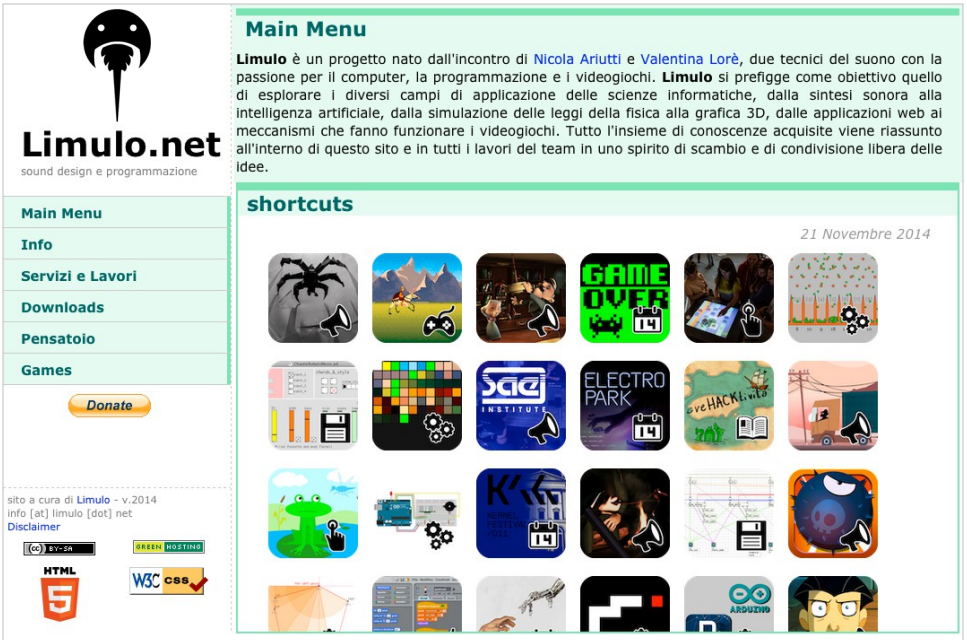


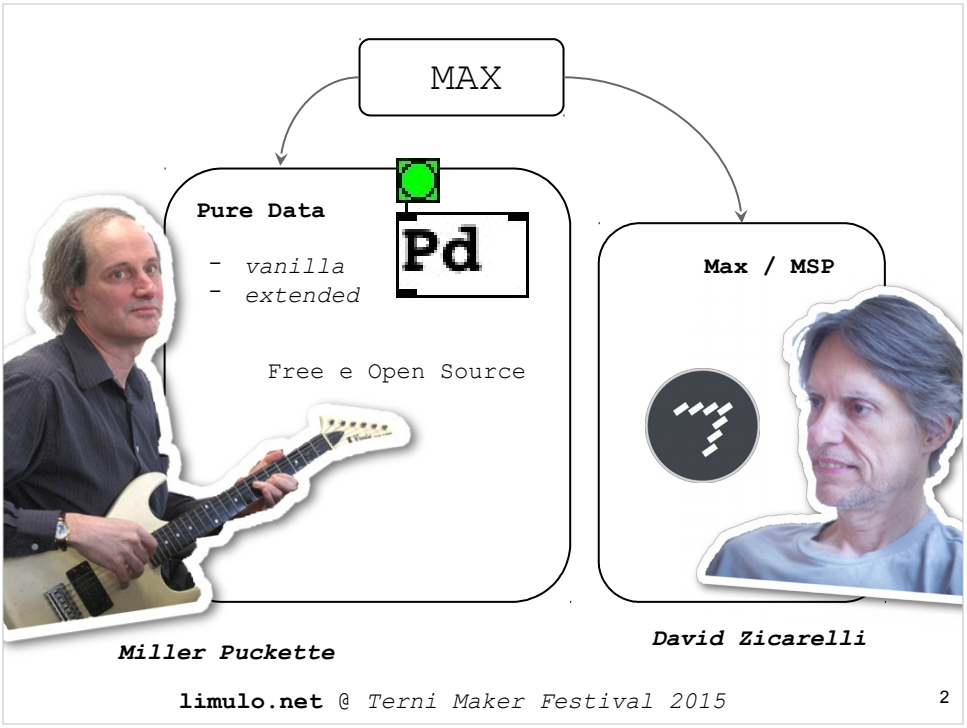
limulo.net @ *Terni Maker Festival 2015* 1

