A Kinect™ and Wiimote™ Based Digital Drum Kit

Sound Capstone, Winter 2011

Jeffrey Booth
University of Washington
boothjm@uw.edu

Benjamin Ullom
University of Washington
ullom@uw.edu

Michael Sloan
University of Washington
mgsloan@gmail.com

Dan Gerdesmeier
University of Washington
danger@cs.washington.edu

ABSTRACT

Have you ever wanted to play the drums, but couldn't due to noise, space, or cost constraints? Our digital drum kit addresses all of these concerns, opening the door for people to play the drums in their home without disturbing neighbors or taking up space in a small apartment. By using rich input and feedback devices that are affordable and commercially available, we have created a system which is both musically useful and enjoyable as a real-time input device. Our solution provides a unique avenue for artists to create music, drummers to practice, and the general public to entertain themselves.

Keywords

Kinect, Wiimotes, Chronos, Drumkit, Virtual, 3D, Xylophone, Instrument, OSC, MIDI, Color Tracking, Accelerometer

1. INTRODUCTION

Drum kits, like many instruments, are too loud to play in most living environments. Drum sets are often large and cannot easily be transported. Currently there are two existing solutions to these problems: electronic drums and virtual drums.

Electronic drums, such as those shown in **Figure 1**, substitute drum pads for standard drums and produce sound electronically. Since electronic drums can be used with headphones, they do solve the noise issues, but they take up just as much space as a real drum kit and are generally quite expensive.

The term *virtual drum kit* typically refers to a completely software based implementation used to produce sounds electronically. Virtual drum kits do address the cost, space, and noise issues, but at the expense of the entire drumming experience. This software interface requires the drummer to either click a drum with a mouse or touch a drum on a touch-screen or trackpad in order to produce a sound. This experience is nothing close to playing a drum in real life, and is often times very limiting. An example virtual drum kit can be seen in **Figure 2**.

Figure 1: A Roland electric drum kit.

Our goal was to produce a product that fits between these two solutions. Our virtual drum kit would mimic a real-life druming experience as closely as possible, while minimizing the cost and space constraints. To do this, we sought to use consumer human interface devices that can capture a user's drumming motions and provide haptic feedback.

Figure 2: A screen capture from a standard virtual drumming application.

2. HARDWARE

For those unfamiliar with the hardware devices used in this project this section will provide a brief overview of each of the device's capabilities and how they were used in this project.

2.1 Kinect™

The Kinect™ utilizes a standard video camera, an infrared camera, and an infrared pattern to produce a color mapped image as well as a depth mapped image. We utilized Nicholas Burrus's "nestk" library, which his "RGBDemo" relies on in order to reliably process the data derived from the sensor into a 3D mesh of the scene. In addition to being able to map objects in three-space, the depth data that they Kinect provides aids in the computer vision techniques we apply allowing us to accurately track colors on the screen. The downside to the Kinect, however; is that the large amount of data received can take sigificant time to proccess. The solutions section of the paper will describe how we overcame this issue.

The Kinect also provides for the more "flashy" parts of the project. The background of the scene is captured to provide a visual reference frame to the user (rather than floating in space). Light "pulses" are emitted upon hitting the drums, and propogate across the surface of the kinect's impression of the user's form. This corresponds to the 3D nature of sound waves.

2.2 Wiimote™

The Wiimote™ is a Bluetooth based gaming controller that we found to be a perfect match for our application. Not only does the stick-like shape of the remote make it feel like you are playing the drums, but the Wiimotes also have four other features that make them ideal for this type of application: built-in accelerometers, Bluetooth data connection, 10 buttons, and force feedback. The accelerometers in the Wiimote were the biggest must-have feature as they gave us more fine grain detail as to how the person's hands are moving in real time. The easy to setup Bluetooth connection, extra buttons, and haptic feedback were supplementary factors that made the Wiimotes easier to setup, provided a more realistic drumming experience, and allowed us to change options within the program.

When the Wiimotes are relatively stationary, normalizing and negating the accelerometer data yields a very close approximation to the orientation vector. Much more fancy integrations of the Kinect's absolute position information and the accelerometer data could be performed to stabilize the orientation representation, but this does not seem to have greatly impeded playability.

2.3 Chronos™

The Chronos™ EZ430 is a development kit for a Texas Instruments microcontroller that is packaged in a sports watch. It has built-in accelerometers, a wireless data connection, and is relatively inexpensive, which made it a perfect choice for our third input. When the watch band is threaded through shoelaces, it is possible to detect the movement that correlates with pressing a bass drum pedal.

3. SOLUTION

Our solution to this problem can be broken into three distinct parts. They are location tracking, hit detection, and audio playback. First, We wanted to be able to track a person's location as it allows us to display a rich visual as well as allows for complex drum sets to be placed in three-space. Secondly, the hit detection feature was important as

Figure 3: The hardware devices used.

we needed to know when the person actually was trying to strike a drum and how hard they hit it. As drumming is a fast placed instrument we needed to make this detection fast. Lastly, we wanted our audio playback to layer ontop of existing sounds and blend in real-time so that the playback sounded natural.

