Binding External Interactivity to X3D

John A. Stewart* CRC Canada
Sarah J. Dumoulin† CRC Canada
Sylvie Noël‡ CRC Canada

Abstract

The VRML and X3D Standards have achieved success as a method for not only 3D Model Interchange, but also for creation of complex synthesized 3D worlds. Shortcomings in VRML and X3D exist in the areas of manipulation of aural soundscapes, and interaction via intuitive devices.

Cognitive and Computer Scientists at the Communications Research Centre, Canada, have embarked on a process of exploration to resolve these shortcomings by binding leading edge audio control software to VRML/X3D, thus using de facto standards to extend I/O control and audio data manipulation. This paper will outline the direction of these experiments.

Keywords: X3D, MIDI, External Sensors, Interaction

1 Introduction

If one observes musicians creating music with electronic devices, one will most likely observe both complex audio software, and intuitive interface devices (often referred to as *control surfaces*) in use by people who may have little experience with a conventional computer. The music industry has successfully married computers with the process of music creation and recording. This has been achieved by:

- investigating and understanding the human interface issues;
- standardizing on one protocol and one wiring methodology;
- publishing the protocol;
- actively creating software and hardware for creating and manipulating audio.

The 3D community, on the other hand, has struggled with interface issues for both sound manipulation and human interactivity. The capabilites of the VRML/X3D *AudioClip* Node have not evolved beyond the *DirectedSound* and *PointSound* nodes of VRML Version 1.1; keyboards and mice are still the de facto human interface for navigation.

Additionally, our experience with devices traditionally associated with 3D Virtual Reality navigation (such as magnetic tracking systems) has shown that they are costly, difficult to configure, and prone to failure. The music industry has been able to manufacture cost-effective "plug and play" products that are available off the shelf in cities and towns world-wide, and has spawned a growth industry in expanding the range and usage of these devices.

*e-mail: alex.stewart@crc.ca
†e-mail: sarah.dumoulin@crc.ca
‡e-mail: sylvie.noel@crc.ca

Web3D 2007, Perugia, Italy, April 15–18, 2007.
© 2007 ACM 978-1-59593-652-3/07/0004 $5.00

We will outline how we are attempting to resolve these I/O control and audio issues in this paper.

2 MIDI

MIDI [MID 2006] is the standard computerized interconnection scheme in the music industry, and has been specified as a possible data type for the *AudioClip* Node in VRML and X3D. MIDI, however, has evolved over time and is now used for general purpose control of instruments and stage effects (the MIDI Show Control extensions [MID 2006, MSC]), and in other unrelated fields for motion capture and playback. The MIDI protocol can be used over specified 5-pin DIN connectors (the MIDI industry standard interconnect scheme), over USB, or over other networking protocols such as MIDI over RTP [Biaz et al. 2005] or CRC's MVIP-II [Robinson et al. 2003].

MIDI has previously been used for interaction with VRML/X3D. For instance MidiSpace [Delerue and Pachet 1998] was created circa 1998, CRC published an interface to a MIDI music synthesizer in 2001 [CRC 2001], and the Xj3D Browser implemented a *MidiSensor* Node circa 2005 [Yum 2005]. Despite the numerous attempts to merge MIDI and VRML/X3D, no scheme has been widely adopted by the VRML/X3D user community.

Our current understanding is that this lack of adoption can be attributed to three general reasons:

- the implementation schemes were not designed to be used by persons without in-depth MIDI knowledge;
- the general application area of each scheme was too restricted to be of general use;
- the full extent of the MIDI protocol was not fully understood.

We are attempting to generate a flexible, concise, extensible mapping that will allow intuitive interaction between VRML/X3D and all types of MIDI devices.

3 ReWire

Propellerhead Software [Pro 2006a] has produced a software product called *Reason* that is a successful music creation package. Bundled into *Reason* is a protocol called *ReWire* [Pro 2006a, ReWire Software Development Kit] that is used to distribute information between different products, so that these products can share both timing information and control/audio data.

We are investigating how ReWire can be used for:

- adapting music control surfaces to enable X3D sensor and navigation interaction;
- routing 3D events to varying types of music controllers;
- offloading the *AudioClip* node operation from the 3D renderer to an autonomous ReWire device;
- device configuration management.

