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 single-player 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, equip, socket, stash, activate, read, sit on, and/or select interactive objects
- Interactive objects, including weapons and magazines, can react to custom gravity
- Replace animation clips at runtime
- Pooling system for generic objects
- Popup menus for interactive-enabled objects
- Supports first-person Virtual Reality (continuous move, snap turn, teleportation and interaction)
- Use hand-held projectile and laser-beam weapons
- Display facial emotions and sync visemes to audio speech if model has blendshapes

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 currently.

- 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 full game-ready weapon system (there many animated and non-animated weapons on the Asset Store and our weapon system is extendable)
- It doesn’t include a full 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 include multi-player functionality. Although technically possible, it would take considerable effort for you to write code and make modifications to create a multi-player game.
- We do not include any melee animations, components, or weapons. You could “probably” configure melee with your own animation controller and send data to it using the Animate tab on a S3D character.
- We do not create the blendshapes on your model for use with our Shapes Module.
- We do not generate the speech audio file. i.e., this is not a Text-To-Speech (TTS) service.
- 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 follow a sliding window which roughly matches the versions supported by Unity. We currently support Unity 2020.3.25+, 2021.x, 2022.x, and 6000.0. We recommend using Unity Editor LTS or Release versions.

Contents

Getting Started

S3D comes bundled with several demo scenes and prefabs to get you started quickly. The demo scenes are set up with a character controller and (legacy) 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., PlayerAmy), 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).

Support Policy

For free support we will investigate reproducible bugs in our code. We may ask you to provide a simple scene with clear instructions on how to repro the issue. We can provide an upload area for the project files.

If this issue is critical to an announced game release date, we give it high priority. We also help with fleshing out new features that can improve gameplay and that we could add to a new version. In addition, we offer customer support for discovering existing features and how to configure them, both in our Unity Forum or on our Discord Channels.

To add "polish" to a game or general help with implementing our products (and even writing custom game-play code) we negotiate a flexible hourly rate which can be time-boxed to fit the studio or indie budget.

You can back us on Patreon (https://www.patreon.com/scsmmedia) to help us improve our products.

We may alter this support policy from time to time without notice.

What’s Changed

Version 1.1.5

[NEW] StickyInputModule - override axis option
[NEW] StickyUIRaycaster - for popups on moving objects
[FIXED] Sitting - Assertion failed on expression: IsNormalized(direction)
[IMPROVED] Sitting obstacle detection
[IMPROVED] Get Support button opens Unity Discussions

Version 1.1.4

[NEW] SSCOutputBridge
[NEW] Support for Meta XR Core in VR
[FIXED] SSCInputBridge.IsInitialised always returns false
[FIXED] StickyZone - Enable Colliders option missing in editor
[FIXED] StickyInteractive - disable rbody gravity if not in use for Unity 6+
[FIXED] StickyInteractive - readable position unstable with sudden movement
[IMPROVED] StickyInteractive - ReInitialiseReadable() can be called multiple times
[IMPROVED] SSCInputBridge APIs
[IMPROVED] Compatibility with Unity 6

Version 1.1.3

[NEW] Option to manually update Celestials from Sci-Fi Ship Controller
[NEW] Set temporary reference frame APIs
[NEW] SampleSitActionPopup works with SSC Multi-stage Seat animator
[NEW] SampleSitActioNPC works with SSC Multi-stage Seat animator
[NEW] StickyZone - option to enable colliders on initialise
[FIXED] Toggle Third Person at runtime in editor on un-initialised causes null reference
[FIXED] NullReference when removing a Speech Audio clip in Sticky Shapes Module
[IMPROVED] Sticky Shapes Module - add speech audio at runtime
[IMPROVED] Third Person focus position when sitting
[IMPROVED] Hand IK when player is sitting

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

Videos and Tutorials

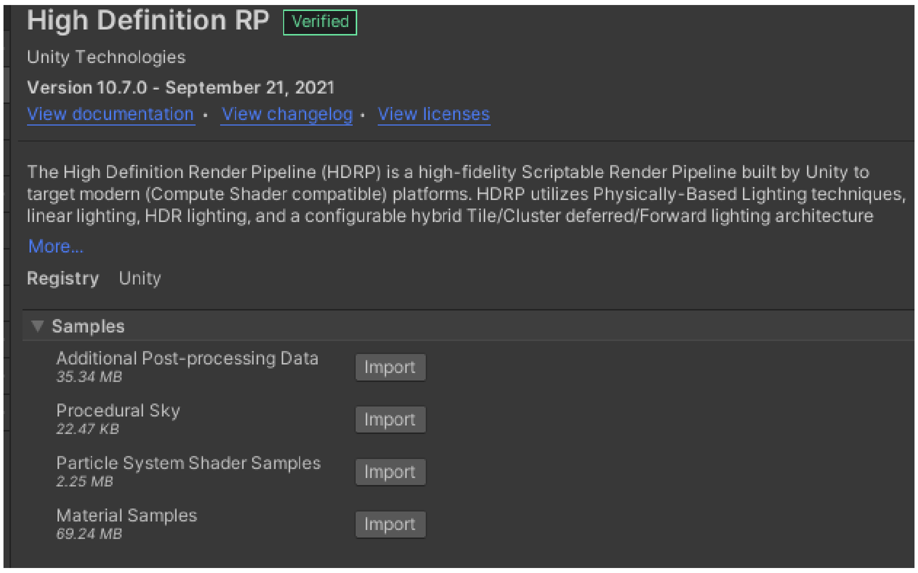
Name	URL
Sticky3D – Get Started Tutorial	https://youtu.be/XGPzSQ61oMM
Reference Frames Tutorial	https://youtu.be/LwC0AQa7AcM
Add Your Model Tutorial	https://youtu.be/w7o17d7hcpc
Mixamo[1] Tutorial	https://youtu.be/an9qFX-i8xI
Sit Tutorial	https://youtu.be/UxlyJCFiuTQ
Interactive Basics Tutorial	https://youtu.be/azgpsSK-_D8
Find Interactive Objects Tutorial	https://youtu.be/HOMX7WncYC0
Holster Weapon Tutorial	https://youtu.be/tuZAEi7XVTo
Socket Basics Tutorial	https://youtu.be/D5ibiYZv1D0
Blendshape Basics Tutorial	https://youtu.be/ZHcH-oCrNSc
VR Flight Setup with SSC + S3D	https://youtu.be/m1OdRGvDOS0
Short object Interaction video	https://youtu.be/ol4r7MteVCc
Promo 5 Trailer	https://youtu.be/ZiTZ0OObk9U
Integration with SSC Seat Animator	https://youtu.be/-JzLLFnwmtY

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

HDRP and URP Scriptable Render Pipelines

When using HDRP and URP with our demo assets, you will need to apply the SRP packages included in the SCSM\Sticky3DController\SRP folder. This folder contains a readme.txt file which you should check out before proceeding.

For HDRP, we use Legacy particle shaders. This is because, out-of-the-box, HDRP unfortunately doesn't include any. You can, however, download them from the Package Manager and modify our demo particle materials.



High Definition RP Verified

Unity Technologies

Version 10.7.0 - September 21, 2021

View documentation · View changelog · View licenses

The High Definition Render Pipeline (HDRP) is a high-fidelity Scriptable Render Pipeline built by Unity to target modern (Compute Shader compatible) platforms. HDRP utilizes Physically-Based Lighting techniques, linear lighting, HDR lighting, and a configurable hybrid Tile/Cluster deferred/Forward lighting architecture

More...

Registry Unity

Samples

Additional Post-processing Data	35.34 MB	Import
Procedural Sky	22.47 KB	Import
Particle System Shader Samples	2.25 MB	Import
Material Samples	69.24 MB	Import

In your versions of HDRP and URP, you may also wish to tune the lighting to your taste in the various demo scenes. Lighting can change between different versions of the render pipelines. This is a Unity thing and outside our control.

Demos

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

Our demo systems were created for Built-in Render Pipeline (BiRP). If using URP or HDRP, see the "S3D_SRP_Readme" text file in the SRP folder and apply the correct SRP package.

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.

Demo Characters

We have included several pre-configured character prefabs. The older characters (Jane, Bob, and Rod) are placeholder characters with various configuration. The newer characters (Amy, Bryce, Kate, and Jeff) are higher quality game-ready characters with more polished animations. The later also include facial emotions and make use of our Sticky Shapes module for controlling blendshapes.

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.

NOTE: This demo does not use the new Weapon System which was added in version 1.1.0.

NavMesh Demo

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

You can pick up a weapon by looking at it, then when the reticle goes green, indicating it as an interactive-enabled object, press the F key. To drop the weapon, press F again. To fire the weapon, use the left mouse button. Right mouse button will toggle aim on/off.

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. The NPC JaneRM character has a child transform, called “Proximity”, which includes the “Sample Look At Player” component and a sphere collider which allows it to look at, and walk toward the player.

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.

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.
Walk Speed	The speed at which the character can walk in metres per second. If you have a walk animation, you can also head over to the Animate tab settings to send the move speed to your animation using Animate Actions. If the feet slide forward, try increasing the Walk Speed. If the feet lag behind the characters forward motion, decrease the Walk Speed. If you think the character should be moving at this speed, you can adjust the Animate Actions. The average walk speed is 1.43m/s (male), 1.34m/s (female).
Sprint Speed	The speed at which the character can sprint or run in metres per second. The average run speed is 2.64m/s (male), 2.23m/s (female).
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)
Allow Sprint Backward	Can the character sprint or run backward?
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.

Property (General)	Description
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. See the “Sticky Surface” chapter for more details.

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
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.

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, if 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
Max LoS Field-of-View	Maximum line-of-sight field-of-view is the angular range that objects are in sight of the character.
SSC Update Celestials	If Sci-Fi Ship Controller is installed and the Celestials (stars) are in the scene, have Sticky3D rotate the stars to match the S3D camera. This can overcome an issue where the stars lag the S3D camera. On the SSC celestials component, ensure Timing Type is set to Manual.

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

Property (First Person)	Description
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

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 or use StickyInteractiveChild components. See the chapter called “Sticky Interactive” for more details.

To look at and interact with Interactive-enabled objects in VR, see the chapter called “Sticky XR Interactor”.

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.
Lock to Camera	Instead of using the mouse or cursor position, always look in the direction the camera is facing.

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 - Overview" 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 in the given posture.
Room Scale	Enable the character to move around when the Head Mounted Device moves.
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".

Look - Sockets

This allows the character to see or detect objects with a StickySocket component. See the "Sticky Socket" chapter for more information.

Property	Description
Layer Mask	The trigger colliders in these Unity layers can be seen when Look Sockets is enabled.

Property	Description
Max Distance	When Look Sockets is enabled, how far away from the camera can the character see objects with a StickySocket component? Try to keep this distance as short as possible to avoid hitting too many colliders in front of the camera.
Lock to Camera	Instead of using the mouse or cursor position, always look in the direction the camera is facing.
Active Material	The (custom) material used to highlight a socket in the scene. If not set, one will be created at runtime.
Auto Show	Attempt to automatically show (and hide) the highlighter when looking a socket

Look - Zoom

This allows the first or third-person camera to zoom in or out, making objects appear closer or further away.

Property	Description
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
First Person Only	
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.
Third Person Only	
Zoom Out Factor	The relative amount the camera can zoom out. [Default 1]

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
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. NOTE: Reference Update Type must be Automatic First or Manual.
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

Property	Description
	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 Ref Frame	Initial or default reference frame transform the character will stick to.
Use RBody Ref Frame	When Reference Update Type is Automatic or AutoFirst, when detecting a collider that is attached to or a child of a rigidbody, the reference frame will be set to the transform of the rigidbody rather than the transform of the collider.

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

Property	Description
	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 - Overview

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.
Aim IK	Aim Inverse Kinematics (or Aim IK for short), is used when holding weapons. It helps the weapon face the potential target. See the "Animate – Aim IK" section below for details.
Head IK	Head Inverse Kinematics (or Head IK for short), helps make a humanoid character's head look towards a target position or object in the scene. See the "Animate –Head IK" section below for details.
Hand IK	Hand Inverse Kinematics (or Hand IK for short), helps move a humanoid character's hand towards a target position or object in the scene. See the "Animate – Hand IK" section below for details.
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.
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.
Hand VR	This helps to animate hands when using Virtual Reality. See the "Animate – Hand VR" section below for details.
Actions	Actions send conditional data to your character animation controller. These enable you to animate your character without writing code.
+ 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.

Animate – Actions

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 with in-place animations, you might need to tell your character it is Grounded AND it should have a MoveSpeed. This would be done using two Animate Actions.

To get started, take a look at the animation controllers for the PlayerAmy or PlayerKate characters, and then at the Animation Actions created for those characters. These can be located in the Demos\Prefabs\Characters folder.

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 – let us know if you have a use-case and we can add support for it!
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

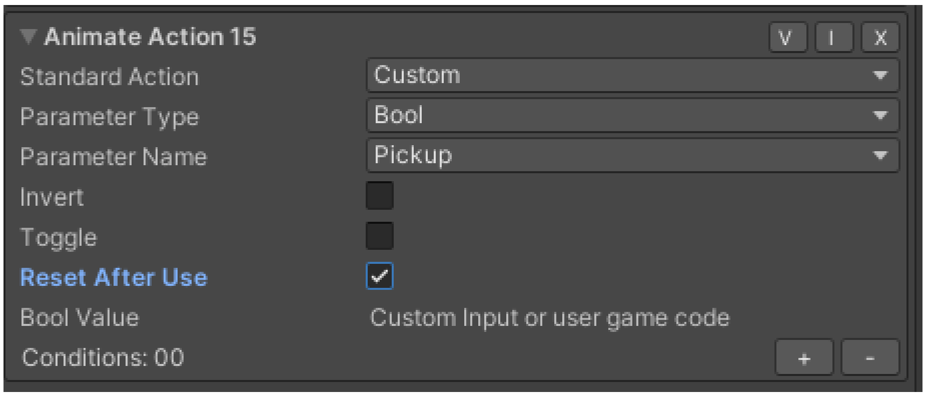
Conditions can be helpful to restrict what data gets sent each frame. Some conditions like HasLanded and HasJumped can help to identify a specific frame in which an animation should start. These two examples only occur on the frame the event happened rather than something like IsGrounded which typically 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

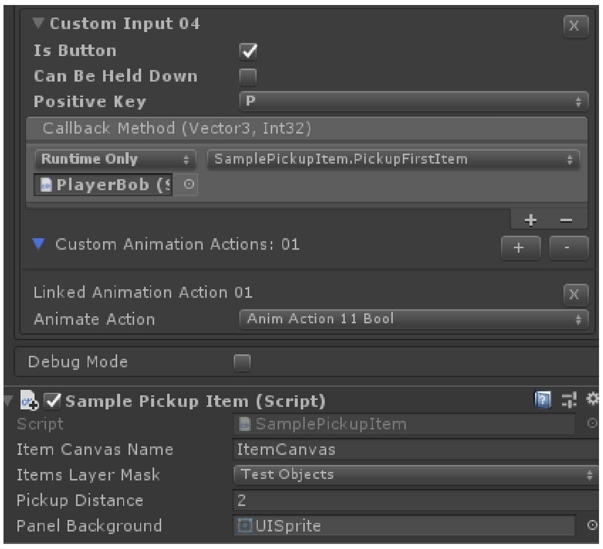
Sometimes you may wish to animate your character when the user presses a particular button on the gamepad or the keyboard. At the same time, you may want to perform some kind of action in the game like pick up or drop an object, or maybe throw an object like a spear or javelin.

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 15

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



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

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.

Animate – Aim IK

Aim Inverse Kinematics (or Aim IK for short), helps rotate the character, and the bones of the character, to aim a held weapon at a potential target.

In First Person for a player character, Hand IK will also be used to adjust the hand positions to match the weapon hand hold positions.

Property or Button	Description
When Not Aiming	If a weapon is held, will it always attempt to face the target using Aim IK? Otherwise, it will only do this when weapon IsAiming is true.
Aim Camera Offset TPS	The camera offset or distance from the character when in third person weapon aiming mode.
Anim IK Pass Layer	The zero-based layer in the animator controller that has the IK Pass enabled for Aim IK. If this layer (not layer 0) has an avatar mask for fingers, it needs to be above the layer than includes the character aim animation. e.g., aim ik layer = 1 and layer with animations is 2.
Aim Down Limit	The maximum rotation, in degrees, each bone for Aim IK can pitch down.
Aim Up Limit	The maximum rotation, in degrees, each bone for Aim IK can pitch up.
Aim Left Limit	The maximum rotation, in degrees, the held weapon can point left.
Aim Right Limit	The maximum rotation, in degrees, the held weapon can point right.
Aim Turn-Look Rate	How quickly, in degrees per second, the character will attempt to turn (or look) toward the aim target.
Bone Weight FPS	The overall bone weight when holding a weapon in first-person. Adjust this if the bones pitch up or down at a different rate to the first-person camera. This can be most noticeable when holding a weapon but not aiming it and Free Look is disabled.
Aim Bones	The (spine) bones used when aiming at a target. Add at least one of the spine bones (E.g., Spine, Chest, Upper Chest, Neck) to help a held weapon point toward where the character is looking. Adjust the weight of each bone to determine how much it influences where the weapon is pointing. If holding a weapon with both hands, it can be helpful to add say a LeftShoulder bone with "Is Paired" enabled, and to give that the majority of weight (say up to 80% of the overall weight). Shoulders can make the bone animation look more natural. Currently, shoulders are ignored in First Person.

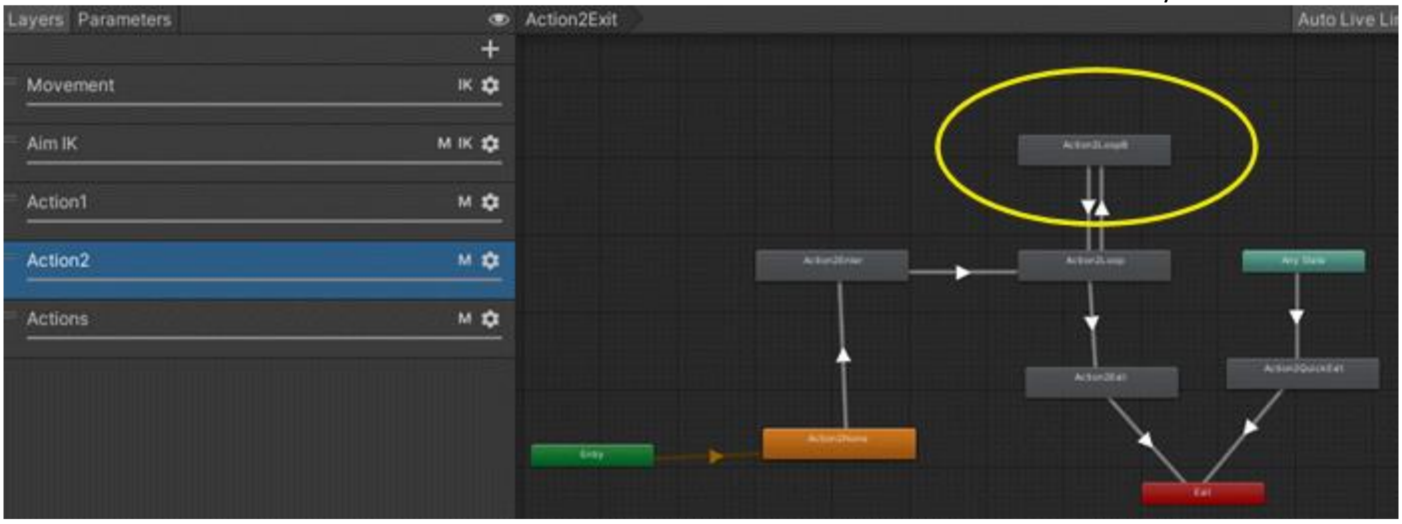
Aim IK First Person Setup

In first person, the hands hold the weapon using some of the character Hand IK settings and the weapon Interactive Hand Hold positions. You should read the "Animate – Hand IK" section to be become familiar with how Hand IK works.

You could set the Aim IK "Pass Layer" to the same as the one used in "Hand IK" or you could choose a different layer if say you wish to retain the finger rotations set in your weapon-based character animations. For more information see "Sticky Weapon – Animate Properties". Character animations for a weapon are set in "Weapon Anim Sets".

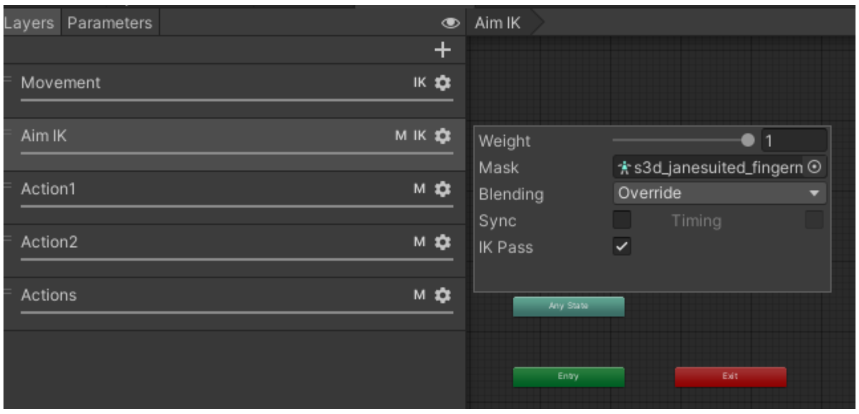
Say you have created an animation for your character holding a particular weapon. See Demos\Animations\ s3d_jane_suited_ik.controller. You have an animation that plays when the weapon is being held or is aiming. This animation includes finger position and rotations so that the fingers are gripping the weapon. E.g., s3d_jane_suited_p320_aim1.

Typically, when Hand IK is used, it changes the position and rotation of the hands. However, it also (by default) sets the fingers back to their default orientation. To resolve this, we need to create a separate animator layer (we've called this "Aim IK", in the example given).



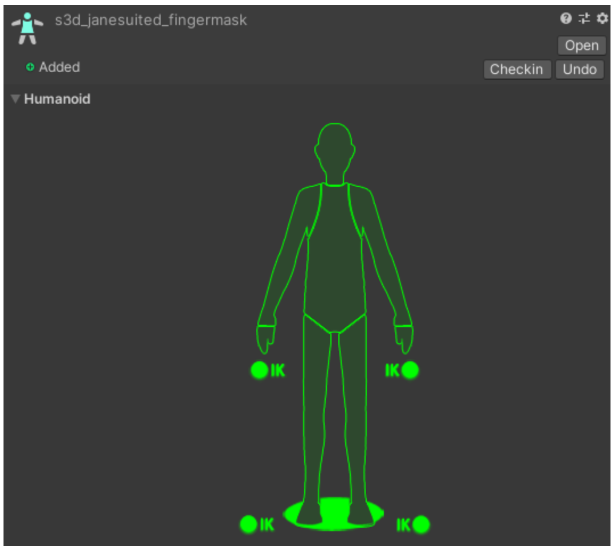
We enable the “IK Pass” property, and we assign an Avatar Mask to not update the fingers when the Hand IK is applied.

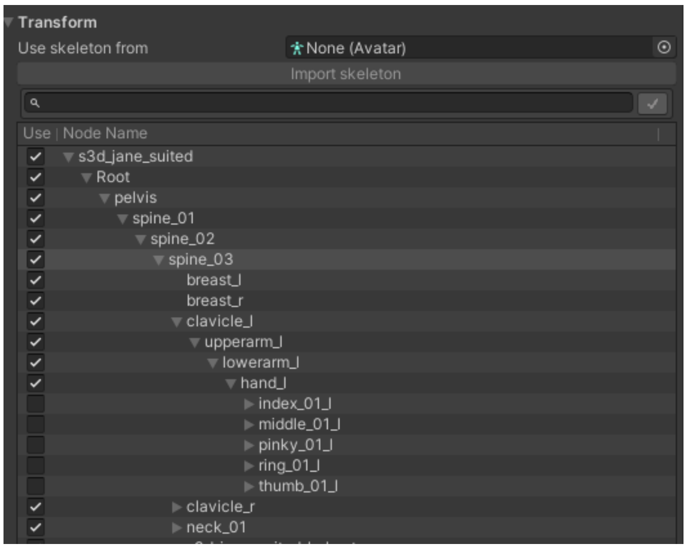
This enables us the position the hands correctly to grip the weapon (while aiming in first person), and allow our aim animation to correctly place the fingers.



Weight 1
Mask s3d_janesuited_fingern
Blending Override
Sync ☐ Timing ☐
IK Pass ✓

In the standard “Humanoid” Avatar Mask settings, only the whole hands can be disabled which isn’t particularly helpful. However, we can use the “Transform” settings to turn off just the fingers on each hand. This way, when our Aim IK runs on the “Aim IK” layer we created, the finger rotations will not be changed when the hands are rotated to target or grab the weapon hand hold positions.





Use skeleton from None (Avatar)

Use	Node Name
✓	s3d_jane_suited
✓	Root
✓	pelvis
✓	spine_01
✓	spine_02
✓	spine_03
✓	breast_l
✓	breast_r
✓	clavicle_l
✓	upperarm_l
✓	lowerarm_l
✓	hand_l
☐	index_01_l
☐	middle_01_l
☐	pinky_01_l
☐	ring_01_l
☐	thumb_01_l
✓	clavicle_r
✓	neck_01

For more information on Avatar Masks see:

https://docs.unity3d.com/Manual/class-AvatarMask.html

Animate – Head IK

Head Inverse Kinematics (or Head IK for short), helps make 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 with an IK Pass layer.

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.

NOTE: If you want the head to follow the mouse point or HUD reticle (when say "Free Look" is enabled under "General Look Settings" on the "Look tab), you should:

1) On the "Look" tab, enable "Interactive" and "Update Looking Point"
2) On the "Animate" tab, enable "Head IK" and "Look at Interactive"

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. Has no effect when aiming a weapon.
Inward Limit (right)	The Hand IK right hand inward movement (to the left) limit in degrees. Has no effect when aiming a weapon.
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. Has no effect when aiming a weapon.
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. Has no effect when aiming a weapon.
Max Reach Dist.	The maximum distance the hand will attempt to reach for an object.
Elbow Hint	These should be at approximately elbow height to the left (LeftHand) or right (RightHand) of the character. An x value closer to the character will tuck the elbow closer to the character body. Z should be local space 0 or slightly in front of the spine. The transform can be attached to one of the spine bones in the prefab. At runtime, you can move the transform to get the optimal position, copy the component position, then past it back into the prefab out of play mode.

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

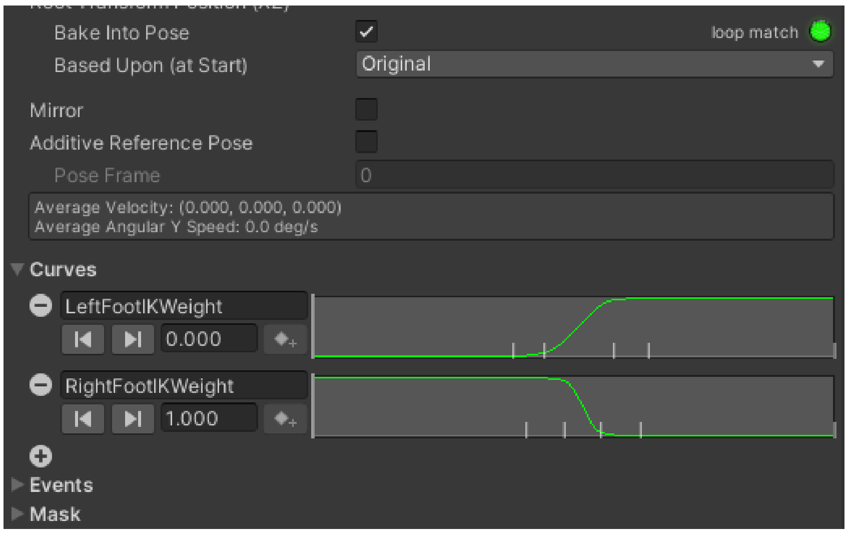
Foot Inverse Kinematics (or Foot IK for short), helps place a humanoid character's feet on the ground. The character must be rigged as a Humanoid character and must include an animation controller and the appropriate animations (idle, walk, sprint etc).

This feature uses Unity's in-built IK system. A side effect of this, is that the knees may be slightly bent even when the legs should be straight in the animation. You can observe this behaviour by placing a character (e.g., PlayerAmy) on a flat surface and observing them in idle mode. Then toggle Foot IK on/off in the editor.

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.



Bake Into Pose ✓ loop match ●
Based Upon (at Start) Original
Mirror
Additive Reference Pose
Pose Frame 0
Average Velocity: (0.000, 0.000, 0.000)
Average Angular Y Speed: 0.0 deg/s
Curves
LeftFootIKWeight 0.000
RightFootIKWeight 1.000
Events
Mask

You will probably want some sloped transition between grounded and not-grounded feet to reduce the effect of "popping".

To copy curves from one animation to another, right click on the animation curve, and click "Copy". Then in the other animation, create the curve and right-click and "Paste" to update it.

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.
Max Foot Pitch	The maximum rotation, in degrees, the foot can pitch forward or back.
Slope Tolerance	The minimum slope that is required before the feet will be rotated to match the slope under the character. This can allow the character to use foot rotations from the animation when standing or moving on relatively flat surfaces.
Toe Distance	The distance, in metres in the forward direction, from the foot bone to the end of the toes or the front of the foot.

Property or Button	Description
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 – Ragdoll

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 (Ragdoll) chapter below.

A ragdoll is often used when you want your character to collapse, like when it's health decreases to 0. Ragdolls require a Unity Animator AND a rigged humanoid model.

To add ragdoll features to your character:

1. Go to the Animate tab on the Sticky Control Module
2. Expand Ragdoll
3. Click "Get Bones" (you should get a list of approximately 15 bones)
4. Check that all the bones match the name of the bone on the left
5. Click "Generate" to add rigidbodies, colliders and joints to the correct bones (they can be removed again by clicking the "Remove" button. NOTE: Remove only works with the list of displayed bones).
6. Run the scene (with the game view not maximized so the inspector is still visible)
7. Click the "Test" button to enable the ragdoll
8. Optionally click the "Test" button again to turn off ragdoll mode.

To enable the ragdoll when the character reaches 0 health, go to the "Engage" tab, expand "Respawn Settings" and click "Ragdoll on Destroy".

Animate – Root Motion

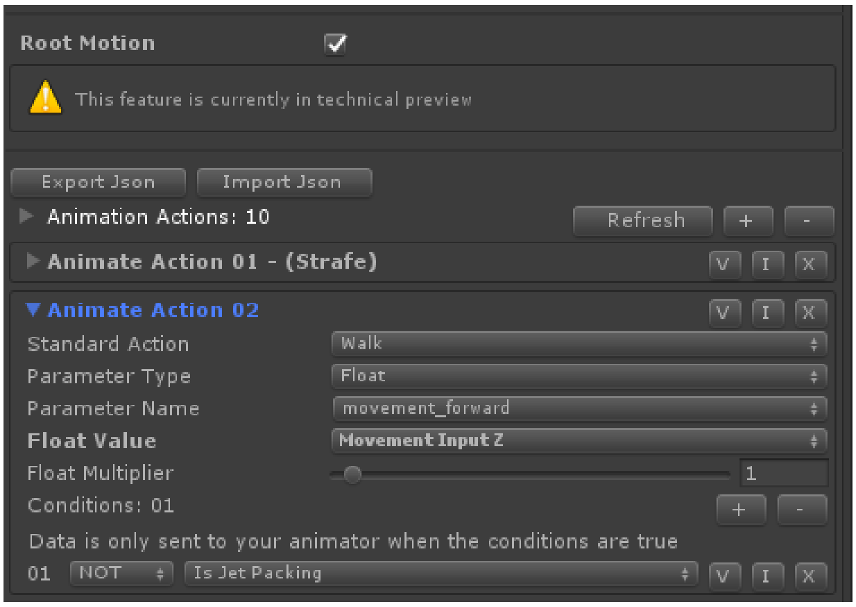
This feature is currently in **Technical Preview** and its behaviour may change without notice with each release to correct issues or to address particular scenarios. If you see a problem, please contact us on our Unity forum or on our Discord channel. See also the Common Issues – Animate (Root Motion)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

Animate Action 02 | V | I | X

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 | V | I | X

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.

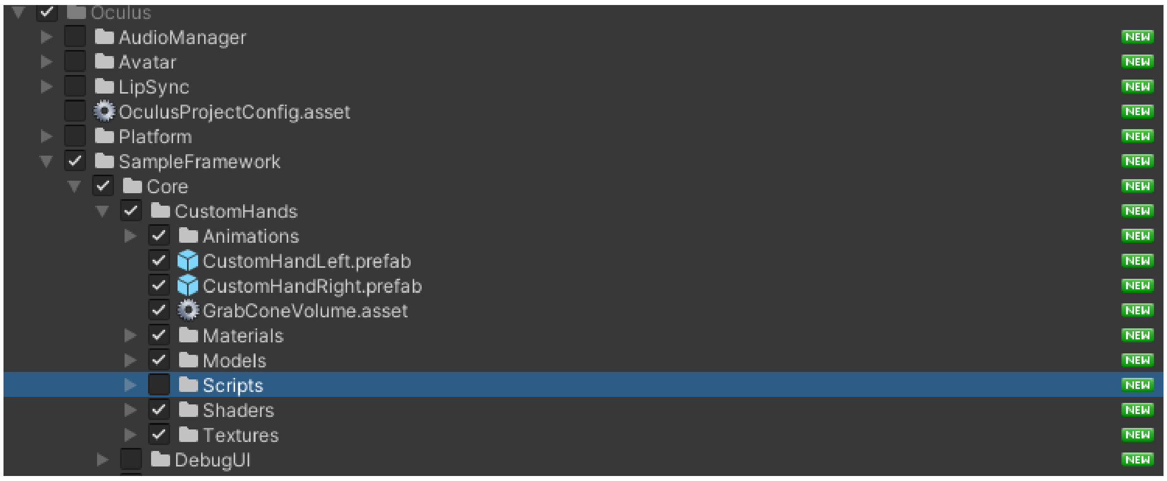
Animate – Hand VR

This feature 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.

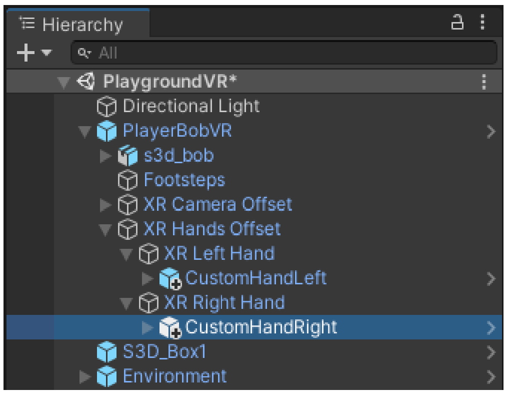
1. Import part of Oculus Integration package from Asset Store (select only SampleFramework, Core, CustomHands folder and untick "Scripts" subfolder). We tested with version 33.0 (30 Sept 2021), 42.0 (20 Jul 2022), 44.0 (4 Oct 2022), and 57.0.1 (10 Oct 2023).

   You can also get this from: https://developer.oculus.com/downloads/package/unity-integration/

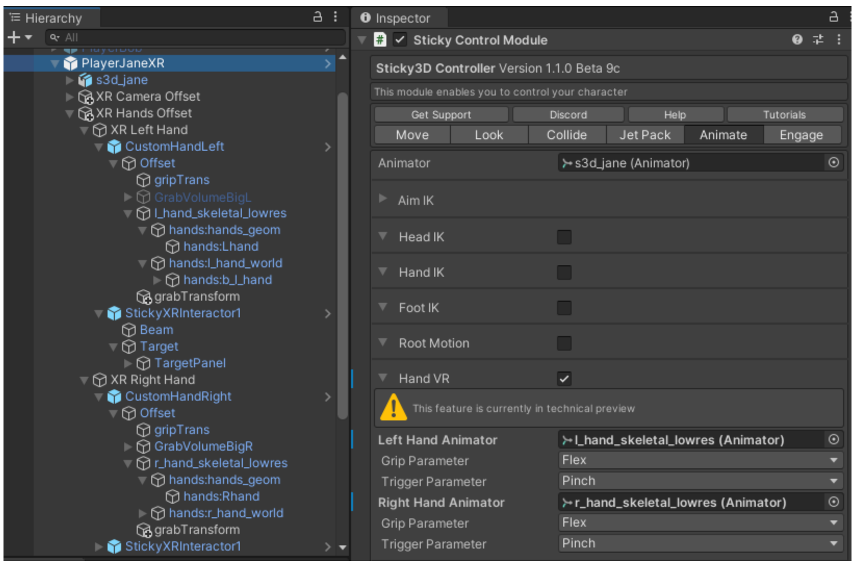


```
✓ Oculus
  ☐ AudioManager                    NEW
  ☐ Avatar                          NEW
  ☐ LipSync                         NEW
  ☐ OculusProjectConfig.asset       NEW
  ☐ Platform                        NEW
  ✓ SampleFramework                 NEW
    ✓ Core                          NEW
      ✓ CustomHands                 NEW
        ✓ Animations                NEW
        ✓ CustomHandLeft.prefab     NEW
        ✓ CustomHandRight.prefab    NEW
        ✓ GrabConeVolume.asset      NEW
        ✓ Materials                 NEW
        ✓ Models                    NEW
        ☐ Scripts                   NEW
        ✓ Shaders                   NEW
        ✓ Textures                  NEW
      ☐ DebugUI                     NEW
```

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 of Sticky3D Control Module, 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”.



Sticky3D Controller Version 1.1.0 Beta 9c

This module enables you to control your character

Animator	s3d_jane (Animator)
Aim IK	
Head IK	☐
Hand IK	☐
Foot IK	☐
Root Motion	☐
Hand VR	✓

This feature is currently in technical preview

Left Hand Animator	l_hand_skeletal_lowres (Animator)
Grip Parameter	Flex
Trigger Parameter	Pinch
Right Hand Animator	r_hand_skeletal_lowres (Animator)
Grip Parameter	Flex
Trigger Parameter	Pinch

Hand VR Property	Description
Left Hand Animator	The animator used to animate a Virtual Reality left hand
Right Hand Animator	The animator used to animate a Virtual Reality right hand
Grip Parameter	The (float) parameter name from the animation controller for the grip action
Trigger Parameter	The (float) parameter name from the animation controller for the trigger action

Engage – General Settings

This tab contains general settings for how the character will engage with other things in the scene that do not apply to the more specific tabs like Move, Look, Collide, Jet Pack, or Animate.

General Property	Description
Character Health	Overall health value of the character. 0 = no health, 100 = full health. Health will affect the speed the character can move.
Max Selectable in Scene	This set the maximum number of Selectable Interactive-enabled objects can be selected by this character at the same time. See also the chapter on "Sticky Interactive".
Engage Colour	The colour associated with engaging with an interactive-enabled object in the scene. Typically used by the StickyDisplayModule reticle when is hovering over an object. See Events, On Look At Changed
Non-engage Colour	The colour associated with not engaging with an interactive-enabled object in the scene. Typically used by the StickyDisplayModule reticle when not hovering over an object. See Events, On Look At Changed
Interactive Tags	The Scriptable Object containing a list of 32 tags. You should only need one per project. They help to identify which interactive objects can be added to particular Equip Points. To create custom tags, in the Project pane, click Create->Sticky3D->Interactive Tags.
Lasso Speed	The rate at which an interactive object is pulled toward or pushed away from a character when interacting with objects. Currently works with socketed objects. When 0, objects are instantly snapped into position.

Engage – Damage Region Settings

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 – Engage chapter below.

Damage regions can define areas of a character that you want to receive individual damage. All characters have a main damage region that applies to the whole "body" of the character.

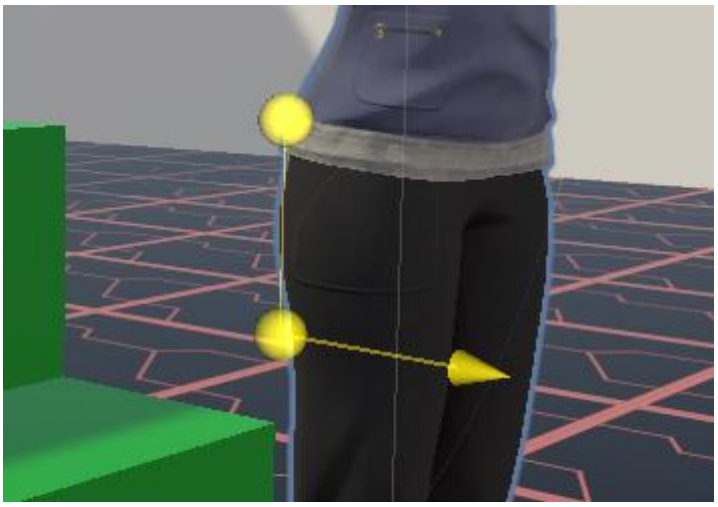
Damage Region Property	Description
Starting Health	The starting health value of this damage region.
Is Invincible	When invincible, it will not take damage however its health can still be manually decreased.
Use Shielding	Whether this damage region uses shielding. Up until a point, shielding protects the damage region from damage. You could also use this to indicate the region has some kind of body armour.
Damage Threshold	Damage below this value will not affect the shield or the damage region's health while the shield is still active (i.e., until the shield has absorbed damage more than or equal to the Shield Amount value from damage events above the damage threshold).
Shield Amount	How much damage the shield can absorb before it ceases to protect the damage region from damage.
Recharge Rate	The rate per second that a shield will recharge (default = 0)
Recharge Delay	The delay, in seconds, between when damage occurs to a shield and it begins to recharge.
Col. Damage Resistance	Value indicating the resistance of the damage region to damage caused by collisions. Increasing this value will decrease the amount of damage caused to the damage region by collisions.
Damage Multipliers	The relative amount of damage a Type A-F projectile or beam will inflict on the character. When the Damage Type is Default, the damage multipliers are ignored. When projectiles (or beams) with different "Damage Types" hit the character, you can apply a relative amount of damage (based on the damage a projectile can normally inflict). For example, if a projectile can normally do 20 health points of

Damage Region Property	Description
	damage, if that projectile had a damage type of say “Damage Type B” and your character is pretty resilient to Type B projectiles, you could set the Damage Type B multiple for the character to say 0.5 (50%). So, when the projectile hits the character, it only does 10 health points of damage (20 points * 0.5 = 10 points of damage).

Engage – Equip Settings

These are used for placing visible inactive interactive objects on the body of a character. Typically, items will be parented to a humanoid bone so that they can move correctly with the character.

Property (Equip)	Description
Min Drop Distance	The minimum distance on the local x or z axis an equipped object can be dropped from the character capsule collider.
Max Drop Distance	The maximum distance on the local x or z axis an equipped object can be dropped from the character capsule collider.
Equip Points	
Name	A description of the location or purpose for the equip point. e.g., right weapon holster
Parent Transform	The transform, typically a bone, that will the objects will be parented to.
Relative Offset	The local space offset from the parent transform.
Relative Rotation	The local space rotation, in Euler angles, from the parent transform.
Max Items	The maximum number of items to equip at this point.
Permitted Tags	The interactive-enabled objects that are permitted to be attached to this Equip Point. See the S3DInteractiveTags scriptableobject under Engage General Settings.



The gizmo in the scene view for the equip point has a small sphere above it which indicates the local space up direction.

The regular Move and Rotate tools can be used to adjust the location and rotation of the equip point.

You can also edit the Relative Offset and Relative Rotation values directly in the Inspector.

Engage – Event Settings

Events are used to take action when certain things occur on the character.

Event Property	Description
On Initialised Event Delay	The number of seconds to delay firing the onInitialised event methods after the S3D character has been initialised.
On Initialised	These are triggered by a S3D character after the character is initialised. This enables you to call your own methods or other S3D API methods after the character is initialised.
On Destroyed	These are triggered when the character is destroyed or when it reaches 0 health.

Event Property	Description
On Look At Changed	These are triggered by a S3D character when they start or stop looking at an interactive-enabled object in the scene. For code-free setup, drag a StickyDisplayModule from the scene into a new event, and set the Function to "Dynamic" StickyDisplayModule.InteractiveLookAtChanged. On the Look tab, "Interactive" must also be enabled. On StickyDisplayModule (if you have a HUD in the scene) consider enabling "Lock Reticle to Cursor" in Display Reticle Settings.
On Pre Start Aim	These are triggered by a S3D character immediately before they start aiming a weapon.
On Post Start Aim	These are triggered by a S3D character immediately after they start aiming a weapon.
On Pre Stop Aim	These are triggered by a S3D character immediately before they stop aiming a weapon.
On Post Stop Aim	These are triggered by a S3D character immediately after they stop aiming a weapon.
On Pre Start Hold Weapon	These are triggered by a S3D character immediately before they start holding a weapon.
On Post Stop Hold Weapon	These are triggered by a S3D character immediately after they stop holding a weapon.
On Respawning	These are triggered when the character is about to be respawned.
On Respawned	These are triggered when the character has just been respawned.

