Graphical NeXTSTEP Objects as FTS Clients to Control Instruments in the new FTS Client/Server Architecture

Todor Toderoff and Carolize Traube
Facülta Polytechnique de Mons
todero@multique.fts.ac.be

Abstract
We are designing graphical objects within NeXTSTEP which connect as Clients to the new FTS Server. Those objects provide better ways of controlling and visualising the FTS processes than those presently available within MAX/FTS, offering new multidimensional graphical controllers and viewers as well as better table editing tools. They are also intended to help mapping and visualising incoming data from a PowerDove, from MIDI faders and from various other sensor devices. At the conference, we will demonstrate those tools in conjunction with several instruments, performing granular morphing, spatialisation of sounds and transformation through analysis-resynthesis.

1 Introduction
The MAX environment is a very powerful tool for developing and prototyping sound processing algorithms. But, though it might seem easy for the experimented MAX user to modify a patch in such a way that it responds as he wishes to external MIDI interfaces, composers who usually lack that knowledge and experience are often helpless when it comes to modify rather simple things. Specially as a MAX/FTS window is usually invaded by patcher boxes and connections one cannot hide.

MAX also lacks of any kind of multidimensional controller: there are buttons, number boxes, sliders and tables, but one cannot call the table object a real multidimensional controller to play with. And composers who have worked with programs like Hyperreum or GRM Tools now the difference a simple x-y controller can make when it comes to controlling sound processing. Going further, a graphical interpolation tool like the one offered on the SYTER at the GRM can transform a well designed sound processing tool into a very expressive instrument. Such a graphical tool was programmed by Adrien Lefèvre for Opcode MAX, but was not yet available for NeXT-1PSW users. We designed one that increases further the expressiveness by making it controllable by any MIDI controller.

All this has been made possible by the new FTS Client/Server Architecture [Dechelis and De Cecco, 1995] which allows bi-directional communications with several clients at a time in a well documented manner. We will first describe with an example how the communication system between NeXTSTEP and FTS works. Then we will propose several graphical objects.

2 NeXTSTEP objects as FTS Clients
Several clients may connect themselves dynamically to the FTS server, each client may either be on the same machine or on a different one connected through Ethernet. To allow communication from a graphical NeXTSTEP object to the FTS server, the client has only to initialise the connection and send messages to the desired "receive" object within FTS (with the same name as the corresponding MAX "send" object) and a special object of the type "get" that will transfer the information to the client. The client then establishes a connection between those two objects and has to poll regularly to check whether some new messages have arrived. Those modules created by the client are not visible in the MAX patch, as they are directly created within FTS. We show on figure 1 how we implemented a simple 2-D controller as well as a 2-D visualisation tool. They might of course be combined, but we will use this as an example. The simple MAX Patch depicted allows to receive lists of x-y co-ordinates through a "receive" object named "from_client" and to send lists of x-y co-ordinates through a "send" object named "to_client".

The graphical client has two windows: one allows to move a circle with the mouse or with sliders, and the other receives the x-y co-ordinates lists and displays a circle at the corresponding position on a 2-D map. Special care has to be taken in dealing with the amount of messages sent to the graphical client and with the frequency of the polling performed by the client. In order to keep the patch simple, we didn't show those modules limiting the sending of new values to one every 100 ms. Without this, the graphics may appear very slow, displaying every intermediate value and needing time to catch up before the last one is displayed. More efficient procedures, like allowing the client to put the entire queue of messages in one call, might be implemented in the future, as FTS is still in Beta version. This mechanism of sending and receiving values allows one client to...
access several 'send' and 'receive' boxes, as well as several clients to access the same ones. It is therefore possible to open many graphical clients at the same time.

![Figure 1: 2-D control and visualization](image)

3 The interpolation window

Inspired by the "Interpol" window which was one of the strong points of the SYTER at the GRM [Terzini, 1991], our object has some added features, like allowing to choose between several interpolation laws and allowing the current point to be moved either with the mouse or through messages sent by FTS that may come from MIDI controllers, like the PowerGlove that we use with the STEIM SensorLab. Each circle may be activated or deactivated and the configurations may be stored and recalled. The compare button is enlightened when the selected preset has been modified. The lists of weights are sent to FTS to control the sound processing. The right part of the window may be hidden at any time to free space on the screen. It is not usually needed as moving and resizing circles is done with the right mouse button. In order to make full use of the PowerGlove as a 3-D controller, we plan to implement a 3-D version of this interpolation concept.

![Figure 2: The interpolation window](image)
4 Control over the spatialisation of sounds

A simple movement generator, with several types of movement will be further developed to include a larger library of shapes and to be able to interact smoothly with MIDI controllers. Though the generator is presently coded within NEXTSTEP, we will move it as an external object within P5X, controlled by the graphical object. This will lower the flow of messages between the client and the server, as well as authorise higher speed and better precision. While the sphere is moving within the 3-D space, it draws behind itself the last ten points of its trajectory, giving both an indication about the shape of the movement and about its speed. This tool may also be used to control the current position in the interpolation window, creating in a timbre space a trajectory of which speed and shape may be controlled by a MIDI controller.

Figure 3: The movement generator with a 3-D viewer

As we want to use the fingers bending to change some of the parameters, we also designed a little viewer to visualise them. We plan to place next to each finger a little button that will open a table to adjust the bending sensitivity according to the user's morphology and to the sound process he wants to control.

Figure 4: The viewer for fingers bending
5 A table editing tool

We wanted a easier way to create and scale table values. With this tool, x and y may be floating points values, they don't have to start at zero, may be scaled simply by entering new limits in the number fields, and the number of points may be changed with ease. Segments of lines may also be easily drawn and the exact values of the points may be displayed in the right side of the window. That part of the window may also be hidden when it isn't needed anymore.

![Graphical interface](image.png)

Figure 5: The table viewer and editor

6 Summary

Those graphical objects are still sketches that we need to improve. They are as new as the FTS Beta version is, but they give an idea of what we want to offer: a more intuitive GUI for the composers of electroacoustic music thanks to a wider library of multidimensional controllers and viewers.

We are also working to give the unexperienced MAX user an easier access to data scaling, mapping and modulation without having to add or remove MAX modules and connections. This will help giving a more transparent access to composers who are not necessarily familiar with the MAX language.

For each signal processing instrument (granular synthesis, spatialisation of sound, transformations by analysis-synthesis, ...), several sets of efficient mappings will be proposed and may be later modified and saved by the composer for future use. This will give a fast idea of the importance a different mapping may have in the use of an instrument and on the sound it produces. Using those presets, the composer will be able to design his own mappings much faster than starting from scratch.

We have discussed in (Todoroff, 1995) how we think that this approach might also help to generate more convincing sound morphologies by creating more "specialised" instruments to choose from on top of more universal and less directive sound processing techniques.

Due to the fact that those tools are build within NeXTSTEP, they offer a high level of modularity and allow the user to easily customize his working environment: using tables or scaling whenever needed, opening several 2-D or 3-D interpolation spaces at the same time, visualising different time or frequency-domain signals, editing filter transfer functions, etc.

References


This research has been funded by the Région Wallonne in Belgium.