sGlobject Trumpet & Tambourine (sGloTaT)

Ryan Brotman
Arts, Media & Engineering
Arizona State University
810 S. Forest Mall
Tempe, AZ 85281 USA
Ryan.Brotman@asu.edu

Byron Lahey
Arts, Media & Engineering
Arizona State University
699 S. Mill Ave., Room 395
Tempe, AZ 85281 USA
Byron.Lahey@asu.edu

Rebecca Stern
Arts, Media & Engineering
Arizona State University
699 S. Mill Ave., Room 395
Tempe, AZ 85281 USA
Rebecca.Stern@asu.edu

Abstract
The sGloTat sonic environment allows participants to originate sound by moving physical objects. It encourages novice users to play by naturally gesturing with two tangible user interface objects emulating a trumpet and tambourine. In this sound space, movements by the users generate visual feedback projected on the floor in the form of a three-dimensional rendering of a cone.

Keywords
Embodied interaction, tangible user interface, physical instrument models.

Introduction
The sGloTat sonic environment allows participants to originate sound by moving physical objects. It encourages novice users to play by naturally gesturing with two tangible user interface objects emulating a trumpet and tambourine. These tangible interfaces are wireless handheld devices embedded with various sensors for capturing real-time motion information. We also used sensing input to produce visual feedback. This visual feedback can enhance the users understanding of their virtual instrument and simultaneously act as a theatrical component for viewers. In this sound space, movements by the users

generate visual feedback projected on the floor in the form of a three-dimensional rendering of a cone.

Two related projects came to mind when considering our goals. One is the Delay Lama [1], a Virtual Studio Technology (VST) instrument. VST instruments have a graphical user interface for manipulating audio generation in a natural way. The Delay Lama generates a sustained tone similar to a monk chanting, and displays a 3D model of a cartoon monk whose mouth moves with the change in sound properties adjusted with simple controls. This is an intuitive and engaging interaction.

Another related project is the MIDI Ironing Board by Ranjit Bhatnagar [4]. This is a synthesized musical instrument interfaced by ironing on an ironing board with embedded heat sensors. The user finds an intuitive understanding of the relationship between the performed actions and the sounds they make, creating as quirky, fun gesture experience.

The following sections will describe the sensing, modeling, and feedback involved in this project. First we will discuss the paradigm of physical instrument models in an interactive space, followed by the description of our tangible interfaces and their senor signal processing. Next we discuss how physical movements of the user control sounds in the environment. We will conclude with a section on classifying physical movements to correspond with different musical dynamic intentions, followed by our observations of using and adjusting our own interactive experience.

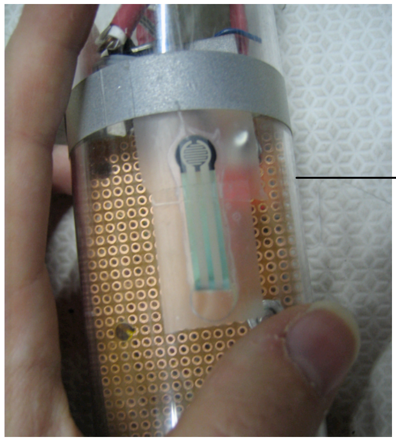
Musical Instrument Models in 3D space

The sGloTat sonic environment allows participants to originate sound by moving physical objects. The choices for sonic output may range from musical samples that can be easily cued by a user to complex physical instrument models that may be very expressive but more challenging to learn to play. Using computational instrument models that mimic real-world musical instruments, as we chose, creates a natural understanding of how to manipulate or "play" that instrument in an embodied environment such as SMALLab. By choosing these outputs thoughtfully, we can tailor the interaction for an optimal experience for novice and experienced musicians.

We also used sensing input to produce visual feedback. This visual feedback can enhance the users understanding of their virtual instrument and simultaneously act as a theatrical component for viewers. In the sGloTat experience, users see a virtual 3D cone projected on the floor. The cone changes shape based on the sounds of the virtual trumpet. The cone mimics a cartoon megaphone, forming an engaging sound visualization for the user. As the user moves about the space, the cone moves to keep the narrow end at the user’s feet. This reinforces the idea that the user is the originator of the sound.

