Intel® RealSense™ SDK Unity* Toolkit

Table of Contents

1 Legal Information

2 Introducing the SDK

The Intel® RealSense™ SDK is a library of pattern detection and recognition algorithm implementations exposed through standardized interfaces. The library aims at lowering barriers to using these algorithms and shifting the application developers' focus from coding the algorithm details to innovating on the usage of these algorithms for next generation human computer experience.

This document describes the SDK Unity Toolkit feature.

Notational Conventions

This SDK document uses the Calibri typeface for normal prose.

With the exception of section headings, captions and the table of contents, all code-related items appear in the Courier New typeface (`pxcStatus`).

Hyperlinks appear underlined in blue, such as pxcStatus.

This is a note that provides additional information to aid your understanding of the procedure or concept.

This is a tip that provides alternate methods or shortcuts.

✓ This is a result statement which indicates what you can expect to see or happen after performing a step.

3 SDK Unity Toolkit

The SDK Unity Toolkit is a set of scripts, prefabs and other utilities aimed to facilitate the use of Intel® RealSense™ technology when creating interactive Unity applications.

The toolkit is presented as a Unity Editor extension. Many basic and advanced capabilities are available directly from the Unity's Editor user interface (UI). Game developers and designers can use this toolkit to add interactions with minimal code writing. Advanced developers can also look at the SDK Unity C# scripting support, described in the main SDK Reference Manual. As illustrated in Figure 1, the SDK Unity Toolkit and the SDK Unity C# scripting supports are layers complement each other to enable Unity applications.

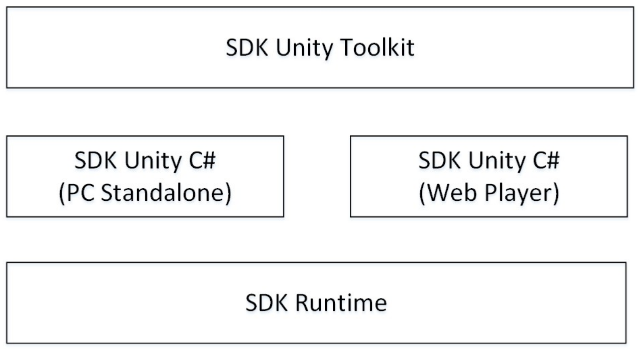


Figure 1: SDK Unity Support Layers

To add the SDK features such as hand tracking and gesture detection to a game application, simply attach the provided `Actions` scripts (such as `TrackingAction` or `RotateAction`) to an existing game object in the scene.

This Reference Manual assumes that you are familiar with the basics of the Unity Editor.

Prerequisites

The toolkit requires Unity v4.1.0 PRO (for the plugin capability), or higher.

3.1 Main Concepts

The SDK Unity Toolkit consists of the following main components:

Actions

`Actions` are `MonoBehavior` scripts. Each action defines a certain behavior on a game object. For example, the `Hide` action disables the `Mesh Renderer`.

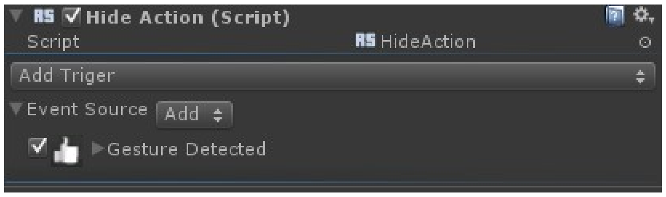


Figure 2: The Hide Action

In Figure 2, the `Event Source` is the trigger associated with the `Hide` action. Each action has a set of triggers. The action is executed whenever one of its triggers is fired.

See Actions for the detailed description of available actions ready to be used in the toolkit.

Triggers and Rules

A `Trigger` defines a family of interactions that can be implemented in several ways. For example, the `Tracking Trigger` defines all interactions that track a certain object in the real world such as a hand or a face. Each interaction is implemented as a `Rule`. Thus each trigger has its own set of rules.

A trigger also contains a set of properties that can be accessed by the action. These properties are information that relates to the interaction. For instance, the `Tracking Trigger` contains the position of the tracked object and its orientation. The action can access these properties and move the game object as needed.

`Event Source` is the trigger for general events from the SDK. In Figure 3, the game object has one rule attached, which is the `Gesture Detected` rule.

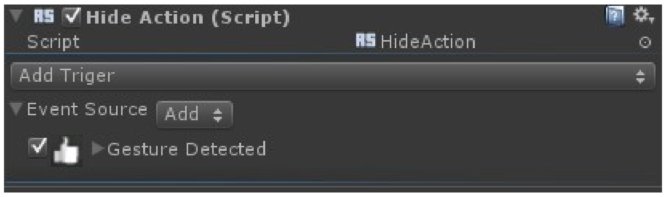
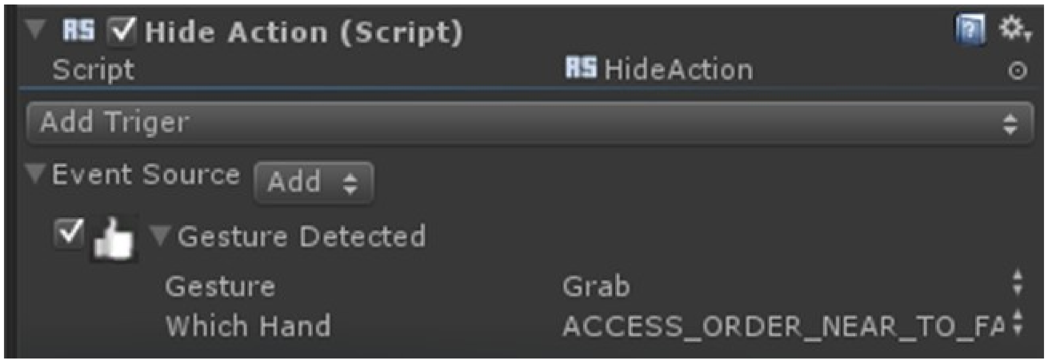


Figure 3: The Hide Action With Gesture Detected Attached

A rule implements a defined interaction. For example, the `Hand Detected` rule uses the hand tracking module in the SDK to fire an event whenever a hand is detected. This specific rule belongs to the `Event Trigger`, which indicates that an event has occurred.

In Figure 4, `Gesture Detected` is a rule for the gesture detected event. You can choose which gesture to monitor as well as which hand in the rule's parameters.



Script: HideAction

Event Source

Gesture Detected

Gesture: Grab

Which Hand: ACCESS_ORDER_NEAR_TO_FA

Figure 4: The Hide Action With Gesture Detected Parameters

This design (see Figure 5) provides flexibilities in the Unity game development. The triggers act as a buffer between the Unity game object manipulations (`Actions`) and the SDK implementation (`Rules`). You can build many actions without the need to write code if the actions can be based on available triggers and rules provided in the toolkit.

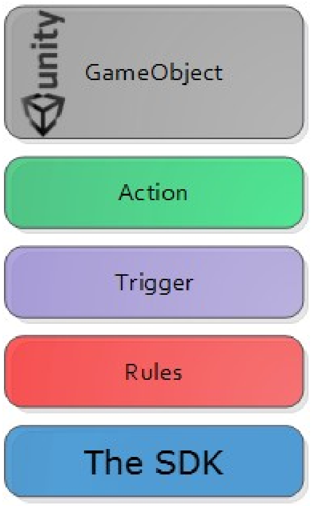


Figure 5: Unity Toolkit Components

3.2 Actions

The `Actions` are `MonoBehavior` classes that implement a specific interaction on a Unity game object. Actions use triggers to configure the specific interactions.

There are four categories of actions:

Predefined Triggers Actions

The action triggers are predefined. You can only configure the rules attached to the trigger. This category includes actions where the interaction is tightly coded with a specific trigger such as `Rotate Action` or `Tracking Action`.

Custom Trigger Actions

The action triggers are variable. You can add and remove triggers to the action. This is for more generic actions such as `Hide Action` or `Enable Behavior`.

Continuous Actions

The action implements an interaction that uses continuous flow of data. For example, the `Tracking Action` uses the location of the tracked object every frame.