Engage – Identification Settings

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. Characters with the same rig and/or animation controller could all have the same Model ID. For your own models, use numbers above 100 as numbers 1 to 99 are reserved for Sticky3D models.

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

Engage – Respawn Settings

These properties help you determine what happens when your character runs out of health. For example, you might have an NPC robot that is shot multiple times by the player. Does the NPC get automatically respawned after 5 seconds at a pre-determined location or does it simply get destroyed? What happens if it is holding a weapon or has items stashed? There are several options to choose from.

There are also Engage Events (see above) you can configure or callbacks to setup in your own game code (see Runtime and API chapter).

Property	Description
Unlimited Lives	The character never runs out of lives.
Start Lives	The number of the lives the character has when it is initialised.
Max Lives	The maximum number of lives that the character can have.

Respawn Mode	How respawning happens if the character has unlimited or at least 1 spare life.
Respawn Time	How long the respawning process takes (in seconds). Only relevant when Respawn Mode is not set to “Don’t Respawn” OR “Ragdoll on Destroy” is enabled.
Respawn Position	Where the character respawns from in world space when Respawn Mode is set to “Respawn From Specified Position”.
Respawn Rotation	The Euler rotation angles the character respawns from in world space when Respawn Mode is set to “Respawn From Specified Position”.
Drop Held on Destroy	Should the character attempt to drop any held interactive objects when health reaches 0?
Drop Equip on Destroy	Should the character attempt to drop any equipped interactive objects when health reaches 0?
Drop Stash on Destroy	Should the character attempt to drop any stashed interactive objects when health reaches 0?
Ragdoll on Destroy	Should the ragdoll be enabled when health reaches 0?

Engage – Stash Settings

Players and NPCs can stash interactive-enabled items in an invisible “Stash” or inventory. They are stored in the character hierarchy under a parent “Stash Parent” transform.

We have a number of Stash API methods that can be used at runtime in your game to manage adding and removing items from the “Stash”.

Property	Description
Stash Parent	The child Transform under which all items are stashed
Min Drop Distance	The minimum distance on the local x or z axis a stashed object can be dropped from the character capsule collider.
Max Drop Distance	The maximum distance on the local x or z axis a stashed object can be dropped from the character capsule collider.

Engage – Stashing Objects

A player may wish to explore an environment, look at, and “stash” those items. They may also have an item in their hand that they wish to stash (or drop).

To configure a player to look at, then optionally stash it:

1. On the player “Sticky Control Module”, go to the “Sticky Input Module”.
2. Add a new “Custom Input”
3. Configure the button to use to look at and stash interactive-enabled objects. E.g., “L” button on the keyboard
4. Add a new event (Callback Method) and drag the “Sticky Control Module” from the Hierarchy into the “Object” field. For the “Function”, select “StickyControlModule.StashLookedAtInteractiveNoReturn”
5. On the S3D “Look” tab, enable “Interactive”
6. If you have a StickyDisplayModule (a.k.a. “HUD”) in your scene, on the “Engage” tab of the S3D character, expand “Event Settings”, add a new event to “On Look At Changed” and drag the “StickyDisplayModule” from the scene into the “Object” field. Set the “Function” to “StickyDisplayModule “ (Dynamic) InteractiveLookAtChanged.
7. Now your character should be able to walk around, look at a weapon or magazine and grab it into their right hand.

To configure a player to look at, then optionally grab the item (from a distance), then stash it:

1. Ensure the interactive-enabled object is setup up to be Grabbable and Stashable (see the StickyInteractive chapter for more details).

2. On the player “Sticky Control Module”, go to the “Sticky Input Module”.
3. Add a new “Custom Input”
4. Configure the button to use to look at and grab interactive-enabled objects. E.g., “G” button on the keyboard
5. Add a new event (Callback Method) and drag the “Sticky Control Module” from the Hierarchy into the “Object” field. For the “Function”, select “StickyControlModule.ToggleHoldInteractive” (the checkbox would use the left hand)
6. Optionally, you could configure a Custom bool Anim Action with Toggle enabled on the Animate tab to send data to your animation controller to play a “Grab and hold” animation. To send the data whenever the player presses the “G” button, on the same “Custom Input”, add a “Custom Animation Action” and select the “Animate Action” from the drop-down list.
7. On the “Sticky Input Module” add another Custom Input.
8. Configure the button to use to stash the held item. E.g., “I” button on the keyboard
9. Add a new event (Callback Method) and drag the “Sticky Control Module” from the Hierarchy into the “Object” field. For the “Function”, select “StickyControlModule.StashItemFromHand” (the checkbox would stash items in the left hand).
10. If you added the optional “Custom Animation Action” to the (G)rab event above, add it to the (I)nventory event here also so that your “grab and hold” animation is turned off when the item is stashed.
11. On the S3D “Look” tab, enable “Interactive”
12. If you have a StickyDisplayModule (a.k.a. “HUD”) in your scene, on the “Engage” tab of the S3D character, expand “Event Settings”, add a new event to “On Look At Changed” and drag the “StickyDisplayModule” from the scene into the “Object” field. Set the “Function” to “StickyDisplayModule “ (Dynamic) InteractiveLookAtChanged.
13. Now your character should be able to walk around, look at a weapon or magazine and grab it into their right hand.

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.

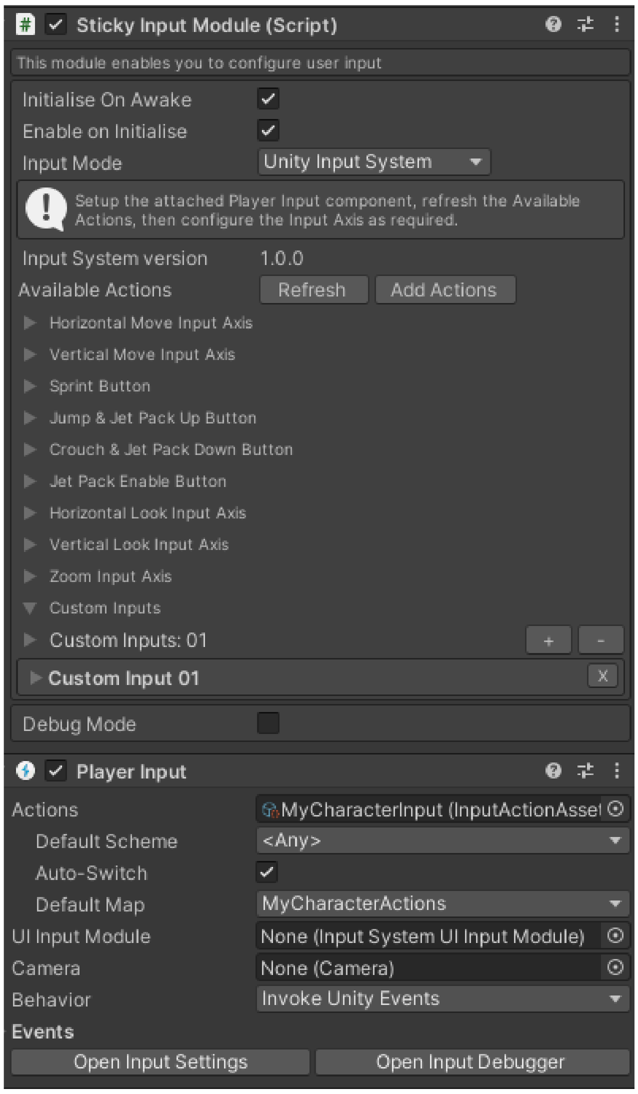
The Sticky Input Module is very flexible and can take input from multiple sources including:

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

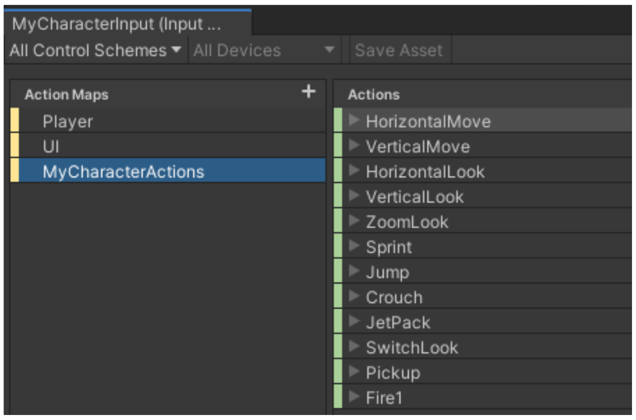
Unity Input System

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 Module component to the S3D character and change the Input Mode to “Unity Input System”.
4. If prompted, click “Fix” to add the UIS Player Input.
5. On the UIS Player Input component, click “Create Actions…”
6. Give it a name and save the asset when prompted (e.g., MyCharacterInput)
7. The asset editor should be displayed. If not, select the new actions asset (e.g., MyCharacterInput) and click “Edit asset”.
8. Add a new Action Map (e.g., MyCharacterActions)
9. Save the asset from the asset editor (and/or close it and save if prompted)
10. 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.
11. On the Sticky Input Module (attached to the same character), 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
Available Actions [Refresh] [Add Actions]
Horizontal Move Input Axis
Vertical Move Input Axis
Sprint Button
Jump & Jet Pack Up Button
Crouch & Jet Pack Down Button
Jet Pack Enable Button
Horizontal Look Input Axis
Vertical Look Input Axis
Zoom Input Axis
Custom Inputs
Custom Inputs: 01
Custom Input 01
Debug Mode ☐
Player Input
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
Events
[Open Input Settings] [Open Input Debugger]

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.


MyCharacterInput (Input ...
All Control Schemes | All Devices | Save Asset
Action Maps: Player, UI, MyCharacterActions
Actions: HorizontalMove, VerticalMove, HorizontalLook, VerticalLook, ZoomLook, Sprint, Jump, Crouch, JetPack, SwitchLook, Pickup, Fire1

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.

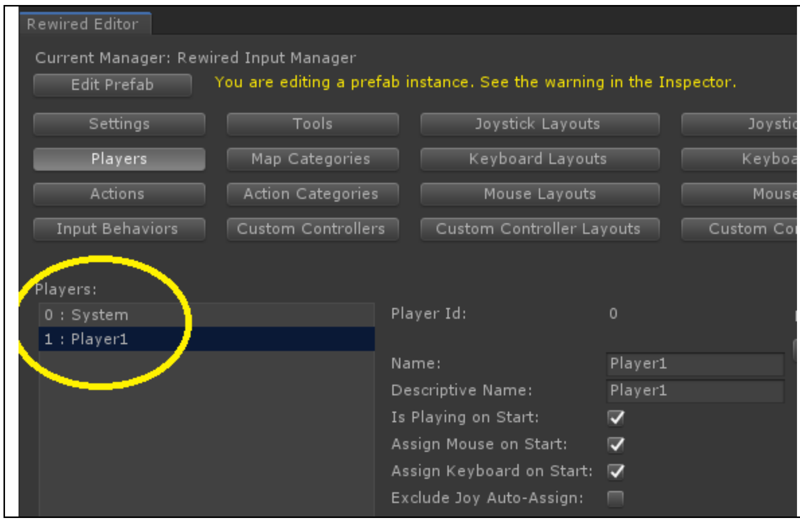
For more information on Rewired see:

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).



Current Manager: Rewired Input Manager

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:

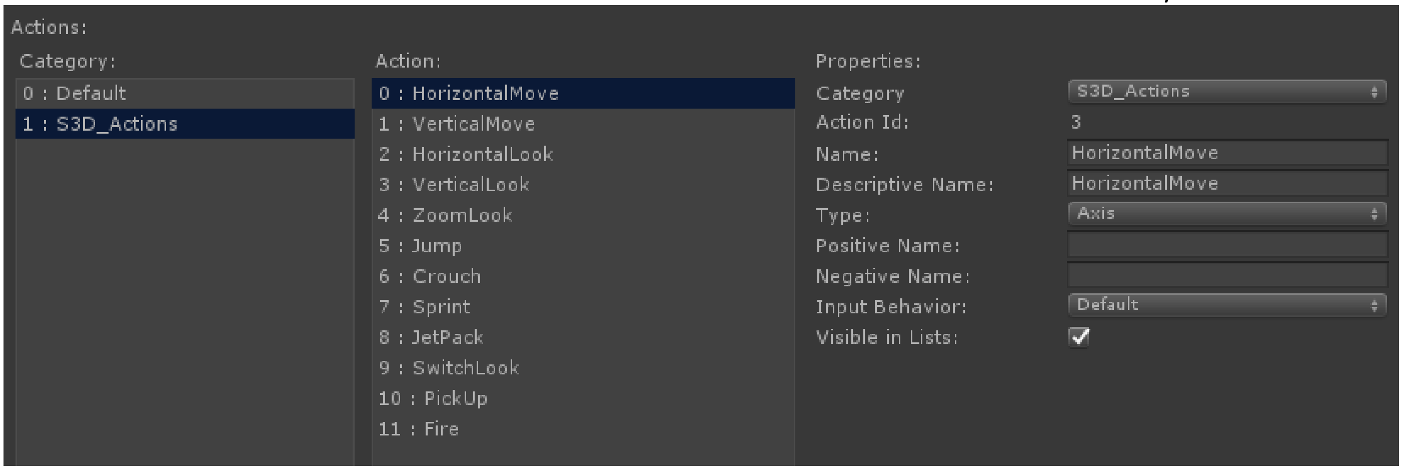
IMPORTANT: Don't forget to create a Player and assign the "Player Number" in the Sticky Input Module.

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

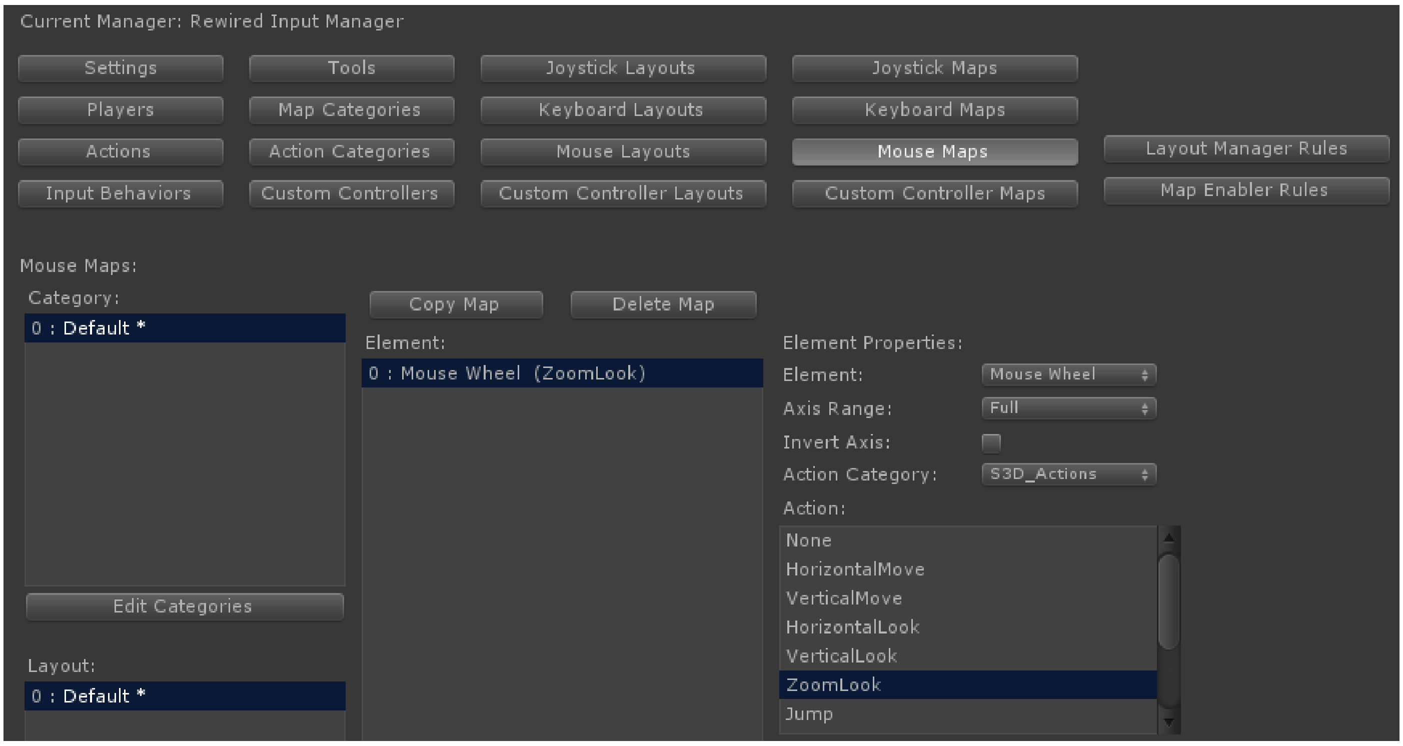
Properties:

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 - Overview

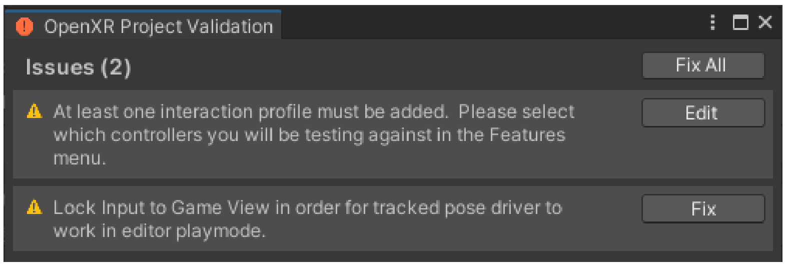
This feature is currently in **Technical Preview** and its behaviour may change without notice with each release to correct issues or to address particular scenarios. If you see a problem, please contact us on our Unity forum or on our Discord channel.

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” (Player Settings) 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 or Meta 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 (or Oculus Open XR).
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 sample “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) Set the LH and RH Offset Rotations to 45, 0, 0.
20) Configure the Input Axis settings in the StickyInputModule editor. See the table below for suggestions.
21) 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”.

22) 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 from Package Manager, 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
HMD Position	XRI HMD	Position	Value	N/A	<XRHMD>/centerEyePosition
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>{RightHand}/primaryButton
Sprint	XR Buttons**	LHandX	Button	Slot1	<XRController>{LeftHand}/primaryButton
Jet Pack Enable	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	<XRController>{RightHand}/triggerPressed

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

Unity XR – Properties

These are the properties available when Unity XR Input Mode is selected.

Property	Description
Input Action Asset	A scriptableobject asset containing Unity Input System action maps and control schemes.
Left Hand	The transform of the XR left hand.
Right Hand	The transform of the XR right hand.
LH Offset Rotation	The left-hand offset rotation, in Euler angles. Used to correct hand controller rotation. E.g., 45 deg on x-axis
RH Offset Rotation	The right-hand offset rotation, in Euler angles. Used to correct hand controller rotation. E.g., 45 deg on x-axis
HMD Position Input Axis	The position input from the VR head-mounted device. This should return a vector3. E.g., <XRHMD>/centerEyePosition
Left Hand Position Input Axis	The position input from the VR left hand controller. This should return a vector3. E.g., <XRController>{LeftHand}/pointerPosition
Left Hand Rotation Input Axis	The rotation input from the VR left hand controller. This should return a quaternion. E.g., <XRController>{LeftHand}/pointerRotation
Right Hand Position Input Axis	The position input from the VR right hand controller. This should return a vector3. E.g., <XRController>{RightHand}/pointerPosition
Right Hand Rotation Input Axis	The rotation input from the VR right hand controller. This should return a quaternion. E.g., <XRController>{RightHand}/pointerRotation
Horizontal Move Input Axis	Left or right movement.
Vertical Move Input Axis	Forward or backward movement.
Sprint Button	Sprint or run
Jump & Jet Pack Up Button	Jump or move up when Jet Pack is enabled.
Crouch & Jet Pack Down Button	Crouch down or move down when the Jet Pack is enabled.
Jet Pack Enable Button	Toggle the Jet Pack on or off.
Look Turn Input Axis	Look or turn left or right. This should return a quaternion. E.g., <XRController>{RightHand}/Primary2DAxis
Look HMD Input Axis	Look in any direction using the VR headset. This should return a quaternion. E.g., <XRHMD>/centerEyeRotation
Switch Look Button	Switch between first- and third-person view.
Zoom Input Axis	Zoom the camera in or out.
Left Fire1 Button	Fire the primary weapon mechanism.
Left Fire2 Button	Fire the secondary weapon mechanism.
Right Fire1 Button	Fire the primary weapon mechanism.
Right Fire2 Button	Fire the secondary weapon mechanism.

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

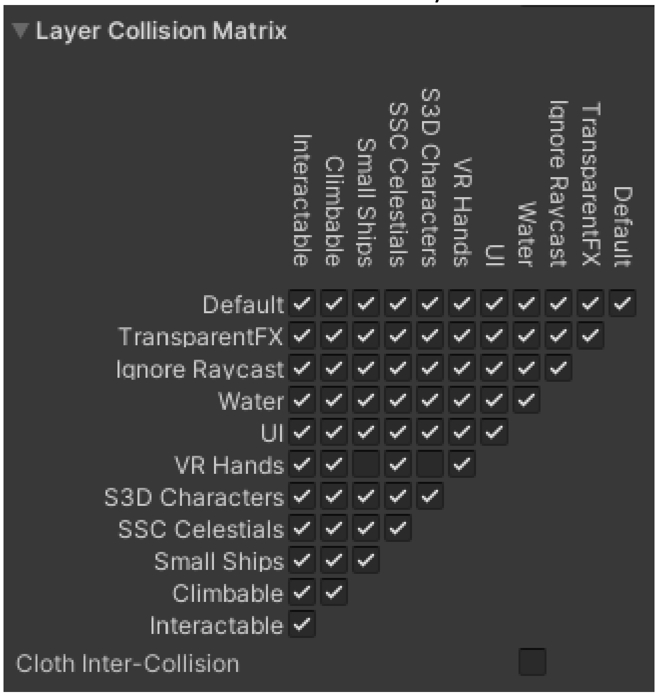
- In Project Settings->Time, set the Fixed Timestep to match your target framerate (0.0138889 = 72 fps, 0.011111 = 90 fps, 0.00833 = 120 fps). You may also need to disable VSync.
- If the above doesn't work, set the Fixed Timestep back to 0.02 and on the Sticky Control Module Move tab, General Move Settings,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

Unity XR – Hands and Physics Collisions

Typically, you don't want your XR hands to collide with the character itself. If the hands have colliders and rigidbodies, they will push the character. That's not what you want.

Essentially, this is what you need to do:

1. Place your player characters into a separate Unity Layer (say “S3D Characters”). On the prefab, change the “Layer” in the Unity Editor to “S3D Characters” and click “Yes, change children”.
2. If you don’t have one already, create a new Unity “Layer” in the Unity Editor called say 20 “VR Hands”.
3. Find “XR Hands Offset” which should be a child of the player character, and change the Layer to “VR Hands”. Again click “Yes, change children” when prompted.
4. If you have a “Stash” gameobject under the character, set that to say the “Default” Layer. Again, click “Yes, change children” on those objects too.
5. As a final step, in the Unity Editor, go to “Project Settings…” and find “Physics”. Now uncheck the intersection between “S3D Characters” and “VR Hands”. This will allow your XR hands to interact with everything in your scene, except the character itself.


Layer Collision Matrix

	Interactable	Climbable	Small Ships	SSC Celestials	S3D Characters	VR Hands	UI	Water	Ignore Raycast	TransparentFX	Default
Default	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
TransparentFX	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Ignore Raycast	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Water	✓	✓	✓	✓	✓	✓	✓	✓			
UI	✓	✓	✓	✓	✓	✓	✓				
VR Hands	✓	✓		✓		✓					
S3D Characters	✓	✓	✓	✓	✓						
SSC Celestials	✓	✓	✓	✓							
Small Ships	✓	✓	✓								
Climbable	✓	✓									
Interactable	✓										

Cloth Inter-Collision ☐

On the hand rigidbodies, assuming they are Kinematic, set the “Collision Detection” to “Continuous Speculative”. Do the same for your interactive-enabled levers and joysticks that are attached to any moving vehicles.

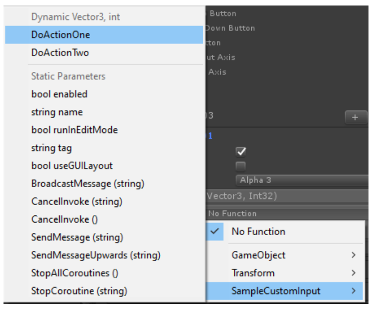
Custom Input

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

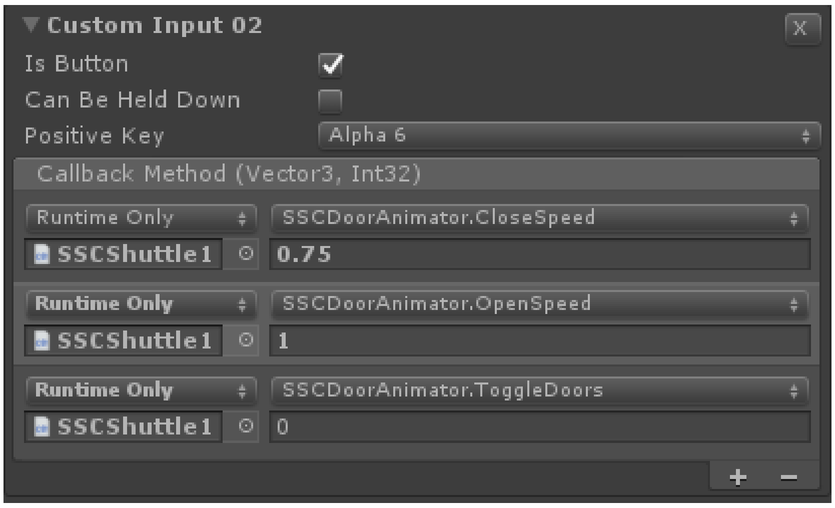
To use this in your own game code:

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 parameters in the Animator Controller.

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

Overriding Player Input Module in Code

Although you can write your own version of an input module and send input to a character with stickyControlModule.SendInput(..), you may wish to do a combination of both. For example, maybe you wish to have the player control everything except forward and backward motion. This could be achieved by overriding the Vertical Move input axis.

In the Sticky Input Module’s editor, you can choose to “Override in Code” each axis. You can also set this value yourself in code. Then in C# code, you could control the forward and back movement. This is demonstrated in the SampleInputOverride.cs script which is included in the Demos\Scripts folder.

Sticky Display Module

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.

Property	Description
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.

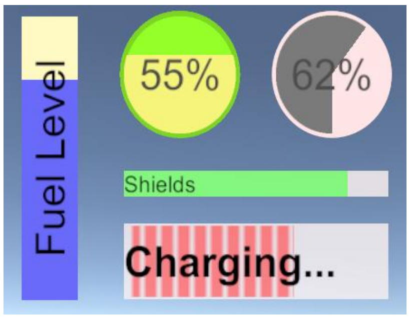
Property	Description
Show Message	Show the message on the display. When a message is created, this is off by design. [Has no effect outside play mode]
Message Name	The name or description of the message. This can be used to identify the message.

Property	Description
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.
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 Anim Replacer

This component can be added to your S3D character (Sticky Control Module), to allow you to replace one or more animation clips at runtime. The component uses S3DAnimClipSets which are ScriptableObjects that are created in your Project.

To create an S3DAnimClipSet:

1. In the Project panel of the Unity Editor, in an Assets folder, Create (right-click or “+” dropdown), Sticky3D -> Anim Clip Set
2. Give the highlighted Anim Clip Set a unique name (e.g., MyCharacter Anim Clip Set1)
3. Add an Anim Clip Pair

4. Find the original animation clip used in your Animator Controller and drop it into the “Original” slot
5. Find the animation clip that you want to replace it with and drop it into the “Replacement” slot
6. Repeat steps 3-6 for any additional pairs you want to replace at the same time.
7. Optionally restrict the Anim Clip Set to only working with one or more characters by adding Model Id to the “Character Model IDs” array.

Each Anim Clip Set represents a set of animation clips that should be replaced at the same time. For some examples of Anim Clips Sets, see the Sticky3DController\Demos\AnimClipSets folder.

To use the Sticky Anim Replacer component, add one or more Anim Clip Sets to the component (there is also an API to add them at runtime if required).

Call the Sticky Anim Replacer API methods from your C# game code, your Visual Scripting engine, or from a Sticky3D Input Module Custom Input.

Eg., ReplaceAnimClipSet(1), RestoreAnimClipSet(1), ToggleAnimClipSet(1).

Sticky Beam Module

Every Weapon that fires a beam requires a prefab with a Sticky Beam Module script attached. For details on using beam weapons see the chapter called “Sticky Weapon” in this manual.

How to Create Beam Prefabs

To create a new Beam prefab:

1. In a scene, using the 3D Object Menu, select Sticky3D Controller -> Sticky Beam
2. Rename the StickyBeam. E.g., MyBeam1
3. Optionally, set the Unity Layer to “Ignore Raycast”, and click “Yes, change children” when prompted. This would allow you to then set weapon scope cameras to not include the “Ignore Raycast” layer in the “Culling Mask”.
4. Create a prefab from the gameobject by dragging the parent gameobject into a folder in the Project pane
5. Reset the prefab parent transform position and rotation to 0,0,0
6. Delete the gameobject from the scene

Sticky Effects Module

This is a poolable particle and/or sound effect used when something is hit, damaged or destroyed. It also has a special “Sound FX” type which can be used with audio clip replacement at runtime.

This component inherits from StickyGenericModule which is managed by StickyManager.

How to Create Effects Prefabs

To create a new Effects Module prefab:

1. In a scene, using the 3D Object Menu, select Sticky3D Controller -> Sticky Effects Object
2. Rename the StickyEffectsObject. E.g., MyEffects1
3. Create a prefab from the gameobject by dragging the parent gameobject into a folder in the Project pane
4. Update the particle system and/or the Audio Source as required.
5. When including an Audio Source, ensure “Play on Awake” is NOT enabled.
6. Reset the prefab parent transform position and rotation to 0,0,0

How to Create Sound FX Prefabs

To create a new Sound FX prefab:

1. In a scene, using the 3D Object Menu, select Sticky3D Controller -> Sticky Sound FX
2. Rename the StickySoundFX. E.g., MySoundFX1

3. Ensure "Play On Awake" is NOT enabled on the Audio Source.
4. Create a prefab from the gameobject by dragging the parent gameobject into a folder in the Project pane
5. Reset the prefab parent transform position and rotation to 0,0,0

Sticky Effects – Properties

Property	Description
Effects Type	Most effects modules will have the Default type. Sound FX is for items that specifically require a Sound FX rather than a general Sticky Effects Module.
Min Pool Size	The starting size of the pool.
Max Pool Size	The maximum allowed size of the pool.
Despawn Time	The object will be automatically despawned after this amount of time (in seconds) has elapsed.
Is Reparented	Does this object get parented to another object when activated? If so, it will be reparented to the pool transform after use.

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, Sittable, Stashable, 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.

For Grabbable objects, you'll need to setup both Hand IK on the S3D character, and the at least the Hand Hold Primary Relative Position on the Interactive object.

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 – General 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.
Priority over Grab	When Is Activable and Is Grabbable is true, Activable will be considered before Grab when EngageLookAtInteractive() is called. This can be helpful when you want to display a popup which could include an option to Grab.
Is Equippable	Can the item be attached to the character body (typically a bone). When “equipped”, interactive-enabled objects become dormant or inactive. The objects remain visible. See also “Is Grabbable” and “Is Stashable”.
Equip Offset	The local space equip position offset from a character equip point.
Equip Rotation	The local space rotation, stored in degrees, around a character equip point.
Equip from Held Delay	The amount of time, in seconds, that a held object will delay being parented to the equip point. This could be used to allow time for say a weapon holster animation to run before the equip operation is completed.
Re-parent on Drop	Attempt to reparent the interactive-enabled object to the original parent Gameobject when it was equipped, grabbed or stashed.
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 equipped, grabbed or stashed.
Disable Regular Colliders	When grabbed, non-trigger colliders on this interactive-enabled object will be disabled. Unless this is a Weapon or Magazine, when dropped, they will NOT be enabled. To enable them add EnableNonTriggerColliders() to OnDropped.
Disable Trigger Colliders	When grabbed, trigger colliders on this interactive-enabled object will be disabled. Unless this is a Weapon or Magazine, when dropped, they will NOT be enabled. To enable them add EnableTriggerColliders() to OnDropped.
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 in-game lever or joystick. See Demos\Prefabs\Props\S3D_Level1 for an example. If you own Sci-Fi Ship Controller and are using VR, check out the SSCInputBridge component.

Property or Button	Description
	The interactive-enabled object must have a kinematic rigidbody attached.
Is Writeable	Can pivot values be updated on the object? This is typically used for rotating a virtual in-game joystick or lever based on some external data. If you own Sci-Fi Ship Controller, check out the SSCOutputBridge component. Writeable objects must also be readable. If your character will be using Hand IK to grip the joystick or lever, the attached kinematic rigidbody on the object may need Interpolate set to "Interpolate" and Collison Detection set to "Continuous Speculative".
Readable Pivot	The point where the readable lever or joystick pivots around.
Dead Zone	The normalised amount the pivot can move before values are updated.
Invert X	Invert the x-axis value (left and right) from the readable pivot.
Invert Z	Invert the z-axis value (forward and backward) from the readable pivot.
Min X (left)	The maximum angle, in degrees, the pivot can rotate left.
Max X (right)	The maximum angle, in degrees, the pivot can rotate right.
Min Z (back)	The maximum angle, in degrees, the pivot can rotate backward.
Max Z (forward)	The maximum angle, in degrees, the pivot can rotate forward.
Auto Recentre	When the readable object is released (dropped) it will automatically return to the centre using the spring mechanism.
Sensitivity X	Speed to move towards target left-right value. Lower values make it less sensitive.
Sensitivity Z	Speed to move towards target forward-back value. Lower values make it less sensitive.
Spring Strength	The strength of the spring used to return the readable pivot back to centre.
Is Selectable	Can this item be selected in the scene? See also Sticky Control Module, Engage tab, Max Selectable in 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?
Sit Target Offset	The local space relative offset that the character should aim for when getting ready to sit down.
Is Seat Allocated	If the object is sittable, is the seat allocated (or reserved)? Typically, used when reserving a seat for a character to sit on.
Is Socketable	Can this item be attached to a StickySocket?
Socket Offset	The local space position offset which will snap to the StickySocket.
Socket Rotation	The local space rotation, stored in degrees, around a StickySocket point.
Is Stashable	Can this item be stashed in the inventory of a character?
Re-parent on Drop	Attempt to reparent the interactive-enabled object to the original parent Gameobject when it was equipped, grabbed or stashed.
Is Touchable	Can this item be touched by a character? Typically used with Hand IK or Sticky XR Interactor.
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.
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.

Property or Button	Description
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.
Popup Relative Up	Use the relative up direction to apply default Popup Offset. This can be useful if you want a popup to appear “above” the interactive object. Only applies when “Use Gravity” is enabled.
Interactive Tags	This is a Scriptable Object containing a list of 32 tags. Typically, you will only need one per project. To create custom tags, in the Project pane, click Create->Sticky3D->Interactive Tags.
Tag	The interactive tag used to determine compatibility with things like StickySockets or character Equip Points.

Sticky Interactive – Gravity Properties

This includes information and settings about how the object responds to gravity.

Property	Description
Use Gravity	The object is affected by gravity
Gravity Mode	The method used to determine in which direction gravity is acting.
Gravity	The gravitational acceleration, in metres per second per second, that acts downward for the object
Gravity Direction	The world space direction that gravity acts upon the object when Gravity Mode is Direction.
Drag	The amount of drag the object has. A solid block of metal would be 0.001, while a feather would be 10.
Angular Drag	The amount of angular drag the object has
Interpolation	The rigidbody interpolation
Collision Detection	The rigidbody collision detection mode
Mass (KG)	The mass of the object in kilograms. Can be useful when the object is dropped.
Initial Reference Frame	Initial or default reference frame transform the object will stick to when Use Gravity is enabled and Gravity Mode is ReferenceFrame.
Inherit Gravity	If gravity is enabled and Gravity Type is Reference Frame, when an object is dropped, it will inherit the reference frame from the character.

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.

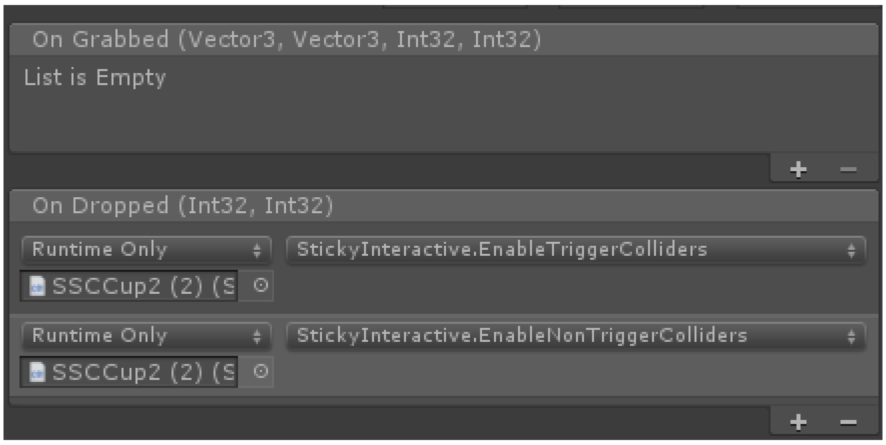
Event Property	Description
On Grabbed	These are triggered by a S3D character when they grab this interactive object.
On Dropped	These are triggered by a S3D character when they drop this interactive object.
On Hover Enter	These are triggered by a S3D character when they start looking at this interactive object.
On Hover Exit	These are triggered by a S3D character when they stop looking at this interactive object.
On Post Equipped	These are triggered by a S3D character immediately after they equip this interactive object.
On Post Grabbed	These are triggered by a S3D character immediately after they grab this interactive object.
On Post Init Event Delay	The number of seconds to delay firing the onPostInitialised event methods after the interactive object has been initialised. This can be useful when other objects in the scene that are used in the On Post Initialised event, are also initialised during Start() and may be called by Unity AFTER this component is initialised.
On Post Initialised	These are triggered immediately after the interactive-enabled object is initialised.
On Post Stashed	These are triggered by a S3D character immediately after they stash (place in inventory) 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”

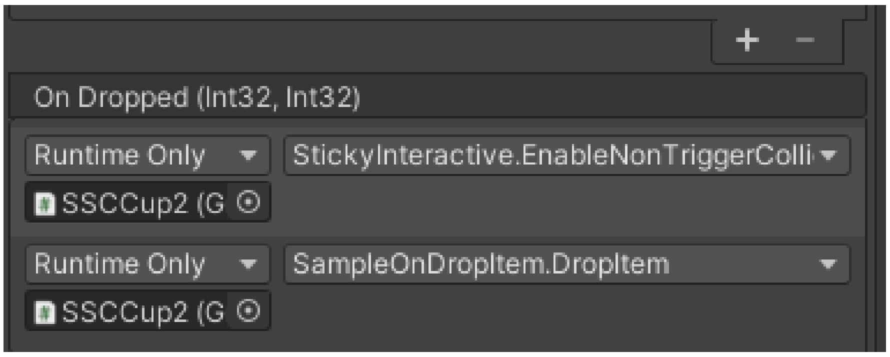

On Grabbed (Vector3, Vector3, Int32, Int32)
List is Empty

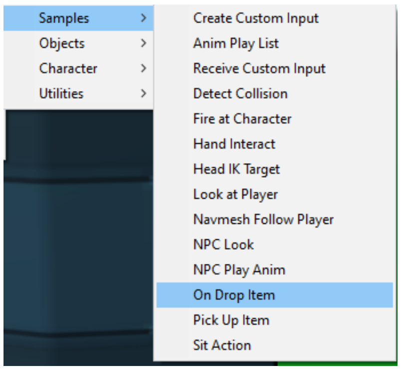
On Dropped (Int32, Int32)
Runtime Only | StickyInteractive.EnableTriggerColliders
SSCCup2 (2) (S
Runtime Only | StickyInteractive.EnableNonTriggerColliders
SSCCup2 (2) (S

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 – Looking at Objects

It is helpful to give players visual feedback when looking at interactive objects in the scene. This can be controlled in code (see Demos\Scripts\SampleHandInteract.cs) or by setting up a few things in the editor.

For code-free setup perform the following task:

1. If you don't already have a StickyDisplayModule (HUD), in your scene, add the Demos\Prefabs\Visuals\StickyDisplay1 to your scene Hierarchy. For more information on configuring the HUD, see the "Sticky Display Module" chapter.
2. In "Display Reticle Settings", consider enabling "Lock Reticle to Cursor".
3. On your S3D character, go to the "Look" tab and enable "Interactive".
4. On your S3D character, go to the "Engage" tab, and expand "Event Settings"

5. Add a new “On Look At Changed” event
6. Drag the StickyDisplayModule from the scene into a new event object and set the Function to “Dynamic” StickyDisplayModule.InteractiveLookAtChanged.
7. Under “General Engage Settings” you can adjust the “Engage Colour” and “Non-engage Colour” as required.

Now, when your character starts or stops looking at an interactive-enabled object, the on-screen reticle will change colour to help the player visually identify interactive-enabled objects.

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 OR 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.

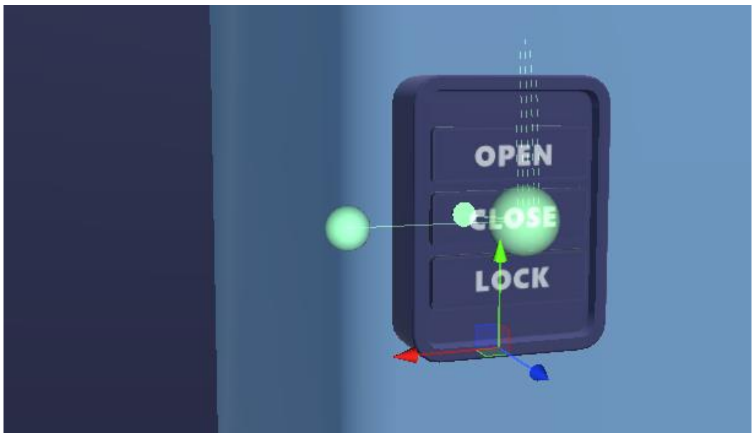
Sticky Interactive – Operating Stationary Machinery

There are multiple ways you could configure stationary machinery to be operated by a S3D character. For these examples, we’re going to be using some non-Sticky3D assets to control doors. You could simply add a box trigger collider to some doors, that get triggered to automatically open or close when the S3D character enters or exits the collider area. However, that’s not very interactive and doesn’t involve any gameplay. For example, what if the door is locked what happens if I’m say lowering an outer door and ramp on a spaceship? Maybe, as the player, I want to sometimes leave the doors open after leaving the collider area.

To make the experience feel a little more interactive and purposeful, we’re going to add a Sticky Interactive component.

Interactive Door Scenario 1

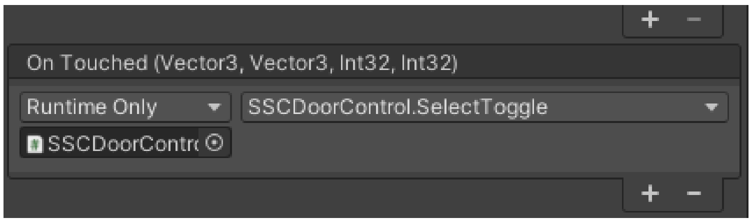
In this scenario we are using a “SSC Door Control” component from “Sci-Fi Ship Controller” (an asset available on the Unity Asset Store). We want out character to reach out and touch the physical door control panel to operate it.



So, we’ll make the interactive-enabled object in our scene “Is Touchable”.

The Hand Hold Primary Relative Position and rotation will have the hand positioned flat against the control when we reach out and touch it.

Our 3rd party asset has a Toggle open/close method, so we’ll add that to our Sticky Interactive OnTouched event. That way, when our character reaches out and touches the panel, the doors will (in this case) open or close the doors.


On Touched (Vector3, Vector3, Int32, Int32)
Runtime Only | SSCDoorControl.SelectToggle
SSCDoorContr

Once our character is configured to look at interactive-enabled objects (see "Sticky Interactive – Looking at Objects" above), we can add a Custom Input on the "Sticky Input Module" to engage with our control panel in the scene just by looking at it and clicking say the F key.

Hand IK need to be configured and enabled for the character to reach out and touch interactive-enabled objects.

