ALMIR
Un Atelier de Lutherie Modulaire Interactif temps Réel

Lirio MARTINEZ
Informaticien musicien, membre du GAIV
agossin@club-internet.fr,
lirio.martinez@alicesystems.com

1. Présentation

ALMIR est un logiciel qui permet de créer et de jouer en temps réel des systèmes instrumentaux virtuels modulaires.
Ces systèmes instrumentaux sont constitués de modules de production et de traitements interconnectés qui véhiculent et traitent des signaux sonores. Ils sont virtuels dans le sens où tous les traitements du signal sont effectués uniquement par logiciel.
Le terme *atelier* identifie un espace avec un ensemble d'outils permettant de créer des objets complexes à partir d'objets de base. La *lutherie* met l'accent sur la création d'instruments. La *modularité* désigne l'architecture basée sur des modules interconnectés. *L'interactivité* désigne la possibilité pour un opérateur d'interagir sur le comportement de l'instrument. Enfin, le *temps réel* désigne la propriété d'exécuter les traitements du son en temps réel, c'est à dire avec un temps de réponse non perceptible à l'échelle humaine.
ALMIR est constitué d'une bibliothèque de classes entièrement développée en C++. Bien que la version actuelle soit écrite pour Windows, le logiciel pourrait être aisément porté sur d'autres systèmes et plate-formes, car seules les entrées / sorties sur les périphériques de sons sont dépendantes du système.
ALMIR s'exécute sur tout PC standard pourvu d'un système Windows 32 bits (95, 98, 2000, XP) et du composant système DirectSound, ainsi que d'une carte son standard. Il ne nécessite aucun autre matériel spécifique.

1.1. Historique

L'histoire d'ALMIR remonte au début des années 1980. Après une introduction aux pratiques musicales électro-acoustiques dans le cadre d'études universitaires en musicologie, nous avons abordé l'informatique musicale au sein du GAIV (Groupe Art et Informatique de Vincennes) dirigé par G.G. Englert (ENGLERT 1980).
Nous utilisions à l'époque le système de synthèse sonore numérique hybride Synclavier 1 de la firme New England Digital (NEDCO 1978). Ce système comprenait un synthétiseur numérique permettant de jouer simultanément 16 canaux de sons, avec la possibilité de définir pour chacun la forme d'onde, la fréquence, une enveloppe audio et une enveloppe modulante sous forme d'une suite de segments, le signal modulant permettant de moduler en fréquence un autre canal.
A la fin des années 1980, désireux d'exploiter les ressources disponibles sur les ordinateurs individuels PC (système, graphique, environnement de développement logiciel), mais soucieux de continuer à utiliser un dispositif de synthèse sonore permettant de définir précisément les paramètres de base du son, nous avons conçu avec A. Llop une interface matérielle et logicielle permettant de piloter le synthétiseur du Synclavier depuis un PC.
Nous avons ensuite développé un " atelier logiciel d'informatique musicale interactif temps-réel " (MARTINEZ 1996) utilisant cette interface.
Avec l'accroissement en puissance des ordinateurs PC au milieu des années 1990, il est devenu possible d'effectuer la synthèse directe du son en temps réel sur l'ordinateur. Cela nous a conduit dans un premier temps à développer le logiciel V5 permettant d'émuler un Synclavier amélioré sur un PC.
ALMIR constitue l'aboutissement actuel de cette activité de recherche sur l'interactivité temps-réel et la synthèse sonore directe. Notre objectif est de créer des systèmes instrumentaux génériques et modulaires qui satisfont les contraintes du temps-réel sans matériel spécifique.
Nos influences ont été :

- MUSIC V pour le concept de modularité généralisé (MATHEWS 1969).
- Le synthétiseur analogique EMS VCS3 pour sa modularité, son interactivité temps-réel et son principe de commandes en tensions généralisé (GAIV 1977).
- Le système NED Synclavier 1 pour son procédé de synthèse sonore numérique offrant l'accès aux paramètres de base du son (NEDCO 1978).