One-Shot Actions

The action implements an interaction that needs to know only when a trigger is fired. For example, the `Hide Action` disables the renderer of the associated game object whenever the trigger is fired.

Action User Interface

All actions have similar layout in the Unity Inspector as follows (as illustrated in Figure 6):

- **Triggers & rules configuration**

You can specify the triggers that are associated with an action. For each trigger, you can add, remove, and configure the rules. Each rule has specific configuration parameters based on the algorithm it implements. For example, the `Gesture Detected` rule can configure which gesture to detect.

- **Action Parameters**

You can perform the following operations in the action UI:

- **List** all the rules associated with a particular trigger. To do this, click a rectangular before the trigger name. In Example 1, the `Tracking` action can configure the `Smoothing Factor` parameter.

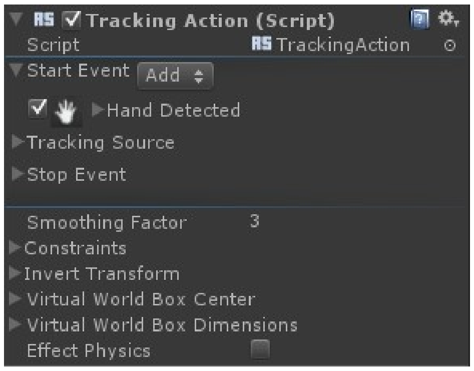


Figure 6: Action's UI – Trigger Open

- **Add** a rule to a trigger. To do this, click on the `Add` button to get a list of available rules (see Figure 7), and then click on the name of the required rule.

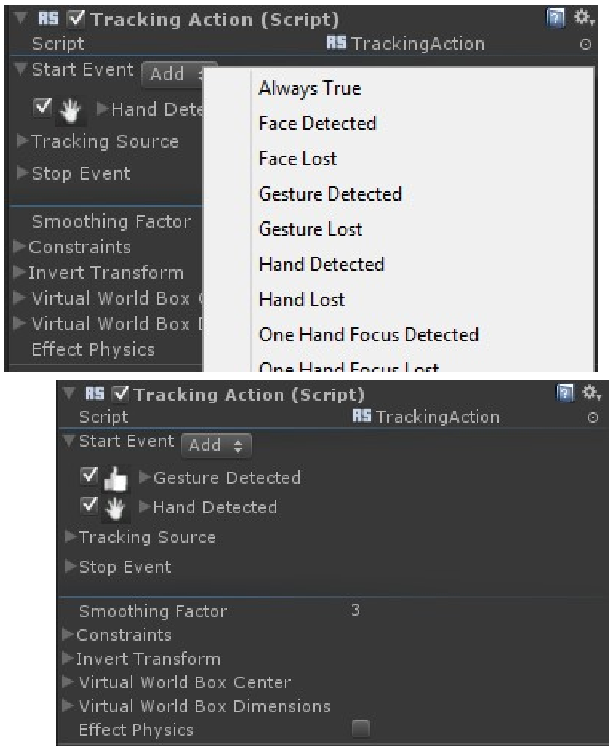


Figure 7: Action's UI – The Add Rule Button

If more than one rule are attached to a trigger, the rules run sequentially in the order of appearing with OR operand relation among them. This means that if one of the rules fires, the trigger is fired with that rule's values. For example, in Figure 7, if a gesture has been detected, the `Hand Detected` rule is ignored.

- **Skip** a rule. To do this, simply uncheck the rule's checkbox (see Figure 8). The rule becomes inactive in the execution.

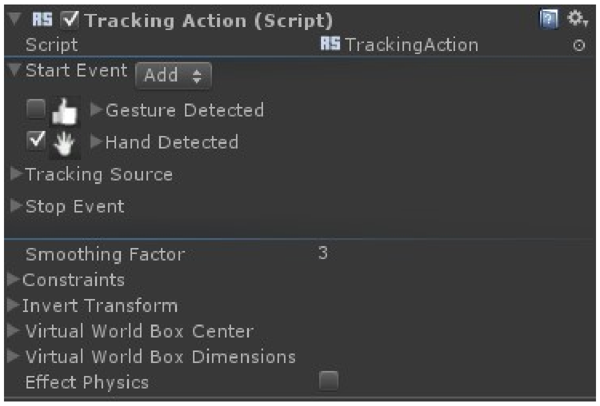


Figure 8: Action's UI – Rule un-checked

- **Get** the rule's specific configuration parameters. To do this, click the rectangular before the rule name. Each rule has specific parameters that can be changed according to the algorithm it implements.

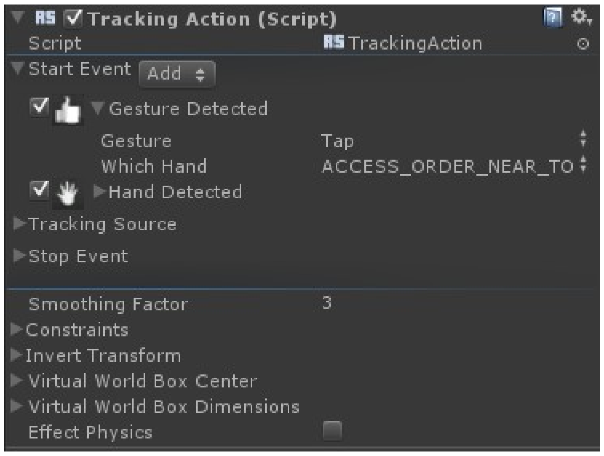


Figure 9: Action's UI – Rule's parameters

- **Remove** or **Reset** a rule/trigger. Right click the name of the rule/trigger and choose the corresponding action from the context menu, as shown in Figure 10. Resetting the rule/ trigger returns its parameters to the default values.

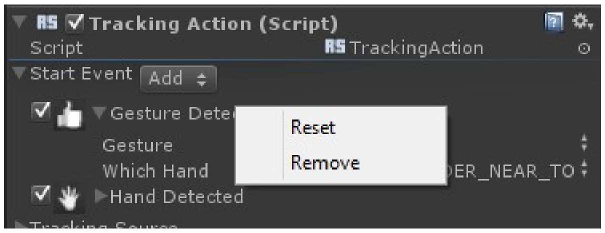


Figure 10: Action's UI – Rule's context menu

3.2.1 Continuous Actions

Continuous Actions describe a process of actions. The toolkit offers three parts to customize such a process, as illustrated in Figure 11:

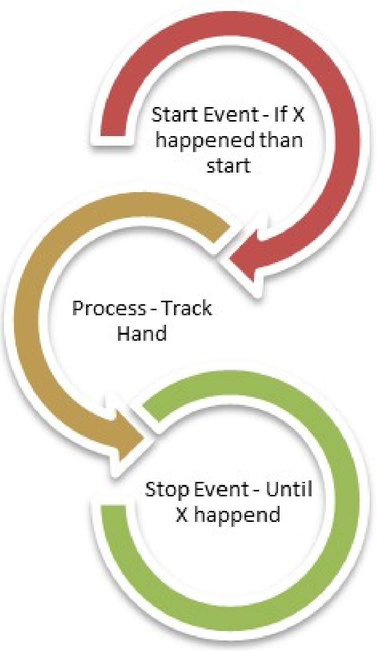
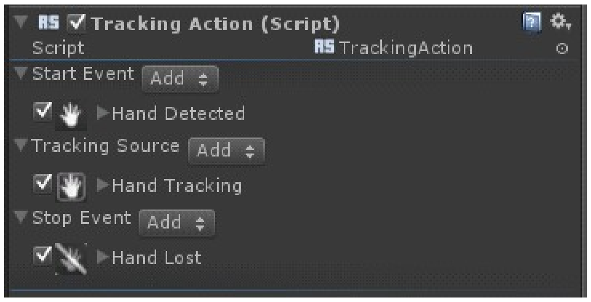

Start Event - If X happened than start
Process - Track Hand
Stop Event - Until X happend

Figure 11: 3-Stage Custom Interaction

1. **Start Event (Event Trigger):** The start trigger defines the event that triggers an interaction starting point. The event can be the detection of a hand or the detection of a gesture. You can choose any rule that implements the event trigger as the start event.

