Mapping Solutions for Kinect's User Tracking via OSC

Jon Bellona
University of Oregon
750 E. 25th Ave.
Eugene, OR 97405
1-315-404-2239
bellona@uoregon.edu

ABSTRACT

In this paper, I present a Max/MSP/Jitter interface for routing and displaying user-tracking data from the XBox Kinect via OSC messages. The paper addresses the OpenNI framework, the OSCeleton proxy software which formats and transmits user tracking data as OSC messages, and mapping strategies for forty-five points of continuous control data from a single user.

Keywords

Kinect, Max/MSP/Jitter, github, OpenNI, Middleware, NITE, jit.freenect.grab, Open Sound Control, OSCeleton, Terminal, Processing, Kyma.

1.INTRODUCTION

The release of the Kinect for the XBox in November 2010 set off a frenzy in the hacking world, with a cash prize for creating computer drivers that would connect to the Kinect camera.[1] Described as a "controller-free gaming and entertainment experience" for the Xbox 360 video game platform,[2] the Kinect device produces a depth map, an image where pixel values represent the distance from the camera (X,Y,Z values). In addition to the possibilities for working with 3D matrices, users can be simultaneously identified and tracked, enabling powerful gestural control tools for the electronic composer or digital artist.

In order to streamline the process for composers and artists alike wishing to access user-tracking data afforded by the Kinect device, I attempted to create an accessible Max/MSP/Jitter interface to the Kinect's depth map bundled with a user-tracking library. My aim was to compile a (somewhat) 'plug-and-play' interface that utilized open-source software and protocols, like OSC message forwarding controls, and offered functionality that overcame mapping challenges and potential performance pitfalls. This document serves to explain user tracking through an examination of the OpenNI framework, to discuss potential issues and challenges inherent in selected Kinect open-source libraries, and to offer a tangible solution (*Kinect-via-OSCeleton* Max interface) for anyone wishing to explore user tracking with the Kinect device for creative applications. Currently, my *Kinect-Via-OSCeleton* interface and installation instructions are available at http://deecerecords.com/kinect.html.

2.NATURAL INTERACTION (NI)

Natural Interaction (NI) refers to human's interaction with technology, especially in terms of experience coming from the senses.[3] From interactive to interface design, which look at enhancing user experience through behaviors and simple design, there has been an increased focus on our natural interaction with technology and its cultural impact, especially with the increased use of mobile devices, like multitouch cell phones and laptop computers. More recently, companies like PrimeSense are helping to create controller-free devices (the Kinect), that change user experience through the concept of natural interaction, specifically body motion tracking. The following sections discuss the overarching framework of natural interaction related to the Kinect, gradually narrowing from the larger framework to the specific open-source drivers that handle user tracking.

2.1OpenNI

OpenNI is a not-for-profit dedicated to promoting "the compatibility and interoperability of Natural Interaction (NI) devices, applications and middleware."[4] OpenNI is also a framework that provides an application programming interface (API) for writing applications utilizing natural interaction.[5] The API covers communication with both low-level devices (e.g. vision and audio sensors), as well as high-level 'middleware' solutions (e.g. for visual tracking using computer vision).[5] The ultimate goal of OpenNI is to develop a standard between NI devices, perhaps similar to what MIDI accomplished with a musical protocol between music devices. Thus far, however, OpenNI has only released open-source drivers for use with OpenNI devices, like the Kinect.

2.2Middleware

Middleware is computer software that connects software components and their applications together. Middleware sits "in the middle" between application software that may be working on different operating systems and "allows multiple processes running on one or more machines to interact."[6] Middleware can be referred to as integration software because the software allows data to be accessed between multiple databases and multiple machines. There are many OpenNI middleware modules to be used with OpenNI devices, including point tracking and user tracking.

