VR Hands Responsive

v1.0

Get 4 responsive VR hand objects for your projects. These hands are designed to work seamlessly with mobile devices and can be used with any compatible headset that has controllers. The hands are incredibly easy to set up; you can simply drag and drop them into your project and they will automatically recognize the surfaces and adjust their pose accordingly. With their high responsiveness, you can be sure that your VR experiences will be more immersive than ever before. These hands are perfect for developers who want to create engaging and interactive VR experiences that can be enjoyed by a wide range of users. Whether you're developing games, training simulations, or educational programs, these VR hand objects are an essential tool for any VR developer. So why wait? Get your hands on them today and start building the future of VR!

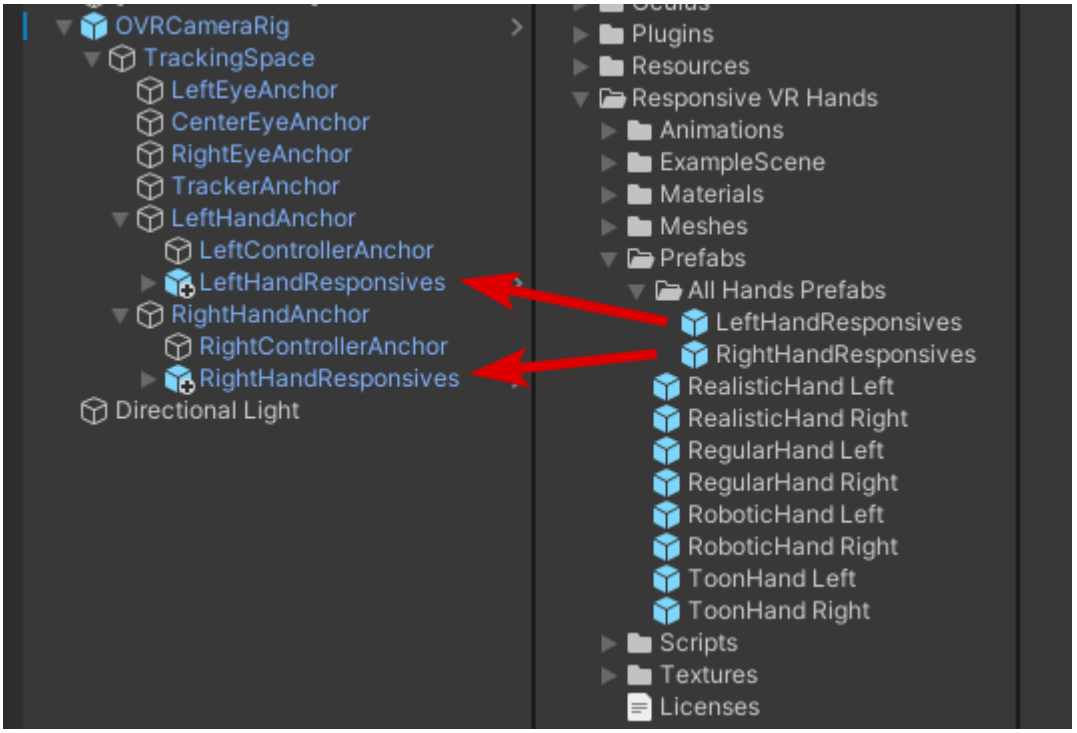
- Great news! This asset is super easy to use!
- You'll have access to 4 different models, each with a finely tuned posing system!
- And if you're looking for something really unique, you can modify it to fit your custom hand models!
- All source code included!
- And because we're using Unity systems in the backend, you can be confident that this asset is highly modular and future-proof!
- It is compatible with all headsets! So you can enjoy it no matter what VR system you're using!

This guide will help you with the package setup and customization process.

