**Communication in Processing and Supercollider: Oscmessages, net address**

OSC stands for [Open Sound Control](https://en.wikipedia.org/wiki/Open_Sound_Control), and consists in a protocol for networking between computers, synths and various multimedia devices.

For instance, it allows a software to communicate with a hardware synth, whenever the latter supports OSC.

One of the most important, or rather most useful difference, though, is that OSC allows to send any type of messages at high resolution to any address.

Differently, the MIDI protocol has its own specific messages, like note On, not Off, pitch, etc., with low resolution.

This means that if you use MIDI to communicate between devices, you’ll be required to translate your original message, say for instance the position of a particle or the color of a pixel at mouse point, via the standard MIDI messages

OSC communication between programs is often done by sending messages from one application to another.

[SuperCollider](http://supercollider.github.io/) has OSC support in its core, as it is based in a client-server model for communicating between the client and the server-side.

On the other hand, [Processing](http://processing.org/) supports OSC communication via the [oscP5 library](http://www.sojamo.de/libraries/oscP5/) which is one of the standard libraries that Processing is shipped.

**Supercollider**

In SuperCollider this communication is done by creating a [NetAddr](https://doc.sccode.org/Classes/NetAddr.html) of the target application and creating an [OSCFunc](https://doc.sccode.org/Classes/OSCFunc.html) to listen to another application.

SoundCollider is a real-time sound-based object oriented programming language for sound synthesis and algorithmic music composition.

To establish the communication to another application (Processing in our case), you need to know on which port that application is listening. We can create a network with NetAddr with the listener.

SuperCollider will listen to messages at the defined port and address.

In a typical configuration server and client are on the same machine and they communicate through the localhost (127.0.0.1), but SuperCollider can be reconfigured to accommodate scenarios where client and server are on different machines.

NetAddr("127.0.0.1",57120);

**Processing**

Processing is a flexible software sketchbook and a language for learning how to code within the context of the visual arts. It is a Java-based language.

In order to establish a communication with Supercollider, we have to import oscP5, which is an OSC implementation for the programming environment processing.

import oscP5.\*;

and some network libraries for processing which supports UDP, TCP and Multicast.

import netP5.\*;

import java.io.File;

import controlP5.\*;

**Anatomy of OSC Messages**

As said before, OSC messages are sent over a network, so you must define where you want to send your message. You must define the following things:

IP Address – The IP address of the device where you send your message. If you want to send data from one software to another on your computer. Usually the localhost IP address is 127.0.0.1

Port – The port number where you are sending your message. This could basically any number, but a lot of ports are reserved for other purposes.

In this case:

NetAddress myRemoteLocation;

...

myRemoteLocation = new NetAddress("127.0.0.1",57120);

myRemoteLocation is a NetAddress that takes the 2 parameters (ip address and port number).

myRemoteLocation is used as parameter in oscP5.send() when sending osc packets to another computer, device or application (Supercollider).

OscMessage myMessage = new OscMessage("/color");

It creates a OscMessage object: the first argument is the OSC Address Pattern, i.e. an OSC string beginning with '/', that specifies the receiver of an OSC packet

After that we add three float numbers to OSC our messages in orde to add some parameters such as saturation, brightness.

myMessage.add(hue);

myMessage.add(sat);

myMessage.add(bri);

At the end we send our OSC message to our address

oscP5.send(myMessage, myRemoteLocation);