VirtualGrasp Unity API

[ENUMS] describe VirtualGrasp enums and are listed first only to allow other parts of this pdf to link back to them.

[EVENTS] describe VirtualGrasp events and are very useful to listen to for your own application.

Any of the (100) API functions from an _API section can be accessed typing 'VG_Controller.' from C# scripts.
They are grouped in different sections for readability.

[ENUMS] (Unity API)

VG_BoneType

An enum to describe a bone type, used for accessing of bones from outside the library.

WRIST: The wrist bone of a hand

ELBOW: The elbow bone of an arm

SHOULDER: The shoulder bone of an arm

CLAVICLE: The clavicle bone of an arm

APPROACH: The approach handle of a grasp

VG_EditorAction

Action towards the grasp editor, see EditGrasp()

PRIMARY_CURRENT: Label the current grasp as primary, so it will be the only grasp for this object

DISABLE_CURRENT: Label the current grasp as disabled, so it will not be accessible for static grasping.

DELETE_CURRENT: Currently the same as DISABLE_CURRENT, since we do not really want to remove grasps.

DELETE_ALL_HAND_GRASPS: Delete the HandGrasp entry for a given object and hand hash

ADD_CURRENT: Add the current grasp as a valid one, so it becomes accessible for static grasping.

CLEAR_PRIMARY: Remove the label of the current object's primary grasp, so all grasps will be valid again.

CLEAR_DISABLED: Remove the label of the current object's disabled grasps, so all grasps will be valid again.

VG_FingerControlType

An enum to describe how fingers are controlled.

BY_NOTHING: When not grasping, fingers are not controlled at all.

BY_SENSOR_FULL_DOFS: When not grasping, fingers are fully controlled by sensor

BY_SENSOR_LOW_DOFS: When not grasping, fingers are controlled by sensor, but less DOF.

BY_ANIMATION: When not grasping, fingers are controlled by animation.

BY_OSCILLATED_ANIMATION: When not grasping, fingers are controlled by oscillating between two state of animations

VG_GestureType

A humanoid hand gesture type enum.

UNKNOWN_GESTURETYPE: Unknown type

PUSH_GESTURE: Index finger push gesture

FIST_GESTURE: Fist gesture

VG_GraspLabel

For labeling grasps (grasp editor functionality).

DISABLED: Labels a grasp as disabled

PRIMARY: Labels a grasp as primary

SUCCEEDED: Labels a grasp as succeeded

FAILED: Labels a grasp as failed

RANK: TBD

VG_GraspType

Animation grasp type enum.

UNKNOWN_GRASPTYPE: Unknown type

POWER: Humanoid power grasp

PINCH: Humanoid pinch grasp

OPENING: Robotic opening grasp

CLOSING: Robotic closing grasp

SUCTION_PIN: Robotic suction pin grasp

VG_HandSide

We support two hands per avatar, left and right in this enum.

LEFT: Left hand

UNKNOWN_HANDSIDE: Unknown hand side

RIGHT: Right hand

VG_InteractionType

An enum to describe a hand interaction type (i.e. a mode on grasp visualization).

TRIGGER_GRASP: Default, hand goes to object at grasp position

PREVIEW_GRASP: Grasp is always previewed once object is selected, trigger will pick up the object

PREVIEW_ONLY: like PREVIEW_GRASP, but trigger will not allow pick up the object

JUMP_GRASP: Object jumps to hand when grasp is triggered

STICKY_HAND: Object sticks to hand without forming grasp pose when grasp is triggered

JUMP_PRIMARY_GRASP: Using mechanism like JUMP_GRASP, but use a primary grasp in grasp DB

VG_JointType

Different articulated joint types supported by VG.

REVOLUTE: revolute joint with constrained rotational movement around an axis

PRISMATIC: prismatic joint with constrained translational movement along an axis

FIXED: fixed, non-moveable joint

FLOATING: floating, unconstrained joint

PLANAR: planar joint with constrained translational movement on a plane

CONE: 3-DOF ball and socket joint modeled with cone joint limit

VG_MotionType

Whether the motion is free or limited

Limited: If motion is limited by the limits in the joint's degree of freedom(s)

Free: If motion is free in the joint's degree of freedom(s)

VG_NetworkSignal

Enum bitmask to compose parts of a NetworkSignal

None: Empty signal

ControllerSignal: Flag for the controller part of the network signal.

SensorSignal: Flag for the sensor part of the network signal.

TriggerSignal: Flag for the trigger part of the network signal.

ObjectSignal: Flag for the object part of the network signal.

VG_QueryGraspMethod

The query grasp method for GetGrasp() function

BY_INDEX: get grasp by index

BY_ID: get grasp by ID

BY_TCP: get grasp by TCP

VG_QueryGraspMode

Decide when query grasp if hand moves and how to move hand.

NO_MOVE: will not move internal object and hand

MOVE_HAND_SMOOTHLY: will move object and hand moves smoothly with a transition period

MOVE_HAND_DIRECTLY: will move object and hand move directly to target grasp pose

VG_ReturnCode

ReturnCode for various VirtualGrasp functions.Most functions in this API provide such a return code.