DEPENDENCIES: This asset require Unity Animation Rigging (com.unity.animation.rigging) if it's not installed automatically, please import it from Package Manager

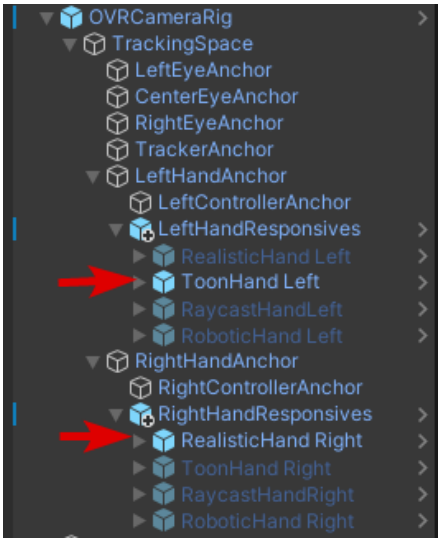
Quick Setup

1. You can drag and drop the "**LeftHandResponsives**" and "**RightHandResponsives**" prefabs under the tracking hand objects of you XR Rig



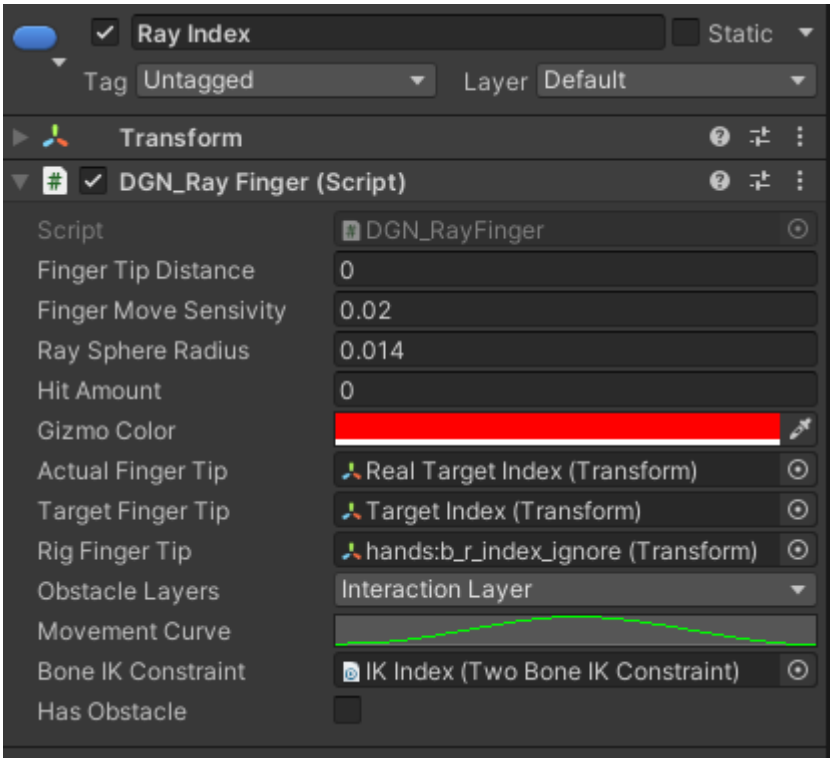
2. Add "**Interaction Layer**" to your Layer List (Layer Number: 8)

3. Enable/Disable Hand objects under prefab to try them out, you are ready!



Advanced Setup

All fingers can be setup differently. For example, index finger:



Finger Tip Distance: It's for debugging, you can check the distance between actual finger tip and posed finger's tip.

Finger Tip Distance: This is for debugging purposes. You can check the distance between the actual finger tip and the posed finger's tip.

Finger Move Sensitivity: This variable checks the distance on the surface points. When the user's actual finger tip moves away with this distance, the posed finger moves. If the user's actual finger tip doesn't exceed this distance, the finger rests on the recent surface point.

Ray Sphere Radius: Fingers seek surface points with spheres. Because the hand's model can be bigger and the fingers can be rounder, if the sphere radius is too low, fingers pass through the surface mesh. Fine-tune this variable for all the models.