Tangible Interfaces

Tangible user interfaces (TUIs) support embodied interaction and encourage play, especially in a musically sonified environment. We have designed two smart objects for use with this system. The sensor glow stick or sGlowStick is an 8-inch tube easily held in one hand (see figure 1) with a pressure sensor used, in this specific implementation, for sound activation and volume control.



The pressure sensor is activated easily by the thumb or forefinger while holding the sGlowStick.

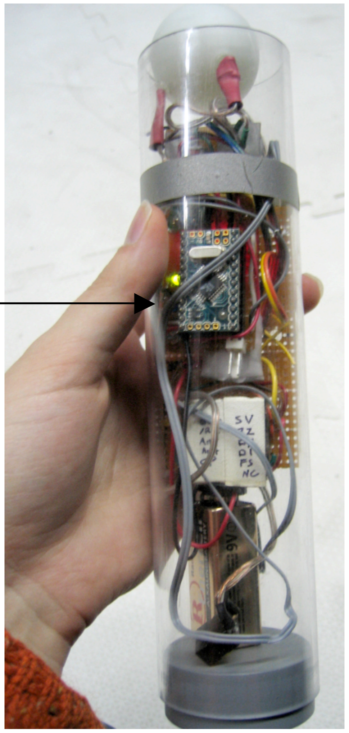


figure 1. The sGlowStick in hand. The orange light diffusion unit is turned off, at the top of the unit.

The sensor glow Bop-it or sGlowBop-it is a modified Bop-it toy [3] that consists of two handles outlining a center mass, similar to an aircraft steering wheel (see figure 2). Both smart glow objects (sGlobjects) contain an Arduino microcontroller, two-axis accelerometer, BlueSmiRF Bluetooth module, and battery. The motion tracking system in SMALLab, a mixed-reality environment [5], uses color detection and infrared (IR) detection to track different objects in its space. Consequently, the sGlobjects also contain a light diffusion unit made of three super-bright orange (sGlowStick) or green (sGlowBop-it) light-emitting diodes (LEDs) and IR-reflective tape.

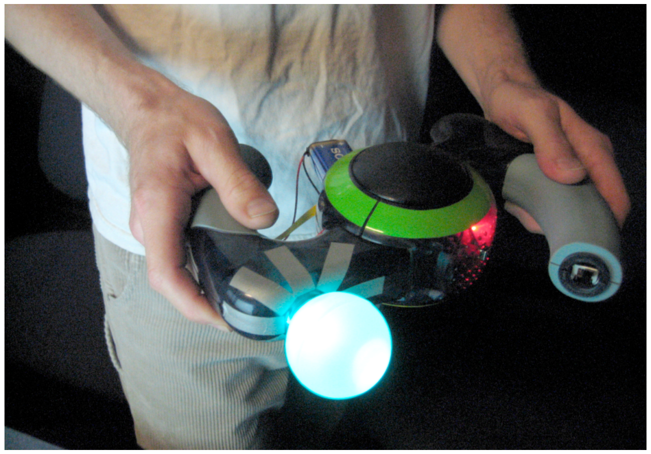


figure 2. The sGlowBop-it with green light diffusion and strips of IR-reflective tape. The red light inside indicates that the Bluetooth module is connected to the computer.

We chose to use a trumpet instrument model for use with the sGlowStick. It’s positioning determines both the pitch and vibrato expression in the instrument. The sGlowBop-it naturally affords shaking like a tambourine, so we mapped its motion to activate a shaker-like instrument model whose pitch and volume are adjusted by rotating and shaking the object. This gives a high level view of the system: tangible objects with sensors allowing a participant's motions to generate sounds and visual elements. We will now go more in depth, describing the sensor signal processing, the relationships of the gestures and the sounds and the modeling of sounds and visual elements.

Sensor Signal Processing

We used data from accelerometers (one in each tangible interface object) and a pressure sensor (in the sGlowStick) for our system. In this section we will describe the signal path and processing of the data from the sGlowStick. The path and processing is identical for the sGlowBop-It until the final stages when the data is used to control the sound and visuals.

