Introduction

Thank you for considering Sticky3D Controller for your game or project. With this asset we endeavour to save you many hours of development time and to help you solve common game play problems.

What is it?

Sticky3D Controller, or S3D for short, is a kinematic first- and third-person character controller for the Unity engine. S3D can be considered a “capsule” controller in that the character’s model will mostly occupy an area that has a capsule-like shape.

Although S3D uses many aspects of Unity’s in-built Physics features including a rigidbody, it does not fully rely on them for movement and interaction with other physics-based elements in a scene.

What does it do?

Many features can be enabled or disabled. This is by design as most features will incur some amount of performance overhead. Only enabling what you need can let you devote more resources to say rendering which is typically resource hungry. Here are just some of the main features:

- Stick to moving and rotating objects while performing “regular” actions
- Be controlled by player with Sticky Input Module
- Act as an NPC and be controlled by your game
- Be controlled by a NavMesh Agent
- Switch between first person and third person zoomable cameras
- Walk, sprint, strafe, jump and crouch
- Jet Pack with 6 degrees of freedom
- Configurable third person zoom
- Conditionally send data to your animator controller with no coding required
- Be configured and controlled via code using our API methods
- Interact / push dynamic rigidbody objects
- Be pushed by other objects
- Walk up and down steps and slopes
- Footstep sounds and surface actions
- Align to ground normal
- React to configurable gravity
- Override configuration based on zones within scene
- Perform custom actions and animations based on player input
- Work with your own character humanoid models
- Work with your own humanoid animations
- Take input from (new or legacy) Unity Input System, keyboard & mouse, or Rewired
- Look at objects or other S3D characters with Head IK
- Display information like messages or gauges in a UI for the player
- Look at, touch, grab, drop, and/or select interactive objects
- Pooling system for generic objects
- Popup menus for interactive-enabled objects
- Supports first-person Virtual Reality (continuous move, snap turn, teleportation and interaction)

What doesn’t it do or include?

Obviously, our asset cannot solve all game play challenges, otherwise we’d have written your game for you; where is the fun in that?! Unless we mention it in what S3D can do, please assume it cannot do something. If in doubt, ask us. Here are a few things that we DO NOT do or include at this time.

- Create character animations
- Move the character with Unity Physics (we use our own algorithms)
- Generally, you’ll need to supply your own character model (or buy one from the Asset Store)
- It doesn’t include a weapon system (although you can call your own code to fire a weapon using a Custom Input in the Sticky Input Module)
- It doesn’t include an inventory system (although you can call your own code to pick-up, use, and put down items using our Sticky Input Module)
- We do not include the Rewired asset (although we support using it). Rewired is a 3rd party asset and can be purchased separately if you want to use it with S3D. It is not required to use S3D.
- We do not include a preconfigured Rewired setup (as Rewired does not support creating one in code)
- We do not create your game – you still need to put in effort to make a game.

What versions of Unity do we support?

We following a sliding window which roughly matches the versions supported by Unity. We currently support Unity 2019.4.32+, 2020.x, 2021.1, and 2021.2.

Contents

Getting Started

S3D comes bundled with a number of demo scenes and prefabs to get you started quickly. The demo scenes are set up with a character controller and keyboard input, to allow you to simply open the scenes in the Unity editor and hit play.

WASD for movement and arrow keys or mouse for changing where you look. Check out the Sticky Input Module attached to the Sticky Control Module gameobject for other keyboard input like Jump, Crouch, Jet Pack, first/third person view toggle etc.

We'd suggest adding one of the character prefabs to your scene (e.g., PlayerBob), then creating a new original prefab from our prefab. That way you can modify the settings to see how it all works without them getting overridden when you apply a new S3D update or patch.

If you don't have time to read the whole manual, browse through it so that you can come back to at any time to clarify what a particular setting does. Most of the settings in S3D have brief editor tooltips.

The "Common Issues" chapter is regular updated and is a good place to go if your run into problems. If you can't find a solution, ask on our Unity forum or the Discord channel (see Support at end of this manual for details).

What's Changed

Version 1.0.9

[NEW] StickyManager - object management and pooling system - In Technical Preview
[NEW] Sticky Interactor Bridge for non-S3D character hands - in Technical Preview
[NEW] Sticky Generic Module - to use with object pooling system
[NEW] Sticky Popup Module - pooled clickable popup menus for interactive objects
[NEW] SampleGenericTextModule - shows how to create custom generic objects
[NEW] SamplePopupOptions - shows how to create popup menus for objects
[NEW] SampleMoveTo - shows how a character can be moved in code
[NEW] SampleSitActionPopup - shows how a character can sit on interactive objects
[NEW] DemoS3DTextSpawner - shows how to spawn custom generic objects
[NEW] First person camera option to follow head bone position changes
[FIXED] StickyXRInteractor can report incorrect object when object not Grabbable
[IMPROVED] Add StickyInteractiveTester directly from 3D Object menu
[IMPROVED] Revised readable levers

Version 1.0.8

[NEW] StickyInteractive - Is Readable option for virtual levers
[NEW] Animation API to play states with an offset from the beginning
[NEW] Blend in and out animator layers via API at runtime
[FIXED] 3rd person Free Look vertical movement glitches
[FIXED] Look Horizontal Damping has no effect
[FIXED] Hand Interact sample - displayMessage is null when Show Text disabled
[FIXED] NullReferenceException when Grabbing an object that is not Touchable
[FIXED] Reoccurring "Animator is not playing the AnimatorController" warning
[IMPROVED] 1st and 3rd person Free Look is out of Technical Preview
[IMPROVED] Include parameter name in Sticky Input linked Animation Actions

For the full change log, see Version History at the end of the manual.

Videos and Tutorials

Name	URL
Sticky3D – Get Started Tutorial	https://youtu.be/XGPzSQ61oMM
Reference Frames Tutorial	https://youtu.be/LwC0AQa7AcM
Add Your Model Tutorial	https://youtu.be/w7o17d7hcpc
Mixamo[1] Tutorial	https://youtu.be/an9qFX-i8xI
Sit Tutorial	https://youtu.be/UxIyJCFiuTQ
Interactive Basics Tutorial	https://youtu.be/azgpsSK-_D8
Short object Interaction video	https://youtu.be/ol4r7MteVCc
Promo 4 Trailer	https://youtu.be/JRcWTMsZZGU

Demos

For Demo Scripts, look in the "Runtime and API" chapter later in this manual.

Our demos are set up to work with the (legacy) input system is enabled. In newer versions of Unity, when you enable the (new) Unity Input System, by default the legacy system is disabled. Your options are:

- Try the demo scenes in a project with the legacy input system enabled
- In the Unity editor under Player settings, set "Active Input Handling" to "Both"
- Setup a character with the new Input System and use it in the demo scenes.

Cottage Camera Demo

This simple demo shows how to switch camera control between non-S3D control and S3D character control. Once the user gets control of the character, they can toggle between first and third person cameras at will.

Gravity Shooter Demo

Walk and/or jump between moving and rotating platforms. This scene shows the use of Sticky Moving Platforms with Sticky Zones. "Standard" WSAD moment keys apply as well as space key for jumping and "c" for crouching. Hold the "shift" key down to sprint.

You can fire the weapon with the left mouse button at the targets. If you hit them close enough to the centre they will be dislodged from their positions. The fire button is configured through a Custom Input on the Sticky Input Module. As this is a first person "game" we have disabled the "Switch Look Button" and configured the player to start with Third Person turned off.

NavMesh Demo

This scene show how a Non-Player Character (NPC) can be used with a NavMesh to patrol while also looking for the player in the scene.

NPC Look at Demo

This scene shows how you can get Non-Player Characters (NPC) to use Head IK to look at a player and optionally approach them. The "Sample NPC Play Anim" component is used to trigger an animation.

Playground Demo

This includes a collection of objects that you can use to test out your S3D character. To prevent the platform from rotating, under the Environment gameobject, on the Platform, change the Rigidbody to "Is Kinematic".

By default, the seesaw is turned off because the hinge joint configuration does not work correctly when the platform is rotating. To test the character with the seesaw, change the Platform Rigidbody to "Is Kinematic" before enabling the seesaw.

[1] Mixamo and Adobe are either registered trademarks or trademarks of Adobe in the United States and/or other countries.

Sticky Control Module

This module is the primary component or brain of the controller. It is responsible for:

- Character movement
- Camera movement
- Interaction with other physical bodies in the scene
- Jet Pack operations
- Animation management

Move – General Settings

This tab contains settings for most character movement. For Jet Pack movement, see the Jet Pack tab.

Property (General)	Description
Initialise on Awake	If enabled, Initialise() will be called as soon as Awake() runs. This should be disabled if you want to control when the Sticky Control Module is enabled through code.
Non-Player Character	Is this a non-player character? Set when you want to drive input via code.
Character Health	Overall health value of the character. 0 = no health, 100 = full health. Health will affect the speed the character can move.
Walk Speed	The speed at which the character can walk in metres per second.
Sprint Speed	The speed at which the character can sprint or running in metres per second.
Strafe Speed	The speed at which the character can strafe left or right in metres per second.
Jump Speed	The initial speed added to the character when jumping in metres per second
Jump Delay	The number of seconds the character delays jumping upward. This is useful when a jump animation includes an initial movement when the feet are still on the ground.
Max Acceleration	The maximum character movement acceleration in metres per second per second. For humanoids, we'd suggest a value between 20 and 25.
Max Step Offset	The maximum height of an object the character can step up onto or over without jumping. It must be less than half the height of the character.
Step Up Speed	The speed at which the character rises up a step.
Step Up Bias	Dynamically changes the step-up speed while sprinting up steps. Default value is 1.0 which makes the step-up speed directly proportional to the sprinting speed. It has no effect when walking up steps or going down steps.
Max Slope Angle	The maximum slope in degrees, that the character can walk up
Align to Ground Normal	Will the character's up direction attempt to align with the ground normal? If enabled, Vertical Rotation Rate will apply.
Vertical Rotation Rate	How quickly, in degrees per second, the character will attempt to match the target Up direction. Used for reference frame normal and ground normal matching.
Turn Rotation Rate	How quickly, in degrees per second, the character will attempt to match the Free Look camera direction.
Allow Movement in Air	Whether movement is allowed while in the air (while jumping)
Gravity	The gravitational acceleration, in metres per second per second, that acts downward for the character
Arcade Fall Multiplier	This can add a retro feel to your player when values are greater than 0. Values less than 0 can give the character a parachute-like feel when falling.
Stuck Time	The amount of time that needs to elapse before a stationary character is considered stuck. When the value is 0, a stationary character is never considered stuck.
Stuck Speed Threshold	The maximum speed in m/sec the character can be moving before it can be considered stuck.
Move Update Type	The update loop or timing to use for moving the character.

Move – Climbing

This feature is currently in **Technical Preview** and its behaviour may change without notice with each release to correct issues or to address particular scenarios. If you see a problem, please contact us on our Unity forum or on our Discord channel.

Currently this feature is in the early days of development and is best used with near vertical walls. Overhangs are not supported but the aim is to provide this in a future release.

Property (Climbing)	Description
Climbing	Is the character able to climb walls?
Climb Speed	The speed at which the character can climb, in metres per second.
Min Slope Angle	The minimum slope, in degrees, that the character can climb. It has no effect if it is less than the general walkable Min Slope Angle.
Max Slope Angle	The maximum slope, in degrees, that the character can climb.
Max Grab Distance	The maximum distance in-front of the character that they can reach out to grab a climbable surface when not climbing.
Face Surface Rate	The rate at which the character turns to face the surface they are climbing.
Top Detection	Detect when the characters shoulders have reached the top of a climbable object. Shoulder height is set on the Collide tab.
Climbable Layer Mask	The character can climb objects (colliders) in the scene that are in one of these Unity Layers, provided that they are also in the Collision Mask Layer on the Collide tab. If most objects in your scene are climbable, it may be better to create a “Unclimbable” Unity Layer and just exclude that from this Layer Mask.

Move - Footsteps

Footsteps can be configured to use the speed the character is moving, or it can use when the feet are placed on the ground.

Property (Footsteps)	Description
Footsteps	Is the character using footstep sounds and/or effects?
Use Move Speed	Rather than using foot placement, use the character moving speed
Walk Frequency	This controls the relative foot step frequency assuming the walk speed is 1.0
Sprint Frequency	This controls the relative foot step frequency assuming the sprint speed is 1.0
Left Foot	The (child) left foot transform of the character (applies when “Use Move Speed” is not enabled).
Right Foot	The (child) right foot transform of the character (applies when “Use Move Speed” is not enabled).
Audio Source	The audio source containing the clips to play when the footsteps are used. Must be a child of the character gameobject.
Default Footstep Sound	The default footstep sound when the walking surface is unknown
Overall Footstep Volume	The overall volume of footsteps. Individual footstep surface action volumes are relative to this overall volume. For example, if a relative volume is 0.5, and the overall volume is 0.5, the volume set will be 0.25 (half of the 0.5 overall volume).
Known Surface Types	A shared Scriptable Object in the Project containing a list of common surface types

Footsteps can have optional “Surface Actions”. These can include one or more audio clips that override the default footstep sound.

Property (Surface Action)	Description
Min Volume	The relative minimum volume of the audio clips
Max Volume	The relative maximum volume of the audio clips
Min Pitch	The minimum pitch of the audio clips

Property (Surface Action)	Description
Max Pitch	The maximum pitch of the audio clips
Mesh Surface Types	A list of Surface Types from the “Known Surface Types” scriptable object. Walking on a collider that has a matching Surface Type set on the Sticky Surface component attached to that object, will trigger one of the Audio Clips to play.
Terrain Textures	The case-sensitive (albedo) terrain texture names and the minimum relative weight at a terrain position required to trigger a hit. These are hashed for runtime optimisation so make sure they EXACTLY match the texture names. When in doubt, cut and paste.
Audio Clips	A list of one or more audio clips (chosen at random) to override the Default Footstep Sound.

Move – Identification

These properties help you identify and distinguish between different characters in your scene. At runtime, you can also use the StickyID property to uniquely identify an individual character.

Property (Identification)	Description
Faction ID	The faction or alliance the character belongs to. This can be used to identify if a character is friend or foe. Neutral = 0.
Model ID	The type, category, or model of the character. Can be useful if you have a group of characters with similar attributes.

Identity information can be used in filters on Sticky Zones and/or in your own game code.

Look – First and Third Person

This tab contains settings for the first and third person modes. First person settings are visible when “Third Person” is selected, otherwise Third Person settings are show. At runtime, it is possible to switch between First and Third Person modes, provided that they have been correctly configured.

Property (Common)	Description
Look on Initialise	Is look enabled when the module is first initialised? This will only take effect if the Look Camera is configured.
Third Person	Is this in 3rd person controller mode?
Free Look	Is the free look mode enabled? NOTE: When enabled, Orbit settings in third person have no effect. This currently is not supported when Root Motion is enabled.
Look VR Mode	Is the VR mode enabled? This only works when Sticky Input Module is set to “UnityXR”.
Horizontal Speed	The speed or rate the character can look left or right
Horizontal Damping	The amount of damping applied when starting or stopping to look left or right
Vertical Speed	The speed or rate the character can look up or down
Vertical Damping	The amount of damping applied when starting or stopping to look up or down
Show Cursor	Show the screen cursor or mouse pointer
Auto-hide Cursor	Automatically hide the screen cursor or mouse pointer after it has been stationary for a fixed period of time. Automatically show the cursor if the mouse if moved.
Hide Cursor Time	The number of seconds to wait until after the cursor has not moved before hiding it
Zoom Duration	The time, in seconds, to zoom fully in or out
Unzoom Delay	The delay, in seconds, before zoom starts to return to the non-zoomed position
Max LoS Field-of-View	Maximum line-of-sight field-of-view is the angular range that objects are in sight of the character.

Property (First Person)	Description
Look Transform	The parent transform used for look direction. Anything that should have its position or orientation modified by look direction should probably be parented to this transform
Look Camera	The main first-person camera which is a child of the controller
Auto Camera Height	Automatically adjust the 1st person camera to the average eye height, based on the height of the player
Follow Head Position	If the relative head bone position changes, the first person camera will move relative to it.
Pitch Up Limit	The pitch limit for look upward direction in degrees
Pitch Down Limit	The pitch limit for look downward direction in degrees
Zoomed FoV	The camera field-of-view when the first-person camera is fully zoomed in.
Un-zoomed FoV	The camera field-of-view when no zoom is applied to the first-person camera.

Property (Third Person)	Description
Look Camera	The main third person camera which should not be a child of, or attached to, the controller
Camera Offset	The camera offset or distance from the character when in third person mode. Clicking the (G)izmos button will show a small green dot in the scene view with the desired offset (currently this is a non-clickable gizmo).
Focus Offset	The local space point on the character where the third person camera focuses relative to the origin or pivot point of the character prefab. Clicking the (G)izmos button will show a small blue dot in the scene view with the desired focus point (currently this is a non-clickable gizmo).
Pitch Up Limit	The pitch limit for look upward direction in degrees (Free Look only)
Pitch Down Limit	The pitch limit for look downward direction in degrees (Free Look only)
Orbit Duration	The time, in seconds, to fully orbit the character
Unorbit Delay	The delay, in seconds, before orbiting camera starts to return to the default position
Orbit Damping	The amount of damping applied when starting or stopping camera orbit in third person.
Orbit Min Angle	The minimum anti-clockwise angle to rotate the camera around the character
Orbit Max Angle	The maximum clockwise angle to rotate the camera around the character
Max Shake Strength	The maximum strength of the third person camera shake. Smaller numbers are better.
Max Shake Duration	The maximum duration (in seconds) the third person camera will shake per incident.
Clip Objects	Adjust the camera position to attempt to avoid the camera flying through objects between the character and the camera. This has performance overhead, so disable if not needed.
Minimum Move Speed	The minimum speed the camera will move to avoid flying through objects between the character and the camera. High values make clipping more effective. Lower values will make it smoother.
Minimum Distance	The minimum distance the camera can be from the character position.
Minimum Offset X	The minimum offset on the x-axis the camera can be from the character when object clipping. This should be less than or equal to the Camera Offset X value.
Minimum Offset Y	The minimum offset on the y-axis the camera can be from the character when object clipping. This should be less than or equal to the Camera Offset Y value.
(Clip) Responsiveness	The responsiveness to changes in the clipping distance. When 1.0, it will depend on the Min Move Speed and the Camera Move Speed. Reducing the responsiveness can stabilise clipping when the character is moving erratically or on a vehicle that is.
Clip Object Layers	Only attempt to clip objects in the following Unity Layers

Look – Interactive

This allows your character to look around the scene and discover interactive-enabled objects that have a Sticky Interactive component attached. To be seen, interactive-enabled objects must have a non-trigger collider on the same gameobject as the StickyInteractive component. See also the chapter called "Sticky Interactive".

To get the character to take action, for players, typically you will want to add a Custom Input on the Sticky Input Module, and call one of the extensive Interactive APIs. Some of these can be seen in the SampleHandInteract script.

Property	Description
Layer Mask	The non-trigger colliders and characters in these Unity layers can be seen when Look Interactive is enabled.
Max Distance	When look interactive is enabled, how far away from the camera can the character see objects with a StickyInteractive component? Try to keep this distance as short as possible to avoid hitting too many colliders in front of the camera. When "Look VR" is enabled, set this on the StickyXRInteractor component for each hand.
Update Looking Point	When true, GetLookingAtPoint is updated. This is the point in world space where the user is currently aiming or targeting. It could be an interactive-enabled object.

To test if your character can "see" interactive-enabled object, in play mode in the Unity editor, with "Maximize On Play" not set in the Game view, click on your S3D character in the Hierarchy and click "Debug Mode" on the Sticky Control Module. As the character looks around the scene you will be able to monitor the "Looking At" field.

Look – VR Mode

This feature is currently in **Technical Preview** and its behaviour may change without notice with each release to correct issues or to address particular scenarios. If you see a problem, please contact us on our Unity forum or on our Discord channel.

These properties let you change how the character looks and turns when configured for VR. See also the "Unity XR" section of the chapter on "Sticky Input Module" and the "Sticky XR Interactor" chapter.

Property (VR)	Description
Match Human Height	When Look VR is enabled, the character Height will be modified to match the approximate height of the human player based on the starting head-mounted device position above the floor.
Human Posture	The posture or starting position of the human player when wearing a VR head-mounted device. E.g., Sitting or Standing. This assumes that the VR device has been calibrated with the human player the given posture.
Snap Turn	When enabled with UnityXR input, left and right turn will be incremented by the Snap Turn Amount.
Snap Turn Amount	The number of degrees turned with each snap movement.
Snap Turn Interval	The minimum amount of time required between snap turns.

If you have a character model, you may wish to adjust the "Look Transform" from "Look General Settings". Bring the camera forward on the z-axis so you cannot see the inside of the character's head. You may also need to adjust the Camera near "Clipping Planes".

Collide

The collide tab is use to help configure how the character interacts with the other objects in the scene.

Property	Description
Height	The height of the character collider in metres
Radius	The radius of the character collider in metres

Property	Description
Pivot to Centre Y	The distance, in the up direction, from the pivot point to the centre of the model. If the pivot point is at the feet, this will be half the height.
Crouch Height	The height of the character when crouching
Max Sweep Iterations	The maximum number of sweep iterations allowed per frame
Sweep Tolerance	The tolerance allowed for sweeps and grounded checks in metres
Collision Layer Mask	The layer mask used for collision testing for the character. If your project doesn't have a layer 29, S3D will automatically create one called "Interactable" when the demo scenes are imported into your project.
On Trigger Enter/Exit	Call OnTriggerEnter or Exit when character enters or exits a Trigger Collider in the scene. This must be enabled if using StickyZones.
On Trigger Stay	Call OnTriggerStay EVERY FRAME while the character is inside a Trigger Collider. If you don't need this, keep it turned off.
Trigger Collider	Rather than disabling the capsule collider at runtime, it is converted to a trigger collider so that it can be detected by raycasts.
React to Sticky Zones	When entering or exiting a Sticky Zone, allow configuration changes based on zone settings.
Interaction Layer Mask	The character will attempt to interact with dynamic rigidbodies in these layers. Default: Nothing. If the character collides with a lighter dynamic rigid body, the other object may be pushed away from the character. Generally, should NOT include any layer from the Reference Layer Mask.
Reference Layer Mask	The Unity Layers used to test if the object under the character is a suitable Reference Frame
Reference Update Type	Determines how the reference frame is updated. Manual = via your code or a StickyZone. When type is Automatic, S3D will detect the collider under the character. If the collider under the character changes AND it is not a child object of the current Reference Frame, the Reference Frame will be changed. AutoFirst will automatically detect the object under the character when initialised, then will allow you to change it manually in code or via a StickyZone.
Initial Reference Frame	Initial or default reference frame transform the character will stick to.

Reference Frames Explained

The character uses a Reference Frame transform to determine relative movement. When the Reference Frame moves, S3D moves too. Movement is relative to the Reference Frame. You can still walk, sprint, jump, crouch etc. as if the object you are riding on is not moving and/or rotating.

Your character can move between different objects that can move independently from one another. Consider your character walking around in a space ship, docking with a space station, then walking around in the space station. Another scenario is climbing onto a vehicle moving in one direction, then transferring to say a train moving along railway lines. Reference Frames are the "magic" that makes this possible without the character having to be made a child of the moving object. The character is still able to move around the moving or stationary vehicles and perform all the regular actions your character can normally do.

You can set the Reference Frame using Sticky Zones in your scene, via your game code, or set it as "Automatic".

Jet Pack

The jet pack feature enables the character to take off from the ground and hover. Using familiar controls, the character can then fly around.

The jet pack can have from zero to six visible thrusters. These optional particle effects are made children of the main character prefab. They have no direct influence on the actual character movement but can provide more visual elements to your character. As a starting point, they can be quickly placed in the correct place relative to the character position by clicking the "Auto" button next to each thruster in the inspector. They can then be moved manually in the editor to align them exactly with your character's jet pack model (assuming you have one).

Property	Description
Is Available	Is the Jet Pack feature selectable by the player?
Is Enabled	Is the Jet Pack currently engaged?
Fuel Level	The amount of fuel available to power the Jet Pack
Fuel Burn Rate	The rate fuel is consumed per second. If rate is 0, fuel is unlimited
Max Speed	Maximum speed the jet pack can propel the character
Max Acceleration	The maximum jet pack acceleration in metres per second per second.
Damping Force	The inertia damping force applied to slow down movement when no input is received
Ramp Up Duration	The number of seconds it takes for this jet pack to go from minimum to maximum power.
Ramp Down Duration	The number of seconds it takes for this jet pack to go from maximum to minimum power.
Audio Source	The audio source containing the clip to play when the jetpack is used. Must be a child of the character gameobject.
Health of Jet Pack	Health value of the jet pack. 0 = no health, 100 = full health. A damage jet pack will burn the same amount of fuel but will produce less thrust.
Min Effects Rate	The 0.0-1.0 value that indicates the minimum normalised amount of any particle effects that are applied when a non-zero jet pack input is received. Default is 0. If the full particle emission rate should be applied when any input is received, set the value to 1.0.
Push Forward	The parent gameobject for the backward-facing (particle) thruster effect for the jet pack. Must be a child of the character gameobject.
Push Backward	The parent gameobject for the forward-facing (particle) thruster effect for the jet pack. Must be a child of the character gameobject.
Push Up	The parent gameobject for the downward-facing (particle) thruster effect for the jet pack. Must be a child of the character gameobject.
Push Down	The parent gameobject for the upward-facing (particle) thruster effect for the jet pack. Must be a child of the character gameobject.
Push Right	The parent gameobject for the left-facing (particle) thruster effect for the jet pack. Must be a child of the character gameobject.
Push Left	The parent gameobject for the right-facing (particle) thruster effect for the jet pack. Must be a child of the character gameobject.

Animate - General

This tab helps you hook up your character animations to the S3D Controller.

NOTE: S3D does not create the animation clips for you, nor does it configure the Unity Animator. We assume that the character you have made or have purchased, comes with a pre-configured animation controller and clips.

The aim of this tab is to let you configure S3D to tell your animation controller what to do when. S3D will work with "standard" Unity Animator states and/or Blend Trees.

Property or Button	Description
Animator	The Unity animator component that will control animations for this character.
Foot IK	Foot Inverse Kinematics (or Foot IK for short), helps place a humanoid character's feet on the ground. See the "Animate – Foot IK" section below for details.

Property or Button	Description
Root Motion	Is Animation Root Motion used to influence character position and rotation? Some animations for third parties use "Root Motion" to move the character. See the "Animate – Root Motion" section below for details.
Refresh Button	Refresh the Parameter Names from the Animator Controller. This is useful if you have changed the parameters in your Animator Controller or the S3D GameObject was disabled and you have re-enabled it.
+ Button	Add a new Animate Action to then end of the list.
- Button	Remove (delete) the last Animate Action from the end of the list.

Each Animate Action is used to send data from the S3D Controller to your character animation controller. Sometimes you may need multiple Animate Actions to perform a single activity. E.g., To make your character walk, you might need to tell your character it is Grounded AND it should have a MoveSpeed. This would be done using two Animate Actions.

Action Property	Description
Standard Action	This is the action that happens that causes the animation to take place. It helps you remember why you set it up. Custom Actions can be controlled via code or user input. For more information see Animate – Custom Input and Actions below.
Parameter Type	The type of animation parameter, if any, used with this action
Parameter Name	The parameter name from the animation controller that applies to this action
Invert	Works with bool types. When the value is true, use false instead. When the value is false, use true instead. Not compatible with Toggle
Toggle	Works with bool custom anim actions to toggle the existing parameter value in the animator controller. Not compatible with Invert or "Reset After Use".
Bool Value	The real-time Boolean value from the Sticky3D Controller that will be sent to the model's animation controller
Float Value	The real-time float value from the Sticky3D Controller that will be sent to the model's animation controller
Trigger Value	The real-time trigger value from the Sticky3D Controller that will be sent to the model's animation controller
Integer Value	FUTURE
Float Multiplier	A value that is used to multiple or change the value of the float value being passed to the animation controller. This is useful if you want to modify the speed at which an animation clip is played. DEFAULT: 1.0
Damping	The damping applied to help smooth transitions, especially with Blend Trees. Currently only used for floats. Used with Float values only. DEFAULT: 0.1. For quick transitions to the new float value use a low damping value, for the slower transitions use more damping.
Reset After Use	Only uses with Custom bool types (see below). By default, this is true and sets Boolean values back to false after animate actions are processed each frame. If you notice that your values are returning to say false in your animator controller when you expect them to be true it may be a timing issue between user input and when fixed update runs. Try setting this to false, and see if it corrects the behaviour.

When determining which values to send to your animation controller, it can be helpful to determine what type of data you need. There are three key types:

- In-motion values which typically have "ing" in their name. E.g., MovingSpeed, MovingDirectionX, SprintingSpeed, etc.

- Input values which typically have "Input" in their name. E.g., MovementInputX, MovementInputZ, LookVerticalInput, LookZoomInput, etc.
- Values you configure in the editor or at runtime e.g., WalkSpeed, SprintSpeed, StrafeSpeed, JumpSpeed, etc.