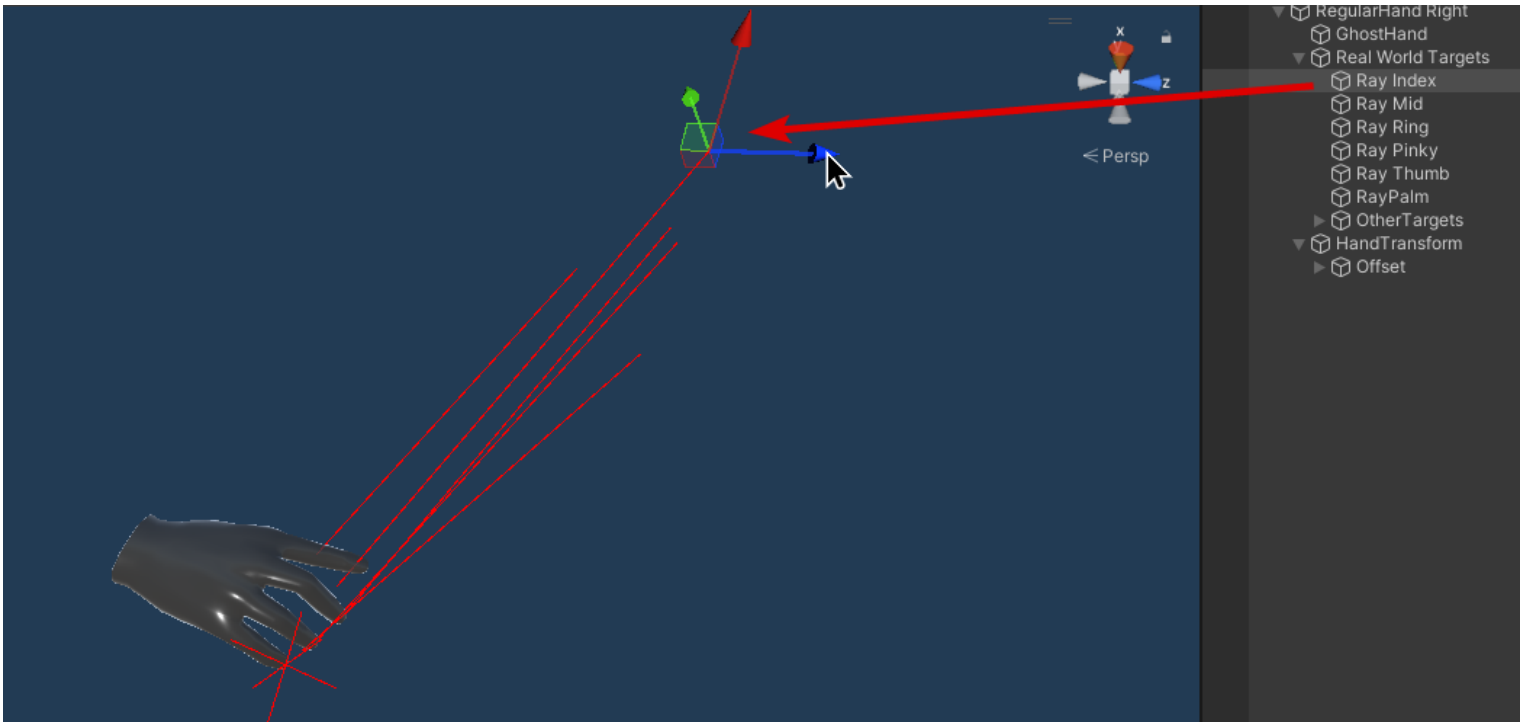
Hit Amount: This is for debugging purposes. You can check how many possible surface points the finger can land on with this variable.

Obstacle Layers: You can choose which layers the hands will interact with. It's recommended to specify certain layers.

Movement Curve: This curve adjusts finger posing with easing. You can change the curve to get more nervous or relaxed movement.

Has Obstacle: This is for debugging purposes. This will become true if the finger finds a proper landing point on surfaces.

