PTL, a new sequencer dedicated to graphical scores

Damien HENRY*

*damien.henry@dh7.net

Abstract

PTL is a new kind of music sequencer in his first stage of development. Limitation of MIDI for contemporary music is well known. Nowadays, there is no sequencer dedicated to real time synthesis environment such as pure data from Miller Puckett or Max from IRCAM. PTL is an attempt to fill this lack. Unlike classical sequencers based on MIDI protocol, PTL permits any kind of message (number, text, array, etc…) to be sent (using different protocols like OSC, or FUDI) to a virtual instrument and to be graphically represented on a time line with a symbol created by the user. With basic shapes like dots, segments and circles, the compositor can use his sensitivity to make a drawing of the score that expresses graphically his music. For convenience, some of the shapes characteristics chosen by the user, like coordinates or colors, can become the virtual instrument parameters. By making the link between a graphical parameter and a musical one (like a Y coordinate translated into a frequency), we create a rule that gives a meaning to the graphical score. So, editing graphical symbols with the mouse will imply some musical changes.
We can define PTL as a graphical language; the user draws symbols and tells to the sequencer how to interpret them automatically as messages to be sent to a virtual instrument. PTL can fit as well for real time and non real time use.

1 Introduction

Many composers, for creating an open score (Cardew 1967) or giving precise indications about how to play their music, choose to use graphical scores instead of the classical notation system (In figure 1, X-axis is time and the Y-Axis is the frequency).

Actually, graphical scores seem to be adapted for electronic music representation: designing curves and symbols is a natural way of driving a virtual electronic instrument.

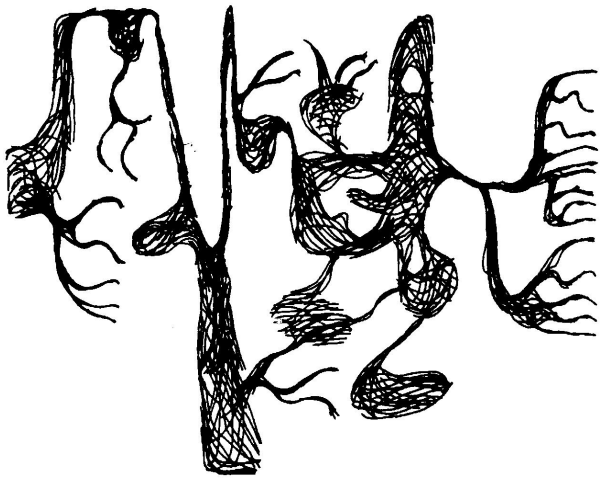

Figure 1. This Graphical score is a part of the famous Mycenae Alpha piece from I. Xenakis.

PTL is a computer tool that permits to create and automate the playing of such scores. It is not a closed notation system: the composer is free to use any symbol he wants for any purpose. It's a graphical tool to draw, translate and scale symbols or curves in way to design scores. It is also a sequencer able to interpret them in an automated way. The composer can create some translation rules to transform automatically a size or width into duration or frequency. This feature allows a very close relation between the graphical score and the generated sounds. It's both possible to draw the whole piece structure and to model each sound independently: PTL gives to the composer the freedom of using a single tool for macroscopic and microscopic notation of his music. It can be used to create open scores for acoustical instruments too.

PTL has also most of a classical sequencer features: it is able to record musical data in real-time, to represent them graphically, edit them, and to interpret them as a musical score.

PTL uses Open Sound Control (OSC) which is an interesting alternative to MIDI and becomes a wide spread technology (Wright and Freed 1997).

1 What is new with PTL

PTL seems to be the only software really dedicated to graphical scores.

A few other ones deal with music composition and have features for music representation, like OpenMusic (Ircam), Elody (Grame), Iannix (la-kitchen) and pure-data (M. Puckett).

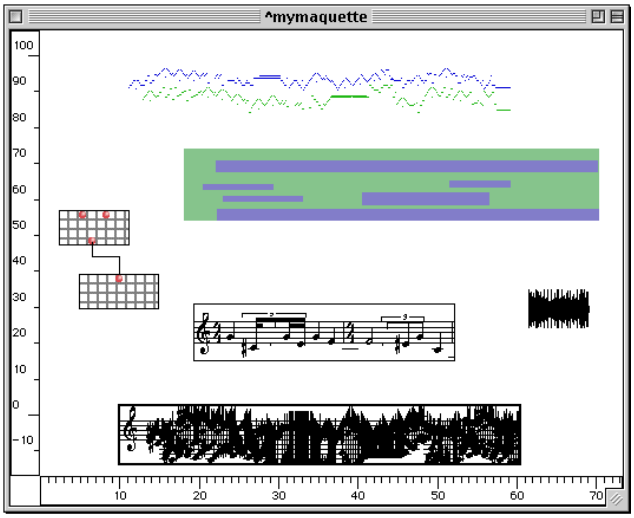

Figure 2. This example shows the "maquette" feature of OpenMusic.

