

# OSC – Open Sound Control Protocol

## Ein Kandidat für die Lichttechnik?

Frank Burghardt, Markus Minini

[www.DMXControl.de](http://www.DMXControl.de)



# OSC Mythos – einige Beispiele

## (Musikbereich)

If you want **the ultimate** in external control of Resolume, the Open Sound Control (OSC) protocol is the answer.  
OSC is **becoming increasingly popular** and is used by programs like MAX/MSP, VVVV and Reaktor (Native Instruments).  
OSC can be seen as a **successor of MIDI** and offers a much higher accuracy and is more flexible because it can be sent over a network including wifi. (Resolume Manual)

OSC opens a **new era** in the field of real-time control and human-machine interfaces  
(Lumor Manual)

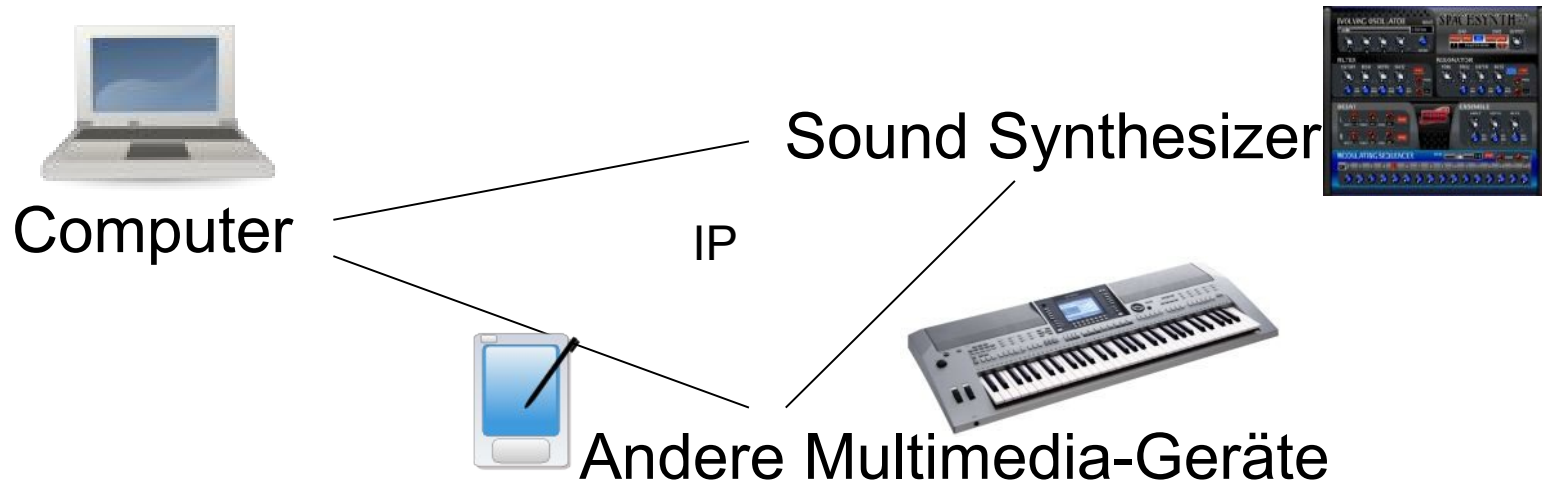
Well, my wish has been granted, the latest AlgoScore supports OSC, **and I'm a happy guy.**

Die meisten Controller müssen erstmal eine **technische Altlast überwinden**, nämlich MIDI. Da viele Controller sowieso schon über USB angeschlossen werden, kann eigentlich auf höher auflösende Protokolle zurückgegriffen werden, vornehmlich natürlich OSC.

# Inhalt

- Was ist OSC?
  - Vergleich von DMX, MIDI und OSC
  - Bekannte Anwendungsfälle von OSC
- Wie funktioniert OSC?
  - Beispielkonfigurationen
  - Grundkonzepte: Messages, Adressen etc.
- Wo wird OSC unterstützt?
  - Geräte und Tools, die OSC unterstützen
- Wie OSC in der Lichttechnik anwenden?
  - Diskussion von OSC-Schemas im Lichtbereich
- Zusammenfassung und Livedemo

# Was ist OSC?



Ein Protokoll zur Kommunikation zwischen Computern, Sound Synthesizern und anderen Multimedia-Geräten

Eine Alternative zu:

- MIDI (auf Musik fokussiert)
- MediaControl (auf Video fokussiert)
- DMX (auf Licht fokussiert)

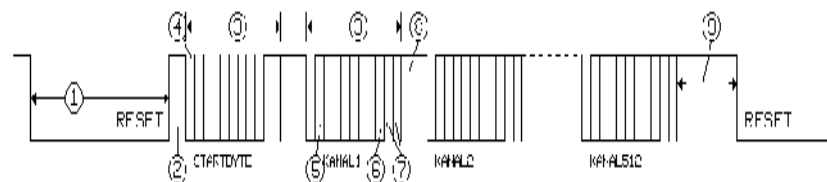
# Vergleich von MIDI, DMX und OSC

	<b>MIDI</b>	<b>DMX</b>	<b>OSC</b>
Name	<b>Musical Instruments Digital Interface</b>	<b>Digital Multiplex</b>	<b>Open Sound Control</b>
definiert	1983	1990	2002-2004
Protokoll	seriell	seriell	unabhängig (UDP)
Geschwindigkeit	31.250kB/sec	250 kbit/s	IP netzabhängig
Auflösung	128 Stufen	256 Stufen	beliebig (float)
Erweiterungen (IP)	RFC 4695	ArtNet	Spec 2.0 in Vorb.
Kommentar	vordefinierte Messages (byte code), am Keyboard-Konzept orientiert	RDM verbessert Funktionalität, aber das low level Byte-Konzept wird dabei beibehalten	liefert nützliche Features, die nicht in MIDI verfügbar sind, z.B. bel. Auflösung, Standardnetzwerk-IF