The data from the sensors is first received by the Arduino microcontroller. The pressure sensor is wired to an analog pin of the Arduino. The Arduino has analog to digital converters built in which translate the analog values (0 – 5 volts DC) into integer values. The accelerometer outputs pulse width modulation (PWM) signals. These signals vary based on the sensed accelerations. To measure these signals it is necessary to wire the accelerometer outputs to digital input pins on the Arduino and calculate the time of the pulses. The raw value of these pulses is an integer that increases or decreases in proportion to the measured accelerations on the two axes.

The accelerometer and pressure data are transmitted as serial data over a wireless Bluetooth signal to the host computer. The host computer runs a Max/MSP/Jitter patch that receives this serial data. This patch (Instrument_SCREM_DuoPort_03.mxt) has two primary functions. The first function is to simply receive the serial sensor data from the microcontrollers and pass it on to other programs that will use this data.

The second function is to normalize this data so it can be used in a standardized form for various programs (different instruments, visualizations, games, etc.). The normalization process scales the incoming values to a range varing between 0.0 and 1.0. (floating point values). To generate these normalized values, it is necessary to determine the minimum and maximum values of the input from the sensors.

We accomplish this in our program by activating a calibration function that passes data from the sensors through objects that identify the minimum and maximum values. These minimum and maximum values are then used as variables for the scaling. When the system is initially activated, the user presses on the pressure sensor and moves the object through a full range of acceleration movements. Once these calibrations are done they can be stored and reset as needed to accommodate different sensor devices or other variations that might change the data input. This ensures that, at the beginning of the experience, the sensor infrastructure will respond optimally for its specific user.