Conditions can be added to an Animate Action to instruct S3D to only send instructions when certain conditions are met. An example could be you don't want your character to be animated when in Jet Pack mode, so you place a condition of Is Grounded for the Move Speed.

Condition Property	Description
AND/NOT	This Animate Action only happens when the following IS or IS NOT true
Property	The property or variable in S3D to check

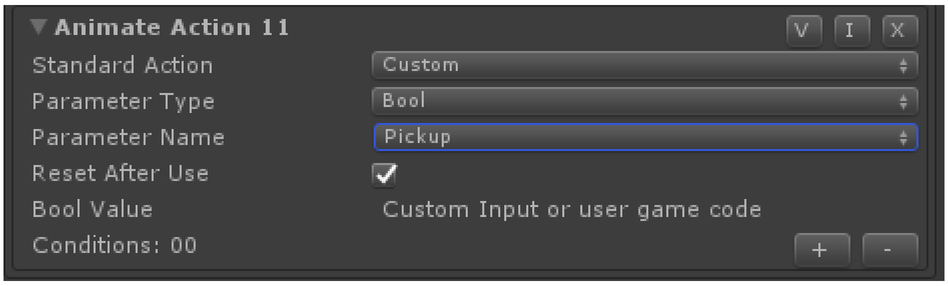
Conditions can be helpful to restrict what data gets sent each frame. Some conditions like HasLanded and HasJumped can help to identify a specific frame in which an animation should start. These two examples only occur on the frame the event happened rather than something like IsGrounded which typically occurs over multiple frames. This type of condition is useful when say using a Jump or Landing Trigger parameter in an animation.

NOTE: When setting up conditions consider that some Parameters in your animator controller may need to be always set – for example IsGrounded or MovingSpeed.

Animate – Custom Input and Actions

Sometimes you may wish to animate your character when the user presses a particular button on the gamepad or the keyboard. At the same time, you may want to perform some kind of action in the game like pick up or drop an object, or maybe throw an object like a spear or javelin.

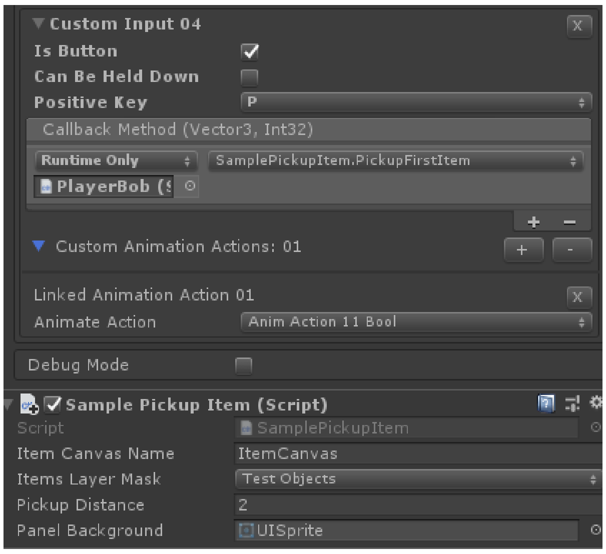
Custom Actions are similar to other Standard Actions in that they change Parameter values in your Unity Animator. However, instead of getting the real-time value from data inside StickyControlModule, they get it from either a user input (via StickyInputModule) or via an S3D API method like SetCustomAnimActionBoolValue(..).


Animate Action 11
Standard Action: Custom
Parameter Type: Bool
Parameter Name: Pickup
Reset After Use: ✓
Bool Value: Custom Input or user game code
Conditions: 00

If this is a human player, on the StickyInputModule, you would create a Custom Input and have it call your custom code. In the example to the left, we are picking up the first item that are in front of the player.

When the user presses the button, the StickyInputModule will tell the StickyControlModule to set the parameter called "Pickup" on your Animator to true. Assuming that you have an animation for pickup, and you have configured your Animator Controller correctly, the pickup animation will also play.

If you have an NPC character, you can call our Animate API methods to achieve a similar result.


Custom Input 04
Is Button: ✓
Can Be Held Down: ☐
Positive Key: P
Callback Method (Vector3, Int32)
Runtime Only | SamplePickupItem.PickupFirstItem
PlayerBob (S
Custom Animation Actions: 01
Linked Animation Action 01
Animate Action: Anim Action 11 Bool
Debug Mode: ☐
Sample Pickup Item (Script)
Script: SamplePickupItem
Item Canvas Name: ItemCanvas
Items Layer Mask: Test Objects
Pickup Distance: 2
Panel Background: UISprite

Animate – Head IK

Head Inverse Kinematics (or Head IK for short), helps place a humanoid character's head look towards a target position or object in the scene. The character must be rigged as a Humanoid character and must include an animation controller.

Property or Button	Description
Anim IK Pass Layer	The zero-based layer in the animator controller that has the IK Pass enabled for Head IK
Head Move Speed	The rate at which the character's y-axis position is adjusted for foot IK placement.
Damping	The amount of damping to apply when starting or stopping to turn the character's head toward a target position.
Look Down Limit	The maximum rotation, in degrees, the head can be tilted down.
Look Up Limit	The maximum rotation, in degrees, the head can be tilted up.
Look Left Limit	The maximum rotation, in degrees, the head can look left.
Look Right Limit	The maximum rotation, in degrees, the head can look right.
Eyes Weight	How much the eyes are used to look towards the target.
Head Weight	How much the head is used to look towards the target.
Body Weight	How much the body is used to look towards the target.
Look at Eyes	When the Head IK target is a character, look toward their eyes.
Look at Interactive	When look interactive and Update Looking Point are enabled in the Look tab, while the character is stationary, should the head face either the interactive enabled object being looked at, or in the direction the character is looking?
Adjust for Velocity	When the character or the target is moving quickly this may help the head adjust quickly to the rapidly changing positions. This may impact performance so only enable when required.
When Climbing	If Head IK is enabled, the head (also) turns towards the target while climbing.
When Move Disabled	If Head IK is enabled, the head can turn towards the target while the character movement is disabled. For example, when the character is seated.
Consider Behind	Will the character's look direction be affected if the target is behind them?

In your animation controller, ensure "IK Pass" is enabled on your main movement layer.

For testing in your scene, this can be combined with the SampleHeadIKTarget or SampleLookAtPlayer scripts.

For an example setup, see the NPCLookAtDemo scene.

Using with Optimize Game Objects.

On the model FBX, "Rig" tab in the Unity Inspector, if you have "Optimize Game Objects" enabled, you will need to enable the head bone. If you don't, a couple of warnings will appear in the Unity editor console window at runtime. To find the head bone name, go into "Configure" on the FBX Rig tab. Once you have the bone name (e.g., Spine.05), enable it under "Extra Transforms to Expose" in the FBX Rig tab.

Animate – Hand IK

This feature is currently in **Technical Preview** and its behaviour may change without notice with each release to correct issues or to address particular scenarios. If you see a problem, please contact us on our Unity forum or on our Discord channel.

Hand Inverse Kinematics (or Hand IK for short), helps move a humanoid character's hand towards a target position or object in the scene. The character must be rigged as a Humanoid character and must include an animation controller. Typically, you will use it in conjunction with Sticky Interactive objects.

Property or Button	Description
Hand IK	Are the hands moved using Inverse Kinematics for Humanoid rigged characters?
Anim IK Pass Layer	The zero-based layer in the animator controller that has the IK Pass enabled for Hand IK
Hand Move Speed	The maximum rate at which the character's hands move toward the target position.
Hand Radius	How close an object must be to a hand before it is considered to be touching it.
Gizmo Colour	The colour of the gizmos shown in the scene view at runtime
When Move Disabled	The hands can move while the character movement is disabled. For example, when the character is seated or stationary.
Left / Right Hand	
(F)ind button	Find / highlight the palm of the hand in the scene view
(G)izmo button	Toggle on/off the gizmos for this hand in the scene view
Palm Offset	The local space offset the palm, or centre, of the hand is from the hand bone position
Palm Rotation	The local space hand palm rotation from the hand bone stored in degrees. The palm normal, stored as a rotation.
Max In Rotation (left)	The Hand IK left hand (wrist) max rotation (to the right) in degrees
Max In Rotation (right)	The Hand IK right hand (wrist) max rotation (to the left) in degrees
Max Out Rotation (left)	The Hand IK left hand (wrist) max rotation (to the left) in degrees
Max Out Rotation (right)	The Hand IK right hand (wrist) max rotation (to the right) in degrees
Max Up Rotation	The Hand IK hand or wrist max rotation up in degrees
Max Down Rotation	The Hand IK hand or wrist max rotation down in degrees
Inward Limit (left)	The Hand IK left hand inward movement (to the right) limit in degrees.
Inward Limit (right)	The Hand IK right hand inward movement (to the left) limit in degrees
Outward Limit (left)	The Hand IK left hand outward movement (to the left) limit in degrees. Currently the left hand cannot reach behind the character.
Outward Limit (right)	The Hand IK right hand outward movement (to the right) limit in degrees. Currently the right hand cannot reach behind the character.
Max Reach Dist.	The maximum distance the hand will attempt to reach for an object.

In your animation controller, ensure “IK Pass” is enabled on the layer that animates the hands and arms.

The palm offsets and rotations can be updated with the Inspector values in the table above or they can be changed by selecting, then moving the gizmos around in the scene view while not in play mode.

We have also included an extensive runtime API with many methods and properties that can be set and called from your own game code. See the “Runtime and API” chapter later in this manual for details.

Animate – Foot IK

This feature is currently in **Technical Preview** and its behaviour may change without notice with each release to correct issues or to address particular scenarios. If you see a problem, please contact us on our Unity forum or on our Discord channel.

Foot Inverse Kinematics (or Foot IK for short), helps place a humanoid character’s feet on the ground. The character must be rigged as a Humanoid character and must include an animation controller and the appropriate animations (idle, walk, sprint etc).

While Foot IK is in Technical Preview, we suggest that you either backup your animation clips or duplicate your existing clips and modify the duplicates with “Curves” as explained below.

Foot IK is an advanced option and requires a bit of setup, including creating animation weight curves for animation clips. The high-level steps include:

1. In your animation controller, create 2 new float parameters (e.g., LeftFootIKWeight and RightFootIKWeight)
2. In your animation controller, ensure “IK Pass” is enabled on your main movement layer.
3. On the S3D controller, go to the Animate tab, with “Foot IK” enabled, set the Left and Right Foot Weight parameters. If the new parameters are not in the drop-down lists, click “Refresh” next to “Animation Actions”.
4. On your animation clips, expand “Curves” and add a new one. Set the name to the left foot parameter that was created in step 1 (they must match exactly). Change the curve (add keys as required) so that the value of the curve is 1 when the foot is on the ground, and 0 when the foot is off the ground in the animation.
5. Repeat step 3 for the right foot.

NOTE: These properties may change while in Technical Preview.

Property or Button	Description
Anim IK Pass Layer	The zero-based layer in the animator controller that has the IK Pass enabled for Foot IK
Body Move Speed Y	The rate at which the character's y-axis position is adjusted for foot IK placement.
Adjust Position Only	Only adjust the position of the feet. Do not modify the rotation.
Max Foot Inward Roll	The maximum rotation, in degrees, the foot can roll or rotate inward
Max Foot Outward Roll	The maximum rotation, in degrees, the foot can roll or rotate outward
Left Foot Weight param	The name of the float parameter in the animator controller used for left foot IK blending.
Right Foot Weight param	The name of the float parameter in the animator controller used for right foot IK blending.

Known Problems – When Foot IK is enabled, the character can sometimes hover above a step. We are currently investigating this issue.

Animate – Root Motion

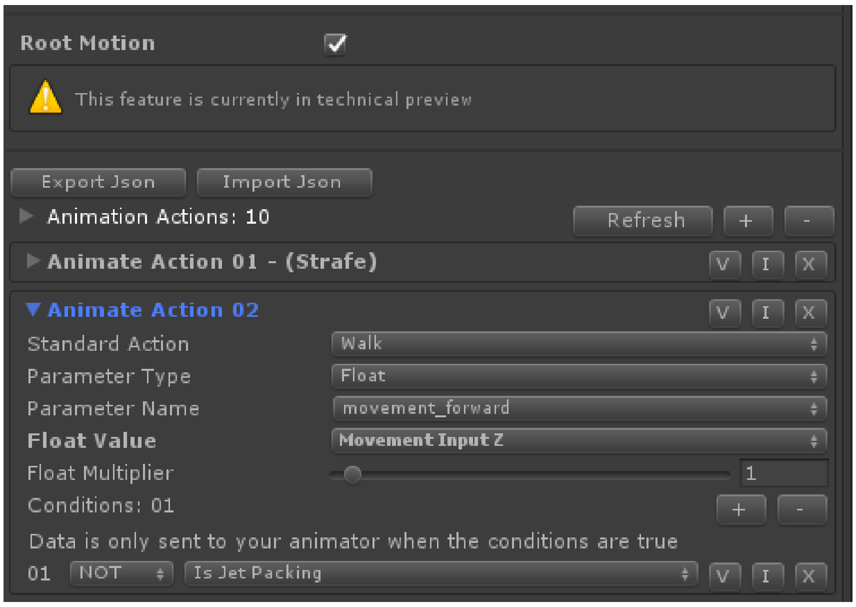
This feature is currently in **Technical Preview** and its behaviour may change without notice with each release to correct issues or to address particular scenarios. If you see a problem, please contact us on our Unity forum or on our Discord channel. See also the Common Issues – Animate (Root Motion) chapter below.

The Unity Animator component has an option to “Apply Root Motion” which allows the root bone transform to be controlled by the animation clip rather than by an external controller like Sticky3D. When enabled on the Unity Animator, Unity will attempt to calculate the position of the avatar using its physics engine.

In Sticky3D, if using Root Motion, we want to still have some control over position and rotation of the avatar as we use a combination of Unity physics and our own internal physics calculations.

Right now, we don’t consider our implementation of Root Motion to be “production-ready”.

Walk/Sprint/Strafe Speed on the Move tab will have no effect when “Root Motion” is enabled unless your Animator Controller uses these values to modify the speed of animations. That is because the amount and direction your character travels will mostly be driven by your animation clips rather than speed and acceleration on the Move tab.


Root Motion ✓

This feature is currently in technical preview

Export Json | Import Json

Animation Actions: 10 | Refresh | + | -

Animate Action 01 - (Strafe)

Animate Action 02

Standard Action: Walk

Parameter Type: Float

Parameter Name: movement_forward

Float Value: Movement Input Z

Float Multiplier: 1

Conditions: 01

Data is only sent to your animator when the conditions are true

01 NOT Is Jet Packing

Typically, you will want to pass the Movement Input float values to the Animator Controller, rather than the Moving Forward Back Speed like you might do without Root Motion.

Movement Input comes from either the user pressing a button or using a gamepad joystick via the Sticky Input Module, or via an NPC script instructing the character to move.

Property or Button	Description
Anim Drives Turning	Do animations drive turn left or right? Input can be sent to the animator controller via Anim Actions, and S3D reads the rotation from the avatar.
Idle Threshold	When root motion is enabled, velocity below this level will be considered idle. Some idle animations move the character around a little. Increasing this value will make IsIdle more stable. Default: 0.0001.

Known Problems – In this Technical Preview, the character can sometimes rebound from stairs or slopes when Root Motion is enabled. Currently, the only workaround is to disable Root Motion.

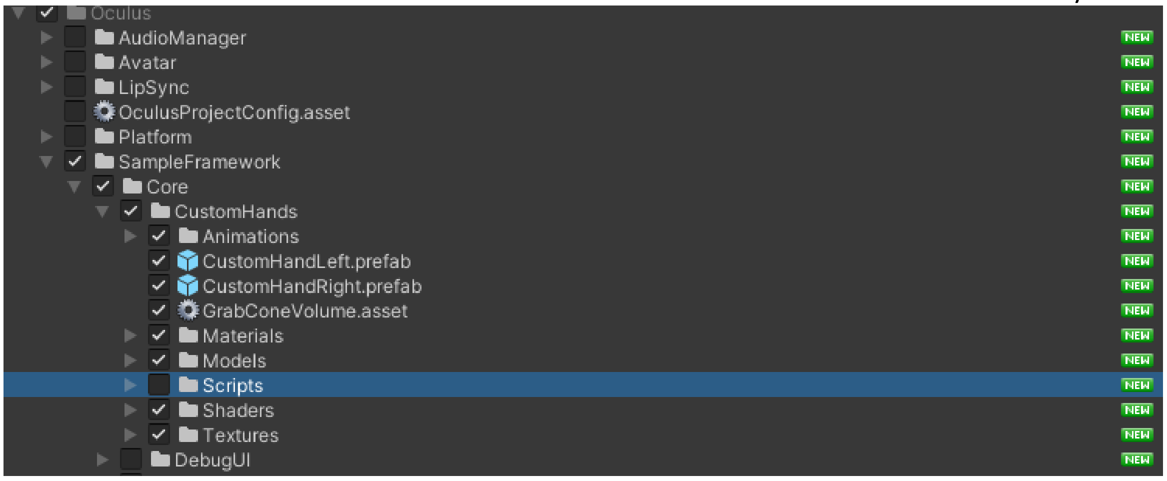
Animate – Hand VR

This feature is currently in **Technical Preview** and its behaviour may change without notice with each release to correct issues or to address particular scenarios. If you see a problem, please contact us on our Unity forum or on our Discord channel. See also the Common Issues – Animate (Hand VR) chapter below.

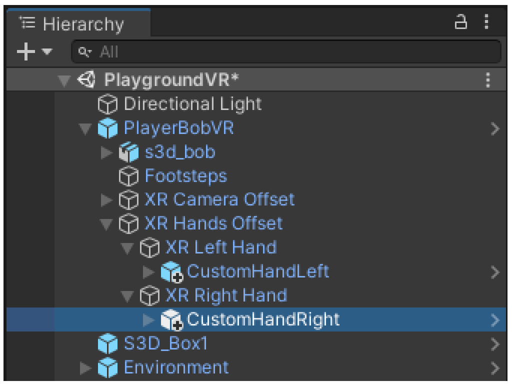
This feature allows you to control your own hand models and animator controller. You can also use third party hands. At this time, we don’t include any hand models in the asset.

Hand VR in S3D requires Unity 2020.3 LTS+ and Unity XR. To get started, first configure the Sticky Input Module for Unity XR. See the next chapter for details.

1. Import part of Oculus Integration package from Asset Store (select only SampleFramework, Core, CustomHands folder and untick "Scripts" subfolder). We tested with version 33.0 (30 Sept 2021).



2. In the Project pane, open the Oculus, SampleFramework, Core, CustomHands, CustomHandLeft prefab. Remove the two missing Script components. Save the prefab.
3. Repeat with CustomHandRight prefab.
4. In scene, under your S3D character, XR Hands Offset, XR Left Hand, add the CustomHandLeft prefab from Oculus to "XR Left Hand" gameobject.



5. On the Animate tab, tick "Hand VR"
6. From the scene, under CustomHandLeft, add “l_hand_skeletal_lowres” to the “Left Hand Animator” slot.
7. From the scene, under CustomHandRight, add “r_hand_skeltal_lowres” to the “Right Hand Animator” slot.
8. Set the Animator “Grip” parameters to “Flex” and the “Trigger” parameters to “Pinch”.

Debug

The Debug option can be used at runtime in the Unity Editor to help determine why a particular behaviour is being observed. Certain S3D values can be seen at runtime. These values can help you to determine why a something is or is not happening.

Sticky Input Module

The Sticky Input Module is a component that should be attached to the parent gameobject of your character along with the Sticky Control Module. It receives input from devices and sends this data to the Sticky Control Module.

Property	Description
Initialise on Awake	If enabled, Initialise () will be called as soon as Awake () runs. This should be disabled if you want to control when the Sticky Input Module is enabled through code.
Enable on Initialise	Is input enabled when the module is first initialised? See also EnableInput() and DisableInput() in Runtime And API – Sticky Input Module.
Input Mode	The system used to collect the input from the input device(s).

The module can receive input for the following items:

- Horizontal Move Input Axis (left and right movement)
- Vertical Move Input Axis (forward and backward movement)
- Sprint Button
- Jump & Jet Pack Up Button
- Crouch & Jet Pack Down Button
- Jet Pack Enable Button (enable or disable the Jet Pack feature)
- Horizontal Look Input Axis (look left or right)
- Vertical Look Input Axis (look up or down)
- Switch Look Button (switches between 1st and 3rd person view)
- Zoom Input Axis (zoom in or out when in third-person)
- Orbit Input Axis (orbit around the character when in third-person without Free Look)

You can also configure S3D to receive data from Custom Input (see below) and call your own runtime C# methods and/or API methods that are in-built.

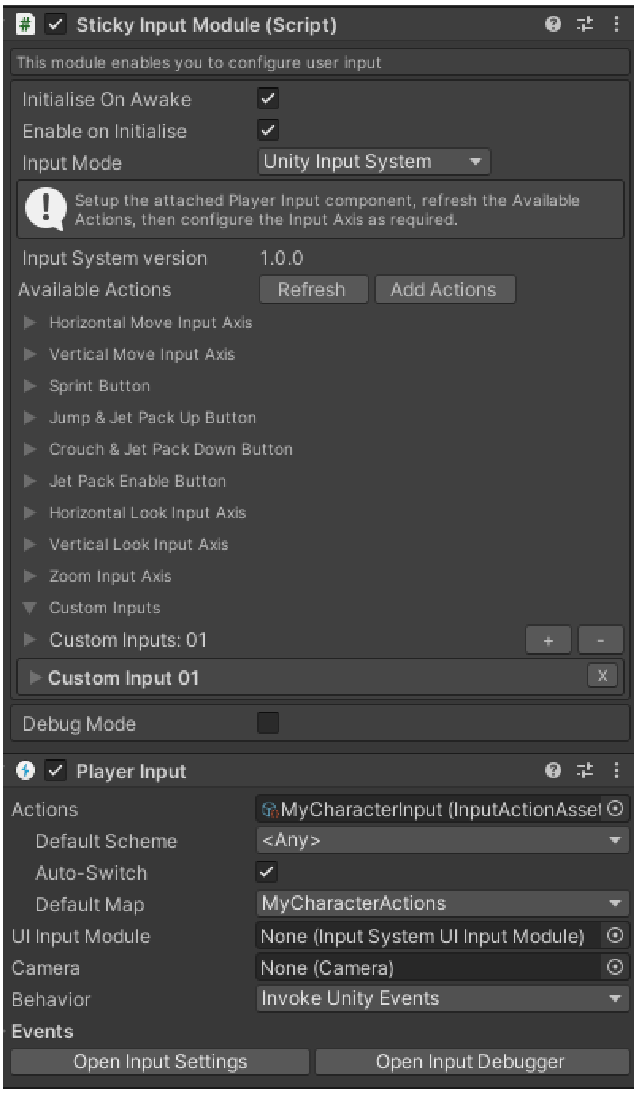
The Sticky Input Module is very flexible and can take input from multiple sources including:

- Direct Keyboard (and mouse)
- Legacy Unity (input system)
- Unity Input System
- Rewired

Unity Input System

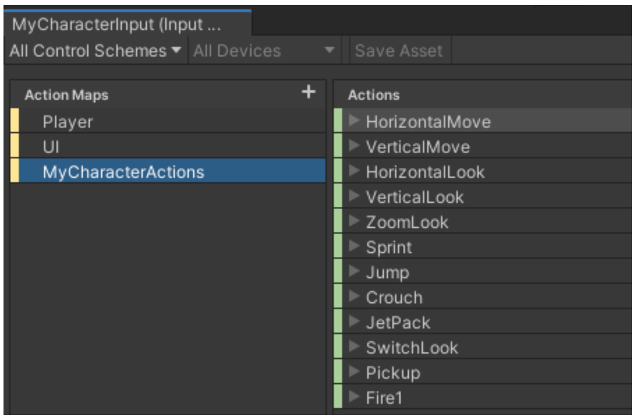
Sometimes referred to as the “New” Input System, v1.0.0 was first available in Unity 2019.1. Installed via the Unity Package Manager, this is an easy to setup, flexible system that supports many devices out of the box. To get started perform the following tasks:

1. Install Input System v1.0.0-preview.4 or newer with the Unity Package Manager
2. Restart the Unity Editor
3. Add the Sticky Input component to the S3D character.
4. On the Sticky Input component, click “Create Actions…”
5. Give it a name and save the asset when prompted (e.g. MyCharacterInput)
6. The asset editor should be displayed. If not, select the new actions asset (e.g., MyCharacterInput) and click “Edit asset”.
7. Add a new Action Map (e.g., MyCharacterActions)
8. Save the asset from the asset editor (and/or close it and save if prompted)
9. On the “Player Input” component attached to your character prefab, set the Default Map to the one you just added (e.g., MyCharacterActions) If the Actions says “None (Input Action Asset)” drag the asset (e.g., MyCharacterInput) into the slot first. Then set the Default Map.
10. On the Sticky Input Module (attached to the same character), change the Input Mode to “Unity Input System”
11. Resolve any configuration issues detected by the Sticky Input Module (this could include changing the default “Behaviour” on the Player Input component.
12. On the Sticky Input Module click “Add Actions” Now configure the various Input Axis in the Sticky Input Module like you would with any other Input Mode.


Sticky Input Module (Script)
This module enables you to configure user input
Initialise On Awake ✓
Enable on Initialise ✓
Input Mode: Unity Input System
Setup the attached Player Input component, refresh the Available Actions, then configure the Input Axis as required.
Input System version: 1.0.0

Actions: MyCharacterInput (InputActionAsset)
Default Scheme: <Any>
Auto-Switch ✓
Default Map: MyCharacterActions
UI Input Module: None (Input System UI Input Module)
Camera: None (Camera)
Behavior: Invoke Unity Events

Actions in the Sticky input Module, map to Actions stored in a Unity Input Action asset (e.g., MyCharacterInput) that are stored in the Project. The Player Input component, not to be confused with Sticky3D Controller’s Sticky Input Module, contains a link to the Unity Input Action asset. Double-clicking on this will open the Unity Input Action editor. This is part of the (new) Unity Input System and is not part of S3D.
The Default Map setting (e.g., MyCharacterActions) in the Unity Player Input component will determine which Actions are visible to the S3D Sticky Input Module. The Unity Input System can have multiple sets of Actions in what Unity calls “Maps”. The S3D Sticky Input System will only use one of those maps.



For more information on the Unity Input System see:

https://github.com/Unity-Technologies/InputSystem/blob/develop/Packages/com.unity.inputsystem/Documentation~/index.md

When using the “Add Actions” button, the mapping the following table is created. You are free to modify this as you wish. It is included as a quick setup option.

S3D Action	**Xbox One Controller**	**Sony Dual Shock 4**	**Keyboard & Mouse**
Horizontal Move	Left Stick X	Left Stick X	D, A
Vertical Move	Left Stick Y	Left Stick Y	W, S
Horizontal Look	Right Stick X	Right Stick X	Right, Left Arrow
Vertical Look	Right Stick Y	Right Stick Y	Up, Down Arrow
Jump	Right Shoulder	Right Shoulder	Space
Crouch	Left Shoulder	Left Shoulder	C
Sprint	A Button	Cross Button	SHIFT
Jet Pack	X Button	Square Button	J
Switch Look	Y Button	Triangle Button	V
Zoom Look	D-Pad Y	D-Pad Y	[and] or Mouse Scroll Wheel
Fire or Pickup	B Button	Circle Button	Left Mouse Button (Fire) or G (Pickup)

Rewired

This popular 3rd party package supports a vast array of input controllers. To use Rewired with S3D you need to have separately purchased it from the Unity Asset Store and imported it into your project.

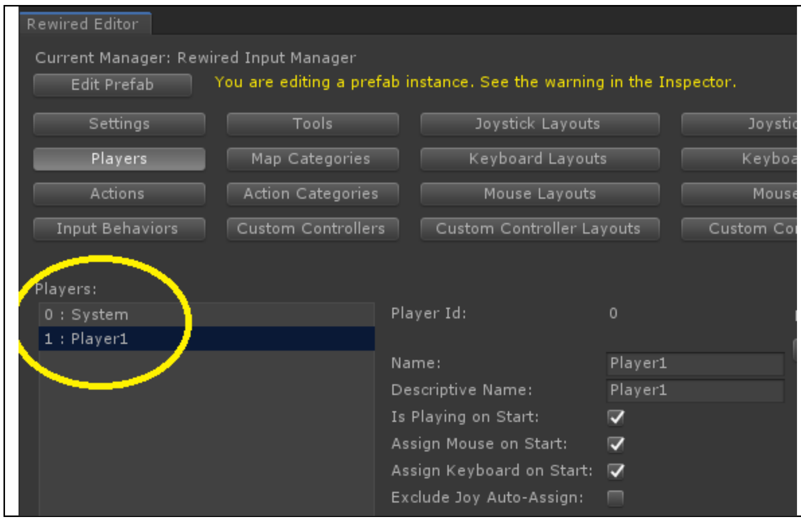
For more information on Rewired see:

https://assetstore.unity.com/packages/tools/utilities/rewired-21676

Before configuring the Sticky Input Module with Rewired, you first need to setup the Rewired Input Manager in the scene.

If you are not familiar with Rewired, here are the basic steps.