**144 60 64**  
**(MIDI Note-on)**



**/wii/ir/x 0.1503**  
**/play-note 15 0.9**



# Was ist Open Sound Control?

- Ein Message orientiertes Framework,
  - aber es gibt keine vordefinierten Nachrichten und Features  
-> flexibel und offen ... **? gut oder schlecht?**
- Neue Eigenschaften:
  - ein intuitives Adressierungs-Schema,
  - die Möglichkeit zukünftige Events zu definieren
  - variable Datentypen
- Funktioniert mit embedded Devices, PDA, iPhone, Wii, über Internet.
- OSC ist kein physikalisches Kommunikations-Medium
  - arbeitet mit beliebigem Übertragungsmedium
  - OSI Schicht 6 „presentation“, nutzt meistens UDP
- OSC hat MIDI nicht ersetzt, weil kein automatisches connect-and-play (or plug-and-play) Konzept existiert
  - Verbundene Geräte (über Ethernet, WLAN, Bluetooth etc) kennen einander nicht - und auch ihre Fähigkeiten nicht.
  - Ein Fileformat ähnlich zu Standard MIDI File existiert nicht, um Daten zwischen Applikationen bekannt zu machen.

# Bekannte OSC-Anwendungsfälle

- Sensor/Gesture-Based Electronic Musical Instruments
- Mapping nonmusical data to sound
- Multiple-User Shared Musical Control
- Web interfaces
- Networked LAN Musical Performance
- WAN performance and Telepresence
- Virtual Reality
- Wrapping Other Protocols inside OSC

aus [1]

- Fernsteuerung: Übertragung der Werte von Slider, Encoder, (drum) Pad, Faderwing und insbesondere über Touchscreen zur Eingabe
- „Befehlsübergabe“ zwischen Tools
- Media-Steuerung
- Table-top tangible user interfaces, see [TUIO]

**Steuerung von Licht -> ???**

# OSC Applikationen



VirtualDJ



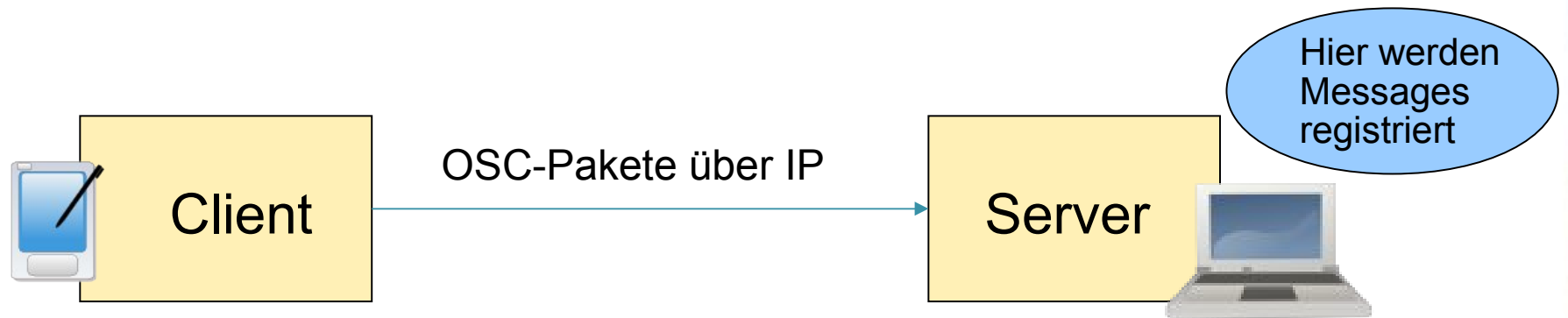
Quintet.net



VJing

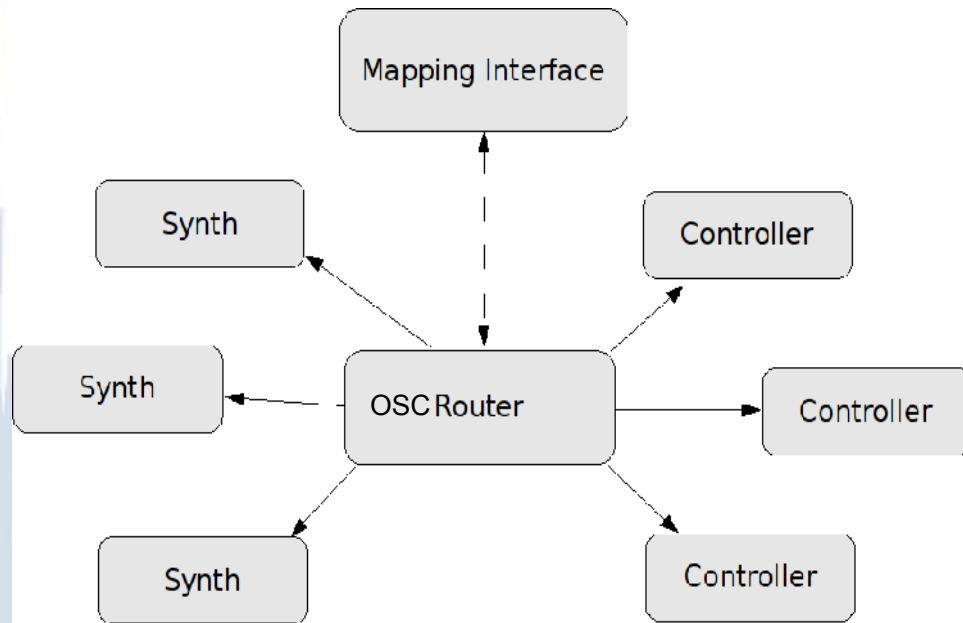
siehe: [3]