Limulo,
Valentina Lorè e Nicola Ariutti

ci occupiamo di sound design principalmente
interessiamo anche di tutti i campi di **applicazione** delle **scienze informatiche** (AI, videogiochi, simulazioni, 3D)

fatevi un giro sul sito

Il nostro intervento di oggi **mirato** a farvi conoscere un software che è uno dei nostri strumenti di lavoro preferiti
Pure Data



accenni storici

Miller Smith Puckette scrisse Max a metà degli anni ‘80, quando era all’IRCAM, come *Patcher* editor per Macintosh, basandosi sul linguaggio MUSIC di Max Mathwes.

A seguire, alla fine degli anni ‘80, l’IRCAM concesse la licenza di Max alla Opcode, e poi a Zicarelli, che ne fecero un prodotto commerciale.

A metà degli anni ‘90 Puckette riscrisse completamente il software e lo rese libero e open-source.

cosa fa Pure Data?

linguaggio visuale (dataflow programming language) per l’elaborazione del **segnale**

segnale

elettrico → sensori, tasti, video, e suono!

externals

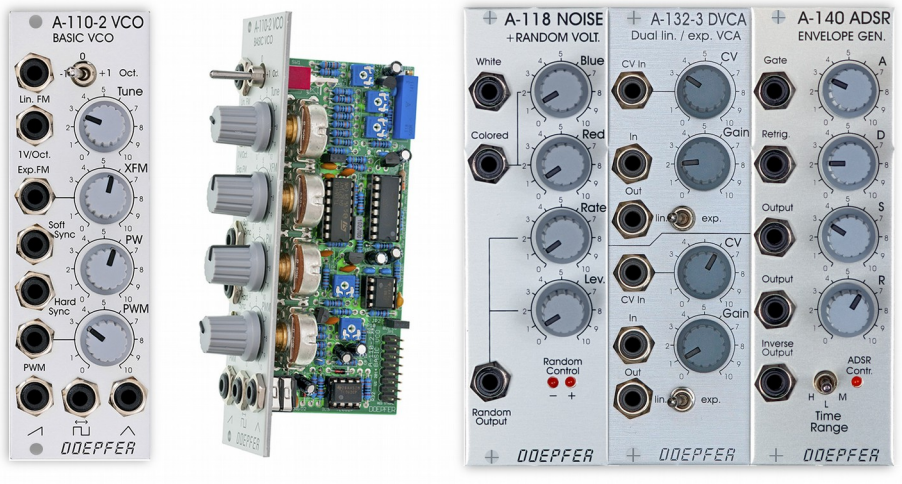
elaborazione video, 3D, interaction design, strumento didattico

le 2 versioni di PD: **vanilla** e **extended**

il suono è una PROPAGAZIONE della PERTURBAZIONE della STATO di QUIETE delle particelle di un MEZZO ELASTICO.

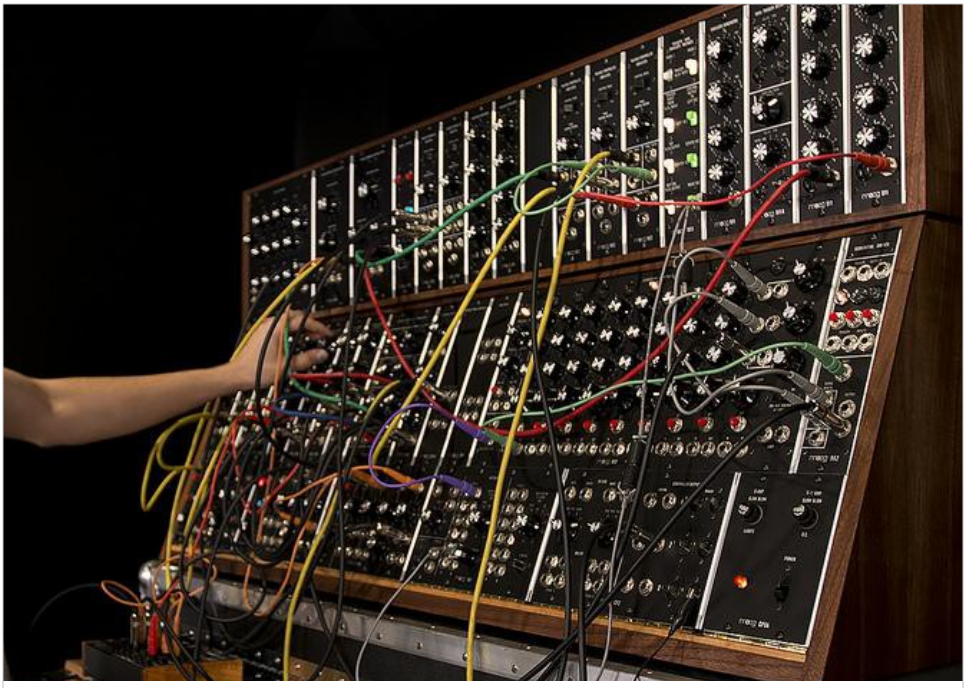
L'ORECCHIO è l'organo che trasduce la vibrazione meccanica in stimoli elettrici interpretati dal cervello come suono.

