DILib: Control Data Parsing for Digital Musical Instrument Design

William Brent
American University
4400 Massachusetts Ave NW
Washington DC, USA
w@williambrent.com

Abstract

The Digital Instrument Library (DILib) for Pure Data is a set of abstractions and externals that were developed for a course on digital musical instrument design. DILib is intended to streamline the process of realizing musical instruments that make use of built-in laptop hardware, accelerometers, infrared fingertip tracking, full body tracking, multitouch surfaces, and other control data streams. In addition to providing convenience, the library's components are designed to establish a level of standardization with respect to the varied methods for obtaining sensor data from widely available hardware.

Keywords

Instruments, controllers, sensors, interfaces

1 Motivation

The number of viable options for physical control over digital synthesis processes has grown tremendously in recent years. Alongside custom-built hardware controllers, several types of commercially available technologies are being used for this purpose as well. These include multitouch surfaces like the iPad, and an array of hardware originally developed for use with video games, such as Nintendo's Wii remote, the Sony PS3eye camera, and Microsoft's Kinect sensor. In addition to being relatively inexpensive, this technology has the advantage of providing sophisticated sensor data in a standardized format. For a geographically dispersed community of digital artists, standardization and accessibility are often critical.

To complement this widely available hardware, there is a need for a Pure Data library that parses the associated data streams to further improve accessibility and ease of use. Such tools are very important for remote collaborations in general, but they are particularly needed for digital musical instrument design—a field in which the creator of an instrument is often its sole performer. Streamlining the process of obtaining sensor data will facilitate the further development and widespread use of instruments shared by the Pd community.

1.1 Introduction

This paper describes the Digital Instrument Library (DILib) for Pure Data, a collection of abstractions and external objects that were developed for a course on digital instrument design in the Audio Technology program at American University. DILib is available under the GNU GPL, and is intended to aid in realizing instruments that make use of built-in laptop hardware, accelerometers, infrared fingertip tracking, full body tracking, multitouch surfaces, and other sources of control data. Each library component parses and routes its data to specified send symbols to be easily received and applied by users. In some cases, raw sensor data is also interpreted and normalized before being transmitted.

DILib will be maintained on a long term basis with the intent of providing designers of new digital musical instruments a stable means of accessing data from an ever-increasing number of control sources.

2 General Design

DILib currently contains abstractions to handle data reported by laptop sensors, Wii remotes, TouchOSC, infrared blob tracking, reacTIVision, and OSCeleton (for the Kinect sensor). The components of DILib do not obtain sensor data themselves. Rather, the purpose of the library is to parse and manage data as it arrives from a source. For example, the computer vision software known as reacTIVision [2]—used for the reacTable interface—streams a great deal of data to Pd. Management of the coordinates, rotation, and speed of fiducial markers as reported by reacTIVision (as well as the distances between these markers and other relational data) is a considerable task in itself. DILib provides tools that dramatically simplify such issues.

DILib is intended to be used cross-platform, and many of its components will be useful on Linux, Macintosh, and Windows. For most data sources, DILib's abstractions provide parsed data via carefully chosen symbolic send names, which are listed in separate text files for reference. Some library components, like that for the Wii remote, provide multiple abstractions specific to the platform or additional software being used. In such cases, the goal is to provide any particular piece of data via the same send symbol regardless of the software or platform in use. Thus, digital instrument patches can be designed more generally and separate from these types of details. Users can employ the abstraction appropriate for their system, and the correct data will be supplied to the receives of the general instrument patch for use in its particular mapping scheme.

3 DILib Components

DILib is divided into directories relative to the control data sources mentioned above. The following sections provide details on each component.

3.1 Laptop

For many computer musicians, the design of instruments using built-in laptop hardware as a primary control interface remains an interesting area of investigation. The growth of laptop orchestras has further encouraged this approach. Regardless of hardware or platform, the on/off states of most keyboard keys are available via standard Pd objects, while the apple library included with Pd-extended offers access to data from various sensors featured on recent Macintosh laptops—most notably, a multitouch trackpad.

DILib's laptop interface abstraction is a convenient wrapper for accessing data from the key, keyname, and apple library objects. To receive the state of a laptop keyboard key (or any other piece of data), all that is needed is a receive object supplied with the appropriate symbol as a creation argument. Send symbols are provided for raw key and sensor data, as well as higher level data related to the multitouch trackpad, from which a variety of dependent and independent continuous data streams can be drawn. In addition to reporting the two-dimensional coordinates and size of each tracked fingertip, the abstraction provides the distances and angles between points, and the coordinates of their centroid. This information can be used to follow various finger gestures—such as pinching and rotation.