# Konfiguration

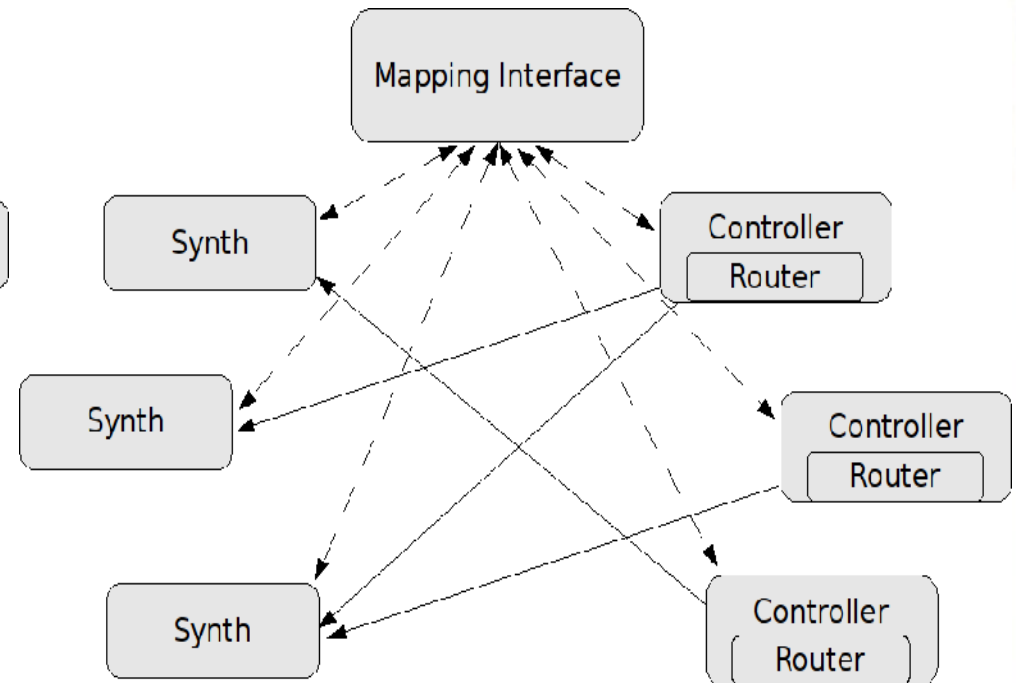


- sendet OSC-Messages an einen oder mehrere Server
  - Adresse kann Wildcards enthalten
  - Eine OSC-Message kann mehrere Methoden eines Server aufrufen
  - empfängt OSC-Messages von Clients
  - führt Methoden aus, die den Adressen zugeordnet sind
- 
- Datenübertragung nur in eine Richtung möglich (trotzdem kann OSC bidirektional verwendet werden)
  - Paket kann OSC-Message oder OSC-Bundle enthalten
  - Paket enthält Angaben zur Länge der Daten und die Daten selber

# Netzkonzepte



sternförmig



dezentral

- (Aus-)Nutzung der IP-Infrastruktur und der OSC Routing-Mechanismen zur Verteilung der Messages
- Quelle: [2]

# Grundkonzepte: Messages, Bundles, Adressen

- Das OSC Adress-Schema liefert 3 wichtige Fähigkeiten:
  - Intuitive Namen für angesprochene Adressen,
  - Unbeschränkte Zahl von Namensbereichen,
  - Adressbereiche innerhalb von Nachrichten (inkl. Wildcards, Patterns).
- Adressen starten mit /
  - meist hierarchisch aufgebaut (-> Namensschema)
  - Können Wildcards enthalten
  - Bezeichnet eine oder mehrere Methoden auf dem Server
- Beispiele: /Gerät/Baugruppe/Steuerelement  
/fader/1/value

