Ultrahaptics Core Asset 1.0.0-beta9 Release Notes & Known Issues

Release Notes

Ultrahaptics Core Asset 1.0.0-beta9 includes an Emitter Log file option, Improvements to the UltrahapticsKit prefab and visualisation of the Interaction zones user can expect from different arrays.

1 System Requirements

- Unity 2019.1

For **hand tracking** via the Leap Motion Controller:

- Leap Motion Orion 4.0.0 SDK
- Leap Motion Unity Core Assets 4.4.0

The following **Ultrahaptics Hardware** is supported:

- STRATOS Inspire (USI)
- STRATOS Explore (USX)

2 What's New

2.1 New Features and Enhancements

- **UCA-137 -** The sensation Emitter no longer produces an error when using the 'Allow Mock Emitter' option.
- **UCA-62 -** Users can now decide if the mock emitter log file is created or not. (Defaults to not create the file).
- **UCA-47 -** The software now detects the connected device and automatically changes the tracking origin to suit, meaning that users no longer have to manually modify a scene to suit different UH kit.
- **UCA-40 –** The Unity transform can now be used to define an SDK device transform without the need for modification.
- **UCA-38 –** The software now shows the different levels of interaction zones users can expect from different arrays.

2.2 Bug Fixes

- **UCA-10 –** The hardware info window now detects the connected UH device without the need to refresh.

2.3 Sensations and Examples

The following Sensation producing Blocks and Example Scenes are included with UCA 1.0.0-beta9 in the StreamingAssets/Python directory (new items for this release included in green):

- **CircleSensation**
- **LineSensation**
- **ExpandingCircleSensation**
- **DialSensation**
- **FingerPatch** (Requires Leap Motion Controller + **Leap Data Source** component)
- **Forcefield** (Requires Leap Motion Controller + **Leap Data Source** component)
- **HandScan** (Requires Leap Motion Controller + **Leap Data Source** component)
- **LissajousSensation**
- **PalmTrackedCircle** (Requires Leap Motion Controller + **Leap Data Source** component)
- **PalmTrackedDial** (Requires Leap Motion Controller + **Leap Data Source** component)
- **PalmPresence** (Requires Leap Motion Controller + **Leap Data Source** component)
- **PalmTrackedPulsingCircle** (Requires Leap Motion Controller + **Leap Data Source** component)
- **Point** – a Sensation for test purposes, outputs a control point at 20cm above the array.
- **Polyline6*** - a Sensation which allows you to draw a Polyline Path constructed of 6 points.

*Polyline Sensations of any number of points can be made via the PolylinePath Block. See *StreamingAssets/Python/Polyline6.py* for details on how this is constructed.

Scene Examples are included in the UltrahapticsCoreAsset/Examples folder:

- **Forcefield** – demonstrates the Forcefield Block to produce a virtual 'forcefield' boundary interaction.
- **HandAsCursor** – demonstrates the use of hand tracking to operate a haptic user interface, with palm tracked haptics including Presence, Dial and a 'Click' sensation, provided by the Unity timeline.
- **Polyline** - demonstrates the use of the Polyine6 Sensation Block to create custom haptic shapes.
- **SensationSourcePlayback** – Basic demonstration of playing back Sensations.
- **Priority** - demonstrates the use of Sensation Source's Priority value, for managing multiple simultaneous haptic playback events.
- **HandTriggeredSensation** - Demonstrates the PalmTrackedPulsingCircle Sensation, which tracks the palm of the hand.
- **SensationAnimation** - demonstrates animating the radius input of CircleSensation via C# and Unity's Timeline window.
- **SensationSequencing** – demonstrates sequencing of animating the radius input of Circle Sensation using Unity's Timeline window.

3 Known Issues

3.1 Known Issues and Workarounds

Below is a list of known issues with Ultrahaptics Core Asset 1.0.0-beta9:

- **UCA-12** – Animating the Running property in the Unity Timeline does not currently work as expected.
- **UCA-6** - It is possible for multiple AutoMappers to exist in the scene which can cause inconsistency with Sensation Block inputs
    - **Workaround**: Ensure that your scene includes only one AutoMapper Component.

4 Developer Notes

4.1 - UCA-40 - An easier method to define an SDK Device Transform has been introduced

Before:

```
var tmpTransform = new GameObject().transform;
tmpTransform.gameObject.hideFlags = HideFlags.HideAndDontSave;

// To use Unity Transform directly to represent the device transform we currently have to swap Y-Z
coordinates.
Vector3 rotationVector = new Vector3(transform.eulerAngles.x, transform.eulerAngles.z,
transform.eulerAngles.y);
Vector3 positionVector = new Vector3(transform.localPosition.x, transform.localPosition.z,
transform.localPosition.y);

Quaternion rotation = Quaternion.Euler(rotationVector);
tmpTransform.SetPositionAndRotation(positionVector, rotation);

// Update the Device Transform
SensationCore.Instance.SetDeviceTransform(deviceID, tmpTransform);
Destroy(tmpTransform.gameObject);
```

Now:

```
SensationCore.Instance.SetDeviceTransform(deviceID, transform);
```