3.1.1 Continuity

When mapping gestural data to synthesis parameters, it is often desirable to refer to specific tracked points by stable identification indices. Raw trackpad data given by the apple library's multitouch object does not attach this type of persistent identifier to its output. Point indices are assigned based on the order in which they appear. This causes problems for maintaining the continuity of a parameter as its associated point disappears and reappears on the tracking surface. Such interruptions of data flow are common in sensor tracking situations. Thus, an additional object is needed to maintain a history of recent point locations. Based on this history, stable indices for tracked points can be assigned.

DILib's continuity external accomplishes this by comparing the coordinates of the most recent set of points with those of the previous set. Once assigned, indices can be used to follow the coordinates of a virtual point that exists independently of the fingertip by which it was most recently manipulated. The benefit of stable indices is demonstrated in the help patch by means of a multi-point two-dimensional slider control. Regardless of the number of fingertips present on the trackpad or interruptions in the flow of data, the virtual markers in the patch maintain stable values. The continuity object is appropriate for other applications as well, such as three-dimensional infrared blob tracking, described below in Section 3.4.

3.2 TouchOSC

TouchOSC[1] is an application that allows users to wirelessly stream data from a number of multitouch devices to a computer under the OSC protocol. Data is generated in the application when users manipulate various types of software controls, such as sliders, knobs, toggles, and buttons. Several fixed multi-layered controller layouts are provided, as well as an editor for creating custom layouts.

DILib's TouchOSC component is comprised of separate abstractions for each of the fixed layouts that route data to given send symbols. Apart from

[1]http://hexler.net/software/touchosc

providing easy access to a swath of multitouch control surfaces, one of the most interesting potential applications of these abstractions is the design of instruments featuring distributed control. By invoking several instances with unique receiving ports, multi-participant control of a global synthesis environment can easily be arranged.

3.3 Wii Remote

The Wii remote has been widely used as a source of gestural control data. In addition to several buttons and an accelerometer, it also houses an infrared (IR) camera with on-board blob tracking. For Linux users, an external Pd object with a thorough set of features is available for this device. An external with more limited capability exists for the Windows platform. At present, the most stable option for Macintosh users is to stream data from a separate application via OSC messages. Abstractions from DILib's Wii remote component are designed to deliver data captured using these various methods in a more standard format.

Separate abstractions are provided for parsing data received from additional software or objects used directly in Pd. In each case, the send symbols used to transmit data are consistent. For instance, whether using the wiimote external under Linux, or receiving data from the OSCulator[2] application on a Macintosh, the Wii remote abstractions report the device's vertical orientation via the receive symbol "wii-1-pitch". Because the methods listed above do not always provide data in standard parameter ranges, the abstractions also perform basic normalization on the data streams before they are transmitted.

The Wii remote's IR blob tracking component, which tracks up to four points at once, executes its own algorithm for maintaining continuity of tracked points. Thus, the use of DILib's continuity external is not required.

3.4 IR Blob Tracking

The most flexible and effective means of tracking IR blobs in Pd is to use a high frame-rate camera fitted with an IR band pass filter. Drivers for Sony's PS3eye USB camera are available on all platforms, and offer rates above 100 frames per second. Although the device includes a built-in filter for rolling off light in the IR spectrum, it can be removed and replaced with one that attenuates light at all but one specific IR wavelength. By constructing an array of IR LEDs for shining light on the area to be tracked, IR light can be directed back toward the array (and camera) with any highly reflective material. Figure 1 shows the camera's view of such a scenario and basic animation of the tracked points, illustrating how restricting the tracking scene to IR light eliminates difficulties common to blob tracking in general (e.g., control of background information).

[2]http://www.osculator.net

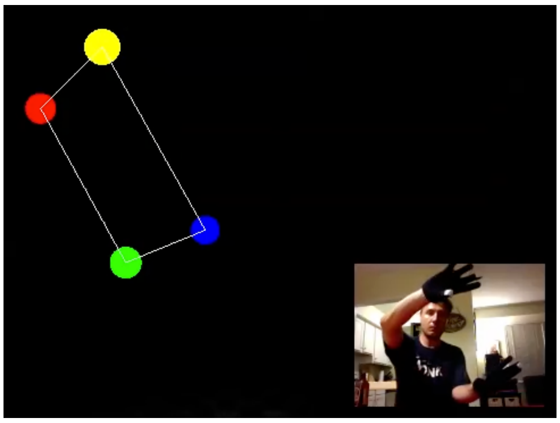

Figure 1: Tracking two fingertips on each hand via infrared reflection.