1. Add Rewired Input Manager to scene
2. Open Input Manager
3. Add a Player (and note the zero-based number in the list – e.g., 1, 2, 3 etc – System is typically 0)
4. Add Action Categories (e.g., S3D_Actions)
5. Add Actions to those Categories
6. Add Joystick Maps (for gamepad use [T] Gamepad Template - see below)
7. Add Keyboard Maps
8. Add Joystick Maps to a Player (ensure one is set to Start Enabled)
9. Add Keyboard Maps to a Player (ensure one is set to Start Enabled)
10. For the Player, ensure Assign Keyboard on Start is enabled
11. In the Sticky Input Module, set the Player Number to the player “number” from step 3 (0 mean unassigned).


Rewired Editor

Current Manager: Rewired Input Manager

Edit Prefab

You are editing a prefab instance. See the warning in the Inspector.

Players:
0 : System
1 : Player1

Player Id: 0
Name: Player1
Descriptive Name: Player1
Is Playing on Start: ✓
Assign Mouse on Start: ✓
Assign Keyboard on Start: ✓
Exclude Joy Auto-Assign:

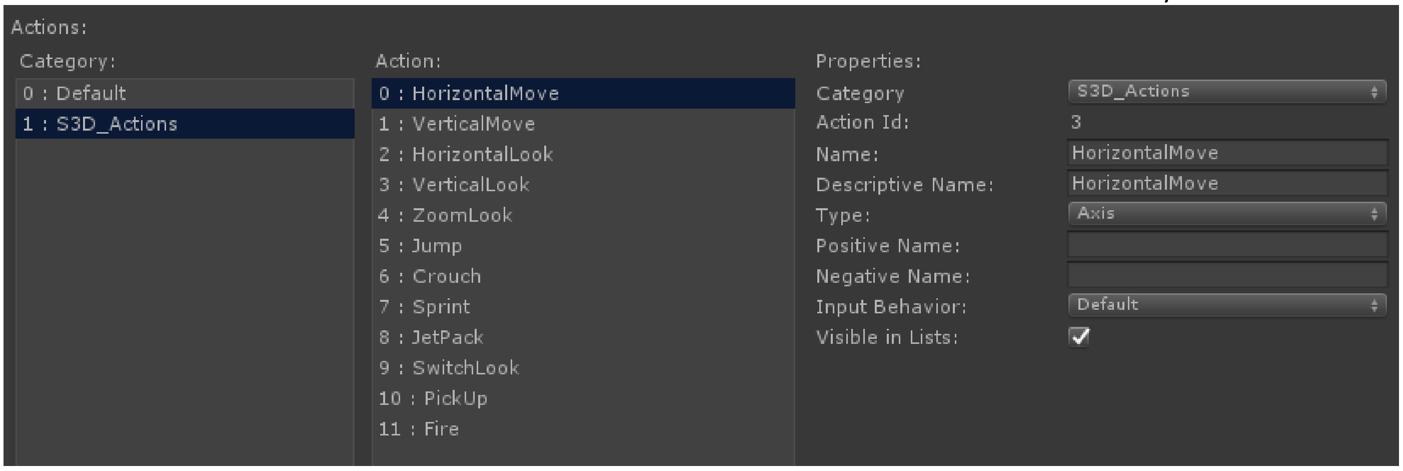
IMPORTANT: Don't forget to create a Player and assign the "Player Number" in the Sticky Input Module.

If you leave the Player Number as 0, your character will not be able to move.

For PC with some kind of gamepad (e.g., Xbox One, Sony Dual Shock 4), we'd suggest setting up a Gamepad Template like in the table below.

S3D Action	**[T] Gamepad Template**	**Xbox One Controller**	**Sony Dual Shock 4**	**Keyboard & Mouse**
Horizontal Move	Left Stick X	Left Stick X	Left Stick X	D, A
Vertical Move	Left Stick Y	Left Stick Y	Left Stick Y	W, S
Horizontal Look	Right Stick X	Right Stick X	Right Stick X	Right, Left Arrow
Vertical Look	Right Stick Y	Right Stick Y	Right Stick Y	Up, Down Arrow
Jump	Right Shoulder 1	Right Shoulder	Right Shoulder	Space
Crouch	Left Shoulder 1	Left Shoulder	Left Shoulder	C
Sprint	Action Bottom Row 1	A Button	Cross Button	SHIFT
Jet Pack	Action Top Row 1	X Button	Square Button	J
Switch Look	Action Top Row 2	Y Button	Triangle Button	V
Zoom Look	D-Pad Y	D-Pad Y	D-Pad Y	[and] or Mouse Scroll Wheel
Fire or Pickup	Action Bottom Row 2	B Button	Circle Button	Left Mouse Button (Fire) or G (Pickup)

In the below image we have added two more actions to our Rewired configuration (PickUp and Fire). We could use them with Custom Inputs to call our own code when the user presses one of these buttons.



Actions:

Category:
0 : Default
1 : S3D_Actions

Action:
0 : HorizontalMove
1 : VerticalMove
2 : HorizontalLook
3 : VerticalLook
4 : ZoomLook
5 : Jump
6 : Crouch
7 : Sprint
8 : JetPack
9 : SwitchLook
10 : PickUp
11 : Fire

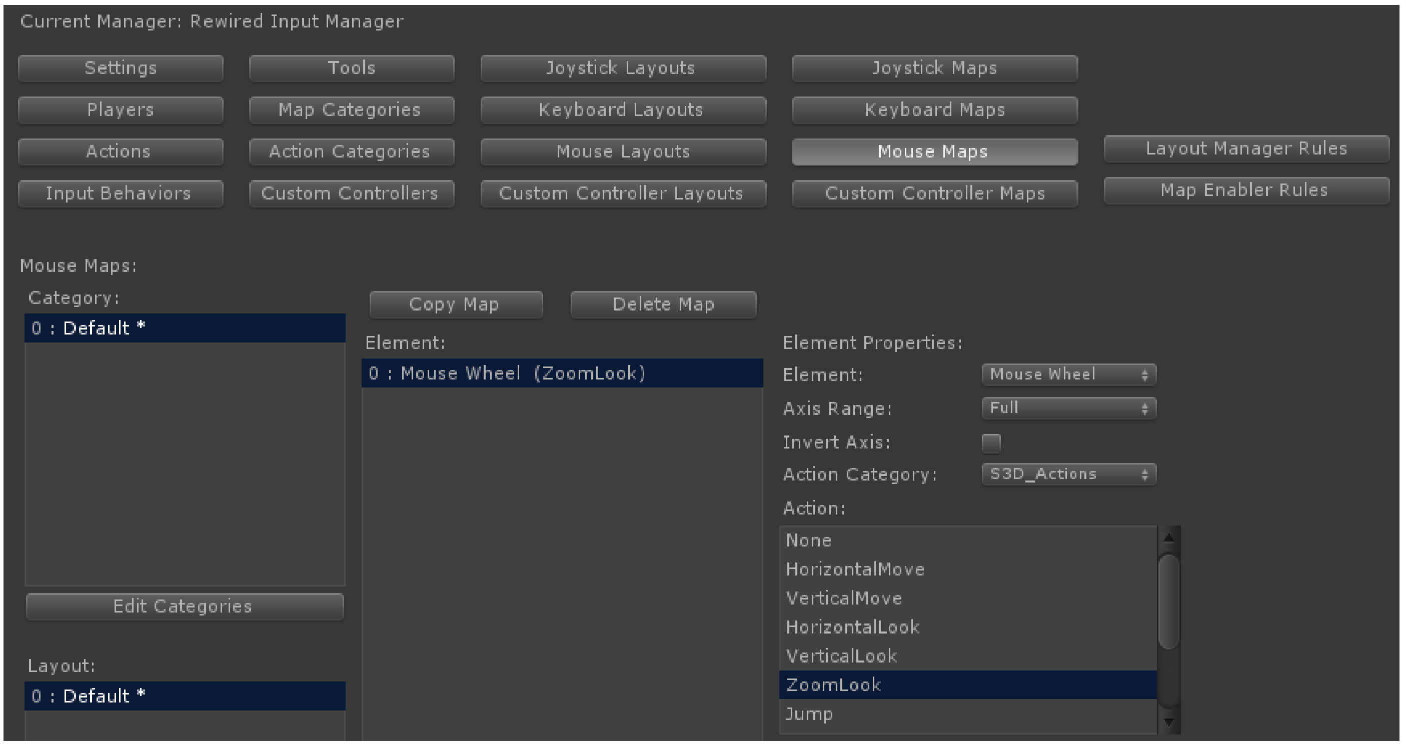
Properties:

Category	S3D_Actions
Action Id:	3
Name:	HorizontalMove
Descriptive Name:	HorizontalMove
Type:	Axis
Positive Name:	
Negative Name:	
Input Behavior:	Default
Visible in Lists:	✓

The Sticky Input Module works for both Axis and Button input. It has been tested with Keyboard Maps and Joystick Maps in Rewired. If you see any issues, please let us know.

To use the mouse scroll wheel for say Zoom Look, create a new Mouse Map in Rewired and assign the ZoomLook action that you created earlier.

Don't forget to assign the Mouse Map to the Player(s) in Rewired.



Current Manager: Rewired Input Manager

Mouse Maps:

Category:
0 : Default *

Layout:
0 : Default *

Element:
0 : Mouse Wheel (ZoomLook)

Element Properties:

Element:	Mouse Wheel
Axis Range:	Full
Invert Axis:	☐
Action Category:	S3D_Actions

Action:
None
HorizontalMove
VerticalMove
HorizontalLook
VerticalLook
ZoomLook
Jump

Unity XR

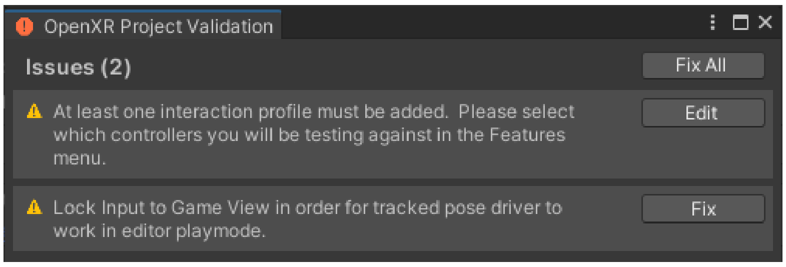
This feature is currently in **Technical Preview** and its behaviour may change without notice with each release to correct issues or to address particular scenarios. If you see a problem, please contact us on our Unity forum or on our Discord channel.

From Unity 2019 LTS, a new framework is available for XR - Virtual Reality (VR), Augmented Reality (AR) and Mixed Reality (MR). S3D has added support for this in Unity 2020.3 LTS or newer. This system uses a XR Plug-in Framework and is built on top of a Unity XR SDK.

The action-based system uses the "new" Unity Input System which is automatically installed during the basic setup steps provided below.

Here are the basic setup steps if using the Oculus Quest 2 headset:

1) In “Build Settings” ensure you have switched to the Android platform and “Texture Compression” is ASTC (the Oculus Quest 2 is an Android device).
2) Project Settings->XR Plugin Management
3) Install XR Plugin Management
4) Tick Open XR (Oculus, Windows Mixed Reality, Unity Mock HMD etc) – you may need to be on the “standalone tab”
5) Click "Yes" to enable the Input System when prompted (after a few moments the editor will restart)
6) If Open XR is not ticked on the Android tab, tick it now.
7) If there is a small yellow warning beside "OpenXR", click it.



OpenXR Project Validation

Issues (2) Fix All

At least one interaction profile must be added. Please select which controllers you will be testing against in the Features menu. Edit

Lock Input to Game View in order for tracked pose driver to work in editor playmode. Fix

8) Click "Fix" next to Lock Input to Game View (and arm64). Leave the other one for now.
9) Project Settings, XR Plug-in Management->OpenXR->Android tab
    a. Interaction Profiles, "+", Oculus Touch Controller Profile.
    b. Feature Groups, Oculus Quest Support
10) If you want to play the scene through the editor while connected with an Oculus Link cable (Oculus Rift mode), you will also need to add an Interaction Profile on the “PC, Mac & Linux Standalone” tab like you did for the Android tab above and optionally set the “Play Mode OpenXR Runtime” to Oculus.
11) On the player S3D character in the scene, locate the Sticky Input Module and change the Input Mode to “Unity XR”.
12) Using the Unity Input System, create an Input Action Asset scriptableobject in the Project pane. TIP: as a shortcut, install the XR Interactive Toolkit and make a copy of the “XRI Default Input Actions” scriptableobject which can be installed with XRI. If the XRI RightHand “Move” action doesn’t have a binding, add one – see he XRI LeftHand “Move” as an example. If you don’t want XRI in your project, you can install it in another Unity project and just import the single asset into your working project. You will also need to manually copy FallbackComposite.cs from the XR Interaction Toolkit, Runtime, Inputs, Composites package cache to your project (right-click on file and select Show in Explorer to copy the file).
13) Add the Input Action Asset to the Sticky Input Module in the slot provided.
14) In StickyControlModule, on the Look tab, turn off “Third Person”, and under “General Look Settings”, enable “Free Look”.
15) Enable “Look VR Mode”
16) If there is an existing first-person camera, remove it (also disable or remove the camera from the character in the scene).
17) Under “General Look Settings”, click “New” next to “Camera”
18) In StickyInputModule, click “New” next to Left and Right Hands.
19) Configure the Input Axis settings in the StickyInputModule editor. See the table below for suggestions.
20) Optionally, add your own hand meshes and scripts under XR Left Hand and XR Right Hand gameobjects. These should be child objects of the transforms. To set up animated hands, see “Animate – Hand VR” in the chapter on “Sticky Control Module”.
21) To deploy a build onto a Quest 2 (rather than running from the editor in Oculus Rift mode via a cable), you will also need to install the Oculus XR Plug, otherwise you will just see a black screen on the device.

Suggested setup using the modified XRI default actions.

S3D Action	Action Map	Action	Action Type	Data Slot	Binding Path
Left Hand Position	XRI LeftHand	Position	Value	N/A	<XRController>{LeftHand}/pointerPosition
Left Hand Rotation	XRI LeftHand	Rotation	Value	N/A	<XRController>{LeftHand}/pointerRotation
Right Hand Position	XRI RightHand	Position	Value	N/A	<XRController>{RightHand}/pointerPosition
Right Hand Rotation	XRI RightHand	Rotation	Value	N/A	<XRController>{RightHand}/pointerRotation
Horizontal Move	XRI LeftHand	Move	Value	Slot1	<XRController>{LeftHand}/Primary2DAxis
Vertical Move	XRI LeftHand	Move	Value	Slot2	<XRController>{LeftHand}/Primary2DAxis
Look Turn	XRI RightHand	Turn	Value	Slot1	<XRController>{RightHand}/Primary2DAxis
Look HMD	XRI HMD	Rotation	Value	Slot1	<XRHMD>/centerEyeRotation
Jump or Jet Pack Up	XR Buttons**	RHandA	Button	Slot1	<XRController>{RightHand}/secondaryButton
Crouch or Jet Pack Down	XR Buttons**	RHandB	Button	Slot1	<XRController>{LeftHand}/primaryButton
Sprint	XR Buttons**	LHandX	Button	Slot1	<XRController>{RightHand}/primaryButton
Jet Pack	XR Buttons**	LHandY	Button	Slot1	<XRController>{LeftHand}/secondaryButton
Switch Look					Currently only first person is supported for XR.
Zoom Look					Currently not supported in XR
Fire or Pickup	XRI RightHand	Activate	Button	Slot1	

** You will need to add your own Action Map called "XR Buttons".

If you experience jitter when looking left or right there are currently two different workarounds.

- In Project Settings->Time, set the Fixed Timestep to match your target framerate (0.0138889 = 72 fps, 0.011111 = 90 fps, 0.00833 = 120 fps). You may also need to disable VSync.
- If the above doesn't work, set the Fixed Timestep back to 0.02 and on the Sticky Control Module Move tab, General Move Settings, change Move Update Type to "Update".

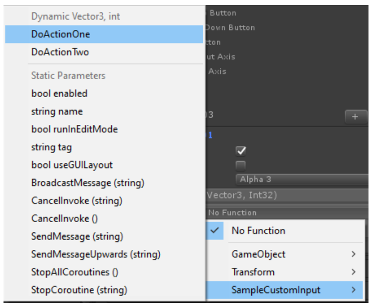
Custom Input

Each of the supported Input Modes (Direct Keyboard, Legacy Unity, Unity Input System, and Rewired), can take input from their various devices, and push that data to your own custom C# method. Example of how you might use this feature include:

- Call your own code when the player presses a button on a controller
- Call your own menu code
- Perform a custom action like change the camera position
- Get the value from say a controller trigger
- Modify a character variable or setting at runtime

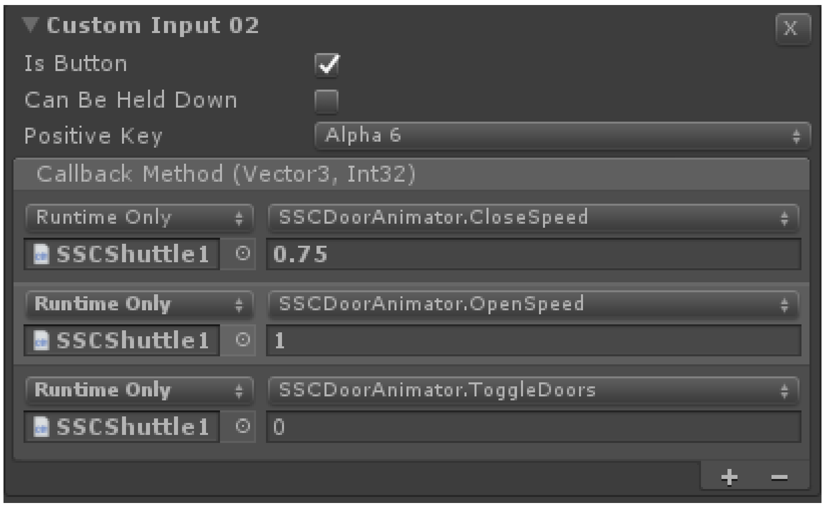
To use this in your own game code:

1. Create a new C# public method in your own game code (ensure it takes a Vector3 and int parameter if you want to get the input value and the input type) **
2. In Sticky Input Module Inspector, add a “Custom Input”
3. Select the button or controller input. If you are using Legacy Unity, Unity Input System, or Rewired you may need to configure this in the current input system first.
4. In Sticky Input Module, under the new Custom Input, add a Callback Method event.
5. Drag the gameobject from the scene that includes your game code script onto the empty Object field.
6. Configure the event function by selecting the method you created in item #1 above which should be under “Dynamic Vector3 int” in the popup menu.



For examples, see Demos\Scripts\SampleCustomInput.cs.

** You can also use “static parameters” or just call a method that takes no parameters. For example, you may wish to simply perform an action when the player presses a button on a controller or the keyboard. Or you may wish to always pass a certain parameter to a method when the user presses a button. Static Parameters only support a single parameter.
On the right, is an example of setting the door opening and closing speeds before toggling the door(s) open or closed.



Custom Input 02
Is Button ✓
Can Be Held Down
Positive Key: Alpha 6
Callback Method (Vector3, Int32)
Runtime Only | SSCDoorAnimator.CloseSpeed
SSCShuttle1 | 0.75
Runtime Only | SSCDoorAnimator.OpenSpeed
SSCShuttle1 | 1
Runtime Only | SSCDoorAnimator.ToggleDoors
SSCShuttle1 | 0

Custom Input and Animation

User input can drive animation within your Unity Animator Controller. For example, when a player wants to bend down and pick up an object by pressing a button on their device or gamepad, you may wish to set particular parameters in the Animator Controller.

For more information, see Sticky Control Module – Animate – Custom Input and Actions.

Sticky Display Module

This feature is currently in **Technical Preview** and its behaviour may change without notice with each release to correct issues or to address particular scenarios. If you see a problem, please contact us on our Unity forum or on our Discord channel.

The Sticky Display Module can be used to display information to the player.

This feature is currently **NOT SUPPORTED with VR** because Unity does not support screen space overlay in VR.

A prefab is included in the Demos\Prefabs\visuals folder that can be dropped into a scene. If you plan to modify the prefab, we suggest creating a copy or “original” prefab from this one to work with. Otherwise, when you next update Sticky3D Controller you may overwrite your changes.

When making changes to a HUD prefab, ensure you edit in the scene OR “Open Prefab” when using 2019.1 or newer. If you attempt to make changes when the prefab is not open or in the scene you will receive many errors and warnings in the console which may leave the HUD prefab in an inconsistent state.

The Sticky Display Module is a UnityEngine.UI or canvas-based solution. Currently it does not use any custom shaders and therefore should work wherever the UI canvas and components are supported. If you see any platform-centric issues, please let us know.

The module currently has the following feature areas:

- Auto-hide cursor
- Reticle selection
- Display Messages
- Display Targets
- API for integration with your own project

Our plan is to add new features over time based on typical user requirements. Where possible, we endeavour to add core features that have widespread appeal. We believe stable low-level features are more important than visual or graphical elements. That’s because we know you will have very particular look and feel requirement which will set you game or project apart from others. Essentially, we want to empower you rather than restrict you.

Don’t feel compelled to use this module. If you want to design a totally different display you can still do so and use our extensive API to pull data from the characters etc.

WARNING: Do not manually change the size, anchor points or settings of the display elements in the scene. This could lead to unpredictable results.

Sticky Display Module – Extending

As the module is constructed on top of the standard Unity UI and has its own built-in API, you should be able to extend its functionality without too much fuss.

We’d recommend the following approach:

1. Create a custom monobehaviour script in your own namespace
2. Attach this component to the same gameobject as the StickyDisplayModule component
3. In your script get a reference to both the StickyDisplayModule and the Canvas.
4. Call any StickyDisplayModule API methods as required
5. Add UI elements giving them unique names on the same canvas.
6. Update your UI elements as required

Sticky Display Module – General Settings

Many of these properties can be adjusted in real-time in the editor at runtime to see the effect they have.

Property	Description
Initialise on Start	If enabled, the Initialise () will be called as soon as Start () runs. This should be disabled if you are instantiating the display through code.
Show on Initialise	Show the display when it is first Initialised
Show Overlay	Show the overlay image on the display.
Auto Hide Cursor	Automatically hide the screen cursor or mouse pointer after it has been stationary for a fixed period of time. Automatically show the cursor if the mouse if moved provided that the Display Reticle is on shown.
Hide Cursor Time	The number of seconds to wait until after the cursor has not moved before hiding it
Main Camera	The main camera used to perform calculations with the heads-up display. If blank will be auto-assigned to the first camera with a MainCamera tag.
Display Width	The head-up display's normalised width of the screen. 1.0 is full width, 0.5 is half width. To see the effect of this outside play mode, enable Show Display Outline and look in the scene view.
Display Height	The head-up display's normalised height of the screen. 1.0 is full height, 0.5 is half height. To see the effect of this outside play mode, enable Show Display Outline and look in the scene view.
Display Offset X	The head-up display's normalised offset between the left (-1) and the right (1) from the centre (0) of the screen. To see the effect of this outside play mode, enable Show Display Outline and look in the scene view.
Display Offset Y	The head-up display's normalised offset between the bottom (-1) and the top (1) from the centre (0) of the screen. To see the effect of this outside play mode, enable Show Display Outline and look in the scene view.
Primary Colour	Primary colour of the heads-up display. This changes the colour of the overlay image. This are affected by Brightness. Calibrate when Brightness = 1.
Brightness	This is the overall brightness of the display relative to its initial state at runtime.
Canvas Sort Order	The sort order of the canvas in the scene. Higher numbers are on top.
Show Display Outline	Show the display as a yellow outline in the scene view [Has no effect in play mode]. Click Refresh if screen has been resized. This can help to gauge the overall size of the display without going into play mode.

Sticky Display Module – Display Reticle Settings

Many of these properties can be adjusted in real-time in the editor at runtime.

Property	Description
Show Active Reticle	Show or render the active Display Reticle on the display. [Has no effect outside play mode]
Active Display Reticle	The currently selected or displayed reticle. This is used to help aim at things in front of your character.
Reticle Offset X	The Display Reticle's normalised offset between the left (-1) and the right (1) from the centre (0) of the screen.
Reticle Offset Y	The Display Reticle's normalised offset between the bottom (-1) and the top (1) from the centre (0) of the screen.
Reticle Colour	The colour of the active Display Reticle
Lock Reticle to Cursor	Should the Display Reticle follow the cursor or mouse position on the screen?
Sprite (for each Reticle)	The sprite (texture) to be displayed in the display for this reticle. Samples of these are provided in the Textures\Display folder. They should be 64x64, white with a transparent background, and have a Texture Type of Sprite (2D and UI).

To create a custom reticle you can perform the following tasks:

1. In an image editor like Photoshop, Corel Paintshop Pro or Gimp, create a new image with a transparent background that has dimensions of 64x64 pixels.
2. Draw your reticle in white (RGBA 1,1,1,1) – you can colour the reticle inside the Sticky Display Module.
3. Import the image into Unity (we have used PNG but the format should not matter)
4. Change the Texture Type to "Sprite (2D and UI)"
5. Use this sprite in your Display Reticle.

Sticky Display Module – Display Message Settings

Display Messages are used to present information to the player. They are displayed and can also be created, shown, hidden, moved or scrolled at runtime via our extensive API. See "Runtime and API" for more details.

At runtime, messages are stacking on the UI canvas in the order they appear in the Sticky Display Module Inspector list. The first item is on placed on the canvas first, and then the second message, and so forth. You can re-order the list by using the small "V" move button.

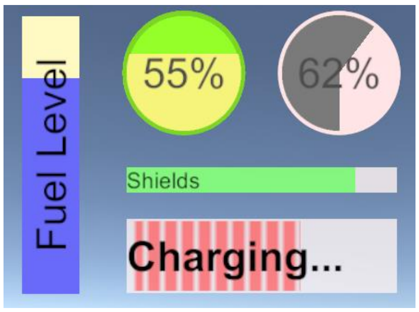
Many of these properties can be adjusted in real-time in the editor at runtime.

Property	Description
Show Message	Show the message on the display. When a message is created, this is off by design. [Has no effect outside play mode]
Message Name	The name or description of the message. This can be used to identify the message.
Message Text	The text to display in the message. It can include RichText markup. e.g., <b>Bold Text</b>
Offset X	The Display Message's normalised offset between the left (-1) and the right (1) from the centre (0) of the screen.
Offset Y	The Display Message's normalised offset between the bottom (-1) and the top (1) from the centre (0) of the screen.
Display Width	The Display Message's normalised width. 1.0 is full screen width, 0.5 is half width.
Display Height	The Display Message's normalised height. 1.0 is full screen height, 0.5 is half height.
Show Background	Show the Display Message background
Text Colour	Colour of the Message text
Text Alignment	The position of the text within the Display Message panel
Is Best Fit	Is the text font size automatically changes within the bounds of Font Min Size and Font Max Size to fill the panel?
Font Min Size	When Is Best Fit is true will use this minimum font size if required
Font Max Size	The size of the font. If isBestFit is true, this will be the maximum font size it can use.
Scroll Direction	The direction (if any) the text should scroll across the screen.
Scroll Speed	Speed or rate at which the text will scroll across the display.
Is Scroll Fullscreen	Scroll full screen regardless of message width and height.

Sticky Display Module – Display Gauge Settings

These simple measuring bars and gauges can be used to let players know the status of things in your game. Some examples include:

- The health of the character
- Fuel level in the jet pack
- The amount of charge in a weapon
- The distance to your destination
- Your game score
- Number of times the player can respawn
- The percentage of enemies left to destroy



Gauges are typically constructed by using a different foreground and background sprite (UI texture). There are several examples included in the Textures\Display folder to get you started. Examples include:

- SSCUICircle1BGnd
- SSCUICircle1FGnd
- SSCUICircle2BGnd
- SSCUIFilled
- SSCUIStripeH1Border (has a transparent border)
- SSCUIStripeH1NoBorder
- SSCUIStripeH2Border
- SSCUIStripeH2NoBorder
- SSCUIStripeV1Border
- SSCUIStripeV1NoBorder
- SSCUIStripeV2Border
- SSCUIStripeV2NoBorder

To build custom background and/or foreground sprites, follow the following basic steps:

1. In an image editor like Photoshop, Corel Paintshop Pro or Gimp, create a new image with a transparent background that has dimensions of 64x64 or 256x256 pixels.
2. Draw your sprite details in white (RGBA 1,1,1,1) – you can colour the sprite inside the Ship Display Module. For darker areas use a grey scale. Don't forget to test how they look when the colours are changed in the display module AND the brightness is modified in the General Settings. This is the primary reason why you draw the sprite using white.
3. Import the image into Unity (we have used PNG but the format should not matter). Typically, you will want to place the new sprite in your own folder within the project (not within the Sticky3DController folder).
4. Change the Texture Type to "Sprite (2D and UI)"

There are many API methods that can help you manage and update gauges at runtime.

Many of these properties can be adjusted in real-time in the editor at runtime.