Reason is an intuitive application to use. Devices are added to a virtual 19" rack, and can be interconnected by turning the rack around with a keypress, then clicking and dragging from one interconnect

point to another to create inter-device routing. Figure 1 shows a Reason rack with the hardware interface, an analog synthesizer, and a MIDI sequencer in the virtual rack. Settings can be saved, and reloaded as one would expect with any complete user application.

Figure 1: Reason User Interface showing MIDI devices in the Reason Virtual Rack

Reason interconnects with external devices via a standard 5 pin DIN MIDI connector, or via USB. The interconnect protocol,Remote [Pro 2006a, Remote Software Development Kit] is designed to interface with external MIDI devices, and can be used to expand Reason's plug and play compatibility.

4 Combining X3D and MIDI

We are currently prototyping a system that allows ReWire to act as a data path in a client/server scenario. We are using the open-source FreeWRL VRML/X3D Browser [CRC 2006], Propellerhead's Reason software, and Apple OS X computers.

ReWire assumes a client/server architecture. We have created a server based on interaction code that resides in the FreeWRL package. Reason and FreeWRL act as clients to this server. Figure 2 shows the basic configuration.

The ReWire protocol is used to query and control the Reason MIDI client. This protocol allows the server to query and set MIDI parameters on the Reason MIDI client, and to handle control requests from this client. Communication with the FreeWRL VRML/X3D Browser is handled via a modified External Authoring Interface (EAI) [Marrin 1997] protocol that allows the FreeWRL client to reside on the server's computer, or to reside externally via a networked connection. The EAI protocol is being modified to allow for additional functionality required in this application as described below. Using EAI allows the server to create, manage, and destroy VRML/X3D Scene Graph Elements, allowing the server to render any required user interaction as an integral part of the FreeWRL client interface.

ReWire allows for 254 MIDI buses. A MIDI bus can address up to 16 devices on a send-only, receive-only, or send/receive basis. Each device can contain up to 128 controllers, and each controller can send/receive integer information in the range of 0 to 127 (7 bit

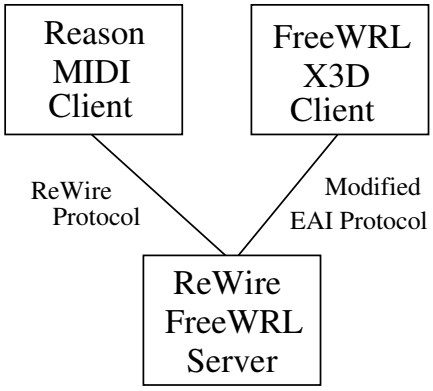

Figure 2: Reason and FreeWRL communicating via the ReWire protocol

resolution), or, by combining controllers as per the MIDI specification, 0 to 16,383 (14 bit resolution). Addressing range should not be an issue for most applications.

In our system, we abstract the detailed MIDI connections by allowing devices and channels to come into being or be removed at runtime, and by allowing routes to specify abstracted device names.

ReWire allows users to add and/or subtract devices in its *virtual rack.* These devices are assigned a unique MIDI address that is dependent on the order in which devices were added; thus the user cannot explicitly assign MIDI addresses. Device and controller information is fixed, and is dependent on the ReWire device type.

The ReWire FreeWRL server queries the ReWire bus for active MIDI busses, and for devices on any active bus. Information concerning the current Reason configuration is returned by the ReWire interface as per the following example:

```
MIDI bus "5"
        MIDI Device "Subtractor 1"
                Controller 1
                        Continuous
                        min 0
                        max 127
                        name "Mod Wheel"
                Controller 2
                        Continuous
                        min 0
                        max 127
                        name "Breath Control"
                Controller 3
                ....
        MIDI Device "Matrix 1"
                Controller 3
                        Step
                        min 0
                        max 31
                        name "Pattern Select"
                Controller 92
                        Step
                        min 0
                        max 1
                        name "Pattern Enable"
        MIDI Device "Matrix 2"
                Controller 3
                ....
```

The above example shows two MIDI Sequencers ("Matrix 1" and

"Matrix 2"); these sequencers are numbered as per their creation order, and this identification can be seen in the Reason User Interface.

The VRML/X3D nodes defined as extensions supported by the FreeWRL Browser allow routing to/from these Reason Device Names as explained below.

5 General ReWire MIDI Node

We have attempted to create MIDI Interface Nodes that are able to handle what we currently view as possible interface scenarios. Experience gained by using this Node, and Human Factors experiments will help determine whether changes to this Node are required.

