IANNIX 0.8

Guillaume Jacquemin
Association IanniX
guillaume@iannix.org

Thierry Coduys
Association IanniX
thierry@iannix.org

Matthieu Ranc
Association IanniX
matthieu@iannix.org

RESUME

IanniX est un séquenceur graphique open source, inspiré des travaux de Iannis Xenakis et destiné à la création numérique [7]. Le logiciel propose une écriture polytemporelle du temps d'événements statiques et dynamiques vers un environnement dédié (Processing, PureData, SuperCollider…).

À l'aide d'une palette d'objets fondamentaux que sont les *triggers (événements)*, les courbes *(trajectoires dans l'espace)* et les curseurs *(progression dans le temps)*, IanniX permet une représentation graphique et interactive du temps dans l'espace 3D et assure un échange bidirectionnel via plusieurs protocoles de communication, dont l'OSC[1].

Depuis mai 2011, l'architecture générale de IanniX a été réétudiée et de nouvelles fonctionnalités innovantes enrichissent ses possibilités : scripts génératifs, écriture récursive, live coding, nouvelles interfaces réseau, gestion intégrale de la 3D. Aussi, pour assurer un financement potentiel (subventions, dons) et garantir les développements futurs, l'association loi 1901 « IanniX » a été créée[2].

Nous présentons ici un état des lieux de IanniX, de ses intentions initiales aux perspectives ouvertes par les nouveaux développements.

1. INTENTIONS

1.1. De l'UPIC à IanniX

Dans les années 1970, Iannis Xenakis entreprenait la réalisation de l'UPIC[3]. Cet outil, extension de la notation du solfège traditionnelle, permettait au musicien de composer dans un espace temps / fréquence et de l'interpréter de manière simple et directe, en conservant une notion de geste instrumental.

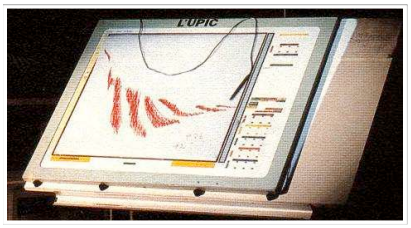

Figure 1. Partition dessinée sur l'UPIC

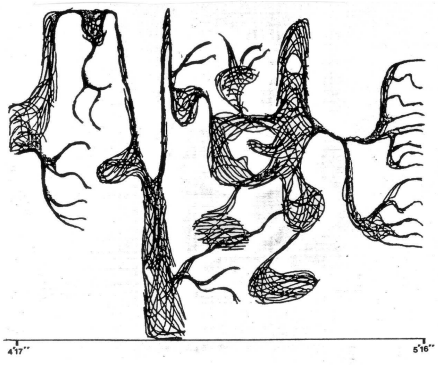

Figure 2. Partition graphique de Iannis Xenakis, Mycenæ Alpha, composée sur UPIC

Les développements successifs qu'a inspiré l'UPIC *(Metasynth, HighC…)* ont reproduit un outil de dessin spectral, avec une synthèse de son intégrée — donc imposée —, sans introduire la notion de geste instrumental ni de temps réel.

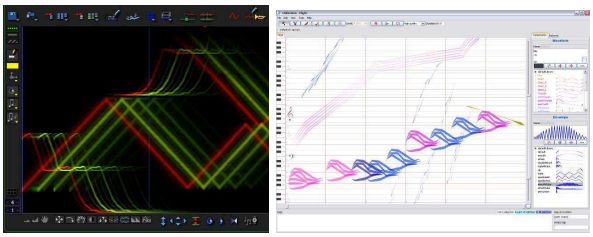

Figure 3. MetaSynth et High-C

1.2. Séquenceurs Open Sound Control

L'intégration progressive de l'Open Sound Control [6] dans les outils de création numérique a ouvert le champ à la création de nombreux séquenceurs OSC. Cependant, ces séquenceurs sont des adaptations de séquenceurs traditionnels et n'intègrent pas le geste instrumental ni de technique d'écriture / composition.

Seuls quelques séquenceurs (TimelinerSA [1], Open Timeline [3], la plateforme VIRAGE [4], AlgoScore [5]) introduisent une gestion graphique du temps avec une seule timeline en une dimension.

[1] Open Sound Control : protocole UDP pour le contrôle temps réel
[2] Association IanniX, 6/8 rue Notre-Dame de Nazareth – 75003 Paris — Statuts : http://iannix.org/ressources/statuts.pdf
[3] Unité Polyagogique Informatique du CEMAMu

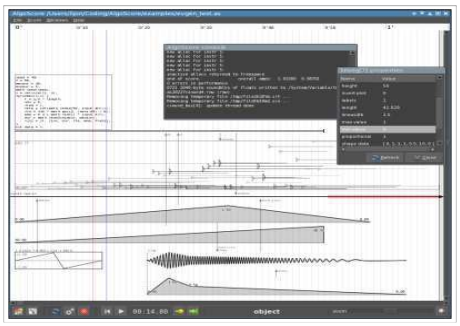

Figure 4. Capture d'écran d'AlgoScore

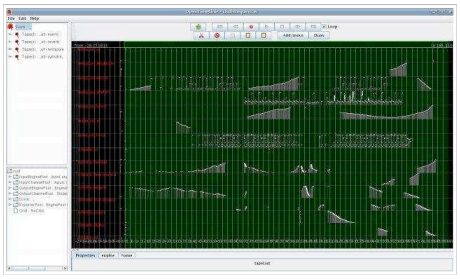

Figure 5. Capture d'écran d'OpenTimeline

1.3. Principe de IanniX

Sur les bases de l'UPIC précédemment décrites et dans une démarche innovante de séquencement Open Sound Control, IanniX a été imaginé comme un séquenceur graphique temps réel, permettant de construire et de composer des partitions graphiques. Une forte volonté d'abstraction a été imposée afin que IanniX ne soit pas dédié uniquement à l'informatique musicale, mais aussi au traitement du signal vidéo, au show-control pour les installations, et à la gestion de différentes interfaces hardware…

