DIGITAL MUSIC WORKSHOP / 01 / ELECTRONIC MUSIC PROTOCOLS + INTER-APPLICATION + DEVICE COMMUNICATION

DIGITAL MUSIC WORKSHOP / 01 / ELECTRONIC MUSIC PROTOCOLS + INTER-APPLICATION + DEVICE COMMUNICATION

- MIDI
- OSC
- DSP with *virtual audio driver*
- interfacing applications

MIDI


MIDI

MIDI

- Musical Instrument Digital Interface (MIDI)
- MIDI is a standard:
  - communications protocol
  - digital interface
  - electrical connectors
- MIDI @wikipedia
- idea: "playing notes on a piano keyboard"

MIDI

- messages types (channel + system messages):
- notes (channel, pitch, volume)
  - trigger: `note_on` + `note_off`
  - value range: `[0...127]`
- system messages (e.g `MIDI Beat Clock` (24 PPQN))

OSC

- Open Sound Control (OSC)
- OSC is a standard:
  - communications protocol
  - ethernet networks (usually `UDP`, sometimes `TCP`)
- successor to MIDI

OSC

- sending+receiving messages
- anatomy of a message: `address pattern` + `type tags`
- example message: `/note_on/48,100`
  - `address pattern`: `/note_on`
  - `type tags` : `48,100` > `ii` (two integers)
- processing library `oscP5`
- @funfact(SuperCollider communicates interally with OSC)

DSP WITH *VIRTUAL AUDIO DRIVER*

- rerouting sound output internally with virtual sound cards
- `BlackHole` @MacOS
- @todo(add @Windows)
- `JACKAudio` @todo(add @Linux)

INTERFACING APPLICATIONS

- Digital Audio Workstations (DAW)
  - GarageBand
  - Logic Pro
  - Abelton Life
- Processing.org

TON

Ton a framework for exploring and teaching generative music making and algorithmic compositions. it facilitates simple ways of playing musical notes, facilitates easy access to low-level digital signal processing (DSP) and supplies rhythm and timing as well as some standard muscial mechanics. the library acts as a simple adapter to various sound in- and outputs like JSyn, MIDI, OSC, or analog audio.

TON / MIDI

- setting up virtual (internal) MIDI
  - IAC (Inter-application communication) in MacOS
    `/System/Applications/Utilities/Audio MIDI Setup.app`
  - `loopmidi` @todo(add Windows)
- list available ports + devices (see `MidiView.app`)
- `ExampleBasics05MIDI` @note(channel is an `instrument()` in `Ton`)
- connecting external devices: `ExampleEvent03MIDIExternalKeyboard`
- `ExampleEvent02MIDIClock`
  - with CLI tool
  - with Logic Pro

TON / OSC

- `ExampleEvent01ReceiveMIDIandOSC`
- receive messages from `Processing.org` (localhost)
- receive messages from TouchOSC (iOS)

TON / DSP

- list available ports
- AppDSPwithJSynToneEngine

COMMAND LINE INTERFACE (CLI)

#cherryontop

- CLI: another kind of interface
- `ExampleSpeechSynthesis`

APPENDIX

```
/* receiving and sending OSC messages */

import oscP5.*;
import netP5.*;

OscP5 oscP5;
NetAddress mRemoteLocation;

void setup() {
    size(640, 480);
    background(0);
    oscP5 = new OscP5(this, 7000);
    mRemoteLocation = new NetAddress("127.0.0.1", 7001);
}

void draw() {}

void mousePressed() {
    OscMessage myMessage = new OscMessage("/note_on");
    myMessage.add(48);
    myMessage.add(100);
    oscP5.send(myMessage, mRemoteLocation);

    println("+++ sent message:");
    print(" addrpattern: "+myMessage.addrPattern());
    println(" typetag: "+myMessage.typetag());
}


void oscEvent(OscMessage pOscMessage) {
    print("+++ received an osc message:");
    print(" addrpattern: "+pOscMessage.addrPattern());
    println(" typetag: "+pOscMessage.typetag());
}
```