IanniX

Thierry Coduys, Adrien Lefèvre, Gérard Pape

▶ **To cite this version:**

Thierry Coduys, Adrien Lefèvre, Gérard Pape. IanniX. Journées d'Informatique Musicale, Jun 2003, Montbéliard, France. hal-02994165

HAL Id: hal-02994165
https://hal.science/hal-02994165

Submitted on 7 Nov 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

IanniX

Thierry Coduys : Thierry.Coduys@la-kitchen.fr
Adrien Lefèvre : support@adlef.com
Gérard Pape : gerard.pape-ccmix@wanadoo.fr

La kitchen – Paris
CCMIX - Alfortville
Ministère de la Culture et de la Communication - DMDTS

Résumé

IanniX a le potentiel pour être un descendant intéressant de l'UPIC. Fidèle à ses origines, il traverse néanmoins ses propres chemins. Sans se confondre avec l'UPIC, il propose des nouvelles pistes innovantes :
- Une partition graphique multidimensionnelle et multi-formelle.
- Au niveau du contrôle du déroulement du temps et de la représentation graphique de la macro-forme, un séquenceur à la géométrie poly-topologique et poly-temporel est proposé.
- Au niveau micro-formel, des techniques de synthèse créées par IanniX sous forme de « plug'ins » pouvant être représentées et éditées graphiquement dans les deux cas.
- Grâce aux capteurs de mouvement, les gestes servant d'éléments de micro et macro-forme sont saisis et représentés graphiquement d'une façon dynamique et multidimensionnelle.

Introduction : De l'UPIC à IanniX

Lorsque dans les années 1970 Iannis Xenakis entreprenait la fabrication de l'UPIC (Unité Polyagogique Informatique du CEMAMu), il réunissait dans un même cadre plusieurs outils en un seul :
• Au sein du temps plan/fréquence, l'UPIC offre une extension de la notation du solfège traditionnelle.
• Dans un même espace, on pouvait enfin décrire simultanément une partition (vue comme étant une succession de sons) et le son lui-même dans sa texture intime : le timbre et son évolution au cours du temps.
• De même que pour le geste, le musicien interprète et/ou compositeur avait enfin la possibilité de s'exprimer de manière simple et directe, tout en gardant une trace de son action grâce au formidable enregistreur qu'est l'ordinateur.
Les développements successifs qu'a inspiré l'UPIC (Metasynth, Hyperupic, etc.) n'ont pas intégré cette donnée. Les concepteurs ont assigné des paramètres sonores globaux à l'image sans travailler en détail la microstructure du son.
Dès lors, nous proposons l'élaboration d'un nouveau logiciel intégrant parfaitement la théorie initiale de l'UPIC tout en élargissant ses possibilités de manière significative.
Nous cherchons toujours une représentation qui convienne à la partition électroacoustique, qui soit une symbolisation de la structure formelle de la composition et qui permette d'agir directement, grâce à la souplesse des graphismes, sur la synthèse du son à tous les niveaux (micro ou macro).
Aussi, avons-nous conçu ce nouveau prototype logiciel IanniX, témoin d'une évolution à la fois technique et artistique.

Une volonté d'innovation

Le développement de IanniX se base sur une série de concepts innovants forts, dont l'ensemble peut être considéré comme le guide de notre recherche.

Une partition graphique multidimensionnelle et multi-formelle

Le niveau de la page dans l'UPIC va devenir le niveau micro de la partition de *IanniX*, c'est-à-dire la représentation d'un seul « son ». Quand on dessine une page dans l'UPIC, on dessine une succession de sons. Nous pouvons imaginer dans *IanniX* un niveau supérieur où le niveau micro est caché et où nous représentons des séquences de « sons » déjà riches et complexes grâce à leur synthèse timbrale, et cela au niveau micro. Nous pouvons donc distinguer la représentation graphique d'une synthèse de son individuel au niveau micro-formel, et celle d'une séquence de sons complexes au niveau macro-formel.

Un séquenceur multidimensionnel et poly-temporel

L'UPIC propose au niveau macro-formel un univers en deux dimensions dans lequel la partition graphique est jouée à vitesse unique, même si le tempo est variable.
Nous proposons une solution ouverte sur les multi-topologies de l'espace et les multi-tempi.
Dans IanniX, les sons complexes apparaissant au niveau macro de la partition graphique, peuvent être comparés à des planètes, chacune évoluant dans son propre système, immobile ou en mouvement. Dès lors, imaginons plusieurs pages UPIC se jouant simultanément, chacune dans sa propre géométrie et connectée à différents types de synthétiseurs. Enfin, IanniX par le tracé des trajectoires de temps, suggère la représentation du mouvement d'un son dans l'espace.

