音 楽 情 報 科 学 23－1
（1 9 9 7. 1 2. 1 3）

Pure Data: Recent Progress

Miller S. Puckette
Department of Music, UCSD
msp@ucsd.edu; http://man104nfs.ucsd.edu/ mpuckett/

Abstract:

The Pd program has been used in two live performances to date, both of which have used real-time analysis of live musical instruments to control graphical synthesis. To support these performances Pd has been equipped with new audio analysis modules, one targeted for picthed instruments and one for unpitched. Network support has also been added to Pd to allow several machines to communicate in real time, either locally or remotely. The graphical synthesis package, named GEM, is being developed by Mark Danks.

Pure Data：開発の近況

ミラー S.パケット

カリフォルニア大学サンディエゴ校
msp@ucsd.edu; http://man104nfs.ucsd.edu/ mpuckett/

「Pure data」プログラムは、今日までに2つのライブ作品で用いられた。その両者ともステージ上の楽器演奏を実時間分析し、画像合成をコントロールする作品であった。これらの演奏をサポートするために、「Pure data」は、新しい音響分析モジュールを装備している。一つは音高を持った楽器を対象にしたものであり、もう一つは音高を持たない楽器のためである。また、他のいくつかのコンピュータがローカルでもリモートででもリアルタイムにコミュニケートできるネットワーク・サポート機能も加えられている。画像合成パッケージは「GEM」と名付けられ、Mark Danksによって開発が進められている。

1 Introduction

The Pd program [2]; [3] has been under development for about a year and a half. At the same time, Mark Danks has been developing Gem [1], a real-time graphical synthesis/processing/rendering environment which runs in concert with Pd. The goals of the Pd and Gem development efforts can be summarized as:

- a real-time patchable environment ala Max;
- management of audio and image processing in the same environment;
- adaptability to a wide range of platforms;
- long-term stability;
- a newer and more flexible set of tools to manipulate data.

Pd follows the Max paradigm for object interconnection and message-passing between objects. Along with Opcode Max-style "control" messages, Pd supports ISPW-style signal connections, so that a part of a Pd patch can be a Moog-style patch. Similarly, using Danks's Gem package, a Pd patch can include a chain of Open GL directives linked in much the same way as the audio, but giving rise to a graphical rendering procedure.

Some progress has been made toward the goals shown above. Pd's development has recently been driven by the production of *Lemma 1* (Thessaloniki, Sep. 27, 1997) by Vibeke Sorensen, Rand Steiger, and Miller Puckette, and of *pushit* (San Diego, Nov. 8) by Mark Danks. Both of these performances integrate live instrumental players with live computer graphics controlled by the sound of the musical instruments.

The setup of *Lemma 1* is shown in Fig. 1. A live trombonist and trap drum set player (George Lewis and Steven Schick) are on stage at either side of a large projection screen. An audio computer does real-time analyses of the two instrumental signals. This analysis data is sent to a graphics computer which uses Gem to render live graphics which can be animated in a variety of ways by the analyzed instrumental sound.

Sorensen, Steiger, and Puckette plan to develop *Lemma 1* further into a series of improvisations over the next few years, incorporating the additional idea of networking two or more sites into a combined event. In this configuration we would send analyzed data from the instruments of each site over the network so that other sites could use the data to control either sound synthesis or graphics.

2 Remote connections between Pd objects

A pair of objects, `netsend` and `netreceive`, serve to connect different computers running Pd. `Netreceive` opens a TCP/IP socket and "listens" for connections on it. (This terminology is taken from the TCP/IP protocol the `net` objects are built upon.)

On another computer, a `netsend` object can then be asked to connect to the `netreceive`'s socket, using its internet host name and port number. If the connection succeeds, the `netsend` can be asked to forward any Pd message to the receiving computer.