3.1 Application Design

Our application is named DigiDrums, and is based on a Model-View-Controller architecture. We have a DrumSet class which serves as the model: it stores the position of the sticks and the drums, and stores the mapping from sticks to drums and drums to sounds. We have a DrumViewer class that renders the application graphics via OpenGL. We have a Controller class for each input device, and a class named KinectWiiController which merges the input from those classes. We have an OpenALSoundPlayer class that plays sounds via OpenAL and sends signals via OSC when drums are hit. Finally, we have a DrumController that coordinates all of the above. DigiDrums has been designed to be very modular, allowing for future expansion or modification in the event that not all of the hardware devices are present, or a different soundplayer is desired. We view this as a great strength of the project because it makes it possible for knowledgeable users or the open-source community to modify the project to take advantage of hardware devices that users already own, thereby reducing the cost of using the kit. For example, while we currently use the Kinect to perform 3D tracking of the Wiimotes, it would be possible to use a webcam and color tracking to provide position information for a simpler drum kit. In the future, an options screen could be added to choose and configure the available input devices and sound output options.

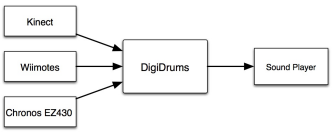

Figure 4: A diagram of the project data flow.

3.2 Location Tracking

The location tracking section of our project is responsible for taking the data generated from the Kinect and finding the (X, Y, Z) coordinates of our colorcoded Wiimotes. It works by marking regions which are simultaneously near and fall within a particular region of HSV space. This depth data helps a lot for disregarding regions of no interest, isolating the user from irrelevant background information. In order to achieve the speed that is required for responsiveness, and to

free up computational resources for other tasks, it is assumed the blob does not move too significantly. A small, 64 by 64 pixel, region is extracted and processed. If there is a likely object in this portion (which there almost always is), then the full scan can be avoided.

3.3 Hit Detection

Hit detection was done using the Wiimotes^{TM} to improve performance. The delay of the Kinect image was significant enough, 100ms-2s, that we wanted some other way to tell our program that a hit has occured. Due to their fast data collection and transmission rates, the Wiimotes were responsible for detecting a hit occuring while the slower Kinect was responsible for picking which drum the Wiimotes has come in contact with. To do this we looked entirely at the accelerometer data. We initially tried to apply machine learning to the accelerometers, but found that it was overkill for such a relatively simple task. We instead recorded accelerometer data while performing hits of varying amplitudees. By analyzing the data by hand we were able to come up with a heuristic using only the z-axis accelerometer that reliably detected intended hits without activating on other motions.

We remember six samples from each Wiimote and constantly shift the oldest sample out of our data array. If the four oldest samples are negative and then two newest samples are positive we know we have a hit. The negative samples represent the downswing of the Wiimotes and the positive ones represent the recoil after the hit. We found this to work pretty well for multiple people and easy for the body to figure out the necessary motion. We used a similar method for the Chronos accelerometer data, but because the poll rate was much lower we used a smaller data buffer size.

After we detected a hit we needed to also specify an amplitude of the hit. A crash symbol that you hit soft sounds much different than one you hit hard. We tried multiple techniques, but in the end we ended up using a bucketing technique. We grouped hits into soft, medium, and hard based on the average acceleration of the downswing. In order to appropriately catagorize the hits, we scaled each hit by the maximum hit seen thus far. This allowed the Wiimotes and Chronos to automatically calibrate themselves for a variety of drummers. After grouping the sounds we played a different pre-recorded audio clip corresponding to the intensity.

3.4 Audio Playback

The audio playback portion of our project is handled in DigiDrums by OpenAL, but could be easily substituted by any program that supports Open Sound Control (OSC); a light-weight network communications protocol designed for use in audio. OSC allows the audio playback to occur on any computer on the network and also gives the user flexibility in choosing sounds or even re-routing the output to another program. For example, it is possible to convert the OSC messages to MIDI (using existing software) and use our digital drum kit as an input to recording software. Our recommendation for an OSC receiver is ChucK, a strongly-timed, on-the-fly audio programming languge.

Audio playback was harder then we originally expected as sounds from the drum set could happen at any time and we wanted them all to layer over each other. Aside from the issue of blending audio on-the-fly we also wanted to make sure that the sum of the sounds did not cause clipping. Our first implementation in ChucK had clipping issues when two or three drums were played at once with high intensity. As each of the sounds did not know much about each other or when another one was going to come it made scaling a nightmare. We thus turned to OpenAL which is commonly used in many gaming platforms to provide audio. The platform allows you to have multiple sources. You specify a sound buffer to play, pick a source from a pool of them, and then have the buffer play. This model worked well as the OpenAL library knew how many sources it had available and how many are playing so scaling the audio worked much better than our initial implementation.

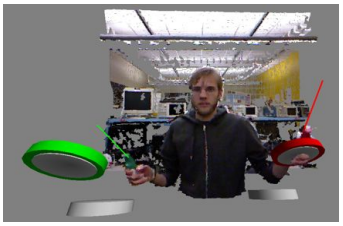

Figure 5: A screenshot of the application while playing.

4. CONCLUSIONS

Using off-the-shelf hardware and cross-platform audio playback software, we were able to develop a digital drum kit that is responsive, intuitive, versatile, compact, and quiet. While there is plenty of work that can be done to improve upon our design, we accomplished creating a playable instrument that can be useful for multiple user types ranging from the general public to aspiring musicians. While one of our initial goals was to be able to use this as a practice device for real drummers, we believe that people wanting to use our drum kit as a MIDI controller might find our project even more appealing. Our solution makes it an ideal alternative to a real drum kit in situations where space and noise are concerns.