Il suono può essere generato anche da strumenti artificiali non solo prodotti artigianalmente con materiali naturali (chitarra, liuto, flauti, violini, percussioni, etc…) ma, grazie alla scoperta dell'elettricità e poi allo sviluppo dell'elettronica, con componenti di questo tipo.

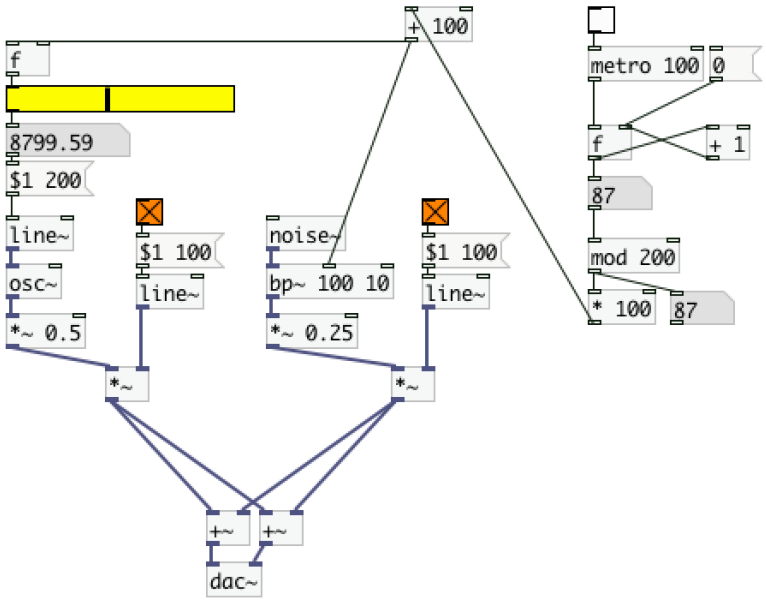


Sintetizzatore Modulare: i moduli sono l'unità minima delal catena di processamente audio.

Interconnessioni tra un modulo e l'altro permettono al segnale audio di subire processamenti successivi.