Ainsi IanniX s'interface naturellement avec l'environnement temps réel habituel du compositeur / créateur (PureData, CSound, Processing, SuperCollider,).

2. LES OBJETS FONDAMENTAUX

Pour composer une partition graphique temps réel dans IanniX, une palette restreinte de trois objets fondamentaux et distincts a été imaginée :

- les *triggers* déclenchent des événements ponctuels ;
- les *courbes* sont des suites de points dans l'espace tridimensionnel ;
- les *curseurs* évoluent sur des courbes et progressent en fonction du temps.

2.1. Événements : triggers

Un *trigger* T se définit littéralement par un point P dans l'espace 3D de coordonnées (x, y, z) :

$$T = P(x, y, z) \qquad (1)$$

La faculté d'un *trigger* est d'émettre un message lorsqu'il est déclenché, au passage d'un curseur.

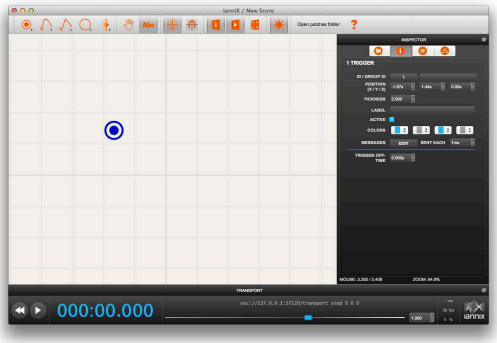

Figure 6. Un trigger dans IanniX

2.2. Trajectoires : les courbes

Une courbe S_n est définie en fonction du temps par une succession de n sections de courbes S_k associées à leur longueur L_k :

$$S_n(t) = \begin{cases} S_0\left(\frac{t}{L_0}\right), & t \in [0, L_0[\\ S_1\left(\frac{t-L_0}{L_1-L_0}\right), & t \in [L_0, L_1[\\ S_2\left(\frac{t-L_1}{L_2-L_1}\right), & t \in [L_1, L_2[\\ \vdots & \\ S_n\left(\frac{t-L_{n-1}}{L_n-L_{n-1}}\right), & t \in [L_{n-1}, L_n] \end{cases} \qquad (2)$$

Chaque section de courbe S_k peut prendre 3 formes : un segment de droite, une ellipse ou une courbe de Bézier cubique.

2.2.1. Segment de droites

Une section de courbe en segment de droite est définie en fonction du temps par deux points $P_{k,0}$ et $P_{k,1}$ telle que :

$$S_k(t) = P_{k,0} \cdot (1 - t) + P_{k,1} \cdot t, \; t \in [0, 1] \qquad (3)$$

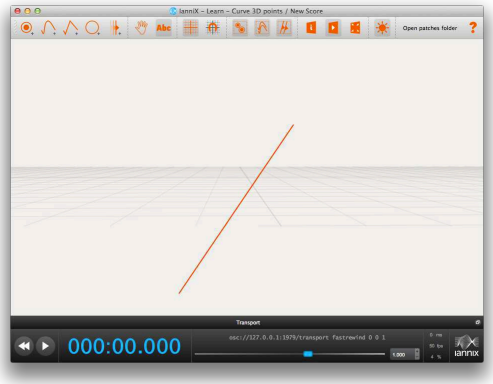

Figure 7. Un segment de droite dans IanniX

2.2.2. *Ellipses*

Une section de courbe en ellipse est définie en fonction du temps par deux rayons r_1 et r_2 et une coordonnée z fixe. Le périmètre de l'ellipse est approximé par :

$$L_k = \pi\sqrt{2({r_1}^2 + {r_2}^2)} \quad (4)$$

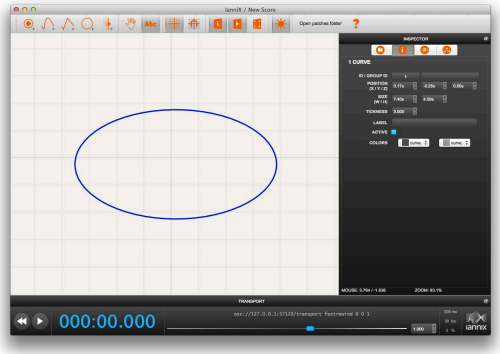

Figure 8. Une ellipse dans IanniX

2.2.3. *Courbes de Bézier*

Une section de courbe en courbe de Bézier cubique est définie en fonction du temps par un point de départ $P_{k,0}$ et un point d'arrivée $P_{k,3}$ ainsi que deux points de contrôle $P_{k,1}$ et $P_{k,2}$:

$$S_k(t) = P_{k,0} \cdot (1-t)^3 + 3 \cdot P_{k,1} \cdot t \cdot (1-t)^2 + 3 \cdot P_{k,2} \cdot t^2 \cdot (1-t) + P_{k,3} \cdot t^3,\ t \in [0,1] \quad (5)$$

La longueur L_k est *approximée* en subdivisant la courbe en segments linéaires.