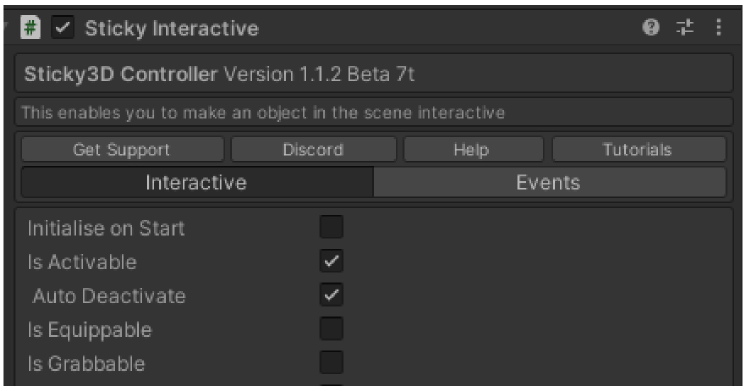
The character to the right, PlayerAmy, from the Demos\Prefabs\Characters folder, comes preconfigured with this. As do PlayerBryce, PlayerKate, and PlayerJeff. They all call "EngageLookingAt" when the F key is pressed.

Interactive Door Scenario 2

This next scenario requires a bit of custom C# code, but it provides a more flexible and dynamic solution.

In this scenario we are also using the "SSC Door Control" component from "Sci-Fi Ship Controller".

However, here, we want to activate a rear door and ramp on our spacecraft when the player popups up a set of options and selects that one that is appropriate. So, we'll make it "Is Activable" and immediately deactivate it once the operation has been actioned.



Sticky Interactive

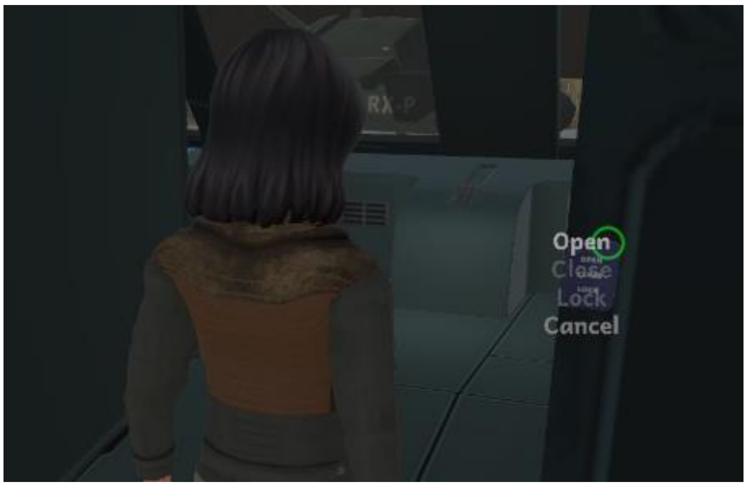
Sticky3D Controller Version 1.1.2 Beta 7t

This enables you to make an object in the scene interactive

Initialise on Start	☐
Is Activable	✓
Auto Deactivate	✓
Is Equippable	☐
Is Grabbable	☐

In the OnActivated event of the interactive-enabled, we'll call a custom method in our game-code to open a pooled StickyPopupModule using the StickyManager in the scene.

This will configure the available options (depending on if the ramp is open, closed or locked). When the player points to and selects the appropriate option, this will call another custom method in our game which will call SelectOpen or SelectClose on our 3rd party "SSC Door Control" script.



Interactive Door Scenario 3

In this scenario we modified some scripts that came with a 3rd party art asset. We added a StickyZone to the elevators so that when our character walks into an elevator, it is assigned as the reference frame. This allowed our character to seamlessly ride up and down between floors.

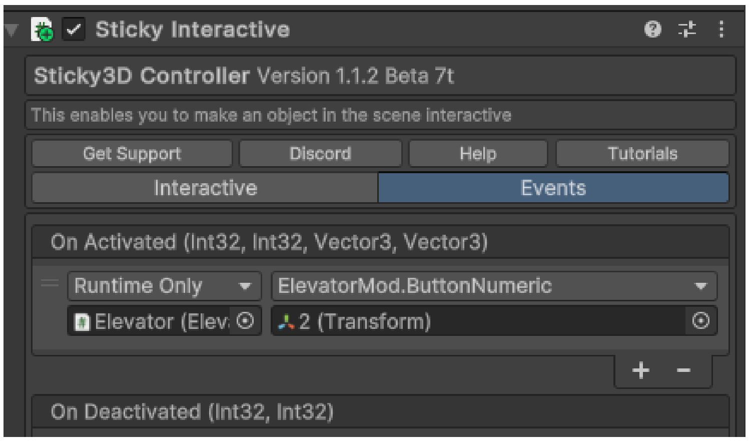
In this scenario we are using a modified Elevator controller from the 3rd party "QA Office and Security Room" asset.

We'll add a StickyInteractive component to both the up and down buttons. We'll make it "Is Activable" and immediately deactivate it once the operation has been actioned.

In the OnActivated event of the interactive components, we'll call a ButtonOpen method we've exposed in the modified 3rd party Elevator script.

Inside the elevator there is a panel of floor numbers. We can add a StickyInteractive component to each of those and hook the OnActivated event to call a ButtonNumeric method.

Each button takes its own transform as a parameter for ButtonNumeric().



Sticky3D Controller Version 1.1.2 Beta 7t

This enables you to make an object in the scene interactive

On Activated (Int32, Int32, Vector3, Vector3)

Runtime Only | ElevatorMod.ButtonNumeric

Elevator (Elev | 2 (Transform)

On Deactivated (Int32, Int32)

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.

To use interactive objects with other assets, see "Sticky Interactor Bridge" and "SSC Input Bridge".

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. The gameobject should include a trigger collider.

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

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.
Is Left Hand	Is this being used for the left hand? (Otherwise assume right hand)

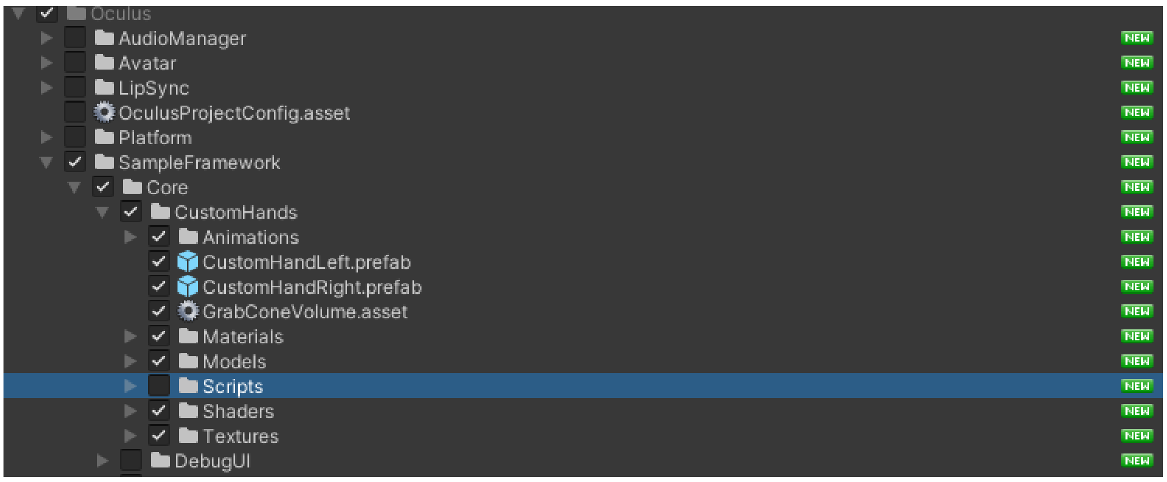
Interactive Property	Description
Can Activate	Can this item activate an interactive-enabled Activable object?
Can Grab	Can this item grab an interactive-enabled Grabbable object?
Auto Drop	Automatically drop (let go of) an object that goes outside the hand collider range. If disabled, and the player releases the grip on the hand controller, the held object will be dropped (let go of).
Auto Grab	The component will attempt to automatically grab an interactive-enabled Grabbable object when it is in range. If this is disabled and the player presses the grip on the hand controller, when in range of an interactive object, it will be grabbed.
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?

The non-Sticky3D hands can be animated if you have animations for them. For example, you may be using the Oculus animated hands with your rig.

If you own Sci-Fi Ship Controller, you can configure XR Hands using “UnityXR – Overview” instructions found in the “Player Input Module” chapter of the SSC manual. Then, follow these additional steps:

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), version 42.0 (20 Jul 2022), and 57.0.1 (10 Oct 2023).

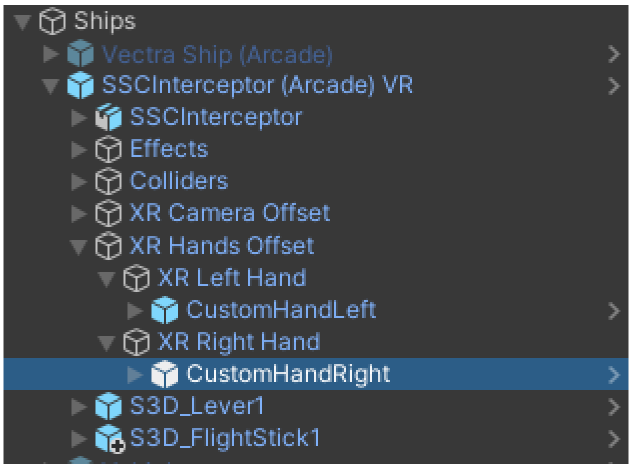
   You can also get this from: https://developer.oculus.com/downloads/package/unity-integration/



```
▼ ✓ Oculus
  ► ☐ AudioManager                         NEW
  ► ☐ Avatar                               NEW
  ► ☐ LipSync                              NEW
    ☐ OculusProjectConfig.asset            NEW
  ► ☐ Platform                             NEW
  ▼ ✓ SampleFramework                      NEW
    ▼ ✓ Core                               NEW
      ▼ ✓ CustomHands                      NEW
        ► ✓ Animations                     NEW
          ✓ CustomHandLeft.prefab          NEW
          ✓ CustomHandRight.prefab         NEW
          ✓ GrabConeVolume.asset           NEW
        ► ✓ Materials                      NEW
        ► ✓ Models                         NEW
        ► ☐ Scripts                        NEW
        ► ✓ Shaders                        NEW
        ► ✓ Textures                       NEW
      ► ☐ DebugUI                          NEW
```

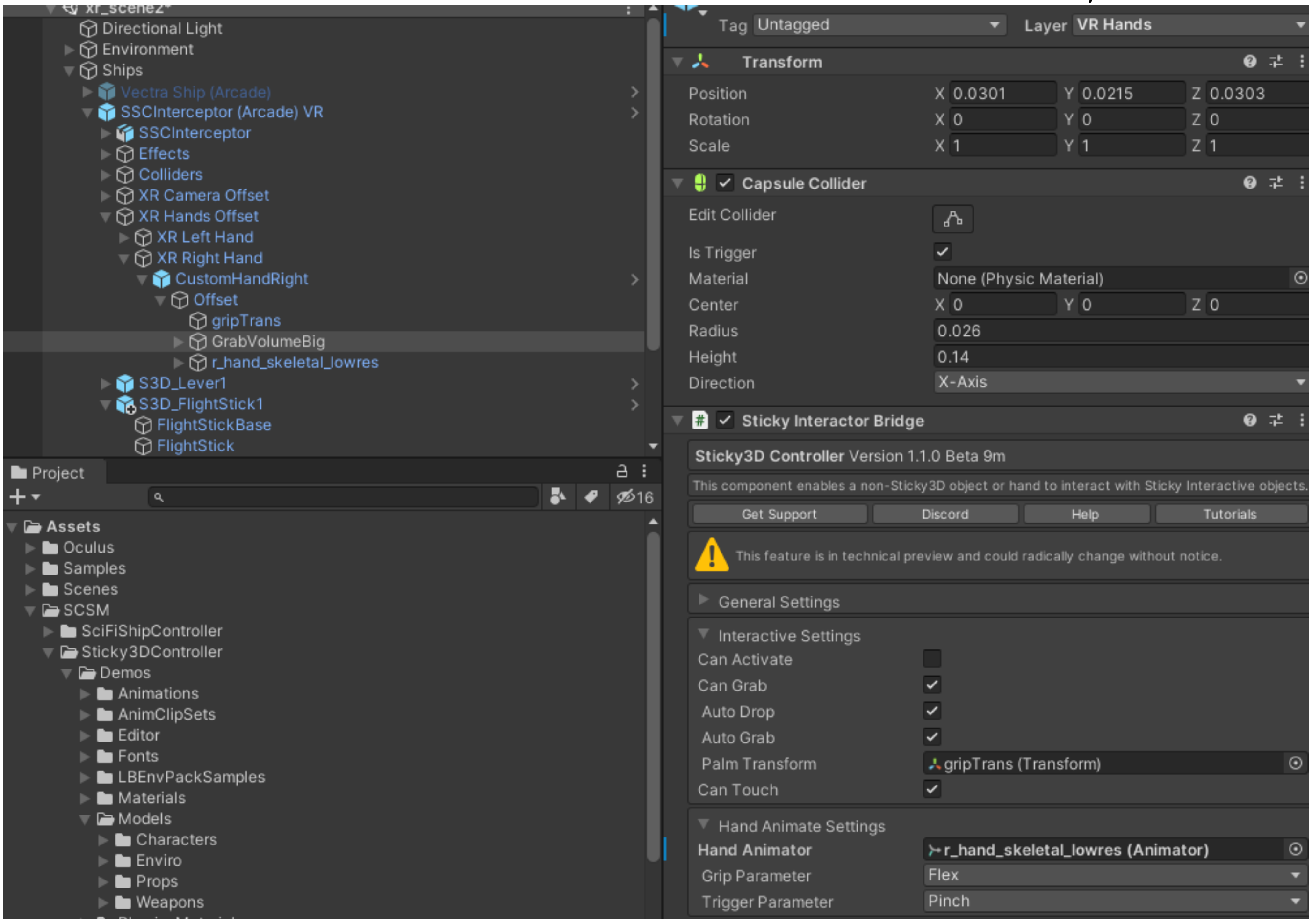
2. In the Project pane, open the Oculus, SampleFramework, Core, CustomHands, CustomHandLeft prefab. Remove the two missing Script components. Save the prefab.

3. Repeat with CustomHandRight prefab.
4. In scene, under your SSC ship, XR Hands Offset, XR Left Hand, add the CustomHandLeft prefab from Oculus to "XR Left Hand" gameobject.



5. In the scene Hierarchy panel, expand “XR Hands Offset”, “XR Right Hand”, “CustomHandRight”, “Offset” and select “GrabVolumeBig” (see image below).
6. Ensure the Capsule Collider has “Is Trigger” set.
7. Add the “StickyInteractorBridge” component (Component menu -> Sticky3D Controller -> Utilities -> Sticky Interactor Bridge)
8. In “General Settings” of the component enable “Initialise on Start” (leave “Left Hand” off for the right hand)
9. In “Interactive Settings”, enable “Can Grab” and add the “gripTransform” to the “Palm Transform” slot.
10. If you want the user to manually grab and release objects (like a flight stick or a lever), untick “Auto Drop” and “Auto Grab”.
11. From the scene, add “r_hand_skeltal_lowres” to the “Hand Animator” slot.
12. Set the Animator “Grip” parameters to “Flex” and the “Trigger” parameters to “Pinch”.
13. Repeat steps 5 to 11 for the left hand.

Hand Property	Description
Hand Animator	The animator used to animate a Virtual Reality left or right hand
Grip Parameter	The (float) parameter name from the animation controller for the grip action
Trigger Parameter	The (float) parameter name from the animation controller for the trigger action



```
Directional Light
Environment
Ships
  Vectra Ship (Arcade)
  SSCInterceptor (Arcade) VR
    SSCInterceptor
    Effects
    Colliders
    XR Camera Offset
    XR Hands Offset
      XR Left Hand
      XR Right Hand
        CustomHandRight
          Offset
            gripTrans
            GrabVolumeBig
            r_hand_skeletal_lowres
    S3D_Lever1
    S3D_FlightStick1
      FlightStickBase
      FlightStick
```

```
Project
Assets
  Oculus
  Samples
  Scenes
  SCSM
    SciFiShipController
    Sticky3DController
      Demos
        Animations
        AnimClipSets
        Editor
        Fonts
        LBEnvPackSamples
        Materials
        Models
          Characters
          Enviro
          Props
          Weapons
```

Tag: Untagged   Layer: VR Hands

Transform

	X	Y	Z
Position	0.0301	0.0215	0.0303
Rotation	0	0	0
Scale	1	1	1

Capsule Collider ✓

Edit Collider	
Is Trigger	✓
Material	None (Physic Material)
Center	X 0 Y 0 Z 0
Radius	0.026
Height	0.14
Direction	X-Axis

Sticky Interactor Bridge ✓

Sticky3D Controller Version 1.1.0 Beta 9m

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

Get Support | Discord | Help | Tutorials

This feature is in technical preview and could radically change without notice.

General Settings

Interactive Settings

Can Activate	
Can Grab	✓
Auto Drop	✓
Auto Grab	✓
Palm Transform	gripTrans (Transform)
Can Touch	✓

Hand Animate Settings

Hand Animator	r_hand_skeletal_lowres (Animator)
Grip Parameter	Flex
Trigger Parameter	Pinch

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

SSC Input Bridge

This component can be used for sending Readable Interactive-enabled object data to a Sci-Fi Ship Controller (SSC) ship or space craft. The most common scenario is when you want to send data from a lever or joystick in VR to a ship you want to fly. Read the above "Sticky Interactor Bridge" chapter before continuing.

NOTE: SSC is not included and is a separate asset available from the Unity Asset Store.

If you want to read data from an SSC ship and send it to a lever or joystick, use SSC Output Bridge instead.

The basic setup is:

1. Add a readable interactive lever or joystick to your SSC ship. See Demos\Prefabs\Props\S3D_Lever1. Make sure it is not inside another collider within the ship (you ship cabin should be hollow – don't use a single convex mesh collider for your ship model).
2. Attach this component to a lever or joystick on the same gameobject as the StickyInteractive component (the StickyInteractive component is on a child gameobject of the S3D_Level1 in the scene – this component must be attached to that same child gameobject).
3. Tick "InitialiseOnStart" or call Initialise() from your code.
4. Drag your Sci-Fi Ship Controller ship or craft into the shipGameObject or configure it in your game code by calling sscInputBridge.SetShip()
5. Configure the InputAxis. e.g., InputAxisZ to Longitudinal

6. In the scene, expand the S3D_Lever1 gameobject and locate the "Lever" child object which contains the “StickyInteractive” component.
7. On the interactive enabled object, go to the Events tab, and add a new “On Readable Value Changed” event.
8. Drag the Lever game object (which contains StickyInteractive and SSC Input Bridge components) into the “Object” field of the event.
9. Set the “Function” to be SSCInputBridge.OnInputChanged.

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 component is enabled through code.
Ship GameObject	The player ship from your scene which should contain a ShipControlModule on the same gameobject. It must contain a Player Input Module.
Input Axis X	Interactive Readable left-right data, to be send to the input axis of the Sci-Fi Ship Controller player ship. Currently we don’t support changing this at runtime as it could cause issues with other components.
Input Axis Z	Interactive Readable forward-back data, to be send to the input axis of the Sci-Fi Ship Controller player ship. Currently we don’t support changing this at runtime as it could cause issues with other components.

Before you attempt to fly your ship, using the SSC Input Bridge, check out the section called “Unity XR – Hands and Physics Collisions” in the “Player Input Module” chapter of the **Sci-Fi Ship Controller manual**.

SSC Output Bridge

This component can be used for getting Sci-Fi Ship Controller (SSC) input data from a ship or space craft and sending it to a writeable Interactive-enabled object. The most common scenario is when you want an in-game lever or joystick to move as the ship is getting input data from a S3D player or AI module.

If you want to read data from a lever or joystick and send it to an SSC ship, then use SSC Input Bridge instead.

NOTE: SSC is not included and is a separate asset available from the Unity Asset Store.

The basic setup is:

1. Add a writeable interactive lever or joystick to your SSC ship. Make sure it is not inside another collider within the ship (you ship cabin should be hollow – don’t use a single convex mesh collider for your ship model).
2. On the StickyInteractive component, set the Readable Pivot. This is most likely the same gameobject that also has the StickyInteractive component attached.
3. The interactive lever or joystick must have a kinematic rigidbody.
4. If your character will be using Hand IK to grip the joystick or lever, the kinematic rigidbody on the interactive object may need Interpolate set to “Interpolate” and Collison Detection set to “Continuous Speculative”.
5. Add a SSCOutputBridge component to the lever or joystick on the same gameobject as the StickyInteractive component. If the StickyInteractive component is on a child gameobject, this component must be attached to that same child gameobject.
6. Drag the Ship’s gameobject into the slot provided.
7. Configure the OutputAxis. e.g., OuptAxisZ to Longitudinal
8. Ensure the ship and the lever or joystick are on different Unity Layers and that they don’t collide with each other. This can be configured in Player Settings -> Physics from the Unity Editor. If the configuration is incorrect, you will see warning errors in the Unity console at runtime.

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 component is enabled through code.
Enable Collider on Init	If the interactive collider is not enabled when initialising, enable it.
Recentre on Configure	When the ship is assigned or an axis is changed, automatically recentre the object in the next frame.
Ship GameObject	The ship from your scene which should contain a ShipControlModule on the same gameobject.
Output Axis X	Interactive Writeable left-right data, to be read from the input axis of the Sci-Fi Ship Controller ship.
Output Axis Z	Interactive Writeable forward-back data, to be read from the input axis of the Sci-Fi Ship Controller ship.
Timing Type	When the data is sent to the interactive-enabled object. Set to Manual if you want to call SendData(..) from your own gameplay loop.

To monitor the data used for an Output Axis, at runtime in the editor use either:

- Debug Mode for the ShipControlModule and enable “Ship Input”.
- Debug Mode for the StickyInteractive component and examine the “IsReadable” data.

Sticky Magazine (a.k.a. “Clip”)

This is an interactive-enabled container that holds ammo for a weapon. See also “Sticky Weapon”. Magazines, also known as “Clips”, fit into a weapon and typically contain multiple bullets (we call these “Projectiles” because in a Sci-Fi setting, they could be bursts or pulses of “energy” from some futuristic weapon).

A magazine can be attached to a weapon, or it can be lying around a scene waiting for a character to discover. It could be full of ammo, empty, or somewhere in between. Magazines can be picked up (grabbed) or dropped.

For a magazine to be compatible with a weapon, the “Mag Type” must match the Firing Button “Magazine Type” in the “Ammo Settings” on the Weapon tab of the Sticky Weapon. ALSO, the “Ammo Type” that the magazine holds, must match one of the “Compatible Ammo Types” of the weapon’s firing button.

Sticky Magazine – Interactive Properties

See the “Sticky Interactive” chapter for a description of these properties.

Sticky Magazine – Magazine General Properties

Property	Description
Available Ammo Types	All weapons and magazines in your scene should use the same (common) set of ammo types. You could use our Demo Ammo Types scriptable objects OR create your own unique set by going to the Project pane, clicking “+” then Sticky3D -> Ammo Types. You can then give them custom names and properties to suit your game.
Available Mag Types	All weapons and magazines in your scene should use the same (common) set of magazine types. You could use our Demo Mag Types scriptable objects OR create your own unique set by going to the Project pane, clicking “+” then Sticky3D -> Magazine Types. You can then give them custom names to suit your game.
Mag Capacity	The amount of ammo the magazine can hold.
Magazine Type	The type of magazine. Magazines with the same type can fit the same weapons.
Ammo Type	The ammo type that can be placed in this weapon. Only one type of ammo can be used in a magazine at any point in time.
Ammo Count	The amount of ammo currently in the magazine. It cannot exceed the Mag Capacity.
Disable Regular Colliders	Disable regular colliders when equipped (attached) to a weapon.

Sticky Magazine – Events

Events are triggered when certain things happen to the magazine. 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 Grabbed	These are triggered by a S3D character when they grab this magazine.
On Dropped	These are triggered by a S3D character when they drop this magazine.
On Hover Enter	These are triggered by a S3D character when they start looking at this magazine.
On Hover Exit	These are triggered by a S3D character when they stop looking at this magazine.
On Post Grabbed	There are triggered by a S3D character immediately after they grab this magazine.
On Post Stashed	These are triggered by a S3D character immediately after they stash (place in inventory) this magazine

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 Beam Modules	The list of beam modules added to the pool at start-up. Beam modules can also be added via code
Startup Decal Sets	The list of decal sets added to the pool at start-up. Decals can also be added via code. Each set can contain multiple StickyDecalModule prefabs.
Startup Dynamic Modules	The list of dynamic modules added to the pool at start-up. Dynamic objects can also be added via code.
Startup Effects Modules	The list of effects modules added to the pool at start-up. Effects can also be added via code.
Startup Projectile Modules	The list of projectile modules added to the pool at start-up. Projectile Modules can also be added via code.
Startup Generic Modules	The list of generic modules added to the pool at start-up. Generic modules can also be added via code. See Demos\SampleGenericTextModule.cs for an example.

Currently there are two general purpose components that can be used to make your objects poolable. Other poolable components include:

- Sticky Beam Module
- Sticky Decal Module
- Sticky Dynamic Modules
- Sticky Effects Module

- Sticky Projectile Module

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.

Property	Description
Min Pool Size	The starting size of the pool.
Max Pool Size	The maximum allowed size of the pool.
Despawn Time	The object will be automatically despawned after this amount of time (in seconds) has elapsed.
Is Reparented	Does this object get parented to another object when activated? If so, it will be reparented to the pool transform after use.

StickyPopupModule

This component inherits from StickyGenericModule and can be used to display a clickable popup menu over an interactive-enabled object. Sample scripts (Demos\Scripts\SamplePopupOptions and SampleEquipGrabStashPopup) are 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. Some test popup prefabs have been included in Demos\Prefabs\Visuals to get you started.

Property	Description
Min Pool Size	The starting size of the pool.
Max Pool Size	The maximum allowed size of the pool.
Despawn Time	The popup will be automatically despawned after this amount of time (in seconds) has elapsed.
Is Reparented	Does this popup get parented to another object when activated? If so, it will be reparented to the pool transform after use.
Button Colour For Text	This can be useful when button has no background image and the Text should change colour based on button transition tint colours.
Pause Weapons Firing	If a character is assigned to the popup, any held weapons will not fire while the popup is shown.
Unpause Weapons Delay	When the popup is closed and it was initiated from a character (directly or indirectly), delaying the unpausing of weapons by a small amount can help to prevent unintentional weapon firing. Smaller delays are typically better. Try 0.1 seconds to start with and increase if required.
Messages	0, 1 or more messages can be added to a popup.
Show Message	Show the message on the popup. [Has no effect outside play mode]
Message Name	The name or description of the message. This can be used to identify the message. It is not displayed in the popup.
Message Label Text	The text to display in the message label. It can include RichText markup. e.g., <b>Bold Text</b>
Message Value Text	The text to display in the message. It can include RichText markup. e.g., <b>Bold Text</b>
Message Panel	The UI Panel that holds the label and value RectTransforms.
Message Label Panel	The UI Panel for the message label.
Message Value Panel	The UI Panel for the message value text.

When using popups in VR, see the example Demos\Prefabs\Visuals\StickyPopup[n]VR prefabs. You may notice that they have a Sticky Graphic Raycaster (VR) component rather than the default Graphic Raycaster. This is to enable StickyXRInteractor components, that you would attach to VR hands, to point at, and click, buttons within your UI.

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 Projectile Module

This is a poolable projectile that can be fired from a StickyWeapon.

This component inherits from StickyGenericModule which is managed by StickyManager.

How to Create Projectile Prefabs

To create a new Projectile Module prefab:

1. In a scene, using the 3D Object Menu, select Sticky3D Controller -> Sticky Projectile
2. Rename the StickyProjectile. E.g., MyProjectile1
3. Create a prefab from the gameobject by dragging the parent gameobject into a folder in the Project pane
4. Reset the prefab parent transform position and rotation to 0,0,0
5. It is assumed that the projectile mesh is pointing forward in z-axis direction. If it is not, you'll need to update the Model Rotation property. For example, if your projectile model is pointing upward (toward the y-axis), you will need to rotate it 90 degrees around the local x-axis. So, set "Model Rotation" to 90, 0, 0.

Sticky Projectile – Properties

Property	Description
Start Speed	The speed the projectile starts traveling in metres per second. Is Ignored for Raycast weapons.
Damage Type	The type of damage the projectile does when hitting a character. The amount of damage dealt to a S3D upon collision is dependent on the character's resistance to this damage type. If the damage type is set to Default, the character's damage multipliers are ignored i.e., the damage amount is unchanged.
Damage Amount	The amount of damage this projectile does to the character or object it hits. NOTE: Non-character objects need a Sticky Damage Receiver component.
Model Rotation	The projectile should point forward on the z-axis. If it, say, points upward (y-axis), it needs to be rotated 90 degrees here around the x-axis.
Min Pool Size	The starting size of the pool.
Max Pool Size	The maximum allowed size of the pool.
Is Reparented	Does this object get parented to another object when activated? If so, it will be reparented to the pool transform after use.
Despawn Time	The object will be automatically despawned after this amount of time (in seconds) has elapsed.
Effects Object	The default particle and/or sound effect prefab that will be instantiated when the projectile hits something. This would typically be a non-looping effect.
Default Decal Set	The default set of Sticky Decal Module prefabs used to select a random decal when the projectile hits an object. To configure different decals for different objects, see the chapter called "Sticky Surface".

Sticky Shapes Module

This module lets you configure blendshapes, typically, for facial expressions. It also enables your character to subtlety move their eyes and move their lips and change facial expressions while a voice-over audio clip is playing.

For this module to work, your character model needs to include blendshapes. Sticky3D does NOT create the blendshapes for you.

This is NOT a text-to-speech (TTS) system. That's up to you. Sorry.

Although you "could" code the use of blendshapes yourself, this component will help automate some of the things you might want to do with blendshapes.

This module includes an extensive API with which you can control it using your own game code. See the "Runtime and API" chapter for more information.

Sticky Shapes – General Settings

Property or Button	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 zone through code.
Non S3D Controller	The character is NOT using a Sticky3DController. i.e., it is using a 3rd party character controller.
Get Skinned Mesh Renders	Attempt to find the mesh renderers that contain the blendshapes.
Skinned Mesh Renderers	A list of skinned mesh renderers that contain the blendshapes on your model.
Get Eyes	Attempt to find the eye bone transforms on the humanoid model.
Left Eye Transform	The left eye bone transforms on the humanoid model.
Right Eye Transform	The right eye bone transforms on the humanoid model.

Sticky Shapes – Blendshapes

This tab contains a list of all the blendshapes discovered from the mesh renderers from the General tab.

Property or Button	Description
Get Blendshapes	Attempt to find the blendshapes based on the list of Skinned Mesh Renderers from the General tab.
Blend Shapes	The list of blendshapes on the model.

Sticky Shapes – Emotions (Blink Settings)

Humans blink naturally. You'd also expect fictious humanoid characters to do the same. These settings can help to make your character to look more realistic, even if you are going for a cartoonish look. Default settings are based on normal human blink patterns.

Property	Description
Enable on Init	Should the eyes start blinking when this component is initialised?
Simple Blink	A lower-quality simplified blink action that is more performant.
Blink Duration	The time, in seconds, the blink takes to occur.
Blink Min. Interval	The minimum time, in seconds, between each blink.
Blink Max. Interval	The maximum time, in seconds, between each blink.
Blend Shapes	The list of blendshapes that make the eyes blink. Typically, there will be a single blendshape that performs the blinking action.
Blend Shape	The blendshape to blink the eyes.
Max Weight	The normalised weight to be applied to the blendshape.

Sticky Shapes – Emotions (Eye Movement Settings)

When humans look straight ahead, their eyes will wonder slightly. When the eyes are fixed straight ahead, the character will look unrealistic. Subtle eye movements can improve the overall realism of your character.

Property	Description
Enable on Init	Should the eyes start moving when this component is initialised?
Move Left-Right	The angle to move left or right. Lower absolute values typically look better.
Move Down-Up	The angle to move down and up. Lower absolute values typically look better.
Gaze Duration	When Eye Movement is enabled, the time, in seconds, the character is gazing in the same direction.
Move Speed	Human eyes can turn at around 700 degrees per second! However, this is over very short distances. Start with a speed of around 10.

Sticky Shapes – Emotions (List of Emotions)

Emotions in this module are essentially any blendshapes, or combination of blendshapes that do not involve lip movement for speech. They can be anything from a frown, a smile, to raised eye brows, or puffed-up cheeks. They can be faded in or out via our API (i.e., from your C# game-code) or automatically played when the character is talking.

Emotions can be described by intensity. There are four sliders with "opposite" traits from left to right. For example, on the Joy slider, all the way to the left (-1) is "sad", while to the right is "joy", with neutral (0) in the middle.

Property or Button	Description
SYN	Attempt to (re)synchronize all emotions with the defined blendshapes.
Name of Emotion	The descriptive name of the emotion.
Fade-In Duration	The time, in seconds, it takes the emotion to reach its full strength.
Fade-Out Duration	The time, in seconds, it takes the emotion to stop affecting the character.
VOX Emotion	Is this emotion randomly used while a speech audio clip is playing? See the Phonemes tab.
Anger Intensity	The emotion's anger intensity ranging from fear (-1) to anger (1). Default is neither fear nor anger (0).
Joy Intensity	The emotion's joy intensity ranging from sad (-1) to joy (1). Default is neither sad nor joy (0).
Surprise Intensity	The emotion's surprise intensity ranging from anticipation (-1) to surprise (1). Default is neither anticipation nor surprise (0).
Trust Intensity	The emotion's anger intensity ranging from disgust (-1) to trust (1). Default is neither disgust nor trust (0).
Blend Shapes	The list of blend shapes to be used with this emotion.
Blendshape	The blendshape.
Max Weight	The normalised weight to be applied to the blendshape.

Sticky Shapes – Phonetics (Settings)

The Phonetics tab, relates to when your character is speaking.

Property	Description
Audio Source	The audio source used to play speech audio clips. Typically, this will be a child component on your character's prefab.

Sticky Shapes – Phonetics (Phonemes)

A phoneme is a distinct sound that can, when combined with others, produce a spoken word or phrase. Technically, you might call them "phones" but that is too confusing for our purposes.

A viseme is the visual representation of the phoneme on the character.

Property or Button	Description
Pre	Preview the Viseme on the character.
Name of Phoneme	The name or title of the phoneme. E.g., AA, OW, TH, AY etc.
Duration	The time, in seconds, that the phoneme will be active on the character. A short phoneme is 40-80ms.
Frequency	The number of times the phoneme would be spoke in a sequence of 100 phonemes.
Blend Shapes	
Blendshape	The blendshape.
Max Weight	The normalised weight to be applied to the blendshape.

Sticky Shapes – Phonetics (Speech Audio Clips)

Speech Audio clips contain the spoken words you want to character to say as an audio file. These settings contain information about how those files should be played.

Property or Button	Description
Speech Audio Clips	The list of speech audio clips used for this character.
Pre	Play the audio clip and preview visemes and VOX-enabled emotions on the character.
Auto Clip	Audio clip that holds the sound of the character speaking.
Speech Speed	The speed to play back the phonemes, relative to the phoneme timings set on the character.
Volume	The relative playback volume of the audio clip.
Silence Threshold	Volume below this level will be considered as silence.
Emotions Ignore Silence	Do emotions with “VOX Emotion” enabled, play through silent sections of the audio clip?
Emotion Interval	The interval between VOX emotions playing when “Emotions Ignore Silence” is true.
Emotion Fade on Silence	Should the emotion immediately start fading out when there is silence in the audio and “Emotions Ignore Silence” is false? Emotions may never reach their maximum set weight; however, it may look a little more random.

Sticky Shapes – Engage (React)

Located on the Engage tab, the React Settings are used to control how the character reacts to others in the scene. For example, when an NPC sees a friendly character, they could smile. However, if they see an enemy, they could frown. This can contribute to general emersion in your game.

For S3D characters to react to others, the “Trigger Collider” option needs to be enabled on the Collide tab of the StickyControlModule.

Engage React Settings

Property or Button	Description
Enable on Init	Should the character get ready to react when this component is initialised?
No Notify Duration	The number of seconds, after initialisation, that reactions will not be triggered by a character entering or exiting the area. This can be useful if you do not want a character within the collider area to immediately trigger reactions when the component is initialised.
React Distance	The distance at which the character starts to react to others in the scene.
Proximity Collider	The trigger collider used to detect nearby characters. It must be on a child gameobject within the prefab.

Reactions

Reactions can be configured for different scenarios. For Example, you might want NPCs to give a fearful or angry emotion when it encounters an enemy (foe). Or maybe give a friendly smile when meeting a player in the same faction.

An enter or exit reaction can only be in use by one nearby character at any one time. That is, if your character is using say Reaction 1 to react to a nearby character, that same reaction cannot be used to perform a reaction until that reaction has been completed. Reactions are NOT queued.

Property or Button	Description
Name of Reaction	The descriptive name so that you can remember what the intention is for this reaction.
React To	

Property or Button	Description
Any	React to any other character that is within the React Distance of this character.
Friendly	React to character with the same faction or has a neutral faction (0)
FriendOnly	React to only characters with the same faction.
Foe	React to characters with a different faction. Does not react to neutral (0) characters.
NeutralOnly	Only react to neutral (0) characters.
Nobody	Does not react to any characters. Useful for temporarily disabling a reaction.
Models to Include (qty)	An optional array of model IDs that will further limit which characters to react to. The quantity can be changed to increase or decrease the size of the array. Model ID of a Sticky3D character is found on the Engage tab of the StickyControlModule, under Identification Settings.
Auto Button	Automatically populate the list of Enter/Exit from the Emotions tab based on emotion intensities and the “React To” setting of the reaction.
Pre Button	Preview the reaction on the character at runtime in the editor.
Enter Settings	
Enter Emotions	An emotion from this list will be randomly played when another character enters the proximity area. If “Auto” is enabled, at runtime, the list will be populated from the Emotions tab based on emotion intensities and the “React To” setting.
Emotion nn	The name of the emotion to be randomly triggered from the Emotions tab.
Hold Duration	The length of time, in seconds, the enter emotion stays at fully active before fading back out.
Enter Events	
On Pre Enter	These are triggered immediately before reacting to when another character come nearby.
On Post Enter	These are triggered immediately after reacting to when another character come nearby. If there are no Enter Emotions or Speech Audios, this will be called immediately after the On Pre Enter events.
Enter Speech Audios	A speech audio clip from this list will be randomly played when another character enters the proximity area.
Exit Settings	
Exit Emotions	An emotion from this list will be randomly played when another character leaves the proximity area. If “Auto” is enabled, at runtime, the list will be populated from the Emotions tab based on emotion intensities and the “React To” setting.
Emotion nn	The name of the emotion to be randomly triggered from the Emotions tab.
Hold Duration	The length of time, in seconds, the enter emotion stays at fully active before fading back out.
Exit Events	
On Pre Exit	These are triggered immediately before reacting to when other characters stop being nearby.
On Post Exit	These are triggered immediately after reacting to when other characters stop being nearby. If there are no Exit Emotions or Speech Audios, this will be called immediately after the On Pre Exit events.
Exit Speech Audios	A speech audio clip from this list will be randomly played when another character leaves the proximity area.

Sticky Socket

This component, can be used to attach or snap interactive-enabled objects to a gameobject. It requires a trigger collider. Be sure to check out the “Socket Basics Tutorial” video.

To attach objects to a character, use the Sticky Control Module Equip Points or Stash, or use a Sticky Parts Module.

S3D characters can interact with StickySockets in the scene using the “Sockets” settings on the “Look” tab.

Sticky Socket – General 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 Socket component is enabled through code.
Max Items	The maximum number of items to attach to this socket.
Disable Regular Colliders	When adding, non-trigger colliders on interactive-enabled objects will be disabled. When removed, they will be re-enabled.
Disable Trigger Colliders	When adding, trigger colliders on interactive-enabled objects will be disabled. When removed, they will be re-enabled.
Interactive Tags	The Scriptable Object containing a list of 32 tags. Typically, only one would be used for all interactive objects as the text is only used to help you remember what each number from 1-32 represents. To create custom tags, in the Project pane, click Create->Sticky3D->Interactive Tags.
Permitted Tags	The interactive-enabled objects that are permitted to be attached to the socket. See the S3DInteractiveTags scriptableobject.
Default Popup Offset	The default local space offset a StickyPopupModule appears relative to the socket.
Default Popup Prefab	The default StickyPopupModule to display when a character engages with this socket.
Empty Popup Offset	The local space offset a StickyPopupModule appears relative to the socket when the socket is “empty” or has no interactive objects attached.
Empty Popup Prefab	The StickyPopupModule to display when a character engages with the socket and the socket is “empty” or has no interactive objects attached.

Sticky Socket - Events

Socket 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 Hover Enter	These are triggered by a S3D character when they start looking at this socket.
On Hover Exit	These are triggered by a S3D character when they stop looking at this socket.
On Pre Add	These are triggered near the start of the AddItem process for a StickyInteractive object.
On PostAdd	These are triggered immediately after a StickyInteractive object has been added to the socket.

Sticky Surface

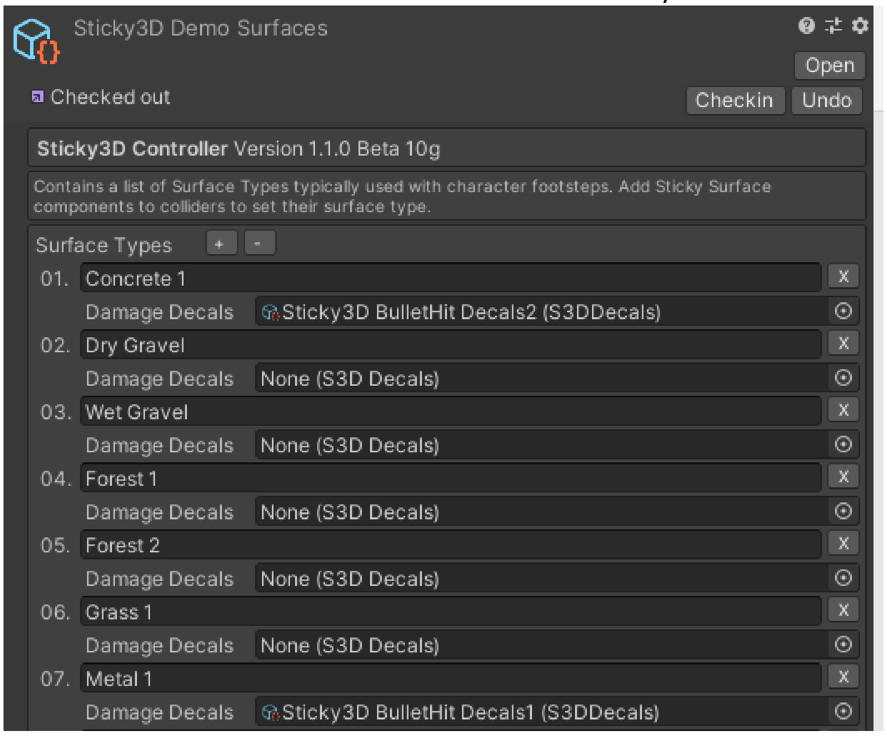
This small component can be added to any collider to help the character determine what type of surface they are stepping on. It is used with the “Footsteps” option on the Sticky Control Module “Move” tab.

It can also be used with the weapon system, to determine what type of surface was hit by a projectile.

[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 Scriptable Object outside the Sticky3DController folder.


Sticky3D Demo Surfaces

Open

Checked out

Checkin Undo

Sticky3D Controller Version 1.1.0 Beta 10g

Contains a list of Surface Types typically used with character footsteps. Add Sticky Surface components to colliders to set their surface type.

Surface Types + -

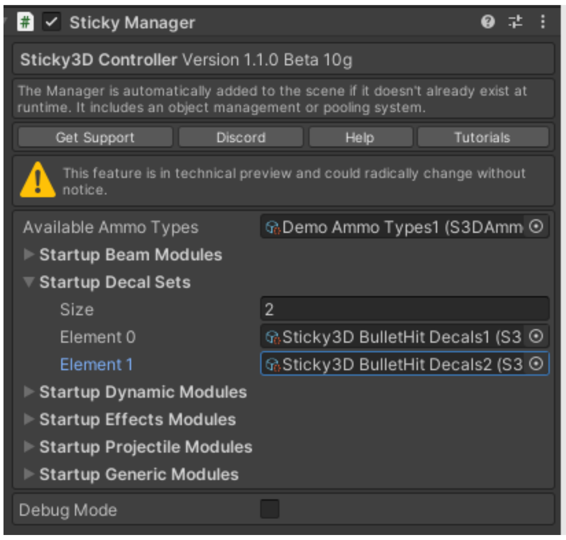
01.	Concrete 1	
	Damage Decals	Sticky3D BulletHit Decals2 (S3DDecals)
02.	Dry Gravel	
	Damage Decals	None (S3D Decals)
03.	Wet Gravel	
	Damage Decals	None (S3D Decals)
04.	Forest 1	
	Damage Decals	None (S3D Decals)
05.	Forest 2	
	Damage Decals	None (S3D Decals)
06.	Grass 1	
	Damage Decals	None (S3D Decals)
07.	Metal 1	
	Damage Decals	Sticky3D BulletHit Decals1 (S3DDecals)

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 or which kind of decals to deploy when hit by a projectile.

