# Defining a control standard for easily integrating haptic virtual environments with existing audio/visual systems

Stephen Sinclair and Marcelo M. Wanderley
Input Devices and Music Interaction Laboratory
Centre for Interdisciplinary Research in Music Media and Technology
McGill University – Montreal, QC, Canada
sinclair@music.mcgill.ca, marcelo.wanderley@mcgill.ca

## ABSTRACT

This paper presents an approach to audio-haptic integration that utilizes Open Sound Control, an increasingly well-supported standard for audio communication, to initialize and communicate with dynamic virtual environments that work with off-the-shelf force-feedback devices.

## Keywords

Haptics, control, multi-modal, audio, force-feedback

## 1. INTRODUCTION

Audio and video systems have historically been very well integrated. In contrast, haptic displays are only beginning to be available to a wider audience. As the number of force-feedback controllers on the market has been increasing, there has been a corresponding interest in making use of haptics as a third sensory mode in audio/visual systems. While custom high-fidelity systems are often used in research for determining the limits of our haptic senses, there is also a need to introduce better integration of ready-made, commercial haptic devices into existing multimedia software.

Typically, creating an experimental setup for haptics research entails programming a 3D environment using some C/C++ framework. While this is powerful, it can be unnecessarily complex for simpler needs. Some tools make this easier by allowing the user to specify the environment in a description language such as VRML [6] [15]. While this approach certainly simplifies things, it does not provide an easy means to inter-operate with tools used in audio. Consequently, audio-enabled haptic demos are usually limited to simple sound file playback. (See, for example, the proSENSE HapticMusic demo.) More importantly, most systems are not designed for run-time interaction with other software. For the purpose of quickly creating interactive environments that can be easily modified, better integration with existing real-time media software is called for.

With the initial thought of creating some kind of "haptics" external for PureData [14], similar to its GEM [4] system for video, we came to the conclusion that it would be more fruitful to run a haptics simulation in a separate process, using Open Sound Control to communicate. This paper describes our reasoning behind this decision, our results to date, and some ideas we are considering for future work.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
*NIME07,* New York, NY, USA
Copyright 2007 Copyright remains with the author(s).

## 2. MULTI-MODAL COMPUTING

A rendering system for a multi-modal display is inherently separable by each of its sensory modes. While some common properties can be shared, each mode has different requirements regarding timing and data throughput. For example, while control changes should be apparent in an audio stream within 10 ms or less for a satisfying user experience [9], visual displays usually update at about 30 Hz, meaning that control changes are allowed up to 33 ms to be received and processed.

In contrast, force-feedback haptics requires the total latency be 1 ms or less. This is because input and output are directly coupled: the user is part of a closed system. The "display" depends entirely on the user's movement, and reactions to position changes must be as instantaneous as possible in order to render the feel of a hard surface. It has been previously found that between a 500 Hz and 1 kHz update rate must be maintained for a good user experience [11].

As such, there are choices to make in terms of how the system architecture will take these differences into account. On the one hand, a single fast processor can be used to perform all operations in synchrony—such an architecture usually provides minimal latency between each sensory mode. On the other, each mode can be considered independent, running with asynchronous timing or even on separate hardware, but communicating events to each other to allow a distributed approach.

The seminal TELLURIS project at ACROE [2] has shown the possibilities of strictly synchronous systems. Originally, three sensory modes were rendered on dedicated vector processors, synchronized on using an external hardware clock. They eventually moved to an SGI system, and currently their ERGOS haptic device depends on algorithms running on a PCI-based DSP board. The physical modelling for haptic and audio processes are calculated synchronously on the DSP, though the graphical display is updated independently by the PC. The Virtual PebbleBox [8], a comparison study between the TELLURIS system and a combination of available software solutions (SensAble OpenHaptics, the Open Dynamics Engine, and Microsoft Direct3DSound), used the system for a very accurate 3 kHz haptic model, with 30 kHz audio and 50 Hz video.