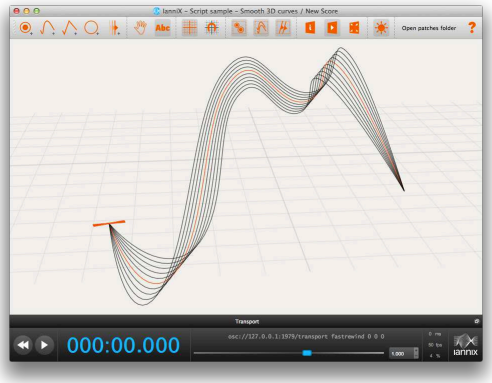

Figure 9. Courbes de Bézier 3D dans IanniX

2.2.4. *Formats vectoriels*

Avec cette palette de 3 sections de courbes différentes (ligne, cercle, Bézier du 3ème ordre), toutes les courbes vectorielles répertoriées dans le format SVG[4] sont possibles.

[4] Scalable Vector Graphics : langage balisé décrivant des graphismes vectoriels bidimensionnels

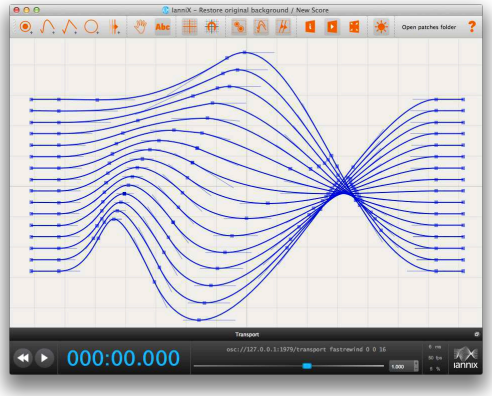

Figure 10. Logo IanniX importé au format vectoriel

2.3. Valeurs : les curseurs

Un ou plusieurs curseurs C peuvent progresser sur une courbe S et exploitent le paramètre t de la courbe. Les coordonnées 3D d'un curseur sont définies par :

$$C = S\big(Z(t)\big),\ t \in [0, L_n] \quad (6)$$

Les propriétés des curseurs dans IanniX :

- les curseurs peuvent envoyer des messages indiquant leur position dans l'espace ou leur progression ;
- les curseurs peuvent déclencher des *triggers* sur leur passage ;
- les curseurs peuvent envoyer des messages lorsqu'ils créent un point d'intersection avec une autre courbe.

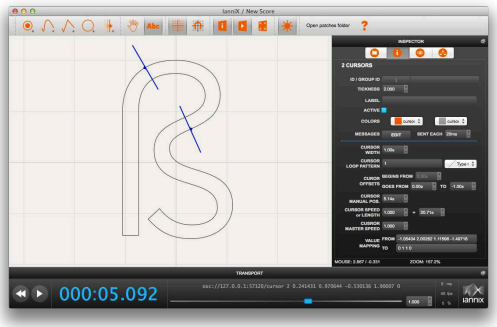

Figure 11. Deux curseurs sur une courbe

IanniX implémente un mécanisme dit de *polytemporalité*, qui permet à chaque curseur de gérer indépendamment sa progression, donc sa relation espace-temps. Celle-ci est réalisée en appliquant une fonction de transfert Z propre à chaque curseur sur le temps, permettant par exemple :

- de progresser normalement sur une courbe si $Z(t) = t$ ou à rebours avec $Z(t) = L_n - t$;
- de doubler la vitesse du curseur sur une courbe, par exemple si $Z(t) = 2 \cdot t$;
- de jouer en boucle une courbe, grâce au reste de la division euclidienne dans les réels ;
- de créer des accélérations et décélérations.

2.4. Temps : le scheduler

IanniX étant un outil informatique, le temps ne peut être vu de manière continue ; il est évidemment échantillonné. La fréquence d'échantillonnage doit être ajustable par l'utilisateur selon ses besoins, mais une valeur comprise entre 200 Hz (*T=5 ms*) et 1 kHz (*T=1 ms*) s'avère suffisante pour couvrir la majorité des usages. L'échantillonnage du temps introduit cependant une difficulté quant au déclenchement des *triggers* :

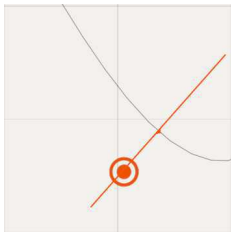

Figure 12. Situation idéale où le curseur "percute" le trigger

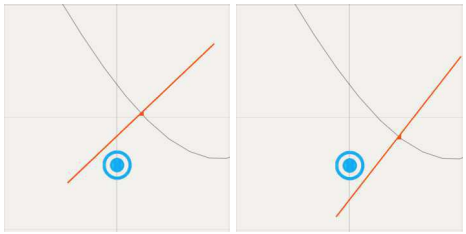

Figure 13. Situation où l'échantillonnage n'est pas suffisant pour déclencher le trigger

IanniX implémente l'interpolation suivante : un curseur déclenche les *triggers* situés entre la position au coup *n* et la position au coup *n-1*.