SUCCESS: Succeeded in processing function

DLL_NOT_INITIALIZED: Failed in processing function because library has not been initialized.

DLL_FUNCTION_FAILED: Failed in processing function because library has not been initialized.

INVALID_AVATAR: Failed in processing function because the provided avatar is invalid.

INVALID_LIMB: Failed in processing function because the provided limb or object is invalid.

INVALID_GRASP: Failed in processing function because the provided grasp is invalid.

INVALID_TARGET: Failed in processing function because the provided target is invalid.

ARGUMENT_ERROR: Failed in processing function because a provided argument is invalid.

UNSUPPORTED_FUNCTION: Failed in processing function because it is unsupported.

OBJECT_NO_GRASPS: Failed in processing function because there are no static grasps baked.

OBJECT_NO_BAKE: Failed in processing function because a baking process failed / there is no bake at all.

LOAD_GRASP_DB_FAILED: Failed to pass a grasp db file into the library and process it.

SAVE_GRASP_DB_FAILED: Failed to export the internal grasp db to a file.

UNKNOWN_AVATAR:

AVATAR_BLOCKED:

ARTICULATION_SETUP_FAILED:

NO_SENSOR_DB:

ARTICULATION_NO_CHANGE:

VG_SensorControlFlags

Enum flag to describe what controller signals a sensor should cover.

POSITION: Enable wrist position signal

ROTATION: Enable wrist rotation signal

FINGERS: Enable finger configuration signals

GRASP: Enable grasp trigger signal

HAPTICS: Enable haptics signals

VG_SensorType

Different sensor (or controller) types that can be used by VirtualGrasp. Note only External Controller is supported.

NO_CONTROLLER: no controller

LEAP: Internal Controller (not supported), Leap motion 3D camera

RAZER_HYDRA: Internal Controller (not supported), Razer Hydra controllers

INTEL_REALSENSE: Internal Controller (not supported), Intel Realsense 3D camera

MANUS: Internal Controller (not supported), Manus VR gloves

KNUCKLES: Internal Controller (not supported), Valve Knuckles controller

VIVE: Internal Controller (not supported), HTC Vive controllers, supported through OpenVR

OCULUS_TOUCH_OPENVR: Internal Controller (not supported), Oculus Touch controllers, supported through OpenVR

VIVE_TRACKER: Internal Controller (not supported), A ViveTracker

OCULUS_TOUCH_OVR: Internal Controller (not supported), Oculus Touch controllers, through OculusVR.

EXTERNAL_CONTROLLER: External Controller, customized controller

BEBOP: Internal Controller (not supported), Bebop VR gloves

VG_VrButton

Enum for setting which (VR) controller buttons.

TRIGGER: Use the trigger button (usually index finger on the controller) to grasp.

GRIP: Use the grip button (usually middle finger on the controller) to grasp.

GRIP_OR_TRIGGER: Use both the trigger and the grip button (logical OR) to grasp.

[EVENTS] (Unity API)

VG_Controller.OnAvatarSpecificObjectSelectionWeightChanged

This event is invoked when an avatar-specific object selection weight is changed. The event carries the object and avatarID for which the weight has been changed and the new weight.

VG_Controller.OnGraspTriggered

This event is invoked in the frame when a hand is starting to grasp an object. The VG_HandStatus it carries includes more information about the interaction.

VG_Controller.OnInitialize

The event to call when we have successfully initialized the library.

VG_Controller.OnObjectCollided

This event is invoked when a grasped object is colliding with another object. The VG_HandStatus it carries includes more information about the interaction.

VG_Controller.OnObjectDeselected

This event is invoked in the frame when a hand is starting to deselect an object. The VG_HandStatus it carries includes more information about the interaction.

Tutorial: VG_Highlighter

VG_Controller.OnObjectFullyReleased

This event is invoked in the frame when an object is fully release by all hands. The Transform it carries includes the object that has just been released.

VG_Controller.OnObjectGrasped

This event is invoked in the frame when a hand has fully grasped an object. The VG_HandStatus it carries includes more information about the interaction.

VG_Controller.OnObjectJointChanged

This event is invoked when an object's articulation / joint is changed. The VG_Articulation it carries includes more information about the joint.

VG_Controller.OnObjectPushed

This event is invoked in the frame when a hand pushing an object. The VG_HandStatus it carries includes more information about the interaction.

VG_Controller.OnObjectReleased

This event is invoked in the frame when a hand is starting to release an object. The VG_HandStatus it carries includes more information about the interaction.

VG_Controller.OnObjectSelected

This event is invoked in the frame when a hand is starting to select an object. The VG_HandStatus it carries includes more information about the interaction.

Tutorial: VG_Highlighter

VG_Controller.OnObjectSelectionWeightChanged

This event is invoked when an object's selection weight is changed. The event carries the object for which the weight has been changed and the new weight.

VG_Controller.OnPostUpdate

This event is invoked in the update loop after VG runs its update. Thus, all other scripts that should update after the VG cycle should listen to this event.

VG_Controller.OnPreUpdate

This event is invoked in the update loop before VG runs its update. Thus, all other scripts that should update before the VG cycle should listen to this event.