- L'ISPW (IRCAM Signal Processing Workstation) pour sa modularité et ses possibilités de traitements du signal en temps-réel entièrement virtuels (LINDEMAN 1990, LIPPE, PUCKETTE 1991, 1993).

2. Description

2.1. Généralités

ALMIR pour Windows est basé sur le composant système DirectSound de Windows[1]. Ce composant définit une interface sonore de bas niveau rapide et indépendante des périphériques de sons installés sur l'ordinateur. DirectSound nous permet d'effectuer des entrées / sorties de signaux sur ces périphériques de sons, de fournir des informations sur leurs capacités, et de définir le format des signaux (fréquence d'échantillonnage, nombre de canaux, taille et nombre d'échantillons).
La figure suivante présente l'architecture système :

ALMIR permet :
- D'émettre et recevoir des signaux sonores sur des périphériques de sons.
- De traiter et véhiculer des signaux sonores d'un module de traitement à un autre.

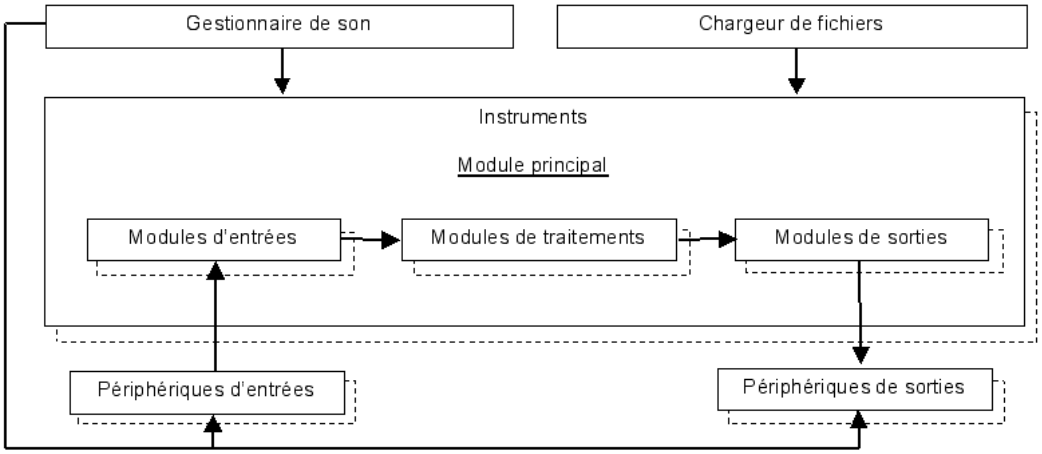
ALMIR utilise principalement deux types de composants :
- Les *modules* permettent de synthétiser et de traiter les signaux sonores.
- Les *interfaces* permettent d'interconnecter les modules pour véhiculer leurs données.

Des modules d'entrées et de sorties permettent de désigner les périphériques de sons sur lesquels les signaux sont reçus ou émis.
Il est également possible d'utiliser des *containers* d'interfaces ou de modules. Un container est un composant générique qui permet de grouper des composants unitaires (interfaces ou modules) de mêmes types, sous forme de listes ou de tableaux.

Un *instrument* est constitué d'un ensemble de modules interconnectés.
Il existe deux manières de construire un instrument :
- En créant individuellement tous les composants de l'instrument et en reliant les interfaces dynamiquement par instructions logicielles.
- En chargeant un *fichier d'instrument* qui contient une description des modules de l'instrument et de leurs interconnexions.

Une fois construit, l'instrument peut être *joué* : c'est la synthèse sonore proprement dîte. Cette opération est supervisée par le *gestionnaire de son*, de sorte que chaque module de l'instrument effectue périodiquement son calcul de signal individuel ; l'ordre dans lequel les différents modules sont sollicités est déterminé par la *chaîne de calcul du signal*, qui est pré-calculée afin d'optimiser les temps de traitements du signal.