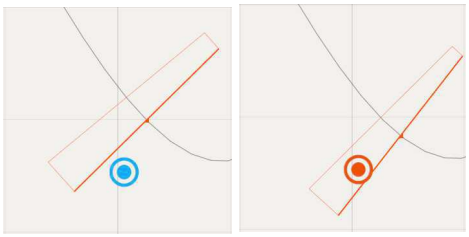

Figure 14. Interpolation des curseurs

3. INTERFACES & MESSAGES

3.1. Interfaçages

3.1.1. Interfaces réseau

La volonté de IanniX est de s'intégrer dans le maximum de contextes applicatifs. Ses interfaces de communications bidirectionnelles sont :

- l'Open Sound Control (OSC) et l'UDP brut, pour les applications temps réel ;
- le MIDI, pour les contextes de contrôle simples ou pour l'interfaçage sur des DAW[5] ;
- le RS232 , pour le contrôle de hardware ;
- l'HTTP, pour le monitoring sur page Web.

[5] Digital Audio Workstation : station dédiée à l'audio numérique

3.1.2. Interface virtuelle ou directe

Une boucle locale a également été développée, afin que IanniX puisse s'envoyer directement des messages, sans consommation de bande passante réseau ni encapsulation dans un protocole.

Nous reviendrons sur cette notion de *récursivité* à la sous-section 5.3 Écriture récursive.

3.1.3. Interface script

Des commandes JavaScript permettent également de générer des messages sous forme de code / script.

Nous détaillerons cette approche dans le chapitre *3.3.* Partitions génératives.

3.1.4. Interface graphique utilisateur

Enfin, la grande majorité des actions sont réalisables depuis une interface graphique utilisateur soignée et organisée. L'interface graphique a été conçue en quatre zones d'interactions :

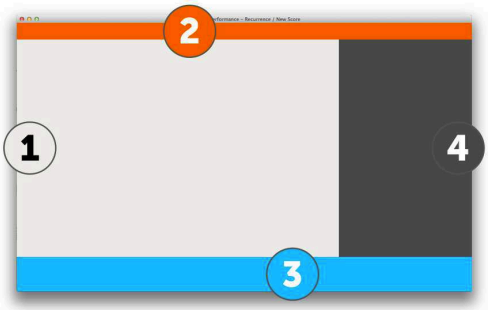

Figure 15. Zoning de l'interface graphique

1. VISUALISATION : représentation graphique de la partition ;
2. CREATION : zone de manipulation de l'espace et outils de création d'objets ;
3. DEROULEMENT : macro-réglages et transport ;
4. INSPECTION : recherche d'objets, propriétés, logs.

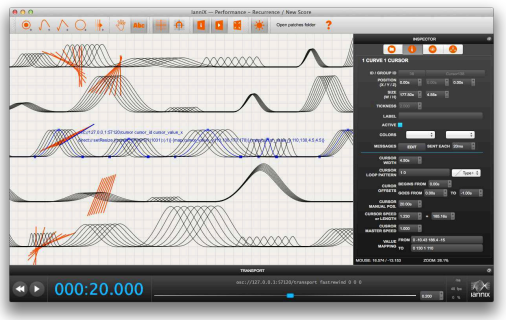

Figure 16. Interface graphique de IanniX

3.2. Gestion par messages

3.2.1. Messages sortants

Les objets de IanniX émettent des messages (les *triggers* en émettent lorsqu'ils sont déclenchés, les *curseurs* en émettent périodiquement lorsqu'ils progressent sur une courbe). Les interfaces et protocoles étant variés (OSC, HTTP, MIDI, RS232[6]…), une homogénéisation du formatage des messages a été imaginée dans IanniX.

Un message IanniX respecte systématiquement un format d'URL :

```
protocole://destination/adresse arguments
```

- **`protocole`** définit l'interface à utiliser (**`osc`**, **`udp`**, **`midi`**, **`serial`**, **`http`**, **`direct`**) ;
- **`destination`** décrit en Open Sound Control, UDP et HTTP, l'IP et le port ; en MIDI ou port série, le nom du port / dispositif de sortie ;
- **`adresse`** représente l'adressage au sein du protocole choisi : en Open Sound Control, l'adresse OSC ; en MIDI le contrôle à moduler : control change (**cc**), note on/off (**note**), program change (**`pgm`**), pitch bend (**`bend`**) ;

Les **`arguments`** sont des variables que IanniX adapte en temps réel en fonction de la partition. Il existe une cinquantaine d'arguments possibles ; quelques exemples[7] :

- **`cursor_zPos`** donne l'élévation *z* d'un curseur ;
- **`trigger_xPos`** retourne la position en *x* d'un *trigger* déclenché ;
- **`collision_yPos`** fournit la coordonnée *y* du point d'intersection entre un curseur et une courbe ;
- **`cursor_angle`** représente l'angle du curseur sur la courbe ;
- **`trigger_distance`** calcule la distance entre le *trigger* déclenché et le point sur la courbe du curseur.

Concrètement, un message IanniX typique serait :

```
osc://192.168.0.10:57120/synth cursor_id
cursor_xPos cursor_yPos
```

Cet exemple envoie sur l'IP **`192.168.0.10`** et le port **`57120`** un message Open Sound Control à l'adresse **`/synth`** avec 3 arguments :

- l'ID du curseur qui envoie le message (*entier*) ;
- la position *x* du curseur (*flottant*) ;
- la position *y* du curseur (*flottant*).