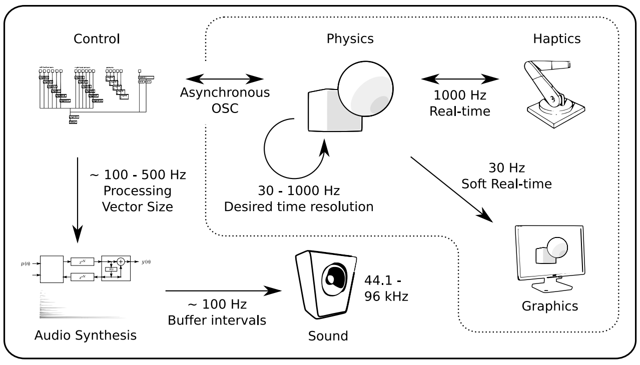


**Figure 1: The system architecture. The dotted line represents a process boundary. Within the process, each sensory mode is computed with independent timing on a separate thread.**

Another example of this approach is the Audio-Haptic Interface project [5]. A microcontroller-based system using timer interrupts to compute haptics and audio under real-time constraints was used to achieve latency below 1 ms between the two subsystems. The result was a system that enabled the authors to measure the lower bound of noticeable latency in an audio-haptic system.

One notices that these projects are specialized, require embedded programming, and are not easily scalable to larger tasks. In contrast, commercial haptic devices usually rely on the CPU as the main processing device, making use of I/O with external hardware to read the device's sensors and control the motors. The implication is that there is a lower-bound on achievable data rates, (most operating systems allow timing *just* within 1 ms), but that the computational architecture can be as flexible as a standard computer—for example, taking advantage of the growing popularity of multi-core processors [18], or even local networking to allow a tiered approach [10].

## 3. TALKING TO HAPTICS

Unlike GEM, which calculates video frames between audio computations, a similar approach would be difficult for haptics servo computation. Non-trivial changes would need to be made to the run-time system, requiring the addition of a whole new set of patch cords running messages at the haptic rate. The audio system would have to take a back seat to the haptics real-time needs.

Additionally, the presence of force-feedback haptics usually implies the existence of some kind of 3D object scene graph—while audio and video don't necessarily need to represent concrete concepts with analogies to something physical, haptics tends to require some model of "what" we are touching. Expressing complex low-level 3D mathematics in the PureData visual programming paradigm seemed to be a somewhat intractible problem. While simple shapes could certainly be implemented, we wanted to be able to express interactive scenarios using collision detection and physical dynamics. Though not necessarily impossible, we felt that Pd would be better suited to *describing* such a scene and reacting to events that occur within it.

Since a scene graph is inherently a hierachical structure, it seemed natural to take advantage of the addressing scheme in Open Sound Control [22]. As a bonus, the program implementing the scene would run in an independent process, allowing for easy parallelism without interfering with PureData's internals, and would also be interoperable with other software that uses the protocol. A diagram of the system is shown in Figure 1.

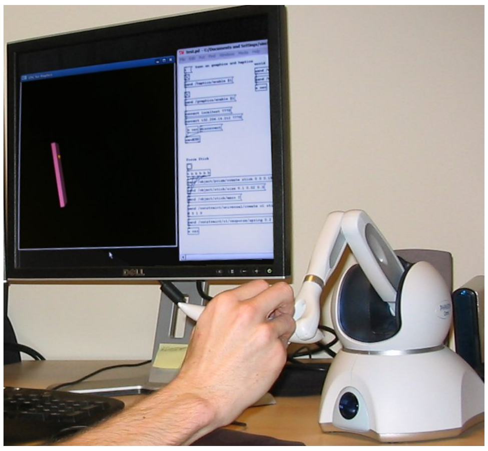


**Figure 2: Using the Phantom Omni from SensAble Technologies to interact with the Force Stick example. The PureData patch that manages the scene is visible on the right-hand side of the screen.**

The task was then to try to define a set of OSC messages that may be used for creating and communicating with a 3D scene graph, and to produce an implementation which could show examples of how this might be used to define musical interactions.

## 4. IMPLEMENTATION

A proof-of-concept implementation, called DIMPLE (Dynamic Interactive Musically PhysicaL Environment), has been created to test these concepts. A photograph of a user interacting with the system can be found in Figure 2. The CHAI 3D [3] haptic scene-graph API is used for calculating device-related forces and to communicate with the haptic display, while the Open Dynamics Engine (ODE) [17] is used for physical simulation of object movement and collision processing. It also uses LibLo [7] for OSC messaging. CHAI 3D supports a number of off-the-shelf haptic devices.

A more complete specification for the proposed OSC namespace is available [16]. The current software responds only to a subset of these messages, but work on a full implementation is currently in progress. It is available for download on the project's website[1].