La figure suivante présente l'architecture logicielle générale et ses différents composants :



Nous présentons ci-après de façon détaillée les différents composants d'ALMIR.

[1] Windows et DirectSound sont des marques déposées.

2.2. Les périphériques de sons

ALMIR utilise les périphériques de sons standards installés dans l'ordinateur et déclarés au niveau du système. Les périphériques sont identifiés au moment du démarrage du logiciel.
Deux types de périphériques sont utilisés :

- Les périphériques d'entrées (réception d'un signal provenant d'un microphone, de l'entrée ligne, d'un CD, etc.)
- Les périphériques de sorties (émission sur la sortie ligne d'un signal audio).

Plusieurs périphériques de sons peuvent être installés sur un ordinateur. La sélection des périphériques dans un instrument s'effectue au niveau des modules d'entrée et de sortie en désignant le périphérique à utiliser.

2.3. Les interfaces

Les *interfaces* permettent d'interconnecter les modules pour accéder à leurs données.
Les interfaces possèdent des propriétés de *liaison*, d'*accès* et de *type de donnée*.
Les *liaisons* entre interfaces sont de type *client / serveur*. Une interface cliente ne peut être reliée qu'à une seule interface serveur, mais une interface serveur peut servir plusieurs interfaces clientes. Une interface cliente permet l'accès aux données d'un autre module par l'interface serveur à laquelle elle est reliée. Les interfaces *relais* possèdent les 2 types client et serveur ; ces interfaces sont plutôt destinées à être utilisées avec des *modules composés* (voir la section module).
Les *accès* aux données peuvent se faire :

- en lecture
- en écriture
- en lecture / écriture

avec des *données typées* ou *non typées* (typée signifiant que le type de la donnée est précisé à la création de l'interface). Les interfaces *serveurs avec donnée* contiennent directement la donnée dans l'interface elle-même.
Les principaux types d'interfaces utilisés sont les suivants :

- *interface signal* : permet de véhiculer les signaux sonores d'un module à l'autre. Les signaux sonores sont représentés sous forme de *segments* de données de taille fixe contenant des valeurs numériques sur 16 bits correspondant à l'échantillonnage d'un signal sonore analogique à 44.1 KHz pendant une période de temps de 50 milli-secondes.
- *interface de paramétrage* : permet d'accéder aux données de fonctionnement propres au module. Ces données sont utilisées par le module au minimum pour la durée d'un segment de signal complet.
- *interface de modulation* : interface signal permettant de moduler (en fréquence, amplitude, durée, etc…) un signal par un autre. La valeur de chaque échantillon du signal modulant est utilisée pour moduler chaque échantillon correspondant du signal à moduler.

A titre d'exemple, le *modulateur d'amplitude* (voir plus loin) permet d'effectuer un double contrôle d'amplitude sur le signal à traiter : une *interface de paramétrage* permet d'initialiser une valeur d'amplitude de base *statique*, et une *interface de modulation* permet de moduler *dynamiquement* l'amplitude par un signal modulant.
Certains modules implémentent également des containers d'interfaces (listes ou tableaux) afin de regrouper des interfaces possédant des propriétés identiques ou de disposer d'interfaces en nombres variables.

2.4. Les modules

Les *modules* ont pour fonction de synthétiser et de traiter les signaux sonores.
Il existe 2 types de modules :

- Un *module simple* effectue un traitement du signal unitaire.
- Un *module composé* est un composant générique qui peut contenir d'autres modules (simples ou composés) et permet de constituer des groupes de modules.

Un module utilise généralement les signaux sonores d'autres modules auquel il est connecté par ses interfaces signal d'entrées *clientes* pour effectuer un traitement et produire un signal résultant sur une (ou plusieurs) interface signal de sortie *serveur*. Ce signal résultant peut être alors utilisé par les modules suivants dans la chaîne de calcul du signal des modules.
Certains modules fournissent également des interfaces de paramétrages afin de pouvoir modifier certains de leurs paramètres internes de fonctionnement (définition de la fréquence, de l'enveloppe, de la forme d'onde, etc.).