When you use Decals with Sticky Surface components, unlike "Default Decal Sets" in the "Sticky Projectile Module", they are not automatically pooled when your game launches.

By default, a pool is created the first time a different decal is required.

To pre-allocate your decal sets used with the Sticky Surface components, add the Damage Decal sets from your "Sticky3D Services" scriptable object asset to the "Sticky Manager" under "Startup Decal Sets".


Sticky Manager

Sticky3D Controller Version 1.1.0 Beta 10g

The Manager is automatically added to the scene if it doesn't already exist at runtime. It includes an object management or pooling system.

Get Support | Discord | Help | Tutorials

This feature is in technical preview and could radically change without notice.

Available Ammo Types	Demo Ammo Types1 (S3DAmm
▸ **Startup Beam Modules**	
▾ **Startup Decal Sets**	
Size	2
Element 0	Sticky3D BulletHit Decals1 (S3
Element 1	Sticky3D BulletHit Decals2 (S3
▸ **Startup Dynamic Modules**	
▸ **Startup Effects Modules**	
▸ **Startup Projectile Modules**	
▸ **Startup Generic Modules**	
Debug Mode	☐

Sticky Weapon System

Weapon System Overview

This feature is currently in **Technical Preview.** Some aspects of it **may not be production or game-ready**. 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 weapon system consists of several key components.

- Sticky Weapons
- Sticky Magazines (a.k.a. "Clips")
- Sticky Control Module (a.k.a. Characters)
- Sticky Beam Module (for configuring ray or laser beams)
- Sticky Projectile Module (for configuring projectiles or "ammo")
- Sticky Manger (for managing pooling for projectiles, beams, and effects)

The system is designed with a Sci-Fi flavour and is not meant to fully simulate a present-time real-world combat system. Although it can be used with models that depict RL weapons, that isn't our focus.

Setup consists of:

1. Configuring Sticky Weapons (which are special kinds of Sticky Interactive object) – See "Sticky Weapon – Setup Basics"
2. Configuring Sticky Magazines (which are also special Sticky Interactive objects)
3. Configuring a S3D character to be able to "Grab" (pickup) interactive objects (you might have already done this for other objects)
4. Configuring Sticky Input Module so character can fire (Left/Right Fire Button) and aim (with a Custom Input)
5. Configuring the S3D character so they can "Stash" interactive objects
6. Configuring the S3D character so they can switch weapons.

Grabbing and Dropping Weapons and Ammo

You player will probably want to wander around the scene and pickup weapons and magazines (which contain "hopefully" ammo that can be used in your weapon).

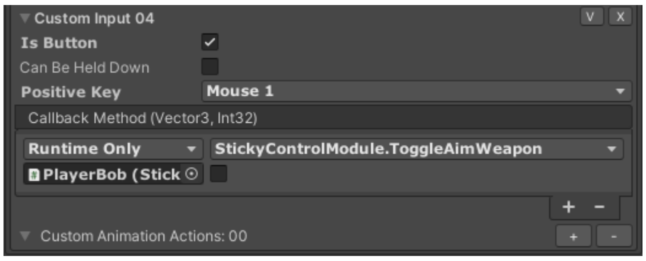
1. On the player "Sticky Control Module", go to the "Sticky Input Module".
2. Add a new "Custom Input"
3. Configure the button to use to grab and drop weapons and magazines. E.g., "G" button on the keyboard
4. Add a new event (Callback Method) and drag the "Sticky Control Module" from the Hierarchy into the "Object" field. For the "Function", select "StickyControlModule.ToggleHoldInteractive"
5. On the S3D "Look" tab, enable "Interactive"
6. If you have a StickyDisplayModule (a.k.a. "HUD") in your scene, on the "Engage" tab of the S3D character, expand "Event Settings", add a new event to "On Look At Changed" and drag the "StickyDisplayModule" from the scene into the "Object" field. Set the "Function" to "StickyDisplayModule" (Dynamic) InteractiveLookAtChanged.
7. Now your character should be able to walk around, look at a weapon or magazine and grab it into their right hand.

Aiming Weapons

When using a weapon with a player character, they may wish to have more precise aiming. This can be achieved by enabling or disabling aiming on a held weapon.

For example, you could call stickyControlModule ToggleAimWeapon(..) from a "Custom Input" on the "Sticky Input Module" attached to your player.

Here the player would use the right mouse button to toggle aiming on and off.


Custom Input 04
Is Button ✓
Can Be Held Down
Positive Key: Mouse 1
Callback Method (Vector3, Int32)
Runtime Only | StickyControlModule.ToggleAimWeapon
PlayerBob (Stick
Custom Animation Actions: 00

A weapon is an interactive-enabled object and can be held by a player (see "Grabbing and Dropping Weapons and Ammo above).

When aiming in first-person, the held weapon is always parented to the player's first-person camera. When coming out of aim mode, the weapon reverts to being parented to the S3D character's hand. The weapon offset is set in the "Weapon Anim Set" for the character-weapon pair.

When switching between first and third-person, aiming is always turned off. Aiming is also turned off if the weapon is dropped.

In third-person, there is an option to use the first-person camera for aiming. This style is popular with several popular third-person shooter games. If you don't wish to use first-person outside of aiming, simply disable the "Switch Look Button" on the player's "Sticky Input Module". See the "TP Aim uses FP Camera" option under "Weapon Anim Sets" for more details.

First-person aiming uses Hand IK to help align the hands with the weapon as the currently running animation may not match the weapon offset.

When a character starts or stops aiming a held weapon, you can call Sticky3D APIs and/or call your own code. See the Sticky Control Module Engage tab (Events) for more information.

Switching Weapons

Weapons for a character can be easily switched between a hand and the character "Stash" (inventory).

1. Configure at least two weapons to be Grabbable (see Sticky Weapon – Setup Basics).
2. Make sure you can grab and drop weapons in play mode.
3. Configure the "Engage – Stash Settings" on the character.
4. On the player "Sticky Control Module", go to the "Sticky Input Module".
5. Add a new "Custom Input"
6. Configure the button to use to switch weapons. E.g., "Z" button on the keyboard
7. Add a new event (Callback Method) and drag the "Sticky Control Module" from the Hierarchy into the "Object" field. For the "Function", select "StickyControlModule.SwitchStashedWeapon". Leave the bool field unchecked if you are using the right hand.

Sticky Weapon

A sticky weapon is an interactive-enabled object that can be grabbed, dropped, equipped, socketed, stashed, fired, and reloaded. It extends the features of a Sticky Interactive object. However, only the relevant properties are exposed in the Unity Inspector.

A weapon can have two firing mechanisms (Fire Button 1 and 2), although most weapons will only have a primary fire button. Unless this is a custom weapon, both firing mechanisms must match the Weapon Type.

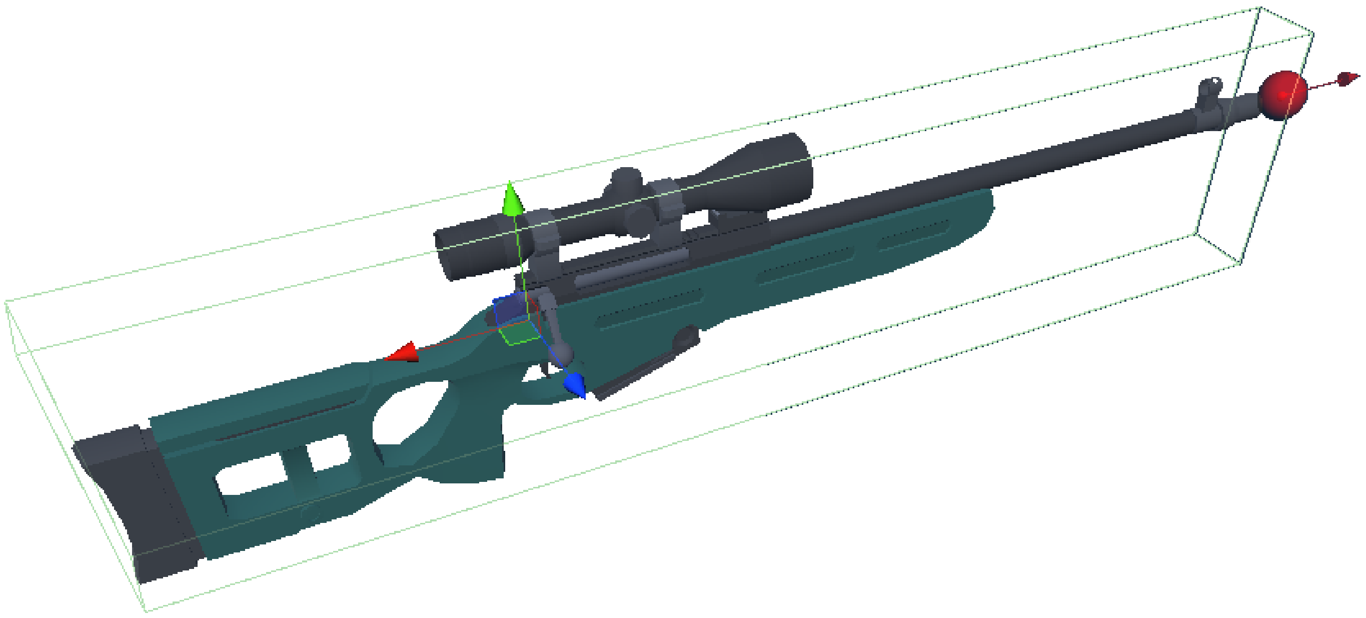
There are two demo weapons included in the pack, a projectile P320 hand gun, and the B43A1 laser rifle (which has a projectile pulse rifle variant). However, you should be able to configure your own weapon models, or those available from a third party.

Sticky Weapon – Setup Basics

Here are the basics of setting up a weapon. For details on each property, see the appropriate section below in this chapter.

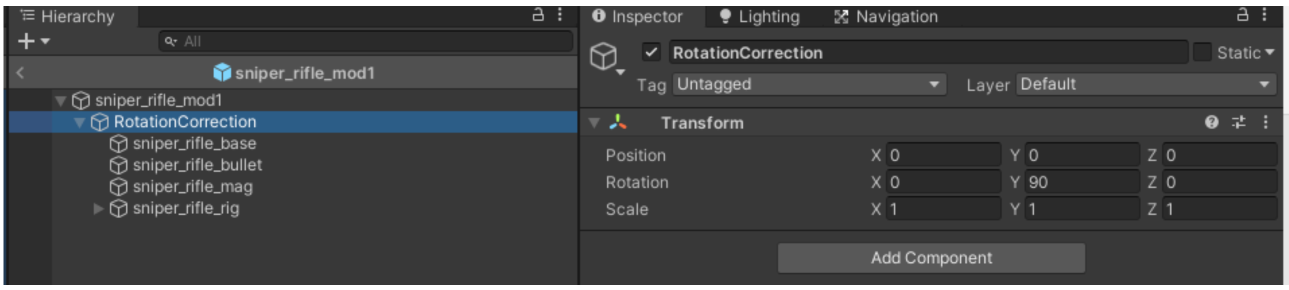
1. First, you'll need a weapon model, which is usually an FBX file. This may, or may not, include weapon animations like "firing" or "reloading" and a basic animation controller. You can find a wide variety of free and paid versions on the Unity Asset Store.
2. Add the weapon model as a child of an empty gameobject. Make sure you don't rotate the child transform of the model, otherwise aiming may not work correctly. **The 3rd party weapon below (not included in this**

pack), is incorrectly setup as the parent transform is not facing in the direction of the barrel (the blue z-axis should be pointing in the firing direction, but instead is pointing to the right). This is because the base of the rifle is rotated -89.98 degrees on the x-axis – mostly like as a result of it being imported directly from Blender.



To correct the above weapon FBX, there are a few options. The best is to re-export the FBX correctly so the model has zero rotation. If you don't have access to the original model files, you could create an empty gameobject under the FBX parent (you might need to first "Unpack Prefab"), and drag all the other weapon objects under the new child gameobject. Then, rotate the new child gameobject by say 90 on y-axis to make the weapon face in the correct direction. You can see the corrected prefab below.

You can now see that the blue z-axis points in the same direction the weapon is going to fire.



Hierarchy
sniper_rifle_mod1
sniper_rifle_mod1
RotationCorrection
sniper_rifle_base
sniper_rifle_bullet
sniper_rifle_mag
sniper_rifle_rig

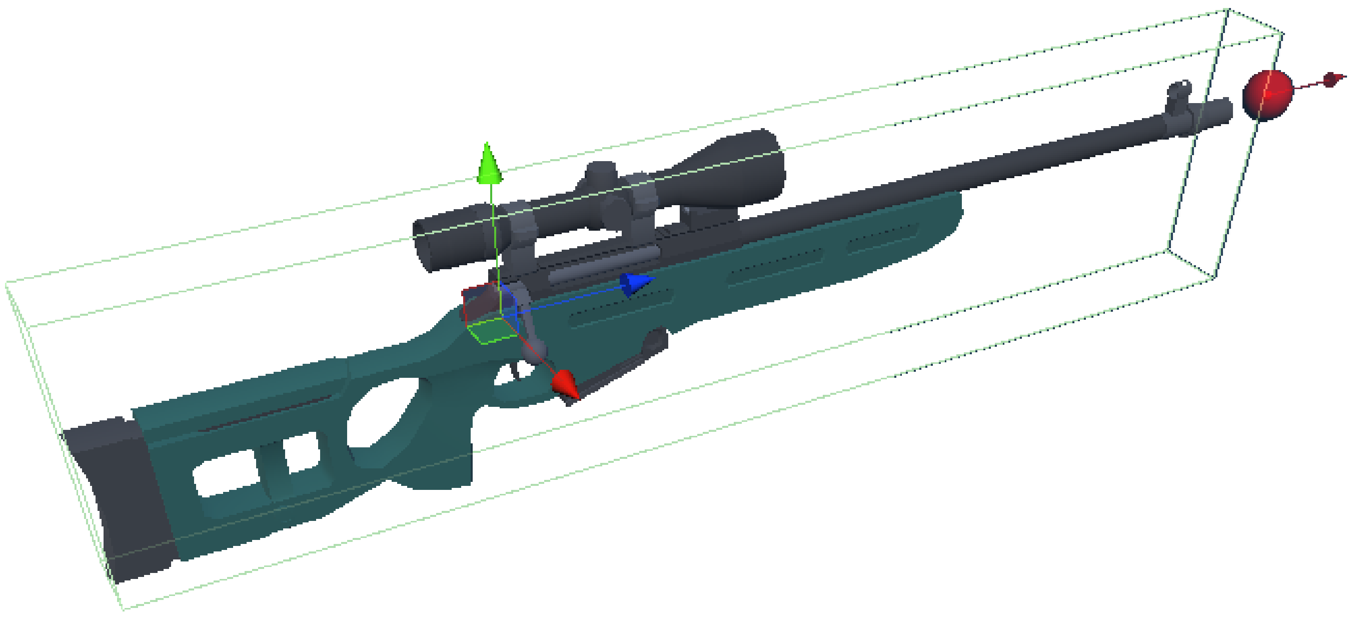
Inspector | Lighting | Navigation

RotationCorrection   Static
Tag Untagged   Layer Default

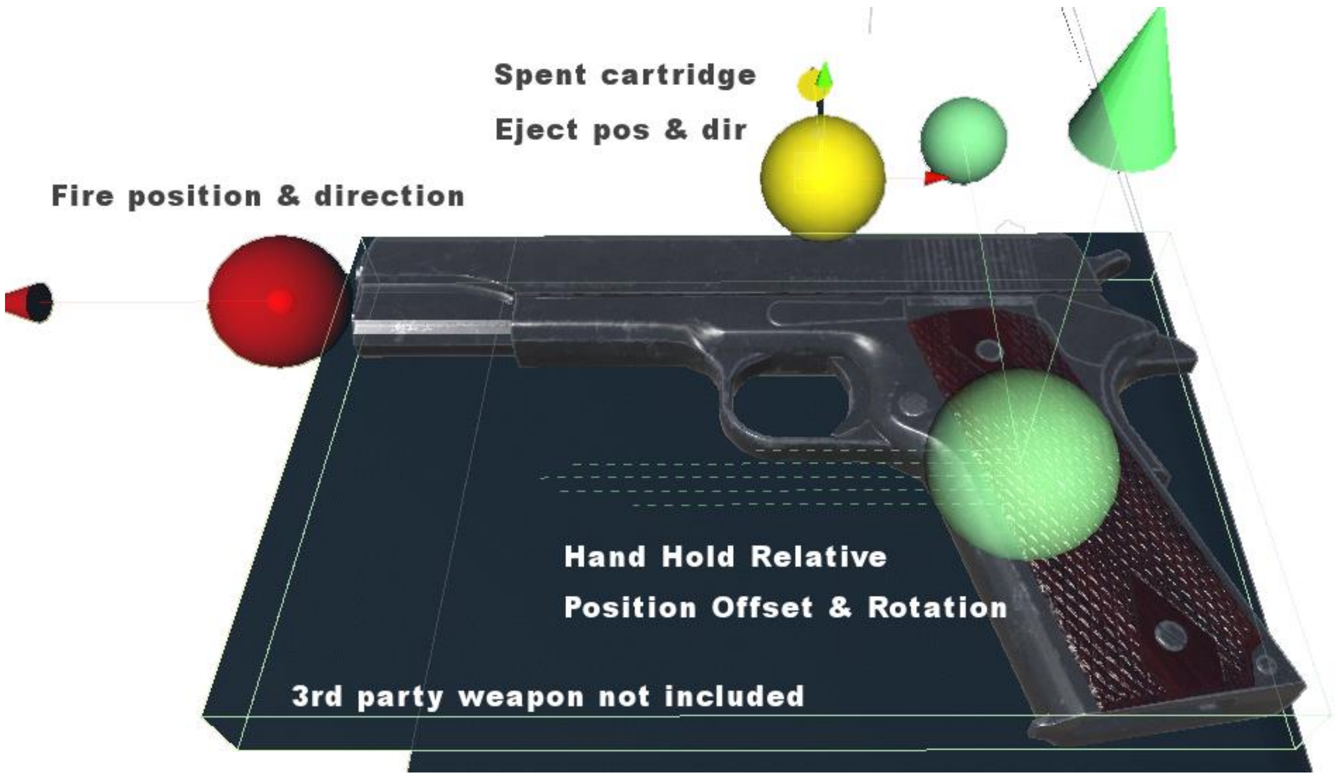
Transform

Position	X 0	Y 0	Z 0
Rotation	X 0	Y 90	Z 0
Scale	X 1	Y 1	Z 1

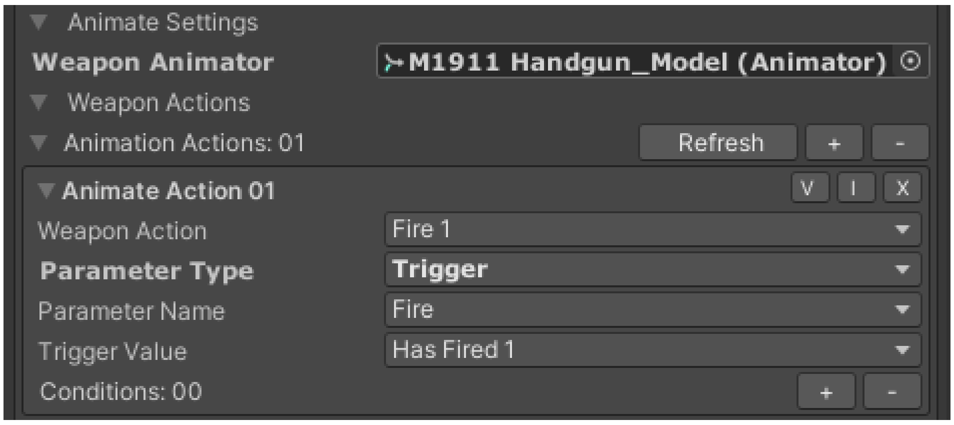
Add Component



3. Rename the empty gameobject. E.g., “My Weapon”
4. Add a StickyWeapon component to the parent empty gameobject
5. On the “Interactive” tab, you’ll typically want to enable “Initialise on Start”, “Is Grabbable”, “Carry in Hand”, “Parented on Grab”, “Re-parent on Drop”, “Disable Regular Colliders”, and “Remove Rigidbody on Grab”
6. If you want to be able to Stash this weapon in a character’s “Stash” (inventory), enable “Is Stashable”.
7. On the “Interactive” tab, configure at least the “Hand Hold Primary Relative Position”.
8. Optionally configure “Gravity Settings”
9. On the “Weapon” tab, set the “Weapon Type”
10. Under “General Settings”, configure the “Fire Direction” (default forward on the z-axis) and “Relative Fire Position”.

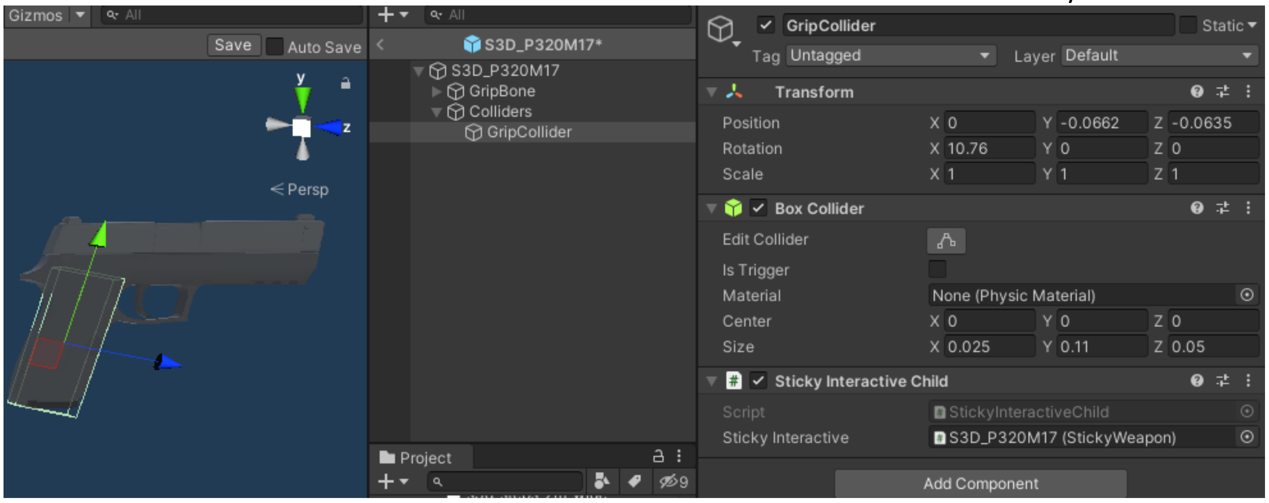

Spent cartridge
Eject pos & dir
Fire position & direction
Hand Hold Relative
Position Offset & Rotation
3rd party weapon not included

11. Add the appropriate Projectile or Beam prefab.
12. Set the Firing Button 1, button, type and interval (most weapons will only use firing button 1)
13. If you don't already have a Scriptable Object for "Ammo Types" used for weapons in your scene, create one now in the Project pane of the Unity Editor, by finding a suitable game folder, and clicking "+" then Sticky3D -> Ammo Types. You can then give them custom names and properties to suit your game. You only need one Ammo Type scriptable object as it is shared by all weapons. Typically, we don't recommend adding game content directly to the Sticky3DController folder.
14. If you don't already have a Scriptable Object of "Mag Types", create one now in the Project pane of the Unity Editor, by finding a suitable game folder, and clicking "+" then Sticky3D -> Mag Types. You only need one Mag Types object as it is shared by all weapons.
15. Under "Ammo Settings", add in your Ammo Types and Mag Types scriptable objects.
16. For Firing Button 1, add at least one compatible Ammo Type (use the + and – buttons next to "Refresh").
17. For Firing Button 1, select the Magazine Type.
18. Configure the reload settings.
19. Under "Animate Settings", if your weapon comes with animations, add the "Weapon Animator" and any "Weapon Actions"



Weapon Animator: M1911 Handgun_Model (Animator)
Animation Actions: 01
Animate Action 01
Weapon Action: Fire 1
Parameter Type: Trigger
Parameter Name: Fire
Trigger Value: Has Fired 1
Conditions: 00

20. If you have any S3D character animations that need to be used when characters hold or use this weapon, create one or more "Weapon Anim Set" scriptable objects in the Unity Editor Project pane. For more information see "Sticky Weapon – Animate Properties", "Weapon Anim Sets (for characters)" later in this chapter.
21. On the Weapon, under "Attachment Settings", if you want a laser sight, if one doesn't already exist, create a transform as a child of the weapon near the end of the barrel where the "Laser Sight" should shine from. Add this to the "Laser Sight" setting in the weapon inspector.
22. If you want a Scope to get a zoomed in-view of distant objects, configure that using the instructions in the "Sticky Weapon – Attachments" section below.
23. If you want your weapon to heat up (and cool down) as a result of firing, under the "Health and Heat Settings" make the appropriate changes.
24. If you want muzzle effects when firing, under "Muzzle FX Settings", add one or more Sticky Effects Module prefabs. As a starting point, make the Muzzle FX Offset the same as the "Relative Fire Position" you configured in "General Settings".
25. Optionally configure the "Smoke Settings". If this is a projectile weapon, you could use our "Sticky3D Smoke Effects Set1".
26. If you want to see spent or used ammo cartridges ejected from the weapon when fired, under "Spent Cartridge Settings" add a "Sticky Dynamic Module" prefab with a model of your empty ammo cartridge.
27. Create at least one primitive non-trigger collider for the weapon.
28. If the non-trigger colliders are children of the StickyWeapon transform, rather than on the same transform, add a Sticky Interactive Child component and hook them up with the parent StickyWeapon component.



GripCollider
Tag Untagged | Layer Default

Transform
Position X 0 Y -0.0662 Z -0.0635
Rotation X 10.76 Y 0 Z 0
Scale X 1 Y 1 Z 1

Box Collider
Edit Collider
Is Trigger
Material None (Physic Material)
Center X 0 Y 0 Z 0
Size X 0.025 Y 0.11 Z 0.05

Sticky Interactive Child
Script StickyInteractiveChild
Sticky Interactive S3D_P320M17 (StickyWeapon)

Sticky Weapon – Interactive Properties

See the "Sticky Interactive" chapter for a description of these properties.

Sticky Weapon – General (Common) Properties

Property	Description
Weapon Type	The type or style of weapon e.g., Projectile Raycast/Standard, Beam etc. We do not support changing the weaponType after it has been initialised.
Fire Direction	The local space direction in which the weapon fires [Default: Forward]
Relative Fire Position	The local space fire position aligned to the end of the barrel.
Fire Position Offsets	An array of local space fire position offsets from the relative fire position. Typically, only used for a weapon with multiple barrels.
Max Range	The maximum range the weapon can shoot.
Firing Button 1	The main trigger for the weapon to fire e.g. None, Manual, AutoFire. Set to None by default so that weapon doesn't automatically fire when a weapon is added at runtime.
Fire Type 1	The main firing type of this weapon. Semi-Auto fires single shots but automatically loads the next projectile if there is available ammo. Full-Auto can continuously fire while there is available ammo.
Only Fire When Aiming 1	Fire button 1 can only fire when the weapon is being aimed.
Firing Button 2	The second trigger for the weapon to fire e.g. None, Manual, AutoFire. Set to None by default so that weapon doesn't automatically fire when a weapon is added at runtime. Most weapons will only use the first firing button or trigger.
Fire Type 2	The secondary firing type of this weapon. Semi-Auto fires single shots but automatically loads the next projectile if there is available ammo. Full-Auto can continuously fire while there is available ammo.
Only Fire When Aiming 2	Fire button 2 can only fire when the weapon is being aimed.
Hit Layer Mask	When fired, the weapon can hit any objects in these Unity Layers.
Only Use When Held	The weapon can only fire, reload, or be animated, when it is held by a character.

Sticky Weapon – General (Beam Standard) Properties

Property	Description
Beam Prefab	Prefab of the beam fired by this weapon. Beam prefabs need to have a Sticky Beam Module script attached to them.
Charge Amount	The amount of charge or power the beam weapon has.
Recharge Time	The time (in seconds) it takes the fully discharged beam weapon to reach maximum charge.

Sticky Weapon – General (Projectile Standard) Properties

Property	Description
Projectile Prefab	Prefab of the projectiles fired by this weapon. Projectile prefabs need to have a Sticky Projectile Module script attached to them. For more details look at the chapter for that module.

Sticky Weapon – General (Projectile Raycast) Properties

Property	Description
Projectile Prefab	Prefab of the projectiles fired by this weapon. Projectile prefabs need to have a Sticky Projectile Module script attached to them. For more details look at the chapter for that module.

Sticky Weapon – Aiming and Reticle Properties

Reticles are small sprites placed on the screen to assist a character to look or aim in a certain direction. When a weapon is held by a player, these are typically shown on a Sticky Display Module (a.k.a. HUD). If a HUD is found in the scene, the appropriate reticle for this weapon will be displayed. There are also options to NOT display a reticle under certain conditions.

Property	Description
Aim First Person FoV	When aiming the held weapon, this is the character first person camera field of view.
Aim Third Person FoV	When aiming the held weapon, this is the character third person camera field of view unless “TP Aim uses FP Camera” is enabled in the “Weapon Anim Set”.
Aim Speed	The rate at which the transition between aiming and not aiming will progress.
Default Reticle	The reticle (sprite) used when the weapon is held by a S3D player
Aiming Reticle	The reticle (sprite) used when the weapon is held by a S3D player and aiming. If none, it will fall back to defaultReticleSprite unless No Reticle when “Aiming the Weapon [FPS/TPS]” is true.
No Reticle When:	
Aiming the Weapon FPS *	Do not display a reticle on the StickyDisplayModule (HUD) when the weapon is aiming with first-person camera. Only applies if weapon is held by a player (non-NPC) character and there is an active StickyDisplayModule in the scene.
Aiming the Weapon TPS *	Do not display a reticle on the StickyDisplayModule (HUD) when the weapon is aiming with third-person camera. Only applies if weapon is held by a player (non-NPC) character and there is an active StickyDisplayModule in the scene.
Held by a Player *	Do not display a reticle on the StickyDisplayModule (HUD) when the weapon is held. Only applies if weapon is held by a player (non-NPC) character and there is an active StickyDisplayModule in the scene.
Held with Free Look OFF *	Do not display a reticle on the StickyDisplayModule (HUD) when the weapon is held and Free Look on the character is NOT enabled. Only applies if weapon is held by a player (non-NPC) character and there is an active StickyDisplayModule in the scene.

* When the Reticle is not shown, the default pointer (cursor) may be visible. To prevent the pointer appearing in place of the Reticle, ensure “Auto-hide Cursor” is NOT enabled on the character “Look” tab or the HUD “General Settings”.

Sticky Weapon – Ammo Properties

These are the properties that apply to projectile weapons. It includes:

- Types of compatible ammo
- Weapon reloading
- Magazine (a.k.a. clip) details – for magazine attach points, see Attachment Settings below.

There are two possible firing button mechanisms: primary (1), and secondary (2). Almost all weapons will only have a primary firing button. The second firing button is mainly for use with custom weapons which require a decent amount of programming knowledge to create a class which inherits from StickyWeapon.cs.

Property	Description
Available Ammo Types	All weapons and magazines in your scene should use the same (common) set of ammo types. You could use our Demo Ammo Types scriptable objects OR create your own unique set by going to the Project pane, clicking "+" then Sticky3D -> Ammo Types. You can then give them custom names and properties to suit your game.
Available Mag Types	All weapons and magazines in your scene should use the same (common) set of magazine types. You could use our Demo Mag Types scriptable objects OR create your own unique set by going to the Project pane, clicking "+" then Sticky3D -> Magazine Types. You can then give them custom names to suit your game.
Compatible Ammo Type	Each Firing Button can use one or more types of ammo. For example, the primary firing button (1) could use standard 9 mm and 9 mm hollow point ammo.
Magazine Type	The type of Magazine that can fit this weapon primary (Firing Button 1) or secondary mechanisms. Most weapons only use Firing Button 1.
Is Sticky Mag Required	Does the firing mechanism require a StickyMagazine to be attached before it can fire? See also "Weapon Mag Setup" under the "Sticky Weapon – Attachments" section later in this chapter.
Unlimited Ammo	Can this projectile weapon keep firing and never run out of ammunition? This does not apply when "Is Sticky Mag Required" is true, as the amount of ammo is determined by the amount of ammo in the current StickyMagazine attached to the weapon (if any).
Ammunition	The quantity of projectiles or ammunition available for this weapon. This is read-only and automatically updated when "Is Sticky Mag Required" is true.
Reload Type	The method for reloading the button mechanism.
[Manual Only]	The firing mechanism (primary [1], or secondary [2]) can only be reloaded by calling: stickyWeapon.Reload (weaponButtonNumber) It can be called either in code or via a Sticky Input Module, Custom Input.
[Auto Reserve]	Get the spare (full) magazines from an invisible "reserve". The number in "reserve" is controlled by the "Mags in Reserve" property.
[Auto Stash]	Stash used mags when reloading. Get the first compatible magazine from either the character's hand or Stash (inventory) that has at least 1 piece of compatible ammo in it. This has no effect if the weapon is not held by a S3D character.
[Auto Stash With Drop]	Drop used mags when reloading. Get the first compatible magazine from either the character's hand or Stash (inventory) that has at least 1 piece of compatible ammo in it. This has no effect if the weapon is not held by a S3D character.
Reload Delay	The short delay, in seconds, between when the button mechanism fired, and it attempts to reload.
Reload Equip Delay	The delay during reloading, in seconds, between when the mechanism unequips the old mag and starts to equip the new mag.
Reload Duration	The time, in seconds, it takes the firing mechanism to reload (including any Equip Delay).
Reload Sound FX	The Sound FX for the firing mechanism used when the weapon begins reloading. The Effects Type must be "SoundFX". See the Sticky Effects Module chapter for more details.

Property	Description
Reload Equip Sound FX	The Sound FX for the firing mechanism used when the weapon equips a new mag during reloading. The Effects Type must be SoundFX. This can be useful if you want a separate sound for when reloading starts (Reload Sound FX above) and then another sound when the new mag is equipped or fitted to the weapon. See the Sticky Effects Module chapter for more details.
Mag Capacity	The amount of ammo the primary firing mechanism magazine (clip) can hold. Becomes read-only when “Is Sticky Mag Required” is true. It is automatically updated when a magazine is retrieved from Stash.
Unlimited Mags	Is there an unlimited number of magazines with which to keep reloading? This is not available when the Reload Type is “Auto Stash”.
Mags In Reserve	The number of additional magazines available for the firing mechanism. This is read-only and automatically updated when Reload Type is Auto Stash.
Fire Empty Effects Set*	The set containing one or more StickyEffectsModules to be randomly selected when attempting to fire with no available ammo on fire button 1. Ensure “Is Reparented” is NOT selected on each prefab.

* To create an Effects Set, perform the following tasks:

1. In the Unity Editor Project pane, find a suitable folder for your game, and click “+” then Sticky3D -> Effects Set. Typically, we don’t recommend adding your own content to the Sticky3DController folders.
2. Give the new scriptable object a meaningful name. e.g., Weapon Empty Fire Effects Set1
3. Click the “+” button on the new Effects Set
4. Drag an existing StickyEffectsModule prefab into the new slot. E.g., “StickyEmptyFireFX1” from the Demos\Prefabs\Effects folder.
5. Repeat steps 3 and 4 for additional empty fire effects that you want to include in the set.

Sticky Weapon – Animate Properties

Weapon animate setting allow you to control weapon and/or character animations. For example, you might want to animate the weapon model when it fires. Some 3rd party weapons (like ones you purchase from the Unity Asset Store), come with an animator and animations. Otherwise, you can configure your Unity Animator and make some animations using the Unity Animation editor.

We have included the S3D_P320M17 projectile handgun that has a firing animation. See the project Demos\Prefabs\Weapons folder.

Your humanoid character most likely already is set up with idle, walk, and run animations. When your character grabs a weapon, you probably want to use different idle, walk and run animations. For example, when idle, the character should be holding the weapon out in front of them. Weapon Anim Sets help you replace existing character animations to match the weapon their grab (without you writing any code). They can also send weapon data to the animation controller on a character.

Property	Description
Weapon Animator	The animator that will control animations for this weapon. This should be attached to, or a child of, the weapon gameobject
Weapon Actions	The animate actions that interact with your weapon animation controller.
Weapon Anim Sets (for characters)	These can replace animations on your S3D character and can also send weapon data to the character animation controller.

Weapon Actions

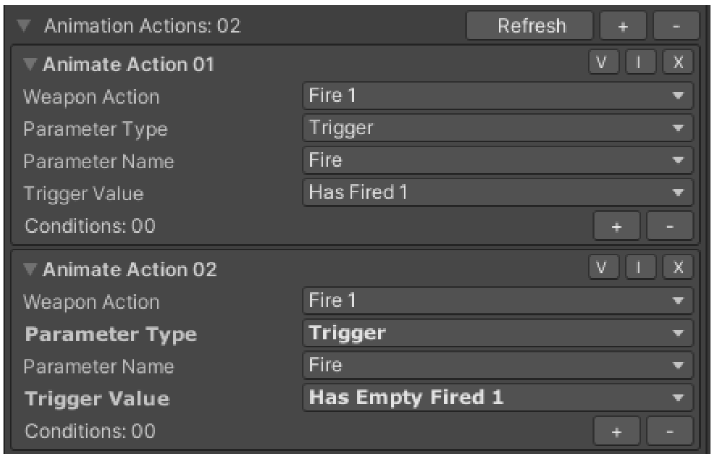
Each Weapon Animate Action sends data from your StickyWeapon component to the weapon animation controller.

Action Property	Description
Weapon 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. For more information see Custom 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 weapon that will be sent to the model’s animation controller
Float Value	The real-time float value from the weapon that will be sent to the model’s animation controller
Trigger Value	The real-time trigger value from the weapon that will be sent to the model’s animation controller
Integer Value	FUTURE – let us know if you have a use-case and we can add support for it!
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.

If your weapon has an animation for firing, you could create two actions. One for when it has fired (Has Fired 1) and one when there was no ammo available when the character attempted to fire (Has Empty Fired 1).

Note, that “Has Fired 1” only occurs when there is ammo available on the primary (1) firing mechanism.

For this weapon, we want it to run the “Fire” animation in both cases.



Animate Action 01

Weapon Action	Fire 1
Parameter Type	Trigger
Parameter Name	Fire
Trigger Value	Has Fired 1
Conditions: 00	

Animate Action 02

Weapon Action	Fire 1
Parameter Type	**Trigger**
Parameter Name	Fire
Trigger Value	**Has Empty Fired 1**
Conditions: 00	

Custom Actions

Custom Actions are similar to other Weapon Actions in that they change Parameter values in your Unity Animator. However, instead of getting the real-time value from data inside StickyWeapon, they get it from via an S3D API method like stickyWeapon.SetCustomAnimActionTriggerValue (..) or SetCustomAnimActionFloatValue(..).

For an example, see Demos\Scripts\SampleWeaponCustomAnimAction.cs.

Weapon Anim Sets (for characters)

These are scriptable objects you create in the Unity Editor Project pane. You will probably require different animation for your characters when they hold different weapons. You might also need different sets of animations for different characters holding the same weapons.

Weapon Anim Sets allow you to achieve this without writing code (although you'll still need to have a way of making or obtaining the different animations required – sorry, we can't do everything for you).

Some sample Weapon Anim Sets are included in the project Demos\AnimClipSets folder.

To create a Weapon Anim Set:

1. In the Unity Editor Project pane, find a suitable folder for your game, click "+" menu, and Sticky3D -> Weapon Anim Set. Typically, we don't recommend adding content to the Sticky3DController folder.
2. Give the new scriptable object a meaningful name. e.g., Enemy NPC Weapon Anim Set1
3. Decide which animations you want to replace for this weapon when used by the Enemy NPC.
4. Click "+" next to the Anim Clip Pairs
5. Add the original animation clip from the character's animation controller.
6. Add the replacement animation clip that you want the character to use when using the weapon.
7. Add extra pairs to cater for all animations you want to use for this weapon for the Enemy NPC.
8. If you want your character to play particular animations when say firing or reloading a weapon, you can add some Anim Actions. These will send data from your weapon directly to the character's animation controller (you can also animate the weapon itself by using "Weapon Actions" discussed earlier in this section.

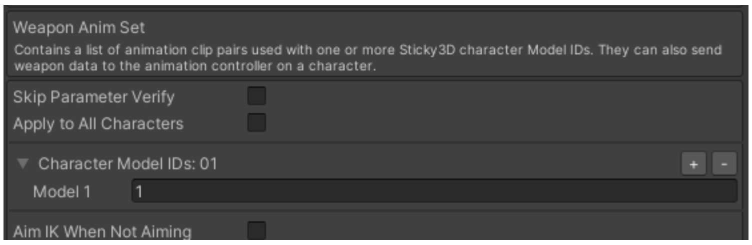
Although you can achieve similar outcomes with many different Animation Controller setups, there are two main approaches to playing animations when holding a weapon in Sticky3D.

Option 1:

Replace all your "regular" animations with ones specifically designed for the character and weapon pair.

An example is the "Sticky3D Weapon Anim Set Bob P320". This is used when Bob picks up a P320M17 handgun.

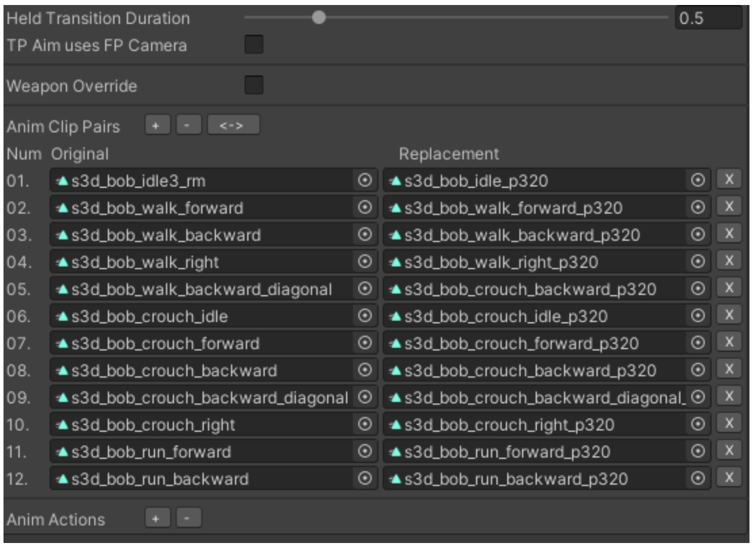
In this example, the Bob model (Model ID 1) needs to have 12 animations replaced.



Weapon Anim Set

Contains a list of animation clip pairs used with one or more Sticky3D character Model IDs. They can also send weapon data to the animation controller on a character.

Skip Parameter Verify
Apply to All Characters
Character Model IDs: 01
Model 1: 1
Aim IK When Not Aiming



Held Transition Duration: 0.5
TP Aim uses FP Camera
Weapon Override
Anim Clip Pairs

Num	Original	Replacement
01.	s3d_bob_idle3_rm	s3d_bob_idle_p320
02.	s3d_bob_walk_forward	s3d_bob_walk_forward_p320
03.	s3d_bob_walk_backward	s3d_bob_walk_backward_p320
04.	s3d_bob_walk_right	s3d_bob_walk_right_p320
05.	s3d_bob_walk_backward_diagonal	s3d_bob_crouch_backward_p320
06.	s3d_bob_crouch_idle	s3d_bob_crouch_idle_p320
07.	s3d_bob_crouch_forward	s3d_bob_crouch_forward_p320
08.	s3d_bob_crouch_backward	s3d_bob_crouch_backward_p320
09.	s3d_bob_crouch_backward_diagonal	s3d_bob_crouch_backward_diagonal_
10.	s3d_bob_crouch_right	s3d_bob_crouch_right_p320
11.	s3d_bob_run_forward	s3d_bob_run_forward_p320
12.	s3d_bob_run_backward	s3d_bob_run_backward_p320

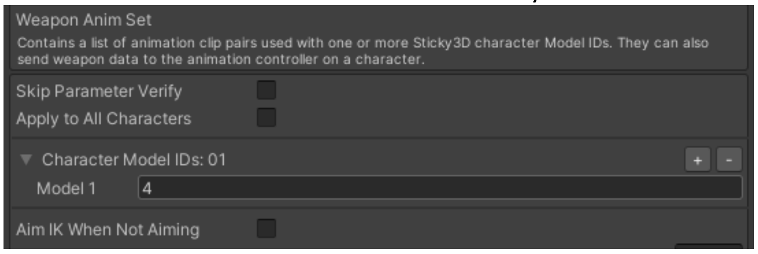
Anim Actions

Option 2:

An example is the “Sticky3D Weapon Anim Set Jane Suited P320”. This is used when the Suited Jane character picks up a P320M17 handgun.