# Grundkonzepte: Adressen

Blätter des Baumes sind

**Methoden** des Servers

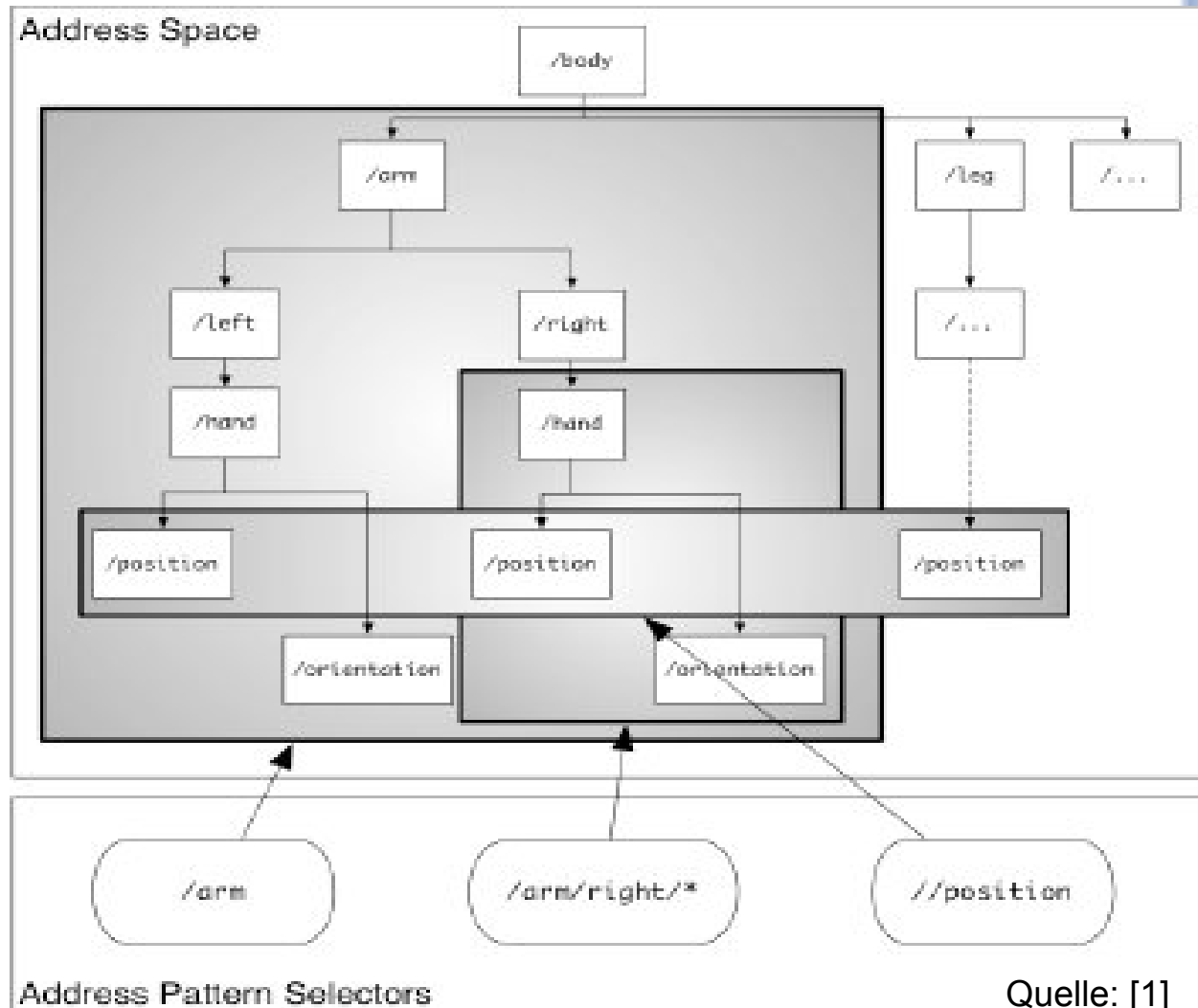
- `/fader/1/1`
- `/fader/2/6`
- `/buttons/1/parsvorne`
- `/buttons/2/parshinten`

Beispiele für **Pattern**:

- `/fader/?/[1-3]`
- `/buttons/1/pars*`

Alle anderen Stufen sind  
Container(-methoden)

- Namensschema präsentiert das Objektmodell
- Durchdachtes Namensschema profitiert für pattern matching



Quelle: [1]

# Grundkonzepte: Message

- Eine OSC Message besteht aus:
  - **OSC-Adresse**
  - **OSC-Typ-Tag -> (Integer, float, string ... [arrays])**
    - > **Datentypen der Argumente**
  - **OSC-Argumente -> (Werte)**
- OSC-Argumente sind optional
  - Beispiele:
    - `/oscillator/4/frequency ,f 440`
    - `/layer1/clip2/video/effect ,ii 3 35`
    - `/track1/start`

# Grundkonzepte: Bundles

- Gruppe von OSC-Messages, enthält eine oder mehrere OSC-Messages und einen Timetag
- OSC-Methoden müssen bei Erreichen des Timetags aufgerufen werden
- kann weitere Bundles enthalten

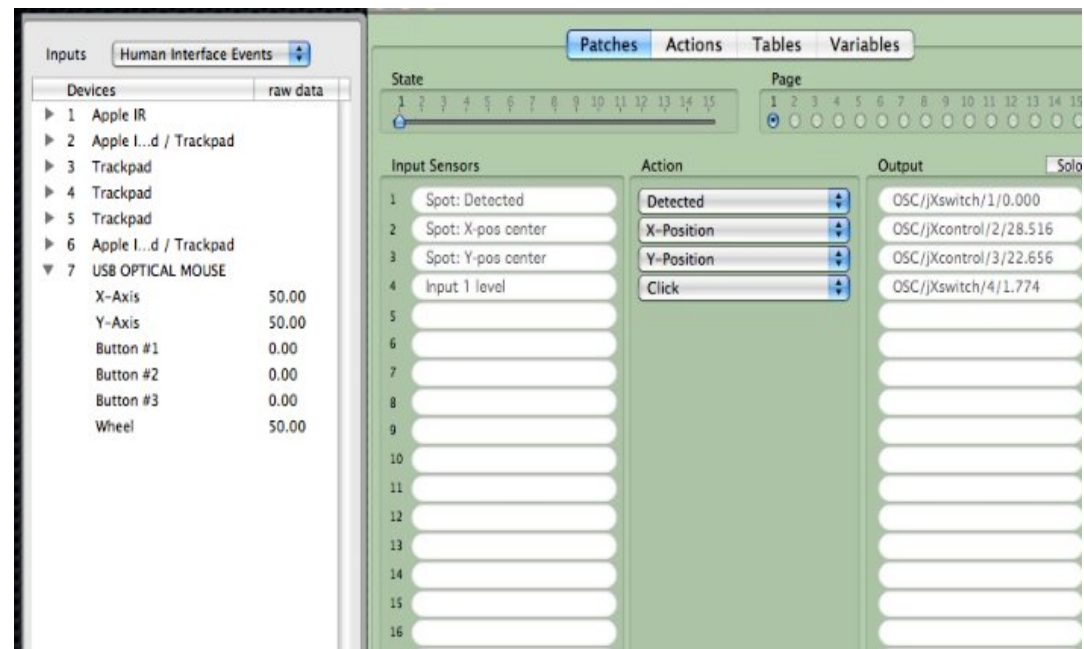
Open Sound Control *Packet/Message*:  
[ **address type-tags (i,f,s,b) arguments . . .** ]