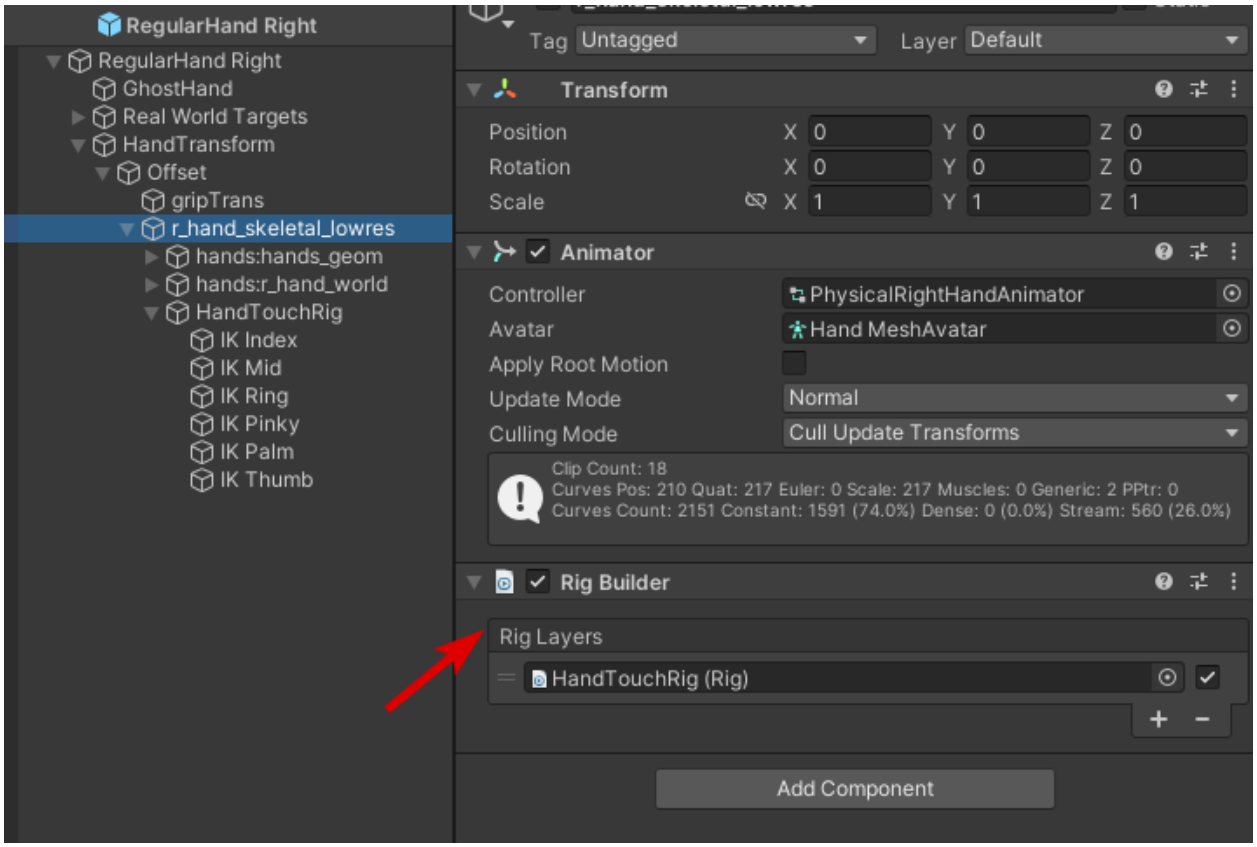
You can also adjust the ray angles of the finger with the inspector.