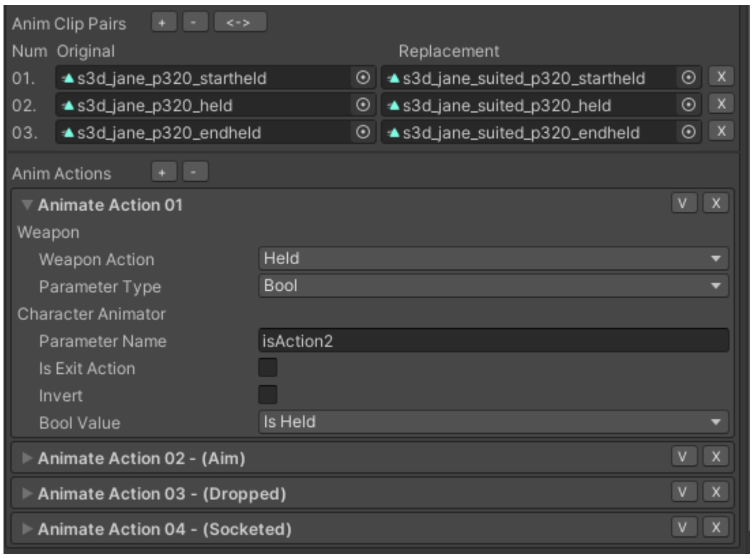
It uses a separate Animator Layer within the Animation Controller with an upper body mask. Only a few animations are used to control arm, head and hand positions.

Some character Animate Actions to tell the Animator what to do when a weapon is held.



Weapon Anim Set
Contains a list of animation clip pairs used with one or more Sticky3D character Model IDs. They can also send weapon data to the animation controller on a character.

Skip Parameter Verify ☐
Apply to All Characters ☐
Character Model IDs: 01
Model 1: 4
Aim IK When Not Aiming ☐



Anim Clip Pairs

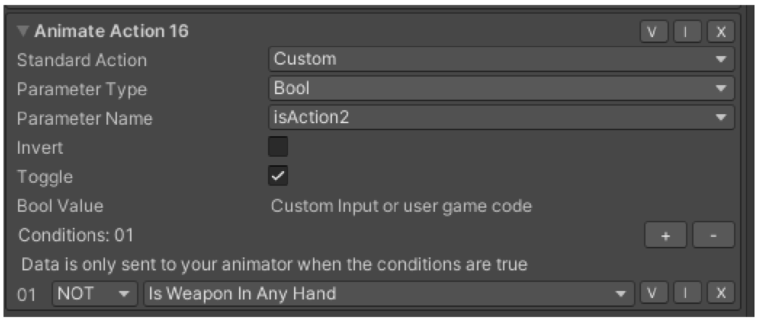
Num	Original	Replacement
01.	s3d_jane_p320_startheld	s3d_jane_suited_p320_startheld
02.	s3d_jane_p320_held	s3d_jane_suited_p320_held
03.	s3d_jane_p320_endheld	s3d_jane_suited_p320_endheld

Anim Actions

Animate Action 01
Weapon
Weapon Action: Held
Parameter Type: Bool
Character Animator
Parameter Name: isAction2
Is Exit Action ☐
Invert ☐
Bool Value: Is Held

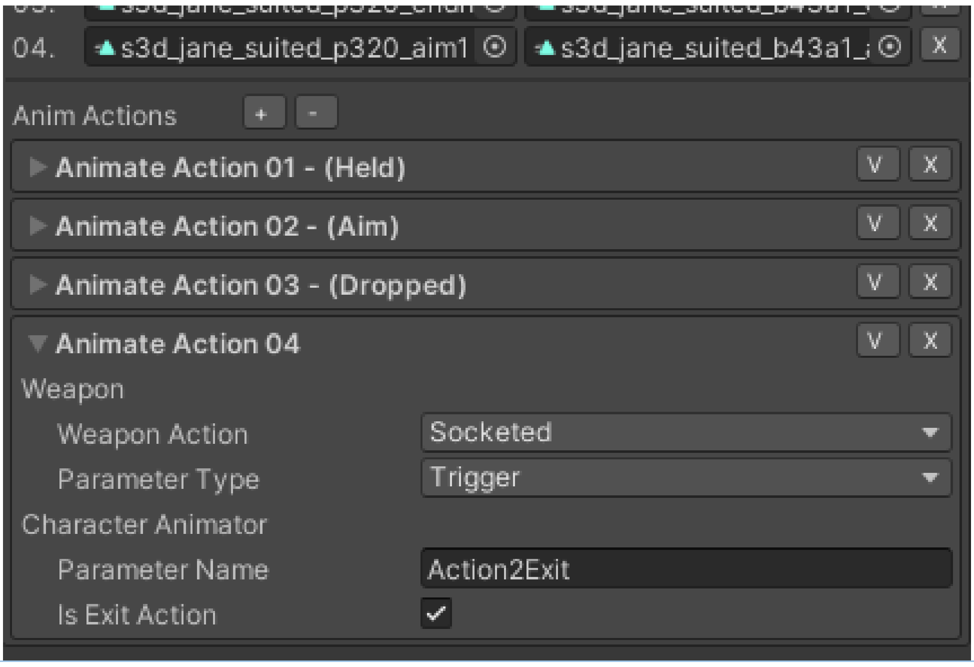
Animate Action 02 - (Aim)
Animate Action 03 - (Dropped)
Animate Action 04 - (Socketed)

If you have an Animate Action on your character Animate tab that sets the same Parameter Name, you probably will want to add a condition to prevent it being overwritten when a weapon is being held.



Animate Action 16
Standard Action: Custom
Parameter Type: Bool
Parameter Name: isAction2
Invert ☐
Toggle ✓
Bool Value: Custom Input or user game code
Conditions: 01
Data is only sent to your animator when the conditions are true
01 NOT Is Weapon In Any Hand

Anim Set Property	Description
Skip Parameter Verify	Should anim action parameter name verification be skipped when the set is applied to a weapon? It can be faster, but will cause errors if parameter names don't exist in the character’s animator. Once you have it all working correctly, you could turn this off to improve performance.
Apply to All Characters	Should this be applied to all characters?
Character Model IDs	An array of Sticky3D Controller Module Model IDs. A Model ID (found on the Engage tab) groups characters with similar attributes together. E.g., Same rig or animation controller.
Aim IK When Not Aiming	When the weapon is held by a character, it will always use Aim IK, even when it is not specifically being aimed.
Aim IK Turn Delay	The time, in seconds, to delay the character turning to face the target when aiming starts. This allows time for to transition from a held animation, to an aiming animation.
Aim IK Near Clipping	The first-person camera near clipping plane when aiming.
Aim IK Weapon Offset	The weapon local space offset from the first-person camera when aiming
Clip Revert Delay	The number of seconds to delay reverting animation clips for a character to their originals when the weapon is dropped.

Anim Set Property	Description
Free Look When Held	If a non-NPC character has Free Look enabled before the weapon is held (but not aimed), does Free Look remain enabled? Useful when a weapon is held in a relaxed pose not pointing forward.
Held Transition Duration	When grabbing a weapon, this is the time, in seconds, it takes the animation to play that transitions from not holding, to holding the weapon. You can get this time from your Animation Controller in the Unity Editor, or from the animation duration itself. If in doubt, use a short time like 0.5 seconds. This DOES NOT change the animation duration or transition time.
TP Aim uses FP Camera	In third-person games, it can be a challenge to precisely aim a weapon. This option, provided the character has a first-person camera configured, will switch to first-person when the character is aiming the weapon. This is particularly useful if the weapon is equipped with a scope.
Weapon Override	Should weapon settings be overridden when the character picks up this weapon?
Only Fire When Aiming 1	Fire button 1 can only fire when the weapon is being aimed
Only Fire When Aiming 2	Fire button 2 can only fire when the weapon is being aimed
Anim Clip Pairs	
Original	The original animation clip from the character's animation controller. This will also be the one restored when the weapon is dropped or stashed.
Replacement	The replacement animation clip that you want the character to use when using the weapon
Anim Actions	
Weapon 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, by creating a new (custom) weapon class and overriding the AnimateCharacterCustomAction(..) method.
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
Is Exit Action	Used for weapon actions like dropped, equipped, socketed or stashed to trigger an immediate exit from an Animation Layer while holding a weapon. For example, when equipping a held weapon in a holster, you might want the holster animation to run to completion (with say a transition Exit Time set to 0.99 in the Animator Controller), but immediately exit when the weapon is dropped or socketed. To achieve this, you'd set the Exit Time in the Animator transition to 0.99, but add two Animate Actions to the Sticky Weapon Anim Set. 

04. s3d_jane_suited_p320_aim1 | s3d_jane_suited_b43a1_

Anim Actions

Animate Action 01 - (Held)

Animate Action 02 - (Aim)

Animate Action 03 - (Dropped)

Animate Action 04

Weapon

Weapon Action: Socketed

Parameter Type: Trigger

Character Animator

Parameter Name: Action2Exit

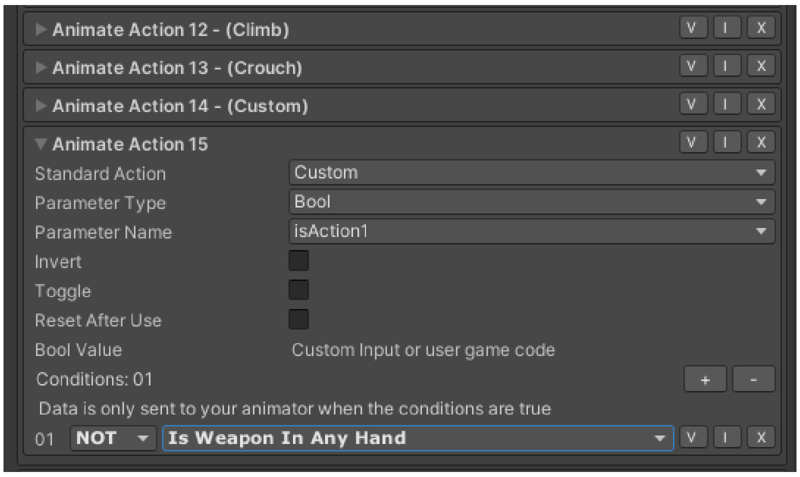
Is Exit Action: ✓

Anim Set Property	Description
	See the weapon anim set "Sticky3D Weapon Anim Set Jane Suited B43A1" and the "s3d_jane_suited_ik" animator for an example.
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 weapon that will be sent to the model's animation controller
Float Value	The real-time float value from the weapon that will be sent to the model's animation controller
Trigger Value	The real-time trigger value from the weapon that will be sent to the model's animation controller
Integer Value	FUTURE – let us know if you have a use-case and we can add support for it!
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. By default, this is true and you should set Boolean values back to false after animate actions are processed each frame.

IMPORTANT: Weapon Anim Sets directly update the character's animation controller.

When using "Anim Actions" in a Weapon Anim Set, care must be taken not to cause conflict with existing "Anim Actions" on the "Animate" tab of a character.

This can be achieved by placing a condition on the potentially conflicting "Anim Action" on the character's Animate tab.


Animate Action 12 - (Climb)
Animate Action 13 - (Crouch)
Animate Action 14 - (Custom)
Animate Action 15
Standard Action: Custom
Parameter Type: Bool
Parameter Name: isAction1
Invert
Toggle
Reset After Use
Bool Value: Custom Input or user game code
Conditions: 01
Data is only sent to your animator when the conditions are true
01 NOT Is Weapon In Any Hand

Sticky Weapon – Attachments

This includes information and settings about items that attach to the weapon.

Weapon Attachment Properties

Property	Description
Laser Sight	The child transform that determines where the laser sight aims from.
Laser Sight On	Should the laser sight beam be showing?
Laser Sight Auto On	Should the laser sight beam automatically turn on when it is equipped?
Laser Sight Colour	The colour of the laser sight beam.
Mag Attach Point 1	The child Transform where the primary magazine (clip) attaches to the weapon.

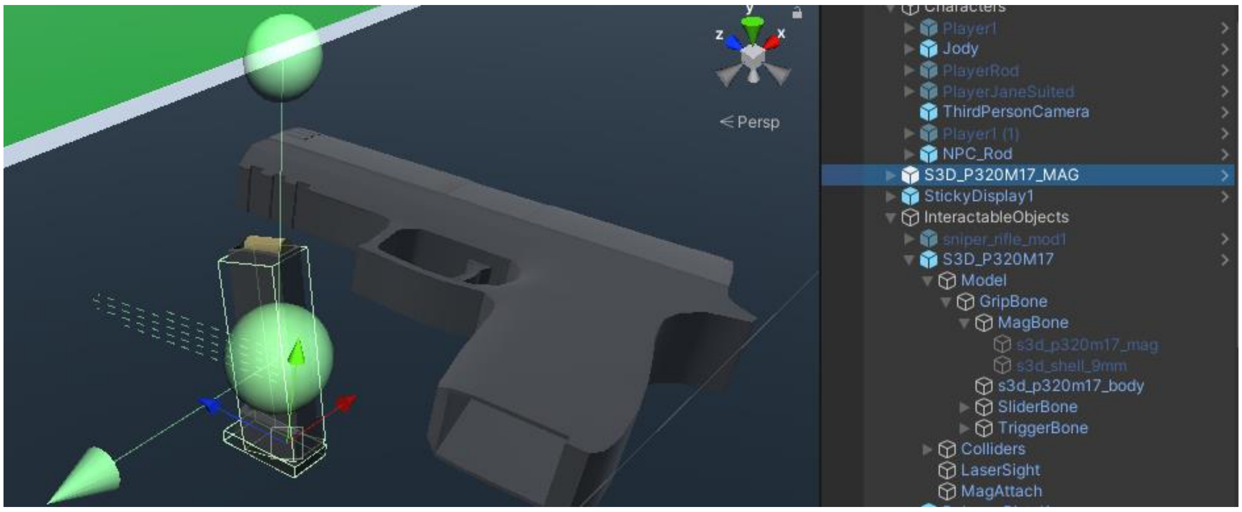
Property	Description
Mag Attach Point 2	The child Transform where the secondary magazine (clip) attaches to the weapon.
Scope On	Should the Scope currently be used for more precise visual aiming?
Scope Camera	The camera used to render the scope display
Scope Renderer	The child mesh renderer to project the Scope Camera onto. Use the "F" button to highlight the demo prefab weapons folder where we have some examples. In your weapon prefab, make sure the Scope Renderer does not have a collider, else you'll see an error at runtime.
Scope Auto On	Should the Scope be automatically turned on when grabbed by a character? Essentially, it will stay on all the time the weapon is held.
Scope Allow NPC	Allow NPCs to use the Scope. Typically, you don't want to enable this as each Scope used in the scene requires the whole scene to be rendered again which has a significant performance overhead.
Torch	TODO

Weapon Mag Setup

You may wish your weapon to have interactive magazines (also known as a "clips"). This allows your character to find mags in your scene, pick them up, stash them, reload weapons, and drop them.

The steps include:

1. Configure the magazine as an interactive item (See chapter "Sticky Magazine")
2. On the weapon prefab, in the "Ammo Settings" section on the "Weapon" tab, ensure "Is Sticky Mag Required" is enabled for the firing button
3. Set the "Reload Type" to "Auto Stash" or "Auto Stash with Drop"
4. In the "Attachment Settings" section on the "Weapon" tab, set the "Mag Attach Point"
5. If your weapon model includes a magazine mesh, ensure it is disabled (as this won't be used with this setup)



6. If you want the weapon to start with a mag, add the magazine prefab under the Mag Attach Point transform within the weapon prefab. When the weapon is initialised, the weapon will recognise the magazine. Ensure your StickyMagazine "Mag Type" matches the weapon's "Magazine Type".

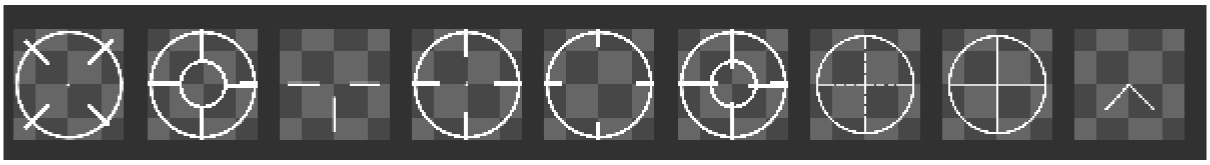
Weapon Scope Setup

To add a Scope to an existing weapon prefab, perform the following tasks:

1. Open the weapon prefab
2. Add a scope display prefab as a new child object inside the prefab from Demos\Prefabs\Weapons. Click "F" button in the inspector for a shortcut to the demo folder. E.g., S3D_Scope_Display64mm (there is also a smaller 32mm version if this is too big).

3. Remove the Box Collider from the S3D Scope Display child object (it is only included in the prefab to comply with Asset Store publishing rules)
4. Position the new child object behind the scope on the weapon. The child object should be facing “backward” on the weapon. That is, facing the character when the weapon is held.
5. If the display is too big for your scope, try changing it for a smaller one of our prefabs. If it is still too big, scale the child gameobject (e.g., S3D_Scope_Display32mm) on the X and Y axis.
6. On the “Weapon” tab of the Sticky Weapon component, expand “Attachment Settings” and either hook up the “Scope Camera” or click “New”.
7. The S3D_Scope_Display64mm prefab material uses a render texture (Demos\Textures\Weapons\s3d_scope1). Add this to the Scope Camera “Target Texture”.
8. Optionally, uncheck the “Ignore Raycast” Unity layer in the camera “Culling Mask”. If you’ve set Sticky Beam Module prefabs to the “Ignore Raycast” Unity layer, this will prevent the beam from appearing in the scope when firing (which may look a little odd).
9. Save and exit with weapon prefab.
10. On your Sticky3D player character(s), go to the Sticky Input Module, add a new Custom Input, configure a new button (like Mouse 1 for right mouse button), add a new event, drag the character parent object in the Object field, and set the function to “StickyControlModule.ToggleAimWeapon”. For the weapons held in the right hand, leave the bool field unticked.

Scope display prefabs have a child S3DUIScope overlay which sits slightly in-front of the render texture. You could duplicate an existing prefab, copy it to your game project folder, and replace the S3DUIScope sprite with another one from the Demos\Textures\Scopes folder. Be sure to copy the child transform position and scale onto the new S3DUIScope child object before deleting the original, otherwise it will not render correctly.



Sticky Weapon – Health and Heat Properties

Property	Description
Health	The overall health of the weapon. Must be in the range 0 to 100.
Heat Level	The heat of the weapon - range 0.0 (starting temp) to 100.0 (max temp).
Heat Up Rate	The rate heat is added per second for beam weapons. For projectile weapons, it is inversely proportional to the firing interval. If rate is 0, heat level never changes.
Cool Down Rate	The rate heat is removed per second. This is the rate the weapon cools when not in use.
Overheat Threshold	The heat level that the weapon will begin to overheat and start being less efficient. Beam weapons loose charge up to twice as fast when they are overheating.
Burnout on Max Heat	When the weapon reaches max heat level of 100, will the weapon be inoperable until it is repaired?
Firing Paused	Is firing paused? This can be useful when only wanting to prevent the weapon from firing like when displaying a Sticky popup.

Sticky Weapon – Muzzle FX Properties

This includes information and settings about muzzle effects that occur when the weapon primary mechanism fires.

Property	Description
Muzzle FX Offset	The distance in local space that the muzzle Effects Object should be instantiated from the weapon firing point. Typically, only the z-axis will be used.

Property	Description
Effects Objects	An array of randomly selected Sticky Effects Modules spawned when the weapon fires. Beams should use looping particle systems.

Sticky Weapon – Recoil Properties

This includes information and settings for optional recoil actions that occur when the weapon primary or secondary mechanism fires.

This feature currently only works with in first-person with player characters while aiming. It is our intention to have this work for most if not all scenarios.

Property	Description
Recoil Speed	The rate at which the weapon recoils when fired. NOTE: Currently only applies to FPS players when aiming.
Return Rate	The rate at which the weapon returns to its stable position.
Fire Button 1 or 2	The settings for the primary (1) or secondary (2) firing mechanism
Recoil Angle X	The angle the weapon pitches up when the firing.
Recoil Angle Y	The maximum angle the weapon rotates around the local y-axis when firing.
Recoil Angle Z	The maximum angle the weapon rotates around the local z-axis when firing.
Recoil Kick Back	The maximum distance, in metres, the weapon will kick back on the local z-axis when firing.

Sticky Weapon – Smoke Properties

This includes information and settings about smoke effects that occur when or slightly after the weapon fires. Smoke is fired from the

Property	Description
Max Active Smoke FX	The maximum number of smoke effects that can be active at any time on this weapon.
Smoke Effects Set 1	The set containing one or more StickyEffectsModules to be randomly selected when primary mechanism is fired.
Smoke Effects Set 2	The set containing one or more StickyEffectsModules to be randomly selected when secondary mechanism is fired.

Typically, each StickyEffectsModule prefab used in the sets, should have:

- Effects Type set to Default
- Is Reparented enabled
- A non-looping particle system with an optional Start Delay. Generally, a delay isn't required as the particle system takes a moment to get under way.

To create a Smoke Effects Set:

1. In the Unity Editor Project pane, find a suitable folder for your game, and click "+" then Sticky3D -> Effects Set. Typically, we don't recommend adding your own content to the Sticky3DController folders.
2. Give the new scriptable object a meaningful name. e.g., Weapon Smoke Effects Set1
3. Click the "+" button on the new Effects Set
4. Drag an existing StickyEffectsModule prefab into the new slot. E.g., "StickySmokeFX1" from the Demos\Prefabs\Effects folder.
5. Repeat steps 3 and 4 for additional smoke effects that you want to include in the set.

Sticky Weapon – Spent Cartridge Properties

This includes information and settings about spent cartridges that can be ejected when the weapon primary mechanism fires.

Property	Description
Spent Cartridge Prefab	The dynamic object prefab for the spent or empty cartridge that is ejected from the weapon after it fires. It must have a StickyDynamicModule component on the parent gameobject.
F (gizmo) Button	Located on the right of the inspector, this will "Find" (select) the gizmo in the scene view that is used to edit the Spent Eject Position and Direction.
G (gizmo) Button	Toggle the Spent Eject gizmo on/off in the scene view.
Spent Eject Position	The local space position on the weapon from which the spent cartridge is ejected.
Spent Eject Direction	The local space direction the spent cartridge is ejected
Spent Eject Rotation	The local space rotation, in Euler angles, of the spent cartridge when ejected. This is not shown by the gizmos in the scene view.
Spent Eject Force	The force used to eject the spent cartridge from the weapon in the Spent Eject Direction.

Sticky Weapon – Events

Events are triggered when certain things happen to the weapon. 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 Grabbed	These are triggered by a S3D character when they grab this weapon.
On Dropped	These are triggered by a S3D character when they drop this weapon.
On Hover Enter	These are triggered by a S3D character when they start looking at this weapon.
On Hover Exit	These are triggered by a S3D character when they stop looking at this weapon.
On Post Grabbed	There are triggered by a S3D character immediately after they grab this weapon.
On Post Stashed	These are triggered by a S3D character immediately after they stash (place in inventory) this weapon.

Sticky Weapon – Conversion from Beam to Projectile

If you have already set up a weapon as a "Beam Standard" weapon but now wish to make a "Projectile Standard" variation, the process is quite simple. In this example we will convert the S3D_B43A1 to a pulse firing laser rifle.

1. In the Project panel, locate the prefab (S3D_B43A1) and duplicate it.
2. Rename the new prefab (e.g., S3D_B43A1_LaserPulse)
3. On the Weapon tab of the prefab, change the "Weapon Type" to "Projectile Standard".
4. Next to "Projectile Prefab", click the "F"ind button, expand the highlighted demo "Projectiles" folder and drag in the StickyProjectile3 prefab (see the "Sticky Projectile Module" chapter to create your own projectile prefabs).
5. Close the "General Settings" section on the "Weapons" tab.
6. You may have noticed two warnings above the "Ammo Settings" section. We'll fix that now. Expand the "Ammo Settings" section.
7. To the right of the "Available Ammo Types", click the selection icon and chose "Demo Ammo Types1" or your custom Ammo Types scriptable object.
8. To the right of the "Available Mag Types", click the selection icon and select and chose "Demo Mag Type1 or your custom Mag Types scriptable object.

9. Under “Fire Button 1”, set the “Compatible Ammo Type” to “laser pulse 1” (if using our demo ammo types)
10. This laser pulse rifle doesn’t have a separate removable mag (a.k.a. “clip”), so we’re don’t need to configure the Magazine settings. For an example of a projectile weapon that uses a removable mag, see the demo “S3D_P320M17B” prefab.
11. Close the “Ammo Settings” and open the “Muzzle FX Settings” section.
12. Set “Effects Objects” to 1
13. Add the “StickyMuzzleFX6” from the Demos\Effects folder.

Sticky Weapon – Custom Weapons

Custom weapons are an advanced subject for developers familiar with creating classes that inherit from parent classes (in this case StickyWeapon.cs and StickyWeaponEditor.cs).

Feel free to reach out to us on our Discord channel if you get stuck creating custom weapons.

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.
Point Weapon Held	Is pointing permitted when a weapon is held? By default, this is disabled so that the same button can be configured for both pointing and firing a weapon. For example, a controller trigger.

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

Interactive Property	Description
	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 Target sprite or 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 Target sprite or 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.

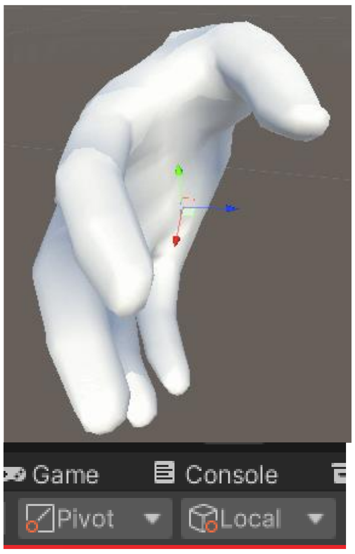
When teleporting is enabled, you may wish to disable “Vertical Move Input Axis” on Sticky Input Module to prevent the character moving forward or backward with continuous move. Similarly, disable “Horizontal Move Input Axis” for continuous left or right movement.

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 or target reticle
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.

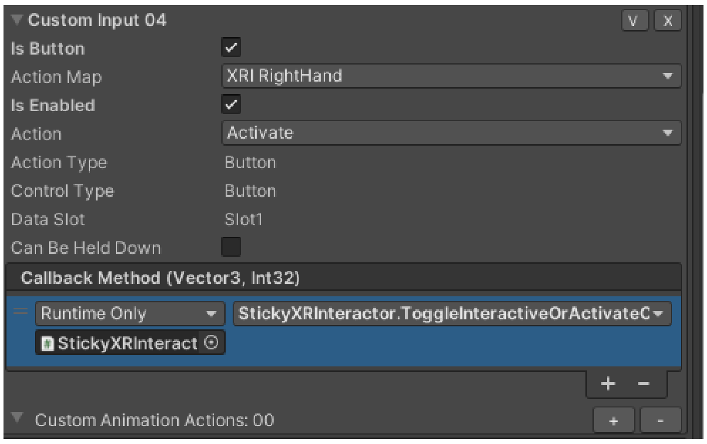


To grab an interactive-enabled object, you need to either touch the object (so, the Sticky Interactive object would also need to have IsTouchable enabled) or, you need to be looking at the object (so, you’d need to be able to toggle Look interactive on/off).

To grab items to that are being looked at, configure a button to toggle Look at interactive on/off.

There are two applicable toggle options. ToggleInteractiveOn and ToggleInteractiveOrActivateOn. The later also enables you to use additional features on an interactive-enabled object that is being held. Both will allow you to look at, then grab objects.

Here we use the XRI RightHand Activate action. <XRController>{RightHand}/triggerPressed



Custom Input 04

Is Button	✓
Action Map	XRI RightHand
Is Enabled	✓
Action	Activate
Action Type	Button
Control Type	Button
Data Slot	Slot1
Can Be Held Down	

Callback Method (Vector3, Int32)

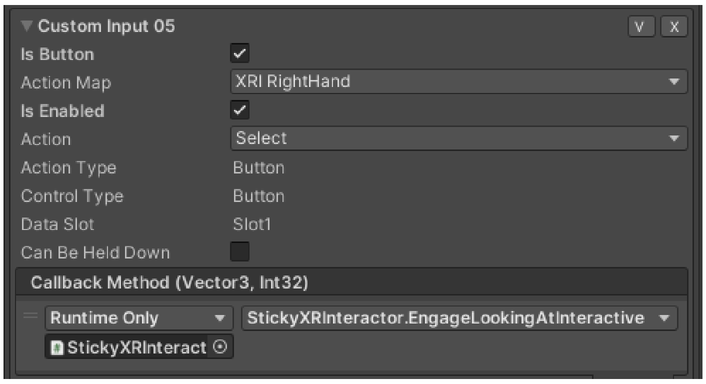
Runtime Only | StickyXRInteractor.ToggleInteractiveOrActivateC
StickyXRInteract

Custom Animation Actions: 00

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.

In the example to the right, the XRI RightHand Select action is used:

<XRController>{RightHand}/gripPressed



Custom Input 05

Is Button	✓
Action Map	XRI RightHand
Is Enabled	✓
Action	Select
Action Type	Button
Control Type	Button
Data Slot	Slot1
Can Be Held Down	

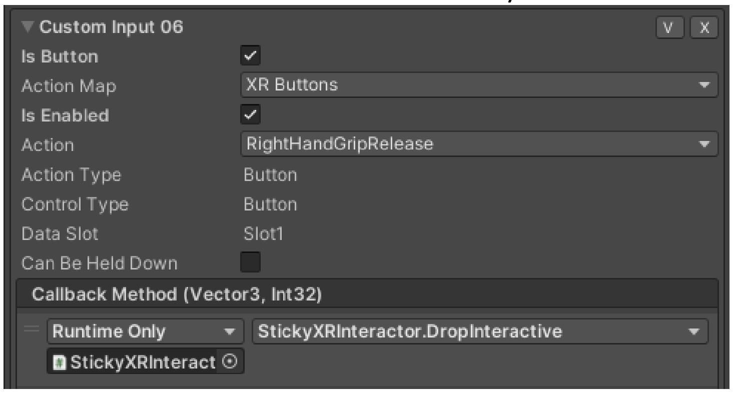
Callback Method (Vector3, Int32)

Runtime Only | StickyXRInteractor.EngageLookingAtInteractive
StickyXRInteract

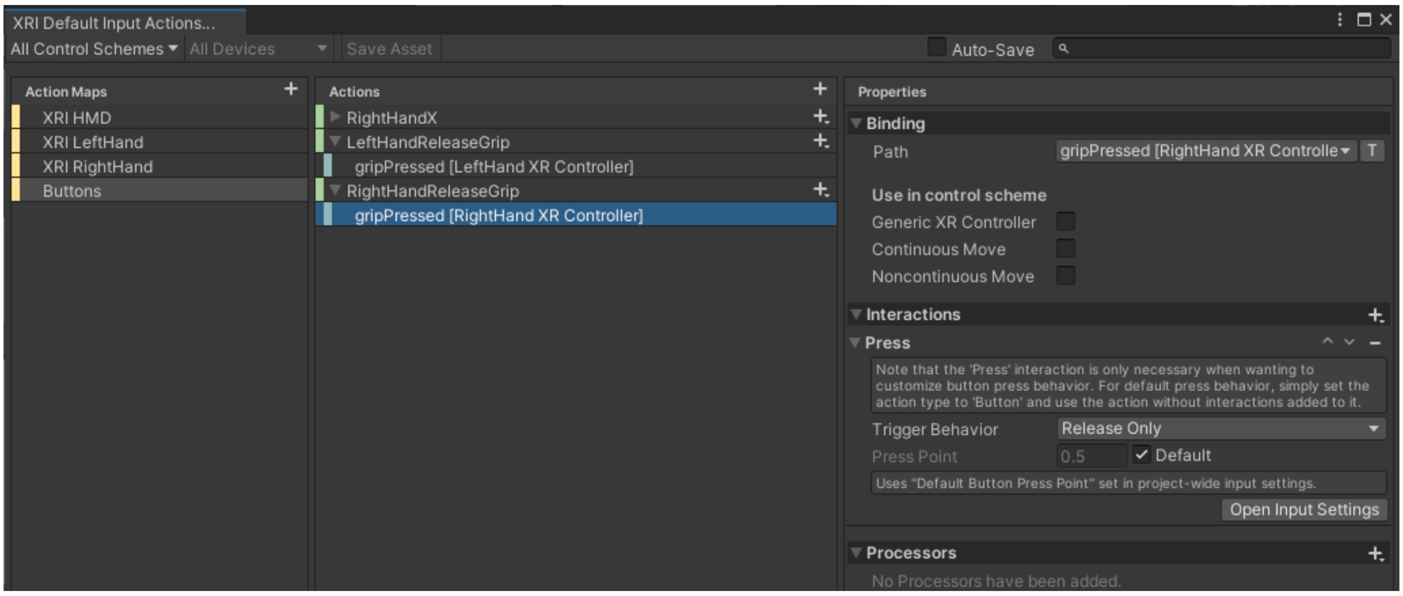
To release your grip on an interactive-enabled object, and drop or throw it, you can setup a separate button action for the hand-held controllers or create a "Release Grip" action. Then link the action to a "Custom Input" on the Sticky Input Module.

Use a pressable button when you want to hold objects for long periods of time.

See next image for Input Action setup.



Custom Input 06
Is Button: ✓
Action Map: XR Buttons
Is Enabled: ✓
Action: RightHandGripRelease
Action Type: Button
Control Type: Button
Data Slot: Slot1
Can Be Held Down: ☐
Callback Method (Vector3, Int32)
Runtime Only | StickyXRInteractor.DropInteractive
StickyXRInteract



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
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 XR Interactor and UI

When interacting with UI (canvas) elements in a scene, the standard Unity Graphic Raycaster assumes pointing originates with the camera. In VR, this will not be the case when you point at things with your hands. Therefore, on a canvas, you need to replace the Graphic Raycaster with our Sticky Graphic Raycaster component. This will let you point at (in Interactive Mode) and click with the left or right controller to activate a button on the UI.

If using Quest 2 controllers, the click buttons are most likely the Triggers.

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. "React to Sticky Zones" must also be enabled. 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.

Sticky Zones can be used to define areas where different animation clips are used for a character. For example, different sets of sitting animation clips may apply to seats of a different height.

If you want a character to be recognised within a zone when the scene first starts, disable the collider on the StickyZone and tick "Enable Colliders".

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.
Enable Colliders	Enable any disabled trigger colliders on initialise. If you are adding the zone or initialising it at runtime, you may wish to start with the trigger colliders disabled, so that when the zone initialises, it detects the initial characters within the zone.
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 if the character's "Reference Update Type" is "Auto First" or "Manual" (see Sticky Control Module – Collide tab for more details).
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.
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. Each Anim Clip Set can optionally be restricted to one or more character Model Ids.
Factions to Filter	An optional array of Faction IDs that will limit which characters this zone applies to.
Models to Filter	An optional array of Model IDs that will limit which characters this zone applies to.

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.

If you own Sci-Fi Ship Controller, be sure to check out the "SSC Input Bridge" and "Sticky Interactor Bridge" components which are included with Sticky3D.

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 physical 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 in the SRP folder 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.
3. When the scene first starts my character suddenly flies off in one direction then goes into what looks like a falling animation. Check that the character doesn't start inside another non-trigger collider or a mesh collider. Check that another active non-trigger collider isn't connected to the character. If your character needs to carry an object with a collider, talk to us on Discord about this scenario as we do have APIs to exclude certain colliders or you can add them to a Unity Layer not affected by collisions.

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”.
6. When “Root Motion” is enabled on the Animate tab, my character won’t move forward. Check that the Animate Actions are set up correctly. See the “Animate – Root Motion” section in the “Sticky Control Module” chapter.

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”.
12. My character cannot see StickyInteractive objects in the scene. See “Sticky Interactive – Looking at Objects” in the “Sticky Interactive” chapter.

13. Clip Objects is not shown in the inspector on the Look tab. This is not shown when in First Person mode. Click “Third Person” and “Clip Objects”, then toggle the fold out triangle if required to show the settings.
14. When my character is in First-Person, and they run or the head position in the animation changes, I can see inside the character’s head. On the “Look” tab, under “General Look Settings” when “First Person” is ticked, try enabling “Follow Head Position”.
15. How do I lock the third-person camera to a particular point in world space? Call stickyControlModule.(Un) LockThirdPersonCameraPosition(..) at runtime when in third-person. In code, you have the option of passing in the new world-space position, or using the existing camera position.
16. When the Celestials (stars) from Sci-Fi Ship Controller are installed in the scene, the stars can lag when the S3D camera rotates quickly from left to right. This can occur if Unity updates the stars before it updates the S3D character look rotation. On the character, go to the Look tab, General Settings, and enable “SSC Update Celestials”. Then go to the SSC Celestials component in the scene and set “Timing Type” to “Manual”.

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.
4. How do I replace one or more animation clips at runtime? This can be done with Anim Clip Sets. See the chapters on Sticky Anim Replacer and Sticky Zones.

Common Issues – Animate (Aim IK)

1. When holding a weapon in first-person, and looking up and down, the hands holding the weapon pitch up and down at a different rate. On the Sticky Control Module, go to the “Animate” tab, expand “Aim IK” and adjust the “Bone Weight FPS”.
2. Third person aiming or holding a weapon when Free Look is disabled, is NOT supported when the character Animator component has “Update Mode” set to “Unity Physics”. As a workaround, set the “Update Mode” to “Normal”. If you need this functionality, chat with us on Discord and mention log #486.

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 reticle 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".
4. Head IK doesn't work when my character is seated. On the Animate tab, under "Head IK", enable "When Move Disabled". This can also be set at runtime by setting s3dCharacter.headIKWhenMovementDisabled.

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. 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.
3. When character is on a moving or rotating object, the legs can jitter. Try reducing the Fixed Timestep in Project Settings, Time. E.g., if your game runs at 60fps, set the Fixed Timestep to 0.01667.

Common Issues – Animate (Ragdoll)

There are no known issues with the ragdoll feature – but please report any on our Discord channel if you see any.

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 – Engage (Damage Regions)

1. I've set the Damage Multipliers but they have no effect. Ensure your Sticky Projectile or Beam Module prefabs have a non-Default Damage Type.

Common Issues – Engage (Equip)

1. ToggleEquipGrabWeapon(..) API method doesn't grab an item equipped on the back or front of my character. If grabbing an equipped item with the left hand, the equip point must be toward the left side of the character. Check the "x" value of the "Relative Offset" of the Equip Point. The "x" value would need to be -ve for a left-hand grab, or +ve for a right-hand grab. If the "Relative Offset" x value is 0 (like in the centre of the character's back), then the object will not be grabbed from the Equip Point. Give the x value a small negative or positive offset (e.g., 0.001 for a right-hand grab from the Equip Point).

Common Issues – Engage (Events)

1. When I create a custom method to get notification when "On Look At Changed", my method is never called. Ensure you method takes the correct parameters. See S3DEngageEvt2 class. On the "Look" tab, ensure "Interactive" is enabled.

Common Issues – Engage (Respawn)

There are no known issues with the respawn feature – but please report any on our Discord channel if you see any.

Common Issues – Engage (Stash)

There are no known issues with the stash (personal inventory) feature – but please report any on our Discord channel if you see any.

Common Issues – SSC Output Bridge

1. The joystick or lever jitters when the Output Axis is set to "Longitudinal" for the Sci-Fi Ship Controller ship. It possible that the data coming from the ship is not stable. For example, on a player ship, check if the Player Input Module has "Auto Cruise" enabled. If it is, try turning it off and see if the problem persists. On the Sticky Interactive component, at runtime in the editor, turn on "Debug Mode" and look at the "IsReadable" data. This is the same data used by the SSC Output Bridge.

Common Issues – NPCs

2. How do I configure an NPC? First, look at our sample NPCs in Assets/SCSM/Sticky3DController/Demos/Prefabs/Characters. On the Move tab, under "General Move Settings", enable "Non-Player Character". On the Look tab, untick "Look on Initialise". An NPC should not have a StickyInputModule attached. NPCs should either not have a first-person camera, or it should be disabled. A third-person camera should not be configured. We're not saying you can't have a first-person camera on an NPC, as sometimes you might want to view the world through the eyes of an NPC. Our demo NPCs don't have an audio listener as there can only be one in any scene.

3. How do I get my character to follow, walk besides, and/or look at a player? Add a child gameobject to the NPC character with a trigger sphere collider. Then add a SampleLookAtPlayer (Component menu, Sticky3D Controller, Samples, Look at Player).
4. How can I get my NPC to move? If using a navmesh, see the NavMeshDemo scene and the SampleNavMeshFollowPlayer script. Otherwise, see the SampleLookAtPlayer and SampleMoveTo scripts for general ideas and methods of sending input data to your NPCs.

Common Issues – Sticky Display Module

1. When I change the Active Display Reticle and then enter play mode in the Unity Editor, the reticle changes back to the previous one. From the list of Display Reticles, delete the one that is does not remain selected when entering play mode. Add it back into the list and add the appropriate Sprite. Optionally, move it in the list using the "V" buttons. Set the Active Display Reticle again.
2. In a Windows build, whenever I move the cursor, the pointer arrow keeps appearing. After a few seconds of inactivity it disappears, but re-appears when the mouse if moved again. If you only want the reticle, and not the pointer, to be shown, on the Sticky Display Module, under General Settings, ensure "Auto-hide Cursor" is NOT enabled. Also go to the player Sticky Control Module, select the "Look" tab, and ensure "Auto-hide Cursor" is turned off there too.
3. In a Windows build, the windows pointer appears when the reticle is shown for the Sticky Display Module (a.k.a. HUD). On the player Sticky Control Module, go to the "Engage" tab, expand "Event Settings", set "On Initialised Event Delay" to a small value (e.g., 0.2), add a new event, drag in the StickyDisplayModule gameobject from the scene, and set the "Function" to "StickyDisplayModule.HideCursor".

Common Issues – Sticky Input Module (General)

1. 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.
2. 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.
3. When using the (new) "Unity Input System" I see "InvalidOperationException while resolving binding 'Rotation:QuaternionFallback' in action map…" or similar errors. This is most likely because you have copied an Input Action Map file from another project and either haven't installed "XR Interaction Toolkit" or copied over the "FallbackComposite.cs" file. Search this manual for "Fallback" for more information.
4. How do I get my player to fire a weapon that isn't setup as a StickyWeapon? (For information on our weapon system, see the "Sticky Weapon System" chapter). 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.

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. My character cannot touch an interactive-enabled object they look at. Make sure your interactive component has IsTouchable enabled and the Primary Handhold Relative Position (offset and rotation) are set. Is the

interactive object initialised (at runtime you can check “Debug Mode” on the component)? On the S3D character, is Hand IK enabled on the Animate tab. You can also check “Debug Mode” on the character at runtime.

2. 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).
3. My character cannot see StickyInteractive objects in the scene. See “Sticky Interactive – Looking at Objects” in the “Sticky Interactive” chapter.
4. 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.
5. I want to use interactive objects in VR. See the chapter on the “Sticky XR Interactor” component.
6. 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.
7. When different characters grab the same object the rotation and/or position of the object in the hand is incorrect. Set it up correctly for one type of character. Then look at the SampleGrabAdjustByCharacter component. There are instructions in the top of the script.