Open Sound Control *Bundle*:  
[ **"#bundle" timestamp integer-length packet-1 packet-2 . . .** ]

- Abarbeitung der OSC-Messages in der Reihenfolge, wie sie im Bundle stehen
- Quasi simultane Verarbeitung der messages im Bündel

# Beispiele für OSC-Router

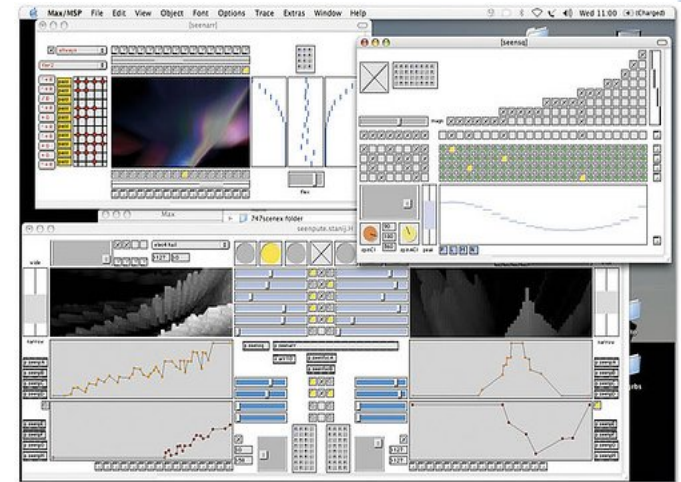
- **junXion** © (Mac OS) ist ein Software Tool, das Daten von USB-Geräten (Joysticks, Touchscreens, Maus), MIDI / OSC Nachrichten, Audio Inputs, WiiRemote Controllers, Arduino Sensorboards and Video Image Tracking verarbeitet. Die Outputdaten (MIDI oder OSC) können zu anderen MIDI/OSC-Controller (SW oder HW) geroutet werden.
- OSCulator
- GlovePie



# Geräte und Applikationen, die OSC unterstützen

- Audio-/Videobereich:

- Reaktor, MAX/MSP
- Mirage, MADJack, Resolume
- Traktor, Ableton, VirtualDJ
- MXWendler, Ventuz, VVVV, VDMX

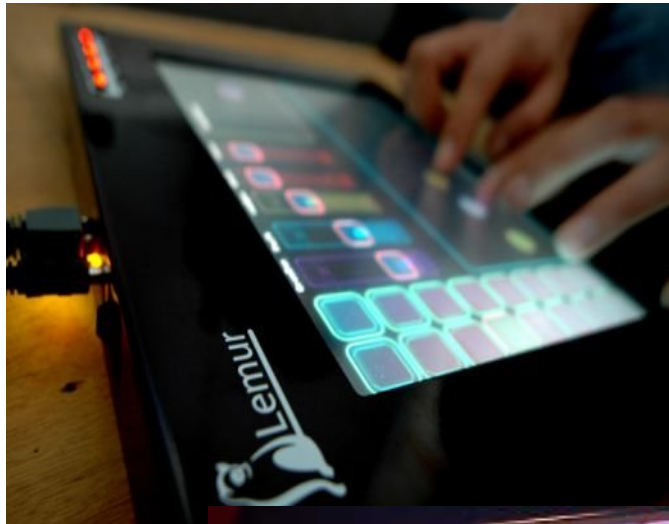


- Hardware:

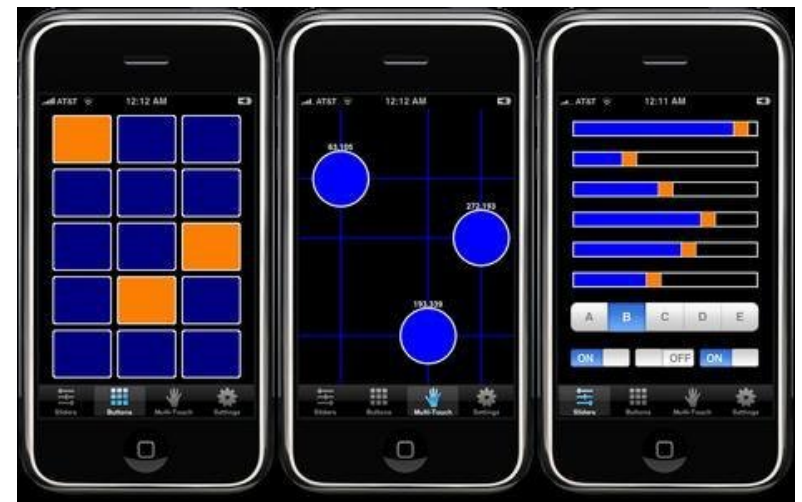
- Make Controller Kit, Monome, KISS Box
- WiSE Box, uOSC
- Lemur