Property	Description
Show Gauge	Show the gauge on the display. When a gauge is created, this is off by design. [Has no effect outside play mode].
Gauge Name	The name or description of the gauge. This can be used to identify the gauge.
Gauge Value	The current amount or reading on the gauge. Value must be between 0.0 (empty/min) and 1.0 (full/max).
Offset X	The Display Gauge's normalised offset between the left (-1) and the right (1) from the centre (0) of the screen.
Offset Y	The Display Gauge's normalised offset between the bottom (-1) and the top (1) from the centre (0) of the screen.
Display Width	The Display Gauge's normalised width. 1.0 is full screen width, 0.5 is half width.
Display Height	The Display Gauge's normalised height. 1.0 is full screen height, 0.5 is half height.
Value Affects Colour	Does the colour of the foreground change, based on the value of the gauge?
Foreground Colour	Colour of the gauge foreground when the value does not affect the colour.

Property	Description
Foreground Low Colour	Colour of the Gauge foreground when value is 0.0
Foreground Medium Colour	Colour of the Gauge foreground when value is 0.5
Foreground High Colour	Colour of the Gauge foreground when value is 1.0
Foreground Sprite	The sprite (texture) for the foreground of the gauge
Background Colour	Colour of the gauge background.
Background Sprite	The sprite (texture) for the background of the gauge
Fill Method	Determines the method used to fill the gauge foreground sprite when the gaugeValue is modified.
Keep Aspect Ratio	Keep the original aspect ratio of the foreground and background sprites. Useful when creating circular gauges.
Text Colour	Colour of the Gauge text
Text Alignment	The position of the text within the Display Gauge panel
Text Direction	The direction of the text within the Display Gauge panel
Text Style	The style of the text within the Display Gauge panel
Is Best Fit	Is the text font size automatically changes within the bounds of Font Min Size and Font Max Size to fill the panel?
Font Min Size	When Is Best Fit is true will use this minimum font size if required
Font Max Size	The size of the font. If isBestFit is true, this will be the maximum font size it can use.

Sticky Display Module – Display Target Settings

These are used to show potential friendly or enemy targets to the player. They can also be used with Sticky XR Interactor to show interactive-enabled objects being pointed at with hands in VR.

Like the "Active Display Reticle", they use the Reticles from the list that is available in the Display Reticle Settings section of the editor.

Display Targets can be created, shown, hidden or moved at runtime via our extensive API. See "Runtime and API" for more details.

Many of these properties can be adjusted in real-time in the editor at runtime.

Property	Description
Show Viewport Outline	Show the rendering limits as a red outline in the scene view [Has no effect in play mode]. Click Refresh if screen has been resized.
Viewport Width	The width of the clipped area in which Targets are visible. 1.0 is full width, 0.5 is half width.
Viewport Height	The height of the clipped area in which Targets are visible. 1.0 is full height, 0.5 is half height.
Viewport Offset X	The X offset from centre of the screen for the viewport
Viewport Offset Y	The Y offset from centre of the screen for the viewport
Targeting Range	The maximum distance in metres that targets can be away from the ship
Per Target Settings	
Show Target	Show the target reticle on the HUD. When a target is created, this is off by design. [Has no effect outside play mode]
Display Reticle	The Display Reticle to use for this Target. To add more Reticles, see the list in the Display Reticle Settings section.
Reticle Sprite	This is what the target will look like on the HUD before colour and brightness is applied.
Reticle Colour	The colour of the active Display Reticle for this Target
Max. Number of Targets	The maximum number of these DisplayTargets that can be shown on the HUD at any one time.

Debug Mode is used at runtime to show useful information that can help with troubleshooting.

Proximity Component

This component let you call your game code, many S3D API methods, and/or set properties on gameobjects when an object with a collider enters or exits an area of your scene. Essentially, it saves you time from having to write collider trigger code. There is also an option to check the object's Unity tag.

If no tags are provided, all objects can affect this area. NOTE: All tags MUST exist.

A typical use case is when a character enters an area and you want to perform some kind of custom task or make something particular happen in your game. It could be something as simple as turning on/off a light as the character enters or exits a room.

To add one to the scene, use the 3D Object -> Sticky3D Controller menu to create a new gameobject with either a sphere or box trigger collider.

Sticky Foot

This is a small optional component added to the feet of a character to detect collisions with the ground. It is typically used when you want more accurate footstep sounds and effects.

In your characters prefab, you should first attach a small sphere, or box collider onto the transform of the left and right foot bones. Then, on to the same transforms, attach a StickyFoot component. Now, in the StickyControlModule, on the Move tab, expand the Footsteps section, make sure "Use Move Speed" is not enabled, and add the Left and Right Foot transforms.

Sticky Interactive

This component can be used to make objects in your scene interactive-enabled. It will be possible to make objects in your scene Activable, Readable, Selectable, Touchable, and/or Grabbable. For example, you might have a panel or button that your character needs to press to cause some action like open a door, turn on a light, or reveal some hidden key.

You can also get the objects to call your own game code and/or S3D APIs when certain things happen. For example, when a character starts looking at or stops looking at an interactive-enabled object. Or maybe when one (or more) are selected in the scene, or when an object is picked up (grabbed) or dropped by a character.

To be seen, interactive-enabled objects should have a non-trigger collider or rigidbody on the same gameobject as the StickyInteractive component. When there are multiple colliders on the same gameobject, Unity creates compound colliders at runtime which may produce behaviour you are not expecting. If you need to place the colliders on child objects, but still want them to be discoverable by the characters, add a StickyInteractiveChild component to the child gameobject(s) containing the colliders, and drag in the parent StickyInteractive component.

Any player or non-player character (NPC) can activate or deactivate the Activable objects in the scene.

Each player or NPC can contain its own list of Selectable objects in the scene. The number selectable at one time are set on the Engage tab of each S3D character. The same object can be selected by one character, while not be selected by another character.

Sticky Interactive can be used with either Hand IK to animate hands, or Sticky XR Interactor in VR games.

To help with testing, we have included a StickyInteractiveTester component that can be added to an empty gameobject in the scene. You can hook this up to any event on an interactive-enabled object to check when those events are fired during game play in the editor at runtime. After testing, you can hook up your own methods in your code and delete the tester from the scene.

Sticky Interactive - Properties

Property or Button	Description
Initialise on Start	If enabled, Initialise() will be called as soon as Start() runs. This should be disabled if you want to control when the Sticky Interactive component is enabled through code.
Is Activable	Can this item be activated? Typically used with grabble objects. If an object is also grabbable, it must be grabbed in order to be activated. Non-grabbable objects can be useful when machinery needs to be activated or deactivated without being picked up.
Auto Deactivate	When the item is activated, it is automatically deactivated making it a single on and off action. Only the OnActivated events are triggered.
Deactivate on Drop	When a grabbable and activable object is dropped while activated, it will be deactivated. Both Deactivate and OnDropped events are triggered.
Is Grabbable	Can this item be grabbed or held?
Carry in Hand	When grabbed, the interactive-enabled object will be held in the palm of the hand. Turn this off for objects like levers with a non-kinematic (dynamic) rigidbody which you do not want the character to carry. Currently, should always be on unless using VR and StickyXRInteractors.
Parented on Grab	When grabbed, the interactive-enabled object will be parented to the hand of S3D character. NOTE: Parenting infers the object will be carried in the palm of the hand.
Re-parent on Drop	Attempt to reparent the interactive-enabled object to the original parent Gameobject when it was grabbed.
Disable Regular Colliders	When grabbed, non-trigger colliders on this interactive-enabled object will be disabled. When dropped, they will be enabled.
Disable Trigger Colliders	When grabbed, trigger colliders on this interactive-enabled object will be disabled. When dropped, they will be enabled.
Remove Rigidbody on Grab	When grabbed, if there is a rigidbody attached, remove it.
Is Readable	Can values be read from this object? This is typically used for reading the position of a virtual lever or joystick.
Readable Joint	The joint to read positional data from when Is Readable is true. For a lever, this would be a hinge joint.
Dead Zone	The normalised amount the joint can move before values are updated.
Invert X	Invert the x-axis value (left and right) from the readable joint.
Invert Z	Invert the z-axis value (forward and backward) from the readable joint.
Is Selectable	Can this item be selected in the scene?
Is Auto Unselect	When the item is selected, it is automatically unselected making it like a button click event. Only the OnSelected events are triggered. This can be used to make the item act like a clickable button. For a toggle on/off button, leave this disabled.
Is Sittable	Is this item suitable for sitting on?
Is Touchable	Can this item be touched by a character? Typically used with Hand IK or Sticky XR Interactor.
Mass	Mass of the object in kilograms. Can be used when dropping the object.
Hand Hold Relative Positions	The gizmos show where the hand would be placed on the object. The arrow points in palm direction. The sphere is in the thumb up direction. Fingers show where they would face.
(LH) Left Hand	The gizmos will show the left-hand position.
(F)ind	Select (find) the hand hold position in the scene view
(G)izmo	Toggle the gizmo for this hand hold position in the scene view
Hold 1 Offset	The first or primary local space hand hold offset – you can modify the value in the Inspector or move the gizmo in the scene view.

Property or Button	Description
Hold 1 Rotation	The first or primary local space hand hold rotation stored in degrees – you can modify the value in the Inspector or rotate the gizmo in the scene view. Arrow gizmo points in direction hand palm is facing. Sphere points towards thumb direction.
Hold 1 Flip for LH	For the first or primary hand hold, flip the rotation for left hand when using Sticky XR Interactor. This can be useful when the hand hold position is on a symmetrical handle like a bat, racket, or spear.
Hold 2 Offset	The second or secondary local space hand hold offset – you can modify the value in the Inspector or move the gizmo in the scene view.
Hold 2 Rotation	The second or secondary local space hand hold rotation stored in degrees – you can modify the value in the Inspector or rotate the gizmo in the scene view. Arrow gizmo points in direction hand palm is facing. Sphere points towards thumb direction.
Hold 2 Flip for LH	For the second or secondary hand hold, flip the rotation for left hand when using Sticky XR Interactor. This can be useful when the hand hold position is on a symmetrical handle like a bat, racket, or spear.
Popup Offset	The default local space offset a StickyPopupModule appears relative to the interactive-enabled object.

Sticky Interactive - Events

Interactive events are triggered when certain things happen. When these happen, you can call your own game code, call S3D APIs, and/or set some properties on other Unity objects in the scene. By using APIs and Unity object properties, you may not even need to write any code yourself.

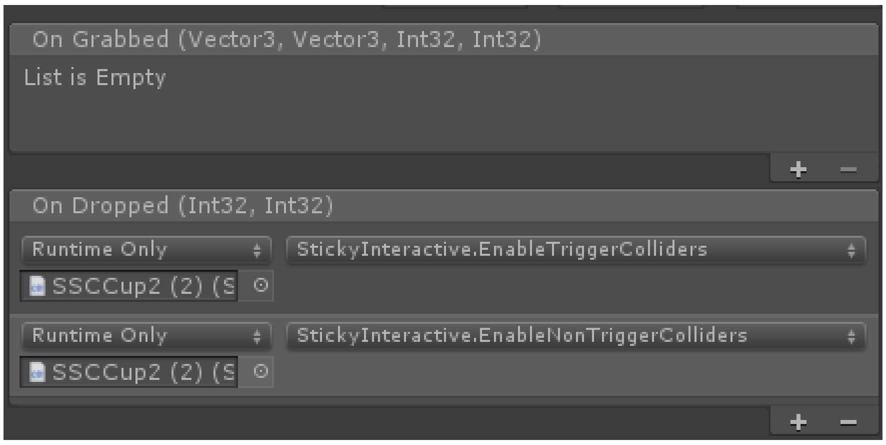
Event Property	Description
On Activated	These are triggered by a S3D character when they activate this interactive object.
On Deactivated	These are triggered by a S3D character when they deactivate this interactive object.
On Grabbed	These are triggered by a S3D character when they grab this interactive object.
On Dropped	These are triggered by a S3D character when they drop this interactive object.
On Hover Enter	These are triggered by a S3D character when they start looking at this interactive object.
On Hover Exit	These are triggered by a S3D character when they stop looking at this interactive object.
On Readable Value Changed	These are triggered when the value of the joint position changes if the interactive object is readable.
On Selected	These are triggered when this interactive object is selected via the API.
On Unselected	These are triggered when this interactive object is unselected via the API.
On Touched	These are triggered by a S3D character when the interactive object is first touched via the API. Touch events only occur when a character is intentionally attempting to touch (target) an object using Hand IK that has “Is Touchable” enabled. They do not occur via casual contact.
On Stopped Touching	These are triggered by a S3D character when the interactive object is no longer being touched via the API.

Sticky Interactive – Dropping Objects

To drop an interactive-enabled object from the hand of a S3D character, it must be Grabbable. If it has colliders attached to it, and they were disabled when the object was grabbed (via setting the Disable Regular/Trigger Colliders settings, the colliders are NOT automatically re-enabled. This is to prevent the scenario where they appear inside the collider(s) of the character.

If it is safe to do so, you could re-enable the colliders by adding the appropriate API calls to the OnDropped event. E.g.

1. Click + below OnDropped to add an event
2. Drag the Sticky Interactive gameobject into where it says “Runtime Only”
3. Set the Function to “StickyInteractive.EnableTriggerColliders” or “StickyInteractive.EnableNonTrigggerColliders”

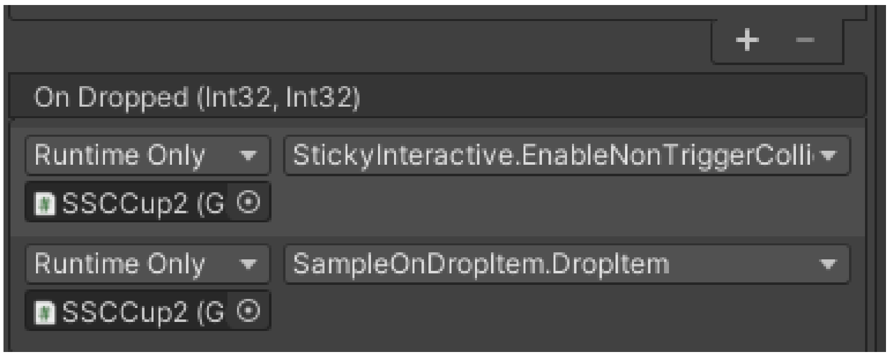

On Grabbed (Vector3, Vector3, Int32, Int32)
List is Empty

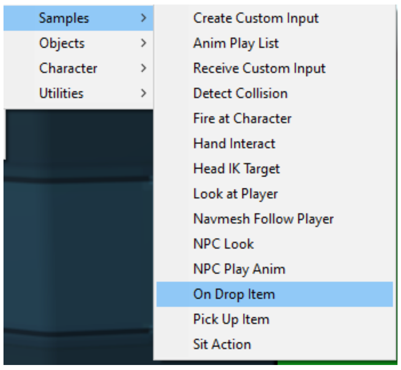
On Dropped (Int32, Int32)
Runtime Only | StickyInteractive.EnableTriggerColliders
SSCCup2 (2) (S
Runtime Only | StickyInteractive.EnableNonTriggerColliders
SSCCup2 (2) (S

We have included a sample drop component (On Drop Item) that can be attached to a Grabbable interactive-enabled object. This can easily be configured in the On Dropped events.

Alternatively, it can be configured without disabling the Rigidbody.

1. Add a non-Kinematic Rigidbody to the object and enable “Use Gravity”
2. Set the Rigidbody “Interpolate” to “Interpolate”. Set the “Collision Detection” to “Continuous Dynamic”
3. On the StickyInteractive component, untick “Remove Rigidbody on Grab”
4. Untick “Disable Regular Colliders”


On Dropped (Int32, Int32)
Runtime Only | StickyInteractive.EnableNonTriggerColli
SSCCup2 (G
Runtime Only | SampleOnDropItem.DropItem
SSCCup2 (G



Alternatively, you might want to write some game code to do things like:

- Move the object below the character’s hand or out of its grasp
- Call the enable collider APIs as mentioned above
- Add a rigidbody with gravity so the object falls to the ground
- Call this custom game code method from the On Dropped event. Don’t forget to write this custom method in your own namespace and not any of the Sticky scripts.

Sticky Interactive with VR

To use Sticky Interactive objects when the character is using Unity XR, add Sticky XR Interactor components to the character hand transforms. See the chapters on “Sticky XR Interactor” and “Sticky Input Module” (Unity XR section) for more information. For API methods, see “Sticky XR Interactor Methods” in the “Runtime and API” chapter.

Sticky Interactor Bridge

This feature is currently in **Technical Preview** and its behaviour may change without notice with each release to correct issues or to address particular scenarios. If you see a problem, please contact us on our Unity forum or on our Discord channel.

This component enables a non-Sticky3D character object or hand to interact with Sticky Interactive-enabled objects.

This could be helpful if you had another asset, like say Sci-Fi Ship Controller (SSC), and you wanted the SSC XR hands in a VR game to activate, touch, or grab interactive-enabled objects without using the Sticky3D (character) Controller. This might be useful when a Sticky3D character gets into the cockpit of a ship and you switch control over to SSC.

This component should be added to the hand gameobject of a non-Sticky3D character.

For related components, see the chapters called “Sticky Interactive”, and “Sticky XR Interactor”.

General Property	Description
Initialise on Start	If enabled, Initialise() will be called as soon as Start() runs. This should be disabled if you want to control when the Sticky Interactor Bridge component is enabled through code.

Interactive Property	Description
Can Activate	Can this item activate an interactive-enabled Activable object?
Can Grab	Can this item grab an interactive-enabled Grabbable object?
Auto Grab	The component will attempt to automatically grab an interactive-enabled Grabbable object when it is in range.
Palm Transform	The z-axis (forward) should point in the direction of the palm is facing. This is the palm “normal”. The x-axis (right) is towards the direction of the index finger and, for the right hand, the y-axis (up) is in a thumbs-up direction.
Can Touch	Can this item touch an interactive-enabled Touchable object?

This component also has overridable API methods. See the Runtime and API chapter for more details.

Sticky Manager

This feature is currently in **Technical Preview** and its behaviour may change without notice with each release to correct issues or to address particular scenarios. If you see a problem, please contact us on our Unity forum or on our Discord channel.

The manager is used to centrally manage multiple objects in the scene. It includes a pooling system which can make your game more performant.

The Sticky Manager can be added to the scene via the editor (GameObject->3D Object->Sticky3D Controller->Sticky Manager) or via code (see “Sticky Manager Methods” in the “Runtime and API” chapter).

Property	Description
Startup Generic Modules	The list of modules added to the pool at start-up. Modules can also be added via code. See Demos\SampleGenericTextModule.cs for an example.

Currently there are two components that can be used to make your objects poolable.

StickyGenericModule

This component is used with the Sticky Manager to make an object in your scene poolable. Instead of instantiating and destroying an object multiple times, the object is pooled and enabled or disable as required.

The component can be added to a prefab to make it poolable. You can extend the behaviour by creating a new component that inherits from the StickyGenericModule class.

StickyPopupModule

This component inherits from StickyGenericModule and can be used to display a clickable popup menu over an interactive-enabled object. A sample script (Demos\Scripts\SamplePopupOptions) is included to help you with integration into your own game.

You can create your own popup prefabs to suit the visual style of your game. The StickyPopupModule uses the default UI.Text components, however, you could create a new class and inherit from StickyPopupModule and add new variables and override virtual methods to use say Text Mesh Pro components. A couple of test popup prefabs have been included in Demos\Prefabs\Visuals to get you started.

See also “Sticky Manager Methods” in the “Runtime and API” chapter.

Sticky Moving Platform

This module helps you control the movement and rotation of a platform. You can optionally add a Sticky Zone to the platform and override the “Reference Frame” to the Moving Platform’s transform. When the S3D character comes in contact with the Sticky Zone, it will move relative to the Moving Platform. If “Restore Default Ref.” is enabled on the Sticky Zone, when the character steps off the platform, it will stop moving relative to the platform.

Property	Description
Initialise on Start	If enabled, Initialise () will be called as soon as Start () runs. This should be disabled if you want to control when the platform is enabled through code.
Move	Does the platform move?
Relative Positions	Use positions relative to the initial gameobject position, rather than absolute world space positions.
Average Move Speed	Average movement speed of the platform in metres per second
Wait Time	The time the platform waits at each position
Movement Profile	The *profile* of the platform's movement. Use this to make the movement more or less smooth.
Rotate	The starting rotation of the platform in degrees
Starting Rotation	Does the platform rotate?
Rotation Axis	The axis of rotation of the platform.
Rotation Speed	The rotational speed of the platform in degrees per second.
Positions	A list of 3D positions that the platform will move between. By default, they are in world-space. When “Relative Positions” is enabled, they are relative to the initial position and orientation of the platform.

Sticky Parts Module

This module enables you to configure individual parts on a character. For example, you the character may have a helmet that they wish to take on or off.

Property - General	Description
Initialise on Start	If enabled, Initialise() will be called as soon as Start() runs. This should be disabled if you want to control when the Sticky Parts Module is enabled through code.

Each part on your character you wish to configure, is defined by a transform.

Property - Part	Description
Part Transform	The child transform on the character for this part
Enable on Start	Should the part be enabled (or disabled) when the module is initialised?

Sticky Surface

This small component can be added to any collider to help the character determine what type of surface they are stepping on. It is used with the “Footsteps” option on the Sticky Control Module “Move” tab.

[NOTE: If you want to know how to make a character “stick” to an object, see “Reference Frames Explained” in the Sticky Control Module – Move section of this manual].

It contains a reference to a shared Scriptable Object that can be created in your Project pane in the Unity Editor. An example is provided in the Demos\Surfaces folder.

To avoid having your Surfaces Scriptable Object being overwritten during a S3D update, we suggest either creating your own (using Create, Sticky3D, Surfaces) or duplicating the Demo version and making your modifications. Don’t forget to move your version outside the Sticky3DController folder.

Property	Description
Is Terrain	Is this object a Unity terrain rather than a regular mesh?
Surface Types	A shared Scriptable Object in the Project containing a list of common surface types
Surface Type	The surface type for this collider. Used to trigger things like footstep sounds for a Sticky3D character.

Sticky XR Interactor

This feature is currently in **Technical Preview** and its behaviour may change without notice with each release to correct issues or to address particular scenarios. If you see a problem, please contact us on our Unity forum or on our Discord channel.

This component enables VR hands on a Sticky3D character to interact with interactive-enabled objects in the scene. It can also teleport the character around the scene. The Sticky XR Interactor prefab should be added as a child object to one or both of the Left- and Right-Hand transforms. These transforms can be configured in the Sticky Input Module when the Input Mode is “Unity XR”.

There is a prefab called “StickyXRInteractor1” included in the Demos\Prefabs\Visuals folder.

Sticky XR Interactor - Properties

General Property	Description
Initialise on Start	If enabled, Initialise() will be called as soon as Start() runs. This should be disabled if you want to control when the Sticky Interactive component is enabled through code.
Sticky Control Module	The Sticky3D character this interactor is a child of.
Beam Width	The width of the beam or ray displayed in the scene.
Interactor Type	Left or right hand
Look Mode	The method the interactor uses to interact with objects in the scene. The options are “Interactive” or “Teleport”.
Max Distance	Maximum distance the interactor can see ahead.
Default Beam Colour	The default colour gradient of the StickyXRInterctor beam or display target
Active Beam Colour	The colour gradient of the StickyXRInteractor beam or display target when it is interacting with an object in the scene.

IMPORTANT: When in "Interactive" mode, this will only work if "Interactive" is also enabled on the Look tab of the Sticky Control Module. This is by design.

Interactive Property	Description
Palm Transform	This needs to be a child transform of the hand. The z-axis (forward) should point in the direction of the palm is facing. This is the palm "normal". The x-axis (right) is towards the direction of the index finger and, for the right hand, the y-axis (up) is in a thumbs-up direction.
Point to Grab	When LookMode is Interactive, the beam can be pointed at a grabbable interactive-enabled object. This enables you to grab or pick up objects that is out of reach.
Point to Touch	When LookMode is Interactive, the beam can be pointed at a touchable interactive-enabled object. This enables you to "touch" objects from a distance. For example, you might want to touch a button that is out of reach.
Throw Strength	When an interactive-enabled object is dropped, the velocity of the hand is applied to it. The strength of the throw, applies more or less velocity to the object. This enables your character to have more strength in one hand that the other.
Target Sprite	The sprite that will be used instead of the pointer beam when the Point Mode is "Target".
Default Target Colour	The default colour of the pointer reticle when the Point Mode is Target
Active Target Colour	The colour of the StickyXRInteractor Target when it is interacting with an object in the scene
Target Offset	When the Point Mode is Target, this is the normalised distance the Target sprite is moved toward the hand away from the object or obstacle to help prevent clipping.
Point Mode	The visual pointing mode used with a LookMode of Interactive. E.g., Beam or Display Target.

Teleport Property	Description
Teleport Reticle	This is the prefab that is instantiated in the scene and is enabled and disabled during Teleport activities. There is a sample prefab called "S3D_XR_TelePort1" in the Demos\Prefabs\Props folder. You can also create your own and even use custom shaders if you want a particular effect.
Show Reticle Normal	Should the teleporter reticle show the destination ground normal?
Auto Disable Teleporter	Is teleporting disabled after the character is teleported to a new location?
Teleport Obstacle Check	When teleporting, check if the character would fit into the space above the location. Currently does not check for S3D characters at that location.

Sticky XR Interactor - Teleporting

Teleporting has a number of API methods that can be invoked from a Custom Input on the Sticky Input Module. These are listed in the "Sticky XR Interactor Methods" section of the "Runtime and API" chapter. A simple scenario would be to turn on the teleporter, teleport to a new location, then disable the teleporter after use. To configure this, perform the following tasks:

1. Locate or create an action in your "Input Action Asset" to enable teleporting. E.g., XRI LeftHand, Teleport Mode Active
2. On Sticky Input Module, add a Custom Input, and configure the button action from the previous step.
3. Add a "Callback Method" to the Custom Input, drag in the Sticky XR Interactor from the left hand of the character, and select "StickyXRInteractor.ToggleTeleportOn" for the function.

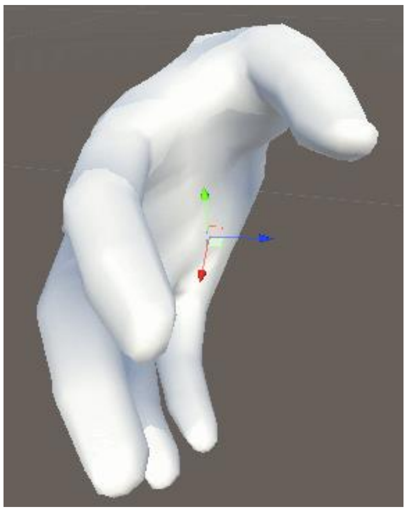
4. Locate or create an action in your “Input Action Asset” to select the new teleport location. E.g., XRI LeftHand, Teleport Select
5. On Sticky Input Module, add a Custom Input, and configure the button action from the previous step.
6. Add a “Callback Method” to the Custom Input, drag in the Sticky XR Interactor from the left hand of the character, and select “StickyXRInteractor.SelectTeleportLocation” for the function.

When teleporting, we currently check if there are any obstacles at the teleport location. However, we do not check for S3D characters. Let us know if you need to be able to do this in your project.

When teleporting is enabled, you may wish to disable “Vertical Move Input Axis” on Sticky Input Module to prevent the character moving forward or backward with continuous move. Similarly, disable “Horizontal Move Input Axis” for continuous left or right movement.

Sticky XR Interactor – Grabbing

To grab objects in your VR scene you need the following:

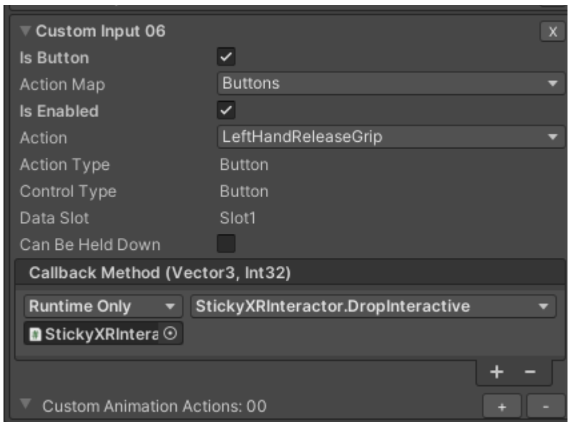


1. A grabbable Sticky Interactive object (see “Sticky Interactive” chapter)
2. A Sticky3D character configured for VR
3. A Sticky XR Interactor component as a child gameobject of a hand transform (the hand transforms are added using the Sticky Input Module when in UnityXR mode)
4. A transform on the hand that depicts the palm position and direction
5. A Sticky Input Module Custom Input to turn on the Interactor beam
6. A Sticky Input Module Custom Input to perform the grab or drop action

The VR hands require a transform that indicates the direction the palm is facing. The z-axis (forward) should point in the direction of the palm is facing. This is the palm “normal”. The x-axis (right) is towards the direction of the index finger and, for the right hand, the y-axis (up) is in a thumbs-up direction.

To perform the grab action, the Sticky Input Module Custom Input can either call the GrabLookedAtInteractive or EngageLookingAtInteractive method. The later, will examine the interactive-enabled object and take the most appropriate action.

To release your grip on an interactive-enabled object, and drop or throw it, you can setup a separate button action for the hand-held controllers or create a “Release Grip” action. Then link the action to a “Custom Input” on the Sticky Input Module.
Use a pressable button when you want to hold objects for long periods of time.