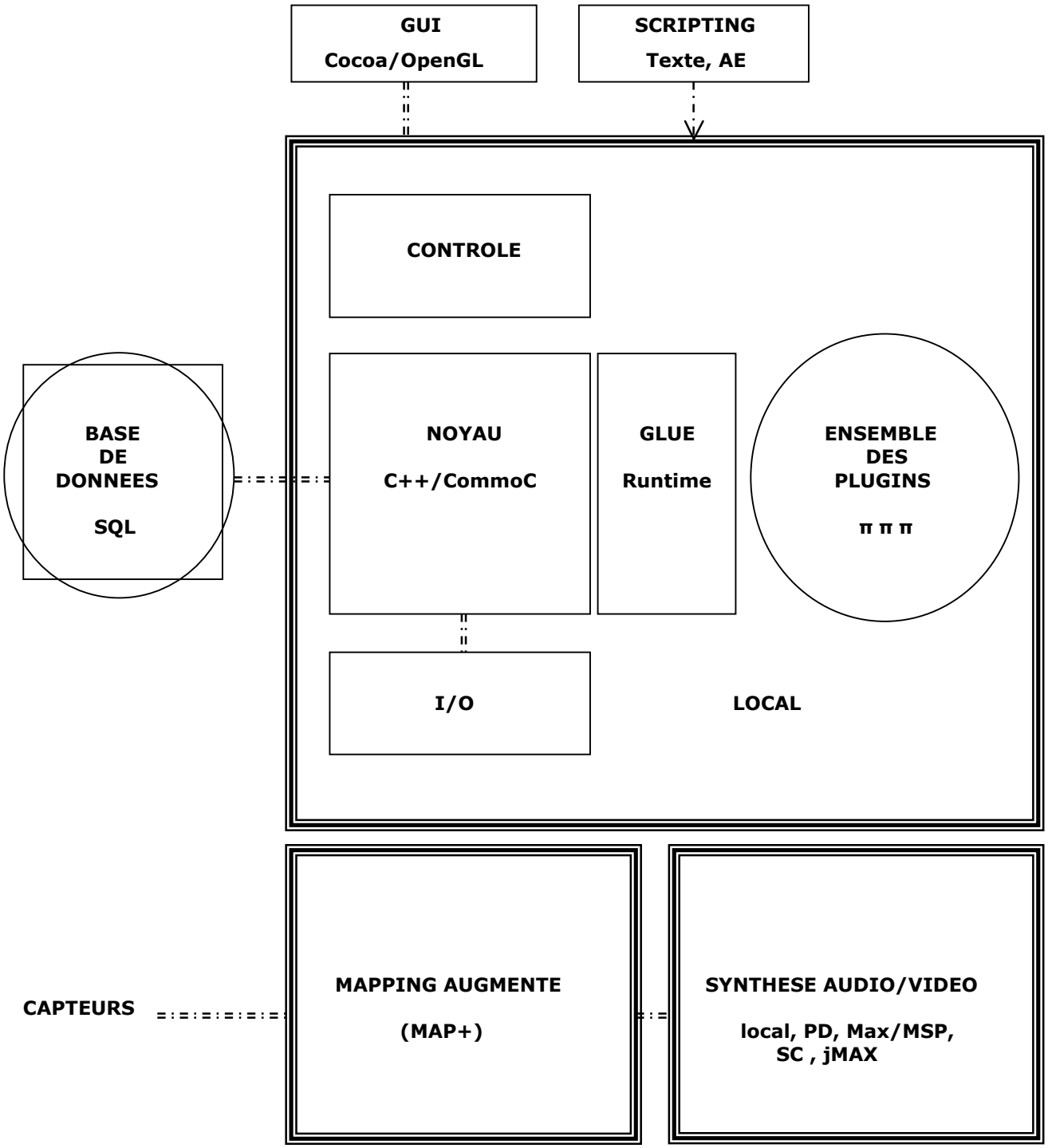
Une saisie des données de la composition dynamique et multidimensionnelle

Par la méthode du dessin sur la tablette d'architecte, Xenakis propose un lien direct entre l'imaginaire du compositeur, son geste et la représentation graphique.
La kitchen développe des capteurs de mouvement sophistiqués permettant de traduire le geste kinétique directement dans un graphique : au niveau micro, le capteur peut saisir un geste qui représente une courbe mélodique ou rythmique; au niveau macro, le geste peut représenter les *accelerandi* ou *rallentandi* des *tempi*.