ALMIR implémente les différents modules simples suivants ; cette liste est non exhaustive et sera étendue au fur et à mesure du développement du logiciel. Chaque module est présenté avec son nom, une brève description, et les types d'interfaces qu'il implémente (une croix dans la case indique la présence de l'interface correspondante) :

NOM	*Description*			
Interfaces	*Signal entrée*	*Signal sortie*	*Paramètres*	*Signal modulant*
SmOutput	Sortie signal sur un périphérique			
	stéréo		périphérique à utiliser	
SmInput	Entrée signal depuis un périphérique			
		stéréo	périphérique à utiliser	
SmOutFile	Sortie signal dans un fichier			
	stéréo		nom du fichier	
SmInFile	Entrée signal depuis un fichier			
		stéréo	nom du fichier	
SmBalance	Balance d'un signal stéréo La balance est paramétrable par une valeur statique et modulable par un signal de modulation			
	stéréo	stéréo	valeur de balance	X
SmFreq	Modulateur de fréquence (échantillonnage d'un signal d'entrée) La fréquence est paramétrable par une valeur statique et modulable par un signal de modulation auquel on peut appliquer un multiplicateur de modulation. En principe, l'entrée signal provient d'un module table d'onde SmWave, mais il est néanmoins possible d'utiliser n'importe quel signal.			
	X	X	valeur de fréquence	X
SmAmp	Modulateur d'amplitude L'amplitude est paramétrable par une valeur statique et modulable par un signal de modulation			
	X	X	valeur d'amplitude	X
SmWave	Générateur de table d'onde			
		X	données pour initialiser la table d'onde (sinus, carré, triangle, table d'harmoniques, table d'onde directe)	
SmEnv	Générateur d'enveloppe L'enveloppe est modulable en durée par un signal de modulation auquel on peut appliquer un multiplicateur de modulation			
		X	données de l'enveloppe	X
SmSwitch	Commutateur (sélection d'un signal d'entrée parmis plusieurs) Le container de signaux d'entrées est dimensionnable			
	container	X	signal à commuter	
SmMixer	Mélangeur (mixage de plusieurs signaux d'entrées) Le container de signaux d'entrées est dimensionnable			
	container	X		
SmMatrix	Matrice à dimension variable permettant d'envoyer X lignes d'entrées sur Y colonnes de sorties			
	container	container	liaisons matricielles	

2.5. Les containers

Les containers sont des composants génériques qui permettent de grouper des interfaces ou des modules possédant les mêmes propriétés. Un container peut être utilisé partout où l'un de ces composants unitaires peut l'être.

Il existe 2 types de containers :

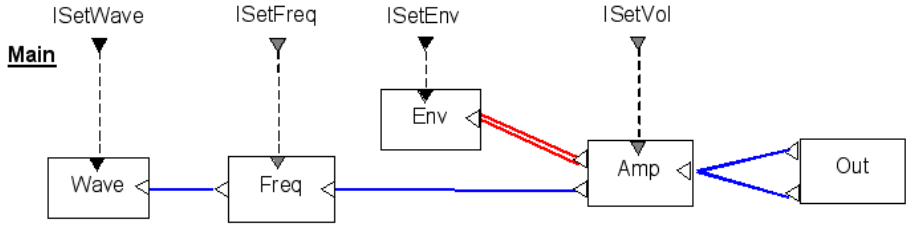
- Les listes : permettent l'accès aux éléments par un nom.
- Les tableaux : permettent l'accès aux éléments par un indice numérique.

Les containers permettent également d'implémenter des composants à dimension variable qui peut n'être définie que lors de leur instanciation.