The "Maquette" Object from "Open Music" (figure 2) is able to combine chords, curves, a MIDI sequence and sound files in the same window but without giving the user a full access to the events graphical representations, which are defined by the software.

In "Elody" the "time-line" and "sequencer" objects are very poor regarding to graphical diversity too.

Iannix is more dedicated to time and space experimentation than to graphical score manipulation.

On the other hand, pure-data is really powerful to design precisely graphical scores (figure 3). Through data-structures and templates (Puckett 1996), the composer can use geometric shapes for any purpose. Using pure-data as a sequencer for recording, drawing and playing a graphical score seems to be possible but uneasy: it's a skilled work to create such a complex pd patch. Moreover, basic features like zooming or splitting events are not implemented.

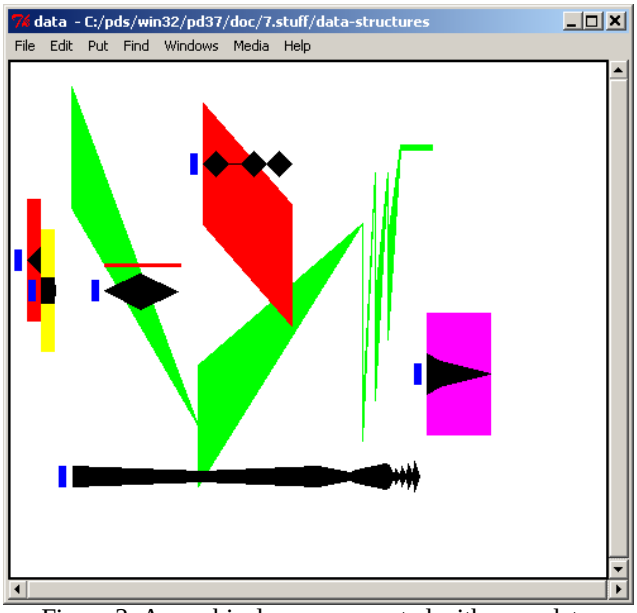

Figure 3. A graphical score generated with pure-data.

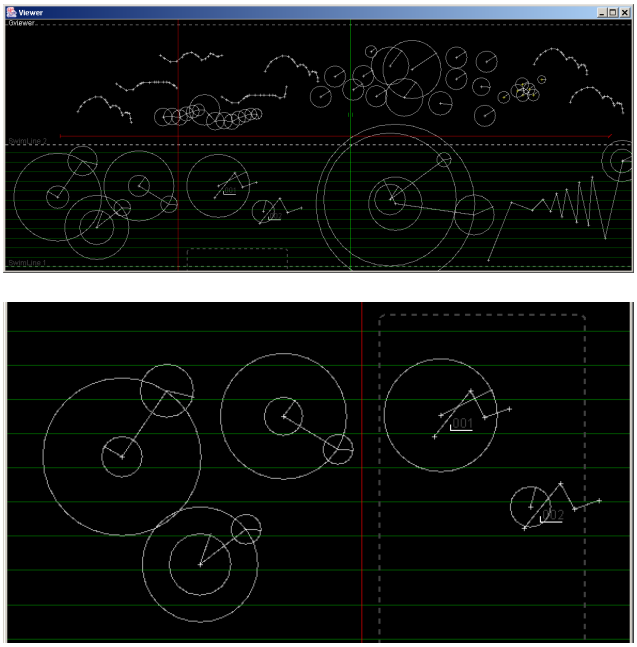

Figure 4. A graphical score generated with PTL. (The whole view and details).

With PTL (figure 4), the graphical aspect of the score comes first. The graphical representation is in the hand of the composer: like this, a sound can look many different ways. The composer can create a graphical score that is expressive by itself: some shapes fit with the idea we have of a sound, whereas some others don't.

2 Design

The PTL paradigm is the *event*. In PTL everything is an *event*. All the *events* take place into a hierarchical tree (figure 4). Other concepts are *messages*, *symbols* and *rules*.

- A *message* is what is exchanged between PTL and a software synthesizer. For a sequencer, a musical piece is a list of *messages* that must be sent at the right time.
- A *symbol* is a geometrical shape located on a time line and created by the composer to symbolize a sound or anything else.
- A *rule* is an automated way to create a *message* from a *symbol*.
- An *event* is a *message* associated with a *symbol* with a *rule*.