Les techniques de synthèse audio et visuelles

Nous conservons le dessin des trajectoires de la synthèse additive et de la modulation de fréquence tel que le propose l'UPIC, grâce à un moteur de synthèse additif intégré dans IanniX.
Concernant les autres méthodes de synthèse et de traitement du signal : certaines pourront être incorporées comme fonction dans IanniX via une architecture « plug'in ». Ou encore via un réseau UDP avec le protocole **OSC** vers d'autre machine compatible, typiquement une architecture Client/Serveur.

Architecture logicielle



Le schéma structurel est construit selon une architecture classique du type client/serveur et permet une communication réseaux via TCP/IP, de manière directe.
La partie **Local** contient des éléments logiciels de base et constitue le programme. Eléments qui assurent les fonctions suivantes :

Le **Noyau** est le cœur du logiciel, il est écrit en C++ et de préférence totalement portable. Il assure le schedulling et la liaison entre les autres modules. Il contient un ensemble de plugins par défaut, qui forment la base de l'API Plug'in. **CommoC** est une première tentative d'implémentation du Noyau, il est notamment conçu pour le scripting.

La **Glue** permet de lier l'ensemble des plug'ins au Noyau. L'extension des capacités du Noyau se fait par spécialisation de classe (ie: héritage d'une classe de base, puis (ré)écriture de certaines méthodes). La Glue est ainsi prise en charge par le **Runtime** du logiciel.

Le **Contrôle** est l'organe de commande du Noyau. Cette partie met en relation le Noyau avec la **GUI** (répartition des évènements, souris, clavier, visualisation à l'écran, etc...), le **Scripting** sous forme de Textes de Commandes ou de AppleEvents (cas MacOSX). La GUI fait des appels à OpenGL et Cocoa (cas MacOSX) ou X-Windows (cas LINUX).

La **partie I/O** communique avec le **Mapping Augmenté** (Map+) en entrée (capteurs) comme en sortie (déclenchement d'événements). A son tour le Map+ communique avec la partie **Synthèse Audio/Vidéo**, soit en laissant passer l'information telle quelle, soit en y appliquant une transformation.

La **Base de Données** permet la persistance des objets traités. C'est le Noyau qui fait les requêtes par l'intermédiaire d'un SGBD (Système de Gestion de Base de Données - non noté sur le schéma).

Eu égard à l'architecture logicielle retenue, nous avons souhaité des choix techniques adaptés. Aussi avons-nous retenu les options suivantes :

Le système UNIX.
Avantages : le système par excellence pour le partage des tâches, la communication réseaux, la gestion mémoire, l'implémentation d'une base de données, etc. A priori le choix se porte sur **LINUX** et/ou **Mac OSX** (convivialité de l'interface utilisateur).

La librairie graphique OpenGL.
Avantages : multi-plateforme, optimisée (notamment par carte graphique), très répandue, permettant la 3D, et bien sûr présente sur UNIX.

Un système de base données.
Avantages : garantie de pérennité et d'extension des formats de données, partage réseaux des ressources, portabilité et masse de stockage virtuellement illimitée. Nous pensons utiliser une base de données très utilisée, comme SQL

Un langage objet
Nous retiendrons C++, au moins pour conduire les premiers tests.
Avantages : souplesse d'utilisation (objet), rapidité (compilé/natif), grande popularité parmi communauté des développeurs, omniprésence sur plate-forme.
A noter : choix non définitif.

Une architecture à plug'ins
Il existe un ensemble de plug'ins par défaut, réunis au cœur du noyau. Le plug'in définit une ou plusieurs classes, héritant de classes génériques appartenant au noyau : extension des capacités du noyau par spécialisation de classe.
Ouverture de l'environnement maximale quand n'importe quelle classe du noyau (à quelques exceptions près) peut être redéfinie toute ou partie par héritage.
Définitions souhaitées dans les plug'ins : visualisation/édition graphique, stockage et traitement de données diverses, importation/exportation de données aux formats variés, fonctions de calcul, de scheduling, de relations entre objets, etc.