2.6. Les instruments

2.6.1. Description

Un instrument est un ensemble de modules interconnectés. Il est constitué d'un *module principal* (module de type *composé* spécifique) qui peut contenir d'autres modules. Il est possible d'utiliser plusieurs instruments simultanément.
Un instrument peut être créé par instructions logicielles en définissant un à un tous ses composants ou par un fichier d'instrument qui contient la description de ses composants et de leurs interconnexions avec une syntaxe spécifique (voir plus loin).
La figure suivante présente un exemple d'instrument contenant un module principal et 5 sous-modules (une table d'onde, un modulateur de fréquence, un générateur d'enveloppes, un modulateur d'amplitude et un module de sortie) et leurs interfaces.



Le module principal Main possède des interfaces clientes qui permettent de définir les paramètres de fonctionnement de ses sous-modules ; chacune de ces interfaces est reliée à une interface serveur du sous-module correspondant (traits en pointillés) :

ISetWave	forme d'onde du module table d'onde Wave
ISetFreq	fréquence du modulateur de fréquence Freq
ISetEnv	enveloppe du générateur d'enveloppes Env
ISetVol	volume statique du modulateur d'amplitude Amp

Le signal audio (en trait simple) circule de gauche à droite en passant successivement par les modules Wave, Freq, Amp et Out. Le module Env génère un signal de modulation enveloppe (en trait double) qui permet de moduler le signal audio en amplitude au niveau du module Amp.
Les sous-modules effectuent les traitements du signal suivants :

Wave	Génère une forme d'onde périodique paramétrable
Freq	Echantillonne cette forme d'onde suivant une fréquence paramétrable
Env	Génère un signal de modulation sous forme d'une enveloppe constituée d'une suite de segments
Amp	Module en amplitude le signal provenant du module Freq en utilisant le signal de modulation émis par Env ainsi que la valeur d'amplitude statique paramétrable par l'interface ISetVol
Out	Emet le signal vers le périphérique de sortie

2.6.2. Les fichiers d'instruments

Les fichiers d'instruments servent à décrire des instruments de façon textuelle au moyen d'une syntaxe simple.
Le chargement d'un fichier d'instrument est effectué par le *chargeur de fichiers* en 2 temps :

- Lecture et analyse du fichier.
- Création de l'instrument décrit.

Les fichiers d'instruments permettent de :

- déclarer des classes de modules simples ou composés avec leurs interfaces.
- instancier ces classes en spécifiant leur état initial (actif / inactif) ainsi que leur cardinalité (objets simples ou containers dimensionnables).
- dimensionner des containers d'interfaces à taille variable.
- relier les interfaces afin de connecter les modules.
- pré-initialiser des interfaces de paramétrages.
- inclure d'autres fichiers d'instruments.

2.7. La synthèse sonore

Afin d'optimiser les temps de traitements du signal, ALMIR effectue la synthèse sonore en utilisant une *chaîne de calcul du signal*. Cette chaîne de calcul contient la liste des modules de l'instrument ; elle est calculée avant chaque exécution en suivant récursivement les liaisons d'interfaces signal de tous les modules à partir des modules de sortie. Elle est également hiérarchisée de sorte que chaque module composé possède sa propre chaîne partielle qui ne comporte que ses sous-modules. Cette chaîne est bien entendu recalculée à chaque modification de l'instrument (ajout de modules, de liaisons, etc.) pendant la phase d'exécution.
ALMIR utilise ensuite cette chaîne pour demander successivement à chaque module d'effectuer son traitement du signal. Cette opération est effectuée toutes les 50 milli-secondes et permet de calculer des *segments de sons* (ces segments sont constitués des échantillons de signal pour une durée de 50 milli-secondes avec un échantillonnage à 44.1 KHz, soit 2205 échantillons de 16 bits permettant des variations d'amplitude comprises entre –32767 et +32767).
Différents processus sont exécutés en parallèle de façon asynchrone pendant la phase de synthèse sonore :

- Le *gestionnaire de son* utilise une horloge afin de cadencer les calculs de segments de signal ; à chaque interruption d'horloge, il demande à chaque instrument d'effectuer les calculs de traitements du signal pour le segment de son suivant. Chaque instrument fait alors suivre la demande à son module principal, qui fait de même pour tous ses sous-modules, et ainsi de suite jusqu'à la fin de la chaîne de calcul. Les gestionnaires de périphériques de sorties sont ensuite sollicités pour mettre en pile leur segment de son calculé.
- Chaque gestionnaire de périphérique possède un système de cadencement propre par notifications venant de DirectSound qui lui permet de recevoir ou d'émettre chaque segment de son empilé en temps voulu.