2.3NITE

PrimeSense NITE is the specific middleware for OpenNI containing the algorithmic infrastructure for user identification and gestures recognition, as well as the control framework that manages the acquisition and release of control between users.[6] Implementing the NITE middleware module enables the use of powerful and complex programming functions and, at the same time, offers a more stable system for development testing.

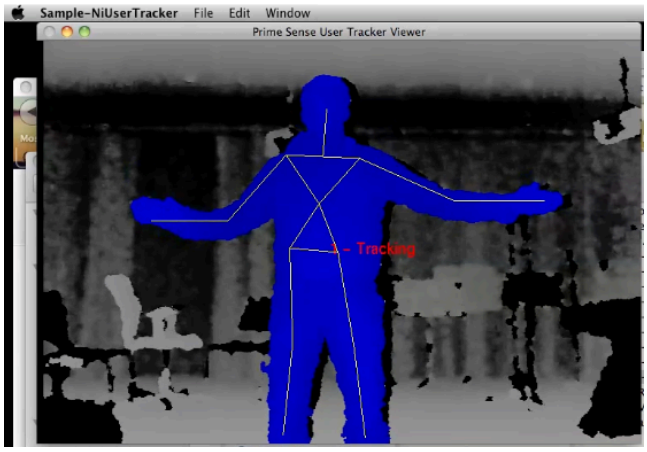

Figure 1. User-tracking skeleton with NITE Middleware in the OpenNI framework.

Not all user tracking utilizes a skeleton framework, but OpenNI's user tracking identifies several key reference points of a user's body, namely body joints that are tracked in real time. Each pixel point, or joint, transmits three coordinate values (X, Y, Z) at a rate of 24-30 frames per second, depending on the software written to run with OpenNI framework. The sample OpenNI user tracking is visually identified by a colored skeleton.

3.INFORMATION RETRIEVAL

3.1Open Sound Control (OSC)

Open Sound Control (OSC) is a stable, 32-bit protocol used for interconnecting hardware controller devices to the computer, as well as software on one or more computers.[7] Because OSC has taken off as a stable and fast protocol with user-defined packets of information sent to/from computers and devices on the same local network, I specifically sought after proxies that would translate and transmit user-tracking skeleton information as OSC messages. This guiding principle lead me to research and integrate the following open-source software.

3.2SensorKinect Library

Before the OpenNI framework can work with a Kinect device, a software driver must be written or installed to create a digital handshake between the computer and the Kinect device. The SensorKinect Library is an open-source software driver that registers the Kinect device with the OpenNI framework (which must also be downloaded and installed) and allows access to the Kinect's depth map.[8] Currently, the SensorKinect Library only grants one application access to the Kinect device at any one time.

3.3OSCeleton

After drivers connect the computer to the Kinect, additional software must be written or installed to access the PrimeSense middleware of the OpenNI framework. OSCeleton is an open source proxy software that takes user-tracking skeleton data created with the OpenNI framework and the Kinect depth map and then translates and transmits skeleton joints' X,Y,Z coordinates as OSC messages. With OSCeleton transmitting joint values as OSC messages, I could harness point specific data through a variety of software, namely Max/MSP/Jitter, Symbolic Sound's Kyma system, and Processing. While OSCeleton was created to work with Windows, Linux, and Mac OSX, my research, including two installs and interface work, was all completed on computers running the Macintosh OSX platform.

3.4Terminal

On the Macintosh platform, OSCeleton software requires Terminal to run.[9] Terminal is a line interface to control underpinnings of the UNIX based OS.[1][10] After executing the OSCeleton program, Terminal sits in the background with minimal load on the CPU.

