NeverEngineLabs.

SendOSC (2024)

`SendOSC` is a tool that takes a list of destination messages in standard OSC syntax and converts them into OSC generating widgets that Kyma can control. These widgets can be used to control a receiving software or device on the same network as the Paca(rana). The outgoing OSC widgets in the VCS will be mirrored and rewritten with your custom naming schemas for use as parameters in the rest of your Kyma sound design.

Basic Example

Original OSC message:

```
'/SrcPos/1/xy,ff'
```

The **`,ff`** means this message has two float numerical arguments. At least one argument is obligatory.

In Kyma default syntax

```
!osc_srcpos_1_xy__2
!osc_srcpos_1_xy__2
```

SendOSC rewrites these messages by default to more meaningful names.

Rewrite to:

```
!helicopterSound.x
!helicopterSound.y
```

⇧

Tip! **Adding .x and .y will automatically create a 2D widget!**

Now you can control sound design and an external spatialisation system with the same Event Expressions all from Kyma with nothing in between. Joy of Joys!

Tips for Happy SendOSC

- Check the correct setting of IP and Port to reach the receiver from Kyma In most common cases, the IP you should use is the IP of the Mac/Windows connection to the Paca. Find it in your system settings.
- Adhere to the OSC Syntax in the **`OSCMessages`** field (see parameter help for **`OSCMessages`**).
- Ensure a signal is associated with each OSC message and each argument (if a message has more than one argument).
- If 'Rewrite' is active, define an **`EventName`** for each message and argument.
- Avoid conflicts when choosing names for **`ExtraGeneratedEvents`**.
- In stealth mode, you can't have a completely empty VCS; keep at least one widget for background OSC to keep streaming.
- Monitor OSC network traffic for debugging and programming (e.g., OSCulator or Max).

Class Design and Programming

Class design and programming by Cristián Vogel for NeverEngineLabs 2024 v24.1.b1. **`SignalsToOSC`** uses code from OSCHelper Tool by Symbolic Sound.
Support hotline: support@neverenginelabs.com

Parameter Guide

Enable

Enable the OSC widgets and outputs. Generally leave this on, but it may be useful for debugging.

SendToIP

Put in the address of the device you wish to send to. The Paca(rana) must be connected to the same network as this IP. In most cases, this is the IP from System Preferences/Network interface that is connected to the Paca APU. Read more about ports, IP, and Kyma here(http://kyma.symbolicsound.com/qa/1895/how-do-i-change-the-sending-port-for-osc).

RewrittenEventNames

In Kyma, outgoing OSC controller messages are transmitted via the Virtual Control Surface, and adhere to the Kyma OSC syntax which looks like this example.
`!osc_virtualSource_pos__1`

Rewriting using SendOSC allows more meaningfully named generated events, which can be easily re-used as parameters throughout your signal flow. This encourages the composer to use values that route to an external system in other aspects of a sound design being simultaneously. For example, using a re-written name for a common spatial parameter like *distance from centre*, means you could implement your own perceptual filtering on Kyma a audio stream which relates to spatial features of a virtual object emitting that audio stream. **This is a powerful workflow.**

Rewrite

When active, duplicates of the Kyma syntax OSC widgets are created using the **`GeneratedEvent`** names defined in the **`ReWrittenEventNames`** field as described above. Nevertheless it is still possible to disable the creation of rewritten VCS widgets to work with raw Kyma OSC syntax widgets.

OutputScaling

An optional array that will scale the outgoing OSC values by any amount. If there are less values here than there are outgoing messages, the last value in the OutputScalling array will be applied to everything following it.

So for example if you had 8 messages you could control the last four all at once with the same Capytalk, like this;

```
▶ 1 1 1 1 {!FadeTopQuad smooth: 3 s}
```

If you had 8 messages and you want to scale the 4th and the 6th, it is required to pad this array. In such case, you could try:

```
▶ 1 1 1 !Scale4 1 !Scale6 1 1
```

alternatively, generate as many as you need

```
▶  ( !Scale copies: 4 )
```

Any Capytalk must be in curlies, like always in Kyma for example:

```
▶ {(1 repeatingRamp: 30s) * 360} { !GoFadeUp smooth: 20 s }
```

And remember, by default you will only observe the scaled signals happening at the OSC receiver, not in the VCS.

InitialWidgetType

Decide what type of Widgets will be created the first time the Sound is run. Each widget will remember its display type after the first time it is created.

1. SmallFader (only numbers)
2. Panpot
3. Gate
4. Fader

SendToPort

Port number to send the signals to. The software or device with the specified IP will receive the Kyma signals through the **OSCMessages** you have defined.

OSCMessages

Enter a list of OSC messages, one on each line, enclosed by single quotes. Each message must be completed by a comma followed by the number of arguments in the message.

Here are some examples;

```
'/cursor/1/xy/,ff'

 "use Smalltalk to splice something at build time"
'/myObject/'&?VoiceNumber&'/xyz/,fff'
```

Signals

Array of CapyTalk, hotvalues, constants or pasted Sounds that will be transmitted as each OSC message value. Enclose Capytalk expressions within curly braces.

PollingTrigger

Messages are sent only when this trigger changes from 0 to positive. Use it to reduce data over the network if necessary, or perhaps to send some initialisation settings.

Smooth

Convenient smooth on all outgoing signals with a duration in seconds.

StealthMode

Hide all OSC controlling generated widgets from the VCS. Continue broadcasting values to the **`RewrittenEventNames`** specified and transparently to the specified OSC.