jMax : un environnement pour la réalisation d'applications musicales temps réel sur Linux

François Déchelle (dechelle@ircam.fr)
IRCAM, 1 place Igor Stravinsky, 75004 Paris.

Résumé : jMax est un environnement programmable pour la réalisation d'applications musicales et multimédia interactives. Ce logiciel est une nouvelle implémentation de l'environnement Max déjà connu sur la plateforme Macintosh. Un bref historique des différentes versions de Max est donné. L'environnement jMax est décrit, explicitant les choix effectués pour obtenir une grande portabibilité. Les spécificités de l'interface graphique utilisateur de jMax sont présentées. La version Linux de jMax est détaillée : intégration de jMax avec les drivers audio et MIDI Linux, comparaisons de performances entre les différentes plateformes.

1 Introduction

jMax est une nouvelle implémentation de l'environnement Max qui a été largement adopté pour la réalisation d'applications musicales interactives mettant en œuvre les technologies informatiques. Remplaçant la "Station d'Informatique Musicale" de l'Ircam par une solution purement logicielle, jMax fonctionne sur le système Linux.

2 Le langage Max

Le langage Max est souvent présenté comme un langage de programmation visuel : l'interface graphique, réutilisant la métaphore du synthétiseur analogique modulaire, permet à l'utilisateur de construire une application sous forme de "patch" par connexion de modules [10, 19]. Ces connexions représentent des communications de paramètres de contrôle ou d'échantillons sonores, avec une sémantique d'envoi de messages à la manière de langages objets du type Smalltalk. Un module peut être une unité de traitement (opérateur arithmétique simple ou transformation complexe du signal) ; il peut aussi représenter des données, par exemple une ligne à retard, ou enfin être un module système d'entrées-sorties audio ou MIDI. Les modules peuvent être des primitives, appelées "objets", ou bien peuvent eux-mêmes être des patchs, donnant à un patch une structure hiérarchique. Enfin, certains modules "contrôleurs" ont un comportement graphique interactif et permettent l'envoi de message depuis l'interface graphique. La figure 1 donne un exemple de patch jMax.

Max peut être pris comme un langage de programmation visuel, et peut à ce titre être rattaché à la famille des langages fonctionnels impératifs [10], les fonctions étant les “objets” du patch. Mais une autre caractéristique intéressante de Max, qui entre pour une part essentielle dans son succès, est qu'il peut aussi être vu par le programmeur de patchs comme un générateur d'interfaces, à la manière des constructeurs d'interfaces disponibles dans beaucoup d'environnements graphiques. Un patch est en effet la description d'un algorithme de traitement, mais également une interface graphique de contrôle de cet algorithme.

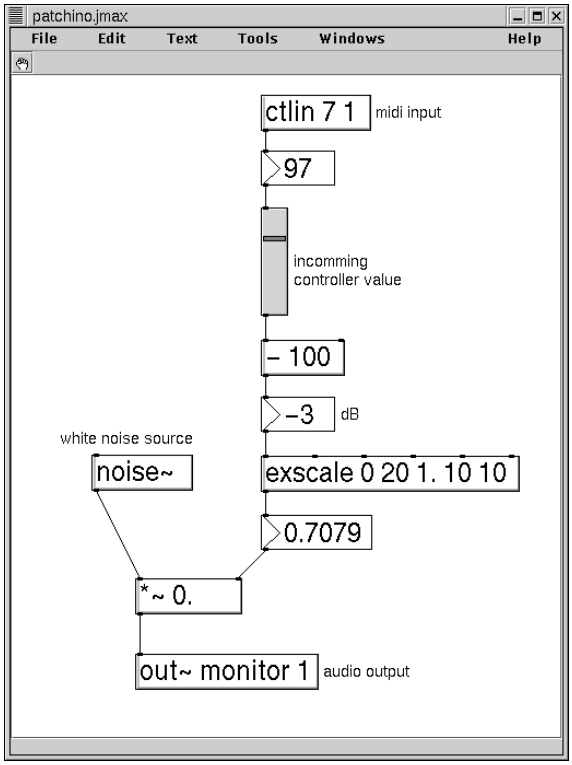

FIG. 1 – Un exemple de patch jMax

3 Un bref historique de Max

Max tire son origine de l'éditeur Patcher écrit par Miller Puckette en 1988 [16]. Ce logiciel, fonctionnant sur Macintosh et dédié au traitement de contrôle MIDI a été utilisé pour la première fois en situation de concert pour la pièce de Philippe Manoury "Pluton", réalisée avec le processeur 4X [12, 16, 19].