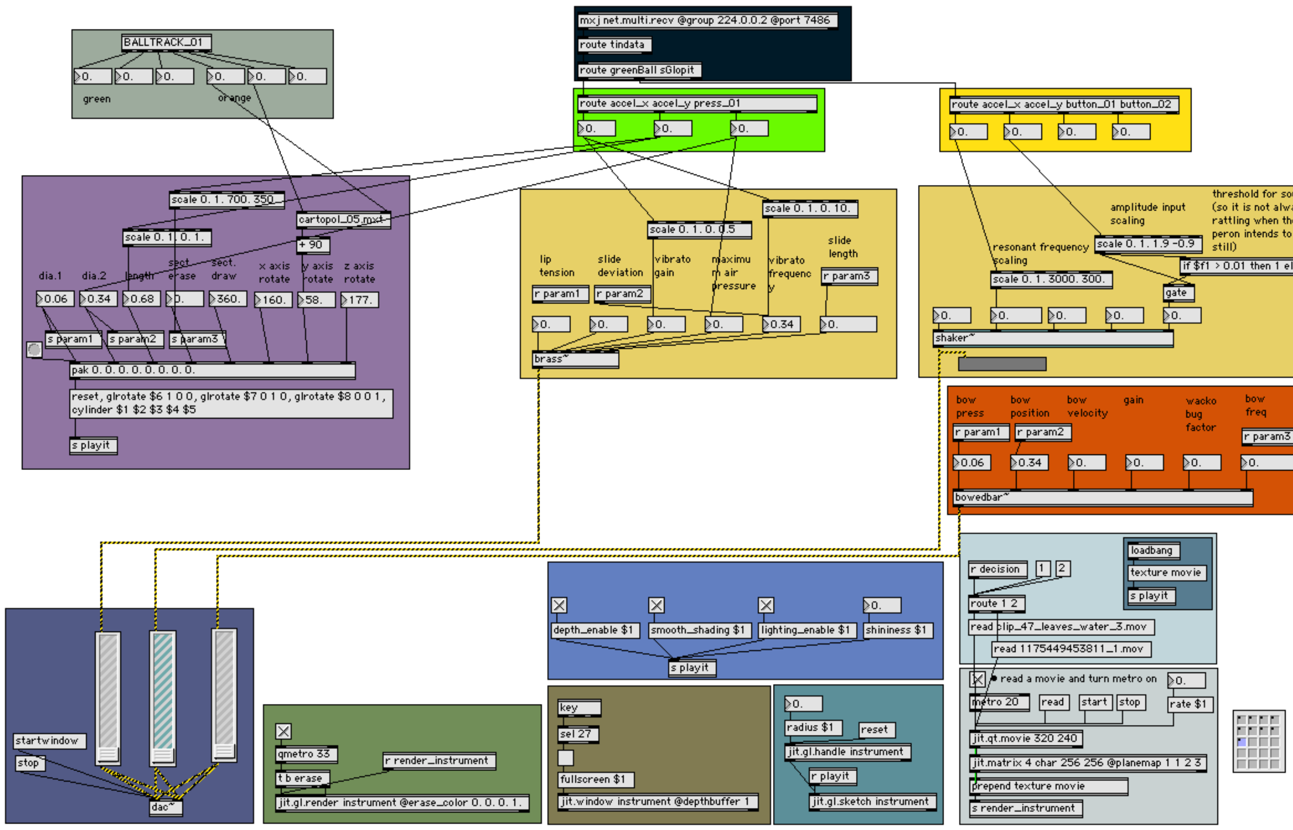


figure 3. The synthesis intrument Max/MSP patch.

At this point in the signal processing chain we can see that the sensors have been read by the Arduino, the Arduino has transmitted sensor values to the Max-MSP patch, the Max-MSP patch has normalized the sensor outputs and finally these values have been sent out for use by other programs (see figure 3).

One program that uses this sensor data is our Max-MSP Class_model patch. This program uses the sensor data to define classes based on the character of the movements. The function of this program will be described in detail in the section, Texture Modeling.

We will continue this section by describing how the sensor data is used to generate the sounds of the instruments and basic visual feedback.

The two instruments used for this system were a brass instrument and a shaker instrument. These instruments are both from the PerColate Max/MSP library [3]. Both of these software instruments generate sound using synthesis techniques with control variables based on physical parameters of the real world physical instruments. The sensor data is scaled to match usable ranges for the various parameters to which they are tied.

The brass instrument model has as its inputs the following parameters: lip tension, slide deviation, vibrato gain and frequency, air pressure, and the length of the brass tube. We controlled the vibrato gain and vibrato frequency with the X acceleration value, the length of the brass tube with the Y acceleration value and the air pressure with the pressure sensor. The other parameters we manually set and left constant for the interaction.

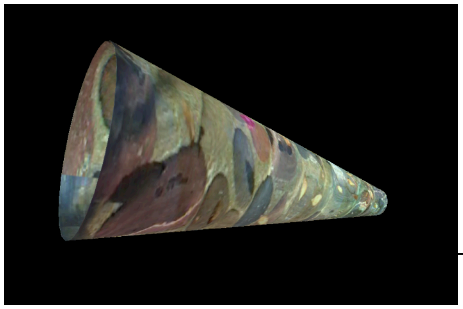
The shaker instrument model has as its inputs the following parameters: the number of beads, the amount of damping, the energy in the shaker, and the resonant frequency. We controlled this instrument with input from the sGlowBop-It and used the X acceleration value to control the resonant frequency and the Y acceleration value to control the energy or amplitude of the shaker. We filtered the Y acceleration input into the shaker model to ignore very subtle shaking. The shaking had to cause an acceleration that exceeded a threshold we defined to pass through.

Finally, we used the sensor inputs to generate visual feedback in the form of a 3-D rendering of a cone. This was done by sending OpenGL drawing commands using sensor values as variables.

We used the Y acceleration of the sGlowStick, the same variable that controlled the length of the brass tube (and therefore the frequency of the brass instrument), controls the length of cone. The pressure sensor, which also controls the air pressure of the brass instrument, is used to control the diameter of one end of the tube. The net effect of these controls is that one can create sound and visuals that have a strong visual correspondence to one another. We considered having the visual model have a direct correspondence to the physical form that would match this instrument model but instead elected to go for a more expressive, cartoon like model with more emphasis on dramatic theatrical visualizations rather than on a model that might be more applicable to an educational goal.

Gestures and Sound

One of the most immediate design decisions that we made involved mapping a change in pitch to a particular motion that a user might make.



A digital rendering of the calm river rock image mapped to the sound cone.