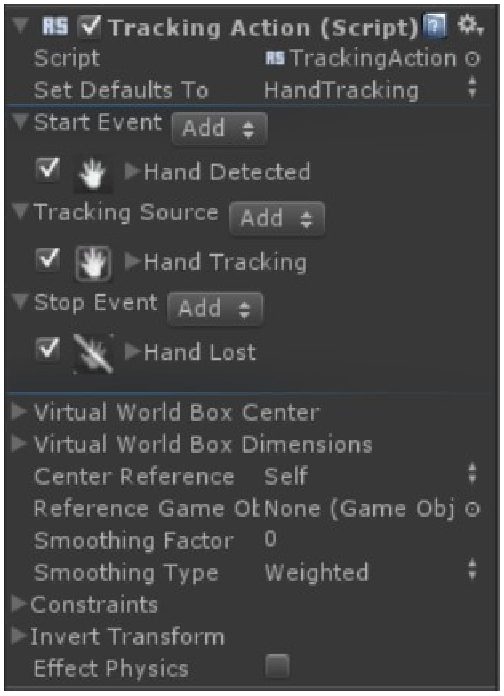
2. **Process Trigger**: The process trigger defines the interaction process. The trigger type varies with different trigger implementations. For example, the `Tracking Action` is a `Tracking Trigger` and you can configure what is being tracked.



Tracking Action (Script)
Script TrackingAction
Start Event Add
✓ Hand Detected
Tracking Source Add
✓ Hand Tracking
Stop Event Add
✓ Hand Lost

3. **Stop Event (Event Trigger):** The stop event defines the stopping criteria of the continuous operation. For example, the stopping event can be that a gesture is lost.

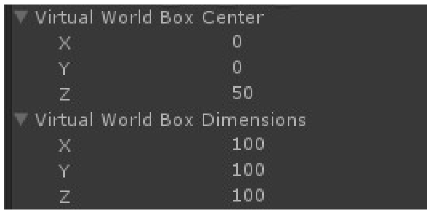
3.2.1.1 Tracking Action



Tracking Action (Script)
Script: TrackingAction
Set Defaults To: HandTracking
Start Event: Add
Hand Detected
Tracking Source: Add
Hand Tracking
Stop Event: Add
Hand Lost
Virtual World Box Center
Virtual World Box Dimensions
Center Reference: Self
Reference Game Ob: None (Game Obj
Smoothing Factor: 0
Smoothing Type: Weighted
Constraints
Invert Transform
Effect Physics

The `Tracking Action` takes the position and orientation of the specified tracked object and changes the transform of the game object. The `Tracking Action` is a `Continuous Action` thus you can define its start/process/stop events, as described Continuous Actions. See additionally the section for details on the action triggers.

The `Constraints` are a set of booleans that enable freezing the movement or rotation of the game object in a specific axis. The `Invert Transform` to enable inverting the movement and rotation of the object with respect to the tracked object.



Virtual World Box Center	
X	0
Y	0
Z	50
Virtual World Box Dimensions	
X	100
Y	100
Z	100

The `Virtual World Box` defines the boundaries of the game object's movement. You can configure how the game object moves in the virtual space. When selecting a game object with a `Tracking Action`, the toolkit visualizes the `Virtual World Box` as red lines in the scene, as similar to Figure 12. You can visually fine-tune how the game object moves in between the `Virtual World Box` and the `Real World Box`.

`Smoothing Factor` and `Smoothing Type` configure the smoother interaction. The smoothing factor of zero means no smoothing. The higher the value, the stronger smoothing is imposed on the game object's movement.

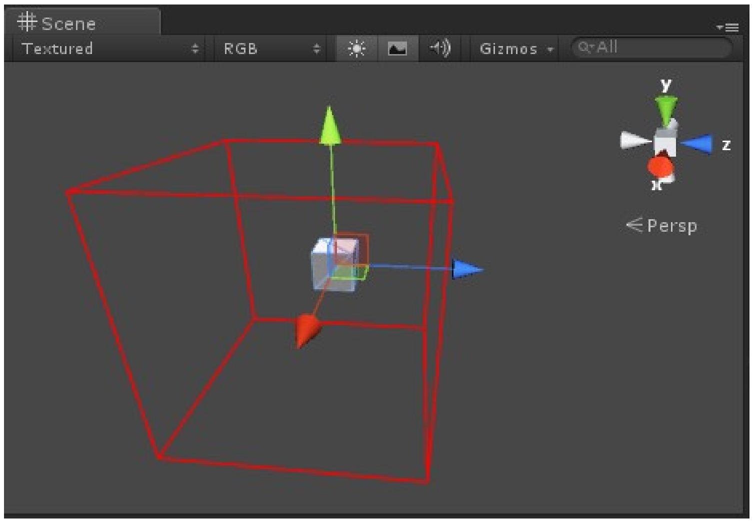
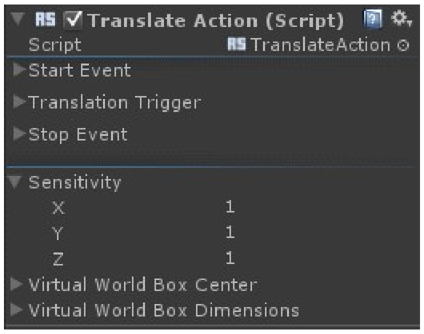


Figure 12: The Virtual World Box Configuration

3.2.1.2 Translation Action



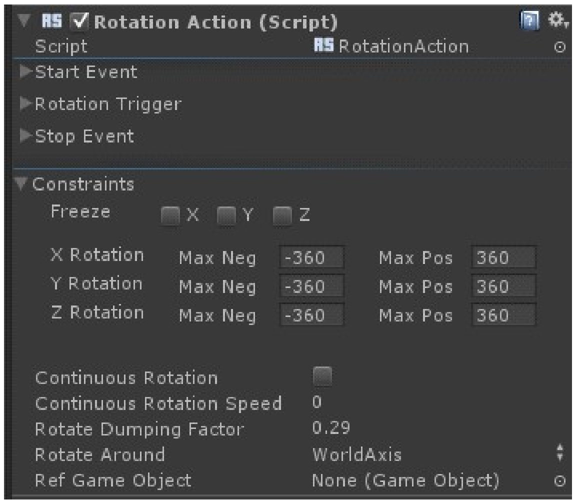
Translate Action (Script)
Script TranslateAction
Start Event
Translation Trigger
Stop Event
Sensitivity
X 1
Y 1
Z 1
Virtual World Box Center
Virtual World Box Dimensions

The `Translation Action` changes the position of the game object according to the corresponding changes in the tracked object position. The `Translation Action` is a `Continuous Action` thus you can define its start/ process/stop events, as described in Continuous Actions. See additionally the section for details on the action triggers.

You can set the `Sensitivity` parameters for each axis. The parameters configure how fast or slow the object moves with respect to the movement of the tracked object. Inserting a negative value causes the movement to flip in the corresponding axis.

This action has the same `Virtual World Box` parameters as the `Tracking Action`, described in Tracking Action.

3.2.1.3 Rotation Action



Rotation Action (Script)

Script	RotationAction
▶Start Event	
▶Rotation Trigger	
▶Stop Event	
▼Constraints	
Freeze	X Y Z
X Rotation	Max Neg -360 Max Pos 360
Y Rotation	Max Neg -360 Max Pos 360
Z Rotation	Max Neg -360 Max Pos 360
Continuous Rotation	
Continuous Rotation Speed	0
Rotate Dumping Factor	0.29
Rotate Around	WorldAxis
Ref Game Object	None (Game Object)

The `Rotation Action` rotates the game object according to the trigger's value. The `Rotation Action` is a `Continuous Action` thus you can define its start/process/stop events, as described in Continuous Actions. See additionally the section for details on the action triggers.