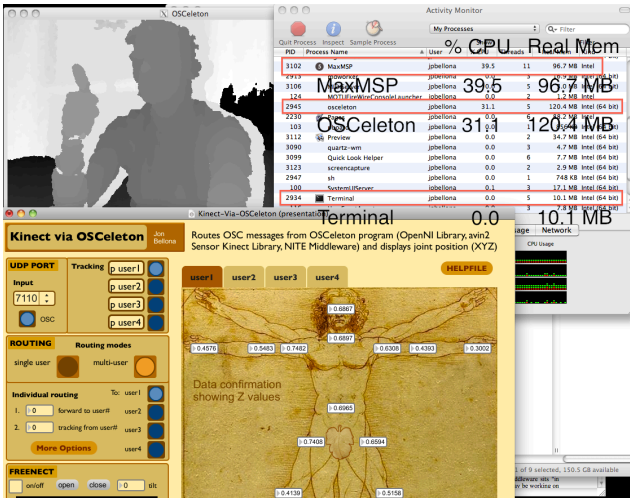

Figure 2. Activity Monitor showing Terminal, OSCeleton, and Max/MSP/Jitter tracking one user. Terminal only uses 10.1 MB of real memory, compared to 96.7 MB for Max/MSP.

3.5Max/MSP/Jitter

Max/MSP/Jitter is a graphic programming environment for music, audio, and multimedia. Due to the programming flexibility, ability to communicate between various software, and dedicated Max community, I chose Max/MSP/Jitter as the environment to build a working interface with the Kinect.

3.5.1Jit.freenect.grab

Matrix data coming directly from the Kinect may be accessed inside of Max/MSP/Jitter with the external jit.freenect.grab by Jean-Marc Pelletier.[11] His jitter external excellently integrates Kinect drivers to Max/MSP/Jitter, requiring no additional software drivers, and jit.freenect.grab can control the Kinect's tilt motor.

---

[1] All Macs have Terminal located in their Utilities folder inside the Applications folder. More about Terminal and Terminal commands may be found online at: http://www.scribd.com/doc/2084227/Mac-OS-X-Terminal-Commands-list

Many open source drivers do not offer servo motor control, making this 'plug-and-play' Max external very useful. Jit.freenect.grab also retrieves RGB images, depth maps, and accelerometer information via matrices; yet, there is no user tracking capability offered. Due to the inability to track points and users, my research shifted to working with the OpenNI framework, ultimately settling upon the OSCeleton proxy.

3.5.2OSC-route

OSC-route is a Max external written by researchers at CNMAT (Center for New Music and Audio Technologies).[12] The external allows OSC messages to dispatch inside of Max/MSP/Jitter. Along with Max 'udpreceive' objects, all OSC messages may be identified and subsequently parsed.

4.POTENTIAL ISSUES & CHALLENGES

4.1OSCeleton Execute

Because the OSCeleton software is only executable once via Terminal, alterations to the UDP port or IP address cannot be changed during runtime. Any change in address means a program kill, as well as a loss in any skeleton data. An important factor in developing my *Kinect-via-OSCeleton* interface was to ensure that UDP port changes could be altered 'on-the-fly' without severing the connection between the Kinect and OSCeleton. [Section 5.2]

4.2Single UDP port

OSCeleton only offers a single UDP port to transmit user tracking data. Because UDP ports cannot be shared, using a single port that is unalterable during OSCeleton's runtime is potentially problematic, especially if a user wants to treat a single data stream differently. For instance, sharing one skeleton for musical mapping with Max/MSP/Jitter and visual mapping with Processing is not possible under the current OSCeleton build. An external interface is required.

4.3Non-Visual Skeleton Confirmation

Since OSCeleton displays no visual confirmation of skeleton data in its viewer window, a user must view the Terminal window in order to know when a user has been identified, has begun calibrating, has successfully completed calibration, and/or when a user has been lost. The lack of color identifying users and a redundant skeleton display make the program less user friendly, especially for users needing quick visual confirmation during a live performance setting.

4.4Non-Hierarchical OSC Message Handling

OSCeleton does not pack OSC messages into a hierarchical framework. The format of incoming OSC messages can be broken down as such: a generic "joint" identifier, the specific joint name, the user number, and the respective X, Y, Z values.