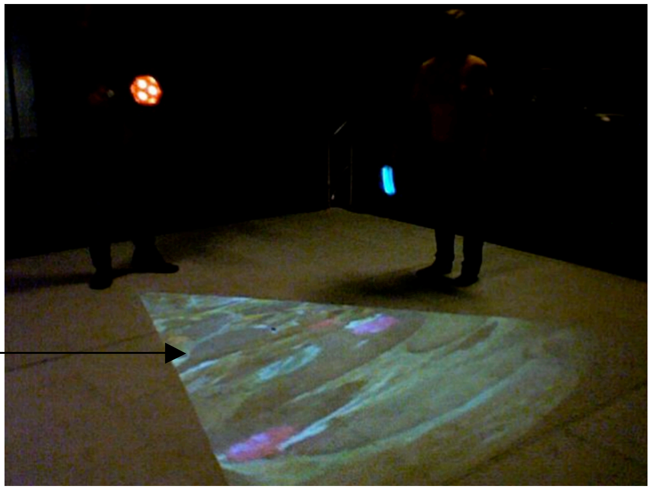


figure 4. The cone mapped with a calm image, pointing away from the sGlowStick (orange light, upper left of image).

Two primary choices were available: a left-right rotation (primarily executed with a twist of the wrist) or a forward–backward rotation (executed with a combination of wrist, elbow and shoulder movements, similar to casting a fishing line). The wide range of motion available from the forward-backward rot

ation provided better resolution and control for the instruments with which we worked. (For other cases the left-right rotation might seem like a natural choice because it would match the left to right arrangement of keys on a keyboard.)

Having decided to use the forward-backward rotation, we then had to decide whether forward (naturally a downward motion) or backward (naturally an upward motion) should correspond with a higher pitch. The language with which we describe a pitch—"higher" or "lower"—strongly suggests the latter. We found that, in fact, it felt very unnatural to map the pitch in opposition to this linguistic association. So we assigned a backward rotation (an upward motion) to create higher pitches and a forward rotation (a downward motion) to create lower pitches. We applied this approach to both the brass instrument controlled with the sGlowStick and to the shaker instrument played by the sGlowBop-it. Careful consideration was taken regarding the placement of the two-axis accelerometer in order to capture the most natural motions while holding the objects.

Texture Modeling

A classifier trained by natural movements with the smart objects determine whether an action is sharp (staccato) or slow and gradual (legato). This dynamic classification model determines the visual texture wrapped around the virtual cone object depending on the gestures performed with the sGlobjects.

When the model outputs the legato class, a photograph of clean water running over smooth river rocks wraps around the virtual sound cone. When the model outputs the staccato class, a computer generated rendering of blackbirds across a white backdrop is displayed instead. We chose these two images to promote a visual understanding of the long, flowing movements associated with legato and the short, granular movements associated with staccato (see figures 4 and 7).