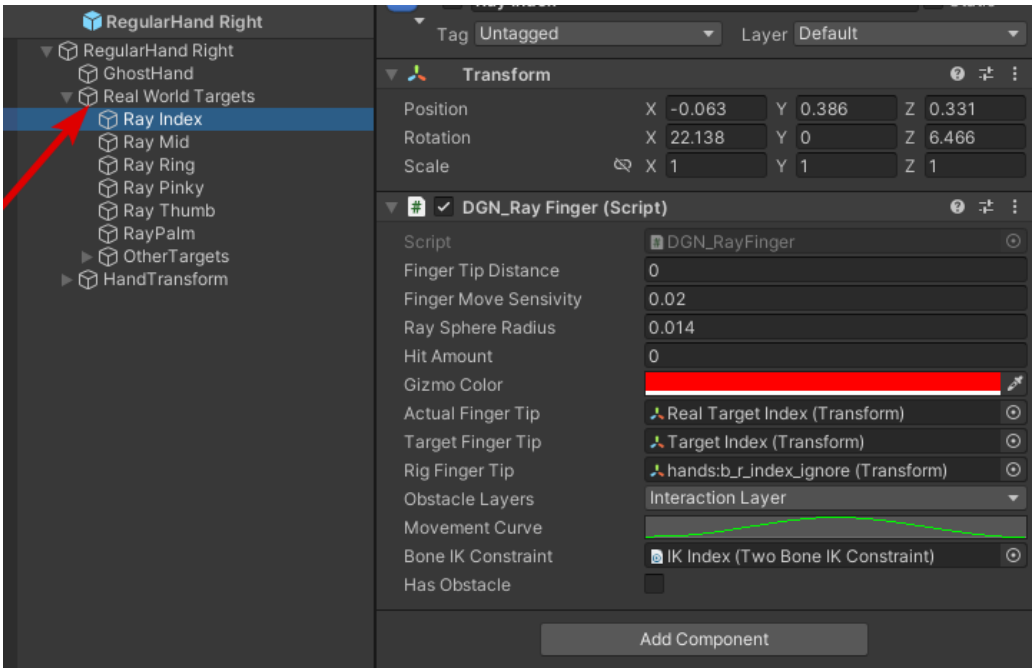
Custom Models

If you have your own rigged hand models, you can make responsive them too!

1. We'll need a proper Unity Animation rig for each hand. For more information, see https://learn.unity.com/tutorial/working-with-animation-rigging.

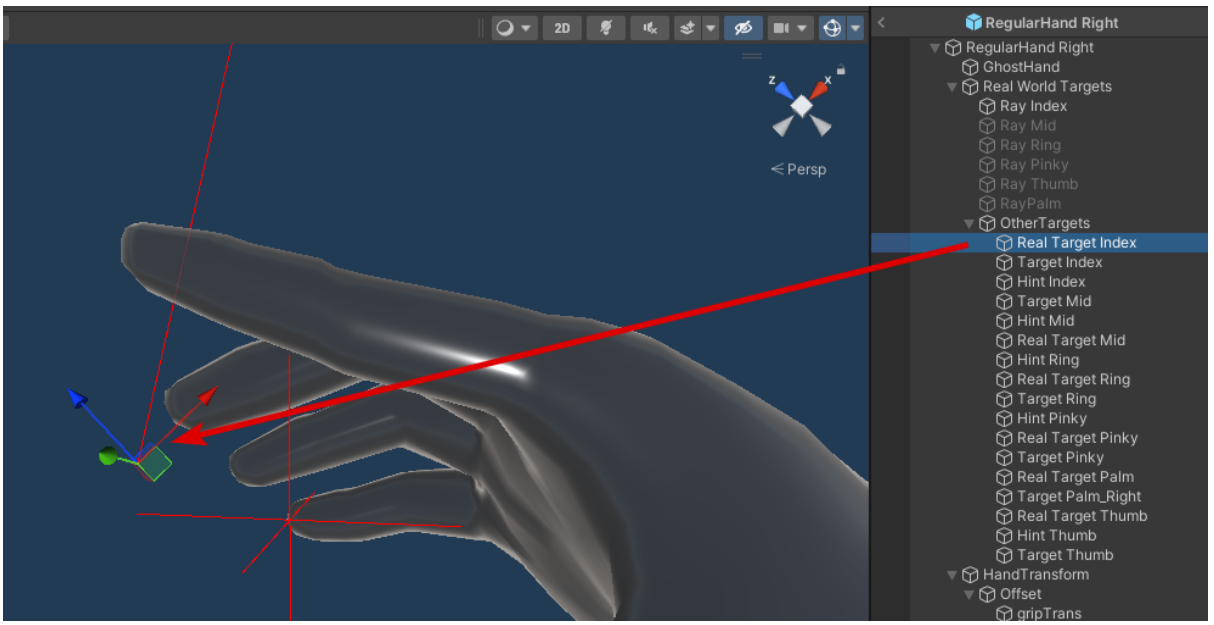


2. After we assign all the fingers and additional bones (sometimes palms are connected to wrists, so you'll need a palm IK too; you can check example models to get that), we're going to add our ray objects;