/joint/l_hand/1 0.5 0.5 3.5	User 1's left hand is approx. middle of the Kinect camera and about 10 feet away.

X values have an initial range 0.0 to 1.0
Y values have an initial range 0.0 to 1.0
Z values have an initial range 0.0 to 7.0

Figure 3. Format of OSCeleton OSC message.

User identification number occurs after the joint identifier, which potentially creates issues for routing. For example, if user1 is being tracked within a performance and user1 is dropped, the user may be re-identified as user2 or user3. Since user# comes after joint identification, effective re-routing of skeleton information must be handled after OSC parsing, which poses challenges for mapping.

4.5User Tracking Handling

The NITE Middleware handles all user-tracking identification. The tracking is fast and efficient; however, if any user is lost during a performance, a re-calibration must occur. Even though re-calibration can be handled quickly 'on-the-fly,' the user tracking number is handled by the middleware, preventing a hard coding of users to their respective user number. Any mapping that requires a specific user number must be handled after the OpenNI framework and after parsing OSC joint messages.

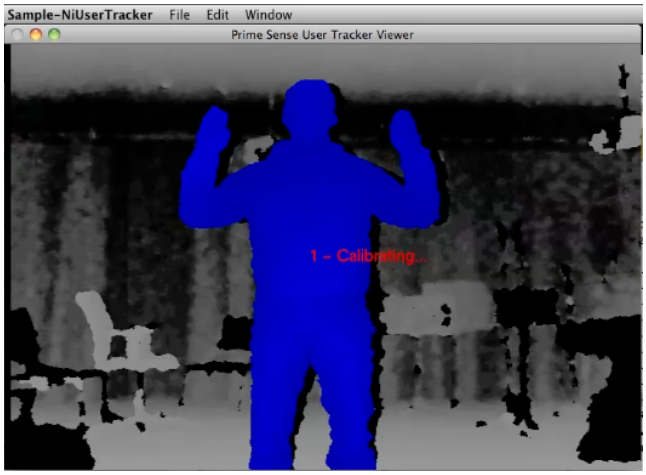

Figure 4. Psi pose used for user tracking calibration.

While I raise several potential issues with OSCeleton, there are many benefits to the software, including the ability to execute a reverse screen for tracking, the ability to simultaneously handle fifteen joint messages for four users which means up to one-hundred eighty continuous control values at any one time, and the ability to provide a fast (30 fps) and stable environment for transmitting user-tracking skeleton data as OSC messages over a UDP/IP port.

5.KINECT-VIA-OSCELETON INTERFACE

The *Kinect-Via-OSCeleton* Max interface is an input/output routing interface for use with the Kinect hardware and OSCeleton software. The interface addresses issues of the OSCeleton software by offering a clean display and robust control panel for disseminating skeleton information and OSC joint messages for up to four users. The open-source interface and installation guide are available for download at http://deecerecords.com/kinect.html

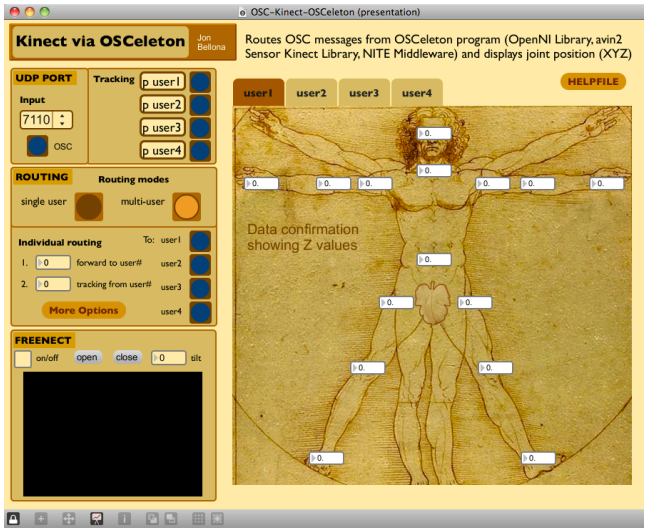

Figure 5. *Kinect-Via-OSCeleton* Max interface.