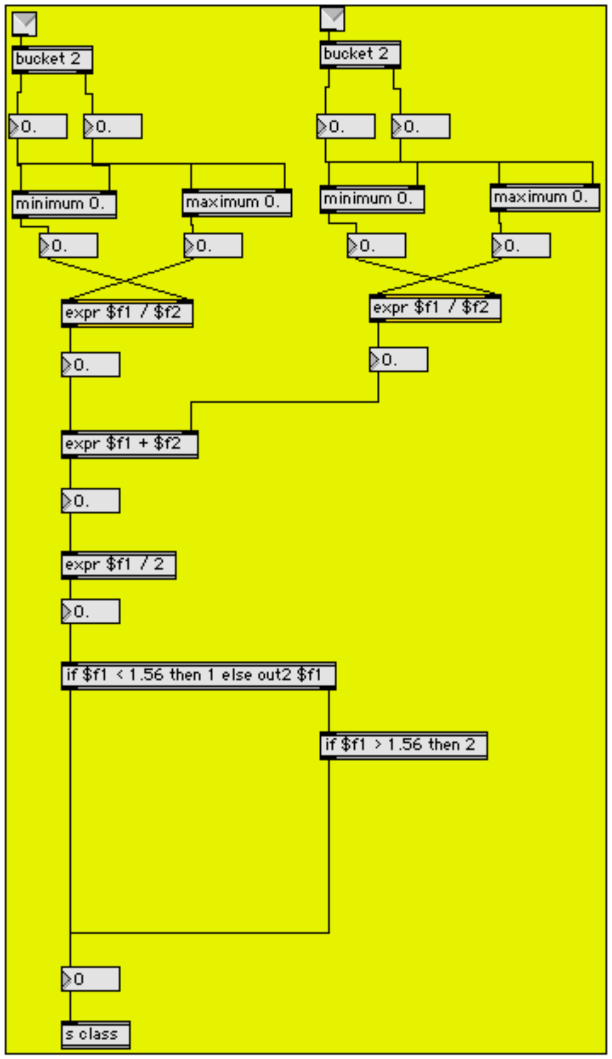


figure 5. The model intakes accelerometer data, creating an average from two consecutive instances.

The model intakes motion information from the two-axis accelerometers embedded in the sGlowStick and sGlowBop-it. The model captures two instances of the x-y planar position data, and then determines the minimum and maximum between the two pairs. From these values, the model produces an x ratio and y ratio by dividing the minimums by the maximums (see figure 5). Finally, the model adds the two ratios and then divides by two, sending the average of the two instances into a low pass filter.

The low pass filter adds nine instance averages and divides by nine to create an event average. We chose to filter the information in this manner due to the quickness at which we receive the initial motion information. The rate of change between the two classes was too rapid to provide users with an intuitive understanding between legato and staccato.

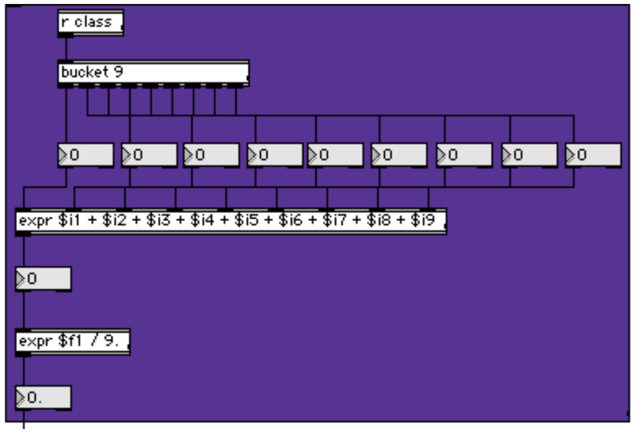
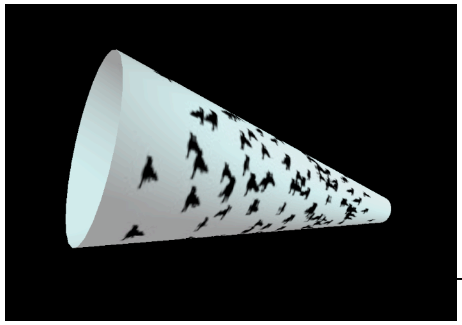


figure 6. A lowpass filter adds nine instance ratios and divides by nine to create an event average.

Using the filter to capture nine instances to compile a single event ratio for the classifier to evaluate presented users with comprehensible changes between the two classes. Once filtered, the model then sends the new event average to the event classifier (see figure 6).



A digital rendering of the flying blackbird image mapped to the sound cone.

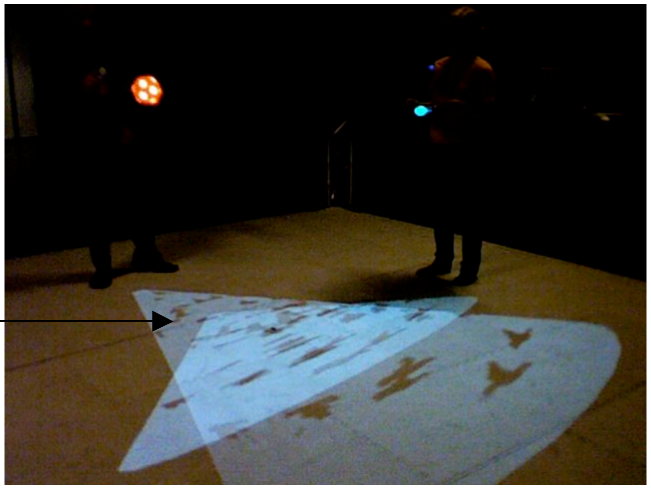


figure 7. The cone mapped with a fluttering blackbirds at high contrast, expressing the erratic motion of the sGlowStick, which is also rapidly changing the shape of the displayed cone.