Le "Patcher" a ensuite été cédé à la société américaine Opcode où, récrit par David Zicarelli, il a donné un produit très populaire [11], introduisant de nombreuses améliorations de l'interface graphique utilisateur ainsi qu'une interface de développement d'objets externes qui permet d'étendre l'ensemble d'objets Max primitifs [21].

Le projet "Station d'Informatique Musicale" de l'Ircam a démarré en 1989, visant à remplacer le processeur 4X [13]. Parmi les développements logiciels de ce projet figurait une nouvelle version du logiciel "Patcher", qui intégrait le traitement temps réel du signal audio dans le même paradigme de connexion [17, 18]. Cette version, distribuée avec le matériel de la Station d'Informatique Musicale et connue sous le nom de Max/ISPW, apportait une nouveauté sous la forme d'une architecture à deux composants logiciels, d'une part l'interface graphique utilisateur fonctionnant sous l'environnement NeXTSTEP, et d'autre part un exécutif temps réel nommée FTS [18], fonctionnant sur les cartes additionnelles de la station (cartes bâties autour du processeur Intel i860).

La nécessité de remplacer le matériel de la "Station d'Informatique Musicale" a conduit à la décision de proposer une solution purement logicielle, cette décision étant justifiée par l'évolution de la puissance de calcul et le coût des développements matériels [9]. La consé-

quence logique de cette décision a été de réécrire Max/ISPW en privilégiant la *portabilité* : dans ce but, l'interface graphique utilisateur a été séparée du moteur d'exécution temps réel, rendant ainsi indépendantes leurs évolutions. Ce moteur d'exécution était une nouvelle version de l'exécutif FTS de la Station d'Informatique Musicale et le nom de Max/FTS a donc été naturellement donné à cette version, distribuée pour stations Silicon Graphics [9, 6, 7, 3, 4].

Le développement du logiciel PD [19, 20] par Miller Puckette a débuté à la même époque ; cherchant à remédier à certaines faiblesses de Max dans le domaine de la gestion de structures de données dynamiques, PD utilisait dans ce but les principes du logiciel Animal [1] développé dans le cadre du projet "Station d'Informatique Musicale" et introduisait une interface graphique portable basée sur l'environnement Tcl/TK. En 1997, David Zicarelli, réutilisant le module de traitement audio de PD, distribuait le logiciel MSP ("Max Signal Processing"), qui introduisait dans Max/Opcode la synthèse et le traitement temps réel du signal.

Le langage Java a donné la possibilité de réaliser des interfaces graphiques qui sont réellement multi-plateformes. L'interface graphique de Max/FTS a été réimplémentée en Java et le nom de jMax (pour Java Max) a été donné à cette nouvelle implémentation du langage Max [14, 3, 4, 5, 8, 2].

4 L'architecture de jMax

Comme indiqué précédemment, jMax réutilise l'architecture à deux composants de Max/FTS (et PD), architecture souvent dénommée "architecture client/serveur" dans d'autres domaines de l'informatique. Cette architecture permet de décorréler les développements de l'interface graphique utilisateur (le "client") et du moteur d'exécution temps réel (le "serveur"), offrant donc une plus grande garantie de portabilité. Elle permet également l'exécution des deux composants sur des machines différentes ; ceci offre des possibilités intéressantes pour la situation de concert où l'interface graphique peut s'exécuter sur un ordinateur portable et le serveur sur une machine machine multi-processeurs plus lourde. Cette architecture rend enfin possible l'exécution du moteur temps réel sans interface graphique, par exemple à l'intérieur d'un environnement à “plug-in” du type VST.