5.1 Input Routing

The *Kinect-Via-OSCeleton* interface addresses OSCeleton's transmission of non-hierarchical OSC joint messages with an associative array. *Kinect-Via-OSCeleton*'s associative array is a Max 'coll' object that uses strings as variable key indices, which may be parsed directly from incoming OSC messages. The variable key values alter 'forward' objects inside of Max/MSP, so that routing may be changed 'on-the-fly' as well as after OSC messages have been parsed. Joint names serve as key indices inside the associative array. All routing within *Kinect-Via-OSCeleton* may be changed while OSCeleton is running and after a user has been identified and calibrated.

5.1.1 Single-User Mode

Single-User Mode is defined as a control mode that, regardless of user#, forwards all joint messages on to user1. The Single-User Mode enables best solo performance practice, in that, regardless of OSCeleton and the NITE middleware, all OSC messages existing inside of *Kinect-Via-OSCeleton* will be handled by user1 'send' and 'receive' objects. Since routing is no longer dependent upon an external identification number, the user gains routing flexibility over parsed OSC joint messages.

5.1.2 Multi-User Mode

Multi-User Mode is defined as a control mode that forwards all user# joint messages on to their respective user#. Since *Kinect-Via-OSCeleton* offers flexibility in routing non-hierarchical OSC messages by changing user# associations, Multi-User Mode resets the routing functions to parse OSC messages as was originally designed. As such, Multi-User Mode is the default mode.

5.1.3 Individual Routing

Individual Routing is defined as a forwarding function that allows any user# joint messages to transmit onto another user#, including itself. If mapping calls for specific user# associations, the Individual Routing function enables specific user# routing. The function may best serve multi-user performance practice, where a user may be lost and needs to be reinserted as a specific user# after re-calibration, or where a mix up between multiple users can be individually re-routed without having to terminate the main OSCeleton software.

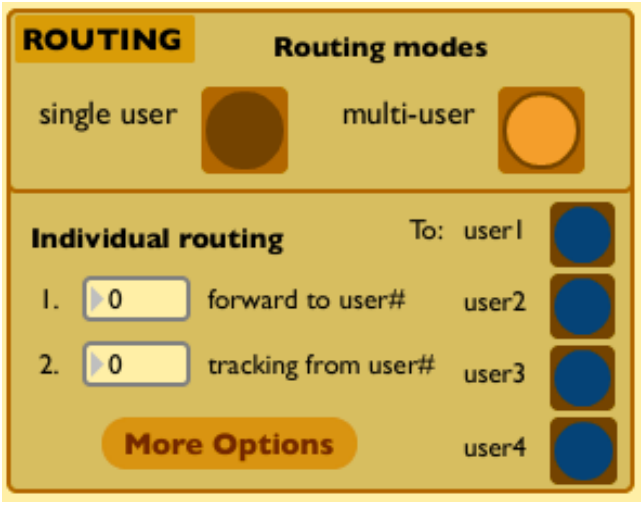

Figure 6. *Kinect-Via-OSCeleton* routing controls.

5.2 Output Routing

The *Kinect-Via-OSCeleton* interface also offers output routing functions that address issues discussed in Sections 4.1 and 4.2. Since OSCeleton only offers a single UDP output port that is unalterable after execute, I decided to apply a multi-machine forwarding approach, similar to that of a star network. As soon as the *Kinect-Via-OSCeleton* receives OSC messages over its UDP port, the interface can immediately forward up to four 'clean' copies of the data over user-defined UDP ports and IP addresses. Other applications and other computers may parse and map these forwarded copies of OSC skeleton data. Any additional computer that is running the *Kinect-Via-OSCeleton* interface to handle a forwarded OSC copy may, in turn, forward up to another four copies.