Custom Input 06

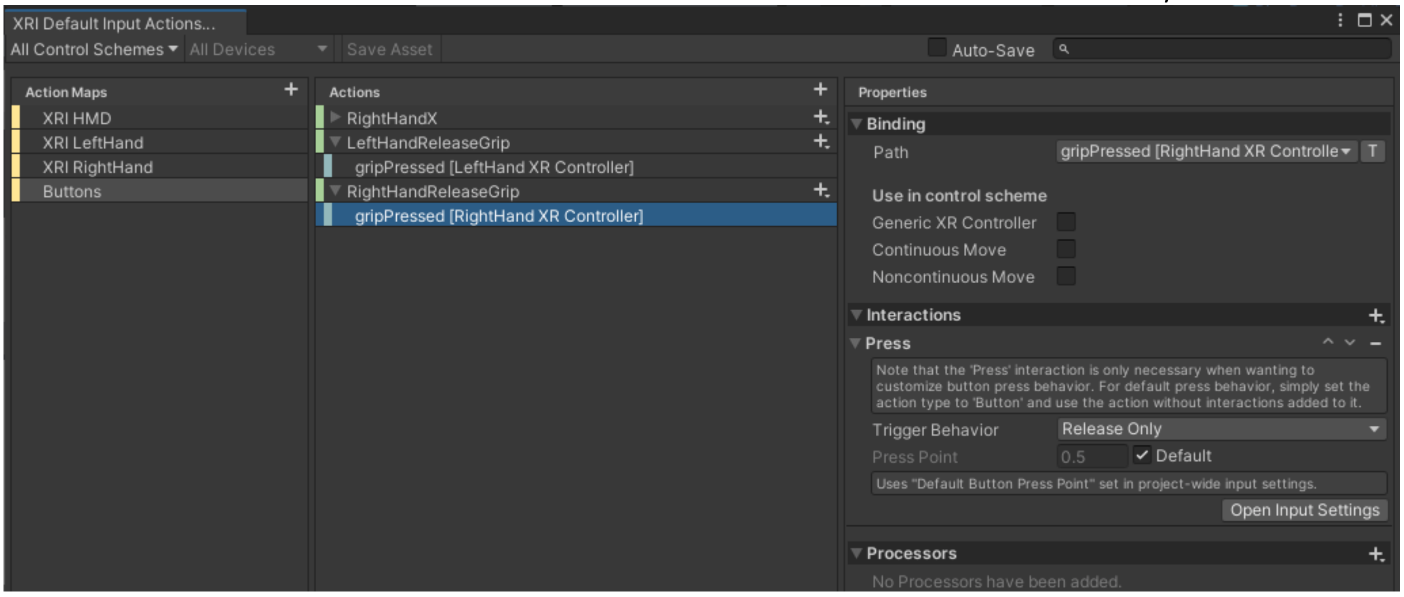
Is Button	✓
Action Map	Buttons
Is Enabled	✓
Action	LeftHandReleaseGrip
Action Type	Button
Control Type	Button
Data Slot	Slot1
Can Be Held Down	

Callback Method (Vector3, Int32)

Runtime Only | StickyXRInteractor.DropInteractive

StickyXRIntera

Custom Animation Actions: 00



XRI Default Input Actions...

All Control Schemes ▾ All Devices ▾ Save Asset Auto-Save

Action Maps
- XRI HMD
- XRI LeftHand
- XRI RightHand
- Buttons

Actions
- RightHandX
- LeftHandReleaseGrip
  - gripPressed [LeftHand XR Controller]
- RightHandReleaseGrip
  - gripPressed [RightHand XR Controller]

Properties

Binding

Path: gripPressed [RightHand XR Controlle ▾ T

Use in control scheme
- Generic XR Controller
- Continuous Move
- Noncontinuous Move

Interactions

Press

Note that the 'Press' interaction is only necessary when wanting to customize button press behavior. For default press behavior, simply set the action type to 'Button' and use the action without interactions added to it.

Trigger Behavior: Release Only

Press Point: 0.5 ✓ Default

Uses "Default Button Press Point" set in project-wide input settings.

Open Input Settings

Processors

No Processors have been added.

Sticky Zone

This is a component that works with a trigger collider to override setting on a StickyControlModule when it enters the zone or area in the scene. The zone is defined by a trigger collider that is attached to the same gameobject as the Sticky Zone component.

NOTE: If the gameobject also contains another collider, we’d recommend moving the script component and the trigger collider onto a child gameobject. The reason for this, is that when multiple colliders exist on a single gameobject, Unity creates a compound collider at runtime which can produce strange results for trigger colliders.

For characters that should react to a Sticky Zone in the scene, “On Trigger Enter/Exit” or “Trigger Collider” must be enabled on the Sticky Control Module. These settings are found on the “Collide” tab.

Sticky Zones should cover either the whole area an override should be in effect, or a doorway leading to/from the area. Sometimes, placing a zone in a doorway, can be effective for overriding the Reference Frame of a S3D Controller. If the S3D Controller is leaving the area through the doorway, there Reference Frame won’t change if the Reference Frame is already set to the current zone (unless Restore Default Ref. is ticked). However, when the S3D Controller enters the area via the doorway it will be changed.

Property	Description
Initialise on Start	If enabled, Initialise () will be called as soon as Start () runs. This should be disabled if you want to control when the Sticky Zone is enabled through code.
Reference Frame	When enabled, the reference frame of a S3D Controller entering the zone (depicted by the collider attached to the zone) will be overridden and set to the “Reference Transform” supplied.
Reference Transform	The reference transform to be used while a Sticky3D Controller is within the collider zone
Restore Default Ref.	If enabled, the default (or initial) reference frame for the S3D Controller will be restored when the controller exits the zone area.
Restore Previous Ref.	Should the previous Reference Frame transform be restored when the Sticky3D Controller exits the zone? If the previous is null, the initial or default one is restored. NOTE: Currently nesting zones within zones several layers deep will not work.
Look First Person	If enabled, when entering the zone area, set Look to First Person on the Sticky3D Controller, by turning off Third Person.

Property	Description
Look Third Person	If enabled, when entering the zone area, set Look to Third Person on the Sticky3D Controller.
Gravity	Does the zone override the gravity of Sticky3D Controllers entering it?
Gravitational Acceleration	If overridden, the gravitational acceleration to apply to Sticky3D Controllers entering the zone.
Animation Clips	Does the zone override some animation clips of the Sticky3D Controllers entering it?
Restore Clips on Exit	Should the original clips be restored when the Sticky3D Controller exits the zone?
Anim Clip Sets	One or more Anim Clip Set scriptable objects that contain original and replacement animation clip pairs. We include a few samples in Demos\AnimClipSets. You can create your own (preferably in your own game folder), by clicking Create -> Sticky3D -> Anim Clip Set in the Unity editor Project window.

Virtual Reality

Getting Started with VR

Sticky3D can be configured to work with VR headsets. In-house, we test with Oculus Quest 2 headsets and Oculus Touch hand held controllers, but it should also work with other equipment that supports Open XR.

Configuration consists of three main items:

1. Sticky Input Module – Unity XR mode
2. Sticky Control Module – Look VR mode
3. Sticky XR Interactor component

Read through the sections of the manual that pertain to the items mentioned above. This will give you a better understanding of how VR configuration works with Sticky3D.

Our VR setup assumes that the pivot point for the character is at the feet. Don’t forget to set the “Pivot to Centre Y” on the Move tab to half the height.

If you run into trouble, check the chapter called “Common Issues and How-to Advice”.

For coders, we have an extensive chapter called “Runtime and API” which includes methods and properties used in VR applications.

Configuring Hand Models

Sticky3D can work with animated hand models. Currently, you’ll need to supply your own models, animation controller, and animations. You could also use third-party hands like those included in the Oculus Integration pack.

To configure animated hands, see the “Animate – Hand VR” section in the “Sticky Control Module” chapter.

VR Comfort Settings

Motion sickness or dizziness can occur then the vision or environment moves in the VR game but in the real world the player’s body does not.

A common occurrence of this is when continuous (smooth) movement is in use, and/or continuous (smooth) turning is configured. You might want to offer:

- Snap Turn (see Sticky Control Module, “Look” tab, “Look VR Mode” settings)
- Teleporting (see Sticky XR Interactor)

Common Issues and How-to Advice

This chapter helps you resolve common setup or configuration issues.

Common Issues – General

1. In the demo scenes everything is pink. If in URP or HDRP, did see the “S3D_SRP_Readme” text file and apply the correct SRP package?
2. I keep getting “StickyControlModule - initialReferenceFrame cannot be null. Did you forget to configure it on [mycharacter]?” when the scene starts. On the “Collide” tab, try setting “Reference Update Type” to “Auto First” (see also the “Reference Frames Explained” section of the “Sticky Control Module” chapter). If that doesn’t work, you might not have correctly configured the Height and “Pivot to Centre Y” values on the “Collide” tab. With most humanoid characters, the pivot point is at the feet, so “Pivot to Centre Y” would typically be half the characters height.

Common Issues - Movement

1. My character doesn’t stop quick enough. On the Move tab, increase the “Max Acceleration”.
2. How can the user freeze character movement but still be allowed to look around in 3rd person mode? Create a Custom Input, configure a Button, and add a “Callback Method” to StickyControlModule.ToggleEnableMovement().
3. My character can’t walk up steps greater than a certain height. On the “Move” tab, increase the “Max Step Offset”.
4. My character aligns to the ground normal when “Align to Ground Normal” is turned off. On the Collide tab, you may have “Reference Update Type” set to “Automatic”. If the different meshes in your scene are not children of the first object your character comes in contact with, the reference frame will change as the character walks on those objects. You can see this at runtime in the editor, by clicking on the “Debug Mode” for the StickyControlModule, and watching the “Current Reference Frame” as the character walks around over different mesh objects. To correct this, either make all the meshes a child of a single “environment” parent gameobject (don’t put your character in this group) and set the Reference Frame to that gameobject, OR manually set the Reference Frame to the mesh with the desire “up” direction. See also “Reference Frames Explained” in the “Sticky Control Module” chapter.
5. My character jitters when it walks. On the “Look” tab, try changing the “Update Type”.

Common Issues – Look

1. Zoom doesn’t work with the mouse scroll wheel when using an Input Mode of Legacy Unity on the Sticky Input Module. Make sure you have an Axis of “Mouse ScrollWheel” set up in the Unity Input Manager AND the sensitivity is 1. The Type in the Unity Input Manager should be “Mouse Movement” and the Axis “3rd Axis (Joysticks and Scrollwheel)”
2. My environment seems to jitter when things are moving. Try setting the rigidbody “Interpolate” settings to “Interpolate”.
3. My third person camera drops through the ground when I start the scene. Did you add it to the current character when “Third Person” is enabled on the Look tab in the editor? The demo ThirdPersonCamera has a rigidbody attached to it. If it isn’t configured or controlled by a character, it will naturally fall due to the effects of Unity gravity.
4. How can I raise the level of the Third Person Camera? On the Look tab, with “Third Person” enabled, increase the Camera Offset Y value.
5. How can I place the Third Person camera behind the player? On the Look tab, with “Third Person” enabled, make the “Camera Offset” z value a negative number. The values are in metres.
6. When I’m using the mouse, the cursor keeps turning back on. On the Look tab, turn off “Auto-hide Cursor”. Optionally, on the Sticky Input Module of the player, add a “Custom Input”, assign a button to toggle the cursor on/off, add “Callback Method”, drop the character gameobject from the scene into the “Runtime Only” slot provided, and select StickyControlModule.ToggleCursor() for the function. Alternatively, you could call the ShowCursor() and HideCursor() methods in your game code.
7. Auto-hide cursor and/or the cursor APIs don’t work in the editor. This is a known problem with some versions of Unity and is outside the control of S3D. However, it should always work in a build.

8. "Look rotation viewing vector is zero" appears in the Unity console. It is most likely that you forgot to set the Third Person "Camera Offset" value on the Look tab. If this is set, please report this to us in our Unity forum or on Discord.
9. How can the player switch between a back and front view of the character? Create a Custom Input, configure a Button, and add a "Callback Method" to StickyControlModule.ToggleThirdPersonLookZ().
10. How can the player reset or cancelled the Third Person Orbit? Create a Custom Input, configure a Button, and add a "Callback Method" to StickyControlModule.SetLookOrbitAmount(). Leave the parameter as 0.
11. In Third Person with Free Look enabled, my character turns too slowly (or too quickly) when they start to move. On the Move tab, adjust the "Turn Rotation Rate".

Common Issues – Collide

1. Character is not pushed by other objects that hit it. The other object must have a collider. On the Collide tab of S3D, "Trigger Collider" must be enabled.
2. My character falls through a platform that moves up and down like a lift. The platform must have a collider and on the S3D Collide tab, "Trigger Collider" must be enabled.
3. When the moving platform moves left or right, the S3D character falls off the platform. See the sections "Sticky Moving Platform" and "Sticky Zones" for how to overcome this issue. See also "Reference Update Type" in the Collide tab.
4. My character walks right through other characters. On the Collide tab of the other S3Ds, "Trigger Collider" must be enabled.
5. My character cannot interact with other rigidbody objects. Create a new Unity Layer and add the other object to that layer. On the Collide tab, of your S3D character, make sure the Unity Layer for the other object is added to the Interaction Layer Mask. Turn on "Tigger Collider" on the Collide tab.

Common Issues – Jet Pack

1. When the Jet Pack Up button is pressed, my character moves a small distance up then stops. To resolve this, on the Sticky Input Module, under "Jump & Jet Pack Up Button", enable "Can Be Held".

Common Issues – Animate (General)

1. The list of Parameter Names is either empty or does not match the parameters in my Animator Controller. Ensure you have the "Animator" component attached to the gameobject or a child of the S3D character. Ensure the component is added in the slot provided on the Animate tab. Ensure the S3D character gameobject is enabled. Click the "Refresh" button on the Animate tab. If that doesn't work, try entering Play mode in the editor for a few seconds.
2. When walking or performing a repetitive action, my character only does it once. In the animation clip settings in the FBX importer, ensure "Loop Time" is enabled. In the Animation Controller, make sure the animation transition setting "Has Exit Time" is unchecked.
3. My character animation suddenly goes from walk to idle or idle to walk and it looks too rigid. On the "Animate" tab, where you set the walk speed, try adjusting the "Damping" value.

Common Issues – Animate (Head IK)

1. Head IK is enabled but it doesn't work. In your animation controller, ensure IK Pass is enabled on the main movement layer and the "Anim IK Pass Layer Index" on the S3D Animate tab matches that layer. Also, try testing it with "Sample Head IK Target" script and a simple cube or sphere in your scene. At runtime in the editor, click "Debug Mode" on the Sticky Control Module, and check that "Is Head IK Enabled" is true, and the "Target Position" is updated as you move the target cube or sphere around in the scene view (you will need to Lock the inspector of the S3D character to allow you to see the Debug inspector while moving the target object).
2. My player character has Head IK enabled, but the head doesn't look where the mouse or the HUD recticle is pointing. On the Look tab, insure "Interactive" and "Update Looking Point" are enabled. Now, on the Animate tab, under "Head IK", enable "Look at Interactive". These can also be set at runtime in code.

3. When the character or target object is moving quickly, the head can take some time to react. Turn on "Adjust for Velocity".

Common Issues – Animate (Hand IK)

1. Hand IK is enabled but it doesn't work. In your animation controller, ensure IK Pass is enabled on the layer that animates hand and arm bones, and the "Anim IK Pass Layer Index" on the S3D Animate tab matches that layer.

Common Issues – Animate (Foot IK)

1. When Foot IK is enabled, with some animation clips, the feet don't touch the ground. Check that the animation clips have 2 "Curves" configured. Check "Foot Distance to Ground" is correct. See the "Animate – Foot IK" section of the manual in under "Sticky Control Module".
2. When standing on a non-kinematic rigidbody, the feet can slowly move away from underneath the body. This can happen when the reference frame has a rigidbody that isn't being updated. In this state the physics location of the collider(s) is not updated by Unity as the object "drifts" in world space. To confirm this is the trouble, in the Unity editor while in play mode, select the reference frame object in the scene, go to its rigidbody, under "Constraints", toggle one of the Freeze Position axes on/off. If the characters feet snap back to their correct position, then this is your problem. To fix, either set your reference frame object to kinematic, or regularly update its rigidbody position, rotation etc.

Common Issues – Animate (Root Motion)

1. With Root Motion enabled, at idle, my character jitters all over the place. Ensure that the idle animation, on the FBX import settings, has "Root Transform Position (XZ), Bake Into Pose" enabled.
2. My root motion animation should rotate the character but does not. This is a known issue and will be resolved before this feature comes out of Technical Preview.
3. When transitioning from/to idle my character moves left or right. This is a known issue and is being investigated.
4. With Root Motion enabled, sometimes my character launches up or backward when either on a slope or up against a step. This is a known issue and is being investigated. Please let us know if you have a reproducible scenario that we can troubleshoot.
5. With Root Motion enabled, my character doesn't move. Firstly, you'll need an Animation Controller that uses Root Motion animations (in the animation the character actually moves, rather than performing the action while remaining stationary). You'll also need to send Movement Input data to the Animation Controller but setting this up on the Animate tab in S3D. See "Sticky Control Module, Animate – Root Motion" for more details.
6. With Root Motion enabled, my character just walks on the spot but doesn't go anywhere. It is possible that your animation clip is a non-Root Motion one. Play the animation in the Unity Inspector to see if it moves forward correctly.
7. With Root Motion enabled, my character doesn't walk in a straight line. On the FBX import settings, try enabling "Root Transform Rotation, Bake Into Pose", with "Based Upon" set to "Original".

Common Issues – Animate (Hand VR)

1. How do I configure animated hands for the Oculus Quest 2? Follow the instructions to setup Sticky Input Module for Unity XR. Now follow the instructions for Oculus Hand Animation in the "Animate – Hand VR" section of the chapter on "Sticky Control Module".
2. My VR hands of my Oculus Quest 2 don't appear directly in front of the character. While in the game, select the Oculus button on the right Oculus Touch Controller, point to and click "Reset view". Click any controller button to reset the view.

Common Issues – Sticky Input Module (General)

1. How do I get my player to fire a weapon? Create a public method in your own gameplay C# code that instantiates projectiles that are fired from your weapon. In Sticky Input Module, add a new "Custom Input".

Turn on “Is Button” and “Is Enabled”. Select an input button (see the Sticky Input Module chapter for more details). Add your “fire weapon” method as a “Callback Method” to the Custom Input. See the “Custom Input” section in the “Sticky Input Module” chapter for more information on configuring Custom Inputs.

2. There is a lot of jitters in the scene. The S3D character rigidbody has the Interpolate set to “Interpolate” when the scene starts if it is currently set to None. You could also set it to “Extrapolate” in the Inspector if that works better for your scene setup. Other objects in your scene that are “near” the character should also have their rigidbody “Interpolate” set to the same as your character.
3. How do I animate my character by pressing a button? For example, how can I pick up an object? See the “Animate – Custom Input and Actions” section of the StickyControlModule chapter.

Common Issues – Sticky Input Module (UnityXR)

1. Black screen when I build a project for Oculus Quest 2 and run it on the device (rather than run it from the Oculus Link (in Rift mode). Ensure you have the Oculus XR Plugin installed in Package Manager. We have tested with version 1.10.0. You can check it is installed on the Sticky Input Module.
2. How can I detect when the user releases the Grip button on a VR controller? In the Unity Input Action Asset, create an action that uses <XRController>{LeftHand}/gripPressed or <XRController>{RightHand}/gripPressed with the “Press” “Trigger Behavior” set to “Release Only”. See also “Sticky XR Interactor – Grabbing” in the “Sticky XR Interactor” chapter.

Common Issues – Sticky Interactive

1. I have Hand IK enabled on the character, but the hands never reach towards an interactive-enabled target. Make sure you IK Pass set in your Animation Controller. Ensure you StickyInteractive object has “Is Touchable” enabled. Try setting up a test scene with the Demos\Scripts\SampleHandInteract component (this shows one scenario of how-to setup Hand IK). At runtime in the editor, enable Debug mode on the StickyControlModule and verify Hand IK targets are being set correctly. Try increasing the “Max Reach Dist” on the hand you have set the interactive target for (Animate tab, Hand IK settings).
2. My character cannot see StickyInteractive objects in the scene. Enable “Interactive” on the StickyControlModule “Look” tab. Make sure your object contains a non-Trigger collider on the same gameobject as the StickyInteractive component OR it has a Rigidbody, and a non-Trigger collider which can be on a child gameobject. Alternatively, add a StickyInteractiveChild component to child non-trigger colliders. Avoid using trigger and non-trigger colliders on the same gameobject as the StickyInteractive component. Ensure the character has clear line-of-sight to the interactive-enabled object in the scene. Enable debugging on the StickyControlModule at runtime in the editor to check where the character is looking.
3. My character cannot select interactive-enabled objects in the scene. On the S3D character Engage tab, ensure the “Max Selectable in Scene” value is at least 1. At runtime in the editor, enable Debug Mode on the character and verify that the character can “see” the interactive-enabled object. If they cannot, check the previous “Common Issue”. On the StickyInteractive component for the object, ensure “Is Selectable” is enabled. It can also be helpful to use the StickyInteractiveTester component (see the chapter on this component) to help troubleshoot this issue.
4. I want to use interactive objects in VR. See the chapter on the “Sticky XR Interactor” component.
5. When my Grabbable interactive-enabled object is dropped, it doesn’t fall to the ground. Either tick “Remove Rigidbody on Grab”, add a Sample OnDropItem component and configure it in the On Dropped event OR add a non-Kinematic Rigidbody to the object, and turn off “Remove Rigidbody on Grab”. See the “Sticky Interactive – Dropping Objects” in the “Sticky Interactive” chapter for more details.

Common Issues – Sticky XR Interactor

1. How can I use a controller trigger or button to activate an interactive-enabled object that is either held or being looked at? How can I also use the same trigger or button to turn on or off the Interactor Beam (or Target)? Create an input action for the Trigger or Button in the Unity Input Action Asset. On the Sticky Input Module, add a “Custom Input”. Configure it as a button. Add a “Callback Method” to the “Custom Input” and

drag in the Sticky XR Interactor for the hand. Set the "Function" to be StickyXRInteractor ToggleInteractiveOrActivateOn.

Common Issues – Sticky Zones

1. Characters do not react to zones. On the "Collide" tab of the character's StickyControlModule, ensure "On Trigger Entry/Exit" or "Trigger Collider" are enabled. Ensure "React to Sticky Zone" is enabled (also on the "Collide" tab).
2. If the Sticky Zone has Filters, check that the character's identification on the "Move" tab will work with these.
3. The animation clip set does not override the default animations in the animator. If the character starts inside the zone area, it may prematurely raise an OnTriggerEnter event before the Sticky Zone has been initialised. Either start the scene with the character outside the Sticky Zone, or disable the Sticky Zone gameobject, and enable it in code a short time after the scene loads. You can do this by creating a method called say "InitialiseStickyZones()" and calling Invoke("InitialiseStickyZones", 0.1f) from Start(). In your new method, write something like:

```
if (!stickyZone.gameObject.activeSelf) { stickyZone.gameObject.SetActive(true); }
if (!stickyZone.IsInitialised) { stickyZone.Initialise(); }
```

Common Issues – Samples

1. When I try to pick-up items with SamplePickupItem script I can't find any items in front of the character. Either the "Pickup Distance" is too small, the objects don't have a collider, or they aren't in the correct Unity Layer to match the "Items Layer Mask".
2. In the Playground demo scene, when I walk or jump on the swing bridge, it doesn't move. Ensure the swing bridge and the child objects are assigned to a Unity Layer that is included in the S3D character's "Interaction Layer Mask" on the Collide tab. Turn on "Trigger Collider" on the Collide tab. On the rigidbody of the character, make sure it has a "reasonable" mass. E.g., 50 to 100 kg.

Runtime and API

Sample and Demo Scripts

Sample and demo scripts can be found in Demos\Scripts folder.

These scripts would not typically be used directly in your game, as they are likely to change without notice as we release new versions of the product. Instead, they are included so that you can reproduce similar results in your own game code by learning from the techniques used in them.

Setup instructions are typically included at the top of each script.

If you want to modify these scripts, create a new script in your own namespace and copy over the code you need. DO NOT MODIFY these scripts as it will be overwritten when you do the next S3D update.

Script Name	Description
DemoS3DFireProjectile	Used in GravityShooterDemo scene to fire a rigidbody projectile from a character.
DemoS3DProjectile	Used in GravityShooterDemo scene. Simple component used when firing projectiles with a collider and rigidbody.
DemoS3DTextSpawner	This demo script shows how to spawn pooled Mesh Text using a customised StickyGenericModule. This demo script is designed to work with in VR but it could be adapter (in your own code) to work with a non-VR character. See also SampleGenericTextModule.cs.

Script Name	Description
DemoRotateObject	Use in demo platform rotation scene(s) to help test character setup and configuration. If the rigidbody is set to kinematic, the object won't be rotated.
SampleAnimPlayList	Simple script to cycle through a list of animations to play for an NPC or Player S3D character.
SampleAnimPlayWithOffset	Simple script to play an animation state, starting a normalised time from the beginning of the clip. This component can be added to a StickyControlModule or an empty gameobject in the scene.
SampleCreateCustomInput	Sample script to show how to add custom inputs at runtime in your code. This sample assumes the (old) Input Manager is enabled and there is a mouse attached to your computer. However, this method could easily be adapted to any supported input mode.
SampleCustomInput	This simple sample script shows how to receive notification when a custom input action is performed.
SampleDetectCollision	Simple script to detect when another object has collided with this object.
SampleFireAtCharacter	Simple script to have a Sticky3D character fire at another character using a ray.
SampleGenericTextModule	Demo script used to show how to create your own custom StickyGenericModule component. See also Demo3DTextSpawner.cs.
SampleHandInteract	Sample script to demonstrate a Sticky3D character (S3D) interacting with an interactive-enabled object in the scene using Hand IK.
SampleLeverValue	Sample script to show reading values from an interactive lever. This example is for VR but you could also use something similar in a non-VR project.
SampleLookAtPlayer	Simple script to have a Sticky3D character look at (and optionally walk toward) a player character. There is also an option to enable Head IK. This script would be attached to a new child gameobject under the character parent gameobject. This avoids Unity creating a compound collider on the component at runtime.
SampleMoveTo	This is a very simple sample script to show how a Sticky3D character can be moved via code. It is NOT meant to be a replacement for a full NPC AI system. It doesn't include any kind of path finding or obstacle avoidance. For Teleporting - see stickyControlModule.TelePort(..).
SampleNavMeshFollowPlayer	Simple script to have a Non-Player-Character (NPC) Sticky3D character follow another character using a navmesh.
SampleNPCPlayAnim	Simple script to have a Sticky3D NPC character play an animation clip when the player comes within range. It also shows how to replace an animation clip in your animator controller. This script would be attached to a new child gameobject under the character parent gameobject. This avoids Unity creating a compound collider on the component at runtime.
SampleOnDropItem	Sample component that can be added to a StickyInteractive-enabled object in the scene that takes action when the item is dropped.
SamplePickupItem	Sample script to show how to pick up one or more items in front of the player. A similar technique could be used to drop items from an inventory system.
SampleSitAction	Simple script to attach to a character to toggle and action (sitting) on or off.

Sticky Control Module Methods - General

Method	Description
AudioMute ()	Turn off all audio on the character

Method	Description
AudioUnMute ()	Enable audio on the character
AudioToggle ()	Toggle audio on/off for the character
CalculateShoulderHeight()	Calculate or estimate the shoulder offset from the feet. See also GetCurrentShoulderCentre().
DisableCharacter (bool resetVelocity = true)	Disables character movement, look, collision detection, jetpack and animation. See also StickyInputModule.DisableInput(..) and EnableInput().
EnableCharacter (bool resetVelocity = true)	Enables character movement, look, collision detection, jetpack and animation. See also StickyInputModule.DisableInput(..) and EnableInput().
GetCurrentCentre()	Returns the current world-space centre point of the character
GetCurrentBottom()	Returns the current world-space centre bottom position of the character
GetCurrentOffsetFromBottom (Vector3 localOffset)	Returns the current world-space centre bottom position of the character with a local space offset
GetCurrentTop()	Returns the current world-space centre top position of the character
GetCurrentRelativePosition()	Returns the current position relative to the current reference frame.
GetCurrentRelativeRotation()	Returns the current rotation relative to the current reference frame.
GetCurrentRelativeVelocity()	Returns the current velocity relative to the current reference frame.
GetCurrentRelativeAngularVelocity()	Returns the current angular velocity relative to the current reference frame.
GetLocalPosition (Vector3 wsPosition)	Get a local space position on the character, given a world space position (converts a world space position to a local space position on the character).
GetLocalRotation (Quaternion wsRotation)	Get a local space rotation of a rotated object relative to the character. (Converts a world space rotation to a local space rotation on the character).
Initialise()	Initialise the Sticky Control Module. Has no effect if called multiple times.
PauseCharacter()	Pause the character but keep the look camera enabled if one is available. This is useful when pausing a game or scene. See also: DisableCharacter(), LockPosition(), UnPauseCharacter().
ResetInputDamping()	Reset the damping of input values.
SetGravitationalAcceleration (float newGravitationalAcceleration)	Sets the value of gravitational acceleration for this character in metres per second per second.
SetMoveUpdateType (MoveUpdateType newMoveUpdateType)	Set the update loop used to move the character. Automatically calls SetLookUpdateType(..) as it depends on the moveUpdateType.
SendInput (CharacterInput characterInput)	Sends input (from a player) to the character for movement control.
UnpauseCharacter()	Unpause the character. This is useful when unpausing a game or scene. See also: EnableCharacter(), UnlockPosition(), PauseCharacter().