OBJECT_SELECTION_API (Unity API)

VG_Controller.ChangeObjectJoint

Change a set of prameters of an object's joint in runtime.

Transform selectedObject: The object to change the object joint parameters.

VG_JointType **new_jointType:** The new joint type.

VG_MotionType **new_motionType:** The new motion type specifying if motion should be limited or free.

Transform new_anchor_transform: The new anchor transform.

Vector2 new_limit: The new limit of the new joint type. For planar joint this is the limit along xaxis of the anchor transform.

float new_screwRate: The new screw rate (>=0, in cm per degree) for revolute joint.

Vector2 new_limit2: The new limit along yaxis of the anchor transform for planar joint.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

Remark: If new_screwRate is set to 0 then rotating of revolute object will not move object position along the joint axis.

Remark: Recommend to use this in LateUpdate to guarantee object pose is in sync with VirtualGrasp library.

VG_Controller.ChangeObjectJoint

Change an object's joint and all other articulation parameters in runtime.

Transform selectedObject: The object to change the joint for.

VG_Articulation articulation: An articulation describing the new articulation parameters.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

Remark: Recommend to use this in LateUpdate to guarantee object pose is in sync with VirtualGrasp library.

VG_Controller.ClearAvatarSpecificObjectSelectionWeights

Clear all avatar specific object selection weights.

int avatarID: The avatar id

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

VG_Controller.GetAvatarSpecificObjectSelectionWeight

Returns the avatar specific object selection weight of an object for interaction.

int avatarID: The avatar id

Transform obj: Which object to specify weight

out float weight: The corresponding weight

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

Remark: Note by default this weight is equal to the object's selection weight for all avatars.

Remark: Use case is mainly to specify relative selection preferences for cluttered objects.

VG_Controller.GetGraspButton

Return the currently selected GraspButton.

VG_Controller.GetGraspingAvatars

Return the avatar/hand pairs that are currently grasping a specified object.

Transform objectToCheck: The object to be checked if it is currently grasped.

out List<KeyValuePair<int, VG_HandSide>> hands: An output list of avatar-handside-pairs describing which hands are currently grasping that object.

returns int: Number of hands grasping the object.

VG_Controller.GetObjectJointState

Get the current joint state of a single-dof articulated object. For planar joint, the joint state along xaxis of the joint anchor.

Transform selectedObject: The object to get the current joint state value for.

out float jointState: The returned joint state. Will be set to 0.0f upon error

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call. when selectedObject is null, or VG_ReturnCode.DLL_FUNCTION_FAILED on an unexpected error.

VG_Controller.GetObjectJointType

Get object's original or current joint type.

Transform selectedObject: The object to get the current joint state value for.

bool original: If true, get the original joint type, otherwise the current type.

out VG_JointType jointType: The returned joint type. Will be set to FLOATING upon error.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call. when selectedObject is null, or VG_ReturnCode.DLL_FUNCTION_FAILED on an unexpected error.

VG_Controller.GetObjectSecondaryJointState

Get the current secondary joint state along yaxis of joint anchor for planar articulated object.

Transform selectedObject: The object to get the current joint state value for.

out float secondaryJointState: The returned secondary joint state. Will be set to 0.0f upon error.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call. when selectedObject is null, or VG_ReturnCode.DLL_FUNCTION_FAILED on an unexpected error.

VG_Controller.GetObjectSelectionWeight

Returns the object selection weight for grasping interaction.

Transform obj: Which object to specify weight

out float weight: The corresponding weight

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

Remark: Note by default this weight is 1 for all objects.

Remark: Use case is mainly to specify relative selection preferences for cluttered objects.

VG_Controller.GetSelectableObjects

Return all interactable objects.

bool excludeHidden: If to exclude objects that have been hidden in the scene.

bool excludeUntagged: If to exclude objects that have been untagged in the scene.

returns IEnumerable<Transform>: All interactable objects in the scene.

VG_Controller.GetSelectableObjectsFromScene

Return all interactable objects from the editor scene.

bool excludeHidden: If to exclude objects that have been hidden in the scene.

bool excludeUntagged: If to exclude objects that have been untagged in the scene.

returns List<Transform>: All interactable objects in the editor scene.

VG_Controller.GetSensorPose

Receive the sensor pose of a given avatar and hand.

int avatarID: The avatar to get the pose from.

VG_HandSide **handSide:** The hand side to get the pose from.

out Vector3 p: The returned position.

out Quaternion q: The returned rotation.

bool absolute: Set True (default) to return the absolute pose, and False to return the relative pose.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

VG_Controller.GetUnbakedObjects

Return all unbaked objects.

returns List<Transform>: A list of all unbaked objects in the scene as Unity Transforms.

VG_Controller.JumpGraspObject

Specify an object to be grasped by a hand no matter how far the object is, and object will jump to the hand.

int avatarID: The avatar id

VG_HandSide **handSide:** The side of the hand

Transform obj: The object that will be jump grasped by this hand

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

VG_Controller.RecoverObjectJoint

Recover an object's original joint, after it has been changed by ChangeObjectJoint().