Notons que les traitements du signal au niveau des modules sont optimisés, particulièrement dans les boucles de signal où l'on évite le plus possible l'utilisation de tests ou d'appels de sous-fonctions.
A titre indicatif, ALMIR permet sur un PC muni d'un Pentium III à 1 GHz avec 256 Mo de mémoire vive de gérer environ 80 canaux contenant chacun :

- une table d'onde
- un modulateur de fréquence
- deux modulateurs d'amplitudes (1 pour le signal audio et 1 pour le signal de modulation)
- deux générateurs d'enveloppes (idem)
- une balance (pour la panoramisation du signal audio)

+ 2 mélangeurs et une sortie ligne pour le signal audio stéréo.

2.8. Extensibilité

ALMIR permet d'ajouter aisément de nouvelles classes de modules ou d'interfaces à partir des classes de bases existantes. Il est ainsi possible d'incorporer de nouveaux modules de traitements à ceux déjà existants sans besoin de recompiler le logiciel.
Un mécanisme d'enregistrement préalable des ces classes auprès d'ALMIR permet aisément leur utilisation ultérieure dans des fichiers d'instruments.
L'utilisateur peut ainsi définir des bibliothèques de classes de modules et d'interfaces en créant des bibliothèques (.LIB) qui contiennent le code binaire de ces classes et en fournissant un fichier d'instrument à inclure qui contient leur description.

3. Conclusion

Comme nous l'avons souligné au début de ce document, notre objectif de départ était de créer des systèmes instrumentaux génériques, modulaires et temps-réel en n'utilisant que du matériel et système standard existant. Quand nous avons commencé le développement du logiciel, la vitesse des processeurs ne permettait pas encore d'effectuer des traitements du signal intensifs. Nous avons fait le pari que l'accroissement rapide en puissance des ordinateurs rendrait bientôt possibles de tels traitements.
Aujourd'hui, la vitesse des micro-processeurs avoisine les 1,5 GHz, cependant que la taille des ordinateurs diminue de façon proportionnelle (un ordinateur portable possède maintenant la même puissance qu'un modèle de bureau). Cette maniabilité du matériel est idéale pour les situations de concerts où l'interactivité peut-être un critère important.
D'autre part, l'accroissement en puissance des processeurs, qui ne pourra aller qu'en augmentant, représente le meilleur gage de pérennité pour ALMIR qui n'effectue que des traitements logiciels et voit donc sa puissance augmenter constamment de façon proportionnelle.
Les perspectives de développement pour ALMIR sont encore nombreuses. Nous apportons des améliorations constantes en tenant compte des observations des utilisateurs.

Vincent Lesbros (LESBROS 1995), créateur du logiciel Phonogramme, utilise d'ailleurs ALMIR pour effectuer la synthèse sonore de ses images avec sa nouvelle version pour Windows.
Nous avons également en projet avec lui la création d'un module graphique permettant de définir et de tester plus facilement les instruments.