Merging X3D and MIDI creates interesting design problems. The intersection set of commonality between MIDI and X3D is relatively small. MIDI expects integer numbers, with a minimum/maximum value dependent on a specific *channel target*, and these MIDI numbers can be either 7 bit, or 14 bit, depending on the *channel target* capabilities, as outlined above. X3D, on the other hand, has much of its interaction based on floating point values, with much greater resolution, and known bounds.

MIDI is also time-based; events are handled at specific time-slices based on a MIDI Time Signature. X3D browsers attempt to render as fast as required for visible uniform motion, and inter-event timing is non-deterministic.

Below is our general purpose ReWire Interface Node as it stands at the time of writing.

```
ReWireMidiControl : FreeWRLInteractiveNode {
      SFInt32  [in,out] bus         -1
      SFString [in,out] deviceName  ""
      SFString [in,out] channel     ""

      SFInt32  [out]    deviceMinVal 0
      SFInt32  [out]    deviceMaxVal 0

      SFInt32  [in,out] minVal       0
      SFInt32  [in,out] maxVal       0

      SFInt32  [in,out] intValue     0
      SFFloat  [in,out] floatValue   0.0
      SFBool   [in,out] useIntValue  TRUE

      SFBool   [in,out] continuousEvents TRUE

      SFBool   [in,out] highResolution   TRUE
      SFString [in,out] controllerType   ""
      SFInt32  [in,out] intControllerType ""
      SFBool   [in,out] devicePresent    FALSE
}
```

Items of interest in this node include the ability to bind bus/device channel/controller information at runtime. This allows the user to generically specify devices, eg, a node might be specified as:

```
DEF rw ReWireMidiControl {
        bus 5
        deviceName "Keyboard 2"
        channel "KeyPress"
        controllerType "Step"
        useIntValue FALSE
        minVal 28
        maxVal 31
}
```

And, a ROUTE from a TimeSensor would look like:

```
DEF Clock TimeSensor {
    cycleInterval 4.0
    loop TRUE
}
ROUTE Clock.fraction_changed TO rw.set_floatValue
```

This would send a series of MIDI messages, ranging from MIDI values 28 through to 31, to the *KeyPress* channel of the *Keyboard 2* device on MIDI bus 5. It should be noted that the input and output values to this node can be interpreted as floating point numbers, (useful for X3D Interpolators), or as integer numbers, which are useful as deterministic absolute values.

The *intValue* is defined as:

```
deviceMinVal <= minVal <= intValue
                and
intValue <= maxVal <= deviceMaxVal
```

Floating point values are mapped to the range:

```
0.0 <= floatValue < 1.0
```

Thus, conversion between the *intValue* and the *floatValue* occurs such that the specified ranges of these types map uniformly.

Having data values available in both integer and floating point formats has benefits for interfacing; for instance, integer values can be routed to an X3D *Switch* statement, while floating point values can be sent to an *OrientationInterpolator*.

6 Physical Interaction

We are developing a physical test-bed that will allow development and connection of different devices via the ReWire Node.

At the time of writing, we have two MIDI interconnect projects underway. These projects are:

- MIDI modules containing Ultrasonic Sensors for spatial mapping of human movement in a Laboratory;
- patchable analog input/output modules to allow interconnection of various sensors (switches, joysticks, etc.) and analog computing modules.

Currently, we have two patchable analog input/output modules completed, each module capable of 8 inputs and 8 outputs in the range of 0-10v. We also have various joysticks and switches that can be connected to these input/output modules. The Ultrasonic Sensor modules are under development, and will use dedicated 8-bit computers.

This Physical Interaction testbed has been designed to allow ease of configuration and adaptation of devices.

It is expected that experience with devices created with this physical interaction testbed will allow further refinement of our VRML/X3D methodology leading to changes to the current ReWire MIDI node.

7 AudioClip Audio Processing

We are also attempting to move the *AudioClip* functionality away from the X3D Browser; implementing it instead via an external audio sampler. This will allow state of the art audio processing to be accomplished on audio samples, and will allow the user to alter

audio samples by using additional ReWire functionality for manipulation, possibly based on the state of other X3D entities. This manipulation can happen in real-time, leading to interactivity beyond the static *AudioClip* functionality available in published standards.