Transform selectedObject: The object to recover the joint for.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

Remark: Recommend to use this in LateUpdate to guarantee object pose is in sync with VirtualGrasp library.

VG_Controller.SetAvatarSpecificObjectSelectionWeight

Specify the avatar specific object selection weight of an object for interaction.

int avatarID: The avatar id

Transform obj: Which object to specify weight

float weight: Should be >=0 value to specify the preferences to select this object. If 0 exclude this object in selection process

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

Remark: Note by default this weight is equal to the object's selection weight for all avatars.

Remark: Use case is mainly to specify different grasp preferences for avatars e.g. master vs. student grasp abilities.

VG_Controller.SetDualHandsOnly

Set if an object can only be manipulated by dual hands from a same avatar.

Transform selectedObject: The object to change the dual hand type for.

bool dualHandsOnly: If dual hand only.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

VG_Controller.SetObjectJointState

Set the current joint to desired state for a single-dof articulated object or planar joint object.

Transform selectedObject: The object to set the joint state value for.

float jointState: The target joint state. If exceed joint limit will be constrained within limit.

float jointState2: The target secondary joint state for Planar joint. If exceed joint limit will be constrained within limit.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

VG_Controller.SetObjectSelectionWeight

Specify the object selection weights for grasping interaction.

Transform obj: Which object to specify weight

float weight: Should be >=0 value to specify the preferences to select this object. If 0 exclude this object in selection process

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

Remark: Note by default this weight is 1 for all objects.

Remark: Use case is mainly to specify different grasp preferences for avatars e.g. master vs. student grasp abilities.

VG_Controller.SwitchGraspObject

Instantaneously switch the grasped object to specified object in the function, the object will jump to hand.

int avatarID: The avatar id

VG_HandSide **handSide:** The side of the hand

Transform obj: The transform of the object to switch to grasp

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

VG_Controller.TogglePrimaryGraspOnObject

Instantaneously switch the grasped object, and continously calling also toggle through primary grasps on this object.

int avatarID: The avatar id

VG_HandSide **handSide:** The side of the hand

Transform obj: The transform of the object to switch to grasp and toggle primary grasps.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

Remark: The specified object should have JUMP_PRIMARY_GRASP interaction type and has added primary grasps in the grasp db.

VIRTUALGRASP_CONTROLLER_FUNCTIONS (Unity API)

VG_Controller.Clear

Reset the plugin.

VG_Controller.GetAvatarID

Get the AvatarID of the given skinned mesh renderer

out int avatarID: The returned AvatarID.

returns VG_ReturnCode: VG_ReturnCode.SUCCESS on successfull avatar id fetch, or VG_ReturnCode.INVALID_AVATAR if avatar is null.

VG_Controller.GetDebugPath

Return the path where VG stores debug files.

returns string: The path (platform dependent).

VG_Controller.GetHand

Receive a specific hand and its status.

int avatarID: The avatar to get the hand status for.

VG_HandSide **side:** The hand side to get the avatar from.

returns VG_HandStatus: A VG_HandStatus.

VG_Controller.GetHands

Receive an enumerator of all registered hands and their status.

returns List<VG_HandStatus>: Enumerator over VG_HandStatus.

VG_Controller.GetSensorControlledAvatarID

Get the AvatarID of the first sensor controlled avatar.

out int avatarID: The returned AvatarID. Will be set to -1 upon error.

returns VG_ReturnCode: VG_ReturnCode.SUCCESS on successfull avatar id fetch, or VG_ReturnCode.DLL_FUNCTION_FAILED on an unexpected error.

Remark: No guarantee on returning the one that was first sensor controlled avatar

VG_Controller.Initialize

Initialize the plugin.

VG_Controller.IsEnabled

Check if the plugin has been initialized and is ready to use.

VG_Controller.IsolatedUpdate

The Update() method has been divided into three parts: IsolatedUpdateDataIn(), IsolatedUpdate() and IsolatedUpdateDataOut() for application of the Burst compiler. IsolatedUpdate() runs the main update loop in VG.

VG_Controller.IsolatedUpdateDataIn

The FixedUpdate() method has been divided into three parts: IsolatedUpdateDataIn(), IsolatedUpdate() and IsolatedUpdateDataOut() for application of the Burst compiler. IsolatedUpdateDataIn() isolates data communication from Unity to VG.

VG_Controller.IsolatedUpdateDataOut

The Update() method has been divided into three parts: IsolatedUpdateDataIn(), IsolatedUpdate() and IsolatedUpdateDataOut() for application of the Burst compiler. IsolatedUpdateDataOut() isolates data communication from VG to Unity.

VG_Controller.RegisterRemoteAvatar

Register a new remote avatar during runtime.

SkinnedMeshRenderer avatar: The skinned mesh renderer of the model that should be registered to VG.

int networkID1: If networking is used, these will be the networkingIDs of the left hand of the new avatar (we assume max 2 hands per avatar).

int networkID2: If networking is used, these will be the networkingIDs of the right hand of the new avatar (we assume max 2 hands per avatar).

out int id: The new avatar ID will be assigned to this value after registration; -1 if it failed.

returns **VG_ReturnCode:** VG_ReturnCode describing the error state of the function call.