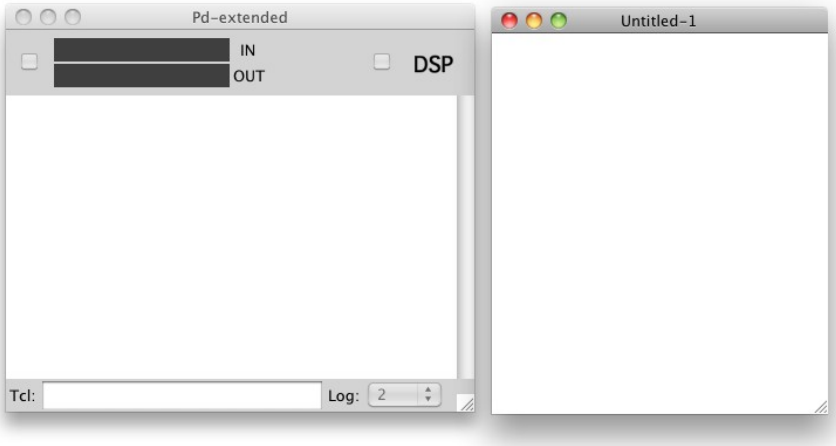
interconnessioni

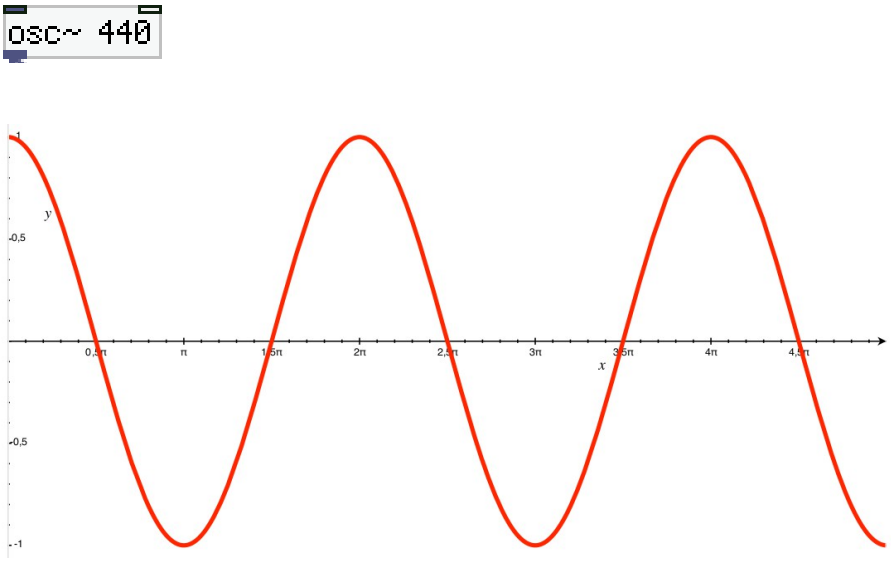


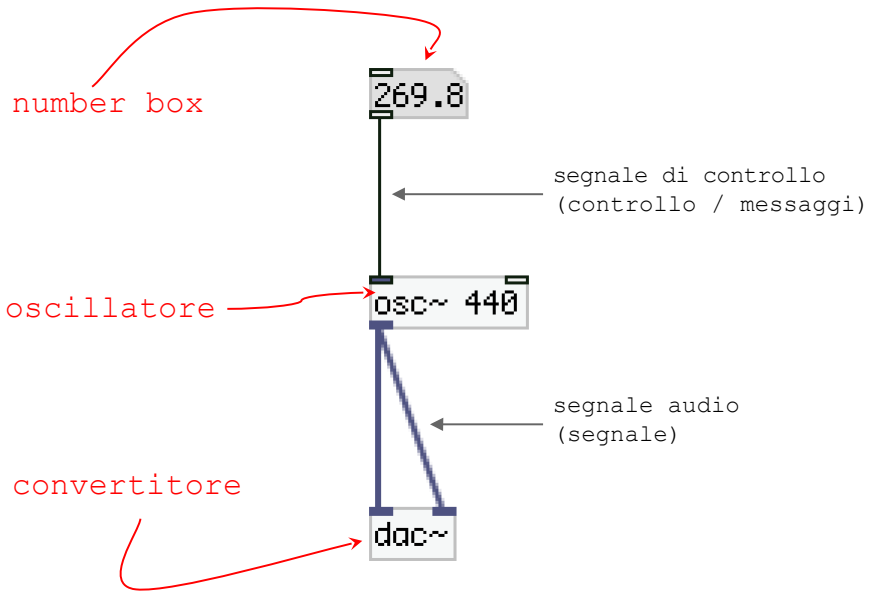
Quando i computer divennereo sufficientemente veloci da consentire calcoliin **tempo reale**, gli strumenti software rirproposero lo stesso principio di interconnessioni tra moduli!