La communication entre les deux composants s'établit via un protocole indépendant du matériel et peut utiliser une connexion réseau (socket TCP/IP ou UDP) comme couche de transport [9, 6, 7].

5 Le serveur de traitement temps réel FTS

L'architecture du serveur de traitement temps réel FTS est décrite dans [6, 3, 5]. Elle est organisée autour de l'interprèteur du langage Max et du moteur d'exécution des opérations de traitement de signal. Un séquenceur gère l'exécution des opérations de contrôle, déclenchées par des opérations d'entrées-sorties (MIDI ou signal), par des actions de l'interface utilisateur ou par des primitives temporelles de type alarme.

Le noyau de FTS utilise plusieurs threads afin de tirer parti des machines multi-processeurs modernes, qui sont pour la plupart des architectures à mémoire partagée et cache cohérent et supportent la programmation au niveau thread. L'interface de programmation actuellement utilisée est l'interface POSIX, mais l'implémentation peut utiliser

d'autres interfaces de programmation.

L'architecture multi-thread est utilisée en particulier pour l'implémentation des accès temps réel aux fichiers de sons : la lecture et l'écriture des fichiers de sons s'exécutent dans des threads séparés, permettant ainsi sur des machines multi-processeurs de disposer de plusieurs canaux de lecture de fichiers de sons tout en conservant la puissance d'un processeur dédiée au traitement de signal et de contrôle.

Une autre utilisation de cette architecture est la possibilité d'avoir plusieurs threads d'exécution du contrôle, ces threads ayant des priorités différentes. Dans cette configuration, des tâches de contrôle lourdes peuvent s'effectuer de manière asynchrone dans des threads à basse priorité sans interrompre le traitement de signal alloué à une priorité supérieure, éliminant ainsi le recours à une augmentation de la taille du tampon audio et donc de la latence.

La portabilité de FTS est déterminée par la séparation stricte du code dépendant du processeur et du système opératif (accès à la mémoire, accès à l'unité flottante) et par l'introduction d'un concept de "devices" explicité ci-dessous.

5.1 Les "devices" FTS

Le problème essentiel de portabilité posé par le serveur FTS est le code lié aux entrées-sorties audio et MIDI, qui dépend non seulement du système opératif mais aussi de l'interface logicielle d'accès au matériel et qui est donc intrinsèquement non portable. L'approche choisie, inspirée du concept de "device" Unix, a été d'établir une interface de programmation générique qui encapsule les fonctions d'entrées-sorties de façon indépendante du système et du matériel.

Une "device" FTS [6] est une abstraction supportant un petit nombre d'opérations d'entrées-sorties et de contrôle. Cette interface contient les opérations définies par le modèle Unix : `''open''`, `''close''` et `''ioctrl''` pour la modification dynamique des paramètres. Les opérations d'entrées-sorties sont toutefois étendues par rapport à ce modèle, fournissant à la fois des opérations sur des caractères et des opérations sur des vecteurs, ces dernières étant introduites pour gérer les tampons d'échantillons.

Cette approche restreint le code devant être récrit pour un portage à un petit nombre de lignes de code, typiquement de l'ordre de la centaine, les fonctions devant être réimplémentées étant par ailleurs complètement spécifiées.

Les "devices" fournissent également à l'utilisateur une grande flexibilité dans la configuration de l'environnement d'exécution d'un patch : un patch ne contient que des objets d'entrées-sorties opérant sur des "devices" logiques qui ne sont connues que par nom, l'association entre le nom et le dispositif physique d'entrées-sorties étant réalisée à l'extérieur du patch. La communication avec un contrôleur externe peut être rendue transparente pour le programmeur de patch, qu'elle s'effectue sur une ligne série, un port parallèle ou une connexion réseau par socket.

6 L'interface utilisateur de jMax

L'interface utilisateur de jMax offre essentiellement les mêmes fonctionnalités que les autres déclinaisons de Max : un éditeur de patch pour la construction et le contrôle des patchs et un ensemble d'éditeurs spécialisés pour les objets qui représentent des structures de données complexes ayant leur propre mécanisme d'édition, telles que fonctions tabulées

ou séquences. Une caractéristique importante de jMax est sa portabibité, découlant pour partie du choix de Java comme langage d'implémentation de l'interface graphique, mais également de choix d'architecture qui ont été évoqués.

jMax fournit par ailleurs une fonctionnalité nouvelle sous la forme d'un langage de script intégré dans l'interface graphique.