[6] Protocole série couramment utilisé pour dialoguer avec des interfaces matérielles

[7] Documentation complète : http://iannix.org/fr/documentation

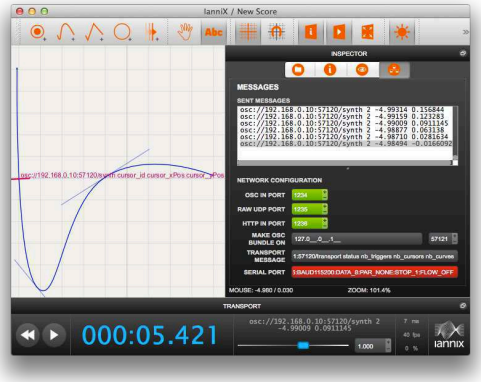

Figure 17. Log des envois dans IanniX

3.2.2. Commandes entrantes

Des commandes entrantes sont possibles, afin de contrôler IanniX depuis une autre application.

Toute la composition de partition repose sur des messages entrants, qu'ils proviennent de l'interface graphique *(les messages ne sont pas visibles par l'utilisateur)*, des interfaces réseau, de l'interface virtuelle ou de scripts.

Par exemple, le message Open Sound Control suivant **`/iannix/add trigger 5`** va ajouter un *trigger* dont l'ID vaut 5 dans la partition ; ou encore le message **`play 0.4`** va jouer la partition avec un facteur de vitesse de 0.4.

Il existe plus d'une soixantaine de commandes possibles, répertoriées dans une API[8] documentée.

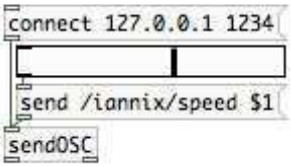

Figure 18. Contrôle de la vitesse de lecture en temps réel depuis PureData

3.3. Partitions génératives

IanniX peut interpréter des scripts JavaScript grâce à son moteur basé sur ECMAScript (norme ECMA-262[9]).

3.3.1. Langage de programmation

Les contraintes de programmation sont assez simples : un script IanniX doit obligatoirement contenir une fonction **`onCreate()`** qui est appelée au moment de la génération de la partition (à l'ouverture du fichier). Le code est ensuite interprété comme un script JavaScript habituel.

[8] Application Programming Interface : interface fournie pour piloter un programme informatique

[9] http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf

Pour envoyer un message à IanniX, une commande spéciale **`run(message)`** a été mise en place. Par exemple, pour ajouter une courbe d'ID = 9, on saisira :

```
run("add curve 9");
```

IanniX apporte cependant quelques enrichissements, notamment pour simplifier le *mapping* de valeurs d'un intervalle à un autre, ou encore quelques fonctions mathématiques usuelles (nombre aléatoire entre deux bornes…).

3.3.2. *Exemples de mise en œuvre*

En exploitant les boucles ou la récursivité, il est possible de générer des partitions graphiques complexes ou mathématiques. Exemple d'un script :

```
function onCreate() {
 //Boucle de 0 à 24
 for(var index=0 ; index < 25 ; index++) {
  //Crée une courbe d'ID 1000, 1001...
  run("add curve " + (1000 + index));
  //Premier point de la courbe
  run("setPointAt current 0 "
                          + (-index) + " 0");
  //Deuxième point de la courbe
  run("setPointAt current 1 0 "
                        + (24-index) + " 0");

  //Crée un curseur d'ID 0, 1, 2...
  run("add cursor " + index);
  //Lie le curseur à la courbe précédente
  run("setCurve current lastCurve");
  //Fixe la durée de parcours à 5 sec
  run("setSpeed current auto 5");
 }
}
```

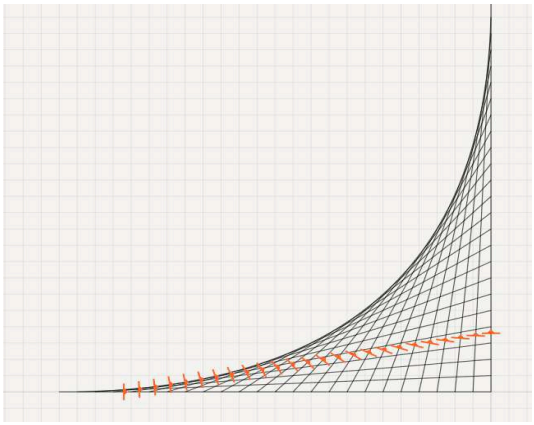

Figure 19. Résultat du script ci-dessus

Quelques autres exemples :