# OSC basierte (Fern-)Steuerungen



Lemur



iPhone mit OSC-Applikationen

# OSC-Softwaretools

- OSC Monitore:

- <http://www.frieder-weiss.de/OSC/index.html>
  - <http://sourceforge.net/projects/braun/>

- iPhone Clients:

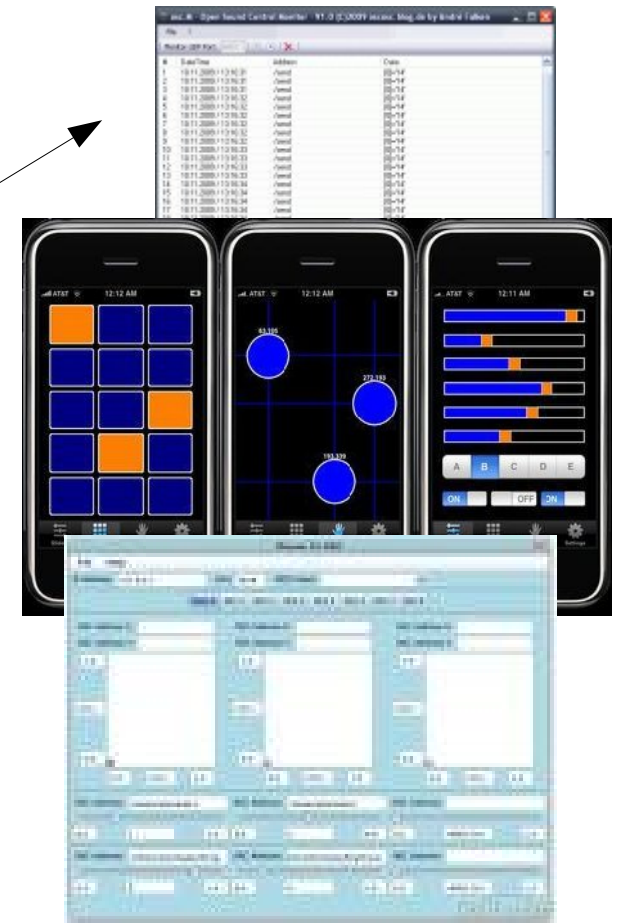
- <http://poly.share.dj/projects/#mrrmr>
- <http://hexler.net/software/touchosc>

- Andere Clients:

- Mouse2OSC

- Entwicklungspakete:

- Processing: osp5, Python: SimpleOSC
- C# : Bespoke OSC, Open Sound Control for .NET (<http://luvtechno.net>)
- C/C++: oscpack, OCS-Kit, etc.
- Java: JavaOSC, NetUtil, etc.



# Namensraum für Lichtsteuerung

- Wir müssen den Einsatzzweck klären!
- Wir benötigen ein Namensschema!
  
- 3 Varianten zur Diskussion
  - elementar (DMX-Ebene)
  - control orientiert
  - abstrakt Geräte orientiert
  
- **Achtung:** Konzepte sind als Software-Interface der Steuerung gedacht, nicht zum Ansteuern der DMX-Geräte!  
(OSC in den Lampen braucht noch seine Zeit, aber ist möglich.)

# Beispiele für Schemas

- Typische OSC control-Elemente:
  - **Objekte** (name space): deck, mixer, track, layer, composition, device, playlist, fader, rotory, button
  - **Parameter:** value, direction, speed, mode, volume, position, pan, scale, rotate, opacity, size
  - **Operationen:** select, clear, solo, connect
- Hierarchischer Namensraum (vs. flacher Namensraum)
  - Adresse für Lautstärke des 1. Layers:
    - **/layer1/audio/volume/values**
  - Erster Video Effekt des 2. Clips im 1. Layer:
    - **/layer1/clip2/video/effect1**
  - Instruktion, dass das System alle 500 ms einen Status sendet:
    - **/constraint/motor/force/magnitude/get 500**

# OSC-Namensraum für „Licht“ (1)

## - elementar (DMX-Ebene) -

- Beispiel OSC-Namensraum bei Umwandlung von OSC-Nachrichten in MIDI-Befehle:
  - /midi/out/noteOn [channel] [key] [velocity]
  - /midi/out/noteOff [channel] [key] [velocity]
- Für DMX ist folgender Namensraum auf elementarer Ebene vorstellbar:
  - /dmx/out ,ii [chn] [val] - Setze Kanal [chn] auf Wert [val]
  - /dmx/in ,ii [chn] [val]
- Oder auch:
  - /dmx/set ,ii [chn] [val] - Setze Kanal X auf Wert Y
  - /dmx/fade ,iif [chn] [val] [s] - Fade Kanal X auf Wert Y über eine Periode von Z Sekunden
- **Nachteil:** wenig Abstraktion  
“nur” alternative Darstellung für DMX-IN  
kaum Nutzung der OSC Routing-Features möglich