DILib's IR blob tracking abstraction uses basic GEM objects to manipulate image contrast, and pix_multiblob to capture blob coordinates and sizes. Blob coordinate information is processed by the continuity external, making it possible to reliably map a specific point to any particular synthesis parameter. The help patch assumes that four fingertips are being tracked (as in Figure 1) and uses the data to manipulate the shape of a polygon rendered in GEM. Unlike the IR blob tracking available using the Wii remote, tracking of any number of blobs can be attempted, and more control is available for setting tracking parameters like minimum blob size and contrast. Assuming that uniformly-sized reflective markers are used, depth information can be tracked much more reliably than with the Wii remote. Further, the information can be obtained at a frame rate higher than the Wii remote's 100 frames per second, and is available without the use of additional software.

3.5 reacTIVision & TuioHub

Jordà et al.'s reacTable project [1] is among the most well known and frequently constructed digital musical instruments currently in use. The associated reacTIVision computer vision software (for tracking markers on the table surface) and accompanying client packages for receiving TUIO data in various computer music programming environments are open source and freely available. The external supplied for Pd is called TuioClient. In order to apply the data it reports in a fully-functional reacTable, however, an additional system must be designed in Pd for storing, analyzing, and retrieving marker coordinates. This can be accomplished using table reading and writing objects, though some tasks—such as sorting markers according to distance—are relatively cumbersome in a patching environment, and better suited as methods of an external object written in C.

DILib's TuioHub external is a storage, analysis, and retrieval object for fiducial marker information reported by the TuioClient external. Similar to the standard Pd object called value, TuioHub is designed so that any number of active instances have access to the same memory space via a common pointer. A primary "collector" hub instance should be created to take input directly from TuioClient, and multiple "retriever" hub instances can be created as needed throughout the patch. Figure 2 provides a basic illustration. The collector is distinguished by supplying creation arguments for: number of fiducials to track, proximity threshold in pixels, and dimensions of the tracking space. Retriever instances are given no creation arguments, and respond to various "get" messages. For instance, to check whether or not fiducial 7 is active, the message "getFid_active 7" should be sent to a retriever TuioHub instance.

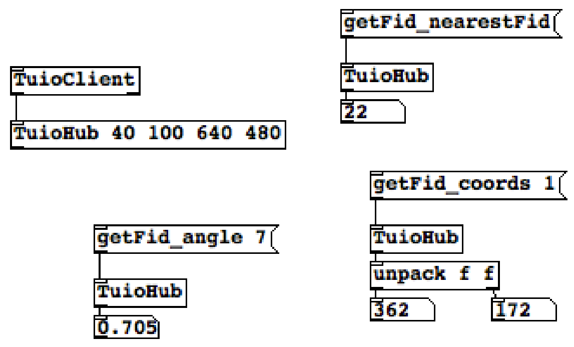

Figure 2: A collector and three retriever instances of TuioHub.

TuioHub's analysis methods can report fiducials in or out of range of any given fiducial, with an option to sort according to distance. Based on this information, connections between audio sources and processing modules can be activated or broken. Cursor data (i.e., tracked fingertip coordinates) can also be sorted relative to fiducials. By finding the nearest fiducial to a fingertip, it is possible use TuioHub to calculate the angle of the fingertip relative to the fiducial, which is useful for modifying various parameters of the associated audio module.

In order to demonstrate TuioHub usage, DILib includes a reacTable patch with 15 audio source modules, 10 processing effects, and 10 LFOs (35 fiducials in total). Basic fiducial marker animation (shown in Figure 3) is rendered in GEM for projection onto the tracking surface. Using the included abstractions for managing connections between audio modules, the patch can be adapted to handle any number of fiducials.