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 and set “Reference Update Type” to “Auto-First” or “Manual” (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(); }
```

4. When the scene first starts, my character ignores any zones that they appear in. Disable the collider on the StickyZone and tick “Enable Colliders”.

Common Issues – Sticky Magazines

1. When the S3D character grabs the magazine in the scene, it just stays in the original location and doesn't move with the character. On the Interactive tab of the magazine, ensure "Initialise on Start", "Is Grabbable", "Carry in Hand", and "Parented on Grab" are enabled.
2. When the scene starts the magazine just falls through the floor when "Use Gravity" is enabled. Ensure your prefab includes a non-trigger collider. If it has a collider, on the Magazine tab, under "Gravity Settings", set the Collision Detection to "Continuous Dynamic".

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.

Common Issues – Sticky Popup

1. How do I prevent a held weapon from firing when a Sticky Popup is being displayed? On the StickyPopupModule prefab, enable "Pause Weapon Firing" and set "Unpause Weapons Delay" to a short value like 0.1 seconds.

Common Issues – Sticky Shapes

1. When a duplicate Reaction is inserted (with the "I" button) in the editor, the events are not duplicated. This is a known issue with UnityEvents. Currently, you will need to add the events manually in the editor (sorry).

Common Issues – Sticky Sockets

1. When a socket is attached to an object containing a rigidbody, how do I prevent interactive-enabled objects that have gravity enabled from being pushed outside the parent colliders? On the StickySocket component, go to the "Socket" tab, and set "Disable Regular Colliders".
2. How do I configure Events at runtime in my own C# code?

   ```
   stickySocket.onPreAdd = new S3DSocketEvt2();
   stickySocket.onPreAdd.AddListener(delegate { MyCustomMethod(); });
   ```

   In your OnDestroy() method you should also call stickySocket.RemoveListeners() to prevent memory leaks.
3. When I grab a weapon from a socket immediately after attaching it to the socket from my character, it gets the wrong weapon held animation. I have a Weapon Anim Set that should always replace the default held animation with the one in the Anim Set. Try going to the Weapon Anim Set for this weapon, and slightly reducing the Clip Revert Delay. If it is 1 second, try making it 0.5 seconds. What might be happening is that when you socket the weapon from the character, it is waiting 1 second before reverting to the default "held" animation. However, before the revert occurs, you grab the weapon and then your held animation is applied. A very short time later, the revert rolls back your desired change.

Common Issues – Sticky Weapons (General)

1. How do I get my character to pick up and fire a weapon? See "Weapon System Overview" in the "Sticky Weapon System" chapter.
2. When the S3D character grabs the weapon in the scene, the weapon just stays in the original location and doesn't move with the character. On the Interactive tab of the weapon, ensure "Initialise on Start", "Is Grabbable", "Carry in Hand", and "Parented on Grab" are enabled.
3. My weapon doesn't fire. Is the weapon being held by a S3D character? (i.e., is the weapon Grabbable?). On the Weapon tab of the weapon, is the Firing Button1 set and does this match the Right (or Left) Fire 1 Button on the Sticky Input Module for your player?

4. My projectile weapon won't fire continuously on the Full Auto setting. In the Sticky Input Module for the player, for the fire button being used, ensure "Can Be Held" is enabled. At runtime, check that the weapon has enough Charge Amount.
5. When I hold the fire button down for my Beam weapon, it doesn't continue to fire. In the Sticky Input Module for the player, for the fire button being used, ensure "Can Be Held" is enabled.
6. When the scene starts the weapon just falls through the floor when "Use Gravity" is enabled. Ensure your prefab includes a non-trigger collider. If it has a collider, on the Weapon tab, under "Gravity Settings", set the Collision Detection to "Continuous Dynamic".
7. Also check 'Common Issues – Sticky Interactive" as weapons inherit many attributes of interactive-enabled objects.
8. When my weapon reloads, it doesn't make any sound. On the Weapon tab of the weapon, ensure the "Reload Sound FX" is using a Sticky Effects Module prefab with an EffectsType of "Sound FX" and the audio source has an audio clip attached to it.
9. When my weapon reloads, I can hear it reload, but then I still cannot fire for several seconds. On the Weapon tab of the weapon, go to the "Reload Sound FX" prefab and make sure the length of the audio file matches (or is slightly less than) the "Reload Duration" of the weapon.
10. When my weapon reloads, the sound gets cut off before it should do. On the Weapon tab of the weapon, go to the "Reload Sound FX" prefab and make sure the "Despawn Time" is slightly longer than the audio file.
11. My weapon cannot fire or be reloaded when it isn't being held by a S3D character. On the StickyWeapon, go to the Weapon tab, under "General Settings", disable "Only Use When Held".
12. When aiming, the weapon doesn't aim up or down to face the target. Ensure Aim IK is being enabled either using a Custom Input on the "Sticky Input Module" or aiming is immediately enabled when a weapon is held (see Aim IK on the character Animate tab). Ensure you have at least 1 "Aim Bone" in "Aim IK".
13. When aiming, my character spine bones jitters all over the place. This tends to happen when the Animator for the character has the "Update Mode" set to "Animate Physics". If not using Root Motion animations, try setting "Update Mode" to "Normal". This is known problem.
14. When a beam weapon fires, the beam flashes within the attached scope (camera) lens, and look a little weird. Set the Sticky Beam Module prefab Unity Layer to "Ignore Raycast". Click "Yes, change children" when prompted. On the weapon's scope camera, remove the "Ignore Raycast" from the camera's "Culling Mask".
15. How can I get my third-person character to accurately aim a weapon? Setup the first-person camera, then in a "Weapon Anim Set", ensure "TP Aim uses FP Camera" is enabled. See "Weapon Anim Sets (for characters)" in the "Sticky Weapon – Animate Properties" section of the "Sticky Weapon" chapter.
16. How can I change my third person camera offset when a character picks up a weapon? See Demo\Scripts\SampleWeaponGrabTPS.cs which uses the On Pre Start Hold Weapon and On Post Stop Hold Weapon events. Also, see the "Aim Camera Offset TPS" on the Animate tab for the aiming offset.

Common Issues – Sticky Weapons (Animate)

1. How can I use the same weapon with two different characters, but use different character animations or selected settings? Use 2 different Weapon Anim Sets (one for each character Model ID) and add them to the Animate Settings on the weapon.
2. When two different characters hold the same weapon, how can I get one to hold it in a relaxed pose while not aiming and allow the character to look around using the character's Free Look option? The other character should have the weapon always pointing forward and Free Look should be disabled? You will need a different held animation for each character and weapon. You can use a Weapon Anim Set for each character and weapon pair. For the character and weapon pair that has the relaxed held pose, enable "Free Look" on the character's "Look" tab, and on the Weapon Anim Set, enable "Free Look When Held". On the Weapon Anim Sets, add "Anim Clip Pairs" to replace your default held animation with the appropriate one for that character and weapon pair.
3. When my character picks up a weapon, the correct animation doesn't play. Check that you have a Weapon Anim Set that matches the character identification Model ID. Ensure the Anim Clip Pairs section contains both the original animations (in the animation controller for this character) and the ones you want to replace them with when the character picks up (grabs) the weapon. You probably need at least one Animate Action for the weapon

that tell the character animation controller when the weapon is held. See the "Sticky Weapon" chapter under "Sticky Weapon – Animate Properties", "Weapon Anim Sets (for characters)" for more details.

Common Issues – Sticky Weapons (Reticle)

1. How can I have two different characters that when holding the same weapon, one shows a reticle while looking around with a relaxed pose, and the other does not show the reticle? Start by setting up two Weapon Anim Sets - see Common Issues – Sticky Weapons (Animate) #2. On the weapon, in the "Weapon" tab within the "General Settings" section, under "No Reticle When:", enable "Held with Free Look OFF".
2. How can I hide the reticle when aiming? Assuming you have a StickyDisplayModule (HUD) in your scene, on the "Weapon" tab of the weapon within the "General Settings" section, under "No Reticle When:", enable "Aiming the Weapon".
3. The pointer (cursor) appears, even if "No Reticle When" "Aiming the Weapon" is enabled on a weapon that is being held by the player. On the S3D player holding the weapon, on the "Look" tab, ensure "Auto-hide Cursor" is NOT enabled. On the StickyDisplayModule (HUD) in the scene, ensure "Auto-hide Cursor" is also NOT enabled. If "Auto-hide Cursor" is enabled on the character or the HUD, it will show the pointer when there is mouse movement.

Common Issues – VR

1. I cannot look at, then grab interactive-enabled objects in VR. See the chapter "Sticky XR Interactor".
2. I get jitter when looking left or right. There are two potential workarounds.
   a. 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.
   b. 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".

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
DemoS3DCottageCamera	Used in CottageCameraDemo scene to show a scenario of switch between gameplay camera control and S3D camera control (there are many other possible options).
DemoS3DCycleMaterials	Demo script to cycle through two or more materials on an object.
DemoS3DCycleRenderers	Demo script to cycle (flash) one or more renderers on/off.
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.
DemoS3DTarget	Simple target that can be hit by a Demo S3D_Projectile (for use with a beam or projectile StickyWeapon, see Demo3DTarget2).
DemoS3DTarget2	Shows how Dynamic Objects can receive fire from a StickyWeapon. See also the SampleHitDynamicObject script.

Script Name	Description
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.
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.
Sample Equip Grab Stash Popup	Simple script to attach to a character to show Equip, Grab or Stash options for an interactive-enabled object.
SampleEquipItemAny	Sample script to equip an interactive object to a character when it is near the 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.
SampleGrabAdjustByCharacter	Simple sample script to attach to an interactive-enabled object like a weapon. It can be used when different characters have different rig setups and may incorrectly rotate objects when they are grabbed.
SampleHandInteract	Sample script to demonstrate a Sticky3D character (S3D) interacting with an interactive-enabled object in the scene using Hand IK.
SampleHitDynamicObject	Simple script to demonstrate how to instantiate a dynamic object and destroy it (return it to the pool) when it has taken too much damage.
SampleInputOverride	This demonstrates how you could override the input from one or more axes on the StickyInputModule. It also shows how you could enable or disable overrides in your game code or call a delegate method in another class.
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.
SampleNPCFireWeapon	Simple script to attach to weapon that an NPC can pick up and shoot. Currently used in NavMeshDemo scene.

Script Name	Description
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.
SampleWeaponCustomAnimAction	Sample script to have a StickyWeapon run an Animation Action in code when a weapon fires. A simpler setup for this scenario is just to use the Fire1 Weapon Action without having to write any code. However, here we want to show how to do things in code.
SampleWeaponCustomReload	Simple sample script to attach to a weapon to perform a custom reload action when the weapon runs out of ammo.
SampleWeaponGrabTPS	Simple sample script to attach to a S3D player character to modify the camera offset when a weapon is held in third person. It uses the onPreStartHoldWeapon and onPostStopHoldWeapon events. This shows how to call APIs when starting to hold or drop a weapon.
SampleXRDataPopup	Sample script which shows a popup in VR when you touch an interactive-enabled object. This example, shows data about a StickyInteractorBridge component.

SSC Input Bridge - Methods

These methods are typically used when you want to send data from a lever or joystick in VR to a Sci-Fi Ship Controller ship you want to fly. See the “SSC Input Bridge” chapter for more details.

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

Virtual Method	Description
ReconfigureAxisData()	Attempt to reconfigure the ship for receiving input. Currently we can only do this once and cannot change the overrides at runtime unless we were to reset them all - which could cause issues with other input override scripts.

Method	Description
Initialise()	Initialise this component. Either set Initialise on Start in the inspector OR call this from your game code. Has no effect if already initialised
SetInputAxisX (InputAxis newInputAxisX)	Attempt to set the InputAxisX. If the component is initialised, it automatically calls ReconfigureAxisData().
SetInputAxisZ (InputAxis newOutputAxisZ)	Attempt to set the InputAxisZ. If the component is initialised, it automatically calls ReconfigureAxisData().
SetShip (GameObject newShipGameObject)	Set the player ship from your scene which should contain a ShipControlModule on the same gameobject.
SetShip (ShipControlModule newShipControlModule)	Set the Sci-Fi Ship Controller player ship or spacecraft from the scene

SSC Input Bridge - Properties

These properties are typically used when you want to send data from a lever or joystick in VR to a Sci-Fi Ship Controller ship you want to fly. See the “SSC Input Bridge” chapter for more details.

Property	Description
InputAxisX	Get or set the interactive Readable left-right data, to be send to the input axis of the Sci-Fi Ship Controller player ship.
InputAxisY	Get or set interactive Readable forward-back data, to be send to the input axis of the Sci-Fi Ship Controller player ship.
IsInitialised	Has the component been initialised?

SSC Input Bridge – Call Backs

Event	Description
OnInputChanged (int stickyInteractiveID, Vector3 currentValue, Vector3 previousValue, Vector3 notused)	This is automatically called by Sticky3D when you hook it up to the lever's OnReadableValueChanged event on the interactive-enabled object.

SSC Output Bridge - Methods

These methods are typically used when you want an in-game lever or joystick to move as the Sci-Fi Ship Contoller ship is getting input data from a player or AI module. See the “SSC Output Bridge” chapter for more details.

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

Virtual Method	Description
ReconfigureAxisData()	Attempt to prepare the bridge for receiving input from the ship

Method	Description
Initialise()	Initialise this component. Either set Initialise on Start in the inspector OR call this from your game code. Has no effect if already initialised
SetShip (GameObject newShipGameObject)	Set the ship from your scene which should contain a ShipControlModule on the same gameobject.
SetShip (ShipControlModule newShipControlModule)	Set the Sci-Fi Ship Controller player ship or spacecraft from the scene/
SendData (bool isFixedUpdate)	Call this to manually update the writeable sticky interactive object. The Timing Type must be Manual.
SetOutputAxisX (OutputAxis newOutputAxisX)	Attempt to set the OutputAxisX. If the component is initialised, it automatically calls ReconfigureAxisData().
SetOutputAxisZ (OutputAxis newOutputAxisZ)	Attempt to set the OutputAxisZ. If the component is initialised, it automatically calls ReconfigureAxisData().
SetTimingType (TimingType newTimingType)	Change the timing type.

SSC Output Bridge - Properties

These properties are typically used when you want an in-game lever or joystick to move as the Sci-Fi Ship Controller ship is getting input data from a player or AI module. See the “SSC Output Bridge” chapter for more details.

Property	Description
BridgeTimingType	Get or set the Timing Type
IsConfigured	Is the component configured and ready to start receiving input from a Sci-Fi Ship Controller ship?
IsInitialised	Has the component been initialised?
OutputAxisX	Get or set interactive Writeable left-right data, to be read from the input axis of the Sci-Fi Ship Controller ship
OutputAxisZ	Get or set interactive Writeable forward-back data, to be read from the input axis of the Sci-Fi Ship Controller ship.

Sticky Control Module Methods - General

Method	Description
AudioMute ()	Turn off all audio on the character
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 (except NPCs), 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.
GetLocalDirection (Vector3 wsDirection)	Get a local space direction relative to the character forward direction.
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.
GetWorldPosition (Vector3 localSpacePosition)	Get a world space position of a local space position on the character (converts a 3D position from local space to world space).
GetWorldRotation (Quaternion localRotation)	Get a world space rotation of a local space rotation on the character (converts a 3D rotation from local space to world space).
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.

Method	Description
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 or AI) to the character for movement, look and weapon 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?
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 (NPC)?
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. If applicable, also teleports the third-person camera.
TelePort (Vector3 newPosition, Vector3 newRotation, bool resetVelocity)	Teleport or move the character to a new place. If applicable, also teleports the third-person camera.
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.
CheckRestoreFollowHead()	Restore isLookCameraFollowHead (first person) to the state before it was saved. WARNING: Be careful not to call it multiple times in a row.
CheckRestoreFollowHeadTP()	Restore isLookCameraFollowHead (third person) to the state before it was saved. WARNING: Be careful not to call it multiple times in a row.
CheckRestoreFreeLook()	If Free Look is off, but should be on, turn it on. Always set savedFreeLook back to false. NOTE: This never turns Free Look off.
CheckRestoreLockedReticle ToCursor()	Re-enable locked reticle to cursor if required on the Sticky Display Module (HUD).

Method	Description
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. This includes camera rotation. See also LockThirdPersonCameraPosition(), LockThirdPersonCameraRotation(), and FreezeCameraPosition().
DisableLookSockets()	The character will no longer be able to see or detect objects with a StickySocket component.
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.
EnableLookSockets()	The character will be able to see or detect objects with a StickySocket component.
EnableSnapTurnVR()	Enable Snap Turn for VR.
FreezeCameraPosition()	Prevent the third person camera from moving. See also DisableLookMovement() and LockThirdPersonCameraPosition().
FreezeCameraPosition (Vector3 newPosition)	Attempt to snap the third person camera to the world space position and prevent it from moving. See also UnfreezeCameraPosition().
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.
GetLookingAtRay (Vector3 screenPosition)	If look is enabled and not an NPC, return a ray from the camera to the screen position. Otherwise, return a character forward ray from the eye position. The ray is in world space. See also GetAimingRay().
GetLookingAtSocket()	Get the StickySocket that the character is currently looking toward.
GetLookingAtSocketId()	Get the unique ID of the sticky socket that the character is currently looking toward. If none, StickySocket.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 (Vector3 lookTo, bool useEyePosition = true, bool debugInSceneView = false)	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. Optionally draw a blue ray in the scene view for debugging in the Unity Editor. This will return false when the character is respawning.
LockThirdPersonCamera Position()	Attempt to lock the third-person camera to the current world space position, relative to the reference frame. Doesn't work if an NPC, or Aim IK is enabled.
LockThirdPersonCamera Position (Vector3 newWorldCameraPosition)	Attempt to lock the third-person camera to the given world space position, relative to the reference frame. Doesn't work if an NPC, or Aim IK is enabled.

Method	Description
LockThirdPersonCamera Rotation()	Attempt to lock the third-person camera to the current world space rotation, relative to the reference frame. Doesn't work if an NPC, or Aim IK is enabled. See also DisableLookMovement() and FreezeCameraPosition().
LockThirdPersonCamera Rotation (Quaternion newWorldCameraRotation)	Attempt to lock the third-person camera to the given world space rotation, relative to the reference frame. Doesn't work if an NPC, or Aim IK is enabled.
ReinitialiseCameraSettings ()	Call this after changing camera properties OR changing the camera with SetLookCamera1(..)
ResetClipObjectMask()	Reset the clip object mask to their default values
SaveAndSetLockedReticleTo Cursor (bool isLockReticleToCursor)	Save if locked reticle to cursor on Sticky Display Module (HUD). Only is saved if it hasn't already been saved. Also sets Lock Reticle to Cursor.
ResetFreeLook()	Attempt to reset FreeLook to default position.
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.
SetLookCamerFocusOffsetTP (Vector3 focusOffset)	Set the local space point on the character where the third person camera focuses relative to the origin or pivot point of the character prefab.
SetLookCameraOffsetTP (Vector3 cameraOffset)	Set the camera offset or distance from the character when in third person mode.
SetLookFirstPerson (Camera firstPersonCamera, Transform firstPersonTransform, bool isAutoEnableLook)	Configure the controller for First Person. Note, this can automatically set look movement to true.
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.
SetLookFollowHeadTP (bool isEnabled)	Set the third person camera to follow the local offset of the head on a humanoid character. Currently Follow Head in third-person has no effect when Head IK is enabled.
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.
SetLookSocketAutoShow	Set if the StickySockets will be highlighted when looked at and Look Sockets is enabled.
SetLookUnorbitDelay (float newValue)	Set the unorbit delay for the third person camera.

Method	Description
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
SetLookZoom (float zoomAmount)	Set the level of zoom. -1.0 if fully zoomed out. 1.0 is fully zoomed in. 0 is no zoom. Also resets the unzoom timer.
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
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.
StopLookAtSocket()	Stop looking at a StickySocket. It does not prevent the character from looking at it again. If you want to stop the character looking at Sockets, use DisableLookSockets() 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.
CutToThirdPersonCamera Target()	If in third person, attempt to cut the camera directly to the target position and rotation.
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. When switching to first-person, it can automatically enable look movement.
ToggleLookInteractive()	Toggle if the character will be able to see or detect objects with a StickyInteractive component.
ToggleLookSockets()	Toggle if the character will be able to see or detect objects with a StickySocket component.
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.
UnfreezeCamera Position()	Allow the third person camera to move after having been frozen. See also FreezeCameraPosition()
UnLockThirdPerson CameraPosition()	Allow the third person camera to move freely again.
UnLockThirdPerson CameraRotation()	Allow the third person camera to rotate freely again.

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

Method	Description
	detection on the character. When grabbing interactive-enabled objects, this is automatically called by S3D.
AttachColliders (Collider[] collidersToAttach)	When attaching objects to the character, like when grabbing an object, call this method with an array of the active non-trigger colliders. See also AttachCollider(..).
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())
IsColliderHittableByWeapon (Collider collider)	Is the collider part of this character and hittable by a weapon?
IsColliderHittableByWeapon (int colliderInstanceID)	Is the collider part of this character and hittable by a weapon? USAGE: bool isHit = IsColliderHittableByWeapon(collider.GetInstanceID());
IsColliderSelf (Collider collider)	Is the collider part of this character? Takes into consideration the capsule collider, any feet or damage region colliders, and any attached or held objects.
IsColliderSelf (int colliderInstanceID)	Is the collider part of this character? Takes into consideration the capsule collider, any feet or damage region colliders, and any attached or held objects. It will NOT automatically ignore all trigger colliders associated with this character. USAGE: bool isSelf = IsColliderSelf(collider.GetInstanceID());
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.
RegisterWeaponNonHitCollider (Collider collider)	Register this (child) collider as one weapons will ignore when firing at with beams, projectiles etc.
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.
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.

Method	Description
SetCurrentReferenceFrame (Transform newReferenceFrame, bool isForceUpdate)	Sets the current reference frame with the option to forcefully update it.
SetReferenceFrameFromTemp (Transform newReferenceFrame)	Attempt to set a new reference frame after having set a temporary reference frame. This doesn't affect the reference frame history. See also SetTemporaryReferenceFrame(..).
SetTemporaryReferenceFrame (Transform newReferenceFrame)	Attempt to set a short-term reference frame. This does not affect the previous reference frame so has no effect on entering and exiting StickyZones. An example usage is sitting on a movable cockpit seat or riding an elevator while in a flying spaceship. See also SetReferenceFrameFromTemp(..).
SetReferenceUpdateType (ReferenceUpdateType newRefUpdateType)	Change the way reference frames are determined.
UnregisterWeaponNonHitCollider (Collider collider)	Unregister this (child) collider as one weapons will ignore when firing at with beams, projectiles etc. The collider will now be hittable by weapons.

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(..).
CheckAnimateConditions (S3DAnimAction s3dAnimAction)	Check to see if all the conditions (if any) are satisfied for an animate action. Returns true if all conditions are satisfied, else returns false.
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.

Method	Description
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.
GetAnimActions (List<S3DAnimAction> animActionList, S3DAnimAction.StandardAction standardAction, string parameterName)	Populate a non-null empty list with all S3DAnimActions that match the supplied StandardAction type with the provided parameter name. Returns true if it found matching actions, else it returns false. This is NOT the kind of method that you should be calling every frame.
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.
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.
GetBoneTransform (HumanBodyBones humanBone)	Attempt to get the transform for a human body bone.
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.
IsAnimatorFixedUpdate (bool forceRefresh)	Check if the character Animator is updating in the FixedUpdate loop (aka Unity Physics). If forceRefresh is true, it will check the Animator component rather than just returning the last known value.
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.

Method	Description
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 and the standardAction is a custom action. 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 and the standardAction is a custom action. 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 and the standardAction is a custom action.
SetCustomAnimActionFloatValue (S3DAnimAction s3DAnimAction, float value)	Given a custom S3DAnimAction set its float value, assuming it has a matching ParamaterType and the standardAction is a custom action.
SetCustomAnimActionIntegerValue (int guidHash, int value)	Given the guidHash of a custom S3DAnimAction set its integer value, assuming it has a matching ParamaterType and the standardAction is a custom action.
SetCustomAnimActionIntegerValue (S3DAnimAction s3DAnimAction, int value)	Given a custom S3DAnimAction set its integer value, assuming it has a matching ParamaterType and the standardAction is a custom action.
SetCustomAnimActionTriggerValue (int guidHash, bool value)	Given the guidHash of a custom S3DAnimAction set its trigger (bool) value, assuming it has a matching ParamaterType and the standardAction is a custom action.
SetCustomAnimActionTriggerValue (S3DAnimAction s3DAnimAction, bool value)	Given a custom S3DAnimAction set its trigger (bool) value, assuming it has a matching ParamaterType and the standardAction is a custom action.
StopRevertAnimClips()	If there is an ReplaceAnimationClipsDelayed(..) pending completion, attempt to stop it. Can be useful if wanting to replace some animations soon after another weapon has been dropped or equipped.
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 found, 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 (Aim IK)

Method	Description
EnableAimAtTarget (bool isEnable, bool isSmooth)	Enable or disable Aim IK. Called from stickyWeapon.EnableOrDisableAiming(..). Rather than calling this directly, devs will typically want to call StartAimingWeapon(..), StopAimingWeapon(..) or ToggleAimWeapon(..).
GetAimingRay()	Get the ray where the character is aiming while holding a weapon. See also GetLookingAtRay(..).

Method	Description
RefreshAimBoneTransforms()	Get the bone transforms for the aimBones array configured in the editor. Typically this will only be called once during Initialise(). NOTE: The list is only populated with bone transforms found on the humanoid rig.
SetAimBone (S3DHumanBone newAimBone, int arrayIndex)	Set (update) an existing aim bone in the current array.
SetAimBones (S3DHumanBone[] newS3DHumanBones)	Set a new array of Aim Bones
SetAimIKWhenNotAiming (bool newValue)	If a weapon is held, will it always attempt to face the target? Otherwise, it will only do this when weapon IsAiming is true.

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.

Method	Description
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)
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.

Method	Description
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 (Ragdoll)

These methods are used with the ragdoll features.

Method	Description
DisableRagdoll()	Attempt to disable the ragdoll.
ConfigureRagdollBones()	Attempt to get and set the bones used to create a ragdoll. Only applies to valid humanoid rigs.
EnableRagdoll()	Attempt to enable the ragdoll.
GetBoneLength (Transform humanBoneTfrm, Transform childBoneTfrm, out int boneAxis, out float boneLength)	Get the length of a bone.

Method	Description
GetRagdollBone (HumanBodyBones humanBodyBone)	Find the persistent bone in the list of ragdoll bones.
GetRagdollBoneIndex (HumanBodyBones humanBodyBone)	Find the index of the bone in the list of ragdoll bones. Returns -1 if not found.
GetRagdollBoneTransform (HumanBodyBones humanBodyBone)	Attempt to find the persistent bone in the list of ragdoll bones, and return the boneTransform.
SetRagdollBones (List<S3DHumanBonePersist> newRagdollBones)	Set a list of bones used for the ragdoll feature.

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

Sticky Control Module Methods – Engage (General)

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
ClearActivePopup (int instanceID)	Remove the record of the instance ID of an active StickyPopupModule that was directly or indirectly opened by this character. See also ClearActivePopups(), GetActivePopups(), IsActivePopup(..), IsActivePopups(), SeActivePopup(..).
ClearActivePopups()	Remove or clear ALL the record of any active StickyPopupModules that where directly or indirectly opened by this character. See also ClearActivePopup(..), GetActivePopups(), IsActivePopup(..), IsActivePopups(), SeActivePopup(..).
DropInteractive (bool isLeftHand)	Drop the interactive-enabled object in the hand of the character.
EngageLookingAt (bool isLeftHand)	Attempt to engage with first sockets, then interactive-enabled objects that the character is looking toward. Look at Interactive and/or Sockets must be enabled to take any action.
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.
EngageLookingAtSocket()	Attempt to take action based on what StickySocket the character is currently looking toward.
GetActivePopupIDs()	Return the HashSet of Instance IDs of the active StickyPopupModules that were directly or indirectly opened by this character.

Method	Description
	See also ClearActivePopup(..), ClearActivePopups(), IsActivePopup(..), IsActivePopups(), SeActivePopup(..).
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(..).
IsActivePopup (int instanceID)	Was the instanceID of a StickyPopupModules, directly or indirectly opened by this character? See also ClearActivePopup(..), ClearActivePopups(), GetActivePopups(), IsActivePopups(), SeActivePopup(..).
IsActivePopups()	Are there any active StickyPopupModules displayed, that where directly or indirectly opened by this character? See also ClearActivePopup(..), ClearActivePopups(), GetActivePopups(), IsActivePopup(..), SeActivePopup(..).
IsFoe (int otherFactionId)	Return true if the faction of another character is a foe (enemy). Otherwise return false.
IsFoe (StickyControlModule otherCharacter)	Return true if the faction of another character is a foe (enemy). Otherwise return false.
IsFriend (int otherFactionId)	Return true if the faction of another character either neutral or friendly. Otherwise return false.
IsFriend (StickyControlModule otherCharacter)	Return true if the faction of another character either neutral or friendly. Otherwise return false.
IsFriendOnly (int otherFactionId)	Return true if other faction is the same faction of this character AND the faction of this character is not neutral (0).
IsFriendOnly (StickyControlModule otherCharacter)	Return true if other character is in the same faction AND the faction of either is not neutral (0).
IsFriendOrFoe (int otherFactionId)	Given a factionId, return 1 if friend, 0 if neutral, or -1 if foe (enemy).
PauseWeapons()	Pause weapons that are held.
PauseWeaponsFiring()	Prevent weapons that are held from firing.
ReinitialiseStoreSettings()	Reinitialise the storage and selection of interactive-enabled objects
ReloadWeaponLeftHand (int weaponButtonNumber = 1)	Attempt to reload a weapon held in the left hand.
ReloadWeaponRightHand (int weaponButtonNumber = 1)	Attempt to reload a weapon held in the right hand.
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.
SelectInteractive (StickyInteractive stickyInteractive)	Select an interactive-enabled item in the scene
SelectLookedAtInteractive()	Select the interactive-enabled object currently being looked at (if any)
SetActivePopupID (int instanceID)	Set the instance ID of the active StickyPopupModule currently being displayed. See also GetActivePopupInstanceID().
SetEngageColour (Color32 newColour)	Set the colour typically used for when the character engages with an interactive-enabled object in the scene.
SetInteractiveTags (S3DInteractiveTags newInteractiveTags)	The common interactive tags scriptableobject used to determine which interactive objects can be added to this character. See also Equip Points.

Method	Description
SetNonEngageColour (Color32 newColour)	Set the colour typically used for when the character is not engaged with an interactive-enabled object in the scene.
SetOnInitialisedEvtDelay (float newValue)	Change the amount of time the configured event methods are called after the character is initialised. This will have no effect after the character is initialised.
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. If neither of the above is true, attempt to grab the object from a distance if they are currently looking at it (and it isn't Touchable).
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.
UnpauseWeapons()	Attempt to resume weapons that are held
UnpauseWeaponsFiring()	Allow weapons that are held to fire
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.
UnselectLookedAtInteractive()	Attempt to unselect or deselect an interactive-enabled item that is currently being looked at by this character.

Sticky Control Module Methods – Engage (Damage)

These methods are used to deal with different aspects of damage for your character.

Method	Description
AddDamageRegion (S3DDamageRegion damageRegion)	Add a damage region to character. Will return the index in the zero-based list, or -1 if it fails.
AddHealth (S3DDamageRegion damageRegion, float healthAmount, bool isAffectShield)	Add health to a specific S3DDamageRegion. If isAffectShield is true, and the health reaches the maximum configured, excess health will be applied to the shield for the specified DamageRegion. NOTE: -ve values are ignored. To incur damage, use the ApplyNormalDamage or ApplyCollisionDamage API methods.
ApplyNormalDamage (float damageAmount, S3DDamageRegion.DamageType	Applies a specified amount of damage to the character at a specified position.

Method	Description
damageType, Vector3 damagePosition)	
GetDamageRegion (int guidHash)	Get the damage region with guidHash in the list of regions. See also GetDamageRegionByIndex().
GetDamageRegionByIndex (int damageIndex)	Get a Damage Region from the zero-based index from the list. To get the main damage region, use damageIndex = 0.
GetDamageRegionIndexByName (string regionName)	Get a Damage Region by name. This will likely impact GC so don't use in an Update loop or get it each frame. Where possible, use GetDamageRegion() or GetDamageRegionByIndex().
GetMainDamageRegion ()	Get the main (overall) damage region for the character.
GetMainDamageRegionID()	Get the unique ID of the main (overall) damage region for the character.
HasDamageRegion (HumanBodyBones humanBodyBone)	Does this character have a damage region with a matching HumanBodyBone? This will ignore the first (Main) damage region.
HasDamageRegion (S3DDamageRegion damageRegion)	Does this character contain this damage region?
MakeCharacterInvincible()	Make the whole character invincible to damage. For individual damageRegions change the isInvisible value on the localised region.
MakeCharacterVincible()	Make the whole character vincible to damage. When hit, the character or shields will take damage.
ReinitialiseDamage()	Initialise or reinitialise damage settings for this character.
ResetHealth()	Reset the health of the whole character. The character must be initialised.
SetHealth (float newHealthValue)	Set the (overall) health of the character

Sticky Control Module Methods – Engage (Equip)

These methods help you work with the character Equip Points.

Method	Description
AddEquipPoint (S3DEquipPoint equipPoint)	Add a Equip Point to character. Will return the index in the zero-based list, or -1 if it fails.
DropEquipAll()	Drop all the equipped items.
DropEquipItem (int storeItemId)	Drop an interactive object currently equipped on the character.
DropEquipItem (S3DStoreItem storeItem)	Drop an interactive object currently equipped on the character.
EquipItem (StickyInteractive itemToEquip)	Attempt to Equip an interactive-enabled object on the first available Equip Point. Equipped objects become unselected if they were previously selected. Return the Equip Point S3DStoreItem ID or S3DStoreItem.NoStoreItem.
EquipFromLookAtSocket()	Attempt to Equip an interactive-enabled object on the first available Equip Point, with the most recent item added to the socket that the character is looking toward. Returns the Equip Point StoreItemID or S3DStoreItem.NoStoreItem.
EquipFromLookAtSocketNoReturn()	Attempt to Equip an interactive-enabled object on the first available Equip Point, with the most recent item added to the socket that the character is looking toward. Same as EquipFromLookAtSocket(), except can be called from a StickyInputModule custom input event.
EquipFromSocket (StickySocket stickySocket)	Attempt to Equip an interactive-enabled object on the first available and compatible Equip Point, with the most recent item added to the socket. Returns the Equip Point StoreItemID or S3DStoreItem.NoStoreItem.
EquipFromSocketNoReturn (StickySocket stickySocket)	Attempt to Equip an interactive-enabled object on the first available and compatible Equip Point, with the most recent item added to the socket.

Method	Description
	Same as EquipFromSocket(..),except can be called from a StickyInputModule custom input event.
EquipItem (StickyInteractive itemToEquip)	Attempt to Equip an interactive-enabled object on the first available and compatible Equip Point. Equipped objects become unselected if they were previously selected. Return the Equip Point S3DStoreItem ID or S3DStoreItem.NoStoreItem.
EquipItem (StickyInteractive itemToEquip, S3DEquipPoint equipPoint)	Attempt to Equip an interactive-enabled object on an Equip Point. Equipped objects become unselected if they were previously selected. Return the Equip Point S3DStoreItemID or S3DStoreItem.NoStoreItem.
EquipItemFromLeftHand (int equipPointIndex)	If the character is holding an interactive object in their left hand, attempt to parent it to the equip point using the zero-based index. Returns the Equip Point StoreItemID or S3DStoreItem.NoStoreItem.
EquipItemFromRightHand (int equipPointIndex)	If the character is holding an interactive object in their right hand, attempt to parent it to the equip point using the zero-based index. Returns the Equip Point StoreItemID or S3DStoreItem.NoStoreItem.
EquipItemFromHand (bool isLeftHand)	Attempt to Equip an interactive-enabled object currently held in a hand onto the first available compatible Equip Point. Return the Equip Point S3DStoreItem ID or S3DStoreItem.NoStoreItem
EquipItemFromHandNoReturn (bool isLeftHand)	Attempt to Equip an interactive-enabled object currently held in a hand onto the first available compatible Equip Point. This API is callable from inspector events. See also EquipItemFromHand(..).
EquipItemNoReturn (StickyInteractive itemToEquip)	Attempt to Equip an interactive-enabled object on the first available compatible Equip Point. Equipped objects become unselected if they were previously selected. This API is callable from inspector events. See also EquipItem(..).
EquipLookedAtInteractive()	If an interactive-enabled object is being looked at, attempt to equip it on the first available Equip Point.
EquipLookedAtInteractive (S3DEquipPoint equipPoint)	If an interactive-enabled object is being looked at, attempt to equip it on the Equip Point provided.
EquipLookedAtInteractiveNoReturn	If an interactive-enabled object is being looked at, attempt to equip it on the first available Equip Point. This API is callable from inspector events. See also EquipLookedAtInteractive(..).
FindEquipPoint (int stickyInteractiveID)	Attempt to find an equip point that has a given interactive object attached.
GetEquipItem (int storeItemId)	Get the S3DStoreItem attached to an equip point.
GetEquipPoint (int guidHash)	Get a Equip Point using the unique guidHash. See also GetEquipPointByIndex()
GetEquipPointByIndex (int equipPointIndex)	Get a Equip Point from the zero-based index from the list.
GetEquipPointIndexByName (string equipPointName)	Get a Equip Point by name. This will likely impact GC so don't use in an Update loop or get it each frame. Where possible, use GetEquipPoint() or GetEquipPointByIndex().
GetEquipPointIndex (int guidHash)	Return the zero-based index of a S3DEquipPoint in the list using the unique guidHash. Returns -1 if not found.
GetEquipPointPosition (S3DEquipPoint equipPoint)	Get the world space position of an equip point. If the point is null or the parentTransform of the point is null, this will return the current position of the character.
GetEquipPointRotation (S3DEquipPoint equipPoint)	Get the world space rotation of an equip point. If the point is null or the parentTransform of the point is null, this will return the current position of the character.
GetFirstAvailableEquipPoint()	Find the first equip point with at least one available slot.

Method	Description
GetFirstAvailableEquipPoint (StickyInteractive stickyInteractive)	Find the first equip point with at least one available slot and is compatible with the interactive-enabled object. The interactive tag will be compared with those that are permitted for the Equip Point.
GetFirstAvailableEquipPointIndex()	Find the first zero-based equip point index with at least one available slot. Returns -1 if not found.
GetFirstAvailableEquipPointIndex (StickyInteractive stickyInteractive)	Find the first zero-based equip point index with at least one available slot and is compatible with the interactive-enabled object. The interactive tag will be compared with those that are permitted for the Equip Point. Returns -1 if not found.
GetLastSocketableEquipItem (S3DEquipPoint equipPoint, StickySocket stickySocket)	Attempt to find the last added interactive object on the given Equip Point, that is compatible with the given Socket.
GetLeftSideEquippedItem()	Attempt to get an interactive object found on an Equip Point from the left-hand side of the character.
GetRightSideEquippedItem()	Attempt to get an interactive object found on an Equip Point from the right-hand side of the character.
GrabEquipped (bool isLeftHand)	Attempt to grab the first equipped item from the specified left or right-hand side of the character with the specified left or right hand.
GrabEquipped (S3DStoreItem storeItem, bool isLeftHand)	Attempt to grab an equipped item with the specified left or right hand.
GrabLeftHandEquipped()	Attempt to grab the first equipped item from the left-hand side of the character with the left hand.
GrabRightHandEquipped()	Attempt to grab the first equipped item from the right-hand side of the character with the right hand.
SetEquipPointTransform (int equipPointIndex, Transform parentTransform)	Set the parent transform of the Equip Point using the zero-based index.
ReinitialiseEquip()	Reinitialise the Equip points. These hold inactive interactive objects attached to the body of the character.
ToggleEquipGrabWeapon (bool isLeftHand)	If a weapon is held in the specified hand, attempt to equip it to the first available equip point on the same side of the character. If a weapon is NOT held in the specified hand, attempt to get it from the first equip point on that side of the character with a weapon attached.
ValidateEquipPoints()	Ensure the Equip Points are correctly configured. If the equipped items have been changed, call ReinitialiseEquip() instead.

Sticky Control Module Methods – Engage (Grab)

These methods help you grab interactive-enabled objects with a humanoid character.

Method	Description
GrabFromLookingAtSocket (bool isLeftHand)	Attempt to grab the most recent item added to the socket that the character is looking toward with the specified left or right hand. Return true if successful, else false.
GrabFromLookingAtSocketNoReturn (bool isLeftHand)	Attempt to grab the most recent item added to the socket that the character is looking toward with the specified left or right hand. Same as GrabFromLookingAtSocket(..)except can be called from a StickyInputModule custom input event.
GrabLeftHandFromSocket (StickySocket stickySocket)	Attempt to grab the last added StickyInteractive item to the socket with the left hand. This will ignore any items recently added but already removed from the socket.

Method	Description
GrabLeftHandInteractive (StickyInteractive stickyInteractive)	Attempt to grab the interactive-enabled object with the left hand using 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.
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.
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.
GrabRightHandFromSocket (StickySocket stickySocket)	Attempt to grab the last added StickyInteractive item to the socket with the right hand. This will ignore any items recently added but already removed from the socket.
GrabRightHandInteractive (StickyInteractive stickyInteractive)	Attempt to grab the interactive-enabled object with the right hand using the primary 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.

Sticky Control Module Methods – Engage (Socket)

These methods help your character work with the StickySockets. See also the "StickySocket" chapter, and the "Sticky Socket API Methods", "Sticky Socket API Call backs", and "Sticky Socket Properties" sections later in this chapter.

Method	Description
LookAtSocketFromLeftHand()	If the character is looking at a StickySocket, attempt to place the interactive object in the left hand into the socket. Returns the storeItemID for the item attached to the socket. If no held object is added to the socket, S3DStoreItem.NoStoreItem is returned.
LookAtSocketFromLeftHandNoReturn()	If the character is looking at a StickySocket, attempt to place the interactive object in the left hand into the socket. Same as LookAtSocketFromLeftHand(..), except can be called from a StickyInputModule custom input event.
LookAtSocketFromRightHand()	If the character is looking at a StickySocket, attempt to place the interactive object in the right hand into the socket. Returns the storeItemID for the item attached to the socket. If no held object is added to the socket, S3DStoreItem.NoStoreItem is returned.
LookAtSocketFromRightHandNoReturn()	If the character is looking at a StickySocket, attempt to place the interactive object in the right hand into the socket. Same as LookAtSocketFromRightHand(..), except can be called from a StickyInputModule custom input event.
LookedAtSocketFromStash()	If the character is looking at a StickySocket, attempt to place the most recently stashed compatible interactive object into the socket. Returns

Method	Description
	the storeItemID for the item attached to the socket. If no stashed object is added to the socket, S3DStoreItem.NoStoreItem is returned.