The hierarchical tree is both used to structure the score (to store events by tracks, instruments or parts; the organization of a score is up to the composer) and to create compound events.

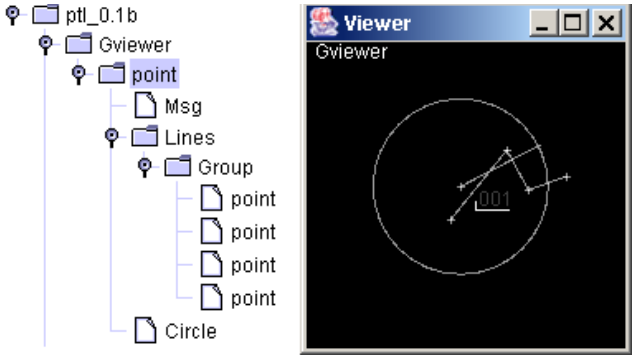

Figure 4. A compound *symbol* is defined by combining basic *events*.

2.1 Events

There are 3 kind of *events* :

- Basic *events* such as dot, circle and msg.
- Transformer *events* that alter the behavior of all the *events* that are their Childs in the tree.
- Compound *events* that are created by the user.

All *events* are objects derived from one class. Characteristics of the object are coded by overriding the following inherited methods:

DoThis(String a_cmd) is called by the user when an additional method is required.

Draw(…) is called to draw the *symbol*.

Split(…when) is called when the user wants to split the *event*.

Render(…) is called when PTL wants the *event* to calculate the *message* in way to send it to the software synthesizer.

ReactMouseEvent(…) is called when the user manipulates the *symbol* with the keyboard or with the mouse.

Through those 5 methods, an *event* can have his proper appearance and behavior.

Creating new *event* objects is the way to extend PTL.

2.2 Transformer Events

A transformer *event* is an event that changes the appearance and behavior of the events who are his children on the hierarchical tree. For example, on figure 4, the line object is a transformer that adds lines between the dots (the four "points" objects). It also interpolates the value between the dots to create a continuous value.

2.3 Compound events

With PTL, it is easy to create complex *symbols* and *messages* from basic shapes, by combining them in the hierarchical tree. Figure 4 shows the example of a *symbol* created by the addition of a circle, a msg object (here 001) and a line object. The *messages* sent to the software synthesizer can be automatically generated by the following parameters:

- A radius and an angle from the circle object.
- The content of the msg object (here 001)
- The shape designed by the 4 points and the line object.

For example the *messages* can be "Instrument number = msg, Level = radius, Envelope=lines".

Note that the *messages* sent could be anything else. This allows a complete dissociation between the graphical and the musical side.

3 Implementation

PTL is written in JAVA 2.0 under the netbeans IDE. All the code is platform independent.

To get the last release of PTL send a mail to ptl@dh7.net

PTL is provided under a bsd licence.

4 Conclusion

PTL is a tool for music notation that uses a graphical interface based on geometrical shapes understandable both by humans and computers. Graphical symbols help the human to have a good mental representation of the piece he is writing and give to the computer precise data to automate the play of the piece.

Such a multi purpose software can be particularly useful for mixt music, where acoustic instruments are played live in addition to electronic sounds.

In the future PTL will offer the possibility to edit one *event* with different *symbols*. This will give the composer the ability to create a musical piece that has more than one graphical representation, and to edit it by different views.

References

Agon, C. (1998). "OpenMusic : Un langage visuel pour la composition musicale assistée par ordinateur", Thèse, Paris VIII.

Assayag, G., Rueda, C. (1993). "The Music Representation Project at IRCAM", Proceedings of the International Computer Music Conference 1993, 206-209, Tokyo.

Braut, C. (1995). *Le livre d'or de la norme MIDI*, tome 2, éditions Sybex, Paris.

Cardew, C. (1967). *Treatise*, Gallery Upstairs Press, Buffalo, New York, U.S.A.

Coduys, T., Lefèvre, A. and Pape, G., (2003). "Iannix" Proceedings of the JIM 2003.

Orlarey, Y., Fober, D. and Letz S. (1997). "Elody : a Java+MidiShare based Music Composition Environment" Proceedings of the International Computer Music Conference ICMA 1997, 391-394

Puckette, M. S. (1996). "Pure Data: another integrated computer music environment" Proceedings of the International Computer Music Conference 1996, 37-41.

Wright, M. and Freed, A. (1997). "Open Sound Control: A New Protocol for Communicating with Sound Synthesizers" Proceedings of the International Computer Music Conference 1997, Thessaloniki, Hellas, pp. 101-104.