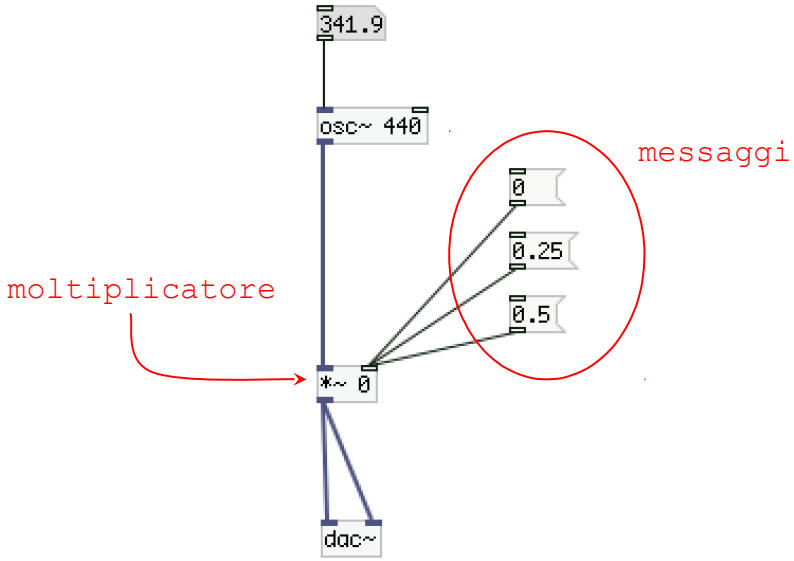
differenze tra **segnale** e **segnale di controllo** (**CV**) → in PD?

L'operatore deve agire fisicamente sui controlli (potenziometri, fader, interruttori o pulsanti) per permettere al modulo di influenzare il sengale elettrico di passaggio. Anche i moduli però possono comportarsi come fossero essi stessi operatori. Operatori che agiscono su altri moduli controllandoli.

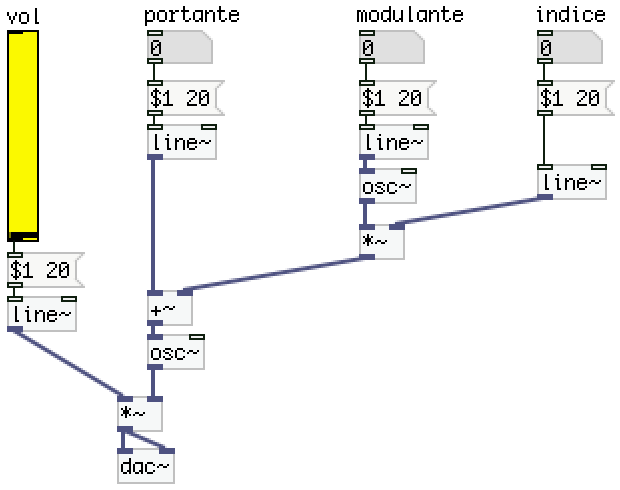

Pd-extended
IN
OUT
DSP
Tcl:
Log:
2
Untitled-1


osc~ 440
1
y
0,5
0,5π
π
1,5π
2π
2,5π
3π
3,5π
4π
4,5π
x
0,5
-1

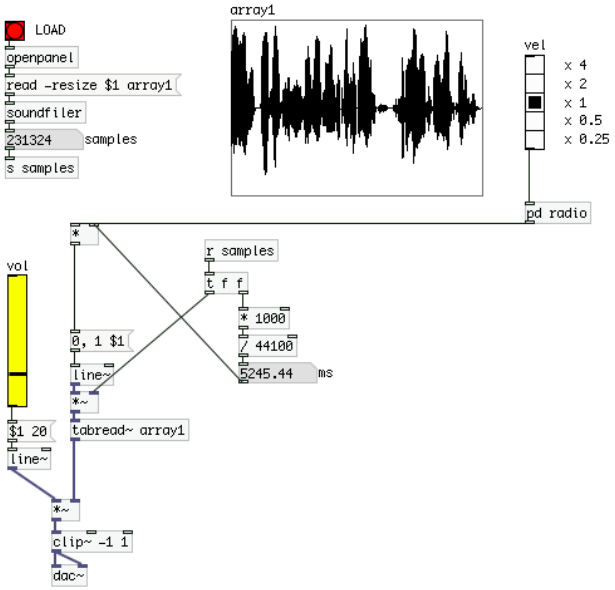

number box
269.8
segnale di controllo
(controllo / messaggi)
oscillatore
osc~ 440
segnale audio
(segnale)
convertitore
dac~


