Persistent Contact Manifold

- 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 30 (default is too slow to match controller rotation speed)



Default Max Angular Speed	50

Layer Collision Matrix

	Releasing	Holding	Grabbing	Grabbable	HandHolding	HandReleasing	Hand	PlacePoint	PostProcessing	UI	Water	Ignore Raycast	TransparentFX	Default
Default	✓	✓		✓	✓		✓		✓	✓	✓	✓	✓	✓
TransparentFX	✓	✓		✓	✓		✓		✓	✓	✓	✓	✓	
Ignore Raycast	✓	✓		✓	✓		✓		✓	✓	✓	✓		
Water	✓	✓		✓	✓		✓		✓	✓	✓			
UI	✓	✓		✓	✓		✓		✓	✓				
PostProcessing	✓	✓		✓	✓		✓		✓					
PlacePoint	✓	✓	✓	✓										
Hand	✓	✓	✓	✓										
HandReleasing			✓											
HandHolding	✓	✓	✓											
Grabbable	✓	✓		✓										
Grabbing	✓	✓												
Holding	✓	✓												
Releasing	✓													

These are the required physics layers and settings for the hand to work. Don't worry about applying these layers to gameobjects, all layers will be applied through the scripts at runtime.

[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.

- Placing the fingertips so that they only cover the outside (fingerprint side) of the hand yields better grabbing result

- You can increase the **hands strength** by **increasing the rigidbody mass**

- Rigidbody **drag** will reduce jitters and movements.
  You can adjust the drag and the hand follow speeds to get different follow results

- The follow speed and hand offset will affect throw timing