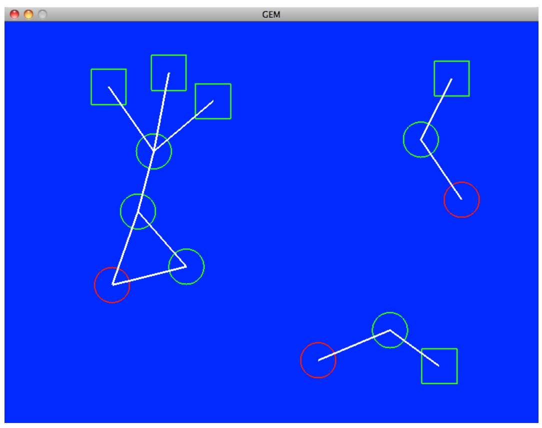

Figure 3: Basic GEM representation of fiducial markers and connections.

3.6 OSCeleton

OSCeleton[3] is open source multi-platform software that interprets data from Microsoft's Kinect sensor and produces three-dimensional coordinates for the primary points of a body being tracked. Its output can be received in Pd via OSC messages. DILib's OSCeleton component includes three abstractions for parsing and routing this skeleton data. The global parsing abstraction receives all data from OSCeleton, and routes the information to send names for each body point, regardless of user number. It also keeps track of which users are currently active. The user routing abstraction receives global data, filters according to a given user number, and routes information to user-specific send names. For instance, the horizontal position of the right hand of user 2 can be received using the symbol "oscel-2-r_hand-x". Finally, a drawing abstraction renders a user's basic skeleton frame

[3]https://github.com/Sensebloom/OSCeleton

to provide visual feedback, as shown in Figure 4. The latter abstractions must be supplied with a user number as a creation argument, but user focus can be changed via the first inlet.

Like other DILib components, the OSCeleton abstractions generate relative data in addition to raw coordinates. Distances between the left and right hands and feet are reported, as well as distances from the abdomen to each extremity. For continuous synthesis parameters, such as the modulation index of simple frequency modulation, these relative measurements are an effective mapping choice.

Another variety of relative data is the offset of an extremity from its attaching joint—such as the three-dimensional position of the right hand in relation to the right shoulder. The raw coordinate of a user's right hand can be polled to control a global parameter, while its relative offset from the right shoulder maintains a high degree of independence and is suitable for mapping to another parameter. Thus, a single element may be mapped to two related parameters. For example, the relative offset can be used to control pitch, while the global position can be made to affect timbre. As with acoustic instruments, such systems present an interesting set of constraints, where individual parameters can be modified with near—but not complete—independence.

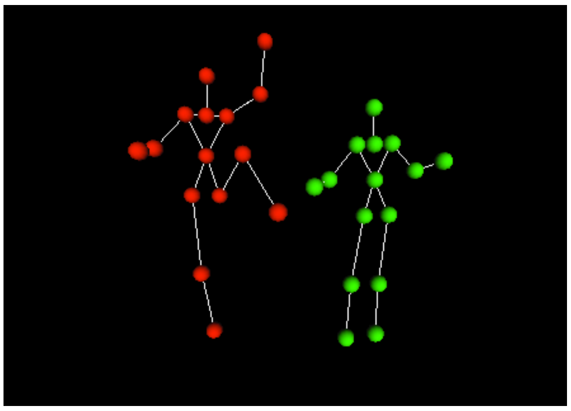
Figure 4: Skeleton frame rendering of two simultaneous users.

There is currently no external object that makes skeleton data based on the Kinect sensor available directly in Pd. When such an object is created, an alternate version of the global parsing abstraction will be supplied, so that any patches making use of this DILib component in its present state will still function properly after a simple replacement.

4 Conclusion

DILib is in its initial stages of development and will be actively expanded as additional mass-produced hardware useful for instrument design becomes available. Whenever possible, future development will maintain the delivery of data in a standard, platform neutral manner as a central aim. Even at this stage, the components described above are suitable for creative and educational use, and will facilitate the creation and re-creation of novel instruments, allowing a focus on higher levels of instrument design rather than the details of data parsing and analysis.

Other than the reacTable example, DILib does not include in-depth example patches that apply its components in actual digital instruments. A further development goal is to compile and release an examples package containing instrument projects created by the author and willing contributors from the Pd community.

References

[1] S. Jordà and G. Geiger and M. Alonso and M. Kaltenbrunner: "The reacTable: Exploring the Synergy Between Live Music Performance and Tabletop Tangible Interfaces." *Proceedings of the 1st International Conference on Tangible and Embedded Interaction,* pp. 139-146, 2007.

[2] M. Kaltenbrunner and R. Bencina: "reacTIVision: a Computer-vision Framework for Table-based Tangible Interaction." *Proceedings of the 1st International Conference on Tangible and Embedded Interaction,* pp. 69-74, 2007.