

MetaMuse: Metaphors for Expressive Instruments

Ashley Gadd, Sidney Fels

Human Communication Technologies Laboratory
Department of Electrical and Computer Engineering
2356 Main Mall, Vancouver, BC, V6T 1Z4
{gadd, ssfels}@ece.ubc.ca

ABSTRACT

We explore the role that metaphor plays in developing expressive devices by examining the MetaMuse system. MetaMuse is a prop-based system that uses the metaphor of rainfall to make the process of granular synthesis understandable. We discuss MetaMuse within a two-axis transparency framework that can be used as a predictor of the expressivity of musical devices. Metaphor depends on a *literature*, which forms the basis for making transparent device mappings. In this context we evaluate the effect of metaphor in the MetaMuse system.

Keywords

Expressive interface, transparency, metaphor, prop-based controller, granular synthesis.

INTRODUCTION

This paper presents MetaMuse, a prop-based instrument (shown in Figure 1) that uses the metaphor of rainfall to make the process of granular synthesis understandable. The sound of rainfall in nature comes from a process very similar to that of granular synthesis: short, discrete sound samples layered and overlapped to create a gestalt. The choice of metaphor for this instrument is best understood in the framework of “transparency,” a concept intended to facilitate the creation of expressive instruments.

We identify transparency as a quality of mappings. Similar to Moore’s [9] notion of control intimacy, transparency provides an indication of the psychophysiological distance, in the minds of the player and the audience, between the input and output of a device mapping. The more transparent the mapping is, the more expressive the device can be. The degrees of mapping transparency for the player and audience form orthogonal axes of a graph, shown in Figure 2, into which devices can be placed. The position of a device in the graph acts as an indicator of its expected expressivity. New technologies, often being poorly understood, tend to sit in the opaque corner of the graph. Metaphor is one technique to facilitate moving from an opaque mapping to a transparent mapping.

Metaphor enables instrument designers, players, and audi-



Figure 1: MetaMuse is controlled by two props: a watering can and a palette.

ence members to refer to cultural bases or elements that are “common knowledge,” which we call *literature*. By grounding a mapping in the literature, it is made transparent to all parties. Metaphor restricts and defines the mapping of a new device. Through metaphor, transparency increases, making the device more expressive.

RELATED WORK

Metaphor has been used in previous instruments. The SqueezeVox [3] and Accordiatron [7] both use the concertina as their control metaphors; SqueezeVox applies the metaphor to breath control, while Accordiatron remaps the control gesture space to MIDI control sequences. BoSSA [1] is also based on a traditional instrument, the violin. It increases the control space of the violin to include bending the neck. The eviolin [6] expands the control space of a violin in a different way, using the instrument’s position and orientation to control sound post-processing.

The metaphor used in Tangible Sound 2 [8], that of running water, is very similar to that of MetaMuse, but Tangi-

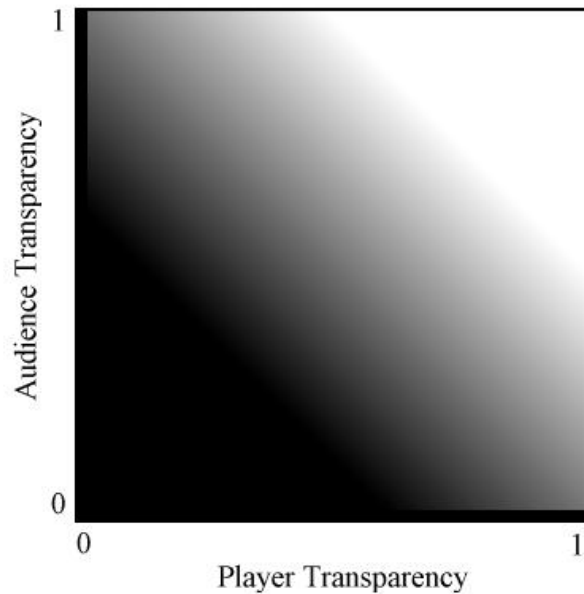


Figure 2: Axes of transparency are defined for the player and the audience. Instruments that are transparent for both are more likely to be expressive.

ble Sound 2 uses real water to produce musical sounds. The link between the fluid motion of water and the fluid motion of music is an interesting one.