Une interface utilisateur standard

Avantages recherchés : permettre à tous une utilisation locale du logiciel, rapidité et simplicité d'installation, possibilité d'ouvrir à tout moment un fichier partition soit dans la base de données (locale ou non) soit par un fichier indépendant.

(**π**) signifie « *implémentation en plug'in* ».

Il s'agit d'un logiciel séquenceur. Dès lors, nous devons pouvoir éditer/visualiser une collection d'objets représentés dans une section du plan et imaginer de quoi sera composé un **objet « événement »** (**π**) de notre logiciel :

- Une coordonnée spatiale (x, y, z ...),
- Une région du plan ou de l'espace, etc.

Cet objet « événement » encapsule d'autres objets (agrégation d'objets) :

- Un objet de représentation graphique, (**π**)
- Une collection d'objets de donnée, (**π**)
- Une collection d'objets I/O (vers Map+), etc. (**π**)

Sur ces objets, on applique les opérations courantes d'un logiciel d'édition :

- Un panneau « Inspecteur », pour l'édition des caractéristiques d'un ou plusieurs objets,
- Sélection / Multi-sélection,
- Couper / Copier / Coller,etc.

Les **objets de représentation graphique** sont très variés, peuvent être représentatifs des objets de données ou non, et font appel aux fonctions graphiques d'OpenGL.

Les **objets de données** gèrent le contenu des objets événement : fonction de lire/écrire/traiter des données diverses telles que des fonctions réelles, des spectrogrammes du signal, des événements ponctuels comme une note MIDI.
Ils contiennent tous une fonction de base, qui à une valeur de temps associe en réponse un « événement » sous un format à déterminer. Cet évènement est envoyé dans un flux via les objets I/O.

Les **objets I/O** assurent la communication avec le Map+ : écriture dans un flux de données (notamment en réseaux ; voir avec le Real Time Protocol RTP).

Tous ces objets sont (re)définissables en implémentant de nouvelles classes dans les plugins.

Une nouvelle géométrie

Objet : tentative de généralisation de la représentation d'une séquence ou d'un son dans le plan.

Remarque : des représentations classiques nous ne retiendrons que deux concepts, *le plan orienté* et la *représentation de l'évènement* dans ce plan. Un arc dans l'UPIC est considéré dans ce modèle comme un évènement (du son).
Dans ce modèle, le **« plan »** au sens large du terme correspond à un grand nombre d'espaces de type différent. En l'espèce, les entités *vecteur, droite, courbe, sphère*, et les opérations *produit scalaire, norme d'un vecteur, projection orthogonale,* sont définies par la géométrie pour de nombreux espaces de dimensions et de topologies différentes. Dès lors, ce modèle est valable pour tous les espaces énoncés ci-dessus et bien d'autres encore. Nous emploierons indifféremment les termes *espace* et *sphère* pour *plan* et *cercle*.

La partition et la trajectoire

Dans ce modèle, l'événement est défini uniquement par sa durée et da date dans la séquence. Il trouve sa représentation dans l'espace, représentation dont nous ne retiendrons ici qu'une abstraction. : **une sphère dont le centre désigne la date dans la séquence, et dont le**

diamètre est proportionnel à la durée. Une collection de sphères dans l'espace définit une partition.
La **trajectoire** fixe l'orientation de l'espace, elle désignera par la suite la ligne du temps. Ce peut être une droite orientée quelconque, ou une courbe dont la fonction est dérivable. Dans tous les cas la trajectoire est orientée, et **c'est la combinaison de la partition et de la trajectoire qui forme la séquence.**
Nous verrons que l'on peut définir simultanément plusieurs trajectoires dans un même espace, donnant plusieurs séquences pour une même partition : notion de partition polymorphe.