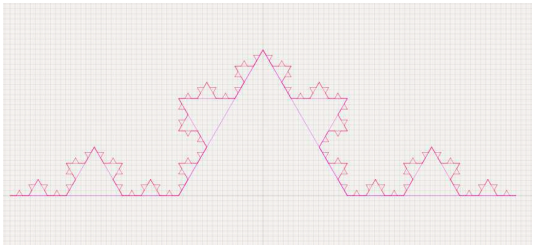

Figure 20. Génération d'une partition à base de fractales

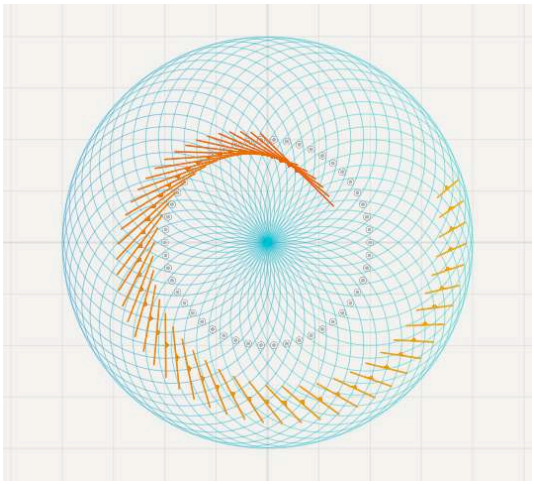

Figure 21. Script générant des cercles

3.3.3. *In-message programming*

Il est aussi possible d'insérer directement du code JavaScript dans les messages émis par les objets IanniX. Pour réaliser cette opération, on insère le code entre accolades.

```
osc://192.168.0.10:57120/synth
        cursor_id
{cursor_xPos + cursor_yPos}
      {random(20,100)}
```

Cet exemple envoie sur l'IP **`192.168.0.10`** et le port **`57120`** un message Open Sound Control à l'adresse **`/synth`** avec 3 arguments :

- l'ID du curseur qui envoie le message ;
- la somme de la coordonnée *x* et *y* du curseur ;
- un nombre aléatoire entre 20 et 100.

4. LE LOGICIEL

4.1. Architecture logicielle

Depuis 2004, IanniX a été développé avec le *framework* de développement Qt. Ce choix s'avère toujours aussi pertinent pour les versions futures pour les raisons suivantes :

- *framework* open source, compatible avec des modèles de licences types GPL[10] ;
- programmation en C++, performant et stable, avec une précision du temps de 100 µs ;
- *cross-platform*, qui peut s'exécuter sur les trois principaux systèmes d'exploitation Linux, Mac OS et Windows ;
- intégration native d'OpenGL, pour le rendu 2D et 3D de manière fluide et interactive ;
- interfaces graphiques riches avec une expérience utilisateur intuitive, élégante et efficiente.

Le logiciel a été divisé en sept grands modules indépendants, fournissant chacun un certain nombre de services.

[10] General Public License : licence pour les logiciels libres

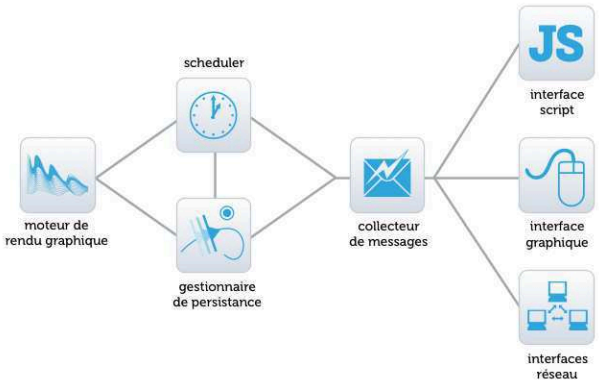

Figure 22. Architecture logicielle

4.2. Performances & optimisations

D'une manière générale, IanniX a été profilé pour deux optimisations :

- le calcul des positions des curseurs sur les courbes, le déclenchement des *triggers* et l'émission de message qui sont réalisés en CPU[11] ;
- l'affichage et l'animation des objets qui est effectuée en GPU[12].

4.2.1. Persistance des données

Pour le stockage des objets et des partitions, chaque référence d'objet est stockée en double :

- une première copie est stockée dans une table de hachage qui permet très rapidement :
  - de parcourir l'ensemble des objets grâce à des itérateurs (pour l'affichage),
  - de trouver un objet par son ID (pour les messages entrants) avec une complexité qui tend vers $O(1)$ [2] ;
- une deuxième copie est stockée dans un arbre qui différentie les types d'objets, leur activité, leur regroupement en groupes et leur position dans l'espace : cette optimisation permet de rechercher très rapidement un ou plusieurs objets en connaissant au préalable ses caractéristiques (pour le calcul).

4.2.2. Représentation graphique

L'intégralité de la représentation graphique des partitions bénéficie de l'accélération graphique OpenGL.

Les courbes sont optimisées pour :

- le calcul de la position d'un curseur sur une courbe est réalisée et optimisés en CPU ;
- le calcul et dessin des courbes se fait intégralement en GPU.

Le *framerate* est fixé par défaut à 40 images par seconde, mais il est ajustable par l'utilisateur selon ses besoins ou ressources disponibles.

4.2.3. Mise en cache de la construction de message

La génération des messages est mise en cache dans IanniX afin d'émettre 1000 messages par seconde avec le minimum de ressources CPU. Au-delà de ce seuil, les optimisations sont négligeables par rapport au coût d'envoi d'un datagramme réseau par le système d'exploitation.

4.3. Modèle de diffusion & de contribution

IanniX est diffusé sur son site Internet (www.iannix.org) en licence GPL 3. Les binaires des versions Linux, Mac OS et Windows sont proposés dans leurs doubles versions 32 bits et 64 bits. Un *repository* Git (www.github.com/iannix/IanniX) permet également aux potentiels contributeurs de participer facilement au développement du logiciel ; la chaîne de compilation ne nécessitant que Qt SDK. Enfin, les utilisateurs peuvent échanger des patchs, astuces, bugs, solutions… dans un forum de discussion.