LookedAtSocketFromStashNoReturn()	If the character is looking at a StickySocket, attempt to place the most recently stashed compatible interactive object into the socket. Same as LookedAtSocketFromStash(..), except can be called from a StickyInputModule custom input event.
SocketFromEquip (StickySocket stickySocket)	Attempt to find the first Equipped interactive object that is compatible with the given socket, and attach it to the socket. Returns the storeItemID for the item attached to the socket. If no equipped object is added to the socket, S3DStoreItem.NoStoreItem is returned.
SocketFromEquipNoReturn (StickySocket stickySocket)	Attempt to find the first Equipped interactive object that is compatible with the given socket, and attach it to the socket. Same as SocketFromEquip(..), except can be called from a StickyInputModule custom input event.
SocketFromLeftHand (StickySocket stickySocket)	Attempt to place the interactive-enabled object held in the left hand, into the StickySocket. Returns the storeItemID for the item attached to the socket. If it is not added, S3DStoreItem.NoStoreItem is returned.
SocketFromLeftHandNoReturn (StickySocket stickySocket)	Attempt to place the interactive-enabled object held in the left hand, into the StickySocket. Same as SocketFromLeftHand(..), except can be called from a StickyInputModule custom input event.
SocketFromRightHand (StickySocket stickySocket)	Attempt to place the interactive-enabled object held in the right hand, into the StickySocket. Returns the storeItemID for the item attached to the socket. If it is not added, S3DStoreItem.NoStoreItem is returned.
SocketFromRightHandNoReturn (StickySocket stickySocket)	Attempt to place the interactive-enabled object held in the right hand, into the StickySocket. Same as SocketFromRightHand(..), except can be called from a StickyInputModule custom input event.
SocketFromStash (StickySocket stickySocket)	Attempt to find the most recently Stashed interactive object that is compatible with the given socket, and attach it to the socket. Returns the storeItemID for the item attached to the socket. If no stashed object is added to the socket, S3DStoreItem.NoStoreItem is returned.
SocketFromStashNoReturn (StickySocket stickySocket)	Attempt to find the most recently Stashed interactive object that is compatible with the given socket, and attach it to the socket. Same as SocketFromStash(..), except can be called from a StickyInputModule custom input event.

Sticky Control Module Methods – Engage (Stash)

These methods help you work with the character "Stash" (also known as Inventory).

Method	Description
DestroyStashItem (S3DStoreItem storeItem)	Destroy the StoreItem.
DestroyStashItem (int storeItemID)	Destroy the store item given a S3DStoreItem.StoreItemId.
GetFirstGrabbableStashItem (int IgnoreInteractiveID)	Get the S3DStoreItem of the first Grabbable item that was Stashed. Ignores items that have already been removed or matches the IgnoreInteractiveID. When this value is 0, it has no effect.
GetLastGrabbableStashItem (int IgnoreInteractiveID)	Get the S3DStoreItem of the last Grabbable item that was Stashed. Ignores items that have already been removed or matches the IgnoreInteractiveID. When this value is 0, it has no effect.

Method	Description
GetLastSocketableStashItem (StickySocket stickySocket, int IgnoreInteractiveID)	Get the S3DStoreItem of the last compatible Socketable item that was Stashed. Ignores items that have already been removed or one with a matching IgnoreInteractiveID. Returns the storeItemID for the Stashed item. If no stashed object is found, S3DStoreItem.NoStoreItem is returned.
GetLastStashitem()	Get the S3DStoreItem of the last item that was Stashed. Ignores items that have already been removed.
GetLastStashitemID ()	Get the StoreItemID of the last item that was Stashed. Ignores items that have already been removed.
GetLastStashInteractiveID()	Get the StickyInteractiveID of the last item that was Stashed. Ignores items that have already been removed.
GetLastStashMagazineItemID (int magTypeInt, int[] compatibleAmmoTypes, bool allowEmpty)	Get the last stashed Magazine of a given MagType, that has an ammoType from the array provided. For performance reasons, the types are provided as their integer values from the enumerations. Return the storeItemID or S3DStoreItem.NoStoreItem
GetStashItem (int storeItemID)	Get the store item given a S3DStoreItem.StoreItemId.
GetStashItemID (int stickyInteractiveID)	Get the StoreItemID of an interactive-enabled object using the StickyInteractiveID. Returns the StoreItemID or S3DStoreItem.NoStoreItem.
GetStashItemID (StickyInteractive stashItem)	Get the StoreItemID of an interactive-enabled object using a StickyInteractive item. Returns the StoreItemID or S3DStoreItem.NoStoreItem.
GetNumberStashedMagazines (int magTypeInt, int[] compatibleAmmoTypes, bool allowEmpty)	Get the number of magazines in the Stash of a given MagType, and has an ammoType from the array provided. For performance reasons, the types are provided as their integer values from the enumerations.
GrabItemFromStash (bool isLeftHand)	Get the last grabbable item that was stashed, and grab it with the left or right hand. If there are no grabbable items stashed or the hand is already holding an interactive object, the method will do nothing. NOTE: Animate must be enabled to grab items.
ReinitialiseStash()	Reinitialise the Stash (inventory). This is automatically called when the character is initialised.
SetStashParent (Transform newStashParent)	Set the child transform under which all items are stashed.
StashFromLookedAtSocket()	Attempt to stash the most recent item added to the socket that the character is looking toward. Returns the Stash StoreItemID or S3DStoreItem.NoStoreItem.
StashFromLookedAtSocketNoReturn()	Attempt to stash the most recent item added to the socket that the character is looking toward. Same as StashFromLookedAtSocket(), except can be called from a StickyInputModule custom input event.
StashFromSocket (StickySocket stickySocket)	Attempt to stash the most recent item added to the socket. Returns the Stash StoreItemID or S3DStoreItem.NoStoreItem.
StashFromSocketNoReturn (StickySocket stickySocket)	Attempt to stash the most recent item added to the socket. Same as StashFromSocket(..), except can be called from a StickyInputModule custom input event.
StashItem (StickyInteractive itemToStash)	Add an interactive item to the stash (inventory). Returns the StoreItemID or S3DStoreItem.NoStoreItem.
StashItemNoReturn (StickyInteractive itemToStash)	Attempt to Stash a StickyInteractive item. A fire and forget version of StashItem(..). Can be called from a StickyInputModule custom input event.
StashItemFromHand (bool isLeftHand)	If the character is holding an interactive object in their hand, attempt to add it to the Stash (inventory). See also StashItemFromLeftHand() and

Method	Description
	StashItemFromRightHand().This has no return code, so can be called from a StickyInputModule custom input event.
StashItemFromLeftHand()	If the character is holding an interactive object in their left hand, attempt to add it to the Stash (inventory). Returns the StoreItemID or S3DStoreItem.NoStoreItem.
StashItemFromRightHand()	If the character is holding an interactive object in their left hand, attempt to add it to the Stash (inventory). Returns the StoreItemID or S3DStoreItem.NoStoreItem.
StashLookedAtInteractive()	Attempt to stash the interactive-enabled (if any) item being looked at. Returns the StoreItemID or S3DStoreItem.NoStoreItem.
StashLookedAtInteractiveNoReturn()	Attempt to stash the interactive-enabled (if any) item being looked at. Same as StashLookedAtInteractive(), except can be called from a StickyInputModule custom input event.
SwitchStashedItem (bool isLeftHand)	If holding an item, attempt to Stash it. Then hold the next available grabbable item from Stash. If no other items are Stashed, the character will no longer be holding an item in the hand indicated.

Sticky Control Module Methods – Engage (Weapons)

These methods help you work with weapons held by a character.

Method	Description
GetNumberHeldMagazines (int magTypeInt, int[] compatibleAmmoTypes, bool allowEmpty)	Get the number of magazines held in the left, right or both hands of a given MagType, and has an ammoType from the array provided. Currently the maxium number would be 2 (one in each hand). For performance reasons, the types are provided as their integer values from the enumerations.
PauseWeapons()	Pause weapons that are held.
PauseWeaponsFiring()	Prevent weapons that are held from firing.
ReloadWeaponLeftHand (int weaponButtonNumber = 1)	Attempt to reload a weapon held in the left hand.
ReloadWeaponRightHand (int weaponButtonNumber = 1)	Attempt to reload a weapon held in the right hand.
ReinitialiseWeapons()	Initialise or reinitialise weapon settings for this character. NOTE: Currently this does NOT reinitialise the actual weapons.
StartAimingWeapon (bool isLeftHand)	If the character is holding a weapon, attempt to start aiming down the sights or through the Scope.
StopAimingWeapon (bool isLeftHand)	If the character is holding a weapon, stop aiming down the sights or through the Scope.
SwitchStashedWeapon (bool isLeftHand)	If holding a weapon, attempt to Stash it. Then hold the next available grabbable weapon from Stash. If no other weapons are Stashed, the character will no longer be holding a weapon in the hand indicated.
ToggleAimWeapon (bool isLeftHand)	If the character is holding a weapon, enable or disable the character aiming down the sights or through the Scope.
ToggleLaserSight (bool isLeftHand)	If the character is holding a weapon, and it has an equipped laser sight, turn it on or off.
ToggleWeaponScope (bool isLeftHand)	If the character is holding a weapon, and it has an equipped Scope for more precise visual aiming, turn it on or off.
UnpauseWeapons()	Attempt to resume weapons that are held.
UnpauseWeaponsFiring()	Allow weapons that are held to fire.

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. 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
AimBones	Get or set the Humanoid bones used to help aim the character toward a target while holding a weapon.
AimIKFPWeaponOffset	Get or set the weapon local space offset from the first-person camera when aiming. This can be overridden by a Weapon Anim Set.
AimIKFPNearClippingPlane	Get or set the first-person camera near clipping plane when aiming a weapon. NOTE: This does not change the camera, it only stores the value to be used when aiming. This can be overridden by a Weapon Anim Set.
AimIKTurnDelay	Get or set the time, in seconds, to delay the character turning to face the target when aiming starts. This allows time to transition from a held animation, to an aiming animation.
EngageColour	Get or set the colour associated with engaging with an interactive-enabled object in the scene. Typically used by the StickyDisplayModule reticle when is hovering over an object.
Nonengage Colour	Get or set the colour associated with not engaging with an interactive-enabled object in the scene. Typically used by the StickyDisplayModule reticle when not hovering over an object.
GetCurrentLocalVelocity	[R] Get the velocity of the character relative to the reference object.
GetCurrentForward	[R] Get the current forward direction the character is facing
CurrentMaxHandIKWeight	[R] The current maximium Hand IK weight available for use in the animator. When smooth enabling or disabling, the maximum weight possible can be between 0.0 and 1.0. Hand IK weight is then limited to this value in the OnAnimatorIK() pass.
GetCurrentInverseRotation	[R] 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
GetCurrentReferenceFrameID	[R] The ID (HashCode) for the reference frame transform.
GetCurrentWorldVelocity	[R] Get the current (calculated) character world space velocity.
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
IsAimAtTarget	Is the (humanoid) character currently attempting to aim toward the target? This is typically used when holding a weapon. For more control see EnableAimAtTarget(..).
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?

Property	Description
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?
IsFreeLookWhenWeaponHeld	[R] If a non-NPC character has Free Look enabled before a weapon is held (but not aimed), does Free Look remain enabled? Useful when a weapon is held in a relaxed pose not pointing forward.
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?
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?
IsLeftHandHoldingMagazine	[R] Is the left hand holding and interactive-enabled magazine?
IsLeftHandHoldingWeapon	[R] Is the left hand holding and interactive-enabled weapon?
IsLeftHandHoldingInteractive	[R] Is the left hand holding an interactive-enabled object?
IsLockCamToWorldPos	Get or attempt to set (un)locking the third-person camera to the current world space position, relative to the reference frame.
IsLookEnabled	[R] Is the player able to look around with the camera? When disabled, the camera will be disabled.
IsLookCameraFollowHead	[R] If the relative head bone position changes, the first-person camera will move relative to it.
IsLookCameraFollowHeadTP	[R] If the relative head bone position changes, the third-person camera will move relative to it. Currently this has no effect when Head IK is enabled.
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 a StickyInteractive component?
IsLookingAtInterative	[R] Is the character currently looking at an object with a StickyInteractive component?
IsLookSocketAutoShow	Get or set if the StickySockets will be highlighted when looked at and Look Sockets is enabled.
IsLookSocketsEnabled	[R] Can the character see or detect objects with a StickySocket component?
IsLookingAtSocket	[R] Is the character currently looking at an object with a StickySocket 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?
IsLookRoomScaleVR	[R] Is the VR Head Mounted Device (HMD) driving character motion?
IsLookVRAvailable	[R] Is Look VR mode available for selection or use? You need to be in first-person, not an 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?

Property	Description
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?
IsRagdoll	Get or set if the character is acting like a ragdoll
IsRightHandHoldingMagazine	[R] Is the right hand holding and interactive-enabled magazine?
IsRightHandHoldingWeapon	[R] Is the right hand holding and interactive-enabled weapon?
IsRightHandHoldingInteractive	[R] Is the right hand holding an interactive-enabled object?
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] Is the point in world space where the user is currently focusing, being updated? Requires Look Interactive to be enabled.
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.
LeftHandInteractive	[R] Get a reference to the item currently being held in the character's left hand
LeftHandInteractiveID	[R] Get the ID of the interactive-enabled object held in the character's left hand
MovingSpeed	[R] The speed in metres per second the character is moving in any direction.
MovingSpeedInvN	[R] Given the maximum walk or sprint speed, calculate what number, when multiplied by the current speed, would give a range of 0.0 to 1.0.
MovingSpeedN	[R] Given the maximum walk or sprint speed, find the normalised (0.0 t0 1.0) value from 0 to max speed.
MovingForwardSpeed	[R] The speed in metres per second the character is moving forward
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
CharacterRigidBody	[R] Get the main rigidbody for the character controller
RagdollBoneList	Get or set the list of bones used for the Ragdoll feature.
RightHandInteractive	[R] Get a reference to the item currently being held in the character's right hand
RightHandInteractiveID	[R] Get the ID of the interactive-enabled object held in the character's right hand
StickyID	[R} The runtime identification number of the character.
StrafingSpeed	[R] The speed in metres per second the character is moving left or right.
StrafingSpeedN	[R] Given the maximum strafe or sprint speed, find the normalised (0.0 t0 1.0) value from 0 to max speed that the character is moving left or right.
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

Property	Description
ZoomAmount	[R] Get the current amount of zoom being applied in third person mode

Sticky Control Module API Call Backs

These (delegates) are used when you wish to call a custom method in your own game code. For example, when the character is about to be destroyed (and therefore has 0 health), you might want to update a UI scoreboard. Be careful not to hold references to objects passed as parameters outside your method otherwise Unity's garbage collector may not be able to release some things from memory.

Method field	Description
CallbackOnChangeCamera callbackOnCameraChange	The name of your custom method that is called immediately after the look camera is changed.
CallbackOnChangeInteractiveTarget callbackOnChangeInteractiveTarget	The name of your custom method that is called immediately after an interactive target is changed
CallbackOnChangeLookAtInteractive callbackOnChangeLookAtInteractive	The name of your custom method that is called immediately after the character changes the interactive object they are looking at.
CallbackOnDestroy callbackOnDestroy	The name of the custom method that is called immediately before the character is destroyed (or when the health reaches 0). Your method must take 1 parameter: StickyControlModule. This parameter will never be null. This should be a lightweight method to avoid performance issues and not hold references past the current frame. See also onDestroyed event.
CallbackOnHit callbackOnHit	The name of the custom method that is called immediately after the character is hit by a projectile or beam. Your method must take 1 parameter of type CallbackOnS3DHitParameters. This should be a lightweight method to avoid performance issues. It could be used to take evasive action while being pursued by an enemy NPC. It could also be used to detect friendly fire.
CallbackAfterRespawn callbackAfterRespawn	The name of the custom method that is called immediately after the character is respawned. Your method must take 1 parameter: StickyControlModule. This parameter will never be null. This should be a lightweight method to avoid performance issues.
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.

Sticky Display Module API Properties

These can be found under "Public Properties" 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

Method	Description
	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.
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.

Method	Description
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
ChangeDisplayReticle (int guidHash)	Change the DisplayReticle sprite on the UI panel. See also GetDisplayReticleGuidHash(..).
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.
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.
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.
GetDisplayReticleSprite (int guidHash)	Get the UI sprite (image) for a DisplayReticle. See also GetDisplayReticleGuidHash(..).
HideDisplayReticle()	Hide or turn off the Display Reticle
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.
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.
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)
ShowDisplayReticle()	Show the Display Reticle on the display. The display will automatically be shown if it is not already visible.
ToggleDisplayReticle()	Show or Hide the Display Reticle. The display will be shown if required.
ValidateReticleList()	Create a new list if required

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(..).

Method	Description
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
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

Method	Description
	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.
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.

Method	Description
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.
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;

Method	Description
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)	Get the (sprite) name of the DisplayTarget using either the DisplayTarget instance or the zero-based index in the list of

Method	Description
(int index)	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 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 (Special Purpose) API Methods

Most methods require that the StickyDisplayModule to be initialised.

Method	Description
InteractiveLookAtChanged (int StickyID, int oldLookAtInteractiveID, int lookingAtInteractiveId, Color32 reticleColour)	This is a special method that can be called from the Engage Tab on a S3D character when looking at interactive-enabled objects in the scene. See the “On Look At Changed” event on the StickyControlModule for more details.

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);

Method	Description
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(..)
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 Anim Replacer Properties and Methods

These methods are used with the Sticky Anim Replacer component that you may have attached to your character. See the “Sticky Anim Replacer” chapter for more details.

Method	Description
AddAnimClipSet (S3DAnimClipSet s3dAnimClipSet)	Add animation clip set to the StickyAnimReplacer
Initialise()	Initialise the Sticky Anim Replacer. Call this if you haven’t enabled “Initialise On Start” in the editor.
IsInitialised	Is this Sticky Zone initialised?
NumAnimClipSets	Get the number of animation clip sets for this Anim Replacer.
ReplaceAnimClipSet (int clipSetNumber)	Replace the animation on the character with those in the Anim Clip Set using the number of the Animation Clip Set on the component.
RestoreAnimClipSet (int clipSetNumber)	Restore the animation on the character with the original ones in the Anim Clip Set using the number of the Animation Clip Set on the component.
RestoreAnimClipSetDelayed (int clipSetNumber)	Restore the animation on the character after a 2 second delay with the original ones in the anim clip set using the number of the Animation Clip Set on the component. This will impact GC first time it runs but saves creating a WaitForSeconds if is never called.
RemoveAnimClipSet (S3DAnimClipSet s3dAnimClipSet)	Remove an animation clip set from the StickyAnimReplacer.
RemoveAnimClipSet (int index)	Remove the zero-based index of an Animation Clip Set assigned to this StickyAnimReplacer.
ToggleAnimClipSet (int clipSetNumber)	Replace or restore a set of animation clips on the character using the Animation Clip Set number on this component.

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
AddRigidbody (bool isKinematic)	Add a rigidbody to the same gameobject. See also HasRigidbody and ObjectRigidbody properties.
AllocateSeat()	Allocate or reserve a seat. This should be used to indicate the seat is taken or occupied. Use IsSeatAllocated property to get the current status.
CheckCanBeEquipped (StickyControlModule equippedBy, S3DEquipPoint equippedAt)	Check if this interactive-enabled object can be equipped onto a EquipPoint on the given character.
CheckCanBeStashed (StickyControlModule stashedBy)	Check if this interactive-enabled object can be placed into the Stash (personal inventory) of the given character. NOTE: By design, it doesn't check if the item is currently attached to a StickySocket.
ClearLasso()	Typically called automatically when a character lasso operation completes.
DeallocateSeat()	Deallocate or unreserve a seat. This should be used to indicate the seat is vacant. Use IsSeatAllocated property to get the current status.
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(..).
GetCharacterReferenceFrame ()	If the object is held, and using a Gravity Mode of Reference Frame, return the current reference frame of the character.
GetEquipPosition()	Get the world space equip point position.
GetEquipRotation()	Get the world space equip point rotation.
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.
GetHandHoldNormal (bool isSecondaryHoldPosition)	Get the world space hand hold normal (or direction it is facing). 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.
GetHandHoldRotation (bool isSecondaryHandHold)	Get the world space hand hold rotation. By default, uses the first or primary hand hold.
GetInteractiveTag()	Get the 32-bit mask used to determine compatibility with things like StickySockets or character Equip Points. Default is 1 << 0.
GetInteractiveTags()	Get the common interactive tags scriptableobject used to determine compatibility with things like StickySocket and character Equip Points.
GetLocalPosition (Vector3 wsPosition)	Get a local space position on the interactive-enabled object, given a world space position (converts a world space position to a local space position on the interactive-enabled object).
GetLocalRotation (Quaternion wsRotation)	Get a local space rotation of a rotated object relative to the interactive-enabled object. (Converts a world space rotation to a local space rotation on the interactive-enabled object).

Method	Description
GetPopupPosition()	Get the world space default popup position. Takes into consideration “Use Ref Frame Up” if enabled.
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.
GetSocketPosition()	Get the world space socket attach point position.
GetSocketRotation()	Get the world space socket attach point rotation.
ReinitialiseColliders()	Call this if you add or remove colliders at runtime
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.
SetInteractiveTag (int bitMask)	Set the 32-bit mask used to determine compatibility with things like StickySockets or character Equip Points. Default is 1 << 0.
SetInteractiveTags (S3DInteractiveTags newInteractiveTags)	Set the common interactive tags scriptableobject used to determine compatibility with things like StickySocket and character Equip Points
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.
SetIsEquippable (bool equippable)	Set this interactive-enabled object to be equippable or not.
SetIsGrabbable (bool grabbable)	Set this interactive-enabled object to be grabble or not.
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.
SetIsSocketable (bool socketable)	Set this interactive-enabled object to be attachable to a StickySocket or not.
SetIsStashable (bool stashable)	Set this interactive-enabled object is stashable 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.

Method	Description
SetSSCSeatAnimator (GameObject seatAnimatorGameObject)	If Sci-Fi Ship Controller is installed, this can link to a SSC Seat Animator component used for a multi-stage seating operation. For example, animating a chair before and after sitting.
ToggleActivateObject (int stickyID)	Attempt to Activate or Deactivate an interactive-enabled object that Is Activable.

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 (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 (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 (stickyID);
	  // Do stuff here
	}

virtual bool EquipObject (StickyControlModule equippedBy, S3DEquipPoint equippedAt)	This is automatically called by stickyControlModule.EquipInteractiveInternal(..). Automatically deactivate IsActivable objects before it is equipped.

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 (stickyID);
	  // Do stuff here
	}

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.

Virtual Method	Description

	public override void Initialise ()
	{
	  base. Initialise ();
	  // Do stuff here
	}

PostEquipObject (int stickyID, S3DEquipPoint equipPoint, int storeItemID)	This is called (automatically) immediately after this interactive-enabled object is equipped by a S3D character. if required, invoke any onPostEquipped items. If required, you can override this method.
	
	public override void PostEquipObject (int stickyID, S3DEquipPoint equipPoint)
	{
	  base.onPostEquipped (stickyID, equipPoint, storeItemID);
	  // Do stuff here
	}

virtual void PostGrabObject (int stickyID, bool isSecondaryHandHold)	This is called (automatically) immediately after this interactive-enabled object is grabbed by a S3D character. if required, invoke any OnPostGrabbed items. If required, you can override this method.
	
	public override void PostGrabObject (int stickyID, bool isSecondaryHandHold)
	{
	  base.PostGrabObject (stickyID, isSecondaryHandHold);
	  // Do stuff here
	}

virtual void PostStashObject (int stickyID, int storeItemID, Vector3 futureValue)	This is called (automatically) immediately after this interactive-enabled object is stashed by a S3D character. if required, invoke any OnPostStashed items. If required, you can override this method.
	
	public override void PostGrabObject (int stickyID, int storeItemID, Vector3 futureValue)
	{
	  base.PostStashObject (stickyID, storeItemID, futureValue);
	  // Do stuff here
	}

virtual void ReadObject()	Read data from the object. Typically, read the x and/or z-axis values from a lever or joystick.
virtual void RecentreReadable (bool isSmooth)	Return the readable lever or joystick back to the original location.
virtual void RecentreWriteable (bool isSmooth)	Return the writeable lever or joystick back to the original location.

virtual void ReInitialiseReadable()	Initialise this Readable object.
	public override void ReInitialiseReadable()
	{
	  base.ReInitialiseReadable();
	  // Do stuff here
	}

virtual bool StashObject (StickyControlModule stashedBy)	This is automatically called by stickyControlModule.StashItem(..).
virtual void SetSticky3DCharacter	Which character (if any) has this interactive-enabled object? NOTE: You need to set IsHeld or IsStashed separately.

Virtual Method	Description
(StickyControlModule sticky3DCharacter)	

virtual void SelectObject (int stickyID, int selectedStoreItemID)	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.
	
	public override void SelectObject (int stickyID, int selectedStoreItemID)
	{
	  base.SelectObject (int stickyID, int selectedStoreItemID);
	  // Do stuff here
	}

virtual void SetStickySocket (StickySocket stickySocketOwner)	Which socket (if any) is this interactive-enabled object attached to? NOTE: You need to set IsSocketed separately.

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 (hitPoint, hitNormal, 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 (stickyID);
	  // 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 (stickyID);
	  // Do stuff here
	}

virtual void WriteToObject (Vector3 writeValue, bool isFixedUpdate)	Attempt to update the position of the writeable interactive object.

Sticky Interactive Properties

Property	Description
CurrentReadableValue	When Readable, when will return the current value of the jointed object. Will be in the range -1.0 to 1.0.
GetCurrentPosition	Get the current world space position of the interactive-enabled object.
GetCurrentRotation	Get the current world space rotation of the interactive-enabled object.

Property	Description
GravitationalAcceleration	Get or set the gravity in metres per second per second
GravityDirection	Get or set the world space direction that gravity acts upon the weapon when GravityMode is Direction.
GravityMode	Get or set the method used to determine in which direction gravity is acting.
HasRigidbody	Is there a rigidbody currently attached to the object?
IsActivable	Can the item be activated?
IsActivated	Has the interactive-enabled object been activated? This should always be false when IsActivable is false.
IsAutoDeactivate	Does the activated item act like a single on/off action?
IsAutoUnselect	Does the selected item act like a clickable button?
IsEquipped	Is this interactive object currently Equipped by a Sticky3D character?
IsEquippedByNPC	Is the interactive object currently equipped on a Non-Player Sticky3D character?
IsEquippable	Can this item be equipped or attached to the body?
IsGrabbable	Can this item be grabbed or held?
IsHeld	Is the interactive object currently held by a Sticky3D character?
IsHeldByNPC	Is the interactive object currently held by a Non-Player Sticky3D character?
IsInitialised	Is the Sticky Interactive-enabled object initialised and ready for use?
IsReadable	Can values be read from this object? This is typically used for reading the position of a virtual lever or joystick.
IsReadableReady	Is the interactive object ready to provide readable data? This typically means it is initialised, Readable, and has a valid pivot transform assigned.
IsSeatAllocated	Is the seat allocated, reserved, or taken? See also AllocateSet(), DeallocateSeat()
IsSelectable	Can this item be selected in the scene?
IsSittable	Is this item suitable for sitting on?
IsSocketable	Can this item be attached to a StickySocket?
IsSocketed	Is the interactive object currently attached to a socket?
IsStashable	Can this item be stashed in the inventory of a character?
IsStashed	Is the interactive object currently in the Stash of a Sticky3D character?
IsStickyMagazine	Is this an interactive-enabled StickyMagazine?
IsStickyWeapon	Is this an interactive-enabled StickyWeapon?
IsTouchable	Can this item be touched by a character? Typically used with Hand IK.
IsUseGravity	Get if the object is affected by gravity
IsWriteable	Can values be written to this object? This is typically used for setting the position of a virtual lever or joystick.
Mass	The mass of the object in kilograms.
ObjectRigidbody	Get the rigidbody attached to the interactive-enabled object (if any)
StickyInteractiveID	Runtime ID of this Sticky Interactive component.
Sticky3DCharacter	Get or set the S3D character that has this interactive-enabled object. NOTE: You need to get or set IsHeld or IsStashed separately.
Sticky3DCharacterID	If the object is held, equipped or stashed, get the StickyID for the character that currently possesses it. Otherwise return 0.
StickySocketedOn	Get or set which socket (if any) is this interactive-enabled object attached to? NOTE: You need to set IsSocketed separately.

Sticky Interactor Bridge - Methods

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 Interactor Bridge - Properties

These properties 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.

Property	Description
HandGripValue	Get the current grip value from the VR hand controller.
HandTriggerValue	Get the current trigger value from the VR hand controller.
IsInitialised	Has the module been initialised?
IsCanTouch	Can this item touch an interactive-enabled Touchable object?
IsCanGrab	Can this item grab an interactive-enabled Grabbable object?
IsHeld	Is this component currently holding an interactive-enabled Grabbable object?
IsTouching	Is this component currently touching an interactive-enabled Touchable object?

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

Method	Description
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.
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 (int sceneHandle = 0)	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. For multi-additive scenes, pass in the current scene handle e.g., gameObject.scene.handle.
Initialise()	Initialise the Sticky Manager. This is called automatically when GetOrCreatemanager() is called.
CheckCharacterHitByProjectile (Vector3 fromPosition, Vector3 direction, float distance, LayerMask	Has a Sticky3D character been hit by a projectile? Applies damage and impact force if hit. Invokes callbackOnHit if configured on hit character.

Method	Description
collisionLayerMask, out RaycastHit hitInfo, StickyProjectileModule projectile, bool skipFastCheck)	
CheckObjectHitByBeam (Vector3 fromPosition, Vector3 direction, float distance, LayerMask collisionLayerMask, out RaycastHit hitInfo, StickyBeamModule beamModule, float hitDuration, bool skipFastCheck)	Has a beam hit? 1) A Sticky3D character? 2) A rigidbody with StickyDamageReceiver attached? 3) A non-trigger collider with a StickyDamageReceiver attached? 4) A regular non-trigger collider If hit: 1) Applies damage and invokes callbackOnHit on hit character (if any) 2) Applies damage to StickyDamageReceiver and invokes callbackOnHit on hit receiver (if any)
CheckObjectHitByProjectile (Vector3 fromPosition, Vector3 direction, float distance, LayerMask collisionLayerMask, out RaycastHit hitInfo, int sourceStickyID, StickyProjectileModule projectile, bool skipFastCheck)	Has a projectile hit? 1) A Sticky3D character? 2) A rigidbody with StickyDamageReceiver attached? 3) A rigidbody with StickyDynamicModule attached? 4) A non-trigger collider with a StickyDamageReceiver attached? 5) A regular non-trigger collider If hit: 1) Apply damage and force and invoke callbackOnHit on hit character (if any) 2) Apply force on dynamic module (if any) 3) Apply damage and invoke callbackOnHit on hit damage receiver (if any)
GetHitOrMaxPoint (Vector3 fromPosition, Vector3 direction, float distance, LayerMask collisionLayerMask, int sourceStickyID, bool isWeapon)	Get the point where we hit a non-trigger collider on an object, or a Sticky3D character. If nothing was hit, return the point at the distance in the desired direction. If isWeapon is true, on characters, only weapon hittable trigger colliders will be detected.

Sticky Manager Methods – Ammo

These methods apply to using a AmmoTypes scriptable object.

Method	Description
GetAmmoDamageMultiplier (S3DAmmo.AmmoType ammoType)	Get the Damage Multiplier for an ammo type.
GetAmmoDamageMultiplier (int ammoTypeInt)	Get the Damage Multiplier for an ammo type.
GetAmmoImpactMultiplier (S3DAmmo.AmmoType ammoType)	Get the (force) impact multiplier for an ammo type.
GetAmmoImpactMultiplier (int ammoTypeInt)	Get the (force) impact multiplier for an ammo type.
SetAmmoTypes (S3DAmmoTypes newAmmoTypes)	Set or update the list of (common) ammo types. Generally, only used Internally. You would need a good reason to have to use this in your code. If in doubt, ask us.

Sticky Manager Methods – StickyBeamModule

These methods apply to the StickyBeamModule which is used with the Sticky Manager pooling system.

Method	Description
virtual void DestroyBeam ()	Destroy the beam by cleaning up and returning it to the pool
virtual void DisableModule()	Temporarily disable the module. Typically called automatically from stickyManager.PauseBeams().
virtual void EnableModule()	Re-enable the module. Typically called automatically from stickyManager.ResumeBeams().
GetBeamPrefab (int beamPrefabID)	Returns the prefab for a beam module given its beam prefab ID.
GetOrCreateBeamPool (GameObject beamPrefab)	Get the beam pool for this prefab or create a new pool if one does not already exist. Return the beamPrefabID for the pool or StickyManager.NoPrefID (-1) if it fails. GetorCreateBeamPool (StickyBeamModule beamPrefab) is slightly faster. See also InstantiateBeam(igParms).
GetorCreateBeamPool (StickyBeamModule stickyBeamPrefab)	Get the beam pool for this prefab or create a new pool if one does not already exist. Return the beamPrefabID for the pool or StickyManager.NoPrefID (-1) if it fails. See also InstantiateBeam(igParms).
virtual uint Initialise (S3DInstantiateBeamParameters ibParms)	Initialise the beam module. This is automatically called from the StickyManager pooling system whenever it is spawned (instantiated) in the scene.
InstantiateBeam (ref S3DInstantiateBeamParameters ibParms)	Instantiates the beam with ID beamPrefabID at the position specified in world space and with the forwards and up directions specified. (ibParms.beamPrefabID is the ID sent back to each weapon after calling weapon.ReInitialiseWeapon() method).
PauseBeams()	Pause all Pooled Beams in the scene. This is useful when pausing your game.
ResumeBeams()	Resume all Pooled beams in the scene. This is useful when unpausing your game.

Sticky Manager Methods - StickyDecalModule

These methods apply to the StickyDecalModule which is used with the Sticky Manager pooling system

Method	Description
CreateDecalPools (S3DDecals s3dDecals)	Create any Decal pools that have not already been created using a set of StickyDecalModule prefabs.
CreateDecalPools (S3DDecals s3dDecals, int[] decalPrefabIDs)	Create any Decal pools that have not already been created using a set of StickyDecalModule prefabs. Populate a pre-created array of S3DDecalTemplate prefabIDs. The length of decalPrefabIDs must match number of decal slots.
DestroyDecals (GameObject decalParent)	Destroy (return to pool) any decals that are children of the given gameobject.
GetDecalPrefab (int decalPrefabID)	Returns the prefab for a decal (object) module given its decal prefab ID.
GetOrCreateDecalPool (StickyDecalModule decalPrefab)	Get the decal pool for this prefab or create a new pool if one does not already exist. Return the decalPrefabID for the pool or StickyManager.NoPrefabID (-1) if it fails. See also InstantiateDecal(idParms).
GetOrCreateDecalPool (GameObject decalPrefab)	Get the decal pool for this prefab or create a new pool if one does not already exist. Return the decalPrefabID for the pool or StickyManager.NoPrefID (-1) if it fails.

Method	Description
	GetorCreateDecalPool (StickyDecalModule decalPrefab) is slightly faster. See also InstantiateDecal(idParms).
GetProjectileDecalPrefabID (int projectilePrefabID)	Returns the decalPrefabID for a decal template from a list of default decals associated with a projectile. If the projectilePrebID is not valid, or there are no matching deal module pools, the decalPrefabID will return StickyManager.NoPrefabID (-1).
InstantiateDecal (ref S3DInstantiateDecalParameters idParms)	Instantiates the decal with ID decalPrefabID at the position specified in world space (decalPrefabID is the ID sent back in idParms).
PauseDecals()	Pause all Pooled Decals in the scene
ResumeDecals()	Resume all Pooled Decals in the scene

Sticky Manager Methods – StickyDynamicModule

These methods apply to the StickyDynamicModule which is used with the Sticky Manager pooling system.

Method	Description
GetDynamicPrefab (int dynamicPrefabID)	Returns the prefab for a dynamic (object) module given its dynamic prefab ID.
GetOrCreateDynamicPool (StickyDynamicModule stickyDynamicPrefab)	Get the dynamic object pool for this prefab or create a new pool if one does not already exist. Return the dynamicPrefabID for the pool or StickyManager.NoPrefabID (-1) if it fails. See also InstantiateDynamicObject(idParms).
GetOrCreateDynamicPool (GameObject dynamicPrefab)	Get the dynamic pool for this prefab or create a new pool if one does not already exist. Return the dynamicPrefabID for the pool or StickyManager.NoPrefID (-1) if it fails. GetOrCreateDynamicPool (StickyDynamicModule dynamicPrefab) is slightly faster. See also InstantiateDynamic(igParms).
InstantiateDynamicObject (ref S3DInstantiateDynamicObjectParameters idParms)	Instantiate a dynamic object in the scene using an instance from the pool if possible.
PauseDynamicObjects()	Pause all Pooled Dynamic Objects in the scene
ResumeDynamicObjects()	Resume all Pooled Dynamic Objects in the scene

Sticky Manager Methods – StickyEffectsModule

These methods apply to the StickyEffectsModule which is used with the Sticky Manager pooling system.

Method	Description
InstantiateEffectsObject (ref S3DInstantiateEffectsObjectParameters ieParms)	Instantiates the effects object with ID effectsObjectPrefabID at the position specified in world space. (effectsObjectPrefabID is the ID sent back in ieParms).
CreateEffectsPools (S3DEffectsSet s3dEffectsSet, int[] effectsPrefabIDs)	Create any Effects pools that have not already been created using a set of StickyEffectsModule prefabs. Populate a pre-created array of S3DEffectsTemplate prefabIDs. The length of effectsPrefabIDs must match number of effects slots.
GetEffectsPrefab (int effectsPrefabID)	Returns the prefab for an effects (object) module given its effects prefab ID.
GetOrCreateEffectsPool (StickyEffectsModule effectsPrefab)	Get the effects object pool for this prefab or create a new pool if one does not already exist. Return the effectsPrefabID for the pool or

Method	Description
	StickyManager.NoPrefabID (-1) if it fails. See also InstantiateEffectsObject(ieParms).
GetOrCreateEffectsPool (GameObject effectsPrefab)	Get the effects pool for this prefab or create a new pool if one does not already exist. Return the effectsPrefabID for the pool or StickyManager.NoPrefID (-1) if it fails. GetorCreateEffectsPool (StickyEffectsModule effectsPrefab) is slightly faster. See also InstantiateEffects(igParms).
GetOrCreateSoundFXPool (StickyEffectsModule effectsPrefab)	Get the Sound FX object pool for this prefab or create a new pool if one does not already exist. Return the effectsPrefabID for the pool or StickyManager.NoPrefabID (-1) if it fails. See also InstantiateSoundFX(sfxParms).
GetOrCreateSoundFXPool (GameObject effectsPrefab)	Get the Sound FX pool for this prefab or create a new pool if one does not already exist. Return the effectsPrefabID for the pool or StickyManager.NoPrefID (-1) if it fails. GetorCreateSoundFXPool (StickyEffectsModule effectsPrefab) is slightly faster. See also InstantiateSoundFX(sfxParms).
InstantiateGenericObject (ref S3DInstantiateGenericObjectParameters igParms)	Instantiates the generic object with ID genericModulePrefabID at the position specified in world space.
InstantiateSoundFX (ref S3DInstantiateSoundFXParameters sfxParms, AudioClip audioClip)	Instantiate a pooled StickyEffectsModule which contains an audio source. If audioClip is null, the existing one (if there is one) attached to the prefab will be used. See also GetorCreateEffectsPool(..).
PauseEffectsObjects()	Pause all Pooled Effects Objects in the scene
ResumeEffectsObjects()	Resume all Pooled Effects Objects in the scene

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
GetGenericPrefab (int genericPrefabID)	Returns the prefab for a generic module given its generic prefab ID.
GetOrCreateGenericPool (GameObject 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 StickyManager.NoPrefID (-1) if it fails. GetorCreateGenericPool (StickyGenericModule genericObjectPrefab) is slightly faster. See also InstantiateGenericObject(igParms).
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 StickyManager.NoPrefID (-1) if it fails. See also InstantiateGenericObject(igParms).
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.
virtual uint Initialise (S3DInstantiateGenericObjectParameters igParms)	Initialise the module after it has been activated in the pool managed by StickyManager.
virtual void DestroyGenericObject()	Destroy (deactivate) the Generic Module. Returns it to the pool.

Method	Description
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().

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 int AddMessage (S3DPopupMessage popupMessage)	Add a message to the popup. Will return the index in the zero-based list, or -1 if it fails.
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 S3DPopupMessage GetPopupMessage (string messageName)	Get the first message with a matching name (is case sensitive). WARNING: This is likely to impact Garbage Collection. Where possible use GetPopupMessageByIndex(..)
virtual S3DPopupMessage GetPopupMessageByIndex (int messageIndex)	Get the message using the zero-based index in the list of messages.
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 void HideMessage (S3DPopupMessage popupMessage)	Attempt to hide the message on the popup.
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 SetCharacter (StickyControlModule stickyControlModule)	Set the character reference for this popup. This enables you to discover which character is showing the popup when an item is clicked. It informs the character this popup is active and parents itself to the character if IsReparented is enabled on the popup prefab. If isPauseWeaponsFiring is true, weapon firing will be paused on the character. See also SetCharacterIndirect(..).
virtual void SetCharacterIndirect (StickyControlModule indirectCharacter)	Set a reference to a character which indirectly showed this popup. For example, when a character engages with a StickySocket which then shows this popup. The popup may be parented to the Socket, but not the character. This enables you to discover which character is showing the popup. when an item is clicked. It informs the character this popup is active. If isPauseWeaponsFiring is true, weapon firing will be paused on the character. See also SetCharacter(..).
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 if Is Reparented is enabled on the popup prefab.

Method	Description
virtual void SetItem (int itemNumber, bool isShown, bool isEnabled = true)	Set the option or item in the popup. The itemNumber must be in the range of 1 to the NumberOfItems.
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 SetMessageLabelText (S3DPopupMessage popupMessage, string newLabelText)	Attempt to update the text of a message label.
virtual void SetMessageValueText (S3DPopupMessage popupMessage, string newValueText)	Attempt to update the text of a message value,
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.
virtual void ShowMessage (S3DPopupMessage popupMessage)	Attempt to Show the message in the popup.

Sticky Manager Properties

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

Property	Description
IsInitialised	[R] Is the StickyManager initialised?
IsBeamObjectsPaused	[R] Are all the beams in the scene currently paused? See also PauseBeams() and ResumeBeams().
IsGenericObjectsPaused	[R] Are all the generic objects in the scene currently paused? See also PauseGenericObjects() and ResumeGenericObjects().
BeamTemplatesList	[R] Get the list of the Beam Templates. To modify, use the public API Methods provided.
GenericObjectTemplatesList	[R] Get the list of the Generic Object Templates. To modify, use the public API Methods provided.

Sticky Shapes API Methods - General

These API methods work with a Sticky Shapes component. See also the “Sticky Shapes” chapter.

Method	Description
GetNonS3DAnimator()	Attempt to get the animator on a character that does not use the Sticky3D Controller component.
GetSkinnedMeshRenderers (List<SkinnedMeshRenderer> smRendererList)	Attempt to populate the list with skinned mesh renderers that contain blendshapes
GetStickyControlModule()	Get the StickyControlModule attached to this StickyShapesModule.
Initialise()	Initialise the Sticky Shapes Module. Has no effect if called multiple times.
PauseShapes()	Attempt to pause the module.
ReinitialiseEyes()	Attempt to reinitialise the eyes and validate them.
SetIsNonS3DController (bool isNotAttached)	Set if this component is attached to a non-Sticky3D Controller. i.e., it is using a 3rd party character controller.
SetNonS3DAnimator (Animator _animator)	Attempt to set the animator for this humanoid character. You might need to set this manually if the Animator component is not on the same gameobject as the StickyShapesModule.
UnpauseShapes()	Attempt to unpause the module.
ValidateEye (S3DEye eye, HumanBodyBones eyeBone)	Attempt to validate an eye bone on a humanoid character.

Sticky Shapes API Methods - Blendshapes