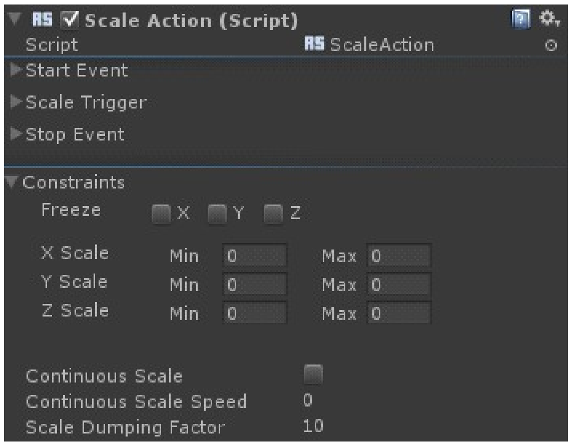
The `Constraints` parameters define any movement constraints along each axis. You can configure which axis the object will rotate and what are the maximum angles in each axis.

When enabled, the `Continuous Rotation` option continues to rotate the game object in a constant speed (specified in the `Continuous Rotation Speed` parameter) as long as the `Rotation Trigger` is fired.

The `Rotate Dumping Factor` reduces the speed of the rotation with respect to the data retrieved from the trigger.

The `Rotate Around` value sets the axis of the rotation. When set to `GameObject`, you need to additionally define a `Ref Game Object` as the reference game object.

3.2.1.4 Scale Action



Scale Action (Script)

Script	ScaleAction	
Start Event		
Scale Trigger		
Stop Event		
Constraints		
Freeze	X Y Z	
X Scale	Min 0	Max 0
Y Scale	Min 0	Max 0
Z Scale	Min 0	Max 0
Continuous Scale		
Continuous Scale Speed	0	
Scale Dumping Factor	10	

The `Scale Action` scales up or down the game object according to the trigger's values. The `Scale Action` is a `Continuous Action` thus you can define its start/process/stop events, as described in the section Continuous Actions. See additionally the section for details on the action triggers.

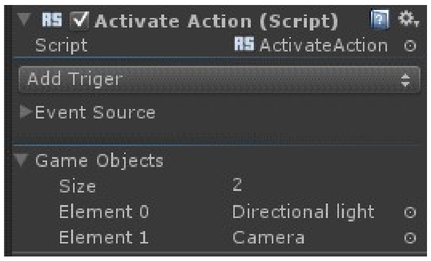
The `Constraints` confines the scaling operation for every axis. You can configure in which axis the object will scale and what are the minimum and maximum values.

When enabled, the `Continuous Scale` option enables continue operation on the game object to grow/shrink in a constant speed (specified in the `Continuous Scale Speed` parameter) as long as the `Scale Trigger` is fired.

The `Dumping Factor` reduces the speed of the scaling operation with respect to the data from the trigger.

3.2.1.5 Activate/Deactivate Actions

Activate Action



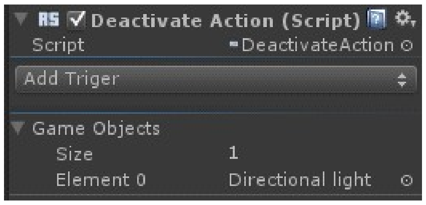
Activate Action (Script)

Script	ActivateAction
Add Triger	
Event Source	
Game Objects	
Size	2
Element 0	Directional light
Element 1	Camera

The `Activate Action` script activates a list of game objects whenever the associated trigger(s) are fired. This action is a `One-shot Custom Trigger Action`.

The `Game Objects` list contains the game objects that the action will activate when triggered. Game objects can be added by simple drag and drop.

Deactivate Action



Deactivate Action (Script)

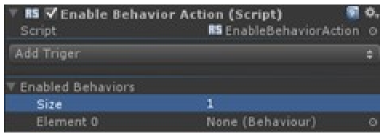
Script	DeactivateAction
Add Triger	
Game Objects	
Size	1
Element 0	Directional light

The `Deactivate Action` script deactivates a list of game objects whenever the associated trigger(s) are fired. This action is a `One-shot Custom Trigger Action`.

The `Game Objects` list contains the game objects that the action will deactivate when triggered. Game objects can be added by simple drag and drop.

3.2.1.6 Enable/Disable Behavior Actions

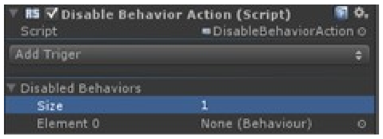
Enable Behavior Action



```
Enable Behavior Action (Script)
Script            EnableBehaviorAction
Add Triger
Enabled Behaviors
  Size            1
  Element 0       None (Behaviour)
```

The `Enable Behavior` action enables a list of Unity behaviors whenever the associated trigger(s) are fired. This action is a one-shot custom trigger action. The behaviors list contains the behaviors that the action will enable when triggered. Behaviors can be added by drag and drop.

Disable Behavior Action



```
Disable Behavior Action (Script)
Script            DisableBehaviorAction
Add Triger
Disabled Behaviors
  Size            1
  Element 0       None (Behaviour)
```

The `Disable Behavior` action disables a list of Unity behaviors whenever the associated trigger(s) are fired. This action is a one-shot custom trigger action. The behaviors list contains the behaviors that the action will deactivate when triggered. Behaviors can be added by drag and drop.

3.2.1.7 Show/Hide Actions

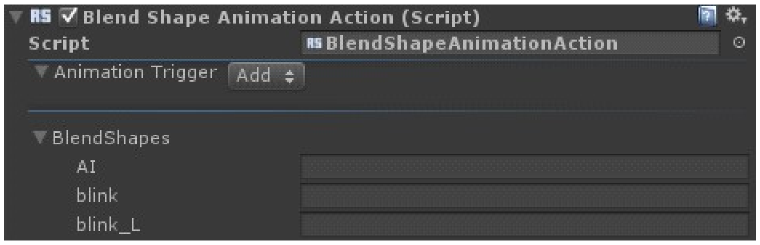
Show Action

The `Show` action enables the renderer whenever the associated trigger(s) are fired. This action is a one-shot custom trigger action.

Hide Action

The `Hide` action disables the renderer whenever the associated trigger(s) are fired. This action is a one-shot custom trigger action.

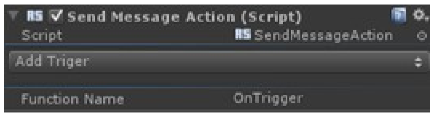
3.2.1.8 Blend Shape Animation


Blend Shape Animation Action (Script)
Script BlendShapeAnimationAction
Animation Trigger Add
BlendShapes
AI
blink
blink_L

The `Blend Shape Animation` action couples Blend shapes to animations based on `Animation Trigger`. When you add the action to a `Game Object` with Blend shapes, the action configuration shows the list of Blend shapes and the associated animation.

This action requires Unity 4.3.0 or higher.

3.2.1.9 Send Message Action



The `Send Message` action uses Unity's "`Send Message`" capability and calls the function listed in "`Function Name`" with the selected trigger as the argument. Note that this function must exist in at least one script that is attached to the same `Game Object`. This action is a one-shot custom trigger action.

3.3 Triggers and Rules

`Triggers` define the user interactions. They hold relevant data about the interactions. In some actions, the triggers are preset and you can only configure the rules attached to each trigger. This is true for actions where the interaction is tightly coded with a specific trigger such as `Rotate` action or `Tracking` action. In other actions, you can add and remove triggers to the action. That is true for more generic actions such as the `Hide` action or the `Enable Behavior` action.

The `Rules` are specific SDK code that implements certain Unity events, actions and data types. You can use rules in one or multiple triggers. When a rule is part of a trigger, the rule and its configuration parameters are shown in the Unity Editor's UI under the specific trigger(s).

When attaching more than one rule to a trigger, the rules are used in an OR relationship. This means that if one of the rules fires, the trigger is triggered with the firing rule's values. The order of which they display in the Unity Editor's UI is the order of priority. Assume a trigger is consist of `Face Detected` and `Gesture Detected` rules, if face is detected, the trigger is fired with the face detection values and the gesture detected rule is ignored, as illustrated in Figure 13.

