Managing Complexity with Explicit Mapping of Gestures to Sound Control with OSC

Matthew Wright, Adrian Freed, Ahm Lee, Tim Madden, Ali Momeni

CNMAT, UC Berkeley, 1750 Arch St., Berkeley, CA 94709, USA
email: {matt,adrian,ahm,tjmadden,ali}@cnmat.berkeley.edu

Abstract

We present a novel use of the OpenSound Control (OSC) protocol to represent the output of gestural controllers as well as the input to sound synthesis processes. With this scheme, the problem of mapping gestural input into sound synthesis control becomes a simple translation from OSC messages into other OSC messages. We provide examples of this strategy and show benefits including increased encapsulation and program clarity.

1. Introduction

We desire expressive real-time control of computer sound synthesis and processing from many different gestural interfaces such as the Boie/Mathews Radio Drum (Boie, et al., 1989), the Buchla Thunder (Buchla, 2001), Wacom Tablets (Wright, et al., 1997), gaming joysticks, etc. Unlike acoustic instruments, these gestural interfaces have no inherent mapping between the gestures they sense and the resulting sound output. Indeed, most of the art of designing a real-time-playable computer music instrument lies in designing the mapping between sensed gestures and control of the sound generating and processing.

We believe that OpenSound Control (OSC) (Wright and Freed, 1997, Wright, 1998) provides many benefits to the creators of these gesture-to-sound-control mappings. It is general enough to represent both the sensed gestures from physical controllers and the parameter settings needed to control sound synthesis, so it provides a uniform syntax and conceptual framework for this mapping. The symbolic names for all OSC parameters make explicit what is being controlled and can make programs easier to read and understand. An OSC interface to a gestural-sensing or signal-processing subprogram is a powerful form of abstraction that can expose all of the important features while hiding the implementation.

We will present a paradigm for using OSC for this mapping task and give a series of examples culled from several years of live performance with a variety of gestural controllers and performance paradigms.

2. OpenSound Control

OpenSound Control (OSC) was originally developed to facilitate the distribution of control structure computations to small arrays of loosely coupled heterogeneous computer systems. A common application of OSC is to communicate control structure computations from one client machine to an array of synthesis servers. The abstraction mechanisms built into OSC—a hierarchical name space and regular expression message targeting—have also proven to be useful in implementations running entirely on a single machine. In this context we have discovered a particularly valuable application of the OSC client/server model in the organization of the gestural component of control structure computations. The basic strategy is to:

- Translate all incoming gestural data into OSC messages with descriptive addresses
- Make all controllable parameters in the rest of the system OSC-addressable

Now the gestural performance mapping is simply a translation of one set of OSC messages to another. This gives performers greater scope and facility in choosing how best to effect the required parameter changes.

3. An OSC Address Subspace for Wacom Tablet Data

Wacom digitizing graphic tablets are attractive gestural interfaces for real-time computer music. They provide extremely accurate two-dimensional absolute position sensing of a stylus, along with measurements of pressure, two-dimensional tilt, and the state of the switches on the side of the stylus, with reasonably low latency. The styli

/{tip, eraser}/{hovering, drawing} *x y xtilt ytilt pressure*
/{tip, eraser}/{touch, release} *x y xtilt ytilt*
/{airbrush, puckWheel, puckRotation} *value*
/buttons/[1-2] *booleanValue*

Table 1: OSC Address Subspace for Wacom Tablets

(pens) are two-sided, with a "tip" and an "eraser." The tablets also support other devices, including a mouse-like "puck," and can be used with two devices simultaneously.

Unfortunately, this measurement data comes from the Wacom drivers in an inconvenient form. Each of the five continuous parameters is available independently, but another parameter, the "device type," indicates what kind of device is being used and, for the case of pens, whether the tip or eraser is being used. For a program to have different behavior based on which end of the pen is used, there must be a switching and gating mechanism to route the continuous parameters to the correct processing based on the "device type." Similarly, the tablet senses position and tilt even when the pen is not touching the tablet, so a program that behaves differently based on whether or not the pen is touching the tablet must examine another variable to properly dispatch the continuous parameters.

Instead of simply providing the raw data from the Wacom drivers, our **Wacom-OSC** object outputs OSC messages with different addresses for the different states. For example, if the eraser end of the pen is currently touching the tablet, **Wacom-OSC** continuously outputs messages whose address is `/eraser/drawing` and whose arguments are the current values of position, tilt, and pressure. At the moment the eraser end of the pen is released from the tablet, **Wacom-OSC** outputs the message `/eraser/release`. As long as the eraser is within range of the tablet surface, **Wacom-OSC** continuously outputs messages with the address `/eraser/hovering` and the same position, tilt, and pressure arguments.