341.9
osc~ 440
messaggi
0
0.25
0.5
moltiplicatore
*~ 0
dac~

Modulazione di frequenza



Campionatore semplice

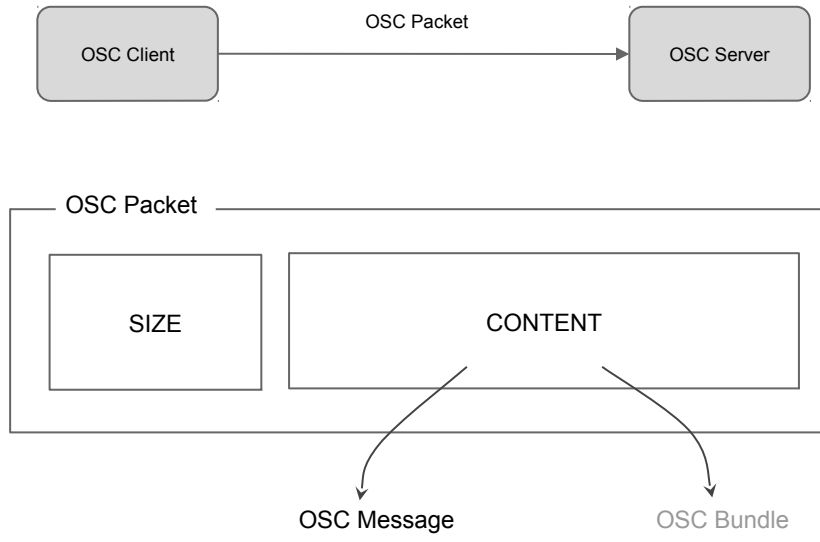


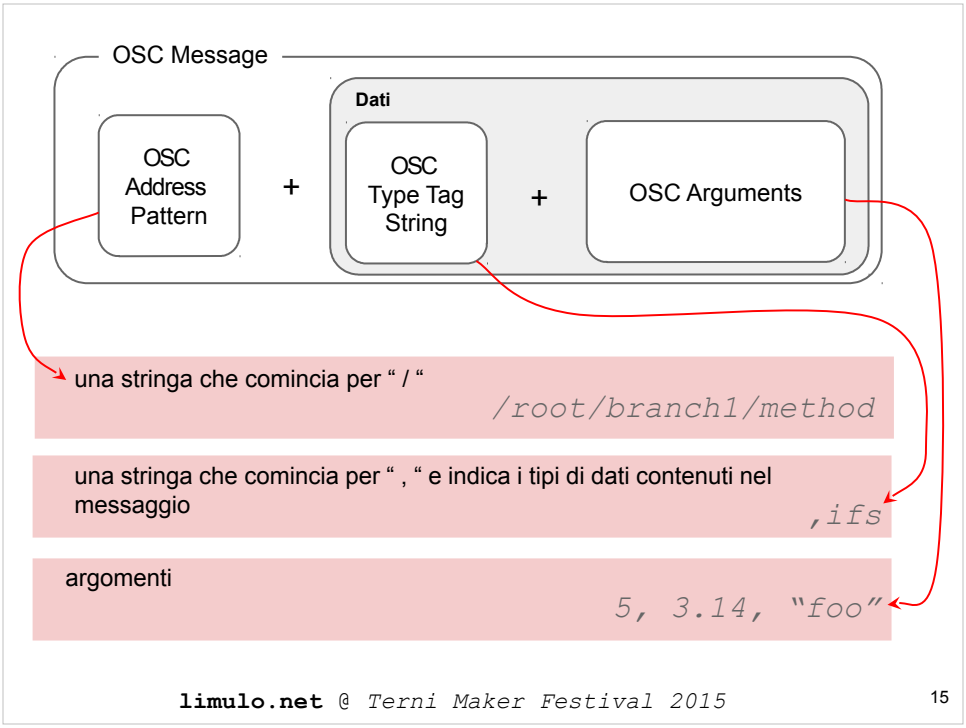
comunicazione tra apparati hardware → CV (anche tra sequencer e synth) ad esempio il Kork Sequencer con MS-20
Tra marche diverse → incompatibilità