## 5. EXAMPLES

The 3D virtual environment can be specified in terms of *objects* and *constraints* on these objects. For receiving feedback from the model, any parameter of an object or constraint can be told to announce its value at regular intervals, based on an internal timer. For now, only basic shapes are considered, but we found that many interesting *virtual musical instruments* (VMI) [12] can be constructed with them. Composite objects can be created by using a hierarchical naming convention. In the future, more complicated shapes

[1]http://www.music.mcgill.ca/musictech/idmil/projects/forcefeedback

will be availble by loading triangle mesh files. The specification for constraints is inspired heavily by the API for ODE.

The first example will give details about the OSC namespace that can be used to define it. The second will describe briefly another model that was implemented using the framework.

## 5.1 Force Stick

Modeled on the original Force Stick described by Verplank [21], this simple VMI consists of a rectangular prism with one end on a rotating hinge. The hinge has varying types of active feedback. For instance, Verplank suggested several effects that can be achieved: *pluck*, *ring*, *rub*, *bang*, *strike*, and *squeeze*. Using OSC, such a system can be constructed as follows:

Initialize the object (named "stick"), and specify its shape, position and mass:

```
/object/prism/create stick
/object/stick/size 0.02 0.1 0.3
/object/stick/position 0 0 0.15
/object/stick/mass 2
```

It can be seen here that once an object is created, it becomes part of the OSC namespace. It can then accept OSC methods which modify it. It is also immediately introduced into the simulation, appears on the screen, and can be touched and manipulated with the haptic device. Next, add a constraint (a hinge) located at the bottom of the prism, named "motor", with a damped spring response:

```
/constraint/hinge/create motor stick world 0 0 0 0 1 0
/constraint/motor/response/spring 20 1
```

The constraint is located at (0,0,0) and its axis points along (0,1,0), the X-axis. This is the axis around which the stick will rotate.

The constraint is defined to be between the object `stick`, and `world`, indicating a fixed position. The stiffness of the spring action is defined as 20 N·m/rad, and the damping coefficient is 1 N·m·s/rad. The object will not move in space except in rotation around the line segment defined by the given point and axis. The spring, named `motor`, will respond according to the given coefficients.

To change the behaviour of the object when it is pushed or pulled by the device proxy, a different `response` message can be sent to the `motor` constraint. For instance, to get a *squeeze* type of response, a negative linear response can be used.

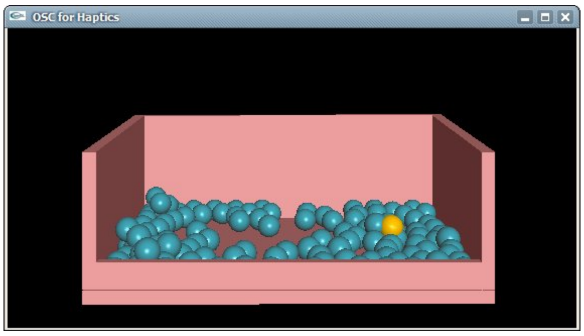
```
/constraint/motor/response/spring -10
```

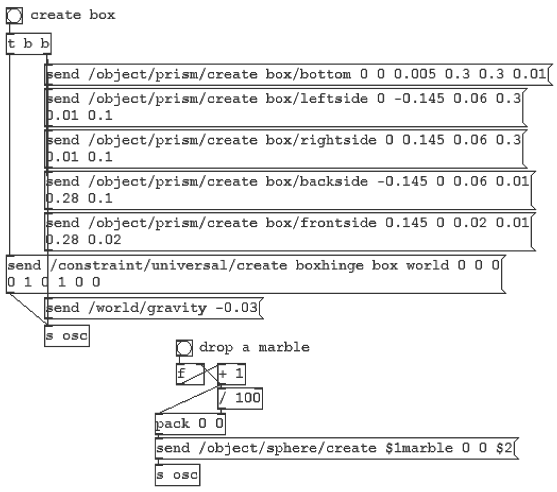
This will reverse the usual spring, so that the stick tends to fall away from the original location, and must be pulled back to the center. An object can be grabbed using the device's button. "Walls" can also be specified on the constraint so that it does not fall all the way around the hinge.