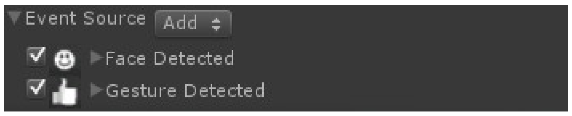


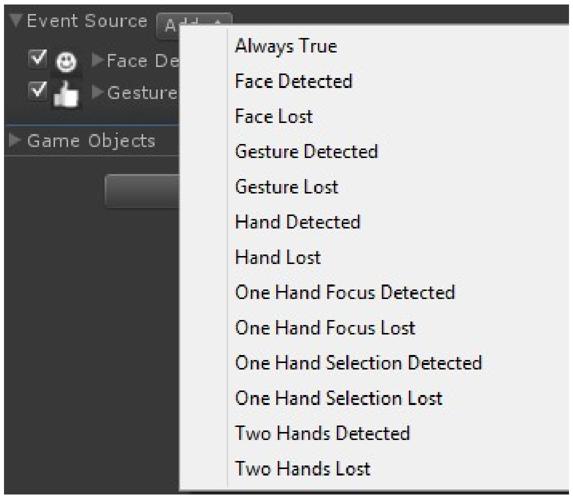
Figure 13: Rules and The Rules Order

3.3.1 Event Trigger and Rules

Event Trigger and the Trigger Values

The `Event Trigger` indicates that an event has been detected. It can be a gesture, a face, and so on.

Data Type	Parameter	Description
`string`	`Source`	Indicate what was detected



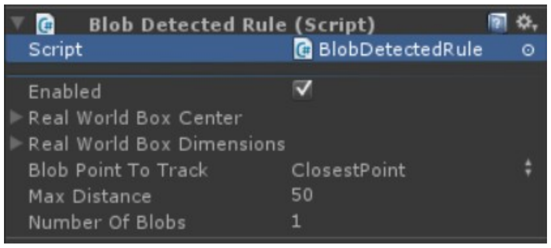
Event Trigger Rules

You can use the following rules with the `Event Trigger`:

- **Always True**

This rule always fires and is used for debug purposes.

- **Blob Detected**

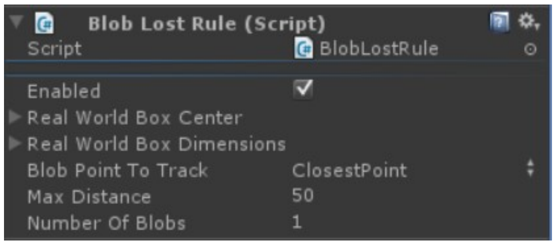

Blob Detected Rule (Script)

Script	BlobDetectedRule
Enabled	✓
Real World Box Center	
Real World Box Dimensions	
Blob Point To Track	ClosestPoint
Max Distance	50
Number Of Blobs	1

The rule fires whenever a blob has been detected in the specified `Real World Box`.

You can change the blob maximum distance and the number of blobs to be detected.

- **Blob Lost**


Blob Lost Rule (Script)

Script	BlobLostRule
Enabled	✓
Real World Box Center	
Real World Box Dimensions	
Blob Point To Track	ClosestPoint
Max Distance	50
Number Of Blobs	1

The rule fires whenever a blob has been previously detected in the specified `Real World Box`, and has now been lost.

You can change the blob maximum distance and the number of blobs to be detected.

- **Face Detected**

Face Detected

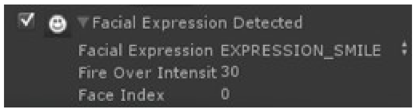
The rule fires whenever a new face has been detected.

- **Face Lost**

Face Lost

The rule fires whenever a face previously detected has been lost.

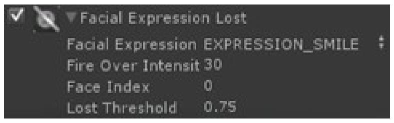
- **Facial Expression Detected**



Facial Expression Detected
Facial Expression EXPRESSION_SMILE
Fire Over Intensit 30
Face Index 0

The rule fires whenever a new facial expression has been detected. The variable `Facial Expression` specifies which facial expression the rule will fire. The parameter `Fire Over Intensity` configures the minimal expression intensity over which the rule will fire, in the range of [0, 100]. Finally, the variable `Face Index` sets the face that will be considered in case of several faces visible to the camera. The index is provided in the ascending order of face appearance.

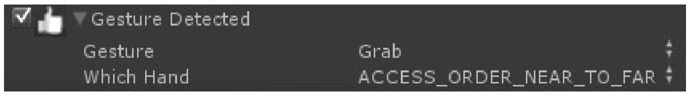
- **Facial Expression Lost**



Facial Expression Lost
Facial Expression EXPRESSION_SMILE
Fire Over Intensit 30
Face Index 0
Lost Threshold 0.75

The rule fires whenever a facial expression that was previously detected has been lost. The field `Facial Expression` specifies the facial expression the rule will fire. The parameter `Fire Over Intensity` configures the minimal expression intensity over which the rule will fire, in the range of [0, 100]. The variable `Face Index` sets the face index if there are multiple faces visible to the camera. Finally, the `Lost Threshold` is used for robustness and controls how long a facial expression needs to be unrecognized to fire the rule.

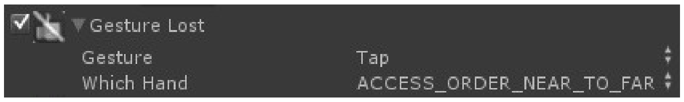
- **Gesture Detected**



Gesture Detected
Gesture Grab
Which Hand ACCESS_ORDER_NEAR_TO_FAR

The rule fires whenever the specified gesture is recognized. You can configure the gesture to be detected from a list of supported gestures and select which hand the gesture belongs to.

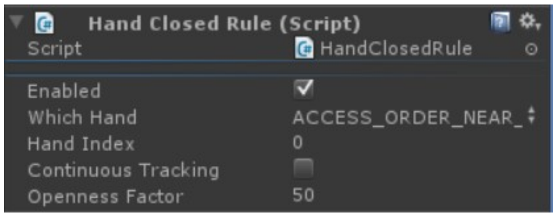
- **Gesture Lost**



Gesture Lost
Gesture: Tap
Which Hand: ACCESS_ORDER_NEAR_TO_FAR

The rule fires whenever the specified gesture is recognized and then lost. You can change the gesture from a list of supported gestures and select which hand the gesture belongs to.

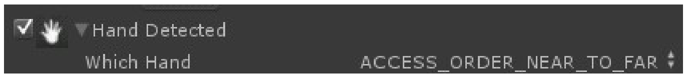
- **Hand Closed**



Hand Closed Rule (Script)
Script: HandClosedRule
Enabled: ✓
Which Hand: ACCESS_ORDER_NEAR_
Hand Index: 0
Continuous Tracking: ☐
Openness Factor: 50

The rule fires whenever the specified hand's openness is lower than the `Openness Factor` value.

- **Hand Detected**

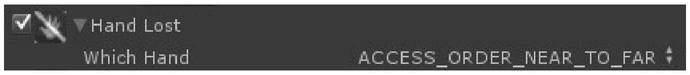


Hand Detected
Which Hand: ACCESS_ORDER_NEAR_TO_FAR

The rule fires whenever a hand has been detected. You can configure which hand to detected using the drop-down list.

Continuous tracking overrides the hand index and tracks the same hand regardless whether other hands show or disappear during the run. For example, a right hand (`index=0`) is visible to the camera and is in tracking. The index may be changed to other values. But the tracking will continue on the same right hand.

- **Hand Lost**

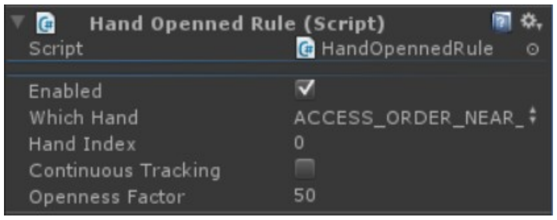


Hand Lost
Which Hand: ACCESS_ORDER_NEAR_TO_FAR

The rule fires whenever a hand previously detected has been lost. You can change which hand to detected using the drop-down list.

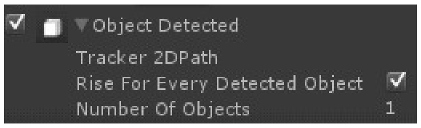
Continuous tracking overrides the hand index and tracks the same hand regardless whether other hands show or disappear during the run. For example, a right hand (`index=0`) is visible to the camera and is in tracking. The index may be changed to other values. But the tracking will continue on the same right hand.