6.1 Langage de script

L'introduction dans jMax d'un langage de script vise à donner à l'utilisateur un langage textuel qui fournit des primitives n'ayant pas d'équivalence graphique simple, par exemple des opérations répétitives telle que la création de bancs d'opérateurs. L'utilisation d'un langage de script permet aussi d'intégrer dans la même représentation le mécanisme de configuration [3, 8].

Le langage de script actuellement utilisé est Tcl [15], langage de script simple ayant une implémentation Java. Etant donné toutefois la faible intégration de Tcl et de Java d'une part, le manque de support de la version Java de Tcl d'autre part, il a été décidé d'abandonner Tcl au profit de Scheme, dialecte du langage Lisp.

La figure 2 présente deux exemples d'utilisation de scripts Tcl dans jMax. Le premier exemple est un extrait du fichier de configuration propre à chaque utilisateur, fichier qui est chargé au lancement de la session jMax. Le second exemple est une fonction Tcl simple qui crée un banc d'objets du même type.

```
# Fichier de configuration
# Machine sur laquelle tourne le serveur
set jmaxHost "astor.ircam.fr"
# Cette machine est une SGI Origin
set jmaxHostType origin

when start {
  jmaxSetSampleRate 44100     # 44.1 kHz
  jmaxSetAudioBuffer 1024     # tampon d'E/S

  openDefaultAudioIn adatIn   # entrée: ADAT
  openDefaultAudioOut adatOut # sortie: ADAT
  openDefaultMidi midi2       # MIDI: port 2
}
```

```
## Cette fonction Tcl crée un banc
## série d'objets de la classe <class>
proc bank { class x y n} {
  set prevObj [ObjNew $class $x $y]
  for {set i 1} {$i < $n} {incr i 1} {
    # Create object
    set obj [ObjNew $class $x $y]
    # Connect to previous
    connect $prevObj 0 $obj 0
    # Move y position
    incr y [expr [ObjGetHeight $obj] + 7]
  }
}
# Exemple d'utilisation:
bank "voice~" 10 10 64  # Banc de 64 voix
```

FIG. 2 – Exemples de scripts : configuration, création d'un banc d'opérateurs

7 Plateformes supportées et distribution

jMax est supporté actuellement sur les plateformes suivantes :

– stations Silicon Graphics fonctionnant sous le système IRIX

– PC ou Macintosh fonctionnant sous le système Linux

La version Linux de jMax utilise pour les entrées-sorties son et MIDI les drivers OSS ou ALSA, suivant le choix de configuration effectué par l'utilisateur.

Des portages sont en cours sur les systèmes Mac OS X et Windows.

7.1 Comparaison de performances

Pour la comparaison des performances des différentes plateformes sur lesquelles jMax est supporté, une mesure utile peut être le nombre maximal d'échantillons produit par un patch en un temps fixé. Cette fréquence d'échantillonnage maximale théorique peut être obtenue en faisant fonctionner le système hors synchronisation avec une horloge d'échantillonnage.

Le tableau 1 présente cette comparaison de performances, les patchs choisis étant un des patchs d'exemple du Spatialisateur et le patch de la pièce *En écho*, pièce de Philippe Manoury pour voix et électronique, créée sur la Station d'Informatique Musicale. On notera l'excellente performance de la version Linux qui, sur une machine de très bas coût, fournit une performance supérieure aux stations Silicon Graphics.

Plateforme			Patchs	
CPU	MHz	OS	Spatialisateur (8c-4r)	En Echo
R10000	225	IRIX 6.5	156	61
Pentium-III	600	Linux	256	73
G3	400	Linux-PPC	195	71

TAB. 1 – Comparatif de performances

7.2 Distribution

jMax est distribué sous la Licence Publique Générale GNU depuis juillet 1999. Les sources de jMax peuvent être accédées depuis le site jMax : `http ://www.ircam.fr/jmax`.