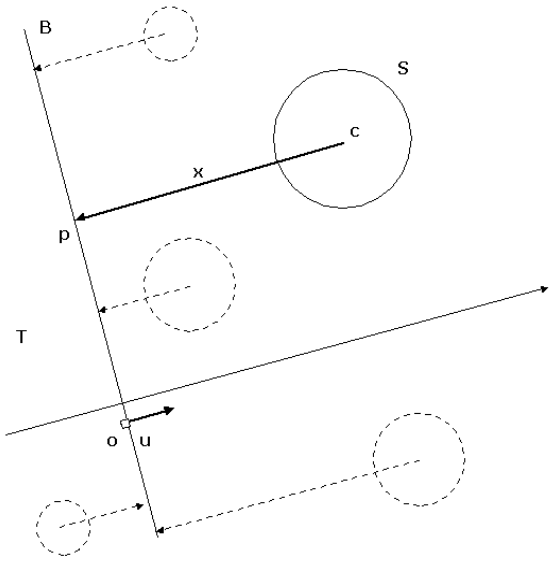
Le curseur

En tout point de la trajectoire, on définit le vecteur unitaire tangent et la droite perpendiculaire à la tangente désignée curseur. À partir de ce vecteur unitaire, on définit un repère cartésien et orthonormé en tout point et se déplaçant sur la trajectoire, dans lequel chaque sphère trouve sa coordonnée locale. À partir de cette coordonnée locale, on déduit la valeur du temps pour l'événement considéré.
En déplaçant le curseur sur la trajectoire, nous créons la séquence, nous en faisons la lecture.
Le mouvement du curseur s'effectue à **vitesse v** (v peut-être positive, négative ou nulle).
C'est **v** (rapport d'une distance à une durée), qui cristallise la transformation de l'espace en temps, de la partition en séquence.

Si le curseur est une droite ou un segment de droite dans le plan, alors il est un plan ou une section de plan dans l'espace tridimensionnel.
Cette droite (ou ce plan) se déplace par définition le long de la trajectoire (qui elle est unidimensionnelle par définition) et perpendiculairement à celle-ci en tout point.
La vitesse du curseur est une fonction de la trajectoire, plus précisément sa dérivée. Par exemple ci-dessous, la fonction paramétrique dans le plan d'une trajectoire rectiligne et uniformément accélérée :

T(t):	x(t) = a.t2	**v(t):**	x'(t) = 2.a.t
	y(t) = b.t2		y'(t) = 2.b.t

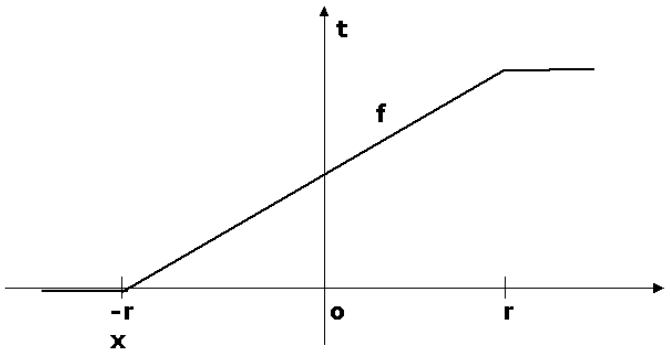
Avec **a** et **b** constants, **t** le temps absolu, **T(t)** la trajectoire et **v(t)** sa dérivée, représentant la vitesse (croissante) du curseur au temps **t**.



Le plan orienté par la trajectoire **T** admet le vecteur unitaire **u** au point **o**.
La droite/curseur **B** est perpendiculaire à **T** en **o**.
La sphère **S** a pour centre **c**, et le point **p** est la projection orthogonale de **c** sur le curseur **B**.
C'est le produit scalaire **(u|cp)** du vecteur **u** par le vecteur **cp** qui nous intéresse, puisqu'il donne la distance orientée du point **c** au curseur **B**. Cette distance orientée sera notée **x** dans la suite de l'exposé.

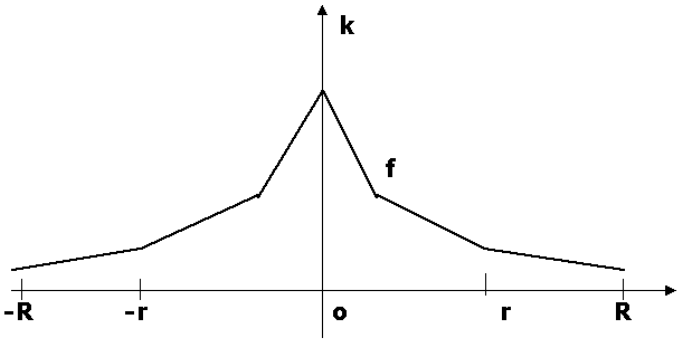
Une fonction d'approche

Un évènement commence lorsque le curseur entre dans la sphère représentant cet évènement, et il se termine quand le curseur en sort. Durant toute cette progression, la quantité **x** est passée continûment de la valeur **-r** à la valeur **+r**, où **r** est le rayon de la sphère. Pour transformer **x** en une valeur de temps **t** locale à l'évènement et utilisable, nous employons la fonction toute simple ci-dessous, nommée **fonction d'approche f** de l'évènement :
Où **t = f(x) = q.(x+r)** sur l'intervalle **[-r,r]**, avec **q** constant et défini comme le rapport de la durée de l'évènement au diamètre de la sphère ; c'est l'inverse de la vitesse.