Granular synthesis, described by Truax [11], blends short, overlapping sound samples to create a gestalt sound, which can be quite different from the original samples. Current controllers for granular synthesis abstract away the details of the synthesis engine, however, controlling the initiation of each granule by using high-level statistical parameters. The granular synthesis example provided with jMax, which was later made into a jMax object by IRCAM, is a good example of such a controller. The player and audience have no understanding of the process underlying the sound creation, creating opacity in the mappings of such devices.

IMPLEMENTING A METAPHOR

MetaMuse follows the rainfall metaphor as a mapping appropriate for granular synthesis. Props and virtual water are used to support the metaphor of the process of rainfall. The props represent a source and a sink for the water drops, while the virtual water falls under a simple gravity model when the source is activated. If the water intersects the sink then granules, prerecorded sound samples of water drops, are initiated in the synthesis engine. The props, then, are used to control the parameters of the falling water.

Two props are used in MetaMuse: a watering can and a flat

palette, as shown in Figure 1. The watering can is the source of the virtual water, and it affords the creation of water drops through the motion of pouring. The palette is a sink for the virtual water, and it creates a surface on which the drops can land. The drops behave like real rain, falling from the spout and hitting the surface.

MetaMuse is implemented in C and jMax [4], with a calibration GUI in Tcl/Tk [10]. The physical simulation of the water drops is implemented in C and uses a simple physics model. Polhemus Fastrak sensors are mounted on the props to provide position and orientation information to the model through a serial port library. The model is updated in real time, and is rendered using the OpenGL libraries. The visualisation is implemented to assist in debugging and calibration, and also helps familiarise novice players with the physical model of the system. It is not required for experienced players as the metaphor provides an understanding of how the water flows from the watering can.

The rainfall metaphor is discussed in detail in the following section; implementation details for the three major components of the system are described thereafter.

A Combination of Metaphors

MetaMuse combines two basic metaphors: pouring from the source, and rainfall on the sink. Pouring is controlled by the watering can, with position, orientation, and tilt affecting the flow of water droplets. Each droplet hitting the sink causes an individual granule of sound to be played. The system also incorporates a third metaphor, that of a virtual landscape, so that moving the droplets across the landscape changes the sound of the rainfall.

The pouring metaphor is the most natural of the three; the act of pouring transfers directly to the virtual watering can. An accumulator increases with a rate that is proportional to tilt, and a droplet is released whenever the value exceeds a fixed constant. This allows the player to control the flow down to the rate of a dripping faucet without restricting high flow at large tilt angles. The droplets are released from the tip of the virtual spout, and begin with a velocity that depends on the horizontal orientation of the can; the water droplets form an arc as expected. Finally, a small, random offset is applied to the initial velocity so that the flow will spread out as though a rose¹ were attached to the spout of the can.

After falling under virtual gravity, the droplets may intersect the sink, the position of which is controlled by the palette prop. This metaphor is also well-based in the literature, with the two main controls being height and flow. The higher the watering can is above the palette, the farther the droplet falls and the greater its relative velocity on hitting the palette.

¹ A common gardening attachment similar in function to a shower head.

This corresponds to an increase in the playback volume of the resulting granule. Greater flow results in more granules being played concurrently, increasing the perceived strength of the rainfall sound.

The third metaphor, that of the virtual landscape, is less successful. The intention was to have a range of surface compositions represented on the palette. Then, depending on where the droplet lands, a different sample is played to reflect that surface. This would allow the player to move between the sounds of rainfall on foliage, standing water, and bare rock, and is the reason behind the green, blue, and red colour pattern seen on the palette in Figure 1. However, the continuous nature of the pattern implies that continuous control of the sample is available, contrary to the discrete nature of the pre-recorded samples. This causes a mismatch of expectation and output, making the metaphor ineffective. The current mapping mixes three samples for each granule, as described below.

Implementation Overview

There are several controllable parameters in the synthesis engine. The choice of sample, sample rate, and sample volume can all be controlled. Post-processing is also possible, but is not implemented in this version of MetaMuse. The ways in which the parameters are mapped to the controller are dictated by the metaphor.