Références

[1] M. De Cecco and E. Lindemann. Animal - graphic data definition and manipulation in a real time environment. *Computer Music Journal*, 15(3) :78–100, 1991.

[2] F. Déchelle. jMax : un environnement de programmation pour l'interactivité et le temps réel. In *Interfaces homme-machine et création musicale*, pages 85–94. HERMES Science, 1999.

[3] F. Déchelle, R. Borghesi, M. De Cecco, E. Maggi, B. Rovan, and N. Schnell. jMax : a new JAVA-based editing and control system for real-time musical applications. In *Proceedings of the 1998 International Computer Music Conference*, San Francisco, 1998. International Computer Music Association.

[4] F. Déchelle, R. Borghesi, M. De Cecco, E. Maggi, B. Rovan, and N. Schnell. Latest evolutions of the FTS real-time engine : typing, scoping, threading, compiling. In *Proceedings of the 1998 International Computer Music Conference*, San Francisco, 1998. International Computer Music Association.

[5] F. Déchelle, R. Borghesi, M. De Cecco, E. Maggi, B. Rovan, and N. Schnell. jMax : an environment for real-time musical applications. *Computer Music Journal*, 23(3) :50–58, 1999.

[6] F. Déchelle and M. De Cecco. The Ircam real-time platform and applications. In *Proceedings of the 1995 International Computer Music Conference*, San Francisco, 1995. International Computer Music Association.

[7] F. Déchelle, M. De Cecco, E. Maggi, and N. Schnell. New DSP applications on FTS. In *Proceedings of the 1996 International Computer Music Conference*, San Francisco, 1996. International Computer Music Association.

[8] F. Déchelle, M. De Cecco, E. Maggi, and N. Schnell. jMax recent developments. In *Proceedings of the 1999 International Computer Music Conference*, San Francisco, 1999. International Computer Music Association.

[9] F. Déchelle, M. De Cecco, M. Puckette, and D. Zicarelli. The Ircam real-time platform : Evolution and perspectives. In *Proceedings of the 1994 International Computer Music Conference*, San Francisco, 1994. International Computer Music Association.

[10] P. Desain and H. Honing. Putting MAX in perspective. *Computer Music Journal*, 17(2), 1993.

[11] C. Dobrian. *Getting Started with Max, Max Reference Manual,and Max Manual Addendum 3.5.* Opcode Systems, Inc., Menlo Park, CA, 1995.

[12] G. Di Giugno and J. Kott. Présentation du système 4X processeur numérique de signal en temps réel. Rapport 32/81, IRCAM, 1, place Stravinsky, 75004, Paris, France, 1981.

[13] E. Lindemann, F. Déchelle, M. Starkier, and B. Smith. The architecture of the Ircam musical workstation. *Computer Music Journal*, 15(3) :41–50, 1991.

[14] E. Maggi and F. Déchelle. The evolutions of the graphic editing environment for the Ircam musical workstation. In *Proceedings of the 1996 International Computer Music Conference*, San Francisco, 1996. International Computer Music Association.

[15] J. Ousterhout. *Tcl and the Tk Toolkit.* Addison-Wesley., 1994.

[16] M. Puckette. The patcher. In *Proceedings of the 1988 International Computer Music Conference*, San Francisco, 1988. International Computer Music Association.

[17] M. Puckette. Combining event and signal processing in the MAX graphical programming environment. *Computer Music Journal*, 15(3) :68–77, 1991.

[18] M. Puckette. FTS : A real-time monitor for multiprocessor music synthesis. *Computer Music Journal*, 15(3) :58–68, 1991.

[19] M. Puckette. Pure Data. In *Proceedings of the 1996 International Computer Music Conference*, San Francisco, 1996. International Computer Music Association.

[20] M. Puckette. Pure Data : another integrated computer music environment. In *Proceedings of the Second Intercollege Computer Music Concerts*, Tachikawa, Japan, 1996.

[21] M. Puckette and D. Zicarelli. *MAX Development Package Manual.* Cycling '74, Menlo Park, CA, 1991.