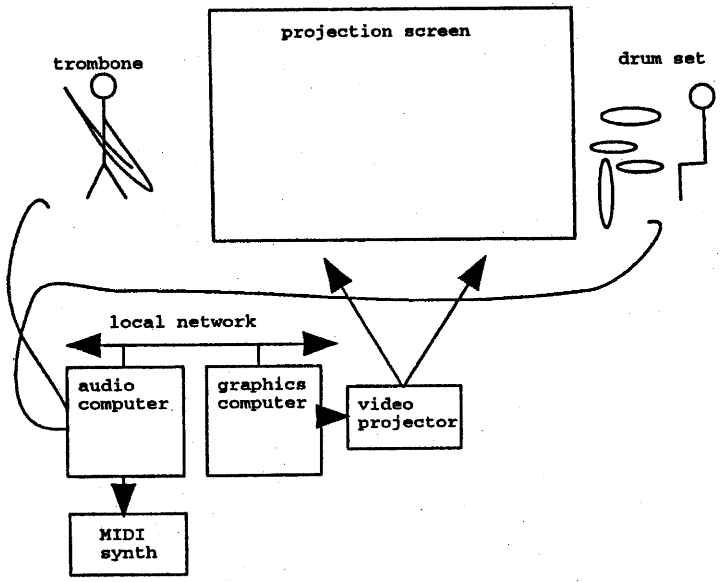

Figure 1: the setup for *Lemma 1*.

There is no guaranteed maximum latency for the transfer; this depends on how the messages are routed over the internet. Overall latency might be reduced by making a direct telephone connection between the two machines, either analog or ISDN, and running TCP/IP directly over the phone line. There is also no built-in time stamping feature in `netsend/netreceive`, although the designer of a Pd patch could easily embed time stamps in some or all of the messages transmitted.

3 Pitch tracking and percussion detection

An important limitation of Max/ISPW has been its weakness in signal analysis, simply because Max lacks the data-handling facilities to be able to use the results of an analysis effectively. So for example the (unpublished) `jack~` object can be instructed to output a spectral analysis of its input (giving the frequencies and amplitudes of its sinusoidal components), but there is no efficient way to *store* collections of such data, so the only effective way to use the analysis is to send it immediately to a resynthesis algorithm. This can be effective (as Cort Lippe's *Music for Piano and ISPW* shows) but much more power and flexibility are needed.

Pd's pitch tracker, `fiddle~`, is a fourth generation adaptation of Terhardt's algorithm; the first implementation was by Boris Doval and John Kitamura. The current implementation can work monophonically or polyphonically (up to three voices), but results are reliable only for well-behaved monophonic sounds. Like the Max/ISPW pitch trackers `pt~` and `pitch~`, the new `fiddle~` has outputs for both continuous and discrete information.

The continuous outputs appear after each analysis period (which is specified by a creation argument to `fiddle~`). The overall amplitude of the signal is output in dB; then for each pitch output (numbering from one to three), the momentary pitch and amplitude is output (or zero for outputs which have no pitch to report.) In principle, then, these outputs could be used to drive a synthetic reconstruction of the sound being analyzed.

The discrete outputs report stabilized pitches and attacks. A stabilized pitch is found if any of the (from one to three) pitch tracks has varied no more than a certain amount over a certain period of time (both quantities may be set by the user.) An attack is a sufficiently abrupt rise in overall amplitude