Outre une refonte logicielle intégrale mi-2011, IanniX a renforcé sa présence sur le web (unique lieu de diffusion) grâce à du *teasing vidéo* et à la création de supports de communication actuels *(Facebook, mailing-list, Twitter).*

Sur ces neuf derniers mois, IanniX compte 20000 visiteurs uniques sur son site web (avec un taux de rebond très faible de 1.7%), 5200 utilisateurs et une quarantaine de lancements quotidiens de l'application. Deux tiers des utilisateurs sont sur Mac, 30% sur Windows et le reste sur Linux. Seuls 15% des utilisateurs sont français.

La communauté d'utilisateurs a également développé des exemples et tutoriaux pour intégrer IanniX dans divers logiciels (Max4Live, SuperCollider, CSound…).

Enfin, l'association « IanniX » encourage la promotion du logiciel dans tous ses aspects : recherche, création, enseignement, formation, publications, études, développements…

5. NOUVELLES POSSIBILITES D'ECRITURE

5.1. Approche macroscopique et microscopique du temps

Contrairement à l'UPIC qui considérait une seule ligne temporelle macroscopique, IanniX intègre une gestion du temps à l'échelle microscopique (une infinité de lignes de temps possibles). Cette double approche permet d'envisager des compositions à plusieurs échelles de lecture et d'interprétation.

[11] Central Processing Unit : processeur logique d'une machine
[12] Graphics Processing Unit : processeur graphique d'une machine

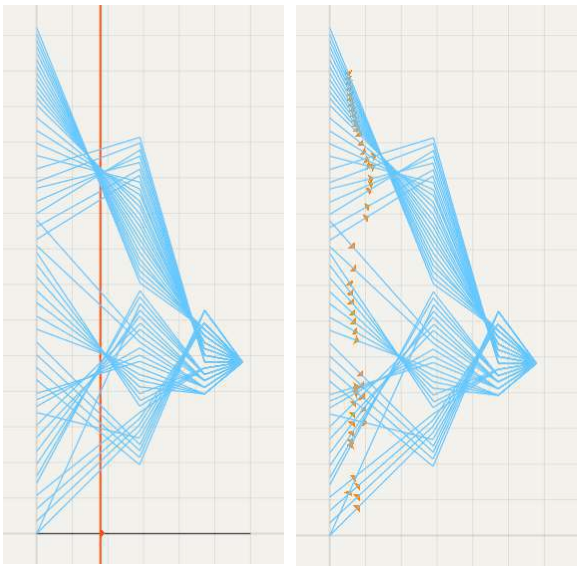

Figure 23. Partition numérisée de Metastasis de Iannis Xenakis jouée sous l'approche macroscopique *(à gauche)* et microscopique *(à droite)*

5.2. Creative & live coding

IanniX est certes un logiciel autonome — et non une librairie — mais son langage de script intégré lui permet de s'inscrire dans la liste des outils de *creative coding* ; l'aspect temps réel lui confère aussi l'appellation de *live coding*. L'intérêt majeur de l'écriture par scripts est d'intégrer l'aléatoire, les boucles, les équations mathématiques et la logique.

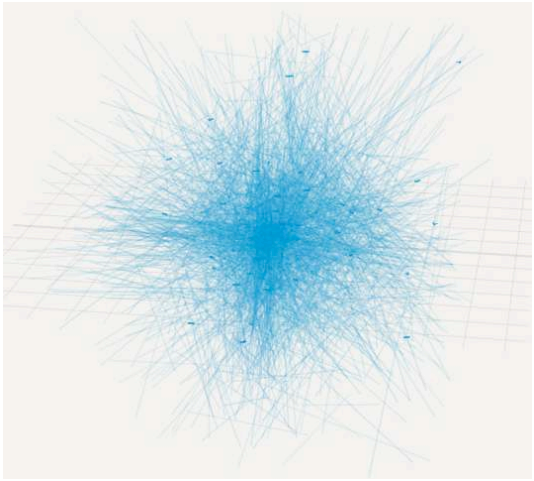

Figure 24. Script générant des lignes aléatoirement dans l'espace, sous contraintes

5.3. Écriture récursive

La fonction principale de IanniX est de générer des messages qui sont envoyés à un environnement dédié. Mais qu'advient-il si IanniX s'envoie des messages à lui-même modifiant ainsi la structure géométrique ou temporelle des partitions ?

Cette nouvelle forme d'écriture, appelée *écriture récursive*, est radicalement innovante et sa complexité la rend encore difficile à maitriser : des phénomènes de stabilisation, d'*exponentialisation*, d'anéantissements ou de chaos se produisent.

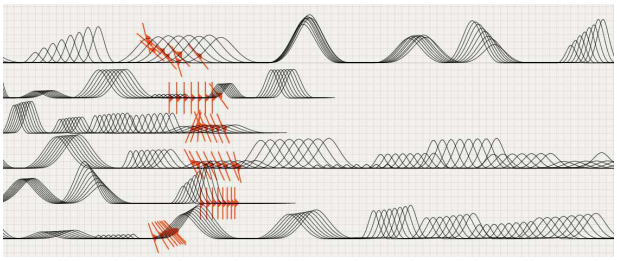

Figure 25. Partition de la performance « Récurrences » de Thierry Coduys au centre DATABAZ d'Angoulême *(partition distribuée librement dans l'application IanniX)*

5.4. Spatialisation 3D

Couplé à un moteur audio de spatialisation (Ambisonic 3D…), la palette d'outils de IanniX permet l'écriture de trajectoires en 3D :

- le *trigger* déclenche un son ;
- la courbe définit la trajectoire du son dans l'espace ;
- le curseur joue sur le temps (accélérations, répétitions, déplacements, etc.)