Physical Model and Graphics

The physical model and graphics component of MetaMuse runs in a multithreaded environment written in C. An interaction library for the Fastrak implements a callback, which runs for each record received. The main thread registers the callback with the Fastrak library to update the physical model with each record received, then enters the OpenGL graphics loop, `glutMainLoop`. This allows the graphics to update at a slower rate than the physical model.

The scene is rendered as shown in Figure 3. The main components are the watering can, the landscape, and a number of water droplets seen flowing between them. The scene is rendered from the same data used by the physical model, which updates the positions of the components underneath. The graphics update as quickly as `glutIdleFunc` will allow, quickly enough for smooth motion.

The physical model updates at 60Hz, the sampling rate of the Fastrak, which is fast enough to produce a smooth flow of water. The model is processed using the Simple Geometry Library, from Steve Baker's PLIB [2]. `sglib` provides functions for manipulating vectors, coordinate frameworks, and transformation matrices, and was originally written to facilitate OpenGL graphics routines. We implemented a function that translates the position and orientation information received from the Fastrak into `sgCoords`, taking into

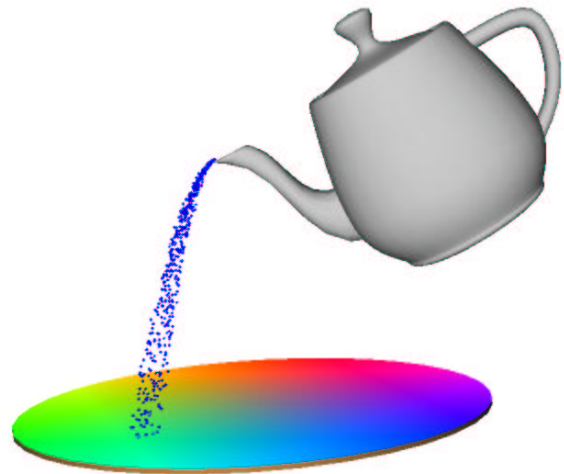


Figure 3: The scene is rendered with a watering can, a landscape, and a number of droplets.

account the Euler angles used by the Fastrak. This information is then used to determine the position and orientation of the two props.

Droplets are produced as described above. Their changes in position, due to gravity, are updated with the rest of the physical model during the Fastrak callback. Each droplet is also transformed from world coordinates to landscape coordinates and tested for intersection. Droplets are removed from the system after a timeout or when they hit the landscape, with the relative positions and velocities of such droplets being sent to jMax by a UDP socket. The jMax `udpsend` object was adapted for use outside of the jMax environment, and is used to compose a UDP packet containing the position and velocity of the droplet. This packet is then sent to the jMax synthesis engine.

jMax Synthesis Engine

The jMax synthesis engine is based on reading samples from sample tables. The parameters received from the physical model are mapped to synthesis parameters and passed to one of many voxels, or voice elements, using the `voxaloc` object. The selection of sample and the sample rate for the granule are passed to the voxel, which plays the granule using `sampread`. The top-level jMax patch is shown in Figure 4.

UDP packets are received by jMax's `udpreceive` object and separated into position (`dx`, `dy`, `dz`) and velocity (`vx`, `vy`, `vz`) parameters. Position and velocity have different effects. `dx` and `dy`, indicating the position of the droplet on the land-

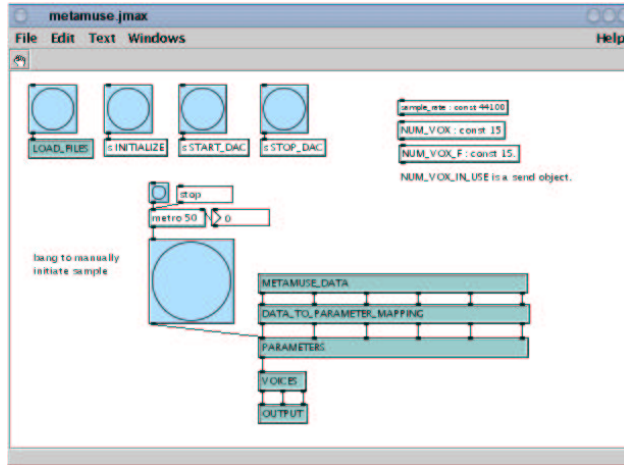


Figure 4: The top-level patch in jMax provides a series of bangs for initialisation, as well as a bang to manually initiate a granule for testing.

scape, are used to determine the “surface” on which the droplet landed. For each granule, three samples are mixed to represent different surfaces, with their relative volumes depending on the droplet’s distance from three (unmarked) points on the landscape. This is non-optimal; ideally only one sample would be played for each granule. dz is a measure of how far into the landscape a droplet fell before the collision was recognised. As such it is a function of the update rate of the system and is not used in the mapping.