Sticky Control Module Methods - Move

Method	Description
DisableClimbing()	Stop the character from being able to climb.
DisableMovement()	Prevent the character from moving.
DisableMovement (bool resetVelocity)	Disable character movement and reset velocity if you wish
EnableClimbing()	Allow the character to climb.
EnableMovement()	Enable the character to move
EnableMovement (bool resetVelocity)	Enable character movement and reset velocity if you wish
GetIsNPC ()	Is this a Non-Player Character?

Method	Description
LockPostion()	Stop the NPC or Player character from moving but keep its position, and rotation in sync with the reference frame. This can be useful when a character is sitting. To stop all movement, use DisableMovement(..).
RefreshFootStepSettings ()	Refresh and validate foot step settings. Call this after changing the audio source or the clips at runtime.
SetIsNPC()	Set whether or not this a Non-Player Character?
StopClimbing()	Stop the character climbing. If they are on a wall, they will fall off.
StopMoving()	Stop the character moving in a single frame. This does not disable movement.
TelePort (Vector3 deltaPosition)	Teleport or move the character by a given offset delta
TelePort (Vector3 newPosition, Vector3 newRotation, bool resetVelocity)	Teleport or move the character to a new place.
ToggleEnableMovement ()	If movement is enabled, disable it. If movement is disabled, attempt to enable it.
UnlockPosition()	Enable the NPC or Player character to move again assuming movement has not been disabled with DisableMovement().

Sticky Control Module Methods - Look

Method	Description
CentreCursor ()	Centre the hardware (mouse) cursor in the centre of the screen. WARNING: This will wait until the next frame before it returns.
DisableLook ()	Disable the Look camera(s) and camera movement
DisableLookInteractive()	The character will no longer be able to see or detect objects with a StickyInteractive component
DisableLookMovement ()	Disable the movement of the character camera
DisableSnapTurnVR()	Disable Snap Turn for VR.
EnableLook ()	Enable the Look camera(s) and attempt to enable camera movement
EnableLookInteractive()	The character will be able to see or detect objects with a StickyInteractive component
EnableLookMovement ()	Attempt to enable character camera movement. This may be unsuccessful if an appropriate first- or third-person camera hasn't been setup. Returns true if successful.
EnableSnapTurnVR()	Enable Snap Turn for VR.
GetCameraOffsetDistance()	Get the non-zoomed distance between the third person camera and the character. If possible, cache this value to avoid the sqrt. Call after changing lookCameraOffset at runtime.
GetHumanPostureVR()	Get the posture or starting position of the human player when wearing a VR head-mounted device.
GetLookingAtInteractive()	Get the interactive-enabled object that the character is currently looking toward
GetLookingAtInteractiveId()	Get the unique ID of the interactive-enabled object that the character is currently looking toward. If none, StickyInteractive.NoID (0) will be returned.
GetWorldEyePosition()	Get the current eye-level position for this character. If in third person mode or there is no camera setup, the eye position will be estimated.
GetWorldLookDirection()	Get the direction the character is looking.
HideCursor()	Hide the hardware (mouse) cursor. NOTE: This will sometimes fail to turn off the cursor in the editor Game View when it doesn't have focus, but will work fine in a build.
IsInLineOfSight	Check if this character can see the lookTo 3D position. If useEyePosition is false, it uses the centre of the character. Other S3D characters can be detected if they have "Trigger Collider" enabled on the Collide tab.

Method	Description
(Vector3 lookTo, bool useEyePosition = true, bool debugInSceneView = false)	Optionally draw a blue ray in the scene view for debugging in the Unity Editor.
ReinitialiseCameraSettings ()	Call this after changing camera properties OR changing the camera with SetLookCamera1(..)
ResetClipObjectMask()	Reset the clip object mask to their default values
SetFreeLook (bool isEnabled)	Attempt to enable or disable the Free Look mode while in third person. This will have no effect if in first person.
SetLookCamera1 (Camera newLookCamera1, bool isAutoEnableLook)	Call this when setting or changing the child camera that will be used to look around with the player. Will return false if the camera is null or it is not changed.
SetHumanPostureVR (HumanPostureVR newPosture)	Set the posture or starting position of the human player when wearing a VR head-mounted device.
SetLookCameraHeight()	If in first person mode, and isAutoFirstPersonCameraHeight is true, will attempt to set the camera height to the "average" eye height based on the player height.
SetLookFirstPerson (Camera firstPersonCamera, Transform firstPersonTransform, bool isAutoEnableLook)	Configure the controller for First Person
SetLookFollowHead (bool isEnabled)	Set the first-person camera position to react to changes in the head bone position of a humanoid character. Has no effect if the character is not initialised.
SetLookThirdPerson (Camera thirdPersonCamera, bool isAutoEnableLook, bool cutToCamera = true)	Configure the controller for Third Person. Optionally start at current camera position and don't cut directly to the offset position.
SetLookTransform (Transform newLookTransform)	Call this when setting or changing the camera transform that will be rotated when the player looks in a particular direction. For First Person, it must be a child of the player character gameobject
SetLookOrbitAmount (float orbitAmount = 0f)	Set the amount of normalised orbit rotation for the third person camera. The range of values is between -1 and 1.0. A value of 0 applies no orbital rotation. Calling SetLookOrbitAmount() will reset the orbit to the Look Camera Offset with no rotation.
SetLookUnorbitDelay (float newValue)	Set the unorbit delay for the third person camera.
SetLookUnzoomDelay (float newValue)	Set the unzoom delay for the first- or third-person camera.
SetLookUpdateType (LookUpdateType newLookUpdateType)	Set the update loop used to move or rotate the camera
SetUpdateLookingAtPoint (bool isUpdated)	Should the LookingAtPoint data be updated every frame? This is useful, if you want to know where in world space the user is looking. Currently it is only updated if Look Interactive is also enabled.
ShakeCamera (float duration, float strength)	Shake the camera for specified seconds which the given relative strength or force. If the camera is not enabled or the duration and/or strength are 0 or less, StopCameraShake() will be automatically called and the inputs ignored. Strength should be between 0 and 1.
ShowCursor()	Show the hardware (mouse) cursor. This also restarts the countdown auto-hide timer if that is enabled.
StopCameraShake()	Stop the camera from shaking

Method	Description
StopLookAtInteractive()	Stop looking at an interactive-enabled object. It does not prevent the character from looking at it again. If you want to stop the character looking at interactive-enabled objects, use DisableLookInteractive() instead.
ToggleCursor()	Toggle the hardware (mouse) cursor on or off. NOTE: This will sometimes fail to turn off the cursor in the editor Game View when it doesn't have focus, but will work fine in a build.
ToggleEnableLookMovement ()	If look movement is enabled, disable it. If look movement is disabled, attempt to enable it.
ToggleFirstThirdPerson ()	Attempt to toggle between first and third person modes
ToggleThirdPersonLookZ ()	Toggles or inverts the third person camera Z position and cuts directly to that camera position if in Third Person mode. Useful for switching between a front or back view of the character.

Sticky Control Module Methods - Collide

Method	Description
AttachCollider (Collider colliderToAttach)	When attaching an object to the character, like when grabbing an object in VR, call this method for each non-trigger collider. This helps with collision detection on the character. When grabbing interactive-enabled objects, this is automatically called by S3D.
DetachCollider (int colliderID)	When detaching or removing an object from the character, like when dropping an object in VR, call this method for each non-trigger collider. This is only required if it was first registered with AttachCollider(..).When dropping interactive-enabled objects, this is automatically called by S3D. USAGE: DetachCollider (collider.GetInstanceID())
IsColliderSelf (Collider collider)	Is the collider part of this character? Takes into consideration the capsule collider, any feet colliders, and any attached or held objects.
IsHit (Vector3 lookFrom, Vector3 lookDirection, float distance, ref RaycastHit raycastHit, bool debugInSceneView = false, int objectLayerMask = ~0)	Fire a ray from the lookFrom position in the lookDirection (which should be normalised) the distance specified. Will return true if the character was hit and the passed in raycastHit structure will be populated with the hit information. Optionally, draw a ray in the scene view while using the Unity Editor. The optional objectLayerMask enables you to only detect objects in certain Unity Layers. Trigger Collider MUST BE enabled on the Collide tab for this to work.
IsHitOther (Vector3 lookFrom, Vector3 lookDirection, float distance, ref RaycastHit raycastHit, bool debugInSceneView = false, int objectLayerMask = ~0)	Fire a ray from a position (lookFrom) typically from some point on this character, in a lookDirection, a maximum of “distance” metres. The optional objectLayerMask enables you to only detect objects in certain Unity Layers.
IsObstacle (Vector3 feetPosition, Vector3 characterUpDirection)	Is there an object preventing the character moving into the position indicated? NOTE: Currently does not check for S3D characters.
IsObstacle (float offsetBottomY, float offsetTopY, Vector3 direction, float distance)	Is there an obstacle in the given direction from the character? The obstacle could be a collider that is in the collisionLayerMask or another S3D character that has "Trigger Collider" enabled on the Collide tab.
RemoveCollider (Collider collider)	Call this before you destroy a collider that the character may be near or inside, or when you no longer wish to receive Stay or Exit events for. NOTE: When the character next enters the trigger collider, new events may be created.

Method	Description
ResetReferenceFramLayerMask ()	Reset the reference frame mask to their default values
RestoreDefaultReferenceFrame ()	Attempt to restore the current reference frame, to the initial or default setting. This is automatically called when exiting a StickyZone.
SetCurrentReferenceFrame (Transform newReferenceFrame)	Sets the current reference frame.
SetReferenceUpdateType (ReferenceUpdateType newRefUpdateType)	Change the way reference frames are determined.

Sticky Control Module Methods – Jet Pack

Method	Description
DisableJetPackAvailability ()	Prevent the Jet Pack from being enabled.
EnableJetPackAvailability ()	Allow the Jet Pack to be enabled. NOTE: This does not enable the Jet Pack, but it does allow the Jet Pack to be enabled or disabled.
ResetJetPackSettings()	Call this if changing the Jet Pack audio source or thruster effects. Correct order: 1. RestoreJetPackInitialSettings() 2. Make changes 3. ResetJetPackSettings()
RestoreJetPackInitialSettings ()	Call before making changes to audio source and/or jet pack thrusters. Correct order: 1. RestoreJetPackInitialSettings() 2. Make changes 3. ResetJetPackSettings()
ToggleIsJetPackAvailable ()	Attempt to toggle if the jet pack can be used or not

Sticky Control Module Methods – Animate (General)

Method	Description
BlendInAnimLayer (int layerIndex)	Blend in an Animator layer over time using the Layer Weight. See also SetAnimLayerBlendInDuration(..).
BlendOutAnimLayer (int layerIndex)	Blend out an Animator layer over time using the Layer Weight. See also SetAnimLayerBlendOutDuration(..).
DisableAnimate()	Disable the sending of animation data to the character animation controller
EnableAnimate()	Enable the sending of animation data to the character animation controller
GetAnimAction (int guidHash)	Get an S3DAnimAction class instance using the unique identifier (guidHash) of AnimAction from the list displayed in editor on the Animate tab.
GetAnimActionByIndex (int animActionIndex)	Get an S3DAnimAction class instance using the zero-based index or sequence number in the list which is seen in the editor on the Animate tab.
GetAnimActionHashByIndex (int animActionIndex)	Get the unique guidHash of an S3DAnimAction using the zero-based index or sequence number in the list which is seen in the editor on the Animate tab.
GetAnimActionIndex (int guidHash)	Get the zero-based index of AnimAction from the list seen in the editor on the Animate tab, using the unique identifier (guidHash) of an AnimAction. Returns -1 if not found.
GetAnimationStateId (string stateName)	Return the state Id (hash) given a state in an Animation Controller. NOTE: This does not test if the state exists in any of the layers.
GetAnimLayerData (int layerIndex)	Get the S3D internal data being stored for the Animator zero-based Layer.

Method	Description
GetArmLength (S3Dside side, ref float armLength)	This is the distance from the upper arm to the hand bone. It does not include the hand. Typically, you want to cache this value rather than calling it often.
GetBoneBottomOffset (HumanBodyBones bone, ref Vector3 offset)	Updates the offset parameter with the local space offset from the bottom of the (humanoid) character if the bone is found and returns true. Returns false if there is no valid animator controller, the character rig is not humanoid, or the bone is not found.
IKCheckEnabler()	Check to see if an IKEnabler component is required. If the Animator is not on the root gameobject of S3D and isFootIK/isHeadIK/isHandIK/isRootMotion is enabled, we need the S3DIKEnabler component to send OnAnimatorIK events to S3D. WARNING: Currently may incur some GC, so use sparingly. If this is an issue, please contact us.
ImportAnimateDataFromJson (string folderPath, string fileName)	Import a json file from disk and return as list of S3DAnimAction
IsValidHumanoid (bool showErrors = false)	Does this character look like a valid humanoid? Optionally show errors in the Unity Editor console.
PlayAnimationState (int stateId, int layerIndex, float transitionDuration = 0.2f)	Play the Animation State in the layer within the Animator Controller. Set the layerIndex to -1 if you want to play the first matching state (on any layer). If the zero-based layerIndex is set, it will also check if the state exists. For no transition, set transitionDuration to 0. For a smoother, but slower transition, increase transitionDuration.
PlayAnimationStateWithOffset (int stateId, int layerIndex, float transitionNormalised, float offsetNormalised)	Play the Animation State in the layer within the Animator Controller starting at a normalised offset from the start of the clip. Set the layerIndex to -1 if you want to play the first matching state (on any layer). If the zero-based layerIndex is set, it will also check if the state exists. An offset of 0.0 starts at the beginning, while 0.9 starts near the end of the clip for the animation state.
RefreshAnimateSettings()	Refresh and validate Animate settings. Call this after changing any Animate settings.
ReplaceAnimationClip (ref AnimationClip currentClip, AnimationClip newClip)	Replace all instances of the currentClip in the Animator Controller with a new AnimationClip. Update the reference to the currentClip when it has been replaced. If the clip does not exist, the method may fail silently.
ReplaceAnimationClipNoRef (AnimationClip currentClip, AnimationClip newClip)	Replace all instances of the currentClip in the Animator Controller with a new AnimationClip.
SetAnimLayerBlendInDuration (int layerIndex, float blendInDuration)	Set the duration, in seconds, it takes for an animator layer weight to reach 1.0. Set also See also BlendInAnimLayer (..).
SetAnimLayerBlendOutDuration (int layerIndex, float blendOutDuration)	Set the duration, in seconds, it takes for an animator layer weight to reach 0.0. See also BlendOutAnimLayer (..).
SetCustomAnimActionBoolValue (int guidHash, bool value)	Given the guidHash of a custom S3DAnimAction set its boolean value, assuming it has a matching ParamaterType. If Toggle is enabled, attempt to toggle the current value in the animation controller.
SetCustomAnimActionBoolValue (S3DAnimAction s3DAnimAction, bool value)	Given a custom S3DAnimAction set its boolean value, assuming it has a matching ParamaterType. If Toggle is enabled, attempt to toggle the current value in the animation controller.
SetCustomAnimActionFloatValue (int guidHash, float value)	Given the guidHash of a custom S3DAnimAction set its float value, assuming it has a matching ParamaterType.
SetCustomAnimActionFloatValue (S3DAnimAction s3DAnimAction, float value)	Given a custom S3DAnimAction set its float value, assuming it has a matching ParamaterType.
SetCustomAnimActionIntegerValue (int guidHash, int value)	Given the guidHash of a custom S3DAnimAction set its integer value, assuming it has a matching ParamaterType.

Method	Description
SetCustomAnimActionIntegerValue (S3DAnimAction s3DAnimAction, int value)	Given a custom S3DAnimAction set its integer value, assuming it has a matching ParamaterType.
SetCustomAnimActionTriggerValue (int guidHash, bool value)	Given the guidHash of a custom S3DAnimAction set its trigger (bool) value, assuming it has a matching ParamaterType.
SetCustomAnimActionTriggerValue (S3DAnimAction s3DAnimAction, bool value)	Given a custom S3DAnimAction set its trigger (bool) value, assuming it has a matching ParamaterType.
VerifyAnimParameter (string parameterName, S3DAnimAction.ParameterType parameterType)	At runtime, verify if a parameter of the given name and type exists on the animator controller. Returns 0 if no matching parameter is fouund, else returns the parameter hashcode. WARNING: This impacts GC and should never be called in an update loop. Use sparingly.

Sticky Control Module Methods – Animate (Head IK)

Method	Description
DisableHeadIK (bool isSmoothDisable)	Disable Head Inverse Kinematics. SmoothDisable will attempt to blend out the change over several frames.
EnableHeadIK (bool isSmoothEnable)	Enable Head Inverse Kinematics. SmoothDisable will attempt to blend the change over several frames.
GetHeadIKLookAtEyes()	When the HeadIK target is a Sticky3D character, should this character look at their eyes?
GetHeadIKLookAtInteractive()	When look interactive and Update Looking Point are enabled, while the character is stationary, should the head face either the interactive enabled object being looked at, or in the direction the character is looking?
GetHeadIKTarget ()	Get a reference to the transform (if any) Head IK is targeting. NOTE: This does not check if Head IK is enabled or not.
GetHeadIKTargetPosition ()	Get the world space position of the Head IK target (if any). Otherwise, return 0,0,0.
SetHeadIKLookAtEyes (bool isLookAtEyes)	When the HeadIK target is a Sticky3D character, should this character look at their eyes?
SetHeadIKLookAtInteractive (bool isLookAtInteractive)	When look interactive and Update Looking Point are enabled, while the character is stationary, should the head face either the interactive enabled object being looked at, or in the direction the character is looking?
SetHeadIKTarget (Transform targetTransform)	Change the object that the head should be looking at. Automatically checks if it is a S3D character. EXAMPLE: SetHeadIKTarget (myTargetObject)

SetHeadIKTarget (Transform targetTransform, Vector3 targetOffset, bool isReset = false, bool isCharacter = false)	Change the object that the head should be looking at. The target offset is in the local space of the target. When isReset is true, the head will attempt to start looking straight ahead.
	EXAMPLES:
	SetHeadIKTarget (myTargetObject, Vector3.zero);
	SetHeadIKTarget (myS3DCharacter, new Vector3(0f, 1.2f, 0f), true);
	SetHeadIKTarget (null, Vector3.zero);

ToggleEnableHeadIK ()	Attempt to turn on or off Head IK (Inverse Kinematics).

Sticky Control Module Methods – Animate (Hand IK)

Method	Description
ClearLeftHandIKInterativeTarget()	Clear or reset the interactive IK target for the left hand.
ClearRightHandIKInterativeTarget()	Clear or reset the interactive IK target for the right hand.
DisableHandIK (bool isSmoothDisable)	Disable Hand Inverse Kinematics. SmoothDisable will attempt to blend out the change over several frames.
EnableHandIK (bool isSmoothEnable)	Enable Hand Inverse Kinematics. SmoothDisable will attempt to blend the change over several frames.
GetLeftHandPalmPosition()	Get the world space palm position, or centre, of the left hand. If the hand transform is not available, Vector3.zero is returned.
GetLeftHandPalmRotation()	Get the world space rotation of the left-hand palm.
GetLeftHandPalmOffset()	Get the world space palm position, or centre, of the right hand. If the hand transform is not available, Vector3.zero is returned.
GetRightHandPalmRotation()	Get the world space rotation of the right-hand palm.
GetRightHandPalmOffset()	Get the local space offset the palm, or centre, of the hand is from the right bone position.
GetLeftHandIKPreviousPosition()	Get the world space left hand IK previous position (if any). Otherwise return 0,0,0.
GetRightHandIKPreviousPosition()	Get the world space right hand IK previous position (if any). Otherwise return 0,0,0.
GetLeftHandIKTarget()	Get a reference to the transform (if any) the left-hand IK is targeting. NOTE: This does not check if Hand IK is enabled or not.
GetRightHandIKTarget()	Get a reference to the transform (if any) the right-hand IK is targeting. NOTE: This does not check if Hand IK is enabled or not.
GetLeftHandIKTargetInteractive()	Get a reference to the Sticky Interactive object (if any) the left-hand IK is targeting. NOTE: This does not check if Hand IK is enabled or not.
GetRightHandIKTargetInteractive()	Get a reference to the Sticky Interactive object (if any) the right-hand IK is targeting. NOTE: This does not check if Hand IK is enabled or not.
GetLeftHandIKTargetPosition()	Get the world space position of the left hand IK target (if any). Otherwise return 0,0,0.
GetRightHandIKTargetPosition()	Get the world space position of the right hand IK target (if any). Otherwise return 0,0,0.
GetLeftHandMaxReachDistance()	The maximum distance the left hand will attempt to reach for an object.
GetRightHandMaxReachDistance()	The maximum distance the right hand will attempt to reach for an object.
SetLeftHandPalmOffset (Vector3 newOffset)	Set the local space offset the palm, or centre, of the hand is from the left bone position.
SetLeftHandPalmRotation (Vector3 newRelativeRotation)	Set the local space rotation of the palm, relative to the left hand bone.
SetRightHandPalmOffset (Vector3 newOffset)	Set the local space offset the palm, or centre, of the hand is from the right bone position.
SetRightHandPalmRotation (Vector3 newRelativeRotation)	Set the local space rotation of the palm, relative to the right hand bone.
SetLeftHandIKTarget (Vector3 targetPosition)	Change the world space position of where the left hand should be reaching toward.
SetLeftHandIKTarget (Transform targetTransform)	Change the object that the left hand should reach toward. EXAMPLE: SetLeftHandIKTarget (myTargetObject)
SetLeftHandIKTarget (Transform targetTransform, Vector3 targetOffset, Vector3 targetRotation, bool isReset = false)	Change the object the the left hand should reach toward with a local space offset and rotation from the object. EXAMPLE: SetLeftHandIKTarget (myTargetObject, new Vector3(0f, 0.5f, 0f), new Vector3(0f, 30f, 0f), false)

Method	Description
SetLeftHandIKTargetLookingAtInteractive ()	Set the left-hand target to be the interactive-enabled object the character is currently looking toward. If no interactive-enabled object is being looked at, the existing target is set to null.
SetLeftHandIKTargetInteractive (StickyInteractive stickyInteractive, bool isSecondaryHandHold, bool isNotifyStopTouchingOnNull = true)	Set the left-hand target to be an interactive-enabled object in the scene. The object must be touchable. WARNING: If this is called directly, or indirectly from stickyInteractive.StopTouchObject(..) you MUST set isNotifyStopTouchingOnNull as false, to avoid a StackOverflowException.
SetLeftHandMaxReachDistance (float newReachDistance)	Set the maximum distance the left hand will attempt to reach for an object.
SetRightHandIKTarget (Vector3 targetPosition)	Change the world space position of where the right hand should be reaching toward.
SetRightHandIKTarget (Transform targetTransform)	Change the object that the right hand should reach toward. EXAMPLE: SetRightHandIKTarget (myTargetObject)
SetRightHandIKTarget (Transform targetTransform, Vector3 targetOffset, Vector3 targetRotation, bool isReset = false)	Change the object the the right hand should reach toward with a local space offset and rotation from the object. EXAMPLE: SetRightHandIKTarget (myTargetObject, new Vector3(0f, 0.5f, 0f), new Vector3(0f, 30f, 0f), false)
SetRightHandIKTargetLookingAtInteractive ()	Set the right-hand target to be the interactive-enabled object the character is currently looking toward. If no interactive-enabled object is being looked at, the existing target is set to null.
SetRightHandIKTargetInteractive (StickyInteractive stickyInteractive, bool isNotifyStopTouchingOnNull = true)	Set the right-hand target to be an interactive-enabled object in the scene. The object must be touchable. WARNING: If this is called directly, or indirectly from stickyInteractive.StopTouchObject(..) you MUST set isNotifyStopTouchingOnNull as false, to avoid a StackOverflowException.
SetRightHandMaxReachDistance (float newReachDistance)	Set the maximum distance the left hand will attempt to reach for an object.
StopHandIKDisable()	If Hand IK is currently being smoothly disabled over several frames, this attempts to reverse Hand IK to being smoothly enabled.
ToggleEnableHandIK()	Attempt to turn on or off Hand IK (Inverse Kinematics). Smoothly enable or disable Hand IK.

Sticky Control Module Methods – Animate (Foot IK)

Method	Description
DisableFootIK()	Disable Foot Inverse Kinematics. This is currently in Technical Preview.
EnableFootIK()	Enable Foot Inverse Kinematics. This is currently in Technical Preview.

Sticky Control Module Methods – Animate (Root Motion)

Method	Description
DisableRootMotion()	Disable Animation Root Motion. This is currently in Technical Preview.
EnableRootMotion()	Enable Animation Root Motion. This is currently in Technical Preview.
GetRootMotionIdleThreshold()	Get the current root motion idle threshold. Character velocity magnitude below this value is considered idle.

Method	Description
SetRootMotionIdleThreshold (float newThreshold)	Change the current root motion idle threshold. Character velocity magnitude below this value is considered idle.

Sticky Control Module Methods – Engage

These methods complement the Look and Animate (Hand IK) methods and properties. They are designed to work with Sticky Interactive objects in your scene. If you are working in VR, see the “Sticky XR Interactor Methods” section.

Method	Description
DropInteractive (bool isLeftHand)	Drop the interactive-enabled object in the hand of the character.
EngageLookingAtInteractive (bool isLeftHand)	Attempt to take action using a hand, based on what the interactive-enabled object the character is currently looking at. This will be based on what features are enabled on the object. This will clear any current target if the character is not looking at an interactive-enabled object.
GetNumberSelectedInteractive()	Get the number of currently selected interactive-enabled objects in the scene. This does not verify if any of the selected items have been destroyed.
GetSelectedInteractiveByIndex (int selectedIndex)	Get one of the interactive-enabled objects currently selected by the character, using the zero-based index of the items currently selected. NOTE: “selectedIndex” is NOT the StickyInteractiveID of the object itself.
GetSelectedInteractiveByStoreItemID (int storeItemID)	Get one of the interactive-enabled objects currently selected by the character, using the StoreItemID returned from AddSelectedInteractive(..).
GrabLookedAtInteractive (bool isLeftHand)	Grab the interactive-enabled object currently being looked at (if any). Use the primary hand hold position on the object. Grabbed objects become unselected if they were previously selected.
GrabLeftHandLookedAtInteractive (bool isSecondaryHandHold)	Grab the interactive-enabled object currently being looked at (if any) with the left hand. Use the primary or secondary hand hold position on the object. Grabbed objects become unselected if they were previously selected.
GrabRightHandLookedAtInteractive (bool isSecondaryHandHold)	Grab the interactive-enabled object currently being looked at (if any) with the right hand. Use the primary or secondary hand hold position on the object. Grabbed objects become unselected if they were previously selected.
GrabInteractive (StickyInteractive stickyInteractive, bool isLeftHand, bool isSecondaryHandHold)	Grab an interactive-enabled object in the left or right hand. Use the primary or secondary hand hold on the object. Grabbed objects become unselected if they were previously selected.
ReinitialiseStoreSettings()	Reinitialise the storage and selection of interactive-enabled objects
SelectLookedAtInteractive()	Select the interactive-enabled object currently being looked at (if any)
SelectInteractive (StickyInteractive stickyInteractive)	Select an interactive-enabled item in the scene
SetStoreMaxSelectableInScene (int newValue, bool isModifyCapacity = false)	Change the maximum number of interactive-enabled items that can selected in the scene at the same time. Valid values 0-5. NOTE: Doing this multiple times at runtime with isModifyCapacity true may impact GC.
ToggleHoldInteractive (bool isLeftHand)	If an interactive-enabled object is being held in the hand, drop it. Otherwise, if a grabbable object is the target, attempt to grab it using the primary hand hold on the object.
ToggleSelectInteractive (StickyInteractive stickyInteractive)	Attempt to toggle a selectable interactive-enabled object on or off.
ToggleSelectLookedAtInteractive()	Attempt to select or unselect an interactive-enabled item in the scene that is being looked at.

Method	Description
UnselectLookedAtInteractive()	Attempt to unselect or deselect an interactive-enabled item that is currently being looked at by this character.
UnselectInteractive (StickyInteractive stickyInteractive)	Attempt to unselect or deselect an interactive-enabled item in the scene. Send notification to the object if it is unselected. Do NOT call this directly or indirectly from a onUnselected StickyInteractive event.
UnselectInteractiveByIndex (int selectedIndex)	Attempt to unselect or deselect and interactive-enabled item using the zero-based index of items currently selected by this character. Send notification to the object if it is unselected. Do NOT call this directly or indirectly from a onUnselected StickyInteractive event.
UnselectInteractiveByStoreItemID (int selectedStoreItemID)	Attempt to unselect or deselect and interactive-enabled item using the StoreItemID returned from AddSelectedInteractive(..). Send notification to the object if it is unselected. Do NOT call this directly or indirectly from a onUnselected StickyInteractive event.