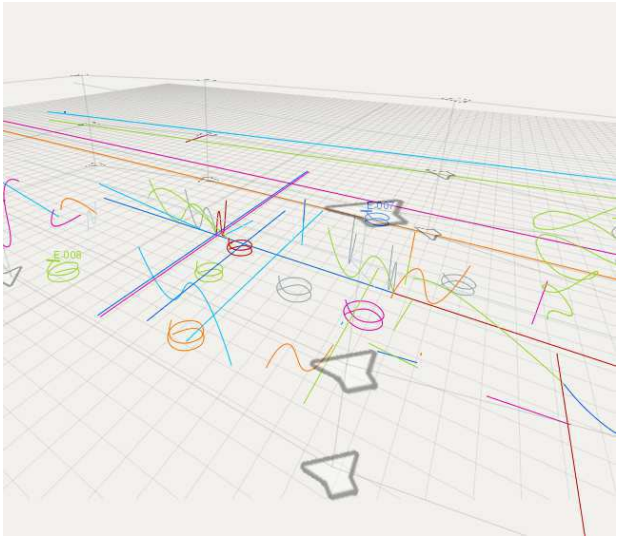

Figure 26. Trajectoires de spatialisation pour l'exposition universelle de Yeosu 2012

5.5. Interfaces temps réel

La versatilité des entrées / sorties de IanniX permettent d'utiliser des capteurs ou actionneurs récents de toutes natures.

L'intérêt d'un tel dispositif de captation réside dans la manipulation, modification ou la génération de courbes en temps réel tel un geste instrumental interprétant une partition.

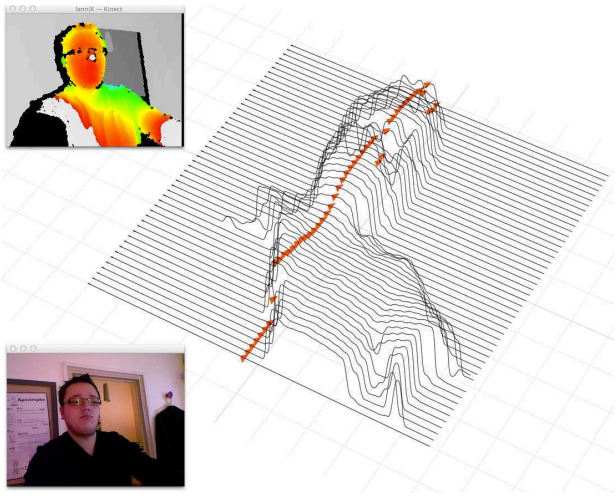

Figure 27. Extrusion des courbes à l'aide d'un capteur Kinect à la manière d'un scanner 3D

6. CONCLUSION & PERSPECTIVES

IanniX offre une approche graphique et 3D du séquencement temps réel à l'aide de trois objets fondamentaux (les triggers, les courbes et les curseurs). Les interfaces implémentées, quelles soient physiques (OSC, MIDI, RS232…), graphiques (interface utilisateur), logicielles (JavaScript) ou virtuelles (récursivité), contrôlent, modifient ou génèrent des partitions de séquencement grâce notamment à un système d'abstraction par messages uniques.

Les travaux menés pour IanniX 0.8 depuis mai 2011 offrent à l'outil des performances accrues et une meilleure stabilité. Une communauté d'utilisateurs s'est rapidement créée et un modèle open source contributif a été mis en route.

De nouvelles formes d'écritures sont apparues au fil du développement et des projets réalisés[13] avec IanniX, notamment l'écriture récursive, l'intégration du geste instrumental et le *live coding*.

Les prochains développements de IanniX porteront notamment sur :

- les courbures de l'espace-temps, afin de conférer à IanniX des propriétés physiques relativistes ;
- l'intégration de propriétés physiques aux objets (masses, ressorts, modèles physiques…) ;
- un passage progressif d'une logique booléenne vers une logique floue ;
- la possibilité de visualiser une partition sous différentes représentations symboliques (feuilles de style).

Au-delà des aspects logiciels, la dissémination des travaux autour de IanniX est un objectif fort, aussi bien dans le cadre scientifique (publications), universitaires (workshop et utilisation dans des formations) ou intellectuel (accompagner les créations).

[13] Exemples de projets : http://iannix.org/fr/projects.php

7. REFERENCES

[1] TimelinerSA, http://vvvv.org/contribution/timelinersa

[2] Algorithmic Complexity of Container Classes, *Qt Documentation*, http ://qt-project.org/doc/qt-4.8/containers.html

[3] Henry, D. "PTL, a new sequencer dedicated to graphical scores", International Computer Music Conference Proceedings, Miami, USA, 2004

[4] Virage, http://www.virage-platform.org/

[5] AlgoScore, http://kymatica.com/Software/AlgoScore

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

[7] Coduys, T. et Ferry G., "IanniX, aesthetical / symbolic visualisations for hypermedia composition", Sound and Music Computing, 2004