The velocity parameters are used to determine the squared velocity of the droplet, which is mapped to volume. This means that droplets that fall farther will sound louder. The horizontal velocity, composed of v_x and v_y , is compared to the total velocity; this value is used to vary the playback rate of the granule, effectively varying pitch. This mapping is an arbitrary attempt to assign meaning to the tilt of the landscape.

Pitch and volume are also varied depending on the number of voices in use. Pitch, or playback rate, is given a random variation that increases with more voices. This gives the impression of different granules being played, even though similar granules are initiated by many adjacent droplets. Volume is also increased with number of voices, but not linearly. In order to allow single drops to be heard while preventing digital clipping with many voices, the volume increases as a sigmoid of the number of voices. A sigmoid object was created in jMax to allow this mapping.

Tcl/Tk Calibration GUI

A Tcl/Tk GUI is provided for calibrating the Fastrak sensor offsets and controlling the physical model. The GUI, shown

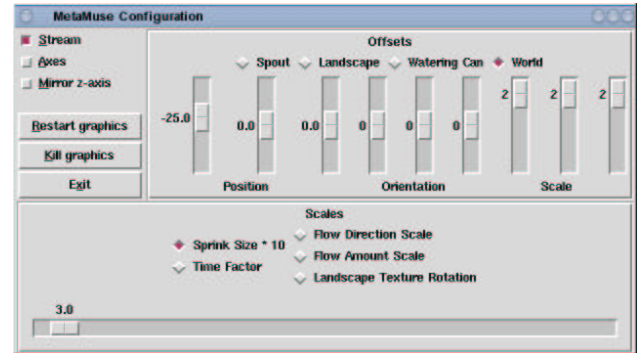


Figure 5: The calibration GUI includes control widgets for Fastrak offsets and model scaling parameters.

in Figure 5, has three main components: program control buttons, Fastrak parameters, and model parameters. Commands are sent to the C engine using a Tcl socket library written by the authors.

The program control buttons, in the top left, allow the user to start and stop the graphics and model or exit the program. Radiobuttons are also provided to turn Fastrak streaming on and off, show coordinate axes, and mirror the z axis. Displaying axes is very useful during calibration as it identifies the difference between the Fastrak sensor positions and the virtual prop positions.

The Fastrak offsets in the top right of the window provide position, orientation, and scaling calibration of the world, the props, and the spout. Selecting a radiobutton at the top changes the function of the sliders, which are then used to control the parameters. For example, the watering can position offset can be adjusted to account for the mounting position of the sensor on the can – the sensor is mounted on the bottom of the can, not in the centre.

The bottom of the window provides access to several parameters used by the model or the graphics engine. These include the rendered droplet size (for visual effect only, it has no effect on the model) and a flow multiplier to change the number of droplets produced. A factor affecting the passage of time is also present, allowing droplets to fall in slow motion.

ANALYSIS AND RESULTS

MetaMuse has been implemented as described above. Though no formal user testing has been completed, informal evaluation has illustrated some advantages and disadvantages of the system. Several people of varying backgrounds have played the device, including human-computer interface researchers, musicians, and non-technical non-musicians. Subjects provided feedback on their experiences. Audience feed-

back was not a priority at this stage of the research, so only a little was gathered.

Subjects reported that the metaphors of pouring and falling rain are very intuitive and aid in the understanding of the granular synthesis process. These aspects of the mapping are shown to be transparent. However, the metaphor breaks down when players try to vary the position on the landscape. The output does not vary as expected when players pour water onto the different areas of the landscape. This indicates that the implementation of this component of the mapping is insufficient.