poi

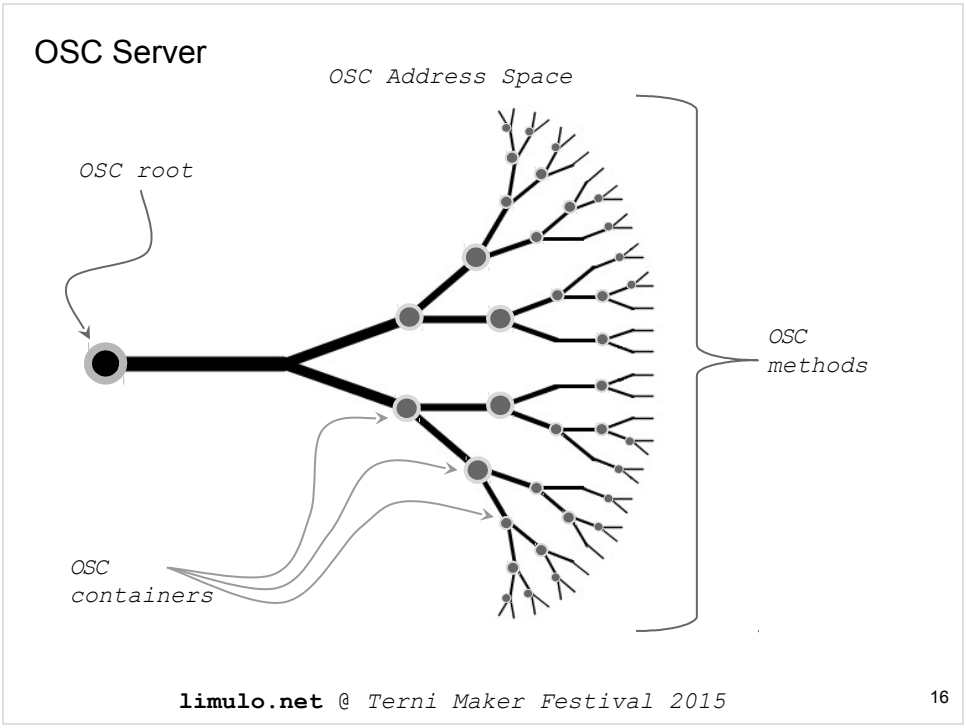
Inizi anni ‘80 nasce il MIDI come accordo tra le maggiori case produttrici di hardware e prodotti per musicisti (SCi, Roland, YAMAHA, KAWAI).
standard di comunicazione tra apparati musicali.

nel 2002 sviluppato il protocollo OSC
protocollo TCP e UDP

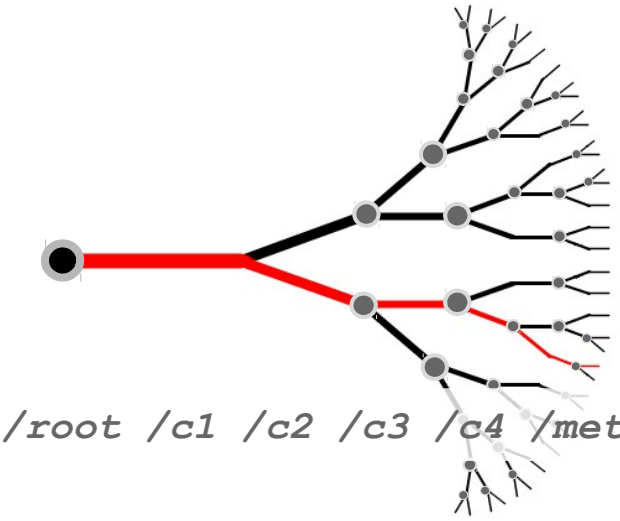

OSC Client
OSC Packet
OSC Server
OSC Packet
SIZE
CONTENT
OSC Message
OSC Bundle



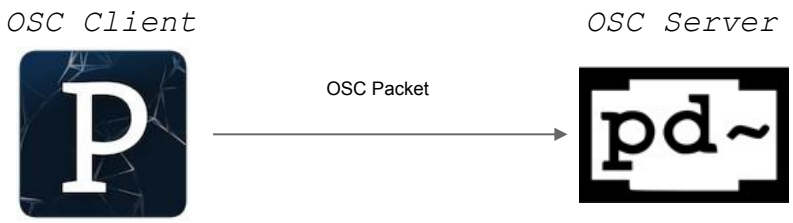
OSC Address Pattern +
OSC Type Tag String +
OSC Arguments