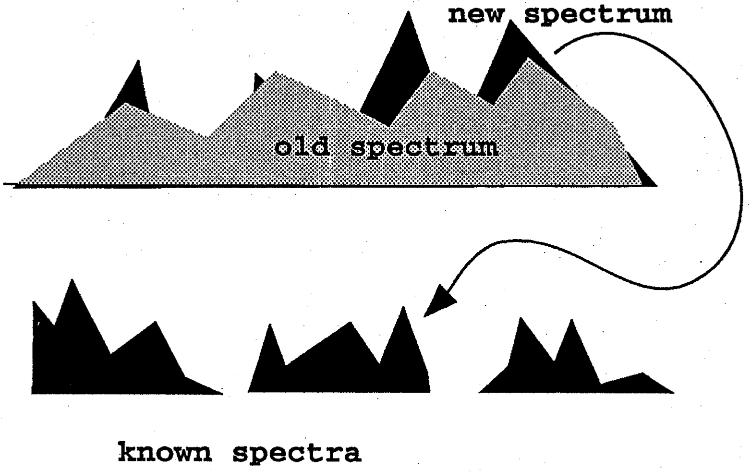

Figure 2: The "bonk" object measures the growth in the spectrum of a sound over a short interval of time. If the energy rises enough overall, the "new" energy (in black at top) is compared with a collection of pre-stored spectral templates to determine what kind of attack has occurred.

(where "abruptness" is defined through other user-specifiable parameters.) Discrete pitches and attacks might be interesting in a score following context, whereas the continuous outputs might be more useful for analysis/resynthesis applications.

It would be very useful to extend the functionality of `fiddle~` to report spectral envelopes, both of detected pitch tracks and of the signal overall, not only for analysis/resynthesis applications but also because the spectral envelope might be an interesting source of control (over a graphical synthesis process as in *Lemma 1*, for example.) This development will have to await a time when Pd offers a bit more of the necessary infrastructure.

The bonk~ object is intended for analysis of noisy sounds (as opposed to sounds with discrete spectra, which jack~ and fiddle~ are concerned with.) Rather than starting with an FFT and finding peaks, the analysis front end of bonk~ is a constant-Q (half-octave) filterbank giving 21 envelopes at different regions of the spectrum. Since bonk~ is intended for use with percussion instruments, the analysis window is kept extremely small (six milliseconds); the analysis is carried out every three msec. (In contrast, a typical analysis window for fiddle is 20-40 msec.)

It is well known that envelope following alone is usually inadequate to detect percussive events. In most percussion playing, new instruments are often hit while the same or a different instrument is still "ringing;" the relative amplitude increase may be small compared to the natural amplitude fluctuations in the sound of ringing instruments. The approach taken for bonk~ is to hope that a new instrumental strike will at least be marked by a rapid rise in the energy in *some* frequency band, even if the overall energy does not rise. The energy growth (shown in black in Fig. 2) can then be compared to a collection of energy templates corresponding to the known spectra of the instruments to be played. The spectrum which matches most closely furnishes a guess as to which instrument has been hit.

4 Platforms

Four platform families now enjoy some level of Pd support. At UCSD, we maintain Pd for Intel platforms running the NT operating system, and for IRIX (the SGI operating system.) The Gem package also runs on both of these, using their indigenous OpenGL systems. In principle, Pd could be run under Windows 95 as well as NT, but there will doubtless be sound and MIDI driver trouble, and no OpenGL package is known to exists there.

Guenther Geiger has ported Pd and Gem to Linux for PC compatible computers. This platform offers a high-quality, low-cost alternative to Microsoft operating systems.

Finally, in a most unexpected but welcome return to Max's root platform, David Zicarelli has taken Pd's audio package as a starting point for his very highly acclaimed MSP ("Max Signal Processing") package, which runs under Opcode Max. Although the long-term viability of the Mac as a real-time platform is uncertain, in the next few years (and perhaps for much longer than that) MSP will be the most popular real-time signal processing package available.

5 Acknowledgements

This research, and the series of improvisations are both made possible by a generous grant from the Intel Corporation. We have also received support from SGI and E-mu corporations.

References

[1] Danks, M., 1997. "Real-time image and video processing in GEM." Proc. ICMC, Thessaloniki, pp. 220-223.

[2] Puckette, M., 1996. "Pure Data: another integrated computer music environment." Proc. the Second Intercollege Computer Music Concerts, Tachikawa, pp. 37-41.

[3] Puckette, M., 1997. "Pure Data." Proc. ICMC, Thessaloniki, pp. 224-227.