Sticky Control Module Properties

Properties in the controller are typically used in your game-code to determine the current state or condition of the character. These Read Only properties are marked in the following table with [R]. A few properties, like Health and JetPackHealth, can be read or updated.

Property	Description
GetCurrentLocalVelocity	[R] Get the velocity of the character relative to the reference object.
GetCurrentForward	[R] Get the current forward direction the character is facing
GetCurrentInverseRotation	[R] [READONLY] Get the current inverse rotation in world space. Helpful to calc local space rotation GetCurrentInverseRotation * otherObjectWSRotation
GetCurrentPosition	[R] Get the current world space position
GetCurrentRotation	[R] Get the current character world space rotation
GetCurrentUp	[R] Get the current up direction for the character
GetCurrentReferenceFrame	[R] The current reference frame transform
GetLastGroundSlope	[R] Get the last recorded slope of the surface under or immediately in-front of the character. Ground slope data is only collected when walking up a slope.
GetLookingAtPoint	[R] The world space point where the camera is looking at. Look Interactive must be enabled and IsUpdateLookingAtPoint must be true. This could be a StickyInteractive object position or a point lookMaxInteractiveDistance from the camera if no interactive-enabled objects are within view.
GetReferenceUpdateTypeInt	[R] Get the current method or type for updating the reference frame
Health	The overall health of the character. Must be in the range 0 to 100. See also JetPackHealth
IsAudioMuted	[R] Is audio muted for this character?
IsAnimateEnabled	[R] Is animate enabled and ready to animate the character?
IsClimbing	[R] Is the character currently climbing a wall?
IsClimbingAtTop	[R] Is the character climbing and the shoulders are level with, or above, the top of the object being climbed?
IsCrouching	[R] Is the character crouching?
IsFootIKEnabled	[R] Is Foot IK enabled and ready for use?
IsGrounded	[R] Is the character on the ground?
IsHeadIKEnabled	[R] Is Head IK enabled and ready for use?
IsHeadIKDisabling	[R] Is Head IK in the process of being smoothly disabled over several frames?
IsInitialised	[R] Has the character movement script been initialised?

Property	Description
IsJetPackEnabled	[R] Is the jet packed feature engaged?
IsJumping	[R] Has the character started a jump in this frame?
IsLanding	[R] Is the character landing on the ground this frame?
IsLookEnabled	[R] Is the player able to look around with the camera? When disabled, the camera will be disabled.
IsLookFreeLookEnabled	[R] Is the character able to look around without the character rotating?
IsLookFixedUpdate	[R] Is the camera being moved or rotated in the FixedUpdate() loop?
IsLookInteractiveEnabled	[R] Can the character see or detect objects with an StickyInteractive component?
IsLookingAtInterative	[R] Is the character currently looking at an object with an StickyInteractive component?
IsLookMovementEnabled	[R] Is the camera permitted to be moved or rotated?
IsLookThirdPersonEnabled	[R] Is the character in third-person mode?
IsLookDownInput	[R] Is look down input being received by the character?
IsLookUpInput	[R] Is look up input being received by the character?
IsLookLeftInput	[R] Is look left input being received by the character?
IsLookMatchHumanHeightVR	[R] When Look VR is enabled, the character Height will be modified to match the approximate height of the human player based on the starting head-mounted device position above the floor.
IsLookRightInput	[R] Is look right input being received by the character?
IsLookVRAvailable	[R] Is Look VR mode available for selection or use? You need to be in first-person, not a NPC character, and have UnityXR available in the Sticky Input Module (which requires Unity 2020.3+ and XR Management).
IsLookVREnabled	[R] Is Look VR mode enabled?
IsLookSnapTurnVREnabled	[R] Is Snap Turn VR enabled? Look VR must also be available and enabled for this to be true.
IsLookWhileIdle	[R] Is the character looking left, right, up or down while not moving?
IsLookLeftOrRightWhileIdle	[R] Is the character looking left or right while not moving?
IsLookUpOrDownWhileIdle	[R] Is the character looking up or down while not moving?
IsMoveFixedUpdate	[R] Is the character being moved or rotated in the FixedUpdate() loop?
IsMovementEnabled	[R] Is character movement enabled?
IsPositionLocked	[R] Is the character position only modified when the reference frame object moves?
IsSitting	Is the character sitting down? See also Demos\Scripts\SampleSitActionPopup.cs
IsSprintInput	[R] Is sprint input being received by the character?
IsSprinting	[R] Is the character currently running? To set, call SendInput(..).
IsStepping	[R] Is the character walking up a step?
IsSteppingDown	[R] Is the character walking down a step?
IsStrafing	[R] Is the character strafing or moving sideways?
IsStrafingLeft	[R] Is the character strafing left or moving sideways to the left?
IsStrafingRight	[R] Is the character strafing right or moving sideways to the right?
IsStrafeInput	[R] Is strafe left or right input being received by the character?
IsStuck	[R] Is the character trying to move, but cannot?
IsUpdateLookingAtPoint	[R]
IsWalking	[R] Is the character walking?
IsWalkingBackward	[R] Is the character walking backwards?
IsWalkingForward	[R] Is the character walking forwards?
IsWalkingOrStrafing	[R] Is the character walking or strafing?
IsWalkingOrSprinting	[R] Is the character walking or sprinting?
IsWalkInput	[R] Is walk forward or backward input being received by the character?
JetPackHealth	The health of the Jet Pack. Must be in the range 0 to 100.
MovingSpeed	[R] The speed in metres per second the character is moving in any direction.
MovingForwardSpeed	[R] The speed in metres per second the character is moving forward

Property	Description
MovingBackwardSpeed	[R] The speed in metres per second the character is moving backward
MovingForwardBackSpeed	[R] The speed in metres per second the character is moving forward or backward
NumberOfAnimActions	[R] The number of Anim Actions configured on the Animate tab
StickyID	[R} The runtime identification number of the character.
WalkingSpeed	[R] The speed in metres per second the character is walking forward or backward
WalkingForwardSpeed	[R] The speed in metres per second the character is walking forward
WalkingBackwardSpeed	[R] The speed in metres per second the character is walking backward
SprintingSpeed	[R] The speed in metres per second the character is sprinting forward or backward
ZoomAmount	[R] Get the current amount of zoom being applied in third person mode

Sticky Control Module API Call Backs

Method field	Description
CallbackOnStuck callbackOnStuck	The name of your custom method that is called immediately after a character is detected as stuck. To avoid performance issues, action should be taken otherwise your method may be called each subsequent frame. See also stickControlModule.stuckTime and stuckSpeedThreshold.
CallbackOnChangeCamera callbackOnCameraChange	The name of your custom method that is called immediately after the look camera is changed.

Sticky Display Module API Properties

These can be found under “Public Variables” in the StickyDisplayModule.cs script. Comments and descriptions are included.

[IMPORTANT] If you are scripting display elements at runtime during the Awake() or Start() events in your game, it is possible that they run before the StickyDisplayModule is initialised even though you have “Initialise On Start” enabled in “General Settings”. If this happens, you can simply write the following code in your game (you just need a reference to stickyDisplayModule from the scene):

```
if (!stickyDisplayModule.IsInitialised) { stickyDisplayModule.Initialise(); }
```

Sticky Display Module (General) API Methods

Method	Description
Initialise ()	Initialise the StickyDisplayModule. Either set initialiseOnStart to false and call this method in your code, or set initialiseOnStart to true in the inspector and don't call this method.
ReinitialiseVariables ()	Call this if you modify any of the following at runtime. 1) displayRectList 2) displayTargetList 3) The displayReticlePanel size 4) Add or remove the MainCamera tag from a camera
ShowDisplay ()	Show the display
HideDisplay ()	Hide the display
SetCamera (Camera camera)	Set or assign the main camera used by the display for calculations
SetCanvasTargetDisplay (int displayNumber)	Set the display to use a particular monitor. Displays or monitors are numbered from 1 to 8.
SetDisplayOffset (float offsetX, float offsetY)	Set the offset (position) of the display. If the module has been initialised, this will also re-position the display.

Method	Description
SetDisplaySize (float width, float height)	Set the size of the display overlay image and text. If the module has been initialised, this will also resize the display. The values are only updated if they are outside the range 0.0 to 1.0 or have changed.
SetPrimaryColour (Color32 newColour) SetPrimaryColour (Color newColour)	Set the primary colour of the display. Only update the colour if it has actually changed. If the module has been initialised, this will also re-colour the display with the appropriate brightness.
SetBrightness (float newBrightness)	Set the overall brightness of the display.
SetCanvasSortOrder(int newSortOrder)	Set the sort order in the scene of the display. Higher values appear on top.
ToggleDisplay ()	Turn on or off the display. Has no effect if not initialised. See also ShowDisplay() and HideDisplay()

Sticky Display Module (Panels) API Methods

Method	Description
GetDisplayPanel()	Get the Display RectTransform or panel
GetGaugesPanel()	Get the parent panel for the Display Gauges. Create it if it does not already exist.
GetTargetsPanel()	Get the parent panel for the Display Targets. Create it if it does not already exist.

Sticky Display Module (Cursor) API Methods

Method	Description
ShowCursor ()	Show the hardware (mouse) cursor. This also restarts the countdown auto-hide timer if that is enabled.
HideCursor ()	Hide the hardware (mouse) cursor. NOTE: This will sometimes fail to turn off the cursor in the editor Game View when it doesn't have focus, but will work fine in a build.
CentreCursor ()	Centre the hardware (mouse) cursor in the centre of the screen. WARNING: This will wait until the next frame before it returns. Best used when a user closes a menu and returns to flying conditions.
ToggleCursor()	Toggle the hardware (mouse) cursor on or off. NOTE: This will sometimes fail to turn off the cursor in the editor Game View when it doesn't have focus, but will work fine in a build.

Sticky Display Module (Display Reticle) API Methods

Method	Description
ShowDisplayReticle()	Show the Display Reticle on the display. The display will automatically be shown if it is not already visible.
HideDisplayReticle()	Hide or turn off the Display Reticle
GetDisplayReticleGuidHash (int index)	Returns the guidHash of the Reticle in the list given the index or zero-based position in the list. Will return 0 if no matching Reticle is found. Will return 0 if the module hasn’t been initialised.

Method	Description
GetDisplayReticleGuidHash (string spriteName)	Returns the guidHash of the Reticle in the list given the name of the sprite. Will return 0 if no matching Reticle is found. WARNING: This will increase GC. Use GetDisplayReticleGuidHash (int index) where possible.
GetDisplayReticle (int guidHash)	Get a DisplayReticle given its guidHash. See also GetDisplayReticleGuidHash(..).Will return null if guidHash parameter is 0, it cannot be found or the module has not been initialised.
GetDisplayReticleSprite (int guidHash)	Get the UI sprite (image) for a DisplayReticle. See also GetDisplayReticleGuidHash(..).
ChangeDisplayReticle (int guidHash)	Change the DisplayReticle sprite on the UI panel. See also GetDisplayReticleGuidHash(..).
SetDisplayReticleOffset (Vector2 offset)	Set the offset (position) of the Display Reticle on the display. If the module has been initialised, this will also re-position the Display Reticle. Same as SetDisplayReticleOffset(offset.x, offset.y)
SetDisplayReticleOffset (float offsetX, float offsetY)	Set the offset (position) of the Display Reticle on the display. If the module has been initialised, this will also re-position the Display Reticle. Input values range from -1 to 1.
SetDisplayReticleColour (Color32 newColour) SetDisplayReticleColour (Color newColour)	Set the active Display Reticle colour. Only update the colour if it has actually changed. If the module has been initialised, this will also re-colour the Reticle with the appropriate brightness.

Sticky Display Module (Display Gauge) API Methods

Where possible, always use the methods that take the DisplayGauge or guidHash of the gauge as a parameter.

Most methods require that the StickyDisplayModule to be initialised.

Method	Description
AddGauge (string gaugeName, string gaugeText)	Add a new gauge to the display. By design, they are not visible at runtime when first added.
AddGauge (DisplayGauge displayGauge)	Add a gauge to the display using a displayGauge instance. Typically, this is used with CopyDisplayGauge(..).
DeleteGauge (int guidHash)	Delete a gauge from the display. NOTE: It is much cheaper to HideDisplayGauge(..) than completely remove it.
CopyDisplayGauge (S3DisplayGauge displayGauge, string NameOfCopy)	Create a copy of an existing S3DDisplayGauge, and give it a new name. Call AddGauge(newDisplayGauge) to make it useable in the game.
GetDisplayGaugeGuidHash (int index)	Returns the guidHash of the Gauge in the list given the index or zero-based position in the list. Will return 0 if no matching Gauge is found.
GetDisplayGaugeIndex (int guidHash)	Get the zero-based index of the Gauge in the list. Will return -1 if not found.
GetDisplayGauge (int guidHash)	Get a S3DDisplayGauge given its guidHash. Will return null if guidHash parameter is 0, it cannot be found. See also GetDisplayGaugeGuidHash(..)
GetDisplayGauge (string displayGaugeName)	Get the display gauge give the description title of the gauge. WARNING: This will increase Garbage Collection (GC). Where possible use GetDisplayGauge(guidHash) and/or GetDisplayGaugeGuidHash(index)
HideDisplayGauge (int guidHash)	Hide or turn off the Display Gauge

Method	Description
HideDisplayGauge (S3DDisplayGauge displayGauge)	Hide or turn off the Display Gauge
HideDisplayGauges()	Hide or turn off all Display Gauges. StickyDisplayModule must be initialised.
RefreshGaugesSortOrder ()	After adding or moving DisplayGauges, they may need to be sorted to have the correct z-order in on the display.
SetDisplayGaugeValueAffectsColourOn (DisplayGauge displayGauge, Color lowColour, Color mediumColour, Color highColour)	The foreground colour of the gauge will be determined by the gauge value and the low, medium and high colours. LowColour = value of 0, MediumColour when value is 0.5, and HighColour when value is 1.0
SetDisplayGaugeValueAffectsColourOff (DisplayGauge displayGauge, Color newForegroundColour)	The value of the gauge does not affect the foreground colour. When turning off this feature the new foreground colour would typically be the old foregroundHighColour.
SetDisplayGaugeOffset (S3DDisplayGauge displayGauge, float offsetX, float offsetY)	Set the offset (position) of the Display Gauge on the display. If the module has been initialised, this will also re-position the Display Gauge.
SetDisplayGaugeSize (S3DDisplayGauge displayGauge, float width, float height)	Set the size of the Gauge Panel. If the module has been initialised, this will also resize the Gauge Panel. The values are only updated if they are outside the range 0.0 to 1.0 or have changed.
SetDisplayGaugeValue (S3DDisplayGauge displayGauge, float gaugeValue)	Update the value or reading of the gauge. If Value Affects Colour (isColourAffectByValue) is enabled, the foreground colour of the gauge will also be updated.
SetDisplayGaugeText (S3DDisplayGauge displayGauge, string gaugeText)	Update the text of the gauge
SetDisplayGaugeTextAlignment (S3DDisplayGauge displayGauge, TextAnchor textAlignment)	Update the position of the text within the gauge panel
SetDisplayGaugeTextFont (S3DDisplayGauge displayGauge, Font font)	Set the font of the S3DDisplayGauge Text component
SetDisplayGaugeTextFontSize (S3DDisplayGauge displayGauge, bool isBestFit, int minSize, int maxSize)	Set the font size of the display gauge text. If isBestFit is false, maxSize is the font size set.
SetDisplayGaugeTextColour (S3DDisplayGauge displayGauge, Color newColour)	Set the Display Gauge text colour. Only update the colour if it has actually changed. If the module has been initialised, this will also re-colour the gauge text with the appropriate brightness.
SetDisplayGaugeTextDirection (S3DDisplayGauge displayGauge, DisplayGauge.DGTextDirection textDirection)	Update the rotation of the text within the gauge panel
SetDisplayGaugeTextFontStyle (S3DDisplayGauge displayGauge, FontStyle fontStyle)	Set the font style of the S3DDisplayGauge Text component
SetDisplayGaugeForegroundColour (S3DDisplayGauge displayGauge, Color newColour)	Set the Display Gauge foreground colour. Only update the colour if it has actually changed. If the module has been initialised, this will also re-colour the gauge foreground with the appropriate brightness.
SetDisplayGaugeForegroundSprite (S3DDisplayGauge displayGauge, Sprite newSprite)	Set the Display Gauge foreground sprite. This is used to render the gauge value by partially filling it.
SetDisplayGaugeBackgroundColour (S3DDisplayGauge displayGauge, Color newColour)	Set the Display Gauge background colour. Only update the colour if it has actually changed. If the module has been initialised, this will also re-colour the gauge background with the appropriate brightness.
SetDisplayGaugeBackgroundSprite (S3DDisplayGauge displayGauge, Sprite newSprite)	Set the Display Gauge background sprite. This is used to render the background image of the gauge.

Method	Description
SetDisplayGaugeFillMethod (S3DDisplayGauge displayGauge, DisplayGauge.DGFillMethod fillMethod)	Set the Display Gauge fill method. This determines how the gauge is filled
SetDisplayGaugeKeepAspectRatio (S3DDisplayGauge displayGauge, bool isKeepAspectRatio)	Sets whether or not the foreground and background sprites keep their original texture aspect ratio. This can be useful when creating circular gauges.
ShowDisplayGauge(int guidHash)	Show the Display Gauge on the display. The display will automatically be shown if it is not already visible.
ShowDisplayGauge (S3DDisplayGauge displayGauge)	Show the Display Gauge on the display. The display will automatically be shown if it is not already visible.
ShowDisplayGauges()	Show or turn on all Display Gauges. StickyDisplayModule must be initialised. The display will automatically be shown if it is not already visible.
ValidateGaugeList()	Create a new list if required

Sticky Display Module (Display Message) API Methods

Where possible, always use the methods that take the S3DDisplayMessage or guidHash of the message as a parameter. These are much more efficient than using the name of the message which will incur GC overhead.

Most methods require that the StickyDisplayModule to be initialised.

Method	Description
ShowDisplayMessage (int guidHash) ShowDisplayMessage (S3DDisplayMessage displayMessage)	Show the Display Message on the display. The display will automatically be shown if it is not already visible.
ShowDisplayMessages ()	Show or turn on all Display Messages. StickyDisplayModule must be initialised. The display will automatically be shown if it is not already visible.
HideDisplayMessage(int guidHash) HideDisplayMessage (S3DDisplayMessage displayMessage)	Hide or turn off the Display Message
HideDisplayMessages ()	Hide or turn off all Display Messages. StickyDisplayModule must be initialised.
AddMessage (string messageName, string messageText)	Add a new message to the display and returns a reference to the Display Message. By design, they are not visible at runtime when first added.
AddMessage (S3DDisplayMessage displayMessage)	Add a message to the display using a displayMessage instance. Typically, this is used with CopyDisplayMessage(..).
DeleteMessage (int guidHash)	Delete a message from the display. NOTE: It is much cheaper to HideDisplayMessage (..) than completely remove it.
CopyDisplayMessage (S3DDisplayMessage displayMessage, string NameOfCopy)	Create a copy of an existing DisplayMessage, and give it a new name. Call AddMessage(newDisplayMessage) to make it useable in the game.
GetDisplayMessageGuidHash (int index)	Returns the guidHash of the Message in the list given the index or zero-based position in the list. Will return 0 if no matching Message is found. Will return 0 if the module hasn't been initialised.
GetDisplayMessage (int guidHash)	Get a S3DDisplayMessage given its guidHash. See also GetDisplayMessageGuidHash(..).Will return null if guidHash parameter is 0, it cannot be found or the module has not been initialised.

Method	Description
GetDisplayMessage (string displayName)	Get the display message give the description title of the message. WARNING: This will increase Garbage Collection (GC). Where possible use GetDisplayMessage(guidHash) and/or GetDisplayMessageGuidHash(index).
GetDisplayMessageIndex (int guidHash)	Get the zero-based index of the Message in the list. Will return -1 if not found.
ShowDisplayMessageBackground (S3DDisplayMessage displayMessage)	Show the Display Message background on the display. The display the actual message, you would need to call ShowDisplayMessage(..).
HideDisplayMessageBackground (S3DDisplayMessage displayMessage)	Hide the Display Message background on the display. The hide the actual message, you would need to call HideDisplayMessage(..).
SetDisplayMessageOffset (S3DDisplayMessage displayMessage, float offsetX, float offsetY)	Set the offset (position) of the Display Message on the display. If the module has been initialised, this will also re-position the Display Message. Parameter: offsetX is horizontal offset from centre. Range between -1 and 1 Parameter: offsetY is vertical offset from centre. Range between -1 and 1
SetDisplayMessageSize (S3DDisplayMessage displayMessage, float width, float height)	Set the size of the Message Panel. If the module has been initialised, this will also resize the Message Panel. The values are only updated if they are outside the range 0.0 to 1.0 or have changed.
SetDisplayMessageBackgroundColour (S3DDisplayMessage displayMessage, Color newColour)	Set the Display Message background colour. Only update the colour if it has actually changed. If the module has been initialised, this will also re-colour the message background with the appropriate brightness.
SetDisplayMessageScrollDirection (S3DDisplayMessage displayMessage, int scrollDirection)	Set the Display Message scroll direction. USAGE: SetDisplayMessageScrollDirection(displayMessage, S3DDisplayMessage.ScrollDirectionLR)
SetDisplayMessageScrollFullscreen (S3DDisplayMessage displayMessage, bool isScrollFullscreen)	Set the Display Message to scroll across or up/down the full screen regardless of the message width and height. Can also be set directly with displayMessage.isScrollFullscreen = true;
SetDisplayMessageScrollSpeed (S3DDisplayMessage displayMessage, float scrollSpeed)	Set the Display Message scroll speed. Can also be set directly with: displayMessage.scrollSpeed = scrollSpeed;
SetDisplayMessageText (S3DDisplayMessage displayMessage, string messageText)	Update the text of the message.
SetDisplayMessageTextAlignment (S3DDisplayMessage displayMessage, TextAnchor textAlignment)	Update the position of the text within the message panel
SetDisplayMessageTextFont (S3DDisplayMessage displayMessage, Font font)	Set the font of the DisplayMessage Text component. The default is Arial, but you can supply your own.
SetDisplayMessageTextFontSize (S3DDisplayMessage displayMessage, bool isBestFit, int minSize, int maxSize)	Set the font size of the display message text. If isBestFit is false, maxSize is the font size set.
SetDisplayMessageTextColour (S3DDisplayMessage displayMessage, Color newColour)	Set the Display Message text colour. Only update the colour if it has actually changed. If the module has been initialised, this will also re-colour the message with the appropriate brightness.

Sticky Display Module (Display Target) API Methods

Where possible, always use the methods that take the S3DDisplayTarget or guidHash of the target as a parameter.

Most methods require that the StickyDisplayModule to be initialised.

Method	Description
AddTarget (int guidHashDisplayReticle)	Add a new Target to the HUD and returns a reference to the Display Target. By design, they are not visible at runtime when first added.
AddTargetSlots (DisplayTarget displayTarget, int numberToAdd)	Add another DisplayTarget slot to a DisplayTarget. This allows you to display another copy of the target on the HUD. If the DisplayTarget has not been initialised, the new slot panel will be added to the scene but this method will return null. This automatically updates displayTarget.maxNumberOfTargets.
DeleteTarget (int guidHash)	Delete a Target from the HUD. NOTE: It is much cheaper to HideDisplayTarget (..) than completely remove it.
DeleteTargetSlots (int guidHash, int numberToDelete)	Delete or remove a DisplayTarget slot from the HUD. This is an expensive operation. It is much cheaper to HideDisplayTargetSlot(..) than completely remove it. Automatically updates displayTarget.maxNumberOfTargets. NOTE: You cannot remove slot 0.
GetDisplayTarget (int guidHash)	Get a DisplayTarget given its guidHash. See also GetDisplayTargetGuidHash(..).Will return null if guidHash parameter is 0, it cannot be found or the module has not been initialised.
GetDisplayTargetByIndex (int index)	Get a DisplayTarget given a zero-based index in the list.
GetDisplayTargetGuidHash (int index)	Returns the guidHash of the Target in the list given the index or zero-based position in the list. Will return 0 if no matching Target is found. Will return 0 if the module hasn't been initialised.
GetDisplayTargetIndex (int guidHash)	Get the zero-based index of the Target in the list. Will return -1 if not found.
GetDisplayTargetName (DisplayTarget displayTarget) (int index)	Get the (sprite) name of the DisplayTarget using either the DisplayTarget instance or the zero-based index in the list of DisplayTargets on the HUD. WARNING: This will create GC, so not recommended to be called each frame. This is typically used for debugging purposes only.
HideDisplayTarget(int guidHash) HideDisplayTarget (DisplayTarget displayTarget)	Hide or turn off all slots of the Display Target
HideDisplayTargets ()	Hide or turn off all slots of all Display Targets.
HideDisplayTargetSlot (DisplayTargetSlot displayTargetSlot)	Hide the Display Target slot on the HUD. By design, if the HUD is not shown, the Target in this slot will not be show.
SetDisplayTargetOffset (DisplayTarget displayTarget, int slotIndex, float offsetX, float offsetY)	Set the offset (position) of the Display Target slot on the HUD. If the module has been initialised, this will also re-position the Display Target. Horizontal offset from centre. Range between -1 and 1. Vertical offset from centre. Range between -1 and 1.
SetDisplayTargetOffset (DisplayTargetSlot displayTargetSlot, float offsetX, float offsetY)	Set the offset (position) of the Display Target slot on the HUD. If the module has been initialised, this will also re-position the Display Target.
SetDisplayTargetPosition (DisplayTargetSlot displayTargetSlot, float offsetX, float offsetY)	Move the DisplayTarget slot to the correct 2D position on the HUD, based on a 3D world space position. If the camera has

Method	Description
	not been automatically or manually assigned, the DisplayTarget will not be moved.
SetDisplayTargetReticle (int guidHash, int guidHashDisplayReticle)	Set or change the Reticle assigned to a DisplayTarget. The Reticle must belong to the list of available reticles for the HUD.
SetDisplayTargetReticleColour (DisplayTarget displayTarget, Color newColour)	Set the Display Target Reticle colour. Only update the colour if it has actually changed. If the module has been initialised, this will also re-colour the reticle with the appropriate brightness.
SetTargetsViewportOffset (float offsetX, float offsetY)	Sets the viewport offset from the centre of the screen. Values can be between -1.0 and 1.0 with 0,0 depicting the centre of the screen. This is the area of the screen in which Targets will be visible. See also SetTargetsViewportSize(..).
SetTargetsViewportSize (float width, float height)	Sets the clipped viewable area of the screen that DisplayTargets can be shown. When Targets are outside this area, they will be hidden. Width and height values are between 0.1 and 1.0 of the total screen width or height. See also SetTargetsViewportOffset(..).
ShowDisplayTarget (int guidHash) ShowDisplayTarget (DisplayTarget displayTarget)	Show the Display Target on the HUD. By design, if the HUD is not shown, the Targets will not be show.
ShowDisplayTargets ()	Show or turn on all Display Targets.
ShowDisplayTargetSlot (DisplayTargetSlot displayTargetSlot)	Show the Display Target slot on the HUD. By design, if the HUD is not shown, the Target in this slot will not be show.

Sticky Display Module API Call Backs

Callback	Description

CallbackOnBrightnessChange callbackOnBrightnessChange	The name of the custom method that is called immediately after brightness has changed. Your method must take 1 float parameter. This should be a lightweight method to avoid performance issues. It could be used to update your custom display elements.
	stickyDisplayModule.callbackOnBrightnessChange = YourMethod;
	public void YourMethod (float value)
	{
	  // Adjust the brightness of your elements here
	}

CallbackOnSizeChange callbackOnSizeChange	The name of the custom method that is called immediately after the display size has changed. This method must take 2 float parameters. It should be a lightweight method to avoid performance issues. It could be used to update your custom display elements.

Sticky Input Module Methods - General

Method	Description
DisableInput (bool allowCustomInput = false)	Disable input or stop the Sticky Input Module from receiving input from the configured device. When allowCustomInput is true, all other input except Custom Inputs are ignored. This can be useful when you want to still receive actions that generally don't involve character movement.
EnableInput ()	Enable the Sticky Input Module to receive input from the configured device. The module will be initialised if it isn't already.
GetStickyControlModule (bool forceCheck = false)	Retrieves a reference to the StickyControlModule script if one was attached at the time this module was initiated. The optional forceCheck parameter will ignore cached value and call GetComponent when true.
Initialise ()	This must be called on Awake or via code before the Sticky Input Module can be used.
IsLegacyAxisValid (string axisName, bool showError)	This static method will check that a Unity Legacy Input system axis is defined
ReinitialiseCustomInput()	This should be called if you modify the CustomInputs at runtime
ResetInput()	Reset and send 0 input on each axis to the controller
ValidateLegacyInput ()	Validate all the legacy input axis names. Update their status.
ValidateDirectKeyboardInput ()	Validate the legacy mouse input axis we use for Mouse Look in Direct Keyboard and mouse

Sticky Input Module Methods - Rewired

Method	Description
CheckRewired (bool showErrors)	Verify that Rewired's Input Manager is in the scene and has been initialised.
GetRewiredPlayer (int userNumber, bool showErrors)	Get the Rewired Player class instance given a human user number. e.g., Get the first human player. Rewired.Player rewiredPlayer = StickyInputModule.GetRewiredPlayer(1, false);
SetRewiredPlayer (int playerNumber, bool showErrors = false)	Set the human player number. This should be 1 or greater. The player must be first assigned in Rewired, before this is called. NOTE: If Rewired is not installed or the InputMode is not Rewired, the rewiredPlayerNumber is set to 0 (unassigned).
UpdateRewiredActionTypes (bool showErrors)	A Rewired Action can be an Axis or a Button. To avoid looking up the Action within the Update event, the type is set outside the loop and need only be called once at runtime for this character, and then whenever the Actions are changed. Has no effect if Rewired is not installed.