To create a sonic response, for example, by modifying the timbre or pitch of a synthesizer, the following message will indicate that the system should send messages every 30 ms:

```
/constraint/motor/force/magnitude/get 30
```

This will cause the force exerted by the stick's constraint to be sent to the audio system at regular intervals. Conversely, the force exerted *on* the stick can be retrieved by,





```
create box
t b b
send /object/prism/create box/bottom 0 0 0.005 0.3 0.3 0.01
send /object/prism/create box/leftside 0 -0.145 0.06 0.3
0.01 0.1
send /object/prism/create box/rightside 0 0.145 0.06 0.3
0.01 0.1
send /object/prism/create box/backside -0.145 0 0.06 0.01
0.28 0.1
send /object/prism/create box/frontside 0.145 0 0.02 0.01
0.28 0.02
send /constraint/universal/create boxhinge box world 0 0 0
0 1 0 1 0 0
send /world/gravity -0.03
s osc
drop a marble
f
+ 1
/ 100
pack 0 0
send /object/sphere/create $1marble 0 0 $2
s osc
```

**Figure 3: The Bucket of Marbles example, and the PureData patch that created it. Collisions between spheres trigger sounds in the audio portion (not shown). The light-coloured sphere represents the haptic proxy, which can push the others around.**

```
/object/stick/force/magnitude/get 30
```

Generally speaking, any object property can be retrieved similarly, either one time or at regular intervals.

Other constraint responses, such as textures, non-linear springs, or breakable membranes ("plucks") may be specified in a similar manner.

## 5.2 Bucket of Marbles

The PebbleBox [13] is a controller using audio analysis to detect collisions between small polished pebbles, which can be used to excite some synthesis engine, such as physical modelling of water or ice cubes.

It was used, as mentioned above, as a model for the Virtual PebbleBox, a study implemented once using the TELLURIS system, and again using a combination of software packages, including the ODE which is also used here.

A picture of an implementation in DIMPLE, using only a few messages, can be seen in Figure 3. Each part of the box is specified with a `create` message, which is hinged in place so that it is not affected by gravity. Marbles are then dropped into the box. The audio portion receives messages from DIMPLE informing it which objects collided and at what combined velocity. Objects can be pushed around using the haptic device.

The audio initially consisted of a decaying envelope applied to a low-frequency sinusoid, so that the spheres seemed to make a small "bumping" sound. However, we then forwarded these messages a proper modal synthesis algorithm, running in Max/MSP on a separate computer, which seemed to give the marbles a metallic quality.

Another non-haptic task for which we have used this model was to redirect the gravity vector according to the center of balance determined by a force-sensing floor, causing the marbles to roll to one side or another in correspondence with the user's posture. The ball positions were then used to control a spatialization algorithm.

The advantage of using OSC here was clear: in at least two cases, we were able to very quickly connect various modules we had previously made in no more than a few minutes to create new demonstrations.

## 6. DISCUSSION AND FUTURE WORK

We have shown that OSC can be used to describe a virtual instrument, constructed and modelled in a dedicated process, and to get feedback from it to inform an audio (or visual) engine.

Inter-process and inter-computer communication is necessarily slower than tightly integrated systems which share memory. Usually, OSC messages are transmitted using UDP/IP, even when communication is between processes on the local computer. This protocol may have a variety of physical transports, which may introduce latency between events in the physical model and sensory responses in the audio and visual modes. Experiments have shown that the Just Noticeable Difference (JND) for audio-haptic latency is about 24 ms [1], implying that messages must be received and processed by the audio subsystem within this time frame.

We have not performed formal tests on actual latency observed when using OSC to communicate between haptic and audio processes. Informally we have not observed problems with small numbers of objects when the two processes are communicating over a loop-back connection or on the local network. An example such as the PebbleBox simulation could be used to measure how many objects can effectively be tracked this way.

It is important to remember that OSC is a purposely verbose protocol, and that it is transport-independent. It puts clarity over efficiency, with the assumption that the actual transmission medium should be chosen to be fast enough for the intended use. One could imagine an implementation which uses inter-process shared memory, or named pipes, both of which would likely be very fast mediums on a local computer. With some changes, perhaps DIMPLE could even be compiled to an external, so as to make use of Pd/Max messaging. Designing a clear, human-readable namespace makes the program flexible and and easy to use, which overshadows any overhead derived from its verbosity.

In future work, we would like to implement many more haptic effects such as deformable meshes for scanned synthesis [20], stochastic scraping models [19], gravity wells, object "grabbing", and vibrational cues.