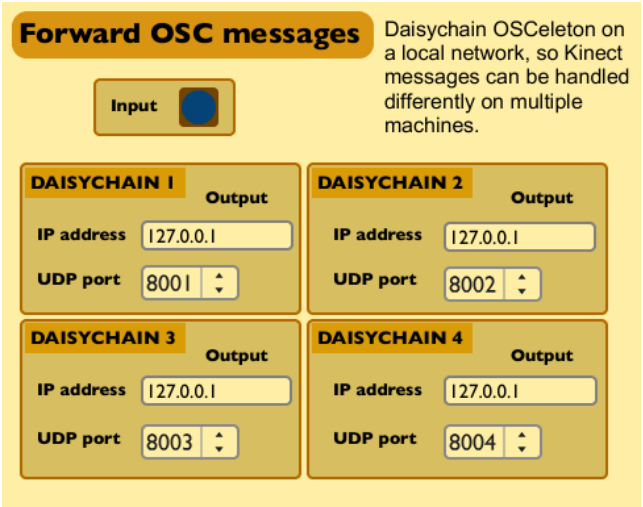

Figure 6. *Kinect-Via-OSCeleton* forwarding controls.

Forwarding skeleton object information should be handled at the root level, not after OSC parsing. Re-routing copies of skeleton data across OSC at the root level provides cleaner handling of the data, streamlining unnecessary strands of singletons. In addition,

since Max ‘send’ and ‘receive’ objects are recognized locally, parsed OSC joint messages used within the *Kinect-Via-OSCeleton* interface will not interfere with other copies on other machines. Therefore, the interface may be easily integrated into computer ensembles. Multiple computers can map a single skeleton data object differently, all running from a single Kinect device.

6.CONCLUSIONS

The OSCeleton software proxy is but one of many open source software applications created to work with the Kinect and the OpenNI framework. Because my aim was to establish a working solution today for users interested in working with user tracking with the Kinect depth map, my decision to use OSCeleton centered around stable user-tracking functions and OSC messaging. My hope is that this paper and downloadable interface may serve new works that utilize the Kinect for real-time electronic performance. The *Kinect-Via-OSCeleton* interface and installation instructions may all be found online for free at http://deecerecords.com/kinect.html.

7.ACKNOWLEDGMENTS

My many thanks to John Park for guiding me throughout the research process and Jeffrey Stolet for his input and support.

8.REFERENCES

[1] Adafruit. 2010. We Have a Winner: Open Kinect driver(s) released. *Adafruit Industries Blog* (Nov. 11, 2010). DOI= http://www.adafruit.com/blog/2010/11/10/we-have-a-winner-open-kinect-drivers-released-winner-will-use-3k-for-more-hacking-plus-an-additional-2k-goes-to-the-eff/.

[2] Wikipedia. DOI = http://en.wikipedia.org/wiki/Kinect.

[3] Valli, A. “The Design of Natural Interaction.” (October 28, 2006). DOI = http://www.naturalinteraction.org/

[4] PrimeSense. DOI = http://www.joystiq.com/2010/12/17/primesenses-tamir-berliner-on-the-future-of-natural-interaction/

[5] Wikipedia. DOI = http://en.wikipedia.org/wiki/Middleware

[6] PrimeSense, NITE Middleware. DOI = http://www.primesense.com/?p=515

[7] Open Sound Control. DOI = http://opensoundcontrol.org/spec-1_0 (accessed February 03, 2011).

[8] SensorKinect. DOI = https://github.com/avin2/SensorKinect

[9] OSCeleton. DOI = https://github.com/Sensebloom/OSCeleton

[10] Terminal. Apple Computer. DOI = http://www.apple.com/macosx/apps/all.html#terminal

[11] Pelletier, J.M. Jit.freenect.grab. DOI = http://jmpelletier.com/freenect/

[12] Wright, M. and Zbyszynski, M. OSC-route. DOI = http://cnmat.berkeley.edu/downloads