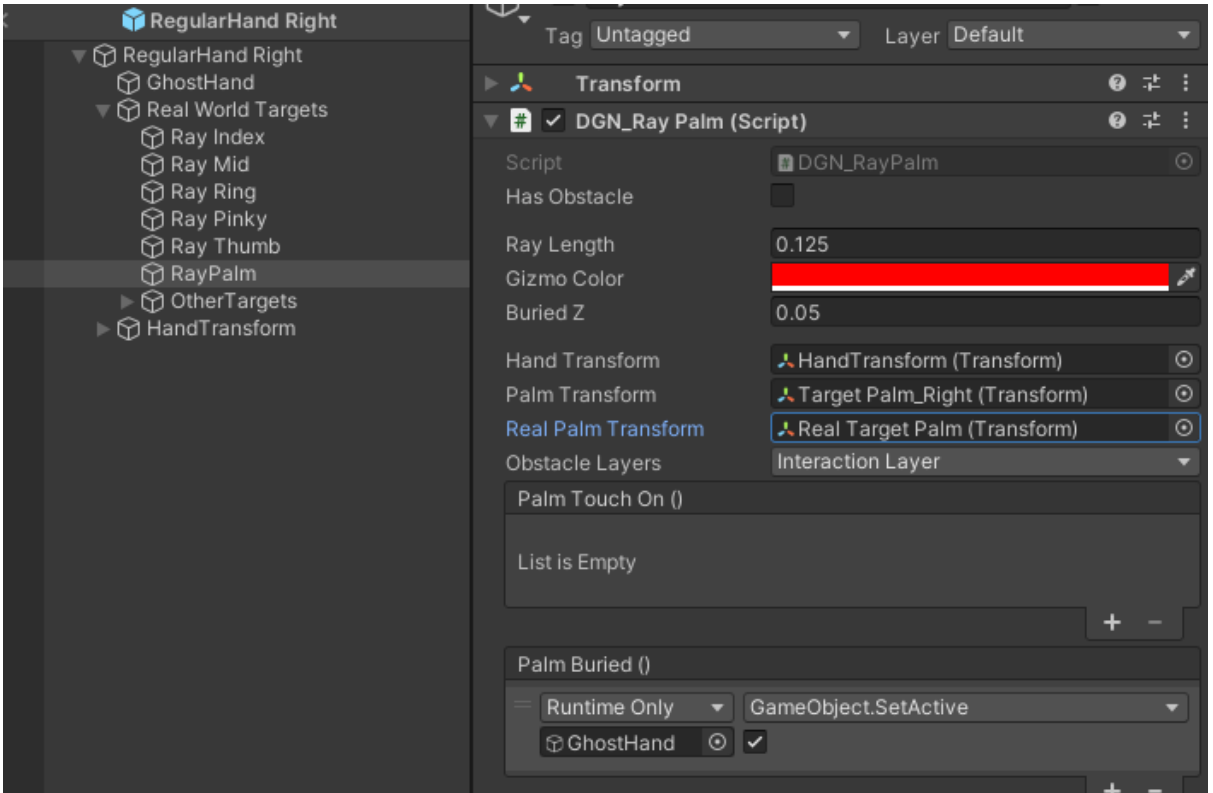


Important Note: Hierarchy is highly important for this IK system, so we can check our hierarchy with example models.