Dualité événement temporel/évènement spatial

Le terme fonction d'approche exprime la relation qui existe entre ce modèle et l'interpolateur par boule dans Syter. En effet, en réglant la fonction d'approche d'une certaine manière, on retrouve le coefficient **k** de présence spatiale de l'interpolateur. La fonction d'approche **f** dois simplement être paire et définie sur **IR**, par exemple :



Pour des raisons pratiques d'implémentation, nous ne pouvons pas définir une fonction d'approche **f** sur tout **IR**. Nous emploierons donc un nouvel intervalle nommé intervalle ou rayon d'action de l'évènement. Ce rayon est noté **R**, il peut être très grand et il est toujours supérieur à **r**. Dans la partie logicielle, ce **rayon R** définit une zone d'activation/désactivation de l'objet évènement, selon que le curseur se trouve dans sa sphère d'action ou non.

Nous pouvons donc définir la fonction d'approche **f** d'un évènement associée à une sphère de rayon **r** comme suit :
- C'est une fonction réelle d'une variable réelle.
- Elle est définie sur l'intervalle **[-R,R]**, ou **R ≥ r** est le rayon d'action de l'évènement.
- Elle est de la forme **f(x) = q.(x+r)** pour un ***évènement temporel*** (avec **R = r**).
- Elle est paire et positive dans le cas d'un ***évènement spatial***.

Synthèse

Nous avons unifié l'évènement temporel et l'évènement spatial, mais seulement d'un point de vue algorithmique. Car la ressemblance s'arrête là où nous donnons un sens à ces deux types d'évènements : dans le cas de l'évènement temporel la sphère représente une *date* et une *durée* dans la partition, alors qu'elle représente un *pôle* et un *poids* dans le cas de l'évènement spatial. En outre le curseur peut se résumer à un point, une droite ou un plan quelconque pour interagir avec les évènements spatiaux.
Les contraintes sur le curseur ne sont donc pas les mêmes lorsqu'il s'agit d'évènements temporels ou d'évènements spatiaux. La combinaison de plusieurs curseurs et d'évènements des deux types peut s'avérer très riche. *D'autres fonctions d'approche sont-elles envisageables, pourraient-elles correspondre à une demande non encore élucidée?*
Voici dans le tableau ci-dessous résumé la dualité évènement temporel/spatial :

Evènement	Représentation - Sphère **S**			Curseur **B**
	Centre **c**	Rayon **r**	Action **R**	
Temporel	Date	Durée	**R = r**	Droite sur Trajectoire
Spatial	Pôle	Poids	**R >> r**	Point ou Droit

Conclusion : les objectifs de développement

Notre premier objectif était de réaliser un nouvel environnement logiciel (IanniX). La suite logicielle est multi plateforme, « Open source » et gratuite. Elle est dotée :

- D'un éditeur de courbe, pour la représentation graphique (IanniX)
- D'un système corrélatif intelligent (mapping augmenté)
- D'un moteur de synthèse orienté multi-environnement (pd, Max/MSP, JMax, SC, etc.)

Concernant plus particulièrement le système corrélatif intelligent : le mapping augmenté est le centre nerveux pour la reconnaissance du geste et sa corrélation avec la représentation graphique en général.

Dans l'état actuel du projet, il n'est qu'en phase de pré-étude, mais nous pouvons dès à présent donner un aperçu des domaines à explorer pour sa réalisation : approche pour l'analyse de la décision, domaine de la théorie de l'action, domaine de la psychologie cognitive et de la sémiologie, traitement et analyse des courbes, reconnaissance de forme historique, apprentissage automatique et propositions de corrélation, etc.

Références :

1) http://www.ccmix.com/indexfr.shtml
2) http://www.iannis-xenakis.org/
3) http://homestudio.thing.net/revue/content/asr1p20.html
4) http://www.info.unicaen.fr/bnum/jelec/Solaris/d07/7caillaud.html
5) http://www.la-kitchen.fr/iannix.html
6) Corlaix Omer, Gallet Bastien, « Entretien avec Iannis Xenakis », *Musica Falsa* n°2, Paris, 1998, p. 29.