# Welcome!

Open Sound Control (OSC) Conference, July 30, 2004

Presented by the UC Berkeley Center for New Music and Audio Technologies (CNMAT) and the UC Discovery Grant Program

## This event was funded by

the UC
Discovery Grant

Success is a great investment

Explore your opportunities for Industry-UC partnerships at
www.ucdiscoverygrant.org

## Acknowledgments

- Richard Andrews, PR and Admin. Liaison
- Gillian Edgelow, Admin. and Coordination Support
- Adrian Freed, OSC Architect
- Peter Nyboer, Technical Support
- Andy Schmeder, Webmaster
- David Wessel, Director of CNMAT and Prof. of Music
- Matthew Wright, Conference Director, Proceedings Editor, OSC Architect
- Michael Zbyszynski, Technical Support

## Conference Schedule

- *…is printed in the proceedings*
- Intro/Keynotes
- Paper Session I: Implementations of OSC
- Paper Session II: OSC Hardware
- Lunch
- Paper Session III: OSC-related research
- Poster Session: OSC Project Gallery
- Presentation of Draft Proposals from standardization working groups

## Where is…?

- Bathroom: outside auditorium, to left
- Lunch: served in hallway, eat outside
- Reception after conference (6-8pm): CNMAT (1750 Arch St.)

# Brief Overview of OSC and its Application Areas

Matthew Wright
Musical Systems Designer
CNMAT

## What is OSC?

- Networking protocol for real-time musical control information
- Introduced by CNMAT in 1997
- Transport-independent (today used with UDP, TCP, WiFi, serial connections, and within applications)

## Goals of OSC

- Lightweight
- Easy to implement
- Flexible
- Simple
- Temporal semantics supporting sound control
- Reasonably space-efficient

## OSC Overview: Messages

- Address: URL-style
- Arguments: strings, binary numbers, “blobs”, etc.

/filters/3/center-freq 3123.45

## OSC Overview: Address Patterns

- Address of an OSC message can be a regular expression pattern
- E.g., /filters/[2-4]/cutoff-freq 2354.1
- Semantics: as if separate messages (with the same arguments) were sent to each matching address
- Small regexp language: *, ?, {a,b,c}, [a-h]

## OSC Overview: Argument Types

| | |
|---|---|
| i | int32 |
| f | float32 |
| s | OSC-string |
| b | blob (binary data) |
| h | int64 |
| t | Time Tag |
| d | float64 (“double”) |
| S | symbol |

| | |
|---|---|
| c | ASCII character |
| r | RGBA color |
| m | MIDI message |
| T | true |
| F | false |
| N | nil |
| I | infinitum |

## OSC Overview: Address Spaces

- Every address space is application-specific
  - – Symbolic names of features, parameters…
  - – Arbitrary arrangement into tree structure
- OSC standard proscribes nothing
  - + Utterly flexible
  - – No automatic “plug and play”

## OSC Address Space Example (*Constellation* synthesis server)

/bell1/cutoff-freq
/glockies/midinote-off
/glockies/midinote-on
/ronbells/mute
/ronbells/play/bell2a
/ronbells/play/bell2b
/ronbells/play/bell2c
/ronbells/play/bell2d
/ronbells/play/bell3a
/ronbells/play/bell3b

/harmtones/osc[1-8]/glissdir
/harmtones/osc[1-8]/glissmag
/harmtones/osc[1-8]/init
/harmtones/osc[1-8]/mute
/harmtones/osc[1-8]/play
/harmtones/osc[1-8]/speaker
/harmtones/osc[1-8]/volume-pedal
/sampler/midinote-on
/sampler/midinote-off
/sampler/multisample
/sampler/speaker
/sampler/master-volume

## OSC Overview: Time

- “Bundle” - group of messages
  - Transmitted together
  - Must take effect atomically
- Bundles have time-tags saying when messages should take effect

## OSC Terminology

- client/server:  sender/receiver
- message: unit of OSC data
- address: target of OSC data within receiver
- address pattern: regular expression
- address space: tree of msgs. server recognizes
- schema: address space + semantics
- argument: data in a msg. (number, string…)
- bundle: collection of msgs. with a time tag

## OSC’s Application Areas

http://cnmat.berkeley.edu/OSC/application-areas.html

# Keynote Address: OSC and Digital Lifestyle Aggregation

Marc Canter
CEO, Broadband Mechanics