Sticky Input Module API Call Backs

Property or Method	Description

Sticky Input Module Properties

Property	Description
IsInitialised	Is the Sticky Input Module initialised and ready for use?
IsInputEnabled	Is the StickyInputModule currently enabled to send input to the Sticky Control Module? See EnableInput() and DisableInput(..)

Property	Description
IsCustomInputOnlyEnabled	Is all input except CustomerInputs ignored? See EnableInput() and DisableInput(..)
GetCharacterInput	Gets a reference to the input being sent from the StickyInputMode to the StickyControlModule

Sticky Interactive Methods

These are the methods used for interactive-enabled objects in the scene. If you are looking for the VR component attached to the character's hands, see "Sticky XR Interactor Methods".

Method	Description
DisableNonTriggerColliders()	Disable all non-trigger colliders that were enabled during initialisation.
DisableTriggerColliders()	Disable all trigger colliders that were enabled during initialisation.
EnableNonTriggerColliders()	Enable all non-trigger colliders that were enabled during initialisation.
EnableTriggerColliders()	Enable all trigger colliders that were enabled during initialisation.
GetActivePopupID()	Return the Instance ID of the active StickyPopupModule. If there isn't one, return 0. See also SetActivePopupID(..).
GetHandHoldLocalOffset (bool isSecondaryHandHold)	Get the local space hand hold offset. By default, uses the first or primary hand hold.
GetHandHoldLocalRotation (bool isSecondaryHandHold)	Get the local space hand hold rotation. By default, uses the first or primary hand hold.
GetHandHoldPosition (bool isSecondaryHoldPosition)	Get the world space hand hold position. By default, uses the first or primary hand hold.
GetHandHoldNormal (bool isSecondaryHoldPosition)	Get the world space hand hold normal (or direction it is facing). By default, uses the first or primary hand hold.
GetHandHoldRotation (bool isSecondaryHandHold)	Get the world space hand hold rotation. By default, uses the first or primary hand hold.
GetPopupPosition()	Get the world space default popup position.
GetSitOffsetPosition()	Get the world space sit offset position. This is the location the character should stand before attempting to sit down. The position may need to be adjusted if characters have a different radius from one another.
RemoveListeners()	Call this when you wish to remove any custom event listeners, like after creating them in code and then destroying the object. You could add this to your game play OnDestroy code.
RestoreParent()	If there is a previous parent transform recorded, re-parent the object to that transform.
RestoreRigidbodySettings()	If there was a rigidbody attached to the object when it was grabbed, this will restore those original settings.
SetActivePopupID (int instanceID)	Set the instance ID of the active StickyPopupModule currently being displayed. See also GetActivePopupInstanceID().
SetHandHoldOffset (Vector3 handHoldOffset, bool isSecondaryHandHold)	Set the hand hold local space offset. By default, sets the primary hand hold offset.
SetHandHoldRotation (Vector3 handHoldRotation, bool isSecondaryHandHold)	Set the hand hold relative rotation. The rotation is stored as Euler angles (degrees). By default, sets the primary hand hold rotation.
SetIsActivable (bool activable)	Set this interactive-enabled object to be activable or not.
SetIsAutoUnselect (bool unselect)	Set this interactive-enabled object to be auto unselected or not in the scene.
SetIsGrabbable (bool grabbable)	Set this interactive-enabled object to be grabble or not.

Method	Description
SetIsReadable (bool readable)	Set this interactive-enabled object to be readable or not.
SetIsSelectable (bool selectable)	Set this interactive-enabled object to be selectable or not in the scene.
SetIsSittable (bool sittable)	Set this interactive-enabled object is suitable for sitting on or not.
SetIsTouchable (bool touchable)	Set this interactive-enabled object to be touchable or not by a character. Typically used with Hand IK.
SetMass(float newMass)	Set the mass of the object in kilograms.
SetObjectParent (Transform parentTfrm)	Parent this object to another transform. If Reparent on Drop is enabled, remember the original transform.
SetReadableJoint (Joint newReadableJoint)	Set the joint to read positional data from when Is Readable is true.

Virtual methods can be overridden to customise the behaviour of the Sticky Interactive objects. It is an advanced feature and generally is not required. Most of the time you can simply add your own methods, call APIs, and/or set properties with the various on[Event]s included and configurable in the editor.

Virtual Method	Description

virtual void ActivateObject (int stickyID)	Attempt to activate the object. Only initialised and activable objects can be activated.
	If required, you can override this method.
	public override void ActivateObject (int stickyID)
	{
	  base. ActivateObject (int stickyID);
	  // Do stuff here
	}

virtual void DeactivateObject (int stickyID)	Attempt to deactivate the object. Only initialised and activable objects can be deactivated.
	If required, you can override this method.
	public override void DeactivateObject (int stickyID)
	{
	  base.DeactivateObject (int stickyID);
	  // Do stuff here
	}

virtual void DropObject (int stickyID)	Re-parent if required, and invoke an OnDropped items. If “Parent on Grab” is enabled, the object will be unparented from the hand. Only initialised and grabbable objects can be dropped. If you wish to reenable colliders, call the API methods from the onDropped events in the editor.
	If required, you can override this method.
	public override void DropObject (int stickyID)
	{
	  base.DropObject (int stickyID);
	  // Do stuff here
	}

virtual void GrabObject (int stickyID)	Disable colliders if required, and invoke an OnGrabbed items. Only initialised and grabbable objects can be grabbed. If there is a rigidbody attached, remove it.
	If required, you can override this method.
	public override void GrabObject (int stickyID)
	{
	  base.GrabObject (int stickyID);
	  // Do stuff here
	}

Virtual Method	Description

virtual void Initialise()	Initialise the StickyInteractive component at runtime. Has no effect if already initialised. If you wish to override this in a child (inherited) class you almost always will want to call the base method first.
	public override void Initialise ()
	{
	  base. Initialise ();
	  // Do stuff here
	}

virtual void SelectObject (int stickyID, int selectedIndex)	Currently interactive-enabled objects can be selected by multiple characters or things at the same time. The thing calling this method needs to keep track of if this object is selected or not. S3D characters do this automatically. Only initialised and selectable objects can be selected.
	If required, you can override this method.
	public override void SelectObject (int stickyID, int selectedStoreItemID)
	{
	  base. SelectObject (int stickyID, int selectedStoreItemID);
	  // Do stuff here
	}

virtual void UnselectObject (int stickyID)	Currently interactive-enabled objects can be selected by multiple characters or things at the same time. The thing calling this method needs to keep track of if this object is selected or not. S3D characters do this automatically. Only initialised objects can be unselected. It doesn't need to be selectable as that may have just been turned off.
	If required, you can override this method.
	public override void UnselectObject (int stickyID)
	{
	  base. UnselectObject (int stickyID);
	  // Do stuff here
	}

virtual void TouchObject (Vector3 hitPoint, Vector3 hitNormal, int stickyID)	The interactive-enabled object has been touched at a point with a given normal.
	If required, you can override this method.
	public override void TouchObject (Vector3 hitPoint, Vector3 hitNormal, int stickyID)
	{
	  base.TouchObject (Vector3 hitPoint, Vector3 hitNormal, int stickyID);
	  // Do stuff here
	}

virtual void StopTouchingObject (int stickyID)	Stop touching this interactive-enabled object. Currently, multiple characters or things can be touching this object at the same time.
	If required, you can override this method.
	public override void StopTouchingObject (int stickyID)
	{
	  base. StopTouchingObject (int stickyID);
	  // Do stuff here
	}

Sticky Interactor Bridge

These methods are typically used for non-Sticky3D character hands attached to a 3rd party asset that you want to interact with objects in your scene. See the “Sticky Interactor Bridge” chapter for more details.

Many of the methods are “virtual” and can be overridden in an inherited class.

Virtual Method	Description
virtual void DropInteractive()	Drop a held interactive-enabled object
virtual void EngageInteractive()	If there is an object being held, attempt to activate or deactivate it. If no object is being held, but is being touched and is grabbable, attempt to grab it. If no object is being held, but is being touched and is non-grabbable, attempt to activate or deactivate it.
virtual void GrabInteractive (StickyInteractive stickyInteractive, bool isSecondaryHandHold)	Grab an interactive-enabled Grabbable object. Override this method to: 1. Perform a specific grab action 2. Determine if you should be using the primary or secondary hand hold position.
virtual void Initialise()	Initialise the StickyInteractorBridge. Has no effect if called multiple times.
virtual void StopTouchingInteractive (StickyInteractive stickyInteractive)	Stop touching an interactive-enabled object.
virtual void TouchInteractive (StickyInteractive stickyInteractive, Vector3 hitPoint, Vector3 hitNormal)	Indicate that this is component is touching a touchable interactive-enabled object.

Method	Description
GetHandPalmPosition()	Get the world space palm position
GetHandPalmRotation()	Get the world space palm rotation
SetIsCanActivate (bool canActivate)	Set if this item can activate an interactive-enabled Activable object?
SetIsCanGrab(bool canGrab)	Set if this item can grab an interactive-enabled Grabbable object?
SetIsCanTouch (bool canTouch)	Set if this item can touch an interactive-enabled Touchable object?
SetHandPalmTransform (Transform newPalmTransform)	Set the transform that represents hand palm position and rotation. Returns true if set successfully.

Sticky Parts Module Methods

You may notice that we never reference the name of the part. This is because strings at runtime impact garbage collection (GC), which is bad for performance. The module must be initialised before calling the majority of methods.

Method	Description
DisableAllParts	Attempt to disable all the parts
DisablePart (S3DPart s3dPart)	Attempt to disable a part
DisablePartByIndex (int index)	Attempt to disable a part using the zero-based index of the part
DisablePartByHash (int guidHash)	Attempt to disable a part using the unique guidHash of the part
EnablePart (S3DPart s3dPart)	Attempt to enable a part
EnablePartByIndex (int index)	Attempt to enable a part using the zero-based index of the part
EnablePartByHash (int guidHash)	Attempt to enable a part using the unique guidHash of the part
GetPartByHash (int guidHash)	Get a S3DPart given the unique guidHash of the part. Always returns null if not initialised or guidHash is 0.

Method	Description
GetPartByIndex (int index)	Get a S3DPart given the zero-based index in the list of parts. If always returns null if not initialised or the index is out of range.
GetPartIndex (int guidHash)	Get the zero-based index of a part with the given unique guidHash value. Always returns -1 if not initialised or guidHash is 0.
GetPartGuidHash (int index)	Get the unique guidHash of a part given its zero-based index in the list of parts. Always returns 0 if not initialised, the index is out of range, or the part is null.
Initialise()	Initialise the StickyPartsModule at runtime. Has no effect if already initialised.
IsPartEnabledByIndex (int index)	Is the part enabled, given the zero-based index of the part in the list.
ResetParts()	Attempt to reset the parts to their states after the module was first initialised.
ToggleEnablePartByIndex (int index)	Toggle the part on or off, given the zero-based index of the part in the list. Will have no effect if not initialised or the index is out of range. You could call this from a Custom Input on the StickyInputModule to say take on/off a helmet.
ToggleEnableParts ()	Toggle all the parts on or off. This has no effect if not initialised. See also ToggleEnablePartByIndex(..) to toggle individual parts.

Sticky Parts Module Properties

Read Only properties are marked in the following table with [R].

Property	Description
IsInitialised	[R] Has the module been initialised?
NumberOfParts	[R] The number of S3DParts configured with this module.

Sticky Manager Methods - General

The object pooling system is centrally controlled by the Sticky Manager.

Method	Description
GetOrCreateManager()	This static method returns the current Sticky Manager instance for this scene. If one does not already exist, a new one is created. If the manager is not initialised, it will be initialised.
Initialise()	Initialise the Sticky Manager. This is called automatically when GetOrCreatemanager() is called.
GetGenericPrefab (int genericPrefabID)	Returns the prefab for a generic module given its generic prefab ID.
GetorCreateGenericPool (StickyGenericModule genericObjectPrefab)	Get the generic pool for this prefab or create a new pool if one does not already exist. Return the genericObjectPrefabID for the pool or -1 if it fails. See also InstantiateGenericObject(igParms).
InstantiateGenericObject (ref S3DInstantiateGenericObjectParameters igParms)	Instantiates the generic object with ID genericModulePrefabID at the position specified in world space.
PauseGenericObjects()	Pause all pooled Generic Objects in the scene. This is useful when pausing your game.
ResumeGenericObjects()	Resume all pooled Generic Objects in the scene. This is useful when unpausing your game.

Sticky Manager Methods – StickyGenericModule

These methods apply to the StickyGenericModule which is used with the Sticky Manager pooling system. Virtual methods can be overridden to customise the behaviour of the StickyGenericModule. It is an advanced feature and generally is not required.

Method	Description
virtual void DestroyGenericObject()	Destroy (deactivate) the Generic Module. Returns it to the pool.
virtual void DisableModule()	Temporarily disable the module. Typically called automatically from stickyManager.PauseGenericObjects().
virtual void EnableModule()	Re-enable the module. Typically called automatically from stickyManager.ResumeGenericObjects().
virtual uint Initialise (S3DInstantiateGenericObjectParameters igParms)	Initialise the module after it has been activated in the pool managed by StickyManager.

Sticky Manager Methods – StickyPopupModule

These methods apply to the StickyPopupModule which is used with the Sticky Manager pooling system. Virtual methods can be overridden to customise the behaviour of the StickyPopupModule. It is an advanced feature and generally is not required.

Method	Description
virtual void DeactivatePopup()	Before the popup is deactivate / destroyed and returned to the pool, this cleans up the state of the items. If there was an interactive-enabled object, inform that object that the popup is no longer active.
virtual void FaceCamera()	Keep the popup facing the camera. This gets automatically called each frame from Update(). Override this is you want the canvas to rotate or move specifically for your game.
virtual void HideItem (int itemNumber)	Hide the option or item in the popup. The itemNumber must be in the range of 1 to the NumberOfItems.
virtual bool InitialisePopup()	Initialise the items in the Popup
virtual void SetCamera (Camera camera)	Sets the world camera of the canvas so that the popup can face the camera.
virtual void SetInteractiveObject (StickyInteractive stickyInteractive)	Set the interactive-enabled object reference for this popup. This enables you to discover which interactive-enabled object is showing the popup when an item is clicked. It informs the interactive object this popup is active and parents itself to the object.
virtual void SetItem (int itemNumber, string itemText, bool isShown = true, bool isEnabled = true)	Set the option or item in the popup. The itemNumber must be in the range of 1 to the NumberOfItems. To avoid setting the strings at runtime, you could have them preset in the prefab.

virtual void SetItemAction (int itemNumber, CallbackOnItemClick onItemClickedMethod)	Assign a callback method for an item. This enables you to call your own game code when an item is clicked. The itemNumber must be in the range of 1 to the NumberOfItems.
	Your custom method must take three parameters. E.g.
	
	public void ItemClicked (StickyPopupModule stickyPopupModule, int itemNumber, StickyInteractive stickyInteractive)
	{
	}
	
	See Demos\scripts\SamplePopupOptions.cs for an example.

virtual void ShowItem (int itemNumber)	Show the option or item in the popup. The itemNumber must be in the range of 1 to the NumberOfItems.

Sticky XR Interactor Methods

These API methods work with the Sticky XR Interactor component that can be attached to the hands of a VR character.

Method	Description
ActivateInteractive()	Attempt to activate an interactive-enabled object. If one isn't being held in the hand, check what is being looked at. Objects that are grabbable, must be held to be activated.
CheckHeldInteractive()	Check if there is an item being held in the hand. If the item being held was destroyed, make sure it isn't selected by the character, and return false.
CheckLookingAtInteractive()	Check if the character is looking or pointing at an interactive-enabled object. Also checks to see if the item hasn't been destroyed.
DeactivateInteractive()	Attempt to deactivate an interactive-enabled object. If one isn't being held in the hand, check what is being looked at. Objects that are grabbable, must be held to be deactivated.
DisableBeam()	Attempt to turn off the StickyXRInteractor beam.
DisableInteractive()	Attempt to turn off Interactive mode.
DisableTeleport()	Attempt to turn off Teleporting.
DropInteractive ()	Drop the interactive-enabled object in the hand of the character.
EnableBeam()	Attempt to turn the on StickyXRInteractor beam.
EnableInteractive()	Attempt to enable interactive mode.
EnableTeleport()	Attempt to turn on teleporting.
EngageInteractive()	If there is an object being held, attempt to activate or deactivate it. If no object being held, attempt to engage with what is being looked at.
EngageLookingAtInteractive()	Attempt to take action using a hand, based on what the interactive-enabled object the character is currently looking at. This will be based on what features are enabled on the object.
GetBeamEndPoint()	Return the last known world space end point of the beam.
GetLookMode()	Return the current LookMode of the StickyXRInteractor.
GetHandPalmPosition()	Get the world space hand palm position.
GetHandPalmRotation()	Get the world space palm rotation.
GetHandPalmTransform()	Get the transform that represents the position and rotation of the palm of the hand. See also SetHandPalmTransform(..).
GetInteractivePointMode()	Get the InteractivePointMode of the Sticky XR Interactor. e.g., Beam or Target
GetInteractorType()	Get the InteractorType of the Sticky XR Interactor. e.g., left or right hand
GetLookingAtInteractive()	Get the interactive-enabled object that the character is currently looking toward or pointing at with the beam.
GetLookingAtInteractiveId()	Get the unique ID of the interactive-enabled object that the character is currently looking toward or pointing at with the beam. If none, StickyInteractive.NoID (0) will be returned.
GetTargetSprite()	Return the sprite that is used to show where the hand is pointing when Interactive Point Mode is Target.
GrabLookedAtInteractive (bool isSecondaryHandHold)	Grab the interactive-enabled object currently being looked at (if any). Use the primary or secondary hand hold position on the object. Grabbed objects become unselected if they were previously selected.
GrabInteractive (StickyInteractive stickyInteractive, bool isSecondaryHandHold)	Grab an interactive-enabled object in the palm of the hand. se the primary or secondary hand hold on the object. Grabbed objects become unselected if they were previously selected.
Initialise ()	Initialise the StickyXRInteractor. Has no effect if called multiple times.
MovePointer()	Move the beam or pointer in the scene at runtime. Typically, this will be called automatically each frame.
ReinitialiseTeleporting()	Reinitialise teleporting. Call this each time you change the TeleportReticle prefab at runtime.
SelectLookingAtInteractive()	Attempt to select the interactive-enabled and store it in characters engage store.

Method	Description
SelectTeleportLocation()	When teleporting is enabled, select the current teleport location and attempt to teleport the character to that location. NOTE: currently does not check if any S3D character is at that location.
SetActiveBeamGradient (Gradient newGradient)	Set the active colour gradient of the StickyXRInteractor beam.
SetDefaultBeamGradient (Gradient newGradient)	Set the default colour gradient of the StickyXRInteractor beam.
SetTargetSprite (Sprite newSprite)	Sets the sprite that is used to show where the hand is pointing when Interactive Point Mode is Target.
SetHandPalmTransform (Transform newPalmTransform)	Set the transform that represents hand palm position and rotation. The StickyControlModule must be set first. Returns true if set successfully.
SetInteractivePointMode (InteractivePointMode newInteractivePointMode)	Set the InteractivePointMode of the Sticky XR Interactor. e.g., Beam or Target
SetInteractorType (InteractorType newInteractorType)	Set the InteractorType of the Sticky XR Interactor. e.g., left or right hand
SetLookMode (LookMode newLookMode)	Set a new LookMode for the StickyXRInteractor.
SetPalmTransform (Transform newPalmTransform)	Set the transform that represents hand palm position and rotation. The StickyControlModule must be set first. Returns true if set successfully.
SnatchHeldInteractive (StickyInteractive stickyInteractive)	Something else snatches an interactive object that is currently being held in the hand. This does not invoke the Drop action or events. If the snatch is successful, return true.
StopLookAtInteractive()	Stop looking at an interactive-enabled object. It does not prevent the StickyXRInteractor from looking at it again.
StopTouchingInteractive (StickyInteractive stickyInteractive)	Stop touching an interactive-enabled object. Typically this is automatically called by a StickyXRTouch component attached to the hand's trigger collider.
StopTouchingLookingAtInteractive ()	If touching an interactive-enabled object that the handing is pointing to, stop touching it.
ToggleBeamOn()	Attempt to toggle the StickyXRInteractor beam on or off.
ToggleInteractiveOn()	Attempt to toggle the StickyXRInteractive interactive mode on or off.
ToggleInteractiveOrActivateOn()	If an activable object is held, activate or deactivate it. If an activable object is being looked at, activate or deactivate it. If neither of the above, toggle Interactive on or off.
ToggleSelectLookedAtInteractive ()	Attempt to select or unselect an interactive-enabled item in the scene that is being looked at.
ToggleTeleportOn()	Attempt to toggle the StickyXRInteractive teleporting on or off.
TouchInteractive (StickyInteractive stickyInteractive, Vector3 hitPoint, Vector3 hitNormal)	Indicate that this is hand is touching a touchable interactive-enabled object. Typically, this is called automatically by a StickyXRTouch component attached to the hand's trigger collider.
TouchLookingAtInteractive()	Attempt to touch an interactive-enabled object that is being looked at or pointed at with the beam. This requires the object to be Touchable, and Point to Touch be enabled on this component.
UnselectLookedAtInteractive()	Attempt to unselect or deselect an interactive-enabled item that is currently being looked at by this character.

Sticky XR Interactor Properties

Read Only properties are marked in the following table with [R].

Property	Description
GetLookingAtPoint	[R] If the interactive beam is active, this could be a StickyInteractive object position or a point maxDistance from the hand if no interactive-enabled objects are being pointed at.
IsInitialised	[R] Has the module been initialised?
IsBeamEnabled	[R] Is the interactor beam enabled and visible in the scene?
IsInteractiveEnabled	[R] Is the interactive feature currently active?
IsTeleportEnabled	[R] Is the teleporting feature currently active?
TeleportLocation	[R] If the teleporter is active, return the potential Teleportation location in world space. Otherwise, return 0, 0, 0.
IsTeleportLocationValid	[R] If the teleporter is enabled, is the location a potential teleportation candidate?

Support

The best places get support are from the community forms which we monitor. We would encourage you to use the social media support options rather than email as you will most likely get a faster response.

Support Option	Location
Unity Forum	https://forum.unity.com/threads/995707/
Discord Channel	https://discord.gg/ZqEeBF584r
Email	support@scsmmedia.com

Version History

First Alpha version November 2020

First Beta version February 2021

Release version March 2021

Version 1.0.1 - 26 Apr 2021

[NEW] Foot IK - in Technical Preview
[NEW] Root Motion support - in Technical Preview
[NEW] PlayerJaneSuited jet pack demo character
[NEW] Animate - pass Input values to the animation controller
[NEW] Animate - use Input data as conditions for Animation Actions
[NEW] Switch between front and rear view with ToggleThirdPersonLookZ API
[NEW] Orbit around a character in third person mode
[NEW] Sticky Input Module - jet pack only option for crouch input
[NEW] Sticky Input Module - option to configure 3rd person orbit controls
[NEW] Tested with Mixamo characters and animations
[NEW] Tested with UMA 2 characters
[FIXED] Third Person - Camera Offset can be incorrect
[IMPROVED] Single button click to create new JetPack audio source
[IMPROVED] Adjust third person zoom speed (duration)
[IMPROVED] Third Person camera stability
[IMPROVED] Third Person - when camera offset is not set, a warning will be shown
[IMPROVED] If the Animator component is misconfigured, a warning will be shown

Version 1.0.2 – 3 June 2021

[NEW] Head IK with API
[NEW] Sticky Parts Module

[NEW] Jet Pack - Availability runtime APIs
[NEW] StickyControlModule - Debug is-grounded scene view indicator
[NEW] Jet Pack Jane character animations
[NEW] 11 Gesture animations
[NEW] Sample root motion animations
[NEW] Climbing - in Technical Preview
[NEW] NPC Look At Demo
[NEW] Third person orbit damping option
[NEW] Replace animation clips at runtime (sample scripts included)
[NEW] Sample Animation Play List runtime script
[FIXED] Jet Pack - IsJetPackEnabled returns true when Jet Pack is not available
[FIXED] Animate - IsJetPacking condition can be true when Jet Pack is not available
[FIXED] StickyInputModule - Custom Input trigger parameters may not respond
[FIXED] Jet Pack Jane running animation
[IMPROVED] Sticky Control Module - Debug Volume rotates with character
[IMPROVED] Animate - improved blend tree transitions

Version 1.0.3 – 29 June 2021

[NEW] 5 In-place sit animation sets
[NEW] 3 additional wave gestures
[NEW] SampleSitAction script
[NEW] Third person camera shake with API
[NEW] Override animations using Sticky Zones
[NEW] Third Person Free Look - In Technical Preview
[NEW] IsObstacle API to do obstacle detection in your own code
[NEW] Custom Anim Actions - Is Reset After option for boolean values
[NEW] Faction and Model IDs for character identification
[NEW] Sticky Zones can filter characters by Faction or Model ID
[NEW] General purpose Proximity component
[FIXED] Character stability on moving objects when move update type is Fixed Update
[IMPROVED] Show or Hide Cursor on initialise
[IMPROVED] Sample Look At Player - avoid compound collider issue
[IMPROVED] Sample NPC Play Anim - avoid compound collider issue
[IMPROVED] Head IK - option to follow a target when movement is disabled

Version 1.0.4 – 31 August 2021

[NEW] First Person zoom
[NEW] Hand IK with API - in Technical Preview
[NEW] Look Interactive with API - in Technical Preview
[NEW] Sticky Interactive objects with API - in Technical Preview
[NEW] Head IK has optional move damping
[NEW] First Person Free Look - in Technical Preview
[NEW] Animate - option to invert bool animation values
[NEW] Animate - option to toggle custom bool animation values
[NEW] Display information like messages or gauges in a UI for the player
[NEW] Sample script to show how to add custom inputs at runtime in your code
[FIXED] DisableLook and DisableCharacter APIs do not disable look movement
[FIXED] Animations may continue to run when DisableCharacter is called
[FIXED] Third Person Free Look pitch up and down limits are reversed
[FIXED] Auto Hide Cursor should have no effect when Look is not enabled
[FIXED] Button custom inputs should not update Animate data unless button is pressed
[IMPROVED] Create Proximity gameobjects from 3D Object menu in the Unity editor

[IMPROVED] Disable Auto-hide Cursor when the character is disabled

Version 1.0.5 - 14 September 2021

[NEW] (Un)PauseCharacter API methods for use with game pause options
[NEW] StickyInteractive - Auto Unselect option
[NEW] Head IK has option to adjust for velocity of target and character
[IMPROVED] StickyInteractive - unused events are hidden in the inspector
[IMPROVED] StickyPartsModule - more API methods

Version 1.0.6 – 10 December 2021

[NEW] Unity XR input support for VR - in Technical Preview
[NEW] Look VR mode including snap turn - in Technical Preview
[NEW] Sticky XR Interactor – in Technical Preview
[NEW] Animate - Hand VR - in Technical Preview
[NEW] Animate - Head IK - Look at Interactive (while idle)
[NEW] Third Person clip object responsiveness
[NEW] StickyInteractive - activate and deactivate events with API
[IMPROVED] StickyInteractive is out of Technical Preview
[IMPROVED] Look Interactive is out of Technical Preview
[IMPROVED] Collision detection for attached objects

Version 1.0.7 – 23 December 2021

[FIXED] DisableHeadIK may not disable Head IK
[FIXED] Head IK - may look up unexpectedly
[IMPROVED] Head IK movement algorithm

Version 1.0.8 – 18 January 2022

[NEW] StickyInteractive - Is Readable option for virtual levers
[NEW] Animation API to play states with an offset from the beginning
[NEW] Blend in and out animator layers via API at runtime
[FIXED] 3rd person Free Look vertical movement glitches
[FIXED] Look Horizontal Damping has no effect
[FIXED] Hand Interact sample - displayMessage is null when Show Text disabled
[FIXED] NullReferenceException when Grabbing an object that is not Touchable
[FIXED] Reoccurring "Animator is not playing the AnimatorController" warning
[IMPROVED] 1st and 3rd person Free Look is out of Technical Preview
[IMPROVED] Include parameter name in Sticky Input linked Animation Actions