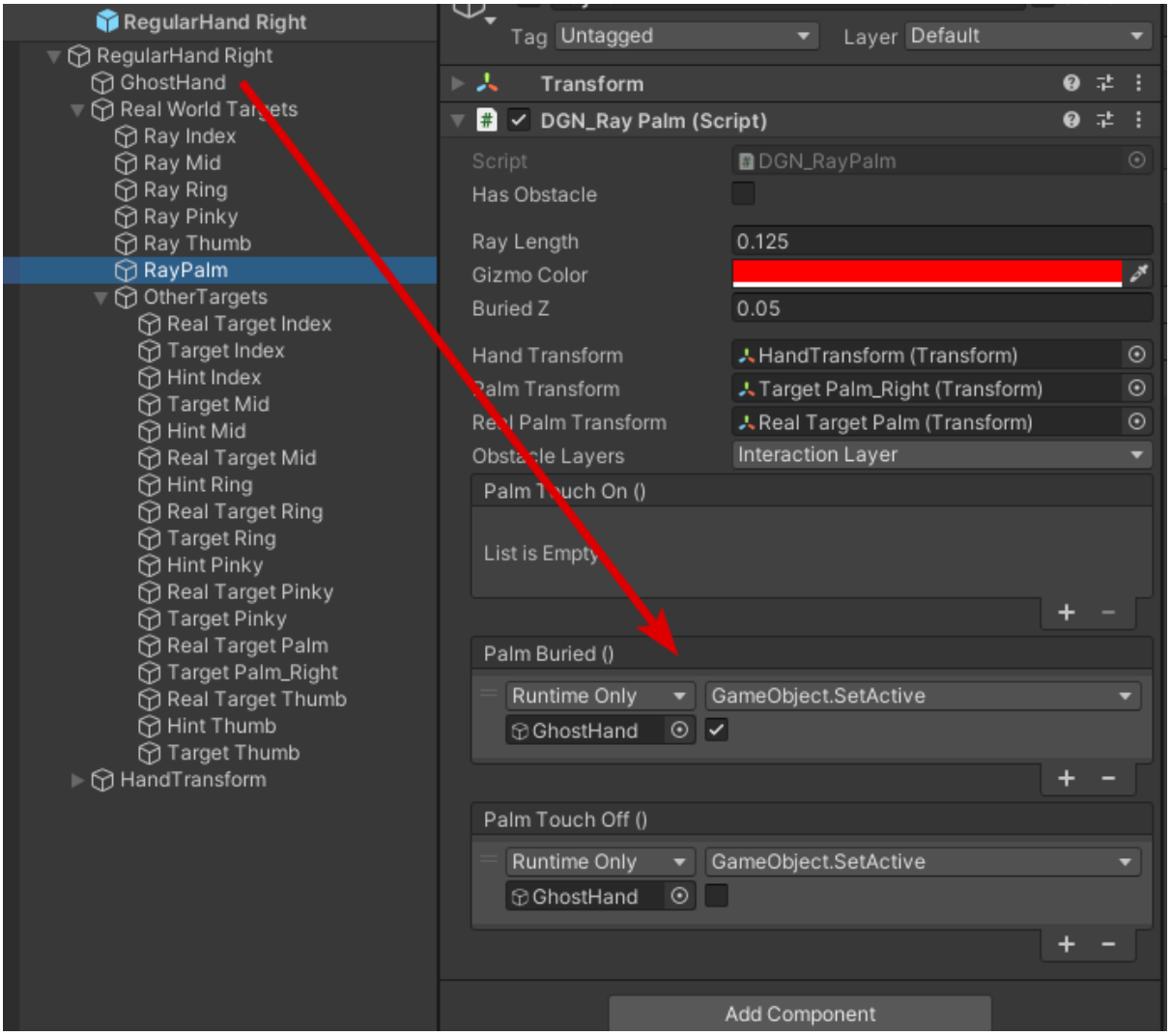
3. Rays need targets to pose the finger. These targets have to be placed in the right places. Again, we can check where to place them according to the hand with examples. **Real Target** will move with the user's controller input, while the Target movement will be calculated in the system. These **Targets** are also used by the **IK Rig's** targets. Hints are also used by **IK**.



4. Palm placing is controlled by another component called **DGN_RayPalm**. Hands not only land their finger, but their palm too. We need to place the ray above the hand's palm and adjust its Ray Length towards the palm. We can check examples to set up palms.



5. If we have ghost hands to show the user's real hand beneath the surfaces, we can use it from the palm's events;



6. We need to fine-tune each finger, its rays, and sensitivity. Check out the previous section in this document for advanced setup. After that, we are ready to use our custom responsive hands. We can also disable the rig in runtime to disable the responsive status of our hands!

Thanks for using VR Hands Responsive in your projects.

coded by Doğan Çetin
for IndieChest

info@sonsofearth.games