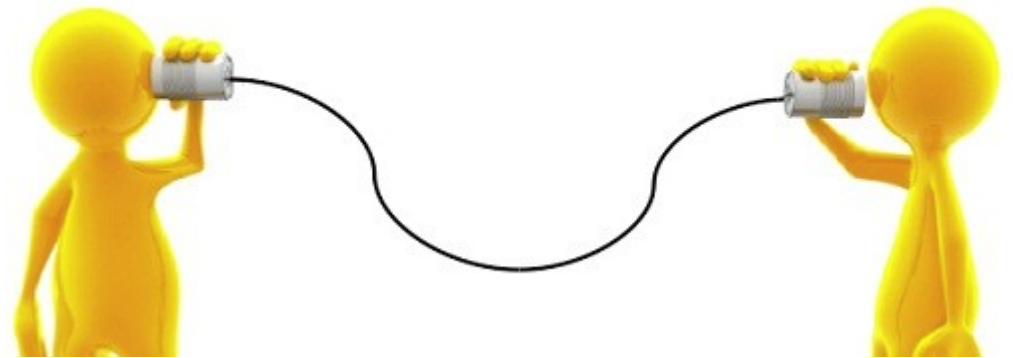


open sound control

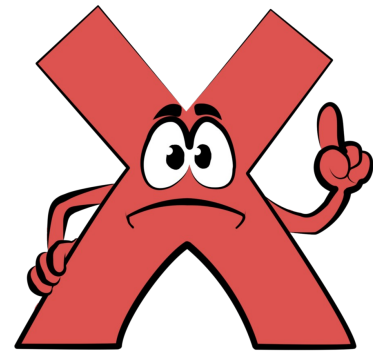


What is OSC?

- Communication protocol for sending messages and data from one software process to another
 - Examples:
 - openFrameworks to/from Max/MSP
 - iPad control surface to openFrameworks
 - my laptop to all of yours




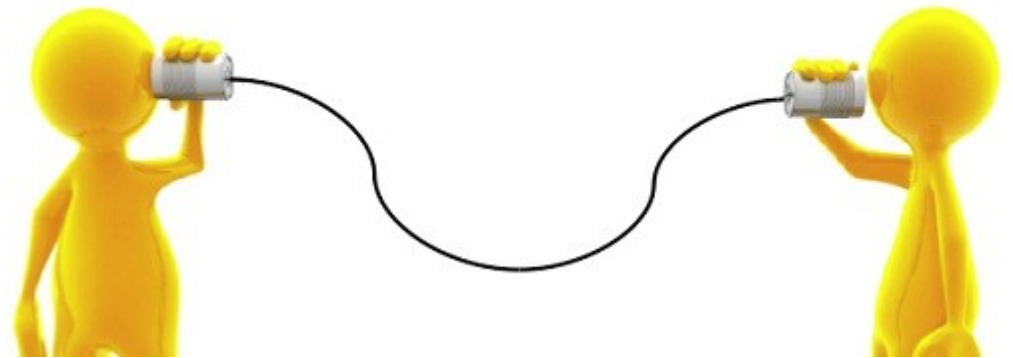
What is it **not**?



- Not a piece of software
 - Just a set of common messaging conventions
 - Any piece of software/hardware can choose to use it
- Not a network
 - Runs on top of existing networking protocols
 - Can be sent over WiFi, Ethernet, Internet, etc.

OSC use scenarios

- Communication between applications
 - oF to oF (option1)
 - see oscSend / oscReceive
 - oF to oF (option 2)
 - oscParametersReceiver / oscParametersSender (in examples/gui)
 - oF to/from Processing
 - Processing has oscP5 library  [procReceive/procSend](#)
 - oF to Max
 - Wekinator to Max



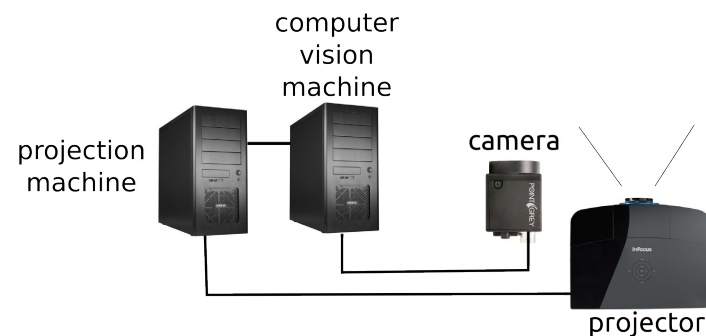
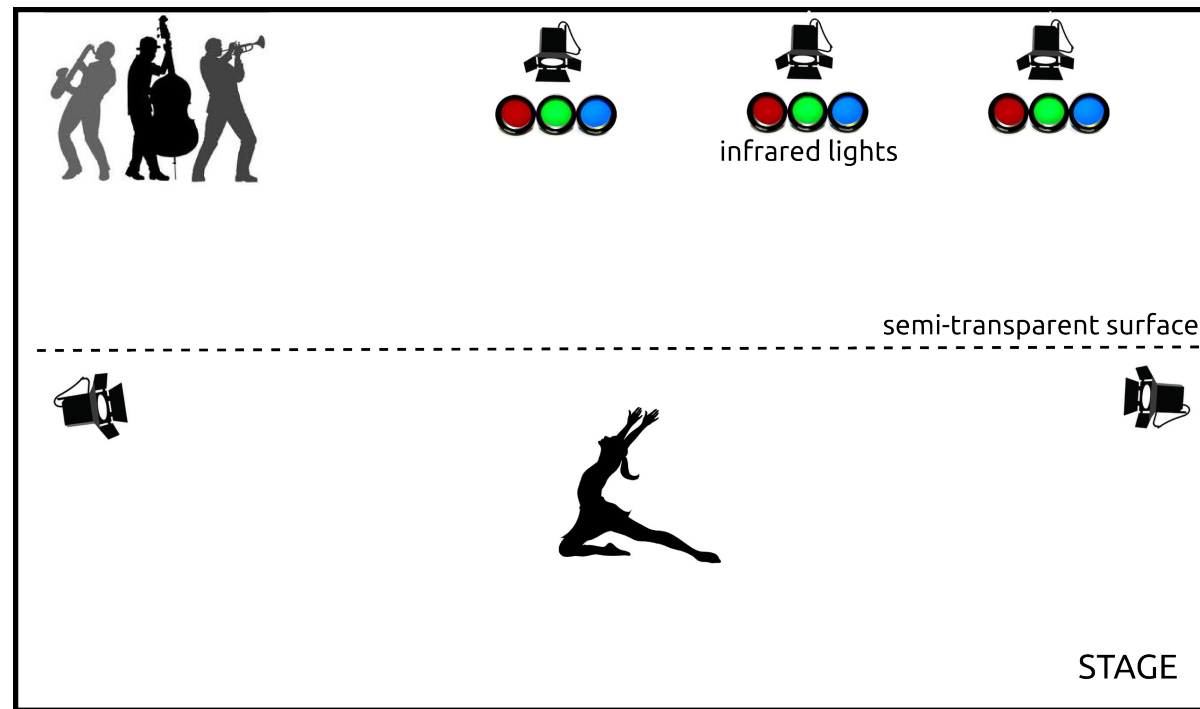
OSC use scenarios