# OSC-Namensraum für „Licht“ (2)

- control orientiert -

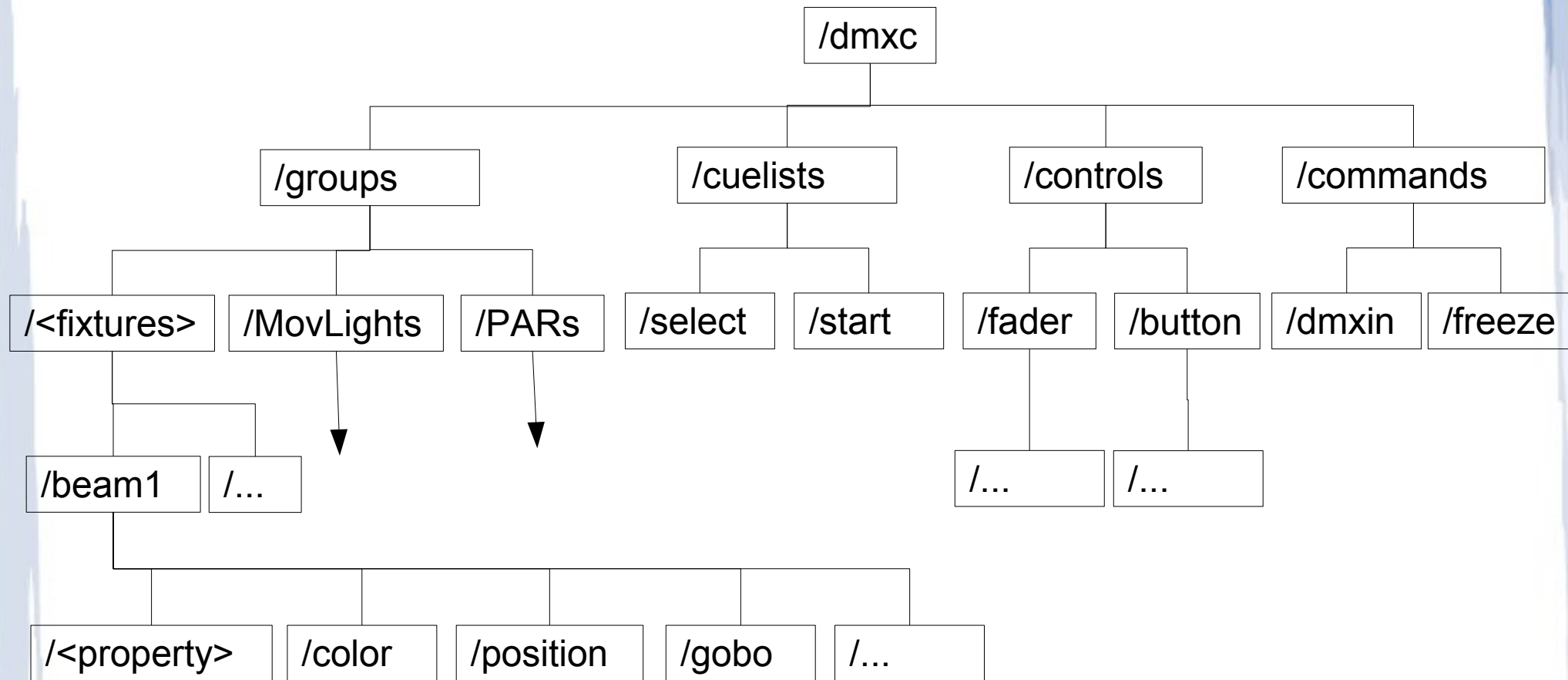
- Definition auf der Basis der Steuerelemente:
  - /dmxc/submaster/<number>/value
  - /dmxc/button/<number>/value
  - /dmxc/fader/<number>/value
- Generisches Konzept, aber aus der Sicht der Steuerung durch eine „Fernbedienung“ (wie iPhone touchOSC) definiert
- Server (DMX-Steuerung) ist für die Abbildung auf Steuerlogik zuständig
- **Nachteil:** Clients bleiben „dumm“, sind reine „Faderwings“

# OSC-Namensraum für „Licht“ (3)

## - abstrakt Geräte orientiert -

- Definition auf der Ebene einer HAL („Hardware Abstraction Layer“):
  - /dmxc/group/devices/beam/“property“
  - Groups: MovingLights, Dimmer, Switch, <eigene Gruppen>
  - Beam: Geräte können mehrere steuerbare Strahlen haben
  - Properties: color, position, gobo, „Helligkeit“
  - Bsp.: **/dmxc/MovLights/\*/beam?/color blue**  
würde bei allen MovingsLights die Farbe blau einstellen
- **Vorteil:**
  - Clients können intelligente Aufgaben übernehmen
  - Verteilte Steuerung aus mehreren Komponenten kann über diese Nachrichten sehr gut interagieren
  - DMX-Steuerung kümmert sich um die Abbildung der logischen Sicht auf physikalische Sicht (DMX-Werte) und ist OSC-Router für Mediasteuerung u.a.
- **Vorschlag:** Initiative für „Licht“-OSC-Schema ins Leben rufen