4. Bibliographie

BATTIER M. " Eléments d'informatique musicale ". Support de cours, université Paris VIII.
CHADABE J. " Interactive Composing: An Overview " in *Computer Music Journal*, 1984, vol 8 n° 1, pp 22-27.
CHOWNING J., D. BRISTOW. "FM Theory and Applications - By Musicians for Musicians". Yamaha, Tokyo, 1986, 195 p, ISBN 4-636-17482-8 COO73.
CHOWNING J. "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation" in *Foundations of Computer Music*, dir.de pub.ROADS C.& J. STRAWN., 1985, pp 6-29, extrait de *Journal of the Audio Engineering Society*, 1973, ISBN 0-262-181142
ENGLERT G.G. "Musique: composition automatique, automation composée" in *Informatique et Sciences Humaines*, n° 45, 1980, pp 37-50.
ENGLERT G.G. "Our Score: A Description of Metro 3, A Compositional and Performance Software Program" in *Leonardo Music Journal*, vol 3, 1993, pp 53-58.
GAIV. "Le système portable de synthèse hybride de Vincennes" in *Informatique Musicale*, CNRS, Textes des conférences, Equipe ERATTO (ER 152), 1977, Collection Calcul et Sciences Humaines, pp 193-204.
LEIPP E. Acoustique et musique. Masson, Paris, 1980, 352 p, ISBN 2-225-64702-X.
LESBROS V. "Atelier Informatique pour la Musique Expérimentale" in *Rapport du Laforia 95/13*, 1995, Deuxième journées d'informatique musicale, pp 67-80.
LESBROS V. Atelier Incrémentiel pour la Musique Expérimentale. Thèse, 1995, Université Paris 8.
LESBROS V. "From images to Sounds - a Dual Representation", MIT Press, Computer Music Journal, 20:3, pp. 59-69, Fall 1996.
LESBROS V., G.G. ENGLERT. Metro 3 v.2: Programme compositionnel pour Macintosh et dispositifs MIDI. Rapport du Groupe Art et Informatique de Vincennes, 1991, Université Paris VIII, Département Informatique.
LINDEMAN E., M. STARKIER, F. DECHELLE. "The IRCAM Musical Workstation: Hardware Overview and Signal Processing Features" in *Proceedings of the 1990 International Computer Music Conference*, Glasgow, 1990, pp 132-135.
LIPPE C., M. PUCKETTE. "Musical Performance Using the IRCAM Workstation" in *Proceedings of the 1991 International Computer Music Conference*, San Francisco, 1991, pp 533-536.
LIPPE C., M. PUCKETTE [et al.]. "The IRCAM Signal Processing Workstation and IRCAM Max user Groups: Future Developments and Platforms" in *Proceedings of the 1993 International Computer Music Conference*, 1993, pp 446-448.
MARTINEZ L. Un Atelier d'Informatique Musicale Interactive Temps Réel. mémoire de maîtrise d'informatique, 1996, Université Paris 8.
MATHEWS M. The Technology of Computer Music. MIT Press, Cambridge, 1969, 188 p.
MATHEWS M., G. BENNETT. Real-Time Synthesizer Control. Rapports IRCAM.
MOORE F.R.. "Table Lookup Noise for Sinusoidal Digital Oscillators" in *Foundations of Computer Music*, dir.de pub.ROADS C.& J. STRAWN., 1985, pp 326-334, extrait de *Computer Music Journal* 1(2) 1977, ISBN 0-262-181142
NEDCO. The ABLE Family of Mini-Computers from NEDCO. Brochure de présentation technique du constructeur, 1978.
POPE S.T. "Real Time Performance via User Interfaces to Musical Structures" in *Interface, Journal of New Music Research*, vol 22 n° 3, 1993, pp 195.
PRESSING J. Synthesizer Performance and Real Time Techniques. Computer Music and Digital Audio Series, Oxford University Press, 1992.
RISSET J.C. Composing in Real Time ?. Rapport du Laboratoire de Mécanique et d'Acoustique, CNRS, Marseille.
RISSET J.C. "Musique et Informatique" in *Musique et ordinateur*, Editions du Centre Experimental du Spectacle, 1983, pp 51-65, ISBN 2-904341-01-3.
VERCOE B. "Real-Time CSOUND: Software Synthesis with Sensing and Control" in *Proceedings of the 1990 International Computer Music Conference*, Glasgow, 1990, pp 209-211.
VIARA E., M. PUCKETTE. "A Real-time Operating System for Computer Music" in *Proceedings of the 1990 International Computer Music Conference*, Glasgow, 1990, pp 270-272.