With this scheme, all of the dispatching on the "device type" variables is done once and for all inside **Wacom-OSC**, and hidden from the interface designer. The interface designer simply uses the existing mechanisms for routing OSC messages to map the different pen states to different musical behaviors.

We use another level of OSC addressing to define distinct behaviors for different regions of the tablet. The interface designer creates a data structure giving the names and locations of any number of regions on the tablet surface. An object called **Wacom-Regions** takes the OSC messages from **Wacom-OSC** and prepends the appropriate region name for events that occur in the region.

For example, suppose the pen is drawing within a region named "foo." **Wacom-OSC** outputs the message `/tip/drawing` with arguments giving the current pen position, tilt, and pressure. **Wacom-Regions** looks up this current pen position in the data structure of all the regions and sees that the pen is currently in region "foo," so it outputs the message `/foo/tip/drawing` with the same arguments. Now the standard OSC message routing mechanisms can dispatch these messages to the part of the program that implements the behavior of the region "foo."

Once again tedious programming work is hidden from the interface designer, whose job is simply to take OSC messages describing tablet gestures and map them to musical control.

4. Dimensionality Reduction for the Tactex Control Surface

Tactex's MTC Express controller (Tactex, 2001) senses pressure at multiple points on a surface. The primary challenge using the device is to reduce the high dimensionality of the raw sensor output (over a hundred pressure values) to a small number of parameters that can be reliably controlled.

One approach is to install physical tactile guides over the surface and interpret the result as a set of sliders controlled by a performer's fingers. Apart from not fully exploiting the potential of the controller this approach has the disadvantage of introducing delays as the performer finds the slider positions.

An alternative approach is to interpret the output of the tactile array as an "image" and use computer vision techniques to estimate pressure for each finger of the hand. Software provided by Tactex outputs four parameters for each of up to five sensed "fingers," which we represent with the following OSC addresses:

- `/x` — X position on the surface
- `/y` — Y position on the surface
- `/z` — Pressure
- `/age` — Amount of time this finger has been touching the surface

The anatomy of the human hand makes it impossible to control these four variables independently for each of five fingers. We have developed another level of analysis, based

on interpreting the parameters of three fingers as a triangle, as shown in Figure 1. This results in the parameters shown in Table 2. These parameters are particularly easy to control and have the advantage of working with any orientation of the hand.

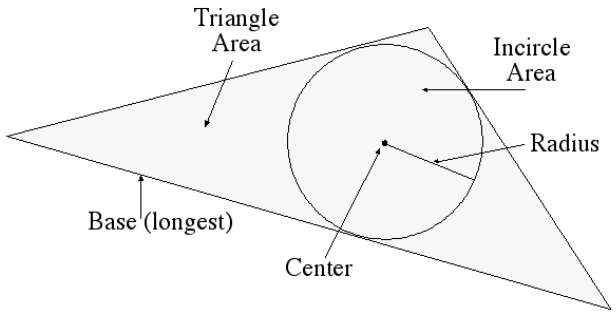

Figure 1: Parameters of Triangle Formed by 3 Fingers

5. An OSC Address Space for Joysticks

USB joysticks used for computer games also have good properties as musical controllers. One model senses two dimensions of tilt, rotation of the joystick, and a large array of buttons and switches. The buttons support "chording," meaning that multiple buttons can be pressed at once and detected individually.

We developed a modal interface in which each button corresponds to a particular musical behavior. With no buttons pressed, no sound results. When one or more buttons are pressed, the joystick's tilt and rotation continuously affect the behaviors associated with those buttons.

The raw joystick data is converted into OSC messages whose address indicates which button is pressed and whose arguments give the current continuous measurements of the joystick's state. When two or more buttons are depressed, the mapper outputs one OSC message per depressed button, each having identical arguments. For example, while buttons "B" and "D" are pressed, our software continuously outputs these two messages:

- `/joystick/b` *`xtilt ytilt rotation`*
- `/joystick/d` *`xtilt ytilt rotation`*

Messages with the address `/joystick/b` are then routed to the software implementing the behavior associated with button "B" with the normal OSC message routing mechanisms.