These API methods work with a Sticky Shapes component. See also the “Sticky Shapes” chapter.

Method	Description
GetBlendShape (int blendShapeId)	Attempt to get a blendshape from the current list
GetBlendShapeIdByNameHash (int blendShapeNameHash)	Attempt to get the BlendShapeId by supplying a hashed name of a blendshape.
GetBlendShapeId (string blendShapeName)	Attempt to find the unique BlendShapeId that matches the blendshape name.
GetBlendShapeNameHash (int blendShapeId)	Attempt to get the blendshape hashed name using the blendShapeId. If not, return 0.
GetBlendShapesFromModel (List<S3DBlendShape> shapeList)	Attempt to populate the list with blendshapes from the model.
GetBlendShapeNamesFromModel (List<string> nameList)	Attempt to populate the list with blendshape names from the model.
ValidateBlendShapes (out bool hasChanged)	Attempt to validate the list of blendshapes against the ones on the model. This includes checking that the hashed names match the blendshape names on the model. The hasChanged parm will be true if any of the S3DBlendShapes are changed as a result of the validation.

Sticky Shapes API Methods - Blink

These API methods work with a Sticky Shapes component. See also the “Sticky Shapes” chapter.

Method	Description
SetBlinkValue (float blinkValue)	Attempt to set the blink value, in the range 0.0 to 1.0 for all the blendshapes associated with this emotion.
StartBlinking()	Attempt to start blinking.
StopBlinking()	Attempt to stop blinking.
RefreshBlink()	Refresh validation state of blink emotions and blendshapes. Call after making any changes to the emotions or blendshapes associate with blinking.
ResyncBlinkShapes()	Attempt to resync all blink blendshapes with the blendshapes from those defined on the BlendShapes tab.

Sticky Shapes API Methods – Eye Movement

These API methods work with a Sticky Shapes component. See also the “Sticky Shapes” chapter.

Method	Description
StartEyeMovement()	Attempt to start eye movement.
StopEyeMovement()	Attempt to stop eye movement.

Sticky Shapes API Methods - Emotions

These API methods work with a Sticky Shapes component. See also the “Sticky Shapes” chapter.

Method	Description
AddEmotion (S3DEmotion emotion)	Attempt to add a new emotion to the list.
GetEmotion (int emotionId)	Get an emotion given the emotionId or the guidHash.
GetEmotion (S3DReactEmotion reactEmotion)	Get an emotion given a reactEmotion.
GetEmotionByName (string emotionName)	Attempt to get an emotion using the emotionName. Use sparingly as it could impact GC.
GetEmotionNames (List<string> nameList)	Attempt to populate the list with emotion names from the Emotions tab.
GetEmotionByNumber (int emotionNumber)	Attempt to get an emotion according the numeric order it appears in the list on the Emotion tab.
GetEmotionHoldDuration (S3DEmotion emotion, int reactionStageInt)	Get the appropriate hold duration for a Reaction stage. WARNING: For the sake of performance, assumes the emotion is not null.
GetEnabledEmotion (S3DReactEmotion reactEmotion, out S3DEmotion emotion)	Get an enable (in use) emotion, given a reactEmotion.
IsEmotionEnabled (S3DEmotion emotion)	Is the emotion enabled (in use)?
IsEmotionEnabled (S3DReactEmotion reactEmotion)	Is the emotion enabled (in use) for this reactEmotion?
PlayEmotion (int emotionNumber)	Play an emotion at the regular speed according to the numeric order it appears in the list on the Emotions tab.
PlayEmotion (S3DEmotion emotion)	Play an emotion at the regular speed.
RefreshVOXEmotions()	Refresh the list which contains the emotions randomly selected when speech audio is playing. Emotions are stored here as the EmotionNumber or 1-based index in the list of emotions on the Emotions tab.
ReinitialiseEmotions()	Reinitialise the list of emotions. Call this if manually adding to or deleting from the list of emotions.
ResyncEmotions()	Attempt to resync all emotion shapes with the blendshapes from those defined on the BlendShapes tab.
StartEmotion (S3DEmotion emotion)	Attempt to start an emotion
StartEmotion (string emotionName)	Attempt to start the first emotion with the given name. Use sparingly as it could impact GC. Where possible use StartEmotion(s3dEmotion). See also GetEmotionByName(emotionName).

Method	Description
StartEmotion (int emotionNumber)	Attempt to start an emotion in the numeric order it appears in the list on the Emotion tab.
StartEmotionInstantly (int emotionNumber)	Attempt to start an emotion instantly in the numeric order it appears in the list on the Emotion tab.
StartEmotionInstantly (S3DEmotion emotion)	Attempt to start an emotion instantly in the numeric order it appears in the list on the Emotion tab.
StopEmotion (S3DEmotion emotion)	Attempt to stop an emotion.
StopEmotion (string emotionName)	Attempt to stop the first emotion with the given name. Use sparingly as it could impact GC. Where possible use StopEmotion(s3dEmotion). See also GetEmotionByName(emotionName).
StopEmotion(int emotionNumber)	Attempt to stop an emotion in the numeric order it appears in the list on the Emotion tab.
StopEmotionInstantly (S3DEmotion emotion)	Attempt to stop an emotion instantly.
StopEmotionInstantly (int emotionNumber)	Attempt to stop an emotion instantly in the numeric order it appears in the list on the Emotion tab.

Sticky Shapes API Methods - Phonemes

These API methods work with a Sticky Shapes component. See also the “Sticky Shapes” chapter.

Method	Description
AddPhoneme (S3DPhoneme phoneme)	Add a phoneme to the list on the Phoneme tab.
AddSpeechAudio (S3DSpeechAudio speechAudio)	Add a speech audio to the list on the Phoneme tab.
DeleteSpeechAudioByNumber (int speechAudioNumber)	Attempt to delete a speech audio from the list on the Phoneme tab. Typically you should not try to remove speech audios while other speech audios may be playing on this character.
GetPhonemeAudioSource()	Get the current audioSource used to play the speech audioclips
GetPhonemeByName (string phonemeName)	Attempt to get a phoneme using the phonemeName. Use sparingly as it could impact GC.
GetPhonemeByNumber (int phonemeNumber)	Attempt to get a phoneme according to the numeric order it appears in the list on the Phonetics tab.
GetSpeechAudioByNumber (int speechAudioNumber)	Attempt to get a speechAudio according to the numeric order
PlayPhoneme (int phonemeNumber)	Play a phoneme at the regular speed according to the numeric order it appears in the list on the Phonetics tab.
PlayPhoneme (S3DPhoneme phoneme)	Play a phoneme at the regular speed.
PlaySpeechAudio (int speechAudioNumber)	Attempt to play a speech audio clip and the phoneme shapes in the numeric order it appears in the list on the Phonetics tab.
PlaySpeechAudio (S3DSpeechAudio speechAudio)	Attempt to play a speech audio clip and the phoneme shapes.
ReinitialisePhonemes()	Reinitialise the list of Phonemes.
ResyncPhonemes()	Attempt to resync all phoneme shapes with the blendshapes from those defined on the BlendShapes tab.
SetPhonemeAudioSource (AudioSource newAudioSource)	Set the audioSource used to play the speech audioclips.
StartPhoneme (S3DPhoneme phoneme)	Attempt to start a phoneme.

Method	Description
StartPhoneme (string phonemeName)	Attempt to start the first phoneme with the given name. Use sparingly as it could impact GC. Where possible use StartPhoneme(s3dPhoneme). See also GetPhonemeByName(emotionName).
StartPhoneme (int phonemeNumber)	Attempt to start a phoneme in the numeric order it appears in the list on the Phonetics tab.
StartPhonemeInstantly (int phonemeNumber)	Attempt to start a phoneme instantly in the numeric order it appears in the list on the Phonetics tab.
StartPhonemeInstantly (S3DPhoneme phoneme)	Attempt to start a phoneme instantly in the numeric order it appears in the list on the Phonetics tab.
StopPhoneme (S3DPhoneme phoneme)	Attempt to stop a phoneme.
StopPhoneme (string phonemeName)	Attempt to stop the first phoneme with the given name. Use sparingly as it could impact GC. Where possible use StopPhoneme(s3dPhoneme). See also GetPhonemeByName(phonemeName).
StopPhoneme (int phonemeNumber)	Attempt to stop a phoneme in the numeric order it appears in the list on the Phonetics tab.
StopPhonemeInstantly (S3DPhoneme phoneme)	Attempt to stop a phoneme instantly in the numeric order it appears in the list on the Phonetics tab.
StopPhonemeInstantly (int phoneNumber)	Attempt to stop a phoneme instantly in the numeric order it appears in the list on the Phonetics tab.
StopSpeechAudio (int speechAudioNumber)	Attempt to stop a speech audio clip from playing based on the numeric order it appears in the list on the Phonetics tab.
StopSpeechAudio (S3DSpeechAudio speechAudio)	Attempt to stop a speech audio from playing.

Sticky Shapes API Methods - Engage

These API methods work with a Sticky Shapes component. See also the "Sticky Shapes" chapter.

Method	Description
CharacterEnter (StickyControlModule nearbyCharacter)	A S3D character has been detected nearby entering the proximity area. Nearby characters must be initialised to be considered.
CharacterExit (StickyControlModule nearbyCharacter)	A nearby character has departed the proximity area. Characters must be initialised to be considered.
AddReaction (S3DReact react)	Attempt to add a new reaction to the list
S3DReact GetReaction (int guidHash)	Attempt to get a reaction using the unique guidHash or ID
GetReactionByName (string reactName)	Attempt to get a reaction using the reactName. Use sparingly as it could impact GC.
GetReactionByNumber (int reactNumber)	Attempt to get a reaction according the numeric order it appears in the list on the Engage tab under Reactions.
PlayEnterReaction (S3DReact react)	Attempt to play an enter reaction.
PlayEnterReaction (string reactName)	Attempt to play the first enter reaction with the given name. Use sparingly as it could impact GC. Where possible use PlayEnterReaction(s3dReact). See also GetReactionByName(reactName).
PlayEnterReaction (int reactNumber)	Attempt to play an enter reaction (in the numeric order it appears in the list of enter reactions on the Engage tab) when another comes nearby.
PlayExitReaction (S3DReact react)	Attempt to play an exit reaction.

Method	Description
PlayExitReaction (string reactName)	Attempt to play the first exit reaction with the given name. Use sparingly as it could impact GC. Where possible use PlayExitReaction(s3dReact). See also GetReactionByName(reactName).
PlayExitReaction (int reactNumber)	Attempt to play an enter reaction (in the numeric order it appears in the list of enter reactions on the Engage tab) when another departs from nearby.
PlayReaction (S3DReact reaction, S3DReactEmotion reactEmotion)	Play an emotional reaction.
PlayReaction (S3DReact reaction, S3DReactSpeechAudio reactSpeechAudio)	Play a speech audio reaction.
ReinitialiseReactions()	Prepare the character to start reacting to others in the scene. See also StartReacting() and StopReacting(..).
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.
ResyncReactions()	Attempt to resync all react emotions with the emotions from those defined on the Emotions tab.
SetReactDistance (float newDistance)	Set the distance the character starts to react to others in the scene.
StartReacting()	Start reacting to others in the scene.
StopReacting (bool isInstant)	Stop reacting to others in the scene.
StopEnterReaction (S3DReact react)	Attempt to stop a reaction that was triggered when another came nearby.
StopEnterReaction (string reactName)	Attempt to stop the first reaction with the given name that was triggered when another came nearby. Use sparingly as it could impact GC. Where possible use StopEnterReaction(s3dReact). See also GetReactionByName(reactName).
StopEnterReaction (int reactNumber)	Attempt to stop a reaction (in the numeric order it appears in the list of reactions on the Engage tab) that was triggered when another came nearby.
StopEnterReactionInstantly (S3DReact react)	Attempt to stop a reaction instantly that was triggered when another came nearby.
StopEnterReactionInstantly (int reactNumber)	Attempt to stop a reaction instantly (in the numeric order it appears in the list of reactions on the Engage tab) that was triggered by another coming nearby.
StopExitReaction (S3DReact react)	Attempt to stop a reaction that was triggered when another departed from nearby.
StopExitReaction (string reactName)	Attempt to stop the first reaction with the given name that was triggered when another departed from nearby. Use sparingly as it could impact GC. Where possible use StopExitReaction(s3dReact). See also GetReactionByName(reactName).
StopExitReaction (int reactNumber)	Attempt to stop a reaction (in the numeric order it appears in the list of reactions on the Engage tab) that was triggered when another departs from nearby.
StopExitReactionInstantly (S3DReact react)	Attempt to stop a reaction instantly that was triggered when another departs from nearby.
StopExitReactionInstantly (int reactNumber)	Attempt to stop a reaction instantly (in the numeric order it appears in the list of reactions on the Engage tab) that was triggered by another departs from nearby.

Sticky Shapes Properties - General

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

Property	Description
GetLeftEye	[R] Get the left eye.
GetRightEye	[R] Get the right eye.
GetSkinnedMeshRendererList	[R] Get the current list of skinned mesh renderers that contain blendshapes.
IsShapesModuleInitialised	[R] Is the StickyShapeModule initialised?
IsNonS3DController	Get or set if the component is attached to a non-Sticky3D Controller. i.e., it is using a 3rd party character controller.

Sticky Shapes Properties – Blendshapes

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

Property	Description
GetBlendShapeList	[R] Is the StickyShapeModule initialised?
NumValidBlendShapes	[R] Get the last number of blendshapes deemed valid.

Sticky Shapes Properties - Emotions

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

Property	Description
GetBlink	[R] Get the blink emotion
GetEmotionList	[R] Get the current list of emotions.
IsBlinkingEnabled	[R] Is the blinking action currently enabled?

Sticky Shapes Properties - Phonemes

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

Property	Description
IsEyeMovementEnabled	[R] Is eye movement currently enabled?
NumberOfEmotions	[R] Get the number of emotions on the Emotions tab.

Sticky Shapes Properties - Engage

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

Property	Description
GetReactList	[R] Get the current list of reactions.
IsReactingEnabled	[R] Is the character ready to react to others with emotions or speech?
IsReactInitialised	[R] Is the character ready to react?
ProximityCollider	Get or set the proximity collider. It must be on a child gameobject within the prefab.
ReactDistance	Get or set the distance the character reacts to others nearby. Assumes the collider is a sphere or box collider.

Sticky Socket API Methods

These API methods work with a Sticky Socket component. See also the “Sticky Socket” chapter.

Method	Description
GetActivePopupID()	Return the Instance ID of the active StickyPopupModule. If there isn't one, return 0. See also SetActivePopupID(..).
GetDefaultPopupPosition()	Get the world space position for the default popup module.
GetEmptyPopupPosition()	Get the world space position for the popup module when the socket is empty.

Method	Description
GetLastItem()	Get the S3DStoreItem of the last item that was added. Ignores items that have already been removed.
GetLastItemID ()	Get the StoreItemID of the last item that was added. Ignores items that have already been removed.
GetLastInteractiveID()	Get the StickyInteractiveID of the last item that was added. Ignores items that have already been removed.
GetItem (int storeItemID)	Get the store item given a S3DStoreItem.StoreItemId.
GetItemByInteractiveID (int stickyInteractiveID)	Get the store item given a stickyInteractiveID.
GetItemID (int stickyInteractiveID)	Get the StoreItemID of an interactive-enabled object using the StickyInteractiveID. Returns the StoreItemID or S3DStoreItem.NoStoreItem.
GetItemID (StickyInteractive stashItem)	Get the StoreItemID of an interactive-enabled object using a StickyInteractive item. Returns the StoreItemID or S3DStoreItem.NoStoreItem.
GetPermittedTags()	Get the 32-bit mask use to detemine which interactive-enabled objects are permitted to be attached to the socket. See the S3DInteractiveTags scriptableobject.
RemoveInteractive (int stickyInteractiveID, bool isRestoreColliders, bool isRestoreGravity)	Attempt to remove the StickyInteractive object from the socket.
SetDefaultPopupPrefab (StickyPopupModule stickyPopupModulePrefab)	Set default StickyPopupModule to display when a character engages with this socket.
SetEmptyPopupPrefab (StickyPopupModule stickyPopupModulePrefab)	Set StickyPopupModule to display when a character enagages with this socket when empty or has no iteractive objects attached.
SetPermittedTags (int bitMask)	The 32-bit mask use to detemine which interactive-enabled objects are permitted to be attached to the socket. Defaults to ~0 (Everything). See the S3DInteractiveTags scriptableobject.
SetInteractiveTags (S3DInteractiveTags newInteractiveTags)	The common interactive tags scriptableobject used to determine which interactive objects can be added to this socket.
SetPopupCamera (Camera camera)	Sets which camera, if any the current popup will face.
ShowPopup()	Show the default or empty socket popup.
ShowPopupFromCharacter (StickyControlModule indirectCharacter)	Show the default or empty socket popup, indirectly triggered by a (player) character. Typically used when a character is looking at a socket, and brings up a popup menu. The popup will be parented to the socket but the popup will know the character indirectly triggered it.

Virtual methods can be overridden to customise the behaviour of the Sticky Socket 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 int AddItem (StickyInteractive itemToAdd)	Attempt to add an interactive-enabled object to the socket. Returns the storeItemID for the item attached to the socket. If it is not added, S3DStoreItem.NoStoreItem is returned. If required, you can override this method. public override int AddItem (StickyInteractive itemToAdd) {

Virtual Method	Description

	Int _storeItemID = base. AddItem (itemToAdd);
	// Do stuff here
	
	return _storeItemID:
	}

virtual bool CanAddItem (StickyInteractive itemToCheck)	Check to see if the interactive-enabled object is eligible to be added to the socket. NOTE: To add the item, see AddItem(..).
virtual void DestroyItem (S3DStoreItem storeItem)	Destroy the StoreItem.
virtual void DestroyItem (int storeItemID)	Destroy the store item given a S3DStoreItem.StoreItemId.
virtual void DestroySocket()	Safely destroy the StickySocket. 1. Return active popup (if any) to the pool 2. Destroy the actual gameobject

virtual void Initialise()	Initialise the StickySocket 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 RemoveItem (int storeItemID)	Remove an interactive-enabled object from the StickySocket given the StoreItem ID.

Sticky Socket API Call Backs

These (delegates) are used when you wish to call a custom method in your own game code. For example, when the character clicks on an item in a popup. Be careful not to hold references to objects passed as parameters outside your method otherwise Unity's garbage collector may not be able to release some things from memory.

Method field	Description
CallbackOnPopupItemClicked callbackOnPopupClicked	The name of your custom method that is called immediately after a popup item is clicked and the popup is closed.

Sticky Socket Properties

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

Property	Description
DefaultPopupPrefabID	[R] The prefabID for the pooled generic module assigned by StickyManager.
EmptyPopupPrefabID	[R] The prefabID for the pooled generic module assigned by StickyManager when the socket is empty or has no interactive objects attached.
IsInitialised	[R] Has the module been initialised?
NumberOfSocketedItems	[R] The number of interactive objects attached to this socket.
SocketID	[R] The identifier for the StickySocket. Only available after it has been initialised.

Sticky Weapon (General) API Methods

These API methods work with a Sticky Weapon component. See also the “Sticky Weapon” chapter.

Sticky Weapons inherit methods from the “Sticky Interactive” component, so also check the “Sticky Interactive Methods” section earlier in this chapter.

Method	Description
DeactivateBeams ()	Deactivate all beam weapons that are currently firing.
DestroyWeapon()	Safely destroy this weapon. 1) Deactivate any beams 2) Return muzzle fx to the pool 3) Return any reload sound fx to the pool 4) Drop weapon if held by character 5) Return any popup to the pool (in DestroyInteractive) 6) Destroy the gameobject (in DestroyInteractive)
DisableWeapon()	Attempt to disable the weapon.
	Drop the weapon 1) Stop firing 2) If equipped, turn off laser sight 3) Re-enable colliders that where disabled when grabbed 4) Configure gravity if Use Gravity is enabled
EnableWeapon()	Attempt to enable the weapon.
GetWorldFireBasePosition()	Get the world space fire position without any offsets.
GetWorldFireDirection()	Get the world space fire direction of the weapon.
GetWorldFirePosition (Vector3 weaponWorldBasePosition, int firePositionOffsetIndex)	Get the world space fire position of the fire position offset (if any) If there are no firePostionOffsets, the firePositionOffsetIndex should be set to 0. See also UpdateWeapon() which fires the weapon.
GetWorldSpentEjectDirection()	Get the world space direction the spent cartridge is ejected.
GetWorldSpentEjectPosition()	Get the world space spent cartridge eject position.
GetWorldSpaceEjectRotation()	Get the world space spent cartridge eject rotation.
Initialise()	Initialise the weapon.
ReInitialiseWeapon()	Call this after making any changes to beams, projectiles or effects for this weapon.
PauseWeapon()	Pause the weapon. This is useful when pausing a game or scene. See also: DisableWeapon(), UnPauseWeapon().
ResetInput()	Reset the weapon input to default settings.
SendInput (WeaponInput weaponInput)	Send input data to the weapon.
SetChargeAmount (float newChargeAmount)	Set the chargeAmount. It is automatically clamped between 0.0 and 1.0.
SetCheckLineOfSight (int weaponButtonNumber, bool newCheckLineOfSight)	Set if line of sight should be checked for an AutoFire mechanism.
SetCheckLineOfSight1 (bool newCheckLineOfSight)	Set if line of sight should be checked for an AutoFire primary mechanism.
SetCheckLineOfSight2 (bool newCheckLineOfSight)	Set if line of sight should be checked for an AutoFire secondary mechanism.
SetFireInterval (float newFireInterval)	Set the fireInterval. It is automatically clamped between 0.05 and 10 seconds.
SetFireDirection (Vector3 newFireDirection)	Set the local space fire direction of the weapon.
SetFiringButton1 (FiringButton newFiringButton)	Set the FiringButton of the first (main) firing button.
SetFiringButton2 (FiringButton newFiringButton)	Set the FiringButton of the second firing button. This is an alternative firing mechanism for custom weapons.

Method	Description
SetFiringType1 (FiringType newFiringType)	Set the FiringType of the first (main) firing button.
SetFiringType2 (FiringType newFiringType)	Set the FiringType of the second firing button.
SetFirePositionOffsets (Vector3[] newFirePositions)	Set the weapon firePositionOffsets.
SetMaxRange (float newMaxRange)	Set the maximum range for this weapon. The minimum range is 1 metre.
SetMuzzleEffects1 (StickyEffectsModule[] newMuzzleEffects)	Set the array of StickyEffectsModules used for Muzzle FX.
SetOnlyUseWhenHeld (bool newValue)	Set if weapon can only fire, reload, or be animated, when it is held by a character.
SetRechargeTime (float newRechargeTime)	Set the rechargeTime. It is automatically clamped between 0 and 30 seconds.
SetSpentEjectDirection (Vector3 newDirection)	Set the local space direction the spent cartridge is ejected.
SetSpentEjectForce (float newForce)	Set the force used to eject the spent cartridge from the weapon in the Spent Eject Direction.
SetSpentEjectPosition (Vector3 newPosition)	Set the local space position on the weapon from which the spent cartridge is ejected.
SetSpentEjectRotation (Vector3 newRotation)	Set the local space rotation, in Euler angles, of the spent cartridge when ejected.
UnpauseWeapon()	Unpause the weapon. This is useful when unpausing a game or scene. See also: EnableWeapon(), PauseWeapon()
WeaponHasLineOfSight (int weaponButtonNumber)	Returns whether a weapon has line of sight to a target (if there is one assigned). NOTE: Currently both primary (1) and secondary (2) firing mechanisms return the same result. As we find use-cases for the secondary firing button, this may change.

Sticky Weapon (Aiming and Reticle) API Methods

These API methods work with a Sticky Weapon component. See also the “Sticky Weapon” chapter.

Method	Description
CheckReticle()	Check to see if a player (non-NPC) is holding the weapon and a different reticle is required on the active display module (if there is one).
RestoreReticle()	If required, attempt to reset the DisplayReticle to the original one used on the active StickyDisplayModule (HUD), before the weapon was grabbed by a non-NPC character.
SaveReticleSettings()	Save the StickyDisplayModule reticle settings when the weapon is picked up by a character.
SetAimingFirstPersonFOV (float newFoV)	Set the character first person camera field of view when aiming a held weapon.
SetAimingReticleSprite (Sprite newSprite)	Set the reticle (sprite) used when the weapon is held by a S3D player and aiming.
SetAimingThirdPersonFOV (float newFoV)	Set the character third person camera field of view when aiming a held weapon.
SetDefaultReticleSprite (Sprite newSprite)	Set the reticle (sprite) used when the weapon is held by a S3D player.
StartAiming (bool isSmoothStart)	Attempt to start aiming the weapon if held by a character.

Method	Description
StopAiming (bool isSmoothStop)	Stop aiming the weapon if held by a character.
ToggleAiming()	Attempt to toggle aiming the weapon if held by a character. If you want an instant transition, call StartAiming(false) or StopAiming(false).

Sticky Weapon (Attachments) API Methods

These API methods work with a Sticky Weapon component. See also the “Sticky Weapon” chapter.

Method	Description
EnableLaserSightIfRequired()	Check to see if we need to automatically turn on the laser sight. This is called automatically when a character grabs the weapon.
EnableScopeIfRequired()	Check to see if we need to automatically turn on the Scope. This is called automatically when a character grabs the weapon.
EquipLaserSight()	Attempt to equip the laser sight on the weapon and get it ready to turn on/off. It is turned off when it is first equipped. See also SetLaserSightAimFrom(newLaserSightTransform).
EquipMagazine (StickyMagazine stickyMagazine, int weaponButtonNumber)	Attempt to equip a magazine on the weapon.
EquipMagazine1 (StickyMagazine stickyMagazine)	Attempt to equip a magazine onto the weapon for the primary firing mechanism.
EquipMagazine2 (StickyMagazine stickyMagazine)	Attempt to equip a magazine onto the weapon for the secondary firing mechanism.
SetLaserSightAimFrom(Transform newLaserSightTransform)	Set or change the child transform that determines where the laser sight aims from.
SetLaserSightColour (Color32 newColour)	Set the laser sight beam colour.
SetMagazineAttachPoint(Transform newMagazineAttachPoint, int weaponButtonNumber)	Set or change the child transform where the magazine attaches to the weapon.
SetScopeCamera (Camera newCamera)	Set the camera that will render the Scope display.
SetScopeCameraRenderer (Renderer newRenderer)	Set the (mesh) renderer to project the Scope Camera onto.
ToggleLaserSight()	Attempt to toggle the laser sight on/off.
ToggleScope()	Attempt to toggle the Scope for more precise aiming on/off.
TurnOnLaserSight()	Attempt to turn on the laser sight.
TurnOffLaserSight()	Turn off the laser sight.
TurnOffScope()	Turn off the Scope.
TurnOnScope()	Attempt to turn on the scope for aiming.
UnEquipLaserSight()	This will unequip the laser sight from the weapon, including removing the line renderer (if any)
UnEquipMagazine (int weaponButtonNumber)	Attempt to unequip a sticky magazine from the weapon.
UnequipMagazine1()	This will attempt to unequip a magazine from the weapon attach point 1
UnequipMagazine2()	This will attempt to unequip a magazine from the weapon attach point 2

Sticky Weapon (Ammo) API Methods

These API methods work with a Sticky Weapon component. See also the “Sticky Weapon” chapter.

Method	Description
CheckMagsInReserve()	This is automatically called when the weapon is initialised, grabbed or dropped.
GetIsMagRequired (int weaponButtonNumber)	Get if the primary (1) or secondary (2) firing mechanism requires a StickyMagazine to be attached before it can fire? See also IsMagRequired1, IsMagRequired2, and SetIsMagRequired(..).
GetMagCapacity (int weaponButtonNumber)	Get the current magazine (clip) capacity for the primary (1) or secondary (2) firing mechanism.
GetCompatibleMagType (int weaponButtonNumber)	Get the current magazine (clip) zero-based index from the magTypes scriptable object for the primary (1) or secondary (2) firing mechanism.
GetMagsInReserve (int weaponButtonNumber)	Get the number of additional magazines available for the primary (1) or secondary (2) firing mechanism. See also MagsInReserve1, MagsInReserve2, and SetMagsInReserve(..).Returns number in reserve or -1 (unlimited).
GetMagsHeld (int weaponButtonNumber)	Get the number of additional magazines available for the primary (1) or secondary (2) firing mechanisms that are held in the hands of the character that has this weapon. If the weapon is held in one hand, currently the maximum number would be 1.
GetMagsInStash (int weaponButtonNumber)	Get the number of additional magazines available for the primary (1) or secondary (2) firing mechanism from the Stash (inventory) of the character that has this weapon.
GetReloadEquipSoundFX (int weaponButtonNumber)	Get the Effects Module for the primary (1) or secondary (2) Sound FX when equipping the new mag begins during reloading.
GetReloadSoundFX (int weaponButtonNumber)	Get the Effects Module for the primary (1) or secondary (2) reloading Sound FX.
GetReloadType (int weaponButtonNumber)	Get The method used for reloading the primary (1) or secondary (2) firing mechanism.
GetReloadTypeInt (int weaponButtonNumber)	Get The method used for reloading the primary (1) or secondary (2) firing mechanism as an Integer.
IsAmmoCompatible (int weaponButtonNumber, int ammoType)	Check to see if the ammo type is compatible with a firing mechanism.
IsAmmoCompatible (int weaponButtonNumber, S3DAmmo.AmmoType ammoType)	Check to see if the ammo type is compatible with a firing mechanism.
IsMagCompatible (int weaponButtonNumber, StickyMagazine stickyMagazine)	Check to see if a magazine is compatible with a firing mechanism.
IsReloading (int weaponButtonNumber)	Is the given firing mechanism reloading?
ReinitialiseEmptyFire (int weaponButtonNumber)	Reinitialise items related to Empty Fire effects. Call this if making any changes to fireEmptyEffectsSet1 or 2.
ReinitialiseEquippedMag (int weaponButtonNumber)	Find and identify a compatible magazine attached to the weapon under the Mag Attach Point for the primary or secondary firing mechanism. This is automatically called when the weapon is initialised so that a magazine can be pre-attached to a weapon in the scene.
Reload (int weaponButtonNumber)	Attempt to reload the weapon for the primary (1) or secondary (2) firing button mechanism.
SetAmmunition (int weaponButtonNumber, int newAmmunitionValue)	Get or set the ammunition currently loaded into the weapon for the primary (1) or secondary (2) firing mechanism. When there isn't a magazine required, ensure the ammunition doesn't exceed the capacity of the (fake) magazine. -1 = Unlimited.

Method	Description
SetCompatibleMagType (int weaponButtonNumber, int newCompatibleMagType)	Set the zero-based index of the magazine type from the scriptable object for the primary (1) or secondary (2) firing button.
SetIsMagRequired (int weaponButtonNumber, bool newIsMagRequired)	Set if the primary (1) or secondary (2) firing mechanism requires a StickyMagazine to be attached before it can fire? See also IsMagRequired1, IsMagRequired2, and GetIsMagRequired(..).
SetMagCapacity (int weaponButtonNumber, int newMagCapacity)	Set the amount of ammo the magazine (clip) can hold for the primary (1) or secondary (2) firing mechanism. This has no effect if Is Mag Required is true for the firing mechanism.
SetMagsInReserve (int weaponButtonNumber, int newMagsInReserveValue)	Get or set the number of additional magazines available for the primary (1) or secondary (2) firing mechanism. -1 = Unlimited.
SetReloadEquipSoundFX (int weaponButtonNumber, StickyEffectsModule newReloadEquipSoundFX)	Set the Sound FX for the primary (1) or secondary (2) fire button mechanism used when the weapon equips a new mag during reloading. The EffectsType must be Sound FX.
SetReloadSoundFX (int weaponButtonNumber, StickyEffectsModule newReloadSoundFX)	Set the reload SoundFX Effects Module for the primary (1) or secondary (2) fire button mechanism. The EffectsType must be Sound FX.
SetReloadType (int weaponButtonNumber, ReloadType newReloadType)	Set the method used for reloading the primary (1) or secondary (2) firing mechanism.
ValidateAmmo()	Validate the ammo types for this weapon.

Sticky Weapon (Animate) API Methods

These API methods work with a Sticky Weapon component. See also the “Sticky Weapon” chapter.

Method	Description
ApplyWeaponAnimSets ()	Apply any matching weapon anim sets for the character holding the weapon.
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 weapon animation controller.
EnableAnimate()	Enable the sending of animation data to the weapon animation controller. This will fall back to Disabled if the current setup does not support it.
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.
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

Method	Description
	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.
RevertWeaponAnimSets()	Revert any matching weapon anim sets for the character holding the weapon back to original settings.
SetAnimActionsList (List<S3DAnimAction> newAnimActionsList)	Set the list of animation actions for this weapon.
SetAnimLayerBlendInDuration (int layerIndex, float blendInDuration)	Set the duration, in seconds, it takes for an animator layer weight to reach 1.0. 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 and the weaponAction is a custom action. 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 and the weaponAction is a custom action.
SetCustomAnimActionFloatValue (S3DAnimAction s3DAnimAction, float value)	Given a custom S3DAnimAction set its float value, assuming it has a matching ParamaterType and the weaponAction is a custom action.
SetCustomAnimActionIntegerValue (int guidHash, int value)	Given the guidHash of a custom S3DAnimAction set its integer value, assuming it has a matching ParamaterType and the weaponAction is a custom action.
SetCustomAnimActionIntegerValue (S3DAnimAction s3DAnimAction, int value)	Given a custom S3DAnimAction set its integer value, assuming it has a matching ParamaterType and the weaponAction is a custom action.
SetCustomAnimActionTriggerValue (int guidHash, bool value)	Given the guidHash of a custom S3DAnimAction set its trigger (bool) value, assuming it has a matching ParamaterType and the weaponAction is a custom action.
SetCustomAnimActionTriggerValue (S3DAnimAction s3DAnimAction, bool value)	Given a custom S3DAnimAction set its trigger (bool) value, assuming it has a matching ParamaterType and the weaponAction is a custom action.
SetDefaultAnimator (Animator newAnimator)	Set the weapon animator. This should be attached to, or a child of, the weapon gameobject.
SetWeaponAnimSetList (List<S3DWeaponAnimSet> newWeaponAnimSetList)	Set the list of weapon anim sets for this weapon.
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 found, else returns the parameter hashcode. WARNING: This impacts GC and should never be called in an update loop. Use sparingly.

Sticky Weapon (Health) API Methods

These API methods work with a Sticky Weapon component. See also the “Sticky Weapon” chapter.

Method	Description
GetHeatLevel ()	Get the heat level of the weapon. 0.0 is no heat, 100 is overheated.
SetHeatLevel (float newHeatLevel)	Set the new heat level on this weapon. Range 0.0 (min) to 100.0 (max).
SetIsFiringPaused (bool newValue)	Set if the weapon firing mechanisms are paused. This will prevent the weapon from firing, but it can still do other operations like reloading etc.

Sticky Weapon (Smoke) API Methods

These API methods work with a Sticky Weapon component. See also the “Sticky Weapon” chapter.

Method	Description
ReinitialiseSmoke (int weaponButtonNumber)	Reinitialise items related to smoke effects. Call this if making any changes to smokeEffectsSet1 or 2.
SetMaxActiveSmokeEffects (int newMax)	Set the maximum number of smoke effects that can be active at any time on this weapon.

Sticky Weapon (Targeting) API Methods

These API methods work with a Sticky Weapon component. See also the “Sticky Weapon” chapter.

Method	Description
AssignTargetCharacter (StickyControlModule newTarget)	Assign a character that the weapon should attempt to target. This is typically used when an autofire weapon, held by an NPC, is determining if the weapon is aiming or pointing toward a Sticky3D character. It is NOT used to change where the weapon is aiming.
AssignTargetTransform (Transform newTarget)	Assign a transform that the weapon should attempt to target. This is typically used when an autofire weapon, held by an NPC, is determining if the weapon is aiming or pointing toward an object. It is NOT used to change where the weapon is aiming.
ResetLineOfSight (int weaponButtonNumber)	Weapon no longer has line of sight to the target.
UnAssignTargetCharacter()	Stop targeting a Sticky3D character.
UnAssignTargetTransform()	Stop targeting a transform.

Sticky Weapon Properties

These properties work with a Sticky Weapon component. Read Only properties are marked in the following table with [R]. See also the “Sticky Weapon” chapter.

Sticky Weapons inherit properties from the “Sticky Interactive” component, so also check the “Sticky Interactive Properties” section earlier in this chapter.

Property	Description
AimAtPosition	[R] Get where the weapon is currently aiming.
ChargeAmount	Get or set the weapon ChargeAmount.
CheckLineOfSight1	Get or set if the line of sight is checked before firing primary mechanism. Only relevant if it is AutoFire.
CheckLineOfSight2	Get or set if the line of sight is checked before firing primary mechanism. Only relevant if it is AutoFire.

Property	Description
EstimatedRange	[R] The estimated range (in metres) of the weapon.
FireDirection	Get or set the local space normalised fire direction of the weapon.
FiringButton1	Get or set the main trigger for the weapon to fire e.g. None, Manual, AutoFire.
FiringButton2	Get or set the second trigger for the weapon to fire e.g. None, Manual, AutoFire.
FireInterval	Get or set the weapon power up time.
FiringType1	Get or set the main firing type for the weapon. e.g., SemiAuto or FullAuto.
FiringType2	Get or set the secondary firing type for the weapon. e.g., SemiAuto or FullAuto.
HasEmptyFired1	[R] Has the weapon primary mechanism attempted to fire with no ammo since the last fixed update?
HasEmptyFired2	[R] Has the weapon secondary mechanism attempted to fire with no ammo since the last fixed update?
HasFired1	[R] Has the weapon fired the primary mechanism since the last fixed update?
HasFired2	[R] Has the weapon fired the secondary mechanism since the last fixed update?
HasReloadStarted1	[R] Has the weapon started reloading the primary mechanism since the last fixed update?
HasReloadStarted2	[R] Has the weapon started reloading the secondary mechanism since the last fixed update?
HasReloadFinished1	[R] Has the weapon finished reloading the primary mechanism since the last fixed update?
HasReloadFinished2	[R] Has the weapon finished reloading the finished mechanism since the last fixed update?
HasLineOfSight1	[R] Returns whether the weapon primary mechanism has line-of-sight to the target. NOTE: This does not calculate line-of-sight, it merely returns the last calculated value. To calculate line-of-sight for a weapon, call WeaponHasLineOfSight(1).
HasLineOfSight2	[R] Returns whether the weapon primary mechanism has line-of-sight to the target. NOTE: This does not calculate line-of-sight, it merely returns the last calculated value. To calculate line-of-sight for a weapon, call WeaponHasLineOfSight(2).
HitLayerMask	Get or set the hit layer mask. When fired, the weapon can hit any objects in these Unity Layers.
MaxRange	Get or set the weapon maximum range.
IsBeamWeapon	[R] Is the weapon one that can fire beams, rays or lasers?
IsFire1Input	[R] Is the weapon being instructed to fire button 1?
IsFire1InputHeld	[R] Is the fire1 button being held down?
IsFire2Input	[R] Is the weapon being instructed to fire button 2?
IsFire2InputHeld	[R] Is the fire2 button being held down?
IsOnlyFireWhenAiming1	Get or set if Fire button 1 can only fire when the weapon is being aimed.
IsOnlyFireWhenAiming2	Get or set if Fire button 2 can only fire when the weapon is being aimed.
IsOnlyUseWhenHeld	Get or set if weapon can only fire, reload, or be animated, when it is held by a character.
IsNoReticleOnAimingFPS	Get or set if not to display a reticle on the active StickyDisplayModule when the weapon is held by a player while aiming with first-person camera.
IsNoReticleOnAimingTPS	Get or set if not to display a reticle on the active StickyDisplayModule when the weapon is held by a player while aiming with third-person camera.
IsProjectileRaycastWeapon	[R] Is the weapon one that can fire a Raycast projectile?
IsProjectileStandardWeapon	[R] Is the weapon one that can fire a Standard projectile?
IsProjectileWeapon	[R] Is the weapon one that can fire a (Standard or Raycast) projectile?
IsStickyWeapon	[R] Is this an interactive-enabled StickyWeapon?
IsWeaponEnabled	Get or set if the weapon is enabled.
IsWeaponInitialised	[R] Has the weapon been initialised?
IsWeaponPaused	Get or set if the weapon is paused.

Property	Description
MuzzleEffects1	Get or set the array of StickyEffectsModules used for Muzzle FX.
NumberFirePositionOffsets	[R] Get the number of local space fire position offsets from the relative fire position. There may be 0 or more. See also RelativeFirePosition.
NumberMuzzleEffects1	[R] Get the number of (unvalidated) StickyEffectsModules for the muzzle FX.
NumberMuzzleEffects1Valid	[R] Get the number of StickyEffectsModules available for the muzzle FX.
RechargeTime	Get or set the weapon recharge time.
RelativeFirePosition	Get or set the local space relative fire position which should be aligned to the end of the barrel.
SpentEjectDirection	Get or set the local space direction the spent cartridge is ejected.
SpentEjectForce	Get or set the force used to eject the spent cartridge from the weapon in the Spent Eject Direction.
SpentEjectPosition	Get or set the local space position on the weapon from which the spent cartridge is ejected.
SpentEjectRotation	Get or set the local space rotation, in Euler angles, of the spent cartridge when ejected.

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

Method	Description
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.
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.

Method	Description
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://discussions.unity.com/t/814533
Discord Channel	https://discord.gg/ZqEeBF584r
Email	support@scsmmedia.com

Version History

First Alpha version November 2020

First Beta version February 2021

Initial 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

Version 1.0.9 – 30 March 2022

[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 1.1.0 – 20 July 2023

[NEW] Weapon System - in Technical Preview
[NEW] Stashable interactive objects - APIs for building custom inventory system
[NEW] Aim Inverse Kinematics
[NEW] Poolable Decal and Effects modules
[NEW] Poolable Beam and Projectile modules
[NEW] Interactive-enabled Magazine and Weapon modules
[NEW] Poolable Dynamic Objects
[NEW] StickyInteractive - Sittable object can be allocated (reserved)
[NEW] StickyInteractive - Post Grabbed event
[NEW] StickyInteractive - Post Initialised event
[NEW] StickyInteractive - tagging for Socket and character Equip Point compatibility
[NEW] SampleEquipGrabStashPopup - shows pickup options using a popup
[NEW] SampleSitActionNPC - shows how an NPC character can sit on interactive objects
[NEW] StickyControlModule - Engage onInitialised event
[NEW] StickyControlModule - Engage interactive look change event
[NEW] StickyControlModule - Engage destroy and respawn events
[NEW] StickyControlModule - Engage Respawning options
[NEW] StickyControlModule - Engage Equip points for interactive objects
[NEW] StickyControlModule - Look Interactive - Lock to Camera
[NEW] StickyControlModule - Look zoom out factor
[NEW] StickyControlModule - Collide reference frames can use parent rigidbodies
[NEW] StickyDisplayModule - ToggleDisplayReticle() API method
[NEW] Animate Action Conditions for held weapons and magazines
[NEW] StickyInputModule - crouch has an optional toggle mode
[NEW] StickyPopupModule - SetCharacter API to associate popups with a character
[NEW] StickyPopupModule - Message labels and text

[NEW] StickySocket - For attaching interactive objects to non-character positions
[NEW] StickyGraphicRaycaster - for use with world space UI (canvas) in VR
[NEW] Sample NPC Fire Weapon script
[NEW] Sample Weapon Custom Reload script.
[NEW] SSC Input Bridge component for sending data to Sci-Fi Ship Controller
[FIXED] First Person Follow Head Bone may be incorrect if starting in 3rd person before sitting
[FIXED] Typo in ResetReferenceFrameLayerMask() API
[FIXED] StickyDisplayModule - Unticking Lock Reticle to Cursor at runtime does not reset to centre
[FIXED] Set correct mass of demo Rod characters
[FIXED] Follow Head Position in first person is not correct if enabled before scene starts
[FIXED] Look Interactive may be disabled when calling EnableCharacter()
[FIXED] StickyInteractive - IsAutoDeactivate
[FIXED] s3d_briefcase1 model offset
[FIXED] Invalid AABB errors with Unity 2022.1 and Trigger Collider enabled on Collide tab
[FIXED] Jet Pack thrusters may not be stopped when feature is disabled during gameplay
[FIXED] StickyInteractorBridge not discovering Quest 2 VR hand controllers on start in build
[FIXED] SampleMoveTo - CancelMoveTo() does not stop a NPC from moving
[FIXED] Switch Look and toggle Jet Pack may not always work on low power devices
[FIXED] Default font in Unity 2022.2+
[FIXED] Demo button1 UVs
[IMPROVED] StopMoving API now also resets input damping
[IMPROVED] IsObstacle API checks for valid direction
[IMPROVED] Added Sticky Anim Replacer documentation to the manual
[IMPROVED] Reduce in-editor GC when looking at interactive-enabled objects
[IMPROVED] Replaced Get/SetActivePopup() APIs with Get/Clear/SetActivePopups(), IsActivePopups()
[IMPROVED] Sticky Control Module - Moved Identification Settings to the Engage tab
[IMPROVED] Sticky Control Module - Condensed debug layout
[IMPROVED] Sticky Control Module - First person camera Update Type options
[IMPROVED] StickyDisplayModule - update main camera when switching between 1st and 3rd person
[IMPROVED] Sticky Input Module - Added change position button for Custom Inputs
[IMPROVED] Sticky Interactive - Readable objects including levers and joysticks
[IMPROVED] Sticky Interactive - Readable value debugging in the editor
[IMPROVED] Sticky Popup Module - Sharpened text in demo prefabs
[IMPROVED] Sample Sit Action Popup has option to only show a single popup at the same time
[IMPROVED] ToggleHoldInteractive API can instantly grab non-Touchable objects
[IMPROVED] Updated link to tutorials

Version 1.1.1 – 08 August 2023

[NEW] StickyManager - support for additive scenes
[FIXED] NPC Jane and Rod should not have an enabled first-person camera
[FIXED] Copied style is null. Using StyleNotFound instead (U2022.3 only)
[IMPROVED] StickyPopupModule - orient toward camera up when camera is assigned

Version 1.1.2 – 02 January 2024

[NEW] Amy, Bryce, Jeff, and Kate stylised animated characters
[NEW] Shape Module with APIs for facial emotions and reactions
[NEW] StickyControlModule - manually set zoom with API
[NEW] StickyControlModule - is friend or foe APIs
[NEW] Option to prevent character sprinting backward
[NEW] Animate Action Move Speed Normalise and InvN options
[NEW] Animate Action Strafe Speed Normalise option
[NEW] Animate Action Turning Speed

[NEW] Carpet footstep sounds
[NEW] More natural crafted walk and run animations
[NEW] Third Person camera align to current position editor option
[NEW] Third Person camera lock to position APIs
[FIXED] Linked animation actions in Custom Inputs ignore conditions
[FIXED] EnableCharacter should not enable look on NPCs
[FIXED] NPCs incorrectly use FreeLook
[FIXED] Missing demo lever materials
[IMPROVED] Auto-activate the character gameobject on initialise
[IMPROVED] Reduced B43A1 rifle LOP from 34.7 to 32.0
[IMPROVED] Updated SRP packages to URP and HDRP 10.7.0
[IMPROVED] SampleLookAtPlayer NPC walk offset from player
[IMPROVED] Updated VR setup instructions for U2022.3
[IMPROVED] SampleSitAction/NPC/Popup onSeated/Stoodup callbacks
[IMPROVED] Foot IK refactor

Version 1.1.3 – 07 March 2024

[NEW] Option to manually update Celestials from Sci-Fi Ship Controller
[NEW] Set temporary reference frame APIs
[NEW] SampleSitActionPopup works with SSC Multi-stage Seat animator
[NEW] SampleSitActioNPC works with SSC Multi-stage Seat animator
[NEW] StickyZone - option to enable colliders on initialise
[FIXED] Toggle Third Person at runtime in editor on un-initialised causes null reference
[FIXED] NullReference when removing a Speech Audio clip in Sticky Shapes Module
[IMPROVED] Sticky Shapes Module - add speech audio at runtime
[IMPROVED] Third Person focus position when sitting
[IMPROVED] Hand IK when player is sitting

Version 1.1.4 – 12 June 2024

[NEW] SSCOutputBridge
[NEW] Support for Meta XR Core in VR
[FIXED] SSCInputBridge.IsInitialised always returns false
[FIXED] StickyZone - Enable Colliders option missing in editor
[FIXED] StickyInteractive - disable rbody gravity if not in use for Unity 6+
[FIXED] StickyInteractive - readable position unstable with sudden movement
[IMPROVED] StickyInteractive - ReInitialiseReadable() can be called multiple times
[IMPROVED] SSCInputBridge APIs
[IMPROVED] Compatibility with Unity 6

Trade Marks

“Unity” is a trade mark of Unity Technologies and is in no way associated with SCSM Pty Ltd.

“Mixamo” is a trade mark of Adobe Systems Incorporated and is in no way associated with SCSM Pty Ltd.

Other names or brands are trademarks of their respective owners.