## 7. ACKNOWLEDGMENTS

This work was sponsored in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), the Canadian Foundation for Innovation (CFI), and the Enactive Network European Project.

## 8. REFERENCES

[1] B. D. Adelstein, D. R. Begault, M. R. Anderson, and E. M. Wenzel:. Sensitivity to haptic-audio asynchrony. In *Proceedings of the 5th International Conference on Multimodal Interfaces*, pages 73–76, Vancouver, BC, November 2003. ACM.

[2] C. Cadoz, A. Luciani, J.-L. Florens, and N. Castagné. ACROE-ICA: Artistic creation and computer interactive multisensory simulation force feedback gesture transducers. In *Proceedings of the Conference on New Interfaces for Musical Expression*, 2003.

[3] F. Conti, D. Morris, F. Barbagli, and C. Sewell. CHAI 3D. http://www.chai3d.org/, November 2006.

[4] M. Danks. Real-time image and video processing in GEM. In *Proceedings of the International Computer Music Conference*, pages 220–223, 1997.

[5] D. DiFilippo and D. K. Pai. The AHI: An audio and haptic interface for contact interactions. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*, pages 149–158, 2000.

[6] Handshake VR. proSENSE Virtual Touch Toolbox. http://www.handshakevr.com/, November 2006.

[7] S. Harris and N. Humfrey. LibLo: Lightweight OSC implementation. http://liblo.sourceforge.net/, January 2007.

[8] C. Magnusson, A. Luciani, D. Couroussé, R. Davies, and J.-L. Florens. Preliminary test in a complex virtual dynamic haptic audio environment. In *2nd Enactive Workshop*, McGill University, Canada, May 2006.

[9] T. Mäki-Patola and P. Hämäläinen. Latency tolerance for gesture controlled continuous sound instrument without tactile feedback. In *Proceedings of the International Computer Music Conference*, Miami, USA, Nov 2004.

[10] W. R. Mark, S. C. Randolph, M. Finch, J. M. V. Verth, and I. Russell M. Taylor. Adding force feedback to graphics systems: issues and solutions. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 447–452, New York, NY, USA, 1996. ACM Press.

[11] M. Minsky, O. young Ming, O. Steele, J. Frederick P. Brooks, and M. Behensky. Feeling and seeing: issues in force display. In *SI3D '90: Proceedings of the 1990 symposium on Interactive 3D graphics*, pages 235–241, New York, NY, USA, 1990. ACM Press.

[12] A. Mulder. *Design of Virtual Three-dimensional Instruments for Sound Control.* PhD thesis, Simon Fraser University, 1998.

[13] S. M. O'Modhrain and G. Essl. PebbleBox and CrumbleBag: Tactile interfaces for granular synthesis. In *Proceedings of the Conference on New Interfaces for Musical Expression*, 2004.

[14] M. Puckette. Pure Data: another integrated computer music environment. In *Proceedings, Second Intercollege Computer Music Concerts*, pages 37–41, Tachikawa, Japan, 1996.

[15] Reachin' Technology. Reachin' API. http://www.reachin.se/products/reachinapi/, November 2006.

[16] S. Sinclair. OSC for haptic virtual environments: Specification. Technical Report MUMT-IDMIL-07-01, McGill University, Music Technology Area, Feb 2007.

[17] R. Smith. Open dynamics engine (ODE). http://www.ode.org, November 2006.

[18] H. Sutter. The free lunch is over: A fundamental turn toward concurrency in software. *Dr. Dobb's Journal*, 30(3), March 2005. Available from www.gotw.ca/publications/concurrency-ddj.htm.

[19] K. van den Doel, P. G. Kry, and D. K. Pai. FoleyAutomatic: physically-based sound effects for interactive simulation and animation. In *Proceedings of the 28th annual conference on computer graphics and interactive techniques*, pages 537–544, 2001.

[20] B. Verplank, M. Mathews, and R. Shaw. Scanned synthesis. *The Journal of the Acoustical Society of America*, 109(5):2400, May 2001.

[21] W. Verplank. Haptic music exercises. In *Proceedings of the 2005 International Conference on New Interfaces for Musical Expression*, pages 256–257, Vancouver, Canada, 2005.

[22] M. Wright, A. Freed, and A. Momeni. OpenSound Control: State of the art 2003. In *Proceedings of the Conference on New Interfaces for Musical Expression*, 2003.