# Ausschnitt „Licht“-Namensraum

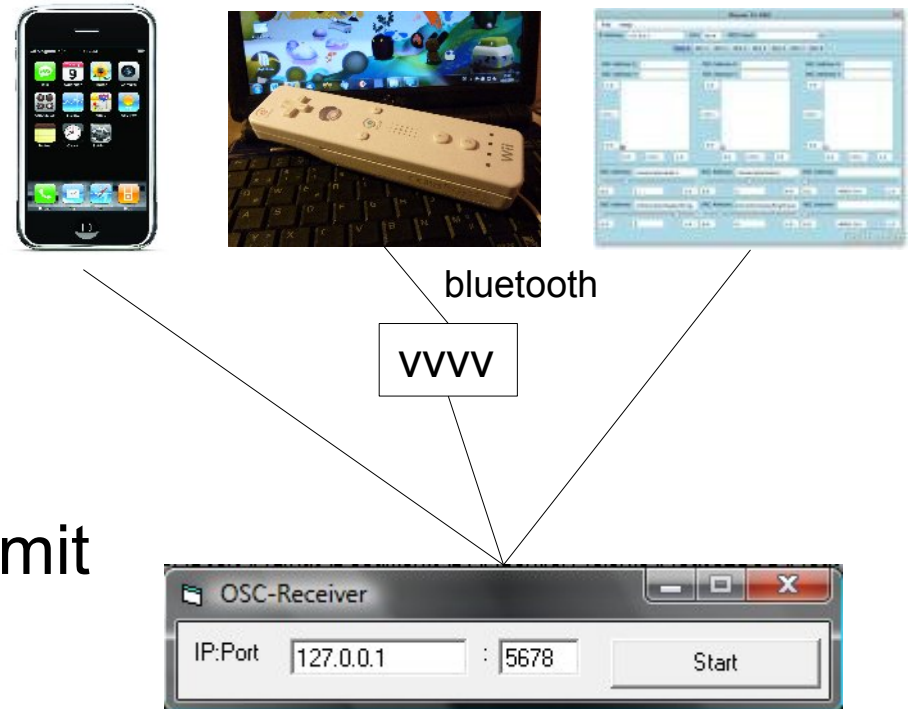


# Zusammenfassung

- **Nutzung von OSC in Lichttechnik:**
  - interessante Alternative zur Fernbedienung (OSC-fähige Endgeräte/Eingabegeräte)
  - kann äquivalent zur heutigen MIDI-Nutzung verwendet werden
  - OSC wird nicht MIDI ersetzen, sondern ergänzen, siehe [4]
  - für verteilte Steuerungen sehr sinnvoll
  - sehr hilfreich für Multimedia-Interaktion, wenn DMX-Steuerung als OSC-Client arbeitet
- **Probleme aus heutiger Sicht:**
  - Zuverlässigkeit (UDP/TCP) und fehlendes plug&play Verhalten
  - keine standardisierten OSC-Schemas zur Steuerung von DMX-Geräten
- **Vorschlag:**
  - Initiative für Licht-OSC-Schema starten – Wer macht mit?

# LiveDemo

- OSC-Apps (iPhone)
- WiiMote (über vvvv)
- Mouse2OSC (Windows)
- Empfänger ist DMXControl mit OSC-Receiver-Plugin



OSC-Receiver Einstellungen

IP:Port 127.0.0.1 : 5678 Lernen Analyzer  Empfangene OSC-Messages 0 Speichern Hauptfenster öffnen

OSC-Address	Wert	Operator	Vergleichswert	inMin	inMax	outMin	outMax	Min	Max	Modul	Gerät/Funktion	Kanal	Flags	Wert
<input checked="" type="checkbox"/> /xy/1/x	1. Float	any	-	0	1	0	255	0	255	Geräte	TS 7 (1)	Pan	-,-...	-
<input checked="" type="checkbox"/> /xy/1/y	1. Float	any	-	0	1	255	0	0	255	Geräte	TS 7 (1)	Tilt	-,-...	-
<input checked="" type="checkbox"/> /xy/2/x	1. Float	any	-	0	1	0	255	0	255	Geräte	TS 7 (2)	Pan	-,-...	-
<input checked="" type="checkbox"/> /xy/2/y	1. Float	any	-	0	1	255	0	0	255	Geräte	TS 7 (2)	Tilt	-,-...	-
<input checked="" type="checkbox"/> /xy/1/dimmer	1. Float	any	-	0	1	0	255	0	255	Geräte	TS 7 (1)	Helligkeit	-,-...	-
<input checked="" type="checkbox"/> /xy/1/farbe	1. Float	any	-	0	1	0	255	0	255	Geräte	TS 7 (1)	Farbe	-,-...	-
<input checked="" type="checkbox"/> /xy/2/dimmer	1. Float	any	-	0	1	0	255	0	255	Geräte	TS 7 (2)	Helligkeit	-,-...	-
<input checked="" type="checkbox"/> /xy/2/farbe	1. Float	any	-	0	1	0	255	0	255	Geräte	TS 7 (2)	Farbe	-,-...	-

# Links / Sources

- [1] <http://opensoundcontrol.org/>
- [2] <http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/diplom/sukale.pdf>
- [3] <http://www.slideshare.net/GamerSize/open-sound-control-as-middleware-for-games-accessibility-and-bodymovement-controlled-gaming>
- [4] <http://www.midi.org/aboutmidi/midi-osc.php>
- [5] <http://www.linuxjournal.com/content/introduction-osc>
- [6] [http://www.steim.org/steim/junxion\\_v4.htm](http://www.steim.org/steim/junxion_v4.htm)

# Fragen ?

[www.DMXControl.de](http://www.DMXControl.de)



Fragen, Anregungen und Bekundungen zur Mitarbeit bitte an  
[info@dmxcontrol.de](mailto:info@dmxcontrol.de)

# Anhang: Adress Pattern

- Pattern Syntax:
  - \* – matches zero or more characters
  - ? – matches any single character
  - [characters] – matches characters
  - Minus, e.g. [1-3] matches range of characters
  - Leading !, e.g. [!0-9] negates the match
  - {string1,string2,string3} – match a string in list
- If more than one destination matches address pattern:
  - Send copy of message arguments to each node
  - Fanout to unknown destinations
  - For example: control all “voices” with volume pedal