- **Hand Opened**


Hand Openned Rule (Script)
Script HandOpennedRule
Enabled ✓
Which Hand ACCESS_ORDER_NEAR_
Hand Index 0
Continuous Tracking
Openness Factor 50

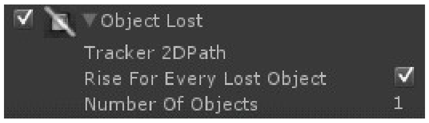
The rule fires whenever the specified hand's openness is higher than the `Openness Factor` value.

- **Object Detected**


✓ Object Detected
Tracker 2DPath
Rise For Every Detected Object ✓
Number Of Objects 1

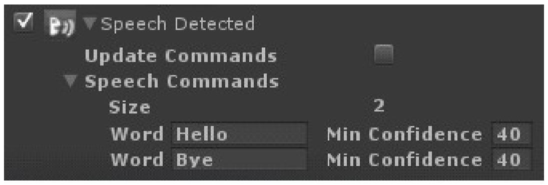
The rule fires whenever an object has been detected. The developer can provide a path to the 2D image that is being tracked. If `Tracker 2DPath` is empty, it will fire on every object detected. `Number of Objects` configures the number of objects the rule will fire in the case of multiple objects visible to the camera. The rule will fire for any object if `Rise For Every Detected Object` is checked.

- **Object Lost**


✓ Object Lost
Tracker 2DPath
Rise For Every Lost Object ✓
Number Of Objects 1

The rule fires whenever an object that was previously detected has been lost. The configuration is similar to the `Object Detected` rule.

- **Speech Detected**


✓ Speech Detected
Update Commands
Speech Commands
Size 2
Word Hello Min Confidence 40
Word Bye Min Confidence 40

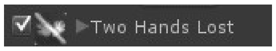
The rule fires whenever a speech command or phrase is recognized. You can provide a list of commands and set the minimum confidence scores, range from 0 to 100. Click `Update Commands` to change settings during playback.

- **Two Hands Detected**

✓ Two Hands Detected

The rule fires whenever both of the user's hands have been detected.

- **Two Hands Lost**



The rule fires whenever the user's hands previously detected are now lost.

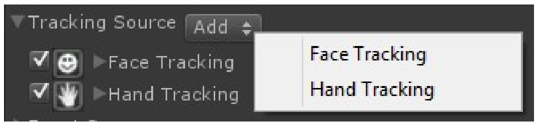
3.3.2 Tracking Trigger and Rules

Tracking Trigger and Trigger Values

The `Tracking Trigger` indicates that the tracking source (hand/joint) has been detected and moved from the last position.

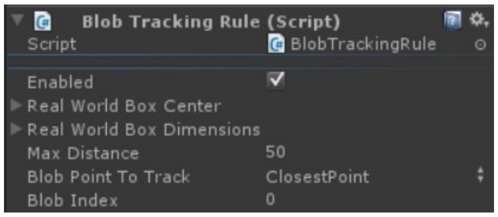
Data Type	Parameter	Description
`Quaternion`	`RotationQuaternion`	The rotation quaternion that the tracked object is in.
`Vector3`	`Position`	The normalized position of the tracked object.

Tracking Trigger Rules



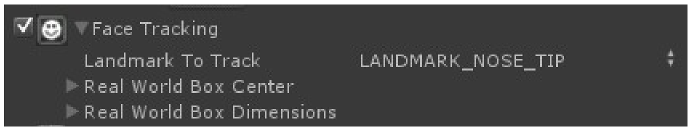
You can use the following rules with the `Tracking Trigger`:

- **Blob Tracking**


Blob Tracking Rule (Script)
Script BlobTrackingRule
Enabled ✓
Real World Box Center
Real World Box Dimensions
Max Distance 50
Blob Point To Track ClosestPoint
Blob Index 0

The rule fires whenever a blob has moved. You can set the maximum distance (in cm) for the blob detection and the point to track on the blob.

- **Face Tracking**


Face Tracking
Landmark To Track LANDMARK_NOSE_TIP
Real World Box Center
Real World Box Dimensions

The rule fires whenever the selected landmark has moved. You can configure the tracked landmark. You can additionally define the `Real World Box`, which sets the boundaries in which the selected face is tracked in the real world. For example, if the `Real World Box` center is set to `X = 0, Y = 0, Z = 50`, the center of this box is set to 50 centimeters away from the camera and on its axis. If the `Real World Box` dimensions are set to `X = 20, Y = 70, Z = 30`, the bounding box's size is 20cm in each direction, as illustrated in Figure 14.

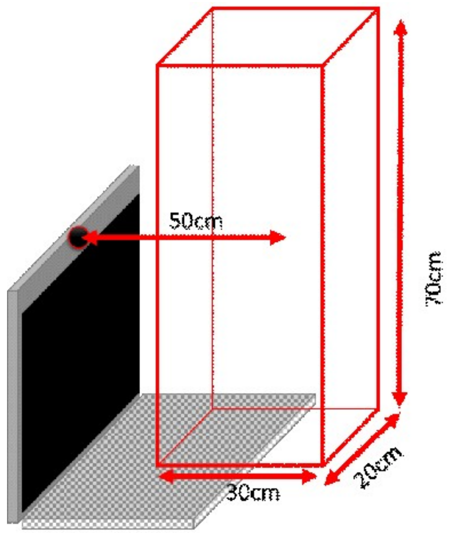
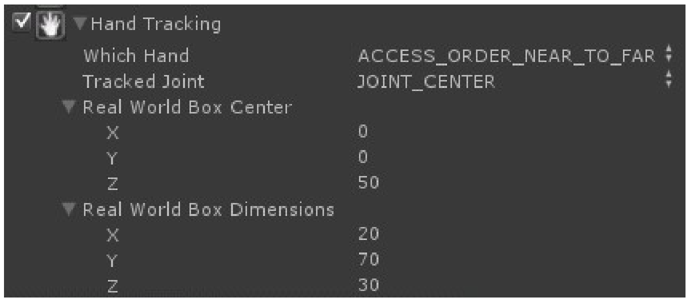


Figure 14 The Real World Box for the Face Tracking Rule

- **Hand Tracking**

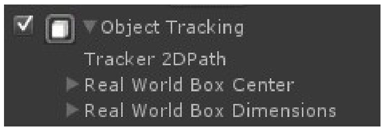


Hand Tracking

Which Hand	ACCESS_ORDER_NEAR_TO_FAR
Tracked Joint	JOINT_CENTER
Real World Box Center	
X	0
Y	0
Z	50
Real World Box Dimensions	
X	20
Y	70
Z	30

The rule fires whenever the selected joint of the selected hand's has moved. You can configure the tracked hand and the tracked joint by choosing from the drop-down lists. Similarly to the `Face Tracking` rule, you can define the `Real World Box` for hand tracking.

- **Object Tracking**



Object Tracking
Tracker 2DPath
Real World Box Center
Real World Box Dimensions

The rule fires whenever the selected 2D object has moved. You can configure the 2D reference image that is being tracked. Similarly to the `Hand Tracking` rule, you can configure the `Real World Box`.

3.3.3 Rotation Trigger and Rules

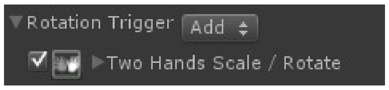
Rotation Trigger and Trigger Values

The `Rotation Trigger` indicates that the user performed a rotation interaction.

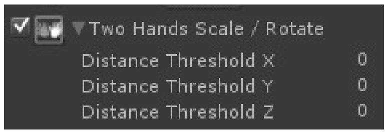
Data Type	Parameters	Description
`float`	`Pitch`	The angle around the X axis to rotate.
`float`	`Yaw`	The angle around the Y axis to rotate.
`float`	`Roll`	The angle around the Z axis to rotate.