- Communication from other hardware
 - iPad/iPhone
 - OSCemote
 - touchOSC
 - GyrOSC
 - Android
 - touchOsc
 - Control (free)
 - Reactivision fiducial tracking
 - Synapse: Kinect skeleton tracking to OSC
 - Oscuino: Arduino to OSC



OSC use scenarios

- Synchrony and distributed systems



How do you send packages?

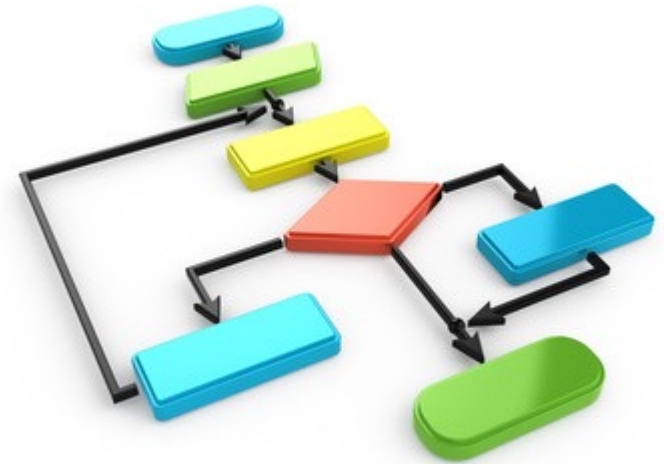
what to set

- **host** network address
 - 127.0.0.1 or “localhost” if on same machine
 - IP address or name (“fray.local”) for other machine
- **port** at that host
 - Common choices: 6448, 6453
- **message name** (a string)
 - /slider/1/
 - /playNote/
- **data** (optional)
 - floats, ints, strings



How OSC packet is sent in pseudocode

- **setup()**
 - create OSC sender object
 - point to host + port
- **as stuff happens**
 - give sender object a message name
 - give sender message data in order
 - tell sender to send



How do you receive packages?

what to set

- **port** to listen on
 - must match sender's port!
- **message** to listen for
 - must match sender's message name



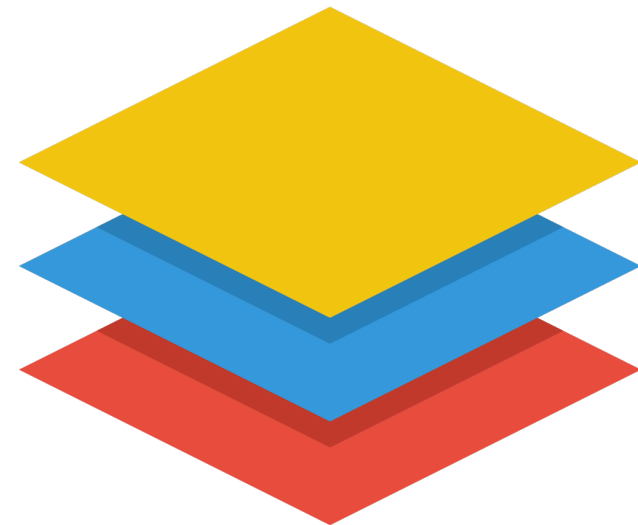
How OSC packet is received in pseudocode

- **setup()**
 - create OSC receive object
 - tell receiver to listen on specific port
- **update()**
 - check if message has been received
 - unpack data in the order it was packed



miscellanea

- OS is responsible for sending/receiving data at a certain port
- OSC addon/library takes care of OS interactions
 - asks OS to use UDP protocol (instead of TCP) for sending data over networks



general tips

- never have more than one program (or process) listening on the same port
- use OSCulator to help debug
- some addresses reserved (ex. 80)
- turn off Firewall off / WiFi on if you're having problems
- send to many computers with just 1 message using multicast IP address: 224.0.0.1
- sending raw audio/video over OSC is usually a bad idea!
 - consider Jacktrip instead for audio



connectivity Tips

- use ping to check connection
- fix addresses if it is permanent exhibition (from the router)
- don't send too many packages per second
- don't use wifi for production work
- use your own WiFi router to make more robust (no need for internet connection)