/area <*area_of_inscribed_triangle*>
/averageX <*avg_X_value*>
/averageY <*avg_Y_value*>
/incircle/radius <*Incircle radius*>
/incircle/area <*Incircle area*>
/sideLengths <*side1*> <*side2*> <*side3*>
/baseLength <*length_of_longest_side*>
/orientation <*slope_of_longest_side*>
/pressure/average <*avg_pressure_value*>
/pressure/max <*maximum_pressure_value*>
/pressure/min <*minimum_pressure_value*>
/pressure/tilt <*leftmost_Z-rightmost_Z*>

Table 2: OSC Messages Output from Tactex Triangle Detection

6. Mapping Incoming MIDI to OSC

Suppose a computer-music instrument is to be controlled by two keyboards, two continuous foot-pedals, and a foot-switch. There is no reason for the designer of this instrument to think about which MIDI channels will be used, which MIDI controller numbers the foot-pedals output, whether the input comes to the computer on one or more MIDI ports, etc.

We map MIDI message to OSC messages as soon as possible. Only the part of the program which does this translation needs to embody any of the MIDI addressing details listed above. The rest of the program sees messages with symbolic names like `/footpedal1`, so the mapping of MIDI to synthesis control is clear and self-documenting.

7. Controller Remapping

The use of an explicit mapping from gestural input to sound control, both represented as OSC messages, makes it easy to change this mapping in real-time to create different modes of behavior for an instrument. Simply route incoming OSC messages to the mapping software corresponding to the current mode.

For example, we have developed Wacom tablet interfaces where the region in which the pen touches the tablet surface defines a musical behavior to be controlled by the continuous pen parameters even as the pen moves outside the original region. Selection of a region when the pen touches the tablet determines which mapper(s) will interpret the continuous pen parameters until the pen is released from the tablet.

8. OSC to Control Hexaphonic Guitar Processing

We have created a large body of signal processing instruments that transform the hexaphonic output of an electric guitar. Many of these effects are structured as groups of 6 signal-processing modules, one for each string, with individual control of all parameters on a per-string basis. For example, a hexaphonic 4-tap delay has 6 signal inputs, 6 signal outputs, and six groups of nine parameters: the gain of the undelayed signal, four tap times, and four tap gains.

OSC gives us a clean way to organize these 54 parameters. We make an address space whose top level chooses one of the six delay lines with the numerals 1-6, and whose bottom level names the parameters. For example, the address `/3/tap2time` sets the time of the second delay tap for the third string. We can then leverage OSC's pattern-matching capabilities, for example, by sending the message `/[4-6]/tap[3-4]gain` to set the gain of taps three and four of strings four, five, and six all at once.

9. Example: A Tactex-Controlled Granular Synthesis Instrument

We have developed an interface that controls granular synthesis from the triangle-detection software for the Tactex MTC described above. Our granular synthesis instrument continuously synthesizes grains each with a location in the original sound and frequency transposition value that are chosen randomly from within a range of possible values. The real-time-controllable parameters of this instrument are arranged in a straightforward OSC address space:

- `/bufpos` — Avg. grain position in input sound
- `/bufposrange` — Range of possible values around `/bufpos`
- `/duration` —duration of each grain
- `/transpose` — Average transposition per grain
- `/transposerange` — Range of possible transposition values around `/transpose`

The Max/MSP patch shown in the figure below maps incoming Tactex triangle-detection OSC messages to OSC messages to control this granular synthesizer.

This mapping was codeveloped with Guitarist/Composer John Schott and used for his composition "The Fly."

10. Conclusion

We have described the benefits in diverse contexts of explicit mapping of gestures to sound control parameters with OSC.

References

Boie, B., M. Mathews, and A. Schloss 1989. The Radio Drum as a Synthesizer Controller. *Proceedings of the International Computer Music Conference*, Columbus, OH, pp. 42-45.

Buchla, D. 2001. Buchla Thunder. http://www.buchla.com/historical/thunder/index.html

Tactex. 2001. Tactex Controls Home Page. http://www.tactex.com

Wright, M. 1998. Implementation and Performance Issues with OpenSound Control. *Proceedings of the International Computer Music Conference*, Ann Arbor, Michigan.

Wright, M. and A. Freed 1997. Open Sound Control: A New Protocol for Communicating with Sound Synthesizers. *Proceedings of the International Computer Music Conference*, Thessaloniki, Hellas, pp. 101-104.

Wright, M., D. Wessel, and A. Freed 1997. New Musical Control Structures from Standard Gestural Controllers. *Proceedings of the International Computer Music Conference*, Thessaloniki, Hellas.