Rotation Trigger Rules

You can use the following rules with the `Rotation Trigger`:


Rotation Trigger Add
Two Hands Scale / Rotate

- **Two Hands Rotate**


Two Hands Scale / Rotate
Distance Threshold X 0
Distance Threshold Y 0
Distance Threshold Z 0

The rule fires whenever the user's two hands are performing a rotation gesture. This gesture is defined by moving the hands in 3D in the following manner:

- To rotate the object around the Z axis, the user should move his hands up and down in opposite directions, as illustrated in Figure 15.

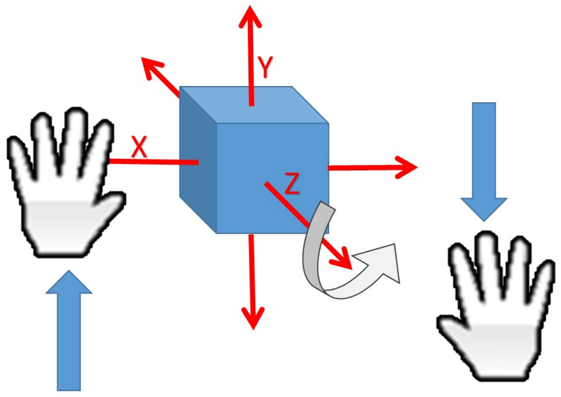


Figure 15: Move Object Around the Z Axis

- To rotate the object around the Y axis, the user should move his hands toward and apart from each other, on the Z axis, as illustrated in Figure 16.

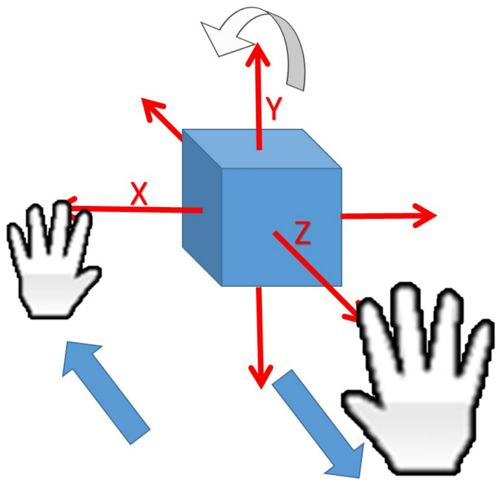


Figure 16: Move Object Around the Y Axis

You can specify the minimal distance the hands need to move from their initial location where they were detected to start the rotation. If the `Distance Threshold` is set to zero, the rotation will start immediately. Otherwise, it will start only when the distance between the hands passes the specified threshold.

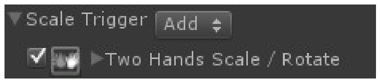
3.3.4 Scale Trigger and Rules

Scale Trigger and Trigger Values

The `Scale Trigger` indicates that the user performed a scale interaction.

Data Type	Parameters	Description
`float`	`Scale`	The scaling factor.

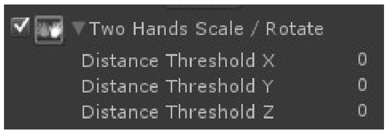
Scale Trigger Rules



Scale Trigger Add
Two Hands Scale / Rotate

The Scale Trigger can work with the following rules:

- **Two Hands Scale**



Two Hands Scale / Rotate

Distance Threshold X	0
Distance Threshold Y	0
Distance Threshold Z	0

The rule fires whenever the user's two hands are performing a scale gesture. This gesture is defined by moving the hands in 3D in the following manner:

o To up-scale the object, the user should move his hands apart, as illustrated in Figure 17.

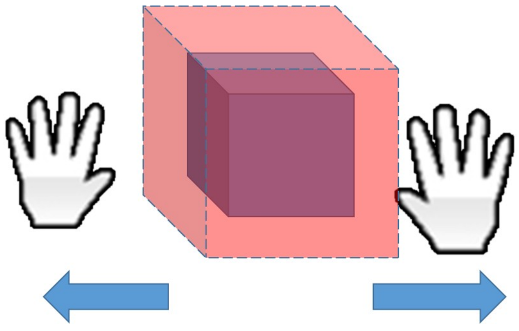


Figure 17: Move Hands Apart to Scale Up

o To down-scale the object, the user should move his hands closer to each other, as illustrated in Figure 18.

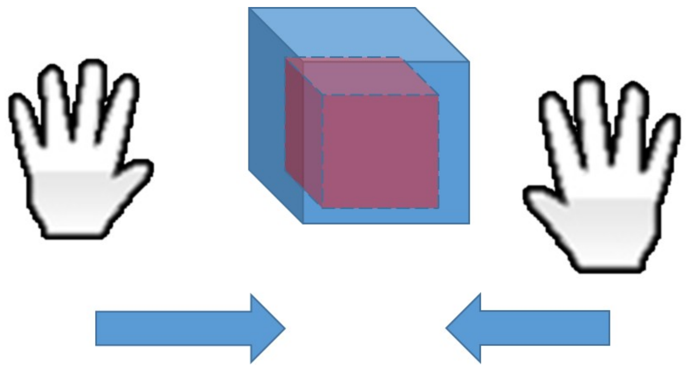


Figure 18: Move Hands Closer to Scale Down

You can specify the minimal distance that the hands need to move from their initial location where they were detected to start the gesture. If the `Distance Threshold` is set to zero, the scaling will start immediately. Otherwise, it will start only when the distance between the hands passes the selected threshold.

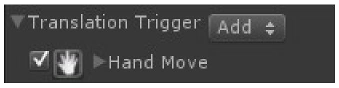
3.3.5 Translation Trigger and Rules

Translation Trigger and Trigger Values

The `Translation Trigger` indicates that the user performed a translation interaction.

Data Type	Parameters	Description
`Vector3`	`Translation`	The amount of movement of the tracked object.

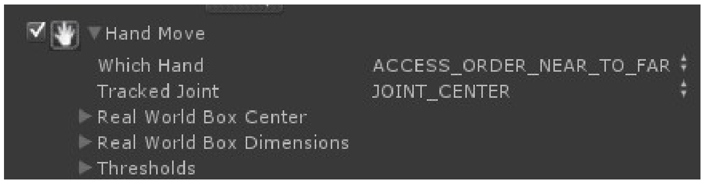
Translation Trigger Rules



Translation Trigger Add
Hand Move

The following rules work with the `Translation Trigger`:

- **Hand Move**



Hand Move
Which Hand ACCESS_ORDER_NEAR_TO_FAR
Tracked Joint JOINT_CENTER
Real World Box Center
Real World Box Dimensions
Thresholds

The rule fires whenever the user's hand moves. Similarly as in the `Hand Tracking` rule, you can specify the tracked hand and joint together with the `Real World Box`. In addition, the developer can specify the `Thresholds` which define the minimal movement that the user needs to move his joint in order to fire this rule. The thresholds help to prevent false movement.

Continuous tracking overrides the hand index and tracks the same hand regardless whether other hands show or disappear during the run. For example, a right hand (`index=0`) is visible to the camera and is in tracking. The index may be changed to other values. But the tracking will continue on the same right hand.

3.3.6 Animation Trigger and Rules

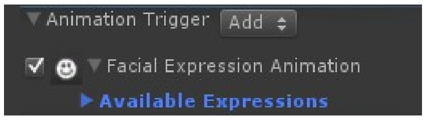
Animation Trigger and Trigger Values

The `Animation Trigger` indicates that there was an animation.

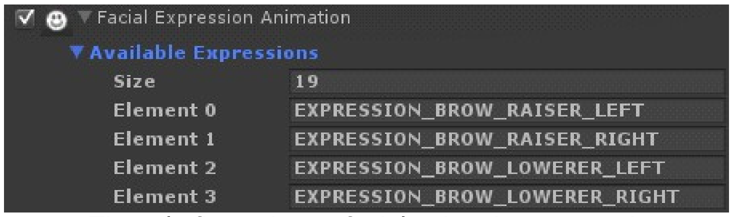
Data Type	Parameters	Description
`Dictionary<String,Single>`	`Animations`	The list of animations and intensities.
`bool`	`IsAnimationDataValid`	Indicates a valid animation.