VG_Controller.RegisterReplayAvatar

Register a new avatar during runtime.

SkinnedMeshRenderer avatar: The skinned mesh renderer of the model that should be registered to VG.

out int id: The new avatar ID will be assigned to this value after registration; -1 if it failed.

returns **VG_ReturnCode:** VG_ReturnCode describing the error state of the function call.

VG_Controller.RegisterSensorAvatar

Register a new avatar during runtime. Single sensor controlling each hand.

SkinnedMeshRenderer avatar: The skinned mesh renderer of the model that should be registered to VG.

out int id: The new avatar ID will be assigned to this value after registration; -1 if it failed.

VG_SensorSetup primarySetup: The primary sensor setup used to control the avatar.

returns **VG_ReturnCode:** VG_ReturnCode describing the error state of the function call.

VG_Controller.RegisterSensorAvatar

Register a new avatar during runtime. Double sensor for each hand.

SkinnedMeshRenderer avatar: The skinned mesh renderer of the model that should be registered to VG.

out int id: The new avatar ID will be assigned to this value after registration; -1 if it failed.

VG_SensorSetup primarySetup: The primary sensor setup used to control the avatar.

VG_SensorSetup secondarySetup: The secondary sensor setup used to control the avatar.

returns **VG_ReturnCode:** VG_ReturnCode describing the error state of the function call.

VG_Controller.Release

Release the plugin.

VG_Controller.SaveState

Save the object hierarchy debug state. This is done automatically when closing VirtualGrasp.

VG_Controller.SetRecordingStatesOnAvatar

Set if use the avatar for recording response states during sensor recording or replay.

int avatarID: The avatar id.

bool recordingStates: If use this avatar to record response states in sensor db.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

Remark: When recordingStates is true, if avatar is used for sensor recording, then recording will also include response states, while if avatar is used for replay sensor db, then replay will update response states in the sensor db.

VG_Controller.UnRegisterAvatar

Unregister avatar during runtime

int avatarID: The id of the avatar to be unregistered.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

RECORDING_INTERFACE_API (Unity API)

VG_Controller.CollectRecording

pro*

Collect recording sensor data.

out byte[] recording: [output] byte array containing the sensor recording.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

Tutorial: VG_Recorder

VG_Controller.GetReplayAvatarID

pro*

Get the AvatarID of the first replay avatar.

out int avatarID: The returned AvatarID. Will be set to -1 upon error.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

Remark: No guarantee on returning the one that was first registered as replay avatar

VG_Controller.GetReplayStartWristPose

pro*

Get the starting wrist poses for full replay of the whole interaction sequence.

int avatarID: The ID of the avatar to play the recording on (note: it has to be an avatar enabled for replay).

Transform selectedObject: If provided, the entire sensor recording will transformed in to object's frame. If not, in global frame.

out Vector3 p_left: The position of the left wrist.

out Quaternion q_left: The orientation of the left wrist.

out Vector3 p_right: The position of the right wrist.

out Quaternion q_right: The orientation of the right wrist.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

Remark: LoadRecording need to be called before this to load recorded sensor data.

Remark: SetProcessByRecordedFrame need to be called before this to set this avatar to be enabled for replay.

Tutorial: VG_Recorder

VG_Controller.IsReplaySuccess

pro*

Check if finished replay had identical response as recorded

returns bool: True if replay was identical, False otherwise.

Tutorial: VG_Recorder

VG_Controller.IsReplaying

pro*

Check if a hand is currently replaying a recorded sensor data.

int avatarID: The avatar to check.

VG_HandSide **handSide:** The hand to check.

returns bool: True if replaying, False otherwise.

Tutorial: VG_Recorder

VG_Controller.LoadRecording

pro*

Load recorded sensor data from a file, but do not start replay

string filename: The filename to load the recording from.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

Tutorial: VG_Recorder

VG_Controller.LoadRecording

pro*

Load recorded sensor data from a byte array.

byte[] recording: The byte array to load the recording from.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

Tutorial: VG_Recorder

VG_Controller.ResumeReplay

pro*

Resume replaying of an avatar.

int avatarID: The ID of the avatar to resume replaying the recording on (note: it has to be an avatar enabled for replay).

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

Tutorial: VG_Recorder

VG_Controller.SaveRecording

pro*

Save recording sensor data and store the whole sequence to a file

string filename: The filename to save the recording to.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

Tutorial: VG_Recorder

VG_Controller.StartRecording

pro*

Start recording sensor data.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

Tutorial: VG_Recorder

VG_Controller.StartReplay

pro*

Start full replay of the whole interaction sequence on an avatar.

int avatarID: The ID of the avatar to play the recording on (note: it has to be an avatar enabled for replay).

Transform selectedObject: If provided, the entire sensor recording will be replayed in this object's frame. If not, in global frame.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

Tutorial: VG_Recorder

VG_Controller.StartReplayOnObject

pro*

Start replaying a specific interaction segment on one object.

Transform obj: The object to play the interaction on.

int avatarID: The avatar to play the interaction with.

VG_HandSide **handSide:** The hand to play the interaction with.

int interactionId: The ID of the interaction segment to be played on this object.

returns **VG_ReturnCode:** VG_ReturnCode describing the error state of the function call.