This shortcoming is understandable and, in retrospect, could have been predicted. The range of control gestures that vary droplet position is continuous. However, the selection process for the granules is more discrete; it simply chooses between three different source samples. The mixing of these three samples in the intermediate regions is an insufficient interpolation method, and the resulting sound is not what the player expects. It would be preferable to be able to select from a continuous range of samples, but this is not technically possible with the current computing resources. It may be possible to create the appearance of a continuous range of samples by using post-processing or by synthesising the samples in real-time with some other synthesis technique.

This problem demonstrates a shortcoming of metaphors. The player (and the audience) expects the system to adhere to the metaphor very strictly. When the system deviates, it can cause greater opacity than a system with no metaphor at all. This is because an expectation is created by the metaphor, but the system behaves against that expectation. Metaphor can restrict a system that could otherwise explore new control interfaces. It can also confuse the player and the audience when the sound interface cannot be adequately created because of technical constraints.

CONCLUSIONS AND FUTURE WORK

We introduced MetaMuse, a prop-based instrument that uses a rainfall metaphor to make the process of granular synthesis understandable. The metaphor of rainfall is broken down into three parts: pouring water, falling rain, and varying landscape. We saw that the first two of these were well-matched to the props and the physical model used, making the instrument intuitive and easy to control. However, we found that the third component of the metaphor, varying landscape, was not well supported by the technology of granular synthesis in our implementation. This result demonstrates the need for discretion when applying metaphors to new instrument mappings.

To better understand MetaMuse and the conditions that make it expressive, we introduced a two-axis transparency scale. Our need for this framework stems from the desire to de-

sign and build new instruments for musical expression. We want our framework to facilitate the acceptance of novel controllers into the literature to allow for new forms of expression. From this perspective, we suggest that metaphor helps both the player and the audience make the mapping of a musical device transparent, hence making the device itself expressive.

Granular synthesis is a complex synthesiser with many controls that are not obvious, potentially providing a very expressive sound engine. Ad hoc controllers for this engine tend to be opaque for both player and audience. We believe that MetaMuse provides a basis for exploring new interfaces based on metaphor. MetaMuse can provide a more transparent, and therefore more expressive, controller for granular synthesis.

ACKNOWLEDGEMENTS

The authors would like to thank the members of the Human Communication Technologies Lab for their continued support. We would also like to thank ATR for their generous funding. Tom Calvert, Teena Carnegie, and Kenji Mase all provided invaluable discussion in the formation of these ideas.

REFERENCES

1. C. Bahn and D. Trueman. interface: electronic chamber ensemble. In *ACM SIGCHI Workshop on New Interfaces for Musical Expression*, page electronic proceedings, Apr 2001.
2. S. Baker. Plib, at <http://plib.sourceforge.net>, 2002.
3. P. Cook and C. Leider. Squeezevox: A new controller for vocal synthesis models. In *Proceedings of the International Computer Music Conference (ICMC 2000)*, Berlin, 2000.
4. I. de Recherche et Coordination Acoustique/Musique (IRCAM). <http://www.ircam.fr/produits/logiciels/jmax-e.html>, 2002.
5. A. Gadd and S. S. Fels. Metamuse: a novel control metaphor for granular synthesis. In *Proceedings of the ACM Special Interest Group on Computer Human Interaction (SIGCHI'02)*, page to appear, Apr 2002.
6. C. Goudeseune, G. Garnett, and T. Johnson. Resonant processing of instrumental sound controller by spatial position. In *ACM SIGCHI Workshop on New Interfaces for Musical Expression*, page electronic proceedings, Apr 2001.
7. M. Gurevich and S. Muehlen. The accordiatron: A midi controller for interactive music. In *ACM SIGCHI*

Workshop on New Interfaces for Musical Expression,
page electronic proceedings, Apr 2001.

8. K. Mase and T. Yonezawa. Body, clothes, water and toys – media towards natural music expressions with digital sounds. In *ACM SIGCHI Workshop on New Interfaces for Musical Expression*, page electronic proceedings, Apr 2001.
9. F. R. Moore. The dysfunctions of midi. *Computer Music Journal*, 12(1):19–28, Spring 1988.
10. J. K. Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley, New York, 1994.
11. B. Truax. Real-time granular synthesis with a digital signal processor. *Computer Music Journal*, 12(2):14–26, 1988.