Animation Trigger Rules

The following rules work with the `Animation Trigger`:



- **Facial Expression Animation Rule**


Facial Expression Animation

Available Expressions

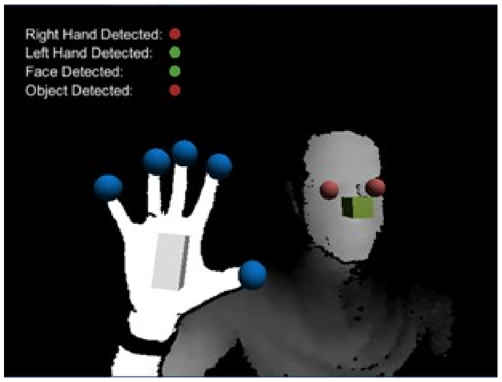
Size	19
Element 0	EXPRESSION_BROW_RAISER_LEFT
Element 1	EXPRESSION_BROW_RAISER_RIGHT
Element 2	EXPRESSION_BROW_LOWERER_LEFT
Element 3	EXPRESSION_BROW_LOWERER_RIGHT

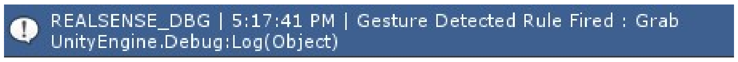
The rule fires when a facial animation expression was detected. Under `Available Expressions`, the rule lists available expressions and their intensities.

3.4 Prefabs

The Prefabs are provided for convenience. They are under `$(RSSDK_DIR)/framework/Unity/RSUnityToolkit/Prefabs`.

Debug Viewer

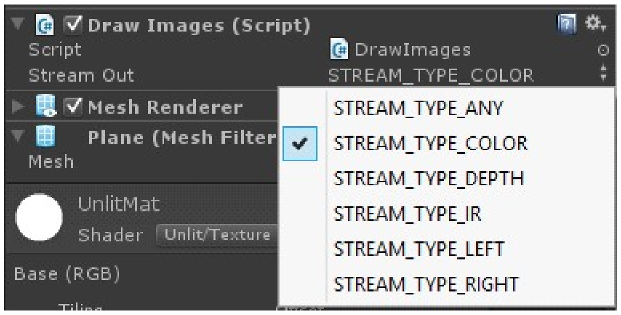




```
REALSENSE_DBG | 5:17:41 PM | Gesture Detected Rule Fired : Grab
UnityEngine.Debug:Log(Object)
```

The `Debug Viewer` prefab is provided for debugging purposes. It prints indicators of face, hands and object detection to the Unity log.

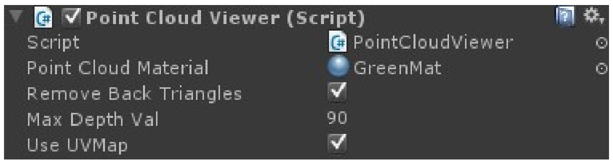
Image



Draw Images (Script)
Script: DrawImages
Stream Out: STREAM_TYPE_COLOR
Mesh Renderer
Plane (Mesh Filter
Mesh
UnlitMat
Shader: Unlit/Texture
Base (RGB)

The `Image` prefab renders any stream of choice, by the `Stream Out` field.

Point Cloud Mesh

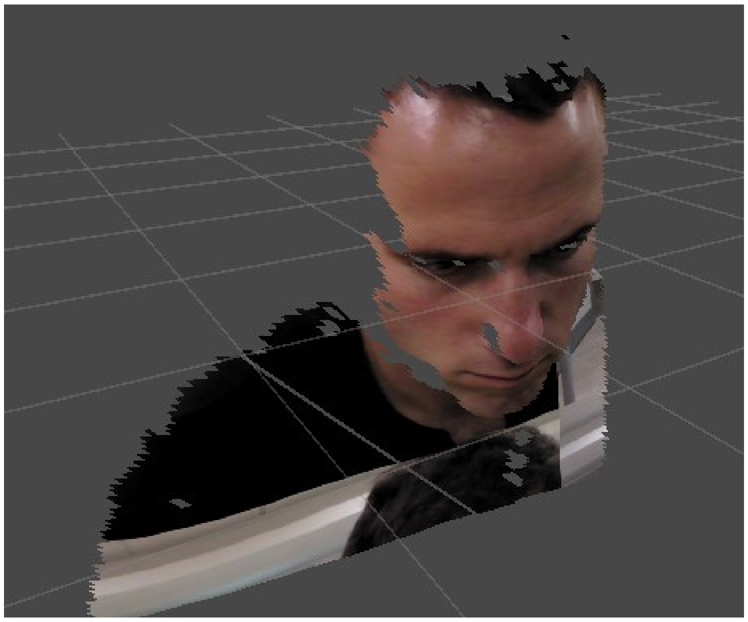


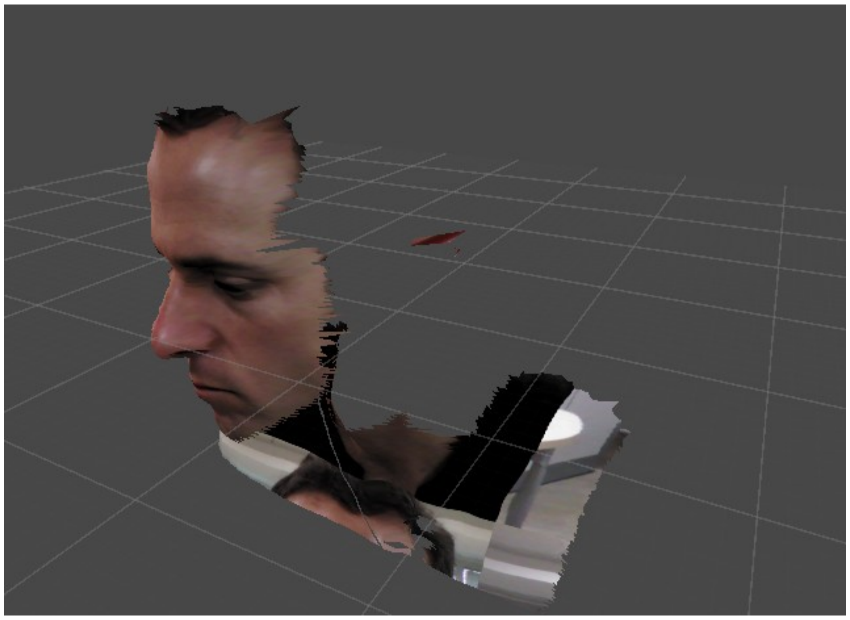
Point Cloud Viewer (Script)
Script: PointCloudViewer
Point Cloud Material: GreenMat
Remove Back Triangles: ✓
Max Depth Val: 90
Use UVMap: ✓

The `Point Cloud Mesh` renders the camera depth data as a 3D mesh.

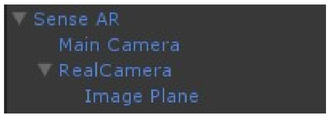
You can set different materials on the 3D mesh. If `Use UVMap` is checked, the RGB data will be drawn on the selected material.

The developer can check the `Remove Back Triangles` option to remove saturated points indicated by the depth value bigger than the value specified in the `Max Depth Value` field.





Sense AR

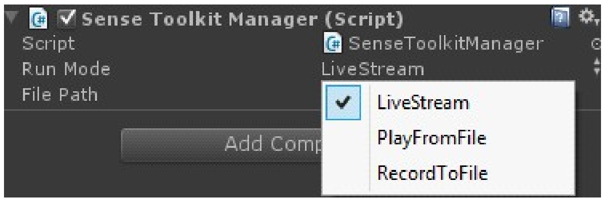


▼ Sense AR
  Main Camera
  ▼ RealCamera
    Image Plane

The `Sense AR` prefab is provided for augmented reality purposes. It consists two cameras – Main and Real.

The real camera displays the color stream from the camera, whereas the main camera has a higher Z value and thus displays the Unity objects over the color stream.

Sense Manager


Sense Toolkit Manager (Script)

Script: SenseToolkitManager

Run Mode: LiveStream

File Path

The `Sense Manager` prefab is the manager of the toolkit. The prefab will be added to the scene if any action is used. The prefab provides living streaming, recording and playback modes, based on the `Run Mode` setting. For recording and playback, the developer must configure the `File Path` field.