OSC Address Space
i metodi sono le parti più periferiche di queste ramificazioni



/root /c1 /c2 /c3 /c4 /method_A


OSC Client
OSC Server
OSC Packet
pd~

```
import oscP5.*;
import netP5.*;

OscP5 oscP5;
NetAddress myRemoteLocation;
```

```
void setup() {
  [...]
  oscP5 = new OscP5(this,12000);
  myRemoteLocation = new NetAddress("127.0.0.1",12000);
}
```

```
void inviaMessaggio() {
  [...]
  OscMessage myMessage = new OscMessage("/root/a/uno");
  myMessage.add("bang");
  [...]
  oscP5.send(myMessage, myRemoteLocation);
}
```


oggetto oscP5 per l'invio, ma anche per la ricezione dei messaggi OSC.
indirizzo di **loopback**

import mrpeach

oggetto **import**: fondamentale per importare una particolare libreria per la patch in esecuzione. Nessuno degli oggetti nominati di seguito funzionerebbe senza che si importi la libreria ***mrpeach***

udpreceive udpsend

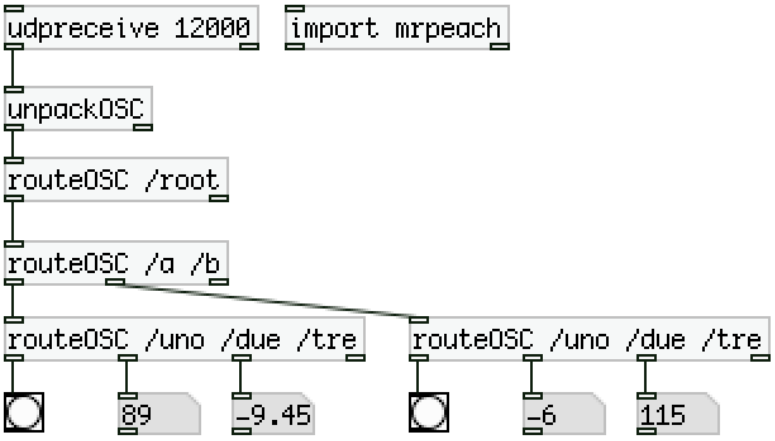
oggetto **udpreceive**: riceve dati attraverso una connessione UDP e li rilascia sottoforma di byte grezzi. Nel nostro esempio, creeremo l'oggetto passandogli un solo argomento: il numero di porta sul quale l'oggetto deve rimanere in ascolto per ricevere dati. Il suo corrispettivo per l'invio di dati è **udpsend**.

unpackOSC packOSC

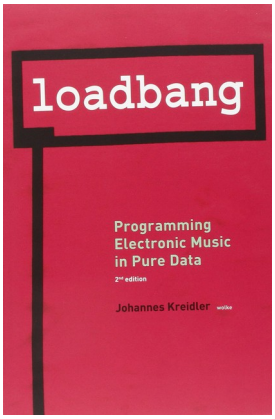
oggetto **unpackOSC**: processa i byte in ingresso interpretandoli come pacchetti OSC. Il suo duale è **packOSC**.

routeOSC

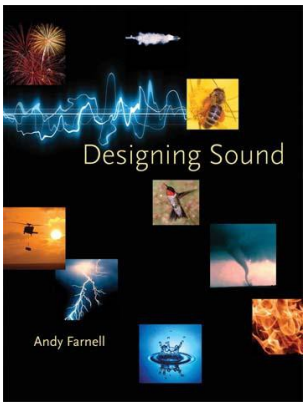
oggetto **routeOSC**: effettua il processon noto come *dispatching*: smista i messaggi OSC in ingresso su base della conformazione dell'indirizzo (OSC Address Pattern, una stringa che comincia per '/').


udpreceive 12000
import mrpeach
unpackOSC
routeOSC /root
routeOSC /a /b
routeOSC /uno /due /tre
routeOSC /uno /due /tre
89
-9.45
-6
115

Alcuni Spunti



loadbang di Johannes Kreidler
wolke verlag 2009



Designing Sound di Andy Farnell
MIT press 2010