The event classifier passes the event average through a comparative expression to determine the class. The comparative expression determines if the event average is less than, equal to or greater than 1.33. The majority of event averages we observed ranged from 1.05 to 1.62. Additionally, we observed anomalous averages as low as 0.73 and as high as 3.17. We used initial user interaction tests with the system to determine that 1.33 worked as a baseline value. If the comparator determines the event average is less than 1.33, then the model classifies the event average as legato, mapping the river water photo to the virtual object. If the comparative determines the event average is greater than or equal to 1.33, then the model classifies the event average as legato, mapping the blackbirds rendering onto the virtual object.

Observations

In this section we will describe our design decisions based on our practical observations and use of the system. We will discuss the visual, tangible, and classification considerations we made.

In our initial plan, we thought to visualize the shape of an object through which the generated sound would travel. We thought creating a shape whose properties change to express how sound would physically be amplified would engage the user and create a stronger relationship between the gesture performed and the sound created. We decided to simplify this concept, making instead a cartoon illustration. Similar to a cartoon megaphone squawking open and closed, our sound cone visualization resembles the flared end of a trumpet and a human mouth.

This is the first application to use the sGlowStick, although it had already been in development before this use. We decided to duplicate its hardware in the sGlowBop-it, which has much different affordances already available to us without creating a new form factor. Hacked toys promote these alternate uses because so much research into their physical usability has already taken place. This promotes the expansion of these tested designs for other uses, without duplicating the original usability research.

After their initial design and construction, we performed testing on our physical interfaces. We used both the sGlowStick and the sGlowBop-it in natural and intuitive ways, and communicated about how we thought those motions should sound. These tests resulted in some configuration changes both in the software and hardware, including adjusting which instrument parameters are controlled by which motions, and rotating the sGlowBop-it's two-axis accelerometer by 180 degrees from its original position.

We performed similar testing to train the gesture classifier. We each made what we considered to be staccato and legato movements then programmed those thresholds to react accordingly.

Conclusions

Real world instruments are extremely complex and are challenging to model. Interactions with these real instruments allow very nuanced, subtle performances. Our instruments, as they currently operate, come no where near providing this level of subtlety. They are, however, very flexible and allow for the generation of performances that are not possible with traditional instruments. Our instruments can dynamically transform based on programmed settings or actions of the user. These and other sensors can be embedded in clothing or in the architecture of a space. The possible modes of implementation are limited only by the creativity of the developers.

We found that our two-axis accelerometer limited the natural gestures we wanted to perform with our tangible interface objects. In the future we may consider the use of a three-axis accelerometer, perhaps with gyroscope sensing abilities as well. Battery life is, too, a concern for the sGlowStick and sGlowBop-it. More research into longer lasting battery configurations is necessary.

Further investigation would also benefit the visual feedback of the sGloTaT experience. The use of more subtle visual changes according to gesture and spacialized sound are two interesting future possibilities.

Overall, we think we have created an engaging and fun experience for gestural music creation by people of all skill levels.

Acknowledgements

The authors would like to thank the Computational Principles for Media Arts class in Arts, Media & Engineering at ASU for their generous support.

References

[1] AudioNerdz Delay Lama, a VST Max/MSP instrument

http://www.audionerdz.com/

[2] PeRColate synthesis and signal processing algorithms for Max/MSP

http://www.music.columbia.edu/PeRColate/

[3] Bop-it electronic game by Parker Brothers, Milton Bradley, Tiger Electronics, and Hasbro

http://en.wikipedia.org/wiki/Bop_It

[4] Bhatnagar, R. MIDI Ironing Board. Wired.com 2007.

http://www.wired.com/entertainment/music/news/2007/09/handmademusic

[5] SMALLab, an ASU AME research project.

http://ame2.asu.edu/projects/ameed/smallab/smallab.php