Tutorial: VG_Recorder

VG_Controller.StopRecording

pro*

Stop recording sensor data.

returns **VG_ReturnCode:** VG_ReturnCode describing the error state of the function call.

Tutorial: VG_Recorder

VG_Controller.StopReplay

pro*

Stop replay of the recorded interaction sequence on an avatar.

int avatarID: The ID of the avatar to play the recording on (note: it has to be an avatar enabled for replay).

returns **VG_ReturnCode:** VG_ReturnCode describing the error state of the function call.

Tutorial: VG_Recorder

GRASP_EDITOR_API (Unity API)

VG_Controller.EditGrasp

movie

Call grasp editor functionality on a currently selected object and grasp.

int avatarID: The avatar to call grasp editor functionality on.

VG_HandSide **handSide:** The hand side to call grasp editor functionality on.

VG_EditorAction **action:** The grasp editor function / action to call.

Transform obj: The object to call the action on (if not provided, the object in the hand).

int grasp: The grasp ID to call the action on (if not provided, the current grasp of the hand).

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

Tutorial: VG_GraspStudio

VG_Controller.GetGrasp

Receive a grasp in the grasp DB by index.

Transform selectedObject: The object to receive a grasp for.

int avatarID: The avatar to receive a grasp for.

VG_HandSide **handSide:** The hand side to receive a grasp for.

int graspIndex: The index of grasp to receive.

out Vector3 p: The received wrist position of the grasp.

out Quaternion q: The received wrist orientation of the grasp.

out VG_GraspType type: The received VG_GraspType of the grasp.

out VG_GraspLabel label: The received VG_GraspLabel of the grasp.

VG_QueryGraspMode **queryGraspMode:** Can be used to define if and how the grasp should be applied also.

VG_QueryGraspMethod **queryGraspMethod:** Can be used to define how the graspIndex should be interpreted.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

Tutorial: VG_GraspStudio

VG_Controller.GetInteractionTypeForObject

Get the current interaction type assigned to an object.

Transform selectedObject: The object to receive the interaction type for.

returns VG_InteractionType: VG_InteractionType describing the current interaction type of the object.

VG_Controller.GetNumGraspsInDB

Receive the number of saved grasps in the grasp db for a specific object, and optionally a specified hand.

Transform selectedObject: The object to get the number of available grasps for.

int avatarID: If a valid avatarID together with handSide, receive only the available grasps for this hand (otherwise all available grasps).

VG_HandSide **handSide:** If a valid handSide together with avatarID, receive only the available grasps for this hand (otherwise all available grasps).

returns int: The number of saved grasps in the grasp db for the selected object (either all or for the specified hand).

Tutorial: VG_HintVisualizer

VG_Controller.GetNumPrimaryGraspsInDB

Receive the number of primary grasps in the grasp db for a specific object, and optionally a specified hand.

Transform selectedObject: The object to get the number of available grasps for.

int avatarID: If a valid avatarID together with handSide, receive only the primary grasps for this hand (otherwise all available grasps).

VG_HandSide handSide: If a valid handSide together with avatarID, receive only the primary grasps for this hand (otherwise all available grasps).

returns int: The number of primary grasps in the grasp db for the selected object (either all or for the specified hand).

Tutorial: VG_HintVisualizer

GRASP_SELECTION_API (Unity API)

VG_Controller.ForceReleaseObject

untested

Force the release of a grasp.

int avatarID: The avatar to release grasps on all its hands.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

VG_Controller.ForceReleaseObject

untested

Force the release of a grasp.

int avatarID: The avatar to release a grasp for.

VG_HandSide **side:** The hand which to release the grasp for.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

VG_Controller.GetBone

Return the pose (i.e. position and orientation) of a specific bone.

int avatarID: The avatar to get the bone pose from.

VG_HandSide **handSide:** The hand side to get the bone pose from.

VG_BoneType **boneType:** The BoneType to get.

out Transform t: The returned pose of the bone.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

Tutorial: VG_HandVisualizer

VG_Controller.GetBone

Return the Transform that corresponds to a provided instance ID.

int transformID: The instance ID.

returns Transform: The Transform that corresponds to the transformID.

Tutorial: VG_HandVisualizer

VG_Controller.GetBone

Return the pose (i.e. position and orientation) of a specific bone.

int avatarID: The avatar to get the bone pose from.

VG_HandSide **handSide:** The hand side to get the bone pose from.

VG_BoneType **boneType:** The BoneType to get.

out int instanceID: The returned ID of the bone transform.

out Vector3 p: The returned position of the bone.

out Quaternion q: The returned rotation of the bone.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

Tutorial: VG_HandVisualizer

VG_Controller.GetBone

Return the pose matrix of a specific bone.

int avatarID: The avatar to get the bone pose from.

VG_HandSide handSide: The hand side to get the bone pose from.

VG_BoneType boneType: The BoneType to get.

out int instanceID: The returned ID of the bone transform.

out Matrix4x4 m: The returned pose matrix of the bone.

returns Transform: The Unity Transform that corresponds to the requested bone.