8 AudioClip MIDI Control Processing

In our proposed system architecture, MIDI data within an *AudioClip* node is offloaded to a ReWire device. This will allow the creator or user of a virtual world to not only use the *AudioClip* as MIDI music that can be played at runtime, but also to use that *AudioClip* (and associated location information in the associated *Sound* node) to control interaction of real-world devices that may possibly not be audio devices.

This will allow the creator of the virtual world to easily extend the location-orientation control existing in X3D to control other devices.

9 Configuration Control

As shown above, ReWire and the FreeWRL VRML/X3D Browser are configured as clients to a ReWire FreeWRL server.

Configuration control, and mapping on individual hardware, is handled as follows:

- individual Reason data files contain device configuration and internal interconnectivity;
- individual VRML/X3D files contain textual links to exported Reason interfaces;
- the server exports audio ports and possibly locally connected devices to both Reason and the VRML/X3D Browser.

The server keeps track of the list of devices that have been requested by a running VRML/X3D program, and the list of devices exported from Reason via ReWire. It is possible, therefore, to deliver to the user indication of success and/or failure of the run-time mapping process for routing of events.

10 Human Factors Decisions

We understand that for success, development is a cyclical process. At the time of writing, we are creating the hardware and software outlined above that will allow our Cognitive Scientists to experiment and better understand user interfaces for shared 3D Virtual Reality. The results of these experiments will determine whether modifications to the software and hardware platforms are required.

11 Future Directions

While we are using the ReWire protocol for our initial work, we expect that our research will produce agnostic results whereby the abstraction layer can be bound to other protocols (such as the MVIP-II protocol [Robinson et al. 2003]) to leverage device interactivity for shared virtual worlds.

There are many questions that still need research before they can be answered. Among them are how best to handle MIDI time sequencing, how to intuitively allow manipulation of devices via VRML/X3D, and how best to abstract interfaces between diverse physical devices, such as motion capture, visual augmentation, and our analog computing modules.

We hope, by releasing this work as part of the Open Source FreeWRL VRML/X3D Browser, that others will experiment and build on the ideas presented in this paper.

12 Conclusion

We expect that our research into binding the graphical powers of X3D with the hardware and control of Audio/MIDI specific programs will bring advantages to both the music creation/manipulation and the 3D Virtual Reality application areas.

References

BIAZ, S., CHAPMAN, R. O., AND WILLIAMS, J. P. 2005. Rtp and tcp based midi over ip protocols. In *ACM-SE 43: Proceedings of the 43rd annual southeast regional conference*.

CRC CANADA. 2001. *FreeWRL EAI MIDI Synthesizer Interface*. 3701 Carling Avenue, Box 11490, Station H Ottawa, On K2H 8S2 Canada. http://www.crc.ca; http://freewrl.cvs.sf.net/freewrl/freewrl/freewrl/tests/Synth.

CRC CANADA. 2006. *FreeWRL Browser*. 3701 Carling Avenue, Box 11490, Station H Ottawa, On K2H 8S2 Canada. http://www.crc.ca; http://freewrl.sf.net.

DELERUE, O., AND PACHET, F. 1998. Midispace, un spatialisateur midi expérimental. In *JIM 98*.

MARRIN, C. 1997. External authoring interface reference. Web3D Consortium, http://www.web3d.org.

MIDI MANUFACTURERS ASSOCIATION INCORPORATED. 2006. *MIDI Specification*. P.O. Box 3173 La Habra CA 90632-3173 USA. http://www.midi.org.

PROPELLERHEAD SOFTWARE. 2006. *Propellerhead Software*. Rosenlundsgatan 29c, S-118 63 Stockholm, Sweden. http://www.propellerheads.se.

PROPELLERHEAD SOFTWARE. 2006. *ReMote Software Development Kit*. Rosenlundsgatan 29c, S-118 63 Stockholm, Sweden. http://www.propellerheads.se.

PROPELLERHEAD SOFTWARE. 2006. *ReWire Software Development Kit*. Rosenlundsgatan 29c, S-118 63 Stockholm, Sweden. http://www.propellerheads.se.

ROBINSON, J., DUMOULIN, S., AND STEWART, J. 2003. Mvip-ii: a protocol for enabling communication in collaborative virtual environments. In *Web3D '03: Proceeding of the eighth international conference on 3D Web technology*.

YUMETECH INCORPORATED. 2005. *Xj3D Browser*. 999 Third Avenue, Suite 3800 Seattle, WA 98104-4023 USA. http://www.yumetech.com; http://www.xj3d.org.