Tutorial: VG_HandVisualizer

VG_Controller.GetFingerBone

Return the pose of a specific finger bone as a matrix.

int avatarID: The avatar to get the bone pose from.

VG_HandSide handSide: The hand side to get the bone pose from.

int fingerID: The finger to get the bone pose from (from 0 as thumb to 4 as pinky).

int boneID: The bone index (from 0 as proximal to N as distal) to get the bone pose from. Use -1 for fingertip.

out int instanceID: The returned ID of the bone transform.

out Matrix4x4 m: The returned pose of the bone.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

Tutorial: VG_HandVisualizer

VG_Controller.GetFingerBone

Return the pose (i.e. position and orientation) of a specific finger bone.

int avatarID: The avatar to get the bone pose from.

VG_HandSide handSide: The hand side to get the bone pose from.

int fingerID: The finger to get the bone pose from (from 0 as thumb to 4 as pinky).

int boneID: The bone index (from 0 as proximal to N as distal) to get the bone pose from. Use -1 for fingertip.

out int instanceID: The returned ID of the bone transform.

out Vector3 p: The returned position of the bone.

out Quaternion q: The returned rotation of the bone.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

Tutorial: VG_HandVisualizer

VG_Controller.GetFingerBone

Reflect the pose of a specific bone on a Transform.

int avatarID: The avatar to get the bone pose from.

VG_HandSide handSide: The hand side to get the bone pose from.

int fingerID: The finger to get the bone pose from (from 0 as thumb to 4 as pinky).

int boneID: The bone index (from 0 as proximal to N as distal) to get the bone pose from. Use -1 for fingertip.

out Transform t: The returned pose of the bone.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

Tutorial: VG_HandVisualizer

VG_Controller.MakeGesture

Make a gesture with a hand.

int avatarID: The avatar to make gesture for.

VG_HandSide **side:** The hand which to make gesture for.

VG_GestureType **gesture:** The gesture to make with the [side] hand of avatar [avatarID].

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

VG_Controller.ReleaseGesture

Release a gesture on a hand

int avatarID: The avatar to release a grasp for.

VG_HandSide **side:** The hand which to release the grasp for.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

VG_Controller.SetBlockRelease

Specify if on this hand should block release or not in runtime.

int avatarID: The avatar to release a grasp for.

bool block: If block release signal or not on this avatar.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

VG_Controller.SetBlockRelease

Specify if on this hand should block release or not in runtime.

int avatarID: The avatar to release a grasp for.

VG_HandSide **side:** The hand which to release the grasp for.

bool block: If block release signal or not on this hand.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

VG_Controller.SetGlobalInteractionType

Set the global interaction type method. The interaction type defines how the hand and the object should get together during a grasp.

Remark: This will overwrite the specific grasp interaction type (see SetInteractionTypeForObject) for all objects.

VG_InteractionType **interactionType:** The method to switch to for all objects.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

VG_Controller.SetGlobalThrowAngularVelocityScale

Set the global throw angular velocity scale. The throw angular velocity scale defines how powerful the throw is in terms of rotation movement.

Remark: This will overwrite the specific throw angular velocity scale (see SetThrowAngularVelocityScaleForObject) for all objects.

float throwAngularVelocityScale: The throw angular velocity scale.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

VG_Controller.SetGlobalThrowVelocityScale

Set the global throw velocity scale. The throw velocity scale defines how powerful the throw is in terms of linear movement.

Remark: This will overwrite the specific throw velocity scale (see SetThrowVelocityScaleForObject) for all objects.

float throwVelocityScale: The throw translational velocity scale.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

VG_Controller.SetInteractionTypeForObject

Set the interaction type for a selected object. The interaction type defines how the hand and the object should get together during a grasp.

Remark: This will overwrite the global interaction type (see SetGlobalInteractionType) for that object.

Transform selectedObject: The object to modify the interaction type for.

VG_InteractionType **interactionType:** The interaction type to switch to for the object.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

VG_Controller.SetInteractionTypeForSelectedObject

Set the interaction type for a selected object. The interaction type defines how the hand and the object should get together during a grasp.

Remark: This will overwrite the global interaction type (see SetGlobalInteractionType) for that object.

int avatarID: The avatar which is selecting an object.

VG_HandSide **side:** The hand which is selecting an object.

VG_InteractionType **interactionType:** The interaction type to switch to for the object that is selected by the [side] hand of avatar [avatarID].

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

VG_Controller.SetThrowAngularVelocityScaleForObject

Set the throw angular velocity scale for a selected object. The throw angular velocity scale defines how powerful the throw is in terms of rotation movement.

Remark: This will overwrite the global throw angular velocity scale (see SetGlobalThrowAngularVelocityScale) for that object.

Transform selectedObject: The object to modify the throw velocity scale for.

float throwAngularVelocityScale: The throw angular velocity scale.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

VG_Controller.SetThrowAngularVelocityScaleForSelectedObject

Set the throw angular velocity scale for a selected object. The throw angular velocity scale defines how powerful the throw is in terms of rotation movement.

Remark: This will overwrite the global throw angular velocity scale (see SetGlobalThrowAngularVelocityScale) for that object.

int avatarID: The avatar which is selecting an object.

VG_HandSide **side:** The hand which is selecting an object.

float throwAngularVelocityScale: The throw angular velocity scale.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.

VG_Controller.SetThrowVelocityScaleForObject

Set the throw velocity scale for a selected object. The throw velocity scale defines how powerful the throw is in terms of linear movement.

Remark: This will overwrite the global throw velocity scale (see SetGlobalThrowVelocityScale) for that object.

Transform selectedObject: The object to modify the throw velocity scale for.

float throwVelocityScale: The throw translational velocity scale.

returns **VG_ReturnCode:** VG_ReturnCode describing the error state of the function call.

VG_Controller.SetThrowVelocityScaleForSelectedObject

Set the throw velocity scale for a selected object. The throw velocity scale defines how powerful the throw is in terms of linear movement.

Remark: This will overwrite the global throw velocity scale (see SetGlobalThrowVelocityScale) for that object.

int avatarID: The avatar which is selecting an object.

VG_HandSide **side:** The hand which is selecting an object.

float throwVelocityScale: The throw translational velocity scale.

returns **VG_ReturnCode:** VG_ReturnCode describing the error state of the function call.

SENSOR_INTERFACE_API (Unity API)

VG_Controller.GetGrabStrength

Returns the current grab strength of a hand. The grab strength is 0 for a fully open hand, 1 for a fully closed hand.

int avatarID: The avatar to receive the grab strength for.

VG_HandSide **handSide:** The hand side to receive the grab strength for.

returns float: The current grab strength of the [side] hand.

VG_Controller.GetGrabVelocity

Returns the current grab velocity of a hand. The current velocity of the grab strength (see GetGrabStrength), so negative when the hand is opening, and positive when the hand is closing.

int avatarID: The avatar to receive the grab velocity for.

VG_HandSide **handSide:** The hand side to receive the grab velocity for.

returns float: The current grab velocity of the [side] hand.

VG_Controller.GetPushCircle

Get the push cirle for this hand side of an avatar as a visual hint for object selection for push without physics.

int avatarID: The avatar to get the push circle for.

VG_HandSide **handSide:** The hand to get the push circle for.

out Vector3 p: The push circle's position.

out Quaternion r: The push circle's rotation (zaxis is normal).

out float radius: Radius of the push circle,

out bool inContact: True if contact (i.e. pushing), False otherwise.

returns Transform: The selected object's Unity Transform, or null if none.

Tutorial: VG_HintVisualizer

VG_Controller.IsMissingSensorData

Check if a hand has invalid sensor data.

int avatarID: The avatar to check for.

VG_HandSide **handSide:** The hand side to check for.

returns bool: True if sensor data is invalid, False otherwise.

VG_Controller.SetAvatarActive

Set the active state of the avatar sensor(s) and mesh.

int avatarID: The avatar id.

bool enableSensors: If the sensor(s) that control this hand should be active or not.

bool enableMesh: If the mesh of this hand should be visible or not.

Vector3 resetPos: If an avatar is deactivated, hand positions will be reset to here (default (0,0,0)).

VG_Controller.SetCalibrationMode

Enable or disable wrist calibration mode (WCM). During enabled WCM, different ranges of motion of the wrist or grab strength will be calibrated.

Remark: untested

int avatarID: The avatar for which to enable/disable WCM.

bool enabled: True for enabling WCM, False for disabling it.

VG_Controller.SetExternalGrabStrength

Send an external controller grab signal to the plugin (for EXTERNAL_CONTROLLER sensors).

int avatarID: The avatar to set external sensor pose for.

VG_HandSide **handSide:** The hand side to set external sensor pose for.

float strength: The grab strength signal to set.

Tutorial: VG_ExternalControllerManager

VG_Controller.SetFingerCalibrationMode

Enable or disable finger calibration mode (FCM). During enabled FCM, the hand opening range will be calibrated. After disabling it, grasp and release signals will work in this range.

int avatarID: The avatar for which to enable/disable FCM.

bool enabled: True for enabling FCM, False for disabling it.

VG_Controller.SetSensorActive

Set the active state of the sensor(s) that control the specified hand of an instance avatar.

int avatarID: The avatar id.

VG_HandSide **handSide:** The side of the hand (remark: UNKNOWN will not have any effect).

bool active: If the sensor(s) that control this hand should be active or not.

Vector3 resetPos: If a hand is deactivated, its position will be reset to here (default (0,0,0)).

Remark: By default sensors are all active, and this function can be used in runtime to change this.

VG_Controller.SetSensorOffset

Change the sensor offset in runtime. The sensor offset is the offset between the pose that the current sensor is measuring and where the virtual hand is appearing in the scene.

Remark: Also treating left hand (LHS) and right hand (RHS) is considered, so the offset is applied symmetrically.

int avatarID: The avatar to set the offset for.

VG_SensorType **sensor:** The sensor type to change the offset for.

Vector3? position: The offset position. Set to null if position should not be modified.

Vector3? rotation: The offset rotation. Set to null if rotation should not be modified.

returns VG_ReturnCode